DEEP LEARNING FOR MULTI-ATTRIBUTE VEHICLE RECOGNITION IN VEHICLE ACCESS CONTROL SYSTEM

WONG YUAN ZHEN

UNIVERSITI TUNKU ABDUL RAHMAN

DEEP LEARNING FOR MULTI-ATTRIBUTE VEHICLE RECOGNITION IN VEHICLE ACCESS CONTROL SYSTEM

WONG YUAN ZHEN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Software Engineering (Honours)

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2025

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Name WONG YUAN ZHEN

ID No. : 2200191

Date : 19/09/2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. Lee Ming Jie as my research supervisor for his invaluable advice, guidance and enormous patience throughout the development of the research.

I would also like to extend my appreciation to the lecturers and staff members of the Lee Kong Chian Faculty of Engineering & Science, Department of Computing in Universiti Tunku Abdul Rahman (UTAR), for their assistance and for providing a conducive learning and research environment during my study period.

Lastly, I am deeply grateful and thank to my family and friends for their unwavering support and encouragement which gave me the strength and motivation to complete this whole project successfully.

ABSTRACT

Vehicle recognition systems are becoming increasingly essential for intelligent transportation, traffic surveillance, and security applications. For the purpose to identify vehicle attributes in real-time, this research provides a hybrid vehicle recognition system that is implemented as a web and mobile application. It combines deep learning and computer vision techniques. To extract license plates, colors, makes, models, and production years of vehicles, the system mainly uses EasyOCR and YOLOv8 with multi-attribute detection. A refined GPT-4o visual-language model (VLM) acts as a fallback, improving recognition reliability in edge instances when YOLOv8 and EasyOCR would not yield reliable results. The approach involves capturing pictures of vehicles from cameras or user uploads, processing them using the pipeline for detection and OCR, then using the GPT-4o VLM to verify the outcomes. The system's modular design provides seamless connection with web and mobile platforms, enabling real-time performance and scalability. It is expected to work reliably across a variety of lighting situations, angles, and occlusions. The study shows a potential approach for enhancing the recognition of vehicle attributes. Future research will involve adding more unusual vehicle kinds to the dataset, refining the model inference for edge devices, and integrating predictive analytics for anomaly detection and vehicle tracking.

Keywords: vehicle recognition; YOLOv8; EasyOCR; GPT-4o; deep learning; computer vision; license plate recognition

Subject Area: T58.5-58.64 Information technology

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation and Durability
AFPN	Asymptotic Feature Pyramid Network
AI	Artificial Intelligence
ALPR	Automated License Plate Recognition
ANPR	Automatic Number Plate Recognition
API	Application Programming Interface
ASFF	Adaptive Spatial Feature Fusion
AWS	Amazon Web Services
BiFPN	Bi-directional Feature Pyramid Network
BLIP	Bootstrapping Language-Image Pre-training
BLOB	Binary Large Object
BYTEA	Byte Array
CCTV	Closed-Circuit Television
CDN	Content Delivery Network
CI/ CD	Continuous Integration/ Continuous Deployment
CLI	Command Line Interface
CLIP	Contrastive Language-Image Pre-training
CNNs	Convolutional Neural Networks
CRUD	Create-Read-Update-Delete
CSR	Client-side Rendering
CSRF	Cross Site Request Forgery
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CTEs	Common Table Expressions
DB	Database
DBMS	Database Management System
DFD	Data Flow Diagram
EA	Enterprise Architect
EALPR	Efficient Automatic License Plate Recognition
ERD	Entity Relationship Diagram
Fast-SCNN	Fast Segmentation Convolutional Neural Network
FLAN	Fine-tuned LAnguage Net

FPN	Feature Pyramid Networks
GPT	Generative Pre-trained Transformers
GPU	Graphics Processing Unit
HFIE	High-Frequency Information Extraction
HOG	Histogram of Oriented Gradients
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
IoU	Intersection over Union
ITS	Intelligent Transportation Systems
JSON	JavaScript Object Notation
JSONB	JSON Binary
JSONL	JSON Lines
JSX	JavaScript XML
LeSS	Large-Scale Scrum
LIDAR	Light Detection and Ranging
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LOB	Large Object
LPD	License Plate Detection
LPR	License Plate Recognition
LVA	Language-Vision Alignment
MAE	Masked Autoencoder
mAP50	Mean Average Precision
MAS	Multi-Agent System
MIoU	mean Intersection over Union
ML	Machine Learning
MVC	Model View Controller
NLP	Natural Language Processing
OCR	Optical Character Recognition
ORM	Object Relational Mapping
PEFT	Parameter-Efficient Fine-Tuning
PHP	Hypertext Preprocessor
PLC	Project Life Cycle

PWA	Progressive Web App
QA	Quality Assurance
QR	Quick Response
RFC	Residual Feature Connection
RFID	Radio-frequency Identification
RLHF	Reinforcement Learning from Human Feedback
SAFe	Scaled Agile Framework
SAM	Segment Anything Model
SDG	Sustainable Development Goal
SDLC	Software Development Life Cycle
SEO	Search Engine Optimization
SIFT	Scale-Invariant Feature Transform
SMNN-MSFF	Smooth Modulation Neural Network with Multi-Scale Feature Fusion
SQL	Structured Query Language
SSG	Static Site Generation
SSR	Server-side Rendering
STNs	Spatial Transformer Networks
SURF	Speeded-Up Robust Features
SUS	System Usability Scale
UAT	User Acceptance Test
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
URS	User Requirements Specifications
UX	User Experience
VCR	Vehicle Color Recognition
ViT	Vision Transformer
VLM	Visual Language Model
VMMR	Vehicle Make and Model Recognition
WBS	Work Breakdown Structure
XSS	Cross-site Scripting
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Systems for controlling vehicle access are essential for improving security, controlling traffic, and safeguarding restricted areas including residential complexes, industrial zones, and private parking lots. Conventional and current primarily utilized access control techniques frequently depend on technologies like RFID cards or tags and license plate recognition (LPR) or on security personnel manually verifying entries (Cayetano, 2024). However, human mistake can cause security breaches with manual checks and unfavorable weather can cause RFID-based devices to malfunction and reducing their dependability. Furthermore, license plate identification by itself has limits because access constraints can be bypassed by using forged or duplicate plates. These difficulties show that more sophisticated, reliable, and secure car recognition systems are required.

Recent developments in artificial intelligence, specifically in multimodal learning and Large Language Models (LLMs), have encouraging possibilities for enhancing car recognition systems. Multimodal techniques can greatly improve recognition accuracy and system resilience by integrating many data types including textual and visual. Building more dependable and flexible access control systems is made feasible by utilizing LLMs' capacity to recognize complicated relationships across many data modalities.

The development of a multimodal large language model-based vehicle recognition system suitable for access control applications is the idea behind this project. By combining several vehicle characteristics, the system aims to go beyond conventional plate-based and single-modality identification techniques, in order to lowering the possibility of unwanted entry, improve security and enhancing system resilience in a range of operational scenarios. This chapter will focus on importance of study, problem statements, project objectives, project scope and scope out of scope, proposed solution and proposed approach to set the groundwork for a thorough examination of current technologies and approaches in next chapter.

1.2 Importance of the Study

In the past, vehicle access control systems were created to meet the requirement to protect private or sensitive spaces from unwanted access. It became more crucial to control and regulate the flow of cars into restricted areas like residential neighborhoods, government buildings, business buildings, and toll highways as cities grew and the number of cars on the road increased. These systems' main objective was to assure that only authorized personnel may enter, safeguarding infrastructure, people and assets from possible dangers including theft, vandalism and other violations of security. By eliminating down the amount of time required for entry processing and minimizing the need for manual supervision, these systems aimed to increase operational efficiency in addition to security.

Conventional and current mainly used access control techniques were frequently used to achieve these objectives. These methods include license plate recognition (LPR) technology, RFID tag or card-based systems and manual verification by security personnel. Although these approaches have shown some degree of success but they have significant flaws that leave systems vulnerable to operational inefficiencies and safety risks.

Human mistake can happen during manual verification by security staff especially in crowded environments or when they are tired and distracted. Guards could unintentionally misidentify cars, let illegal entry or overlook credentials that have been faked. Furthermore, manual checking creates bottlenecks at high-volume entry points, which lowers customer satisfaction and causes traffic congestion.

Automation is provided via RFID-based systems which scan tags or cards attached to vehicles. However they are sensitive to environmental factors. Unfavorable weather conditions including intense rain, fog or extreme temperatures can disrupt signal transmission and result in delays or failures in authentication. Furthermore, the intended security of RFID cards can be compromised by loss, theft, duplication or intentional misuse by unauthorized users.

Vehicle identification is automated using License Plate Recognition (LPR) systems, yet these systems mostly depend on the accuracy and visibility of license plates. Accuracy of recognition can be affected by unclean or broken plates, poor lighting, and unusual plate designs. More seriously, there have been breaches of security when unauthorized entrance was obtained by tricking LPR systems with fake or duplicate license plates. A significant instance in Malaysia when fake license plates were used to bypass valet parking payment in hotel was mentioned (Lee, 2024). This has highlighted a significant weakness in depending just on plate-based verification.

These restrictions have important consequences. Within controlled premises, residents, staff and property are at risk of security breaches due to unauthorized vehicle entrance. Theft, vandalism, or unpaid tolls can result in financial losses. Operationally, system unreliability undermines public confidence and makes traffic control more difficult particularly in urban regions integrating smart cities.

With these difficulties, a more sophisticated, reliable, and flexible access control system is absolutely essential. A revolutionary development is provided by the suggested creation of a vehicle recognition system based on a Multimodal Large Language Model (LLM). By having this system, t he system overcomes the single-point failure risks associated with conventional approaches by combining several aspects such as textual data, vehicle visual attributes, and additional sensor inputs. Utilizing LLMs' advanced reasoning and pattern recognition skills improves the system's capacity to identify irregularities, prevent attempts at fraud, and function dependably in a range of environmental circumstances.

Moreover, the study helps to accomplish more general social objectives such those stated in Sustainable Development Goal (SDG) 11 of the United Nations which is to "make cities and human settlements inclusive, safe, resilient, and sustainable." This research contributes to the development of safer and more secure urban infrastructures by improving access control technology which is consistent with the idea of future smart cities.

Overall, this study is significant because it not only addresses serious shortcomings in current vehicle access control systems but also advances the use of modern artificial intelligence techniques for increased security, operational effectiveness, and public trust.

1.3 Problem Statement

The shortcomings of the existing vehicle access control techniques lead to a number of serious problems that stimulate the creation of a more advanced system.

This section addresses the three main issues that were found during the problem formulation phase. These issues include the vulnerability of traditional access control systems, the susceptibility of ALPR systems to forged license plates, and inability to adapt to environmental and situational challenges.

1.3.1 Vulnerabilities in Traditional Access Control Systems

The authentication procedure in an ideal car access control system would be precise, safe, and effective by reducing human mistake and involvement. Regardless of human or environmental influences, approved vehicles should always be identified and unauthorized ones should always be blocked.

However, there are significant flaws in traditional access control techniques including security guards' human verification and RFID card scanning. Typically during peak hours or while under stress or fatigue, human judgment can be inconsistent. Despite being automated, RFID systems are vulnerable to technological malfunctions brought on by environmental elements like dust or moisture as well as RFID tag cloning and theft. Due to these flaws, traditional systems are unstable and vulnerable to attacks in restricted areas.

A more intelligent, automated system that relies less on single-point RFID authentication and human involvement is required to address these issues. A multimodal system that uses cutting-edge technology to cross-verify vehicle features can greatly increase security by delivering more reliable and precise access control.

1.3.2 Susceptibility of ALPR Systems to Forged License Plates

In an ideal world, each vehicle would be accurately identified by Automatic License Plate Recognition (ALPR) devices using a safe, unchangeable license plate. Fast and completely dependable vehicle verification would stop any illegal access using forged or modified vehicle identities.

In reality, ALPR systems are susceptible to fraud since they mostly depend on the validity of license plates. ALPR cameras and software can readily be tricked by forged, stolen or duplicate license plates that allowing unauthorized vehicles to enter incorrectly. The security of ALPR-only systems is seriously compromised as evidenced by reports of faked plates being used in illegal activities.

An improved recognition system that relies on more than just license plate readings is required to decrease this vulnerability. A multimodal system can identify anomalies and more effectively stop unwanted entries by combining extra vehicle characteristics such as color, make and model with intelligent verification through the use of Large Language Models (LLMs).

1.3.3 Inability to Adapt to Environmental and Situational Challenges

An ideal vehicle access control system would function dependably in every kind of weather including areas with poor lighting, heavy rainfall and unclean car surfaces. It would ensure constant security coverage by adjusting to changing circumstances without compromising identification accuracy or system reliability.

A lot of traditional and current systems rely on just one authentication method including RFID scanning or license plate reading. These techniques frequently fail in difficult environmental circumstances. For instance, inadequate lighting might make it difficult to see license plates and persistent rain can obstruct the detection of RFID signals. These situational flaws affect the efficiency and reliability of access control procedures.

These environmental restrictions can be addressed by a multimodal system that integrates data from many vehicle information sources and is backed by LLMs' strong reasoning and contextual awareness. The system can retain high accuracy even under less-than-ideal circumstances by cross-referencing several vehicle parameters. This can bring it one step closer to its goal of continuous and extremely secure access management.

1.4 Project Objectives

The objectives that this project aims to accomplish include:

i. To investigate the primary limitations of conventional vehicle recognition systems.

ii. To develop a robust multi attribute vehicle recognition system.

iii. To integrate visual language model for adaptive recognition, handling edge cases in vehicle access control.

1.5 Project Scope

The main goal of this project is to develop a multimodal vehicle recognition system that improves on traditional ALPR systems by adding image-based vehicle characteristics like color, model, and type to enhance recognition quality. The system will be developed as an online and mobile application for a gated community using Tailwind CSS, Next.js, and React Native for online. The project scope are:

1. Review related applications, software development methodologies and technologies used in vehicle color, model and manufacturer recognition. Conduct a literature study on current approaches and technologies related to vehicle recognition systems, such as Automatic Number Plate Recognition (ANPR), segmentation techniques, Large Language Models (LLM) and similar applications.
2. The development of a real-time vehicle recognition system that combines image-based attribute verification with license plate recognition.
3. Using contextual and environmental data for assisting adaptive decision-making through the integration of Large Language Models (LLM).
4. Using segmentation techniques to isolate vehicles and recognize objects in collected photos.
5. Designing and developing resident and security personnel interfaces with role-specific features like alerts for suspicious activity, vehicle records, notifications, and visitor passes.
6. Testing the recognition framework with test and simulation-based datasets in a range of illumination and angle scenarios.

7. To determine the ideal configuration for the developed vehicle recognition technique by performing experiments in a controlled environment.

1.5.1 Tools

The development tools being used in this project are:

Table 1. 1 : Summarization of development tools

Category	**Tool**	**Purpose**
Frontend	React Native	Cross-platform mobile app development for vehicle access control UI.
	React Native Web	Extends React Native to support web browsers for a unified codebase.
	Tailwind CSS	Styling and responsive design for the application interface.
Backend	Next.js	Server-side logic, API development, and integration with AI/DB components.
	Vercel	Hosting and deployment platform for the Next.js backend to ensure scalability and CI/CD integration.
Database	PostgreSQL	Secure and scalable storage for vehicle records, user data, and logs.
	Supabase	Managed PostgreSQL service providing real-time subscriptions, authentication, API access, and database scalability.

Category	Tool	Purpose
AI/ML Model	Fine-tuned GPT-4o, Yolo-v8 model, EasyOCR	Multimodal vehicle recognition, JSON output generation.
Version Control	Git & GitHub	Collaborative code management, version tracking, and deployment.

1.5.2 Target User

The target user for this project is listed as below:

i. Condominium residents

ii. Apartment residents

iii. Landed area residents with guard house

1.6 Out of Scope

The out of scope of project include:

i. High hardware specifications are needed for training a new model especially when the training set or LLM parameters are big and the LLM model is learned locally. Hence, training of new LLM model will not be included in this project.

1.7 Proposed Approach

A project approach is the process or methodology used to organize, carry out, and finish a project. There are numerous project approaches and each has certain benefits and properties of its own. The research and development approaches were addressed in this part.

1.7.1 Research Approach

In order to collect quantifiable information and obtain unbiased insights on client needs, expectations and opinions regarding a vehicle access control system. This project uses a quantitative research methodology. In this situation, quantitative research makes sense because it makes it possible to gather organized responses that can be statistically examined to aid in the development and validation of the proposed system.

This strategy is put into effect by developing and distributing a questionnaire using Google Forms which makes it easy and effective to gather data from a variety of responders. The questionnaire's closed-ended questions which include multiple-choice and rating scale formats are designed to gather specific data about users' experiences with current vehicle access systems, the difficulties faced by them when they utilizing current system, the important aspects on system to be focus on, and their interest in cutting-edge technology involvement like multimodal verification with LLM and license plate recognition.

By ensuring that data is gathered in a consistent manner, Google Forms helps to reduce bias and simplify the analytic process by having data visualization like pie chart and bar chart. Remote participation by respondents increases accessibility and response rates. Decisions about system design and feature prioritization are based on the patterns and user preferences found in the information gathered after it has been statistically examined.

By using this quantitative method, the study makes sure that the system development is based on actual user feedback, data-driven, and in line with realistic expectations. By demonstrating the degree of user interest and perceived effectiveness of a more secure and intelligent vehicle access control system, it also aids in validating the project's relevance and practicality.

1.7.2 Development Approach

The Agile development methodology was selected as the development approach for this project.

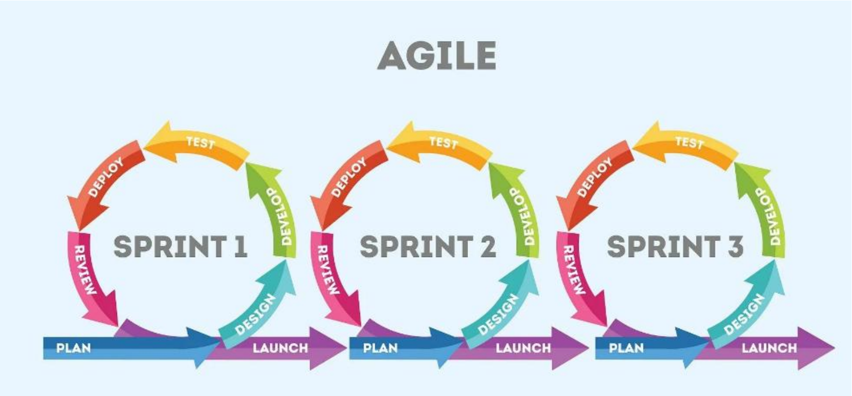


Figure 1.1: Agile development methodology

Source: (Slawek-Polczynska, 2020)

The Agile Software Development Life Cycle (SDLC) technique is applied in the development of the Multimodal Large Language Model-Based Vehicle Recognition for Vehicle Access Control System. Agile is chosen because of its flexible and iterative methodology which facilitates frequent delivery of working software, quick prototyping and ongoing requirement refinement. Projects incorporating cutting-edge technologies like multimodal data processing and huge language models where needs may change during the development cycle. Hence, this project are particularly well-suited for this methodology.

The project is split into manageable, tiny units called sprints under the Agile SDLC. The goal of each sprint which normally lasts one to two weeks is to provide a useful system feature or component. The final result will precisely match the project goals and requirements because of this iterative methodology. This characteristic has enables continuous input and development. Reviews and retrospectives are carried out at the conclusion of each sprint in order to assess progress, resolve issues and make plans for the next sprint.

In the planning and requirement analysis stage of the Agile development process, high-level system needs are determined using problem statements and use case scenarios. Following is the design phase which involves a detailed planning of the system's architecture, data flow and UI/UX components. Core components such license plate verification, vehicle recognition, LLM integration and user interfaces are gradually built and merged during the implementation phase.

Continuous testing is done during each sprint to assure the reliability and quality of every module. This consists of performance checks, image processing validation and functional testing to make sure the system can function in a variety of scenarios. In order to accelerate problem solving and decision-making, the Agile methodology also places a strong emphasis on tight coordination between different roles.

Last but not least, the deployment and maintenance stages guarantee that the system is effectively supplied and updated in response to user feedback and real-world performance. Agile is very successful for developing an intelligent access control system because of its collaborative and adaptable nature which allows individuals to respond quickly to changes and produce a high-quality solution within the time frame specified.

1.8 Proposed Solution

In order to improve the security and effectiveness of vehicle access management, the proposed solution presents a Multimodal Large Language Model-Based Vehicle Recognition System. In comparison with conventional systems that just use license plate recognition, this system uses a variety of data types such as text inputs and vehicle photos to more precisely confirm the identify of the vehicle. The technology intelligently reads and correlates textual and visual data to identify anomalies or mismatches by utilizing the power of a large language model that has already been fine-tuned. For real-time monitoring and management, a React Native-built web and mobile application will act as the interface, while PostgreSQL is selected as the database due to its capacity to securely handle multimedia data. The following subtopic will mainly discuss about the solution to resolve the problems statements that has been mentioned in previous topic and the system architecture of this proposed project.

1.8.1 Solution for problem statements

This project proposes a multimodal vehicle recognition system driven by a Large Language Model (LLM) to increase the security, reliability, and intelligence of vehicle access control systems. In contrast to traditional systems that only use manual verification or license plate recognition, this approach integrates a number of data types including vehicle photos, plate text, and contextual information in order to carry out more precise and flexible verification. The system was designed to cope with real-world complexity and provide more intelligent access decisions by utilizing the reasoning powers of LLMs and the power of multimodal analysis. The implementation consists of a React Native-developed mobile and web interface backed by a PostgreSQL database that can store text and photos.

In order to address the weaknesses of conventional access control systems especially those that depend on hardware-based techniques like RFID tags or cards and human security checks, the suggested approach uses AI-powered recognition to automate the access verification process. This automation offers more consistent, real-time decision-making at access points, lowers the possibility of human error and does away with the need for physical cards that might malfunction in specific situations.

The proposed solution incorporates a multimodal approach that examines both textual and visual inputs in order to address the problem of forged license plates tend to escaping conventional ALPR systems. The system uses a reasoning mechanism made possible by the LLM to match the vehicle's visual characteristics such as color, model and manufacturer with the license plate data rather than relying only on the text on the plate. The ability to intelligently detect inconsistencies between plate and vehicle data makes it far more difficult for malicious people to obtain access via fraudulent or cloned plates.

The system's image preprocessing and segmentation approaches improve clarity and isolate important elements before recognition. This action can help in addressing the difficulties caused by changing environmental conditions like bad weather or low illumination by maximizing the removing the noise in the image. This ensures that the system retains high recognition accuracy even under less-than-ideal circumstances. With further training data, the AI model is further engineered to adjust and get better over time and increasing its robustness and reliability in a variety of scenarios.

1.8.2 Proposed System Architecture

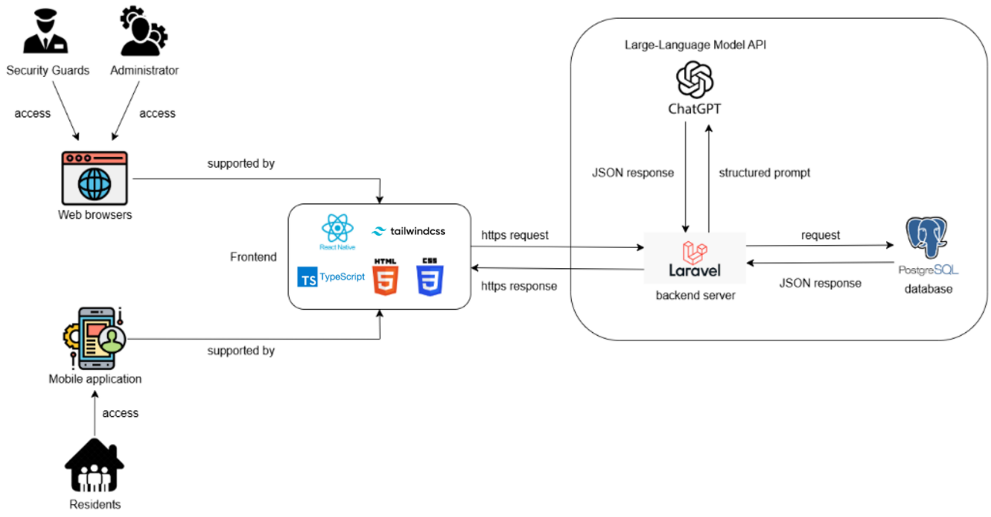


Figure 1.2: System Architecture Flow

(Note: A refined system architecture flow can be found in later chapter 5.2)

Figure 1.2 shows a vehicle access control system that involve usage of Large Language Model (LLM). Security officers, administrators, and residents are among the many stakeholders in the system, and they all use web browsers to communicate with the platform. The main interface for these users is the frontend component, which is intended to be widely accessible. It gathers the required data, including license plate information and car images, and sends it over secure HTTPS connections to the backend Laravel server.

A key component of handling the frontend requests is the backend. In order to analyze the data that has been submitted, it shares with a ChatGPT's API. Clear and machine-readable communication between the LLM and the backend is ensured by the use of JSON-formatted prompts and responses. Once this input has been processed, the LLM returns a decision with a justification then the backend then sends it as a https response to the frontend.

A PostgreSQL database is used in this project to oversee data persistence, houses vital data like user profiles, access logs and car details. This database maintains historical data for audits and troubleshooting, supporting the system's long-term functionality. Sensitive information is kept safe and protected by using HTTPS for all communications including those between the backend and the LLM API and the frontend and backend.

Additionally, the graphic displays the roles of several users such as residents use the platform to seek access, administrators establish system-wide settings and security guards can evaluate and override access decisions. Arrows with labels such as "access," "request," and "response" highlight the dynamic and linked character of the system by showing the data flow between the different components. Overall, the design shows how to effectively combine a vehicle access control systems with LLM-powered logic to improve security and efficiency through automated, data-drivens decision-making.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Vehicle recognition and access control systems have mostly depended on conventional computer vision and machine learning methods in recent years. Basic vehicle recognition and identification have been made possible by these techniques, but they frequently encounter difficulties in complicated surroundings and have limitations when it comes to completely interpreting multimodal data. In locations like parking lots, toll gates, and controlled zones, a precise and effective vehicle recognition is essential for improving security and expediting operations. Since people can fake or alter license plates to get around access constraints, traditional vehicle recognition systems mostly rely on license plate recognition which presents serious security problems. As a result, more sophisticated, multimodal strategies that integrate different data types to enhance security and dependability are becoming more and more necessary.

The goal of this literature review is to examine the state of technology that is relevant to the development of a vehicle identification system for access control applications that is multimodal and based on Large Language Models (LLMs). It specifically looks into current LLM concepts, segmentation methods that are necessary for handling visual input, and related applications in access control view. Additionally, the assessment examines the most recent developments in vehicle recognition systems, reviews web and mobile application frameworks that may support the system's user-facing components and evaluates software development processes appropriate for delivering reliable solutions.

This chapter will pay attention to:

i. Comparison between existing Large-Language Models (LLM)

ii. Identify a suitable Large-Language Model for this project

iii. Comparison between existing segmentation techniques on visual inputs

iv. Identify a suitable segmentation methodology for this project

v. Review similar applications on access control of properties
vi. Identify the key features to be included in the project
vii. Review existing vehicle recognition systems
viii. Comparison among varies of Software Development Lifecycle (SDLC)
ix. Identify a suitable development methodology for this project
x. Comparison between the different type of web and mobile application framework

2.2 Existing Large Language Model (LLM) Review

In the area of artificial intelligence, large language models or LLMs have become a major breakthrough due to their exceptional ability to understand and generate language that is human-like. With a concentration on their possible use in multimodal vehicle recognition systems, this section examines the evolution of important LLM architectures, highlighting their fundamental methods, advantages, and disadvantages.

2.2.1 Instruction Tuning with GPT-4

Based on Peng et al. (2023), one of the most important methods for enhancing large language models' (LLMs') zero-shot generalization is instruction adjustments. Instruction tuning uses a variety of instruction-following instances unlike traditional fine-tuning which requires task-specific datasets. These instructions can be either produced by people or other LLMs to train models. By employing this method, models can carry out invisible activities without the need for clear explanations. According to Li et al. (2024), generated instruction data can significantly enhance model performance. To give an illustration, generated instruction included Alpaca (Stanford CRFM, n.d.) and Self-Instruct Wang et al. (2022). However, the quality of data is a very important factor that will affect the outputs results. Although earlier research used GPT-3.5 to generate data, this paper presented GPT-4 as an improved teacher model, demonstrating that its outputs result in more broad and in-depth models that are better aligned.

The adjustment from human-curated data to machine-generated data is a significant development in instruction tuning. Human annotations were the foundation of early methods like Fine-tuned LAnguage Net (FLAN) and T0, which are expensive to scale. In order to reduce the scale cost, the reduced of the need for manual work by automating the creation of instructions is required. By applying computer-generated information, Alpaca has achieved good performance by producing 52K instruction-output pairs using GPT-3.5 which helps in enhancing this approach. This instruction tuning paper expanded on this and made the case that its outputs are qualitatively better—longer, more diverse, and better organized by utilizing GPT-4. As compare to Alpaca's incremental approach, their simplified one-time generation technique produced extremely effective models. This technique has proved that data quality may outperform generation complexity.

The fine-tuning of LLaMA models on GPT-4-generated data was also investigated in this research with outstanding results. When trained using English instruction data, their LLaMA-GPT4 (7B) model performed better than the bigger Alpaca (13B) model. This has proved that high-quality data can increase the performance on smaller model sizes. They also demonstrated cross-lingual applicability by translating questions and using GPT-4 to generate responses which introduced a Chinese variation for LLaMA-GPT4-CN. In addition to supervised fine-tuning, they also enabled reinforcement learning from machine feedback (RLHF) by training reward models on pairwise comparisons that were gathered through GPT-4. These OPT-1.3B-based models successfully anticipated response quality and providing a scalable solution for human annotations in alignment.

An important aspect of this instruction tuning study was evaluation. In 54.12% of situations, LLaMA-GPT4 outperformed GPT-3.5-tuned Alpaca in human evaluations on the User-Oriented-Instructions-252 benchmark. In some evaluations, the two models performed similarly. These results were further supported by automated assessments that used GPT-4 as a judge. Although LLaMA-GPT4 has a lower size, but it still outperformed Alpaca. Additionally, the reward models in this work validated their application in RLHF pipelines by closely matching the ranks of GPT-4. The advantages of cross-lingual fine-tuning are demonstrated by the fact that translated responses sometimes surpassed GPT-4's native outputs in Chinese tests. The benefits of GPT-4's richer outputs were further shown by ROUGE-L scores on the Unnatural

Instructions benchmark. It demonstrated that GPT-4-tuned LLaMA outperformed Alpaca in creative and long-form production while Alpaca performed excellently in short-answer tasks.

Other than that, Alpaca and Vicuna also demonstrated how open-weight models like LLaMA could compete with patented ones using data that is computer generated while FLAN and T0 illustrated the effectiveness of multi-task instruction tuning. By automating the creation of instructions, Self-Instruct reduced the dependency on human input. Besides from text, models such as OpenFlamingo and LLaMA-Adapter combined LLaMA with vision, but this study emphasized the cross-lingual potential of instruction creating by expanding it to Chinese.

Despite these advances, the study has some drawbacks in which their largest model (7B) is outperformed by 13B variants although their dataset (52K samples) is less than Vicuna's (700K). Future research could explore more languages and modalities, integrate reward models into complete RLHF pipelines, and expand data and model sizes. However, their results highlight how important high-quality automated data is to narrowing the gap between closed and open models. They showed that smaller, instruction-tuned models can function on track with much bigger systems by using GPT-4 as a teacher, opening up the possibility for more effective and easily accessible LLM development.

2.2.2 LLaMA-Adapter V2: Parameter-Efficient Visual Instruction Model

As stated in Gao et al. (2023), large language models (LLMs) have recently shown impressive ability to follow textual instructions, but it is still difficult to apply these capabilities to the visual domain. When compared to robust proprietary models like GPT-4, traditional methods like LLaMA-Adapter have demonstrated shortcomings in addressing flexible visual instructions and multi-modal reasoning mission. This gap led to the creation of increasingly complex parameter-efficient techniques that successfully integrate linguistic and visual understanding without needing expansive model fine-tuning or large amounts of multi-modal training data.

With models like Alpaca and Vicuna showing the value of fine-tuning on machine-generated instruction data, the field of instruction-following LLMs has achieved huge growth. However, t hese methods usually call for updating every model parameter which is computationally costly. This problem has been resolved by parameter-efficient fine-tuning (PEFT) techniques like LoRA and the original LLaMA-Adapter, which introduced low-rank adaptations or lightweight adapters. Models like MiniGPT-4 and LLaVA have tried to connect visual encoders with LLMs in the visual domain, but they still require extensive fine-tuning of the language models and rely on an enormous amount of multi-modal training data. These shortcomings emphasize the necessity for more effective strategies that helps to retain the model's current language capabilities while achieving excellent visual instruction following.

By providing several of important advances, LLaMA-Adapter V2 represents an important milestone in this approach. The model maintains a very low total number of trainable parameters which was just only 0.04% of LLaMA's parameters while revealing new learnable parameters throughout the network. For instance, normalization layers and bias/scale terms in linear layers. This method enables instruction-following knowledge to be learned distributed throughout the model layout. In order to avoid compromising with textual and visual processing in deeper layers, the technique uses an early fusion strategy to introduce visual tokens only into the first transformer layers. This architectural decision is especially crucial for including visual understanding while maintaining the model's powerful language capabilities.

The integrated training methodology of LLaMA-Adapter V2 which employs separate parameter groups for various tasks is a remarkable feature. While instruction-following data trains the late-layer adapters and normalization parameters, image-text pairs mainly update the visual projection layers and early gating mechanisms. This division makes it more difficult for the model's existing instruction-following capabilities to be overridden by the visual features. By integrating expert systems during inference, such as pre-trained captioning and OCR models, which provide more contextual information without requiring additional training.Thus, the model drastically enhances its visual understanding. The model can perform better on visual tasks while preserving parameter efficiency through this modular approach.

Results from experiments show that these innovations are effective. LLaMA-Adapter V2 produces more thorough and precise responses than its predecessor in language instruction following. It even performs competitively against ChatGPT in multi-turn conversation evaluations. Even though the model uses a lot less training data, it performs on standard benchmarks for visual tasks that are equivalent to those of specialized systems like Bootstrapping Language-Image Pre-training (BLIP). Expert system integration is especially useful for solving complicated problems that ask for both textual and visual reasoning such as describing the visual impact of an image or producing recipes from food photos. When it comes to visual content that is not distributed, the model still exhibits limits. This has indicate there are areas that require further development.

The performance of the LLaMA-Adapter V2 points to a number of important areas for further study in the tuning of visual training. The parameter-efficient approach of the model shows that large language models do not require extensive retraining to achieve strong multi-modal performance. The successful application of expert systems indicates potential directions for adding more specialized modules like object detection models in order to improve visual comprehension even further. To find ways to overcome present constraints, future research may examine integrating these approaches with other Parameter-efficient fine-tuning (PEFT) techniques such as LoRA, or examining the possibilities of limited quantities of high-quality multi-modal instruction data. Such parameter-efficient techniques will probably be essential to the development of adaptable, multi-modal AI systems that can process complex linguistic and visual instructions while still being computationally feasible to create and implement as the field develops.

2.2.3 CLIPath: Fine-tune CLIP with Visual Feature Fusion for Pathology Image Analysis Towards Minimizing Data Collection Efforts

Based on Lai et al. (n.d.), The alignment of textual and visual data for zero-shot transfer learning has been achieved with unexpected success by recent developments in language-vision models, especially Contrastive Language-Image Pre-training (CLIP). Because of its contrastive learning approach that trains a text encoder which is Transformer and a vision encoder which is ResNet or Vision Transformer to map inputs into a common feature space, CLIP can generalize across a variety of domains. Although CLIP has been adapted for tasks such as 3D recognition (PointCLIP) and video interpretation (CLIP-ViL), its use in medical imaging especially the field of pathology is still not well established. This gap is important because pathology image analysis has distinct difficulties, such as domain changes between specialist medical data and natural images used to train CLIP, and the lack of labeled datasets because of the high cost and level of skill needed for tagging.

The disadvantages of traditional supervised learning in medical image analysis led to interest in using pre-trained models such as CLIP. However, in order to accomplish modest zero-shot performance (~60% accuracy), current methods like MedCLIP which rely on large-scale curated datasets like 570K image-text pairs may not be practical in many clinical contexts. On the other hand, parameter-efficient fine-tuning (PEFT) techniques, such as prompt tuning (CoOp) and adapters (CLIP-Adapter), attempt to reduce computational cost but frequently encounter difficulties with domain adaptation. For example, CLIP-Adapter's feature combining may not adequately resolve the semantic difference between natural and medical images, while CoOp's prompt optimization can negatively impact performance when utilized on out-of-distribution data. These difficulties show that a customized strategy that strikes a compromise between effectiveness, flexibility, and maintaining CLIP's previously acquired information is required.

In order to overcome these constraints, subsequent research has investigated hybrid approaches like lightweight fine-tuning and semi-supervised learning such as FixMatch, FlexMatch and others, although these techniques both either require high computational resources or fall short of maintaining CLIP's zero-shot capabilities. To close this gap, the proposed CLIPath framework introduces two innovations:

i. A Language-Vision Alignment (LVA) contrastive loss that maintains alignment between image and text features during fine-tuning (Mo, Xia and Markevych, 2023).

ii. A Residual Feature Connection (RFC) module that fuses task-specific features with CLIP's pre-trained embeddings via a lightweight adapter.

Rapid adaption to pathological datasets like PCam and MHIST is made possible by RFC's parameter efficiency (0.04% of CLIP's trainable parameters) and LVA's regularization, which reduce overfitting and achieve notable accuracy gains. For example, there is 19% improvement with only 0.1% labeled data.

The advantages of CLIPath over current techniques are shown by empirical evaluations. RFC fine-tuning improves performance to 81.5% with only 0.5% labeled data, surpassing CLIP-Adapter and CoOp by 25% with 5× quicker training times, while zero-shot CLIP achieves 56.5% accuracy on the PCam dataset. Similarly, CLIPath's scalability is demonstrated on MHIST. It achieves 74.8% accuracy compared to 36.9% zero-shot using 50% of the data. Although there are still difficulties in expanding the framework to multi-class classification and segmentation tasks, these results highlight the potential of lightweight, knowledge-preserving adapters for medical applications. To further reduce the performance difference with fully supervised approaches, future strategies might incorporate expert-guided tagging systems or hybrid architectures. For example, on PCam with 100% data, it achieved 92.8% accuracy.

In conclusion, CLIPath is a potential step toward effective adaption of language-vision models in pathology with little data. The framework achieves near-state-of-the-art performance while decreasing computing costs by combining the alignment preservation of LVA with the parameter efficiency of RFC. This is an important advantage for clinical application. Its expansion to more complicated domain shifts and larger medical imaging jobs may be investigated in future studies.

2.2.4 LLM Model to be included in projects

Table 2. 1 : Comparison among different LLM model for fine tuning

Aspect	**Instruction Tuning with GPT-4** (Peng *et al.*, 2023)	**LLaMA-Adapter V2** (Gao *et al.*, 2023)	**CLIPath** (Lai et al., n.d.)
Focus	Enhancing LLM generalization via instruction tuning	Parameter-efficient visual instruction tuning	Medical image analysis with CLIP
Key Innovation	GPT-4 as teacher model for high-	Early fusion of visual tokens +	Residual Feature Connection

	quality data generation	modular adapters (0.04% params)	(RFC) for pathology
Data Efficiency	52K GPT-4-generated instructions outperform 700K Alpaca data	Minimal multi-modal data required	0.1% labeled data achieves +19% accuracy
Performance	LLaMA-GPT4 (7B) > Alpaca (13B) in human evaluations	Matches ChatGPT in multi-turn conversations	81.5% accuracy (PCam) vs. 56.5% zero-shot CLIP
Strengths	Cross-lingual support (e.g., Chinese LLaMA-GPT4-CN)	Preserves LLM's language capabilities	Maintains CLIP's zero-shot ability
Limitations	Smaller model size (7B) vs. competitors (13B+)	Struggles with out-of-distribution visuals	Limited to binary classification in pathology
Applicability to Your Project	Ideal for contextual reasoning	Useful if adding visual features	Less relevant

The three studies each focus on a different AI model optimization problem. In order to enable zero-shot task performance without explicit training, the Instruction Tuning with GPT-4 study focuses on enhancing the generalization capabilities of large language models (LLMs) through high-quality, machine-generated instructions. As another option, LLaMA-Adapter V2 focuses on visual instruction adapting with the goal of bridging the gap between text and image understanding in a way that uses the fewest possible parameters. With minimal labeled data, CLIPath which is a medical image analysis specialist adapts the CLIP model for pathology diagnosis. Although efficiency and performance are the main focus of all three studies, CLIPath is used for domain-specific visual analysis, LLaMA-Adapter V2 is used for multimodal reasoning, and GPT-4 is used for general language tasks.

High-quality synthetic data production is introduced by the GPT-4 instruction tuning approach, showing that when refined on GPT-4-curated datasets, smaller models like LLaMA 7B can perform better than larger ones like Alpaca 13B. This technique enhances cross-lingual flexibility while reducing dependency on expensive human annotations. With only 0.04% of trainable parameters, LLaMA-Adapter V2 enables vision-language integration through innovative early visual token fusion and modular adapters. In order to maintain CLIP's previously learned information while optimizing for medical imaging, CLIPath presents a Residual Feature Connection (RFC) module, which achieves high accuracy with less than 1% labeled data. In conclusion, a ll three methods optimize for distinct constraints, CLIPath for domain adaptation in data-scarce scenarios, LLaMA-Adapter V2 for lightweight multimodal tuning and GPT-4 for scalable instruction data.

Fine-tuning LLaMA on simply 52K GPT-4-generated samples surpasses Alpaca's 700K GPT-3.5-based dataset, demonstrating that quality is more important than quantity. For deployment to remain cost-effective, this efficiency is essential. By using pre-trained visual encoders and requiring little task-specific adjustment, LLaMA-Adapter V2 significantly minimizes data requirements. This has make it suitable for applications with a small number of labeled images. Because of its hybrid feature fusion, CLIPath performs exceptionally well in low-data regimes and boosting accuracy by 19% on pathology images with just 0.1% labeled data. While all three approaches reduce reliance on data, GPT-4 succeeds at adapting to a variety of linguistic tasks, while LLaMA-Adapter V2 and CLIPath concentrate on efficiency related to vision.

GPT-4-tuned LLaMA (7B) performed better than Alpaca (13B) in 54.12% of human evaluations, demonstrating that better data quality can offset a smaller model size. Despite utilizing significantly fewer parameters, LLaMA-Adapter V2 competes with specialized models such as BLIP in vision-language benchmarks and matches ChatGPT in conversational tasks. CLIPath's effectiveness in medical imaging is demonstrated by its 81.5% accuracy on pathology datasets as compared to 56.5% for zero-shot CLIP. CLIPath dominates domain-specific visual analysis, LLaMA-Adapter V2 succeeds in multimodal tasks and GPT-4 leads in language reasoning.

Given its exceptional cross-lingual compatibility such as Chinese LLaMA-GPT4-CN and flexibility in following instructions, GPT-4 is a great choice for dynamic jobs like fraud detection in vehicle access systems. Effective applications like car model and color identification without the need for expensive full-model retraining are made possible by the LLaMA-Adapter V2. This succeeds at maintaining essential LLM capabilities while incorporating vision. Even though it is tailored, CLIPath preserves CLIP's zero-shot adaptability in a unique way while optimizing for certain domains which may include identifying emergency or customized car models. For various aspects of AI-driven recognition systems, each model has unique benefits like CLIPath for domain-specific visual tuning, LLaMA-Adapter V2 for lightweight multimodal fusion, and GPT-4 for reasoning.

Every model has distinct benefits, but there are also significant drawbacks. For multimodal applications like image-based vehicle detection, GPT-4 must be integrated with a vision model like YOLOv8 as it is mostly text-based. Even though LLaMA-Adapter V2 is effective at combining language and vision, its resilience in real-world situations is limited by its inability to handle out-of-distribution pictures such as veiled or unusual license plates. Although CLIPath works well for narrow adaptations, it is currently limited to binary classification and is not scalable for complex vehicle recognition tasks that involve a variety of classes or fine-grained characteristics. When using AI in real-world, large-scale applications like multimodal vehicle access control, the drawbacks included GPT-4's reliance on external vision systems, LLaMA-Adapter V2's sensitivity to visual abnormalities, and CLIPath's limited scope highlight the necessity for customized solutions.

GPT-4o is the best option for this project because of its easier interaction with visual models, scalability, and superior reasoning skills. The contextual depth of GPT-4 is necessary for fraud detection such as identifying mismatched license plates or cars, even though LLaMA-Adapter V2 performs exceptionally well in parameter-efficient vision-language tasks. Despite its efficiency, CLIPath is unrelated to your use case and excessively specialized for medical imaging. The dynamic decision-making capabilities of GPT-4's instruction-tuning framework include granting access based on contextual rules, integrating seamlessly with segmentation models like YOLOv8 for license plate recognition and fine-tuning at a cheap cost while maintaining excellent performance with smaller datasets. In contrast, CLIPath provides no apparent advantage for vehicle detection and LLaMA-Adapter V2 requires further vision-language alignment work. Consequently, GPT-4 offers the optimal ratio of efficiency, flexibility, and reasoning.

2.3 Existing Segmentation Techniques

Vehicle recognition systems rely on image segmentation because it allows vehicles to be precisely isolated from complex backgrounds. The precision of subsequent processes like classification and identification is increased by efficient segmentation. This section examines current segmentation techniques with a focus on the way they work for processing visual input in order to reduce noise and increase the accuracy of result.

2.3.1 Segment Anything

Natural language processing (NLP) has been transformed by recent developments in large language models (LLMs), which allow for fast engineering to enable zero-shot and few-shot generalization. Motivated by this achievement, Kirillov et al. (2023) present the Segment Anything Model (SAM) which is a foundation model for picture segmentation that requires little task-specific training and can generalize across a variety of tasks. The main objective of SAM is to build a promptable segmentation model that can produce precise masks from different input prompts, such text, boxes, or points, in a manner similar to how models like CLIP and GPT-3 adjust to new tasks through prompting. With this method, computer vision is undergoing an important shift from task-specific designs to adaptable, all-purpose models.

A key contribution of SAM is the introduction of the promptable segmentation task in which the model must produce valid segmentation masks for any given prompt even though in ambiguous scenarios. To address this, SAM predicts several feasible masks. For example, a single point on a shirt could relate to the clothing or the person who is wearing it. There are three primary components to the model architecture:

i. a heavyweight image encoder based on a Vision Transformer (ViT) pre-trained by a Masked Autoencoder (MAE).

ii. a prompt encoder that can process sparse inputs like text, boxes, or points.

iii. a lightweight mask decoder that creates segmentation masks in real-time which is about 50 ms per prompt.

By reusing image embeddings over several prompts, this architecture ensures performance while creating interactive applications possible.

The largest segmentation dataset to date which is the SA-1B dataset was created by the authors using a scalable data engine to train SAM. It contained 1.1 billion high-quality masks from 11 million images. The data engine functions in three stages:

i. semi-automatic annotation. In this stage, SAM pre-generates confident masks for annotators to supplement.

ii. assisted-manual annotation in which SAM helps human annotators refine masks.

iii. fully automatic mask generation where SAM predicts masks by using a grid of point prompts that have been filtered for quality and stability.

The quality of SA-1B's masks was confirmed to be comparable to that of human annotations. 94% of automatically generated masks achieved an Intersection-over-Union (IoU) of above 90% with professionally corrected versions.

Two innovations of SAM include its zero-shot transfer capabilities that enables it to carry out tasks like edge detection, instance segmentation, and object proposal generation without the need for extra training, and its ambiguity-aware design which enables it to handle unclear prompts by anticipating multiple masks. Experiments on 23 different datasets showed that SAM achieves greater human-rated mask quality and performs better in single-point segmentation than specialist models like RITM. Furthermore, SAM's adaptability enables it to be integrated with other models. For instance, it can be used to segment data by using the bounding box outputs of an object detector as prompts.

SAM has drawbacks despite of its advantages. As compared to specialist techniques, it has trouble with fine-grained structures and distinct borders and its text-to-mask capability is still in its early stages. Additionally in certain applications, real-time performance may be limited by the image encoder's computational cost. However, SAM's capacity to be deconstructed makes it an effective tool for more complex systems like ego-centric vision or 3D reconstruction. The release of SA-1B offers a useful resource for developing foundation models in computer vision with a focus on responsible AI which includes eliminating potential biases and guaranteeing dataset variety.

In a nutshell, SAM combines task generalization, effective architecture design, and large-scale data curation. It has marked a significant advancement toward general-purpose segmentation models. Although community adoption is necessary for its long-term success, SAM establishes a standard for promptable vision models and opens the door for further study of flexible, scalable computer vision systems. The model's potential to revolutionize segmentation in both research and practical applications is demonstrated by its capacity to do zero-shot segmentation and blend in seamlessly with larger operations.

2.3.2 Engineering Vehicle Object Segmentation Algorithm Based on Improved YOLOv8x-seg

Based on (Wu, Feng and Zhang, 2024), the necessity for precise object segmentation of engineering vehicles operating close to power transmission infrastructure has been brought to light by recent developments in computer vision for industrial safety applications. Traditional segmentation techniques often come fail in diverse field situations. This has made it difficult for automated monitoring systems to safely operate cranes, excavators, and other machinery near electrical grids. Modern deep learning architectures, especially the YOLO (You Only Look Once) family of models, have shown promising capabilities for real-time object detection and segmentation tasks while traditional methods that rely on spatial-domain processing struggle with background noise and small-object detection. However, recent studies have shown that even the most advanced YOLO implementations have limits when it deals with segmenting small but crucial safety components of engineering vehicles such as excavator arms and crane booms especially against cluttered backdrops close to power transmission lines.

By creatively combining adaptive multi-scale feature fusion and frequency-domain processing, the suggested HFF-YOLO architecture marks a substantial breakthrough in this field. This strategy builds on the YOLOv8x-seg basis by introducing three significant technological advancements that together overcome the shortcomings of current approaches. The High-Frequency Information Extraction (HFIE) module first uses the Fourier Transform to convert input images into the frequency domain. A high-pass filter is then employed to eliminate texture and edge information while reducing background noise. The visibility of minor vehicle components that might be hidden in conventional spatial-domain analysis is improved by this frequency-domain processing. Secondly, a progressive fusion method that better maintains shallow-layer detail information while incorporating deep-layer semantic context is used by the Asymptotic Feature Pyramid Network (AFPN) to replace traditional feature pyramid topologies. Since AKConv supports variable kernel forms, its integration into AFPN significantly improves the model's capacity to handle objects of different scales. Lastly, to optimize the fusion process for multi-scale object recognition and minimize information loss between non-adjacent layers, the Adaptive Spatial Feature Fusion (ASFF) module uses a Softmax-based attention mechanism to dynamically weight features from various network levels.

In this paper, a specific dataset with 3,600 annotated photos which knowas ENV3K is created. It consists of four different kinds of engineering vehicles including trucks, cranes, excavators, and bulldozers that operate beneath electricity transmission lines in order to aid in the construction and assessment of this innovative architecture. This dataset, which offers a variety of instances of vehicles in operational situations with different dimensions, orientations, and backdrop complexities helps to close a significant gap in the training resources currently accessible for this particular application domain. The dataset is especially useful for benchmarking since it focuses on real-world situations close to power infrastructure. Hence, it can ensure that models trained on ENV3K must overcome the same difficulties encountered by real monitoring systems.

According to the experimental results, the HFF-YOLO method is effective across a variety of performance metrics. With a mAP50 of 81.2% on the ENV3K dataset, the model outperforms the baseline YOLOv8x-seg architecture by 0.8%. Particularly, the system exhibits a 4.7% increase in mAP50 for crane arm recognition. This is a crucial skill for averting hazardous connections with power lines. By having this, the system is demonstrating its exceptional strength in small-object segmentation. While keeping a low parameter count of 54.7 million as opposed to 71.7 million in the baseline model, the recall rate increases from 72.6% to 73.9%. Ablation studies show that each of the HFIE, AFPN, and ASFF modules improve performance. Removing any one of them led to a 1.3% drop in mAP50, proving that all three modules work well together. Visual comparisons make it evident that the model performs better at lowering false negatives for minor vehicle parts while retaining strong segmentation of bigger structures.

Nevertheless of these developments, there are several drawbacks to the HFF-YOLO architecture that should be taken into account. The deployment on resource-constrained edge devices may be difficult due to to the large increase in computational cost to 503.5 GFLOPS which is 46% higher than YOLOv8x-seg. Furthermore, not all vehicle types have the same performance advantages. To give an illustration, larger objects such as truck bodywork show relatively moderate gains in comparison to the significant improvements seen with small components. These results highlight areas that require further development, particularly with regard to optimizing the model's performance in real-world deployment scenarios. Possible paths include expanding training datasets to include more vehicle types and operating conditions, improving feature fusion mechanisms to sustain accuracy gains while lowering resource requirements, and using model compression techniques to lower computational overhead.

The HFF-YOLO architecture greatly improves the fields of safety monitoring and industrial computer vision. By effectively combining sophisticated feature fusion approaches with frequency-domain processing, it shows a promising way forward for addressing the long-standing problem of small-object segmentation in complicated contexts. Although there is architectural innovations, especially the HFIE module's innovative use of frequency-domain analysis may inspire new approaches to object segmentation beyond the specific case of engineering vehicles. However, the specialized ENV3K dataset offers a valuable resource for future research in this application domain. The HFF-YOLO technique is an achievable way to improve safety by more precisely identifying possible threats close to transmission lines as computer vision technologies are further included into power infrastructure monitoring systems. Future research in this field would profit from examining how well these methods transfer to other industrial monitoring applications where it is still difficult to detect small objects in cluttered surroundings.

2.3.3 Fast-SCNN: Fast Semantic Segmentation Network

As stated in Poudel, Liwicki and Cipolla (2019), semantic segmentation has emerged as a key feature that makes it possible for vital applications like augmented reality interfaces and driverless cars in current computer vision systems. Although recent advancements in segmentation accuracy have been dominated by encoder-decoder systems, their computational complexity frequently makes them unsuitable for real-time applications, especially on embedded devices with limited resources. Conventional methods that depend on large pre-training datasets and high-capacity networks contribute to efficiency issues, posing major obstacles to implementation in latency-sensitive situations. A novel architecture known as Fast-SCNN (Fast Segmentation Convolutional Neural Network) was created by Poudel et al. to overcome these constraints, offers above-real-time performance on high-resolution imagery (1024 x 2048 pixels) with competitive accuracy and a small memory footprint.

The fundamental innovation of Fast-SCNN is its innovative architecture which carefully achieves a compromise between segmentation performance and processing efficiency. A key component of this strategy is the learning to downsample module. This strategy shares low-level feature extraction across resolution branches and hence removing duplicated computations typical of multi-branch designs. This module's effective three-layer structure which consists of two depthwise separable convolutions after a standard convolution. It allows for fast downsampling while maintaining important spatial information. In order to maintain a lean parameter count of only 1.11 million, the network architecture utilizes depthwise separable convolutions and inverted residual blocks inspired by MobileNet-V2 along with a global feature extractor for contextual understanding and a feature fusion module that integrates multi-scale information. An Nvidia Titan Xp GPU achieved 68.0% mean Intersection over Union (mIoU) on the Cityscapes dataset at 123.5 frames per second. This is a remarkable performance characteristic that greatly outperforms similar real-time models like BiSeNet which achieved 71.4% mIoU at 57.3 fps and GUN which reached 70.4% mIoU at 33.3 fps.

In addition to its innovative architecture, Fast-SCNN challenges standard procedures in network training. Unlike with standard procedures in the industry, the authors show that their low-capacity model benefits only slightly (+0.5% mIoU) from ImageNet pre-training. This result implies that aggressive data augmentation and longer training epochs can be effective substitutes for resource-intensive pre-training processes in well-designed efficient architectures. The model's adaptability is further demonstrated by its capacity to process inputs with lower resolution (512×1024 pixels) while maintaining competitive performance (62.8% mIoU at 285.8 fps) without necessitating architectural changes. As a result, it become especially appropriate for deployment across a range of hardware configurations.

The success of Fast-SCNN is confirmed by experimental assessments on the Cityscapes benchmark as the model maintains its real-time performance advantage while reaching 68.0% class-level and 84.7% category-level mIoU. The significance of the skip connection design is highlighted by ablation tests. The tests reveal a 4.92% mIoU degradation upon its removal which mostly affects border precision and tiny object segmentation. Regardless of the model's outstanding capabilities, a few limitations should be taken into consideration. These include an inherent accuracy-speed tradeoff. For instance, there is about 3% lower mIoU when compared to BiSeNet. Besides that, possible difficulties when deploying to ultra-low-power embedded devices without the use of extra optimization techniques like quantization.

In the future, Fast-SCNN provides a strong basis for more studies on effective semantic segmentation. In this paper, a number of exciting paths, such as hardware-specific optimizations aimed at FPGA or ASIC implementations, domain adaption for particular applications like medical imaging, and network quantization and pruning to further minimize computational overhead were pointed out. Fast-SCNN represents a major advancement in real-time segmentation capabilities by effectively combining the robustness of encoder-decoder frameworks with the efficiency advantages of two-branch designs. This allows for the creation of workable solutions for deployment scenarios where performance cannot be compromised but computational resources are limited.

2.3.4 Comparative Analysis on Segmentation Techniques

Table 2. 2 : Comparison among varies segmentation techniques

Aspect	**Segment Anything (SAM)** (Kirillov *et al.*, no date)	**HFF-YOLO (Improved YOLOv8x-seg)** (Wu, Feng and Zhang, 2024)	**Fast-SCNN** (Poudel, Liwicki and Cipolla, 2019)
Generalization	Excels in zero-shot and few-shot generalization across diverse tasks due to its foundation model nature. Can handle ambiguous prompts by predicting multiple masks.	Specialized for engineering vehicles, particularly small components like crane arms, but lacks generalization beyond its trained domain.	Optimized for real-time semantic segmentation in specific scenarios (e.g., autonomous driving) but not designed for generalization.
Architecture	Uses a Vision Transformer (ViT) image encoder, prompt encoder for text/boxes/points, and lightweight mask decoder for real-time performance.	Combines frequency-domain processing (HFIE module) with adaptive multi-scale feature fusion (AFPN and ASFF) for small-object segmentation.	Employs depthwise separable convolutions and inverted residual blocks for efficiency, with a focus on low computational overhead.
Performance	Achieves high-quality masks (94% IoU > 90%) and	Improves mAP50 by 0.8% over YOLOv8x-	Achieves 68.0% mIoU at 123.5 fps on

	outperforms specialist models in human-rated quality. Computational cost may limit real-time applications.	seg, with a 4.7% boost for crane arms, but computational cost increases by 46%.	Cityscapes, making it highly efficient for real-time use but with slightly lower accuracy than competitors.
Data Requirements	Trained on SA-1B (1.1B masks from 11M images), enabling broad generalization. Requires minimal task-specific training.	Requires specialized dataset (ENV3K) with annotated engineering vehicles, limiting adaptability to other domains.	Benefits minimally from ImageNet pre-training, relying instead on aggressive data augmentation and longer training epochs.
Use Case Fit	Ideal for applications requiring flexibility, multimodal prompts (text, boxes, points), and integration with other models.	Best suited for industrial safety monitoring of engineering vehicles near power lines, with a focus on small-object segmentation.	Designed for latency-sensitive applications like autonomous driving or augmented reality, where speed is critical.

From aspect of generalization, with training on the large SA-1B dataset, Segment Anything (SAM) is a foundation model that excels in zero-shot and few-shot generalization and can handle ambiguous prompts by predicting several masks. This enables broad adaptability without task-specific fine-tuning. On the other hand, HFF-YOLO which is an improved YOLOv8x-seg is a specialized model that focuses on engineering vehicle segmentation, specifically for industrial safety near power lines. It uses adaptive multi-scale feature fusion (AFPN & ASFF) and frequency-domain processing (HFIE module) to detect small components like crane arms, but its generalization is restricted to its specialized ENV3K dataset and does not support multimodal prompts. Fast-SCNN is not appropriate for tasks requiring flexibility or ambiguous prompt handling because it struggles with zero-shot adaptation and multimodal inputs. It is made for real-time semantic segmentation in constrained environments and achieves efficiency through depthwise separable convolutions and inverted residual blocks.

In order to efficiently generate high-quality masks, SAM combines a lightweight mask decoder, a prompt encoder that can read text, boxes, or points, and a Vision Transformer (ViT) as its image encoder. In order to improve small-object segmentation, HFF-YOLO combines adaptive multi-scale feature fusion (AFPN and ASFF) with frequency-domain processing, which is integrated through its HFIE module, to improve YOLOv8x-seg. With an emphasis on reduced latency for real-time applications, Fast-SCNN is built with depthwise separable convolutions and inverted residual blocks to optimize computational efficiency.

With over 94% of its masks having an IoU above 90%, SAM produces extremely accurate segmentation masks that frequently beat specialist models in terms of human-rated quality. Real-time deployment is constrained by its potentially high computing cost. Despite a 46% increase in computational cost, HFF-YOLO greatly increases performance on small components like crane arms and improves mean average precision (mAP50) marginally over YOLOv8x-seg overall. Although its accuracy is marginally lower than that of other models, Fast-SCNN provides extremely quick segmentation, attaining 68% mIoU at 123.5 frames per second, making it perfect for applications requiring speed.

SAM can generalize widely with less task-specific training thanks to training on the large dataset (SA-1B), which comprises 1.1 billion masks from 11 million pictures. HFF-YOLO's adaptation to other domains is limited because it depends on the specific ENV3K dataset with annotations for engineering vehicles. In order to accomplish its effectiveness in semantic segmentation, Fast-SCNN mostly relies on substantial data augmentation and lengthy training schedules, with some pre-training from ImageNet.

Applications requiring flexibility and multimodal fast processing are most suited for SAM, which makes it perfect for complicated picture jobs or integration with other models. When it comes to small-object segmentation in safety-critical situations, such as vehicles near power lines, HFF-YOLO is especially useful for industrial monitoring of engineering vehicles. Fast-SCNN prioritizes real-time speed above wide applicability and is intended for applications like augmented reality and autonomous driving where low latency is crucial.

2.4 Existing Similar Application

2.4.1 i-Neighbour

According to TimeTec (2025), i-Neighbour is a complete smart community management system designed to improve residential areas' connectivity, convenience and security. Numerous features are integrated onto the platform to help residents, management offices and security staff maintain efficient and safe operations. The visitor management system which enables locals to pre-register visitors using a smartphone app, is a fundamental feature of i-Neighbour. A QR code is created upon registration to provide easy access to the guardhouse. For visitors from Malaysia, the system also facilitates MyKad registrationn which guarantee effective and trustworthy identity verification.

In addition to visitor management, i-Neighbour offers emergency alarms, e-billing integration, announcement distribution and facility booking. The portal is a one-stop shop for residential participation, allowing residents to handle payments and take part in community polling. i-Neighbour is scalable and flexible enough to accommodate a range of home configurations thanks to its cloud integration and language support.

2.4.2 MyTaman

As stated in Zoinla (2019), MyTaman is a community platform built on IoT and neighborhood security that prioritizes connectivity and safety in real time. It provides a combination of hardware and software solutions that enhance communication and surveillance in contemporary residential environments.

One notable feature is the MyVMS (My Visitor Management System) which enables locals to use a mobile app to pre-register guests. Visitors can scan their ID or driver's license as they arrive, and the integrated system notifies the guards.

In addition, MyTaman has a 24/7 emergency SOS function that, when used, instantly notifies security personnel and pre-programmed contacts. The system also has smart access features that let people use their smartphones to access common facilities. The platform enhances the efficiency and accountability of security patrols by supporting guard patrol monitoring via a paperless app-based system.

2.4.3 JaGaApp

Based on Red Ideas (2025), the JaGaApp, created by JaGaSolution, acts as a virtual link between gated community residents, security guards and management. It is intended to improve communal living, facilitate communication and guarantee neighborhood safety. Residents can speak with guards remotely thanks to the app's wireless intercom technology. Pre-approval of visitors is made possible by its visitor registration system which guarantees the security of entry and exit points.

Additional interesting characteristics include a resident feedback system, digital notice boards for announcements, facility booking modules and emergency help notifications. JaGaApp lessens dependency on manual procedures while promoting community involvement. The platform's appeal among contemporary residential complexes can be attributed to its mobile-first design which makes it easy to use and accessible for users of all ages.

2.4.4 TimeTec VMS

As stated in TimeTec Cloud (2025), a cloud-based visitor management system called TimeTec VMS was developed to update how commercial and residential facilities monitor and control visitor entry. The system prioritizes real-time control, automation and security. Each visitor is given a QR code for entrance and residents and staff can pre-register guests using the app. The platform offers a comprehensive access control solution by easily integrating with TimeTec IoT devices such as smart doors and turnstiles.

In order to facilitate evacuation processes, TimeTec VMS additionally incorporates emergency list tracking which displays all visitors that are now on the property. The system provides comprehensive analytics and reporting capabilities and supports a number of languages. Because of its modular architecture, it may be utilized in both large corporate settings and modest domestic settings.

2.4.5 Visitorz

According to (VISITORZ TECH PRIVATE LIMITED, n.d.), a visitor management app called Visitorz was created specifically for residential communities with the goal of achieving easy-to-use yet efficient access control. Residents can use the platform to pre-register guests, who will then be given a QR code to scan at the gate to gain admission. In order to expedite check-ins and cut down on wait times, the system alerts the host when guests arrive.

Furthermore, Visitorz keeps a thorough access log that enables management and residents to monitor visitor history for security audits. Although it concentrates on the essential visitor-related features, the platform's ease of use makes it simple to implement and administer, especially for smaller residential areas or communities searching for a cost-effective solution.

2.4.6 Features Analysis on Similar Applications

Table 2. 3 : Comparison among features between similar applications and proposed system

Features	**i-Neighbour** (TimeTec, 2025)	**MyTaman** (Zoinla, 2019)	**JaGaApp** (Red Ideas, 2025)	**TimeTec VMS** (TimeTec Cloud, 2025)	**Visitorz** (VISITORZ TECH PRIVATE LIMITED, no date)	**Proposed System**
Visitor Pre-registration	Yes	Yes	Yes	Yes	Yes	Yes
QR Code Access	Yes	Yes	Yes	Yes	Yes	Yes

ID/License Scanning	No	Yes	No	Yes	No	No
Emergency Alerts (SOS)	Yes	Yes	Yes	Yes	No	Yes
Wireless Intercom	Yes	Yes	Yes	Yes	No	No
Facility Booking	Yes	No	Yes	No	No	No
E-Billing & Payment Integration	Yes	Yes	Yes	Yes	No	No
Guard Patrol Monitoring	Yes	Yes	Yes	Yes	No	Yes
Multilingual Support	Yes	Yes	Yes	Yes	No	No
AI-Based Vehicle Recognition	No	No	No	No	No	Yes
GPT-4o Integration for Communication	No	No	No	No	No	Yes
Real-time Data Analytics	Yes	Yes	Yes	Yes	No	Yes

Although current programs provide a number of functions for community security and guest management, none integrate cutting-edge technology like AI-based car identification or integration with massive language models like GPT-4o. The greater security offered by the suggested system attempts to close this gap. Unauthorized access can be reduced with AI-powered car recognition. Additionally, better communication is made possible by GPT-4o integration, which makes interactions between residents and the management system more organic and effective. Finally, proactive decision-making and better community management are made possible by comprehensive analytics with real-time data analysis. The suggested method provides a more reliable and clever answer for current residential communities by combining these cutting-edge technology.

2.5 Existing Vehicle Recognition System Review

Before beginning a project, examining the existing vehicle recognition systems can yield important information and insights that help with decision-making regarding the project's functionality, direction, and probability of success.

This section will mainly focus on research for vehicle plate recognition part, which is the most usual ways utilized in single modal vehicle recognition systems to recognise a vehicle. Looking at diverse structures of license plate recognition system can assist in understanding the algorithm and pipelines used in it and findings for selecting suitable method for vehicle plate detection and recognition process.

2.5.1 Efficient license plate recognition in unconstrained scenarios

According to Wei et al. (2024), Efficient Automatic License Plate Recognition (EALPR) is a framework that designed for unconstrained scenarios. To give an illustration, license plates that are distorted because of issues of perspective which is one of the unrestricted situations.

A lightweight vehicle plate detection was applied in this framework. This has included the structure of effective object detection approach like anchor-free strategies, CenterNet, EfficientDet, and transformer encoder. Anchor free method can predict objects directly without predefined box sizes (Ultralytics Inc, n.d.). CenterNet is a one-stage object detection model that predicts object centers rather than relying on anchor boxes or region proposals like anchor-based method such as Faster-RCNN (Zhou, Wang and Krähenbühl, 2019). This implies that, CenterNet is a specific anchor-free method that primarily utilized in this project. By utilizing this way in object detection, the computational cost will be decreased since does not required selection and tuning of anchor box sizes and ratios and the flexibility in handling irregular shape object will increase. In other words, anchor-free method helps to simplify the detection process and accelerate the inference process. This gives a big contribution on achieving real-time efficiency with a speed of 74.9 frames per second (FPS) in EALPR framework. The higher the FPS, the better the performance, in which outperforming traditional anchor-based methods that often sacrifice speed for accuracy.

For EfficientDet, it is a family of object detection models developed by Google based on EfficientNet. It relies on Bi-directional Feature Pyramid Network (BiFPN) for multi-scale feature fusion (MingxingTan, Ruoming Pang and Quoc V. Le, 2020). In simpler terms, EfficientNet can be apply in varies size of license plate in the captured image neither small, medium nor large. Based on Ferrer (2024), transformer encoder converts the input tokens into forms that are contextualized. Hence, it is useful when having the scenario that license plate only occupy a small portion of the image. As a result, the system can handle unrestricted distances between the vehicle and the capturing camera in vehicle plate detection.

Moreover, this EALPR also follow unified framework structure in which it combines vehicle and license plate detection into a single pipeline. The advantage of this arrangement is preventing the demand for separate networks and decreasing the I/O overhead. Shared feature maps were also being used among vehicle detection and vehicle plate detection modules in order to improve the efficiency. In this case, the unified framework structure helps with enhancing performance by minimizing the cost of computation.

Despite adopting an anchor-free approach, this framework has demonstrated through testing on numerous datasets with flying colour findings that it surpasses current method in terms of accuracy and speed. With 98.15% on OpenALPR(EU), 95.61% on OpenALPR(BR), 99.51% on AOLP(RP), 88.81% on SSIG, and 79.41% on CD-HARD, it reaches state-of-the-art accuracy.

Training data is one of the important procedures in vehicle plate recognition system. In this project, the model is trained with data augmentation strategies in order to handle different ranges such as distortion of perspective, changes of colour, changes in lighting and other condition that may affect the accuracy of recognised license plate in reality. Consequently, the robustness of the model may be proven.

To recognise the numbers and letters from the pre-processed license plates, the license plate recognition (LPR) employs a per-trained OCR-net. Because the distorted number plates had already been corrected during the earlier detection step, using this per-trained OCR-net made the recognition procedure become simpler. Convolutional Neural Networks (CNNs), which are employed in the EALPR framework, were utilized to extract sequential features from the full license plate region using segmentation-free techniques and OCR-net.

In addition, loss function is also used in the License Plate Detection (LPD) of this project. Loss function is a mathematical function that measures how well a machine learning model is performing (Dave and Cole, 2024). It helps to calculates the difference between the predicted output and the actual (ground truth) value. The loss function in this project was divided into three parts which included the probability error between the center of vehicle plate and prediction, license plate location and affine transformation parameters regression.

In a nutshell, this EALPR framework is efficient and effective when being utilized in unconstrained scenarios. It addresses the challenges of perspective distortion of vehicle plate, varies size of license plate portion in the image and real-time processing.

2.5.2 ALPR- An Intelligent Approach Towards Detection and Recognition of License Plates in Uncontrolled Environments

According to Bakshi et al. (2023), automatic license plate recognition (ALPR) systems are becoming vital tools for law enforcement, intelligent transportation and traffic management. In the past, these systems have performed best in controlled settings with frontal views of license plates and ideal lighting. However, as recent studies have shown, their performance drastically decreases when they are faced with the difficulties of uncontrolled surroundings that include complicated backgrounds, varying illumination, and oblique viewing angles. This drawback of traditional ALPR systems has generated a lot of interest in research into creating more reliable alternatives that can manage real-world situations.

In uncontrolled scenarios, ALPR systems face a variety of difficulties. Environmental elements that significantly increase noise and distortion in license plate photos include motion blur, poor lighting, and extreme weather. Furthermore, even if they work well under ideal circumstances, commercial systems like OpenALPR and Sighthound show significant performance declines in these uncontrolled situations. These challenges highlight the urgent need for increasingly complex methods that can preserve high accuracy in a variety of difficult real-world scenarios.

New developments in deep learning have opened the door for innovative solutions to these problems. YOLOv4's integration for license plate detection is a major advancement because it provides real-time processing capabilities together with excellent accuracy. By adapting the YOLOv4 architecture especially for license plate detection, researchers have improved this method a lot more. They have done this by optimizing factors like batch size and filter configurations to increase performance while lowering computational complexity. This customized version shows how general-purpose frameworks for object detection can be successfully adapted to particular recognition tasks.

Using Spatial Transformer Networks (STNs) to correct for geometric distortions is a very important advancement in ALPR systems. An complicated three-stage procedure is used by the STN module to generate sampling coordinates, estimate transformation parameters using a localization network, and produce the rectified output using a sampler. By correcting for perspective distortions and different orientations, this method successfully normalizes license plate photos and greatly increasing the accuracy of future character recognition. STNs' modular design preserves end-to-end trainability while enabling smooth incorporation into current deep learning pipelines.

Deep learning architecture advancements have also helped character identification in ALPR systems. In order to improve image quality before character segmentation, modern systems use extensive preprocessing pipelines that include morphological operations, adaptive thresholding, and grayscale conversion. To properly identify individual characters, the segmentation method itself combines sophisticated filtering based on dimensional limitations with shape analysis. In order to achieve exceptional accuracy rates above 96% on common benchmarks, researchers have created customized CNN architectures for the recognition phase that are modeled after VGGNet but tuned for license plate character recognition.

The advantage of these sophisticated ALPR systems over conventional methods has been proven via evaluation. Extensive testing on various datasets such as AOLP, SSIG and custom collections with difficult oblique views has consistently shown increases in speed. In challenging situations, it has been demonstrated that the incorporation of STNs alone increases recognition accuracy by about 13%. The benefits of these research-grade methods are demonstrated by comparisons with commercial systems especially when it comes to processing distorted and less-than-ideal license plate photos. With a mean average precision of 90%, the customized YOLOv4 detection component outperforms earlier iterations of the YOLO architecture.

Despite all of these improvements, there are still specific troubles with the ALPR systems in use today. The current limitation to Latin character sets and performance decrease in extremely low light and partial occlusions are still problems. Such limitations highlight crucial areas for further study such as creating more reliable preprocessing methods for dim lighting, integrating multimodal sensing strategies, and expanding recognition capabilities to different writing systems like Devanagari. In addition, investigating hybrid architectures that combine the advantages of various deep learning techniques could result in additional gains in computational efficiency and accuracy.

The development of ALPR systems is typical of more general patterns in pattern recognition and computer vision, where deep learning keeps pushing the envelope in difficult real-world applications. As these technologies advance, they should make it possible for ALPR solutions to become more dependable and adaptable. ALPR solutions shouls also able to function well in a wide range of environmental conditions found in real-world deployment scenarios. In addition to improving the state of ALPR technology, the current research in this area offers important new information to the broader field of object recognition in uncontrolled situations.

2.5.3 Vehicle color recognition based on smooth modulation neural network with multi-scale feature fusion

Based on Hu et al. (2023), in intelligent transportation systems, vehicle color recognition (VCR) has become a vital technology, especially when conventional license plate recognition techniques are ineffective because of obstruction, fraud, or low picture quality. Although traditional VCR systems usually only identify cars into 13 or fewer color categories, real-world applications are calling for more precise detection capabilities. Limited dataset diversity, class imbalance issues, and performance degradation under changing environmental conditions are only a few of the major obstacles facing current techniques. These flaws have prompted studies on more resilient VCR systems that can manage the complex nature of actual traffic situations.

The generation of extensive datasets is a key obstacle to the advancement of VCR technology. The scope of existing public datasets, such the C-, J-, and T-datasets, is constrained as they only cover a maximum of 13 color groups and frequently lack variation in terms of lighting circumstances, weather scenarios, and vehicle kinds. The creation and assessment of increasingly complex VCR algorithms are severely hampered by these constraints. In order to fill this gap, Hu et al. (2023) recently introduced Vehicle Color-24 which is a far more extensive dataset that includes 10,091 images taken from 100 hours of urban road surveillance video. In addition to extending the color taxonomy to 24 fine-grained groups, this dataset incorporates difficult real-world fluctuations in weather and illumination. The data's long-tail distribution which shows that rare colors like purple (0.04%) and pink (0.11%) are greatly underestimated while common colors like white (37.87%) and black (20.08%) are predominant. This offers both a difficulty and an opportunity for creating more reliable detection algorithms.

Innovative architectural solutions for VCR systems have been presented by researchers in order to address these issues. An important development in this field is the Smooth Modulation Neural Network with Multi-Scale Feature Fusion (SMNN-MSFF). A multi-scale feature fusion module that integrates both local geometric details and high-level semantic information using Feature Pyramid Networks (FPN), a lightweight 42-layer VCR-ResNet backbone optimized for color feature extraction, and a novel improved Smooth

L1 loss function (VCR-Loss) created primarily to reduce the effects of class imbalance are some of the major innovations combined in this approach. In contrast to conventional focus loss techniques, the VCR-Loss performs better in ablation studies (94.96% vs. 91.79% mAP) by dynamically reweighting losses for underrepresented classes using a parameter β (empirically adjusted at 0.11).

The outcomes of experiments show how effective these improvements are in a variety of ways. According to ablation studies, the VCR-ResNet backbone outperforms traditional architectures like as VGG16 that get mAP of 62.38% and ResNet50 which has mAP 65.38% with a mAP of 68.17%. While the VCR-Loss achieves an amazing 36.37% gain over baseline Faster R-CNN performance (94.96% vs. 58.59% mAP), the addition of multi-scale feature fusion yields a significant 15.79% boost in accuracy (74.38% mAP). Consistent performance advantages are demonstrated by comparative evaluations across various datasets. For example, the SMNN-MSFF achieved 94.96% mAP on 24-color recognition in which beating out YOLOv4's 62.77% and RetinaNet's 91.79%, 97.25% mAP on the 8-color C-dataset that has beated on Chen et al.'s 92.63%, and the J-dataset's 97.85% and T-dataset's 90.62%. Unexpectedly, the system can process images in real time at 1.021 seconds on CPU hardware which makes it feasible to use in real-world traffic monitoring situations.

Beyond technical measures, these developments have deeper implications. While the SMNN-MSFF architecture shows how careful consideration to both model architecture and training dynamics can generate significant increases in recognition accuracy, the Vehicle Color-24 dataset sets a new standard for assessing VCR systems. The VCR-Loss function's ability to resolve class imbalance without the need for elaborate sampling techniques provides insightful information for other computer vision jobs that deal with comparable data distribution issues. But there are still difficulties especially with identifying very uncommon hue classes and sustaining performance in difficult conditions. This has suggests crucial areas for further study.

There are a number of potential directions that future research in this area might explore. While hybrid model architectures may improve performance in difficult situations like occlusion or low-light environments, extending identification capabilities to incorporate multilingual or multispectral analysis could increase system adaptability. The class imbalance issue may be resolved by looking into self-supervised learning strategies, which might minimize the need for massive annotated datasets. VCR integration with other vehicle identification techniques is expected to become more and more significant in intelligent transportation systems, law enforcement applications, and urban planning projects as the technology develops. By showing how careful integration of dataset design, model architecture, and training methodology may advance the state of the art in vehicle recognition systems. In conclusion, SMNN-MSFF framework builds a solid foundation for these upcoming advancements.

2.5.4 DeepCar 5.0: Vehicle Make and Model Recognition Under Challenging Conditions

As stated in Amirkhani and Barshooi (2023), Vehicle Make and Model Recognition (VMMR) has become an essential part of Intelligent Transportation Systems (ITS) with applications ranging from driverless vehicles to traffic control, surveillance, and law enforcement. Despite its importance, VMMR has a number of shortcomings, primary among them being its fine-grained classification. Accurate recognition is challenging due to the low intra-class variance among similar models and the substantial inter-class variance among different vehicle models. Furthermore, the work is made more difficult by the absence of complete datasets and the dynamic situations that cars operate in such as changing lighting, obstacles, and poor weather. A unique multi-agent system (MAS) used together with ensemble learning approaches is one of the innovative solutions that recent suggested by researchers to address these problems. Through the identification and processing of important areas of interest (ROIs) such as headlights, grills, and bumpers, which contain unique characteristics for vehicle classification, this method focuses on front-view image analysis.

Traditional techniques for tracking and detecting vehicles have depended on hardware-based sensors such as radar and microwave systems. Even while these techniques work well, they are frequently expensive and susceptible to environmental factors. On the other hand, vision-based methods like Light Detection and Ranging (LIDAR), histogram of oriented gradients

(HOG), and closed-circuit television (CCTV) provide more flexibility and cost-effectiveness. These techniques use sophisticated image processing algorithms to track and identify cars in real time. Regarding Vehicle Type Recognition (VTR), researchers have investigated model-based methods using 3D computer-aided design (CAD) models as well as feature-based methods like Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF). Nevertheless, these techniques are frequently vulnerable to changes in lighting and noise which reduces their adaptability in practical situations.

The current methodologies in the field of VMMR can be broadly divided into three categories include hybrid, part-based, and comprehensive. While comprehensive approaches can process full vehicle pictures, they have trouble with partial obstacles and different points of view. On the other hand, part-based methods focus on particular areas such as grills and headlights can provide a higher level of detail but required exact localization. The goal of hybrid approaches is to incorporate the best features of each of them. For example, channel max pooling lowers computing complexity without affecting accuracy. With these developments, the lack of extensive, annotated datasets continues to be a barrier in the field.

In order to resolve this barrier, the DeepCar 5.0 dataset which includes 40,185 front and front three-quarter photos from 50 automakers and 480 vehicle classes was created to close this gap. It focuses on cars made between 2019 and 2022. This dataset is one of the most complete and current resources for VMMR research since it includes manual annotations of ROIs such as headlights, grills, and bumpers and comprehensive technical specifications like engine power and speed. DeepCar 5.0 provides more diversity and relevance for modern vehicle detection tasks than other datasets such as CompCars and CityFlow.

This study's proposed methodology brings about a number of significant advancements. Initially, an attention mechanism is used to pinpoint crucial areas like grills and headlights which are essential for differentiating between car types for feature extraction. The second is the implementation of a multi-agent system (MAS) in which a specialized convolutional neural network (CNN) processes each ROI as an independent agent. For example, Agent 2 (grill) analyzes mesh structures using the DexiNed edge detector, whereas Agent 1 (headlights) employs Canny edge detection to extract "identity watermarks." To improve robustness, Agent 4 (fog light/scoop) uses neural style transfer, whereas Agent 3 (bumper) uses Gabor filters to collect texture and curvature characteristics. An ensemble learning technique is used to aggregate the outputs from these agents. In particular, a blackboard system that enables majority voting for final classification is used.

The effectiveness of this method is shown by the experimental findings which show an accuracy of 96.72% with manual ROI detection and 92.14% with automatic detection using YOLOR. These numbers greatly above the accuracy of conventional single-network techniques like ResNet101, which only attains 58.91%. Additionally, the suggested method shows robustness in the face of challenging conditions such as partial occlusions and changing lighting. With a 99.01% accuracy rate, benchmarking on the CompCars dataset confirms its excellence.

In a nutshell, the combination of MAS with ensemble learning is a major development in VMMR which utilizing localized characteristics and group decision-making to improve robustness and accuracy. The launch of the DeepCar 5.0 dataset opens the way for further study by addressing the urgent requirement for current and varied training data. Expanding the framework to multi-view recognition and refining it for real-time deployment in dynamic contexts are two possible avenues for future research. In addition to advancing the state-of-the-art in VMMR, this work offers a flexible and scalable approach for practical ITS applications.

2.5.5 Algorithms and Techniques used in current vehicle recognition systems

Table 2. 4 : Comparison among different vehicle attributes recognition systems

Aspect	**Efficient License Plate Recognition (EALPR)** (Wei *et al.*, 2024)	**ALPR in Uncontrolled Environments** (Bakshi *et al.*, 2023b)	**Vehicle Color Recognition (VCR)** (Hu *et al.*, 2023)	**DeepCar 5.0 (VMMR)** (Amirkhani and Barshooi, 2023)

Focus	License plate recognition in unconstrained scenarios	License plate recognition in uncontrolled environments	Vehicle color recognition under varying environmental conditions.	Vehicle make and model recognition using multi-agent systems and ensemble learning.
Key Techniques	Anchor-free methods (CenterNet), EfficientDet, transformer encoder, unified framework.	YOLOv4, Spatial Transformer Networks (STNs), customized CNNs for character recognition.	Smooth Modulation Neural Network (SMNN), multi-scale feature fusion, VCR-Loss for class imbalance.	Multi-agent system (MAS), attention mechanisms, ensemble learning, DeepCar 5.0 dataset.
Strengths	High accuracy (up to 99.51%), real-time processing (74.9 FPS), handles perspective distortion.	Improved accuracy with STNs (13% increase), handles geometric distortions, real-time capabilities.	High mAP (94.96%), handles class imbalance, real-time performance on CPU.	High accuracy (96.72%), robust to occlusions and lighting changes, comprehensive dataset.
Limitations	Limited to Latin character sets, performance drops in	Challenges with partial occlusions and extremely low light.	Struggles with rare color classes, performance in extreme conditions.	Requires precise ROI localization, complex implementation.

	extreme conditions	

In terms of focus, each research investigation addresses a distinct vehicle recognition problem. ALPR targets uncontrolled circumstances whereas EALPR concentrates on license plate recognition in unrestricted situations. While DeepCar 5.0 addresses make and model recognition, VCR focuses on color recognition. For thorough vehicle identification, these variations emphasize the necessity of a single system that combines several recognition tasks.

For key techniques, anchor-free detection (EALPR), STNs (ALPR), SMNN (VCR), and MAS (DeepCar 5.0) are among the methods employed. Although each methodology works well for a particular goal, a multimodal strategy that combines these cutting-edge methods might capitalize on their advantages to tackle more general issues.

From perspective of their strengths, these systems' efficacy is demonstrated by their excellent precision and real-time capabilities. However because they are specialized, they are only good at what they do. These advantages could be combined in a multimodal system to provide reliable performance in a variety of situations.

Despite their advantages, every system has cons of its own. For example, they might have encountered difficulties when managing challenging circumstances or uncommon classes. These variations highlight the need for a more adaptable approach that can handle a greater variety of problems, like partial occlusions, dim lighting or a variety of vehicle kinds.

In a nutshell, a Multimodal Large Language Model-Based Vehicle Recognition for Vehicle Access Control System is necessary after reviewing on these different types of vehicle recognition systems. The importance of developing the proposed project including current systems only focus on a single feature like make and model, color or license plates. However, when these functions are integrated, a multimodal approach offers a comprehensive solution for vehicle identification, which is essential for access control systems that need high reliability and accuracy. Moreover, extreme situations are difficult for current systems to handle. To overcome these obstacles and guarantee reliable performance in real-world scenarios, a multimodal LLM-based system can provide contextual reasoning according to the image clarity but not just providing a fault that allow unauthorized bypass of vehicles that may affect in the aspect of security. Importantly, no research has been done that integrates make and model, color, and license plate recognition into a unified framework. By providing a unique solution that expands the capabilities of vehicle access control systems, this proposal has closes the gap.

2.6 Software Development Methodologies

Software systems can be planned, designed, developed, tested, and maintained using an organized framework that is provided by the Software Development Life Cycle (SDLC). In order to make sure that the development process is effective, methodical and in line with project objectives, choosing the right SDLC model is essential. This section analyzes current SDLC approaches, assessing their applicability for creating a multimodal vehicle detection system and related online and mobile applications as well as their advantages and disadvantages.

2.6.1 A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering

Based on Pargaonkar (2023), Software Development Life Cycle (SDLC) is a crucial framework of modern software engineering that offers structured procedures to guide the creation of software products of superior quality. Organizations are under growing pressure to choose SDLC models that strike a balance between efficiency, quality, and adaptability as market expectations and technological demands continue to change. By systematically comparing classic and modern SDLC methodologies, thorough analysis in this paper provides insightful information on this selection process with focus on its significance for software quality engineering. By connecting theoretical model features with real-world quality assurance issues, this study closes a significant gap in the literature and allows development teams and stakeholders to make better decisions.

The conventional Waterfall model and current Agile approaches are the two main SDLC paradigms that are thoroughly compared in this paper. The Waterfall model exhibits major benefits in organized project management and thorough documentation due to its linear, phase-gated methodology. Because of these characteristics, it is especially well-suited for projects with clear, consistent requirements, like safety-critical applications or systems for regulatory compliance. However, the study also identifies significant shortcomings of the Waterfall technique, such as its inherent resistance to evolving requirements and the high chance of identifying defects at a late stage. These results are consistent with previous criticisms by Shylesh (2017, cited in Pargaonkar, 2023, p. 121), who pointed out that Waterfall's applicability in dynamic development contexts is declining. On the other hand, Agile approaches show themselves to be a strong substitute for projects that need flexibility and quick iterations. The report outlines Agile's advantages in early issue identification, improved customer satisfaction, and continuous quality improvement through iterative cycles. Qualities that have made it a preferred approach for consumer-facing apps and quickly changing markets.

The research highlights significant trade-offs between these opposing strategies from the perspective of software quality engineering. Clear audit trails and compliance benefits are provided by waterfall's emphasis on comprehensive documentation and phase completion, which are crucial for industries with strict regulatory requirements. But because it is sequential, quality problems are frequently found too late in the development cycle. This may result in necessitating of expensive rework in order to get a good quality. Although agile's quality assurance procedures are more adaptable, they might be difficult to manage scope creep and maintain thorough documentation. The findings of Gurung et al. (2020, cited in Pargaonkar (2023)) about the documentation-quality paradox in iterative development methodologies are supported by these observations. By including real-world case studies that illustrate how these theoretical trade-offs appear in realistic development settings, the study further strengthens its practical relevance and offers insightful background information for organizational decision-making.

Although the study provides valuable information, it also points out fundamental flaws in the way the SDLC is being implemented. Although different aspects of Waterfall and Agile helpful, but both of them ignores the growing popularity of hybrid techniques that aim to integrate the best features of each. These blended models are mentioned in passing in the study, but they are not thoroughly examined. In other words, this is a topic that needs more research. More thorough analysis of how cutting-edge methodologies like DevOps and continuous integration/continuous delivery (CI/CD) pipelines might support or disagree with established SDLC quality assurance procedures would also be beneficial to the study. The study contributes significantly to industry practice and academic research by offering a clear framework for assessing SDLC models in relation to quality engineering goals regardless of these limitations.

The study offers a number of encouraging possibilities for further research in this area. The creation of increasingly complex hybrid approaches that purposefully blend the flexibility of Agile with the structure of Waterfall may assist organizations in producing higher-quality results for a variety of project kinds. While preserving the benefits of iteration, a deeper integration of automated testing and quality assurance tools into Agile workflows may assist solve the present documentation issues. The report also emphasizes the necessity of industry-specific SDLC modifications, especially in highly regulated fields like healthcare and finance where quality standards vary greatly from those of commercial software development. Together with the study's main conclusions, these future possibilities offer a strong basis for further investigation into how to best optimize SDLC procedures for software quality engineering in a technical environment that is becoming more and more challenging.

2.6.2 Agile Methodology Vs. Traditional Waterfall SDLC : A case study on Quality Assurance process in Software Industry

According to (Sinha and Das, 2021), a quick, iterative, and flexible approach to software development known as Agile methodology that was created to handle short development cycles and shifting requirements. On the other hand, the traditional Waterfall approach adheres to a strict, sequential procedure which makes it less appropriate for dynamic project contexts where adaptability is essential. Agile has become widely used in software engineering as well as other industries like manufacturing because of its capacity to combine incremental improvements with ongoing feedback. Comparing Agile and Waterfall testing methodologies is a major area of research in this field, with studies emphasizing Agile's greater efficiency and effectiveness for modern software development.

In 1970, Winston Royce established the Waterfall model that divides software development into the following discrete, sequential stages: requirements, design, implementation, verification, deployment, and maintenance. This model's testing is done only after the implementation stage, which frequently results in a delayed discovery of defects and increased corrective expenses. Although the Waterfall technique has benefits like organized documentation, manageability, and fit for projects with clear goals but there are serious drawbacks to its inflexibility. Its relevance in rapidly changing project environments is further limited by late-stage defect identification and little developer-tester collaboration.

In contrast, the agile methodology uses an iterative structure called sprints, which are development cycles that include phases for requirement gathering, implementation, and testing. This strategy lowers market risks and guarantees alignment with user needs by emphasizing gradually delivery, flexibility, and ongoing customer feedback. Agile's ability to integrate testing into each sprint, which facilitates early problem identification and resolution is a key benefit. Agile encourages continuous cooperation between developers and testers, reducing knowledge gaps and speeding up issue resolution in contrast to Waterfall, which views testing as a distinct and last process.

The distinctions between Waterfall and Agile are even more visible when looking at testing procedures. Inefficiencies result from waterfall testing's sequential nature, heavy reliance on documentation such as test plans and completion reports, and separation from development. On the other hand, Agile testing is continuing, necessitates little documentation and benefits greatly from cross-disciplinary cooperation. Bugs are fixed in later rounds, avoiding accumulation and guaranteeing more seamless project development. The advantages of Agile testing including faster feedback loops, increased flexibility, and improved client satisfaction as a result of frequent demonstrations and iterative improvements.

The benefits of Agile are further shown by comparisons between Agile and Waterfall testing. The modular methodology of Agile that also known as "divide and conquer" enables targeted testing on smaller code portions, increasing productivity and accuracy. In this case, changes are easily implemented without interfering with the project, but Waterfall's rigidity makes these adjustments expensive and time-consuming. Furthermore, Waterfall's late-stage testing frequently causes delays and budget overruns, whereas Agile's set sprint lengths produce more accurate time and cost estimates. While Waterfall's restricted stakeholder contact might result in misaligned outcomes, Agile's constant engagement and iterative delivery also increase customer satisfaction.

In academic environments, where previous approaches like Waterfall have traditionally predominated, future research topics will examine the applicability of Agile. Agile's advantages in a variety of contexts might be further validated by including it into educational programs. This could yield insightful information about how effective it is in comparison to traditional methods. Studies already conducted support Agile's benefits in terms of quality and productivity, making it the go-to approach for existing software development.

In conclusion, this research paper shows how Agile is better at encouraging flexibility, teamwork, and productivity—especially in testing procedures. Agile's iterative and customer-centric methodology makes it more appropriate for dynamic and changing software development environments even while Waterfall is still useful for stable, clearly defined projects. The scalability and efficacy of Agile should be further investigated in future research in a variety of fields including academics and non-IT sectors.

2.6.3 Traditional SDLC Vs Scrum Methodology – A Comparative Study

Based on (Mahalakshmi and Sundararajan, 2008), the core of software engineering is the Software Development Life Cycle (SDLC) which includes crucial stages like planning, analysis, design, and execution. The industry has historically been ruled by traditional SDLC models, such as Waterfall, Spiral, and V-model which use methodical, structured procedures. However, serious flaws in these conventional frameworks have been revealed by the growing complexity nowadays software projects and the quickly shifting needs of customers. As a result, agile approaches such as Scrum have become adaptable substitutes. Scrum can provide iterative development cycles and flexible procedures more appropriate for the ever-changing software development environment of today.

One of the most historic and best-known approaches to the Software Development Life Cycle (SDLC), the Waterfall model proceeds in a strictly linear fashion through the following phases: requirements gathering, system design, implementation, testing, and maintenance. The inflexible structure of this framework has a number of significant disadvantages, the primary among them being its incapacity to adapt to evolving needs after development has started. Waterfall's sequential structure frequently results in late-stage defect detection, raising expenses and delaying projects. Furthermore, final solutions often fall short of consumer expectations due to a lack of iterative customer input. Although threre are some limitations, w aterfall still has certain benefits in spite of these difficulties, such as easy implementation, low resource needs, and thorough documentation that offers clarity all the way through the development process.

On the additional hand, the Scrum methodology is an alteration in the methods used for software development. Scrum is a popular Agile framework that divides work into time-boxed iterations called sprints. These sprints usually span two to four weeks and produce incremental changes to the final product. The approach identifies three primary roles: the self-organizing Development Team which is in charge of producing functional software, the Scrum Master who streamlines the process and eliminates roadblocks and the Product Owner who ranks requirements in the product backlog. Scrum uses a number of artifacts such as the sprint and product backlogs and burndown charts that show progress visually in order to keep things transparent and focused. Throughout the development cycle, regular practices like sprint planning, daily stand-ups, sprint reviews, and retrospectives ensure continuous collaboration and progress.

There are many valuable benefits that Scrum offers over traditional Waterfall. In today's rapid development settings, Scrum's iterative structure enables regular adaption to changing needs. The framework's frequent meetings and visible artifacts encourage improved cooperation and openness between team members and stakeholders. Most significantly, Scrum's focus on producing functional software at the conclusion of each sprint results in a quicker time to market and satisfied clients. Scrum implementation does have some disadvantages such as the requirement for highly dedicated and well-coordinated team members and the possibility of ambiguity due to the lack of documentation in compared with Waterfall methods.

A thorough analysis of various approaches shows significant variations in a number of areas. For projects that need flexibility, Waterfall's inflexible structure is insufficient whereas Scrum is excellent at adapting to shifting needs. Waterfall places a strong emphasis on thorough documentation whereas Scrum prioritizes functional software over plenty of paperwork. The two methods' approaches to customer interaction differ significantly. For Scrum, it includes ongoing stakeholder input throughout development while for Waterfall it restricts feedback to the very end. Scrum's iterative validation procedure which finds and fixes problems early tends to have higher success rates than Waterfall's late-stage testing methodology. Additionally, there are vital differences in teamwork between Scrum's collaborative, cross-functional teams and Waterfall's organized roles.

The analysis's findings clearly establish Scrum as the best option for dynamic projects with changing requirements since it provides quicker delivery timeframes and better stakeholder alignment. For projects with set, clearly defined objectives and needs that are unlikely to change, waterfall methodology is still suitable. In the end, the features of the project will determine which approach is best. Waterfall is better for stable, predictable projects while Scrum is better for agile development settings. Future studies could look into hybrid models that combine the flexibility of Scrum with the structure of Waterfall, as well as how scalable Scrum is for large-scale enterprise projects. Understanding these methodological variations is becoming more and more important for organizations looking to maximize value for stakeholders and optimize their development processes as the software development environment evolves.

2.6.4 Waterfall Vs V-Model Vs Agile: A Comparative Study on SDLC

As stated in (Murugaiyan, 2012), software engineering projects are based on the Software Development Life Cycle (SDLC) which provides standardized methods to direct development from idea to implementation. The industry has long been dominated by traditional SDLC models, especially the Waterfall and V-Model approaches, which use phase-based, sequential techniques. But the advent of Agile approaches has brought in more adaptable, iterative alternatives that are more suited to the ever-changing needs of today's software. In order to assist organizations in choosing a development method, this literature analysis compares the features, benefits, drawbacks, and best use cases of these three well-known SDLC models.

Firstly, the Waterfall model is a model that follows a strictly linear sequence of phases which includes requirements gathering, system design, implementation, testing, and maintenance. This model is the most conventional method to the Software Development Life Cycle (SDLC). The main advantage of this model is its simplicity and clarity. By employing this model, teams can go forward with less uncertainty if clear requirements are set up at the beginning. Strong traceability is ensured by the thorough documentation created during Waterfall development which also makes the approach ideal for projects with consistent, unchanging needs. However, in modern development environments, the Waterfall technique exhibits obvious limits. Once development has started, its rigid structure makes it very difficult to accommodate requirement changes. As a result, it frequently requiring expensive rework.Another important point is testing only takes place later in the development process which may cause flaws to go unnoticed until they are costly to fix. Furthermore, even while final goods technically satisfy initial specifications, they might not entirely correspond with user needs due to a lack of iterative client feedback.

When it comes to validation and verification, the V-Model offers a more advanced method than the strictly sequential Waterfall model. This methodology forms the distinctive "V" shape by explicitly relating each development phase to its associated testing activity while maintaining the Waterfall methodology's phase-based structure. The main benefit of the V-Model over Waterfall is that it incorporates testing considerations from the very beginning of development. This has allow it for earlier problem discovery. The V-Model allows for requirement modifications at any stage which gives it a little more flexibility than Waterfall while still upholding strict documentation requirements. However, any changes still necessitate significant documentation updates. The approach is especially useful for complicated, high-stakes projects where quality assurance is crucial since it places a strong emphasis on validation at every stage. However, due to the heavy overhead of maintaining parallel development and testing artifacts, the V-Model's extensive review requirements make it less appropriate for smaller projects or those with strict deadlines.

In contrast to these conventional methods, agile techniques emphasize iterative development, ongoing customer collaboration and flexible planning. Agile divides projects into brief development cycles that each of them usually take a durations of 2-4 weeks called sprints. They helps to produce incremental product changes in comparison to the rigid architecture of Waterfall or V-Model. In the fast-paced development environments of today, this method offers many benefits. Most significantly, teams can easily adapt to shifting needs as the project progresses via Agile's adaptability. By offering frequent chances for feedback and course correction, frequent delivery of functional software improves customer satisfaction. Moreover, misunderstandings are decreased and information sharing is encouraged by the methodology's emphasis on in-person interactions and cross-functional teamwork. But there are drawbacks to Agile implementations as well, especially for larger projects where it can be challenging to estimate the amount of work needed. The approach's dependence on self-organizing teams and scant documentation necessitates highly qualified engineers and may lead to problems with knowledge transfer. Additionally, teams that are spread out geographically may find the intensive cooperation approach difficult to implement.

When these approaches are examined, key distinctions are found in a number of important areas. The V-Model permits controlled modifications with effort, Waterfall is totally inflexible and Agile welcomes change at every stage of development. Testing methodologies vary greatly as Agile include continuous testing across iterations, Waterfall focuses testing at the conclusion while the V-Model pairs testing with each development phase. Documentation strategies vary from the lengthy documentation of Waterfall and V-Model to Agile's preference for functional software over extensive documentation. The range of customer interaction is comparable, ranging from Waterfall's restricted final-stage input to Agile's continuous cooperation. The V-Model occupies a middle ground of structured collaboration as team structures evolve from Waterfall's compartmentalized specialists to Agile's cross-functional generalists.

In final terms, particular project characteristics and organizational requirements will determine which of these SDLC models is best. Agile approaches work best in dynamic settings where customer collaboration is valued and requirements change regularly. For large-scale projects with consistent, clearly specified requirements where thorough documentation is crucial, waterfall is still suitable. For complicated projects that need strict validation procedures, the V-Model provides a well-rounded strategy that allows for some flexibility in response to changing requirements. In addition to studies into scaling Agile methods for enterprise-level implementations, future research lines could effectively investigate hybrid models that combine the flexibility of Agile with the structure of older methodologies. Understanding these methodological variations is becoming more and more important for organizations looking to maximize value for stakeholders and optimize their development processes as software development continues to change.

2.6.5 Comparative Analysis on different Software Development Lifecycle methodologies

The successful of a software project is highly dependent on the appropriateness of approach and clearliness of the outlined processes. Every approach framework brings advantages and disadvantages.

Table 2. 5 : Table of comparison between various software methodologies.

Aspect	Waterfall	V-Model	Agile	Scrum
Requirement Flexibility	Rigid	Moderate	High	High
Suitable Project Size	Large	Medium to large	Large	Small to medium
Testing	Late-stage testing	Parallel testing for each phase	Continuous testing in iterations	Continuous testing in sprint
Documentation	Extensive	Extensive	Minimal	Minimal

Customer Feedback	Limited to final stages	Periodic during validation	Continuous and iterative	Continuous through sprints
Team Dynamics	Organized roles	Collaborative but structured	Cross-functional and self-organizing	Collaborative and cross-functional

According to the table above, Waterfall, V-Model, Agile and Scrum are all software development methodologies with distinct approaches, advantages, and disadvantages.

From perspective of requirement flexibility, Agile and Scrum is well-suited for projects with evolving or unclear requirements due to its iterative structure. The waterfall is suitable for projects with stable and well-defined requirements as it follows a rigid, predetermined sequence of phases. V-Model be effective for projects with changing requirements but require rework effort if changes happened. Agile and Scrum is highly flexible as both of them allowing for changes and feedback from customer throughout development. Waterfall is inflexible and only allows changes occured in the planning phase. V-Model is more flexible than a waterfall but less flexible than an Agile.

When considering project size, Waterfall works best on large projects with clearly defined needs that are unlikely to alter. Medium-sized to large projects are best suited for the V-Model especially when system safety and dependability are essential. Although Agile is especially useful in situations where change is expected but it can also manage big projects. Scaled Agile Framework (SAFe) is a framework designed to scale Agile team practices all the way up to the corporate level (Paula, n.d.). Large-Scale Scrum (LeSS) is a framework that allows procedures and methods to be modified to fit the demands of the specific circumstance (Larman and Vodde, 2016). By utilizing SAFe and LeSS frameworks, Scrum is allowed to be scaled for larger projects, however it is most effective for small to medium-sized projects.

From the perspective of testing, Waterfall delays the discovery of bugs because testing takes place in the last stages after the entire development process is finished. This is enhanced by the V-Model, which ensures early issue detection by testing concurrently with each development phase. Continuous testing is used in agile iterations to identify issues early and gradually raise quality. Continuous testing is also supported by Scrum which incorporates it into every Sprint to make sure that increments are actually finished and maybe even shippable.

Based on the documentation perspective, Waterfall requires thorough documentation at every turn in order to confirm that everything can be tracked down and officially approved. In contrast, the V-Model places a strong emphasis on thorough documentation to support its strict validation and verification procedure. However, Agile reduces documentation and emphasizing functional software over extensive documentation. This minimum documentation technique is also used by Scrum. It promotes the creation of just necessary documentation to enable development without slowing team progress.

As look into the customer feedback point of view, Waterfall typically only allows for customer input at the very end which is when the product is almost finished. This makes it expensive to make modifications as required late-stage of changes on the product. The V-Model does not have regular interaction, but it does introduce periodic feedback during validation phases. Agile encourages iterative and ongoing input and including clients at every stage of the development process. By showcasing potentially shippable products at the conclusion of each Sprint, Scrum also guarantees ongoing user feedback by enabling frequent and early modifications.

From the perspective of collaboration, Waterfall projects usually consist of well-organized, role-specific teams with minimal responsibility overlap. The V-Model keeps things organized but allows for a little more cooperation particularly between the testing and development teams. Agile teams are self-organizing and cross-functional which means that members share tasks and oversee their work as a group. In order to accomplish the greatest outcomes, Scrum teams are cross-functional and collaborative, prioritize regular communication, shared accountability, and teamwork within each sprint.

In conclusion, the Agile methodology was selected for this project rather than the Waterfall, V-model, or Scrum models because it permits small adjustments even when a portion has been completed ahead of schedule, which is advantageous in a dynamic project setting. Because of its iterative nature, emphasis on iterative testing and improvement, and prioritizing of important needs, the Agile methodology was considered the best fit for this project.

2.6.6 Web and Mobile Application Framework

Choosing the right web and mobile application framework is essential for modern software development in order to guarantee effectiveness, scalability, and maintainability. Frameworks can help to simplify the development process, minimize duplicate coding and improve overall application performance by providing standardized environments, built-in functionalities and standard procedures. Cross-platform frameworks are gaining a lot of attention due to the increasing need for programs to run smoothly across several platforms such as web browsers and mobile devices. This section examines a number of frameworks that facilitate the creation of online and mobile applications and emphasizing their salient features, benefits and applicability for developing a multimodal access control and vehicle identification system.

2.6.6.1 Frontend-Framework

With the aid of innovative frontend frameworks, the effectiveness to design progressive web apps (PWAs) and cross-platform mobile applications can be increase gradually. These frameworks ensure great performance, versatility and an improved user experience while streamlining the development process with their standardized architectures and robust libraries. The results of the comparison between Flutter and React Native are given in the table below.

Table 2. 6 : Comparative table of React Native and Flutter frameworks

Aspect	React Native	Flutter
Programming Language	JavaScript (with JSX)	Dart
Web Support	Web support via additional libraries like React Native Web	Native web support integrated with Flutter Web
Learning Curve	Moderate	Steeper

Performance	Near-native performance	High performance
UI Components	Relies on native components	Rich built-in UI widgets for consistent look across platforms
PWA Readiness	Requires careful integration and setup	Flutter Web supports PWA development natively with easier configuration
Best for	Applications needing tight native integration and access to a wide ecosystem	Applications needing consistent UI across platforms and easy web and mobile deployment

React Native and Flutter develop web and cross-platform applications using various programming languages. JavaScript and JavaScript XML (JSX) which are well-known to many developers and facilitate adoption especially for those with previous web development experience are used by React Native. In contrast, Flutter uses Dart which is a Google language that is powerful but less well-known and necessitates learning new syntax and concepts before completely productive.

Regarding web support, React Native makes it possible to construct websites using extra libraries like React Native Web which renders components in a web environment. However, this setting may require further setup because it is not built into the main framework. However, Flutter has integrated web support through Flutter Web, which makes it possible for developers to target browsers directly without the use of third-party frameworks. This makes the development process simpler.

Additionally, these frameworks have compared to different learning curves. For developers who are already familiar with JavaScript, React Native offers a moderate learning curve because its component-based architecture allows for easy adaptation. Because it requires learning Dart and adjusting to its widget-driven programming paradigm which differs substantially from conventional web development frameworks, Flutter has a higher learning curve as compared to React Native.

In terms of performance, Flutter definitely preferable. Flutter apps offer seamless rendering and fast speed by compiling directly to WebAssembly for the web and native ARM code for mobile devices. Although React Native achieves speed that is close to native, it depends on a bridge between JavaScript and native modules which sometimes results in minor performance overheads especially in apps that are resource-intensive or complex.

In order to adapt mobile components for the web by employing Progressive Web App (PWA), React Native needs careful setup and dependence on React Native Web which may increase development complexity when considering Progressive Web App (PWA) readiness. Configuring and deploying cross-platform solutions including browsers, is made simpler and faster with Flutter's native support for web and PWA development through Flutter Web.

In general, React Native is a great option for applications that need close native integration and access to a large and developed ecosystem. It has giving it a great way to make use of third-party resources and pre-existing JavaScript expertise. However, Flutter is best suited for applications that prioritize platform consistency and a smooth web and mobile deployment particularly when aiming for a single codebase strategy for PWAs and mobile apps.

In conclusion, React Native has been chosen as the frontend framework for this project because it complies with project specifications and because previous JavaScript and React Native development experience is available. Even though Flutter provides more efficient and integrated support for Progressive Web Application (PWA) development through Flutter Web, React Native is still an acceptable choice because of its well-established ecosystem and robust support for cross-platform development which including web integration through React Native Web . React Native is ideally suited for implementing the multimodal big language model-based vehicle detection system suggested in this study because of these features which also make development and maintenance easier.

2.6.6.2 Backend-Framework

With the help of back-end frameworks, the server-side of web apps, APIs and other software systems can be built. By offering a structure for creating a web application's back end, these frameworks prevent worry about technical programming details and let them concentrate on implementing business logic. The table below shows the results of the comparison between Laravel and Next.js.

Table 2. 7 : Comparative table of Laravel and Next.js frameworks

Aspect	Laravel	Next.js
Architecture Pattern	MVC (Model-View-Controller) with built-in advanced features	Hybrid React-based framework for modern dynamic applications
Learning Curve	Moderate to High	Easy for developers familiar with JavaScript/React
Performance	Slower than CodeIgniter	Highly optimized with SSR and SSG, enabling fast rendering and improved SEO
Built-in Features	Authentication, ORM (Eloquent), Queue, Events, Jobs, API resources, etc.	Routing, API routes, middleware, image optimization, ISR
Flexibility	High	Very flexible as can integrate serverless functions, Node.js APIs, and React libraries
Security	High (CSRF, XSS protection, password hashing, etc. by default)	Basic, need manual implement security measures
Database Handling	Eloquent ORM (advanced, object-oriented)	Works with Prisma, Sequelize or direct database queries

Template Engine	Blade Template Engine	React with JSX (Component-based UI rendering)
Updates and Modern PHP Compatibility	Frequent updates	Regularly updated in line with React and Node.js ecosystem
Deployment	Traditional web servers, Docker, cloud platforms	Vercel, Netlify, AWS, or any Node.js hosting; which optimized for serverless deployment
Best for	Large-scale, enterprise-level, feature-rich applications	Modern, high-performance, SEO-friendly, interactive web applications

The classic Model-View-Controller (MVC) architecture, which Laravel adheres to, neatly divides data management, presentation, and business logic. It has many built-in features and is best suited for full-stack, backend-heavy applications. Next. In contrast, js is a hybrid framework based on React that supports client-side rendering, server-side rendering, and static site generation. Modern online applications with dynamic content benefit greatly from this flexibility, which enables developers to select the rendering approach that best suits the application's performance and SEO needs.

From perspective of learning curve, Laravel's rich built-in features, like Eloquent ORM, queues, events, and Blade templates, make it difficult for developers who are not familiar with PHP or backend frameworks to understand. Next. For developers who are already familiar with JavaScript and React, learning js is rather simple. Without having to learn intricate backend patterns, it is simpler to begin developing both frontend and server-side functionality thanks to its component-based architecture and comprehensive documentation.

Performance is another important aspect that helps to defferentiate between this two frameworks. Laravel is strong for backend-heavy tasks, but because it uses PHP to handle each request, it may render simple websites more slowly than lightweight frameworks. In contrast to Laravel, Next.js is highly optimized for performance, offering SSR and SSG, which reduces page load times and improves perceived performance. Its frontend-heavy architecture allows fast rendering of UI components and better handling of high-traffic scenarios without extensive backend overhead.

When considering to the built-in features, many capabilities that are useful for enterprise-level applications are incorporated into Laravel, including queues, tasks, events, ORM (Eloquent), authentication, and API resources. Routing, API routes, image optimization, incremental static regeneration, and middleware support are just a few of the contemporary web development capabilities that Next.js provides. Although it comes with fewer backend tools, it easily interfaces with external APIs or Node.js modules to provide comparable functionality in a more modular manner.

For flexibility, because of Laravel's great flexibility for full-stack development, programmers may create complex apps wholly within its ecosystem. Although it prioritizes frontend and modern web experiences, Next.js is still very adaptable. It may be expanded by developers utilizing serverless functions, React libraries, and Node.js APIs, allowing for the modular and scalable customization of frontend and backend logic.

The variations between the two frameworks are further emphasized by security considerations. By default, Laravel offers strong security, which includes password hashing, XSS prevention, CSRF protection, and other security features. Developers are need to provide secure authentication, authorization, and input validation procedures because Next.js does not by default incorporate backend security. Nonetheless, developers may maintain high security using best practices thanks to its interaction with contemporary authentication libraries (such as NextAuth.js) and safe API methods.

In terms of database management, complex relationships and queries are easier to manage using Laravel's Eloquent ORM. By utilizing Eloquent ORM, it offers an object-oriented, user-friendly approach to database interactions. Although Next.js lacks built-in database management, developers may simply combine it with ORMs such as Prisma, Sequelize, or direct database queries, allowing them to select the best database solution for their requirements.

The frameworks are additionally distinguished by the templating system. Blade is the template engine used by Laravel, which makes it easier to produce server-side HTML with dynamic data. Next.js uses React components and JSX which enabling highly interactive and reusable UI elements. This approach is more suitable for modern web applications where dynamic, client-side interactions and real-time updates are essential.

In terms of updates and compatibility, Laravel is greatly favored by its regular upgrades and compatibility with contemporary PHP standards. Laravel is updated frequently to take advantage of PHP developments, security enhancements and best practices. For Next.js, it is regularly updated to conform to the most recent React and Node.js standards, giving users access to cutting-edge frontend technologies, performance boosts, and ecosystem improvements.

Last but not least, traditional web servers, Docker containers, or cloud platforms like AWS, DigitalOcean, or Heroku can all be used to host Laravel apps. For Next.js, its serverless deployment optimization, which can be used on Vercel which is its native platform, Netlify, or any Node.js hosting environment makes deployment become easier and more scalable for modern web projects.

In conclusion, although Laravel is a great choice for enterprise-level, backend-heavy apps with a wealth of built-in capabilities, Next.js provides more frontend flexibility, better speed for contemporary online applications, and simpler integration with serverless and static hosting. It is more suited for quick, scalable, and interactive web apps because of its modular ecosystem, React-based user interface, and hybrid rendering capabilities. Next.js is the suggested option for creating a cutting-edge, high-performance online application when taking these benefits into account.

2.6.6.3 Database Configuration

Database management systems (DBMS) enable applications to effectively store, retrieve, and manage data in a consistent and organized way. In order to concentrate on creating application functionality rather than managing low-level data storage processes, these systems offer a basis for organizing and accessing data. Data integrity, security, and performance are guaranteed by a properly chosen database system particularly in applications that use multimedia data like photographs. Two popular relational databases which include PostgreSQL and MySQL are contrasted in the table below.

Table 2. 8 : Comparison table among MySQL and PostgreSQL

Aspect	MySQL	PostgreSQL
Performance	Fast read operations, widely used for web apps	Strong for complex queries, slightly heavier on resources
Image Storage	Supports BLOB for storing binary data	Supports BYTEA and large object storage
Data Integrity	Basic constraints and transaction support	Advanced constraints, ACID-compliant, full transaction support
JSON & Unstructured Data	Basic JSON support	Full JSONB support with indexing
Query Features	Simpler SQL features	Advanced SQL features
Extensibility	Less extensible, fewer custom data types	Highly extensible, supports custom types & functions
Compatibility with Next.js	Excellent, widely supported through Prisma, Sequelize, Knex, or raw queries	Excellent, fully supported with ORMs and works well with APIs and analytics
Security	Good, basic access controls	Strong, includes row-level security and audit features
Best Use Case	Web applications with simple to moderate complexity	Applications requiring complex data relations and analytics

From the prespective of performance, MySQL is a popular choice for online applications that require a lot of read operations and basic queries because of its well-known fast read speed. Because of its optimized performance in managing these processes, it performs exceptionally well in systems that adhere to the traditional Create-Read-Update-Delete (CRUD) structure. Although PostgreSQL requires a little more resources, it performs better than MySQL when managing complicated queries and multiple concurrent transactions. In circumstances where relational complexity, analytical workloads and the blending of structured and unstructured data are common, it works very well. PostgreSQL is a better option for systems needing sophisticated querying and high transaction rates because it exhibits higher efficiency and dependability while handling high transactional loads.

Regarding to image storage, it can be stored in both MySQL and PostgreSQL. For this purpose, MySQL uses the BLOB (Binary Large Object) data type whereas PostgreSQL uses BYTEA or Large Objects (LOB). Nevertheless, PostgreSQL provides more flexibility and performance when handling huge binary objects especially when it comes to visual content. One of PostgreSQL's main advantages in this regard is its support for streaming large files which greatly improves its capacity to handle big datasets. To give an illustration,like the visual data from vehicles used in recognition systems. This capability gives PostgreSQL a clear advantage over MySQL in situations involving big image storage. This can make it ideal for managing media-rich applications that demand reliable file management and effective retrieval.

In terms of data integrity and query features, PostgreSQL is well known for its strict adherence to the Atomicity, Consistency, Isolation and Durability (ACID) principles. In order to guarantee consistency in systems with complicated relationship, it offers extensive support for foreign keys, joins, complex restrictions, and transactional controls. Furthermore, PostgreSQL includes advanced SQL capabilities that are essential for executing complex queries in complex structures such as window functions, recursive queries and Common Table Expressions (CTEs). Although MySQL can enforce basic data integrity, but it lacks some of PostgreSQL's more advanced SQL features and frequently chooses to utilize simpler queries which compromise flexibility for usability.

By comparing the extensibility, PostgreSQL is a big plus in situations that call for sophisticated data handling and customization. It enables specify of unique data types, functions and even modules to expand the fundamental capabilities of the database. Additionally, PostgreSQL provides strong indexing and JSON and JSONB data type compatibility which is very useful when managing multimodal data inputs such integrated image-text data in recognition systems. Because PostgreSQL has more sophisticated indexing and manipulation features than MySQL, it is a better option for projects involving complicated data kinds and structures as MySQL only supports basic JSON.

For security, PostgreSQL is better suited for applications needing strict access control because of its robust built-in security capabilities which include role-based authentication, SSL encryption and row-level protection. In vehicle access control systems where access logs and image data may be sensitive, this is especially crucial. Although MySQL provides basic security safeguards, PostgreSQL's more sophisticated solutions enable more robust data protection.

Conclusion, PostgreSQL is the recommended database solution due to the unique needs of the multimodal large language model-based vehicle recognition system including the necessity to store vehicle images, preserve data consistency and manage extensive query patterns for multimodal data. It is the best option for handling the complicated data structures included in this system because of its strong security features and robust querying capabilities. Furthermore, PostgreSQL's smooth connection with Next.js via ORMs like Prisma or Sequelize guarantees effective server-side system integration and enabling scalability and excellent performance while handling large datasets.

2.7 Chapter Summary

In conclusion, this literature review covers at important research topics that are necessary to create a multimodal vehicle recognition system driven by AI. While pointing out the present shortcomings in visual data integration, the analysis discusses Large Language Models (LLMs) such as GPT-4 and LLaMA-Adapter V2, showcasing their promise for contextual reasoning in security applications. The evaluation of current segmentation methods and vehicle recognition systems like SegementAnything, YOLOv8 and Fast-SCNN reveals accuracy gaps in difficult circumstances like dim lighting or obscured views. Market access control application are also evaluated in the review in order to understand they basic features for defining the requiremennts for this project.

This study offers major innovation by combining visual identification with contextual analysis powered by LLM, in contrast to strict, rule-based commercial systems. In order to support the system's AI components and iterative improvement requirements, the chapter ends by defending the choice of Agile development approaches and certain technical frameworks like React Native, Next.js and so on. The development frameworks and methodology used in later chapters are directly influenced by these findings.

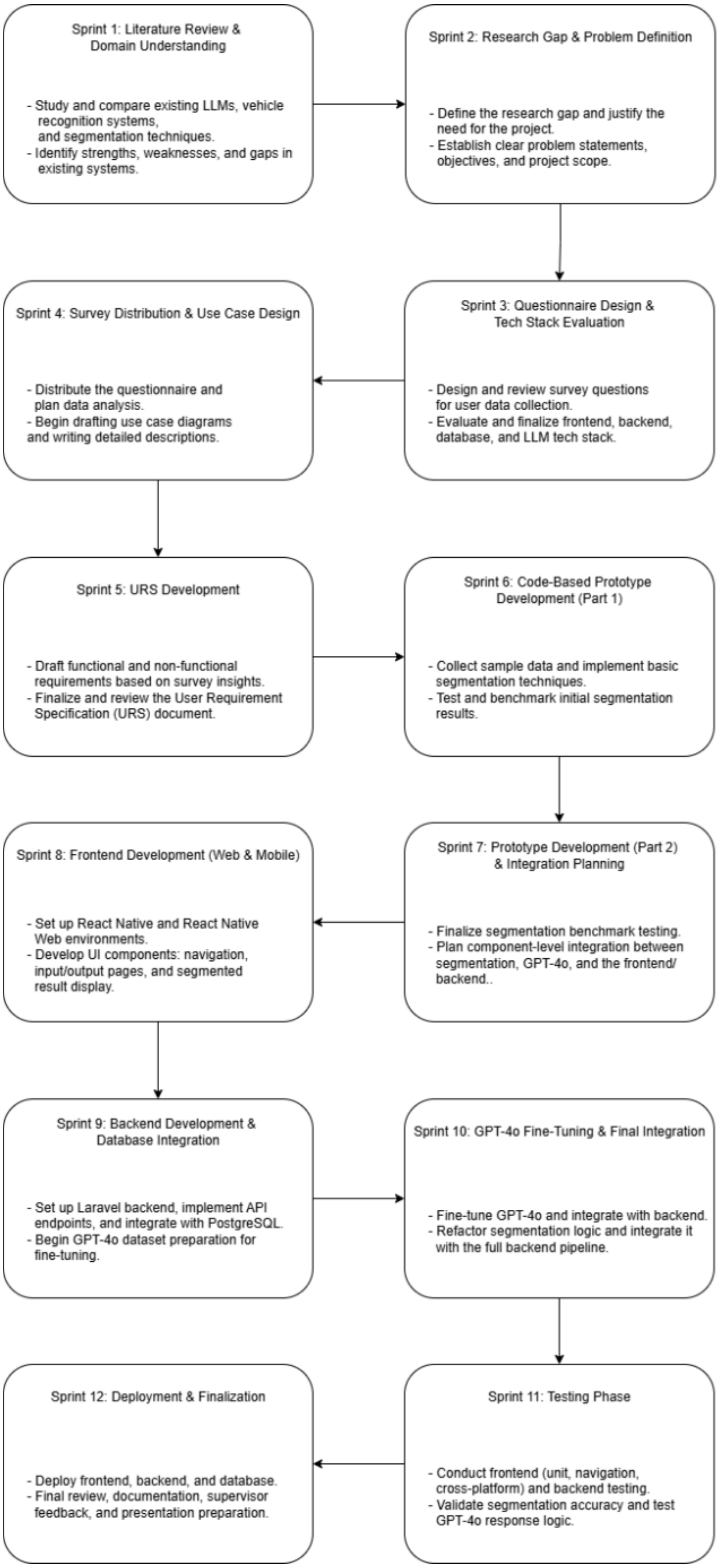
CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter discussed the project's work strategy and methods. The 6 phases with 12 sprints of the Agile approach that was selected were thoroughly explained in this chapter. Additionally, to schedule projects, the Gantt chart and the work breakdown structure, or WBS, were developed. The selection and discussion of the development tools came to a close.

3.2 Project Methodology and System Development Methodology



Sprint 1: Literature Review & Domain Understanding

- Study and compare existing LLMs, vehicle recognition systems, and segmentation techniques.
- Identify strengths, weaknesses, and gaps in existing systems.

Sprint 2: Research Gap & Problem Definition

- Define the research gap and justify the need for the project.
- Establish clear problem statements, objectives, and project scope.

Sprint 3: Questionnaire Design & Tech Stack Evaluation

- Design and review survey questions for user data collection.
- Evaluate and finalize frontend, backend, database, and LLM tech stack.

Sprint 4: Survey Distribution & Use Case Design

- Distribute the questionnaire and plan data analysis.
- Begin drafting use case diagrams and writing detailed descriptions.

Sprint 5: URS Development

- Draft functional and non-functional requirements based on survey insights.
- Finalize and review the User Requirement Specification (URS) document.

Sprint 6: Code-Based Prototype Development (Part 1)

- Collect sample data and implement basic segmentation techniques.
- Test and benchmark initial segmentation results.

Sprint 7: Prototype Development (Part 2) & Integration Planning

- Finalize segmentation benchmark testing.
- Plan component-level integration between segmentation, GPT-4o, and the frontend/ backend..

Sprint 8: Frontend Development (Web & Mobile)

- Set up React Native and React Native Web environments.
- Develop UI components: navigation, input/output pages, and segmented result display.

Sprint 9: Backend Development & Database Integration

- Set up Laravel backend, implement API endpoints, and integrate with PostgreSQL.
- Begin GPT-4o dataset preparation for fine-tuning.

Sprint 10: GPT-4o Fine-Tuning & Final Integration

- Fine-tune GPT-4o and integrate with backend.
- Refactor segmentation logic and integrate it with the full backend pipeline.

Sprint 11: Testing Phase

- Conduct frontend (unit, navigation, cross-platform) and backend testing.
- Validate segmentation accuracy and test GPT-4o response logic.

Sprint 12: Deployment & Finalization

- Deploy frontend, backend, and database.
- Final review, documentation, supervisor feedback, and presentation preparation.

Figure 3.1: Development Methodology Diagram

This project uses an organized approach that combines the Software Development Life Cycle (SDLC) and the conventional Project Life Cycle (PLC) to guarantee careful planning, methodical execution, and successful completion. While the SDLC concentrates on the technical development process, which includes requirement analysis, design, implementation, testing and deployment, the PLC leads the project through the clearly defined phases of initiation, planning, execution, testing and closure. During the design phase, a code-first prototyping technique was employed to accelerate the validation of essential AI components. This hybrid approach is reflected in every WBS phase, ensuring that system development and project management advancements are in line with the creation of a reliable multimodal LLM-based vehicle recognition system.

This hybrid methodology was chosen to ensure a balanced focus on both the technical development and the project management. By combining the SDLC's structured approach to system development with the PLC's clearly defined project phases, the project benefits from careful planning, risk mitigation and consistent monitoring of the progress. The incorporation of code-first prototyping during the design phase allows early validation of the critical AI components, such as the multimodal VLM integration, reducing potential rework and the technical risks. Aligning each WBS phase with both development and the management activities ensures that the deliverables are realistic, achievable, and of high quality. Overall, this approach provides the flexibility needed for the iterative AI development while maintaining the structured oversight necessary for the successful completion of a reliable vehicle recognition system.

3.2.1 Initiation Phase (Project Life Cycle) and Planning Phase (SDLC)

The project's beginning phase which is equivalent to the planning phase of the SDLC, involves determining the problem's scope, identifying its main obstacles and carrying out in-depth literature reviews. This involves thorough evaluations of current large language models (LLMs), car recognition systems, segmentation strategies and software development processes as described in Task 1.1.1.1. Activities involving comparative analysis such as Tasks 1.1.1.1.1.2 and 1.1.1.1.2.2 assist in determining the advantages and disadvantages of each domain which assisting in the selection of appropriate frameworks and technologies. Justifying the necessity of a multimodal LLM-integrated vehicle recognition system requires this basis. The procedure ends with a clear statement of the project's goals and scope (1.2) and a definition of the research gap (1.1.1.2).

3.2.2 Primary Research & Requirements Gathering (Planning Phase - Continued)

During this phase, the project uses survey design and analysis to collect empirical data. The development, dissemination, and assessment of questionnaires are demonstrated in tasks under 1.3. By verifying user need which directly inform requirements analysis in Task 2.1, these activities fill the gap between the initiation and planning stages. A thorough User Requirement Specification (URS) document (2.2) which describes both functional and non-functional needs (2.2.1.1 and 2.2.1.2) is drafted using the information received. To guarantee compatibility and performance for the suggested system, Task 2.4 also focuses on choosing the tech stack by contrasting databases, frontend frameworks, backend solutions, and LLM models.

3.2.3 Design Phase (SDLC)

Planning for system integration and code-based prototyping are the main focus of the design phase. The project moves straight to code-first prototyping (Phase 3), where preliminary segmentation models are put into practice and benchmarked (3.1), in place of high-level wireframes. This practical approach speeds up the early assessment of fundamental functionality. In order to prepare a blueprint for the full-stack development phase, the following integration planning (3.2) describes how the segmentation engine, GPT-4o, frontend and backend would interact.

3.2.4 Development Phase (SDLC)

Phase 4 describes the various components that make up the core development phase. Setting up an environment, designing components such as input/output pages and navigation and connecting to an API are all part of frontend development with React Native and React Native Web (4.1). Laravel is used in backend development (4.2) to build safe database operations, middleware logic, and APIs. For reliable and scalable data handling, PostgreSQL is incorporated (4.2.3). Simultaneously, GPT-4o is included into the system pipeline and refined (4.3) with carefully selected datasets. In order to ensure smooth data flow between components, this step also involves the complete integration and rewriting of the segmentation logic (4.4).

3.2.5 Testing Phase (PLC & SDLC)

Phase 5 will include extensive testing that includes system, integration and unit testing. The frontend (5.1), backend (5.2), and AI modules (5.3) all have their own testing streams to make sure they work properly both separately and together. Prior to deployment, quality assurance is ensured by tasks including cross-platform testing, API validation and accuracy checks for segmentation and GPT replies.

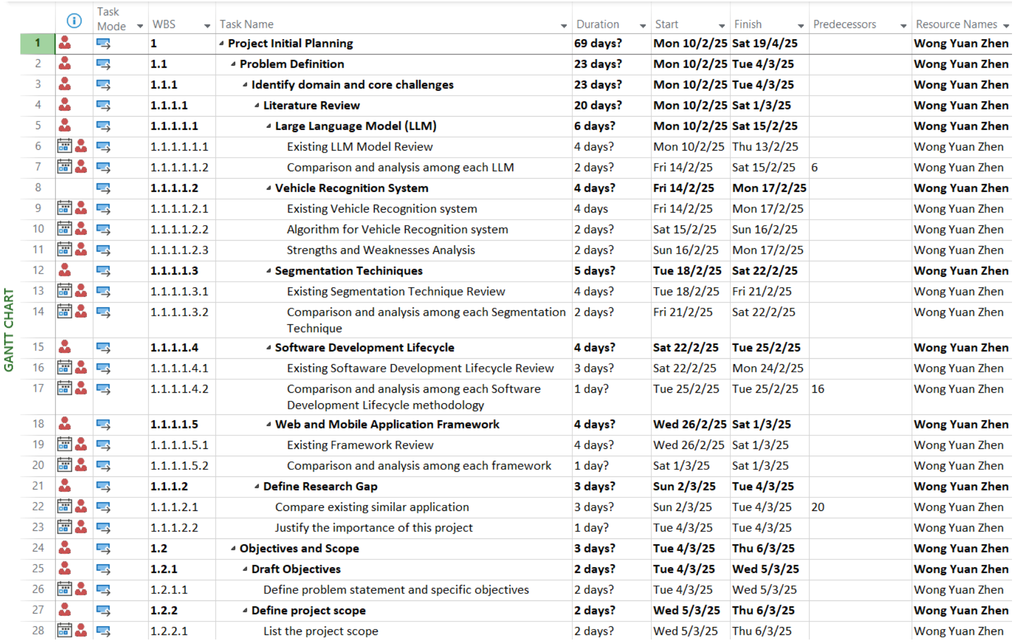
3.2.6 Deployment & Finalization (Closure Phase - PLC)

Project closing and deployment are covered in the last stage. Phase 6 involves deploying the solution to local or cloud infrastructure (6.1) followed by supervisor evaluations (6.2) and internal walkthroughs. With stakeholder input and knowledge transfer, the project lifecycle is concluded with the compilation of documentation and presentation materials (6.3). In addition to guaranteeing that deliverables are finished, this phase gets the project ready for future review, maintenance or scalability.

3.3 Project Schedule

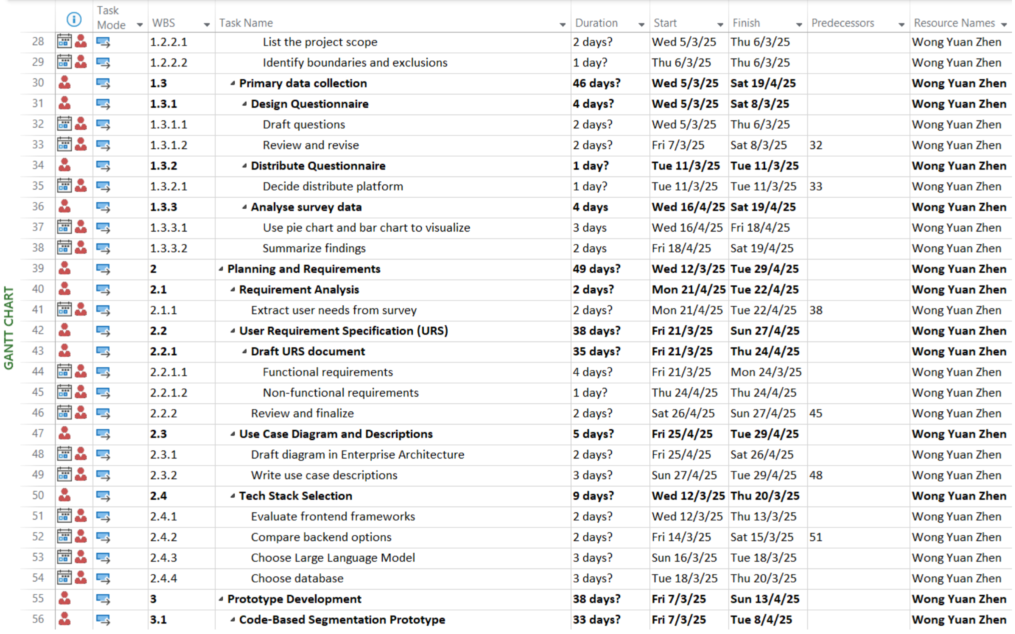
This section will mainly focus on the work breakdown structure (WBS) and Gantt Chart of this project.

3.3.1 Work Breakdown Structure



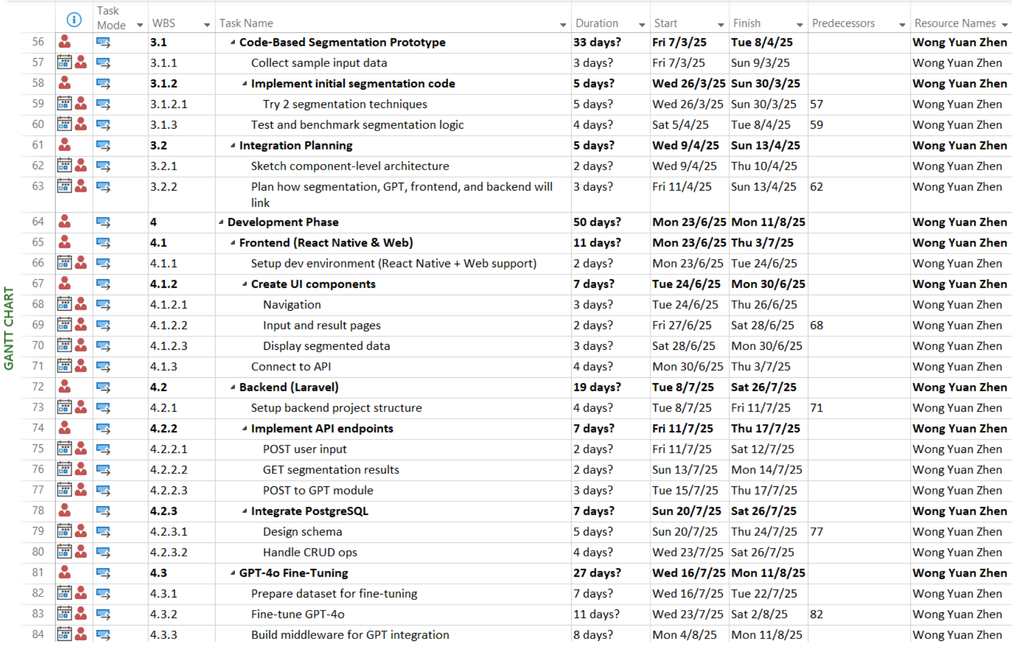
	ⓘ	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1			1	Project Initial Planning	69 days?	Mon 10/2/25	Sat 19/4/25		Wong Yuan Zhen
2			1.1	Problem Definition	23 days?	Mon 10/2/25	Tue 4/3/25		Wong Yuan Zhen
3			1.1.1	Identify domain and core challenges	23 days?	Mon 10/2/25	Tue 4/3/25		Wong Yuan Zhen
4			1.1.1.1	Literature Review	20 days?	Mon 10/2/25	Sat 1/3/25		Wong Yuan Zhen
5			1.1.1.1.1	Large Language Model (LLM)	6 days?	Mon 10/2/25	Sat 15/2/25		Wong Yuan Zhen
6			1.1.1.1.1.1	Existing LLM Model Review	4 days?	Mon 10/2/25	Thu 13/2/25		Wong Yuan Zhen
7			1.1.1.1.1.2	Comparison and analysis among each LLM	2 days?	Fri 14/2/25	Sat 15/2/25	6	Wong Yuan Zhen
8			1.1.1.1.2	Vehicle Recognition System	4 days?	Fri 14/2/25	Mon 17/2/25		Wong Yuan Zhen
9			1.1.1.1.2.1	Existing Vehicle Recognition system	4 days	Fri 14/2/25	Mon 17/2/25		Wong Yuan Zhen
10			1.1.1.1.2.2	Algorithm for Vehicle Recognition system	2 days?	Sat 15/2/25	Sun 16/2/25		Wong Yuan Zhen
11			1.1.1.1.2.3	Strengths and Weaknesses Analysis	2 days?	Sun 16/2/25	Mon 17/2/25		Wong Yuan Zhen
12			1.1.1.1.3	Segmentation Techiniques	5 days?	Tue 18/2/25	Sat 22/2/25		Wong Yuan Zhen
13			1.1.1.1.3.1	Existing Segmentation Technique Review	4 days?	Tue 18/2/25	Fri 21/2/25		Wong Yuan Zhen
14			1.1.1.1.3.2	Comparison and analysis among each Segmentation Technique	2 days?	Fri 21/2/25	Sat 22/2/25		Wong Yuan Zhen
15			1.1.1.1.4	Software Development Lifecycle	4 days?	Sat 22/2/25	Tue 25/2/25		Wong Yuan Zhen
16			1.1.1.1.4.1	Existing Softaware Development Lifecycle Review	3 days?	Sat 22/2/25	Mon 24/2/25		Wong Yuan Zhen
17			1.1.1.1.4.2	Comparison and analysis among each Software Development Lifecycle methodology	1 day?	Tue 25/2/25	Tue 25/2/25	16	Wong Yuan Zhen
18			1.1.1.1.5	Web and Mobile Application Framework	4 days?	Wed 26/2/25	Sat 1/3/25		Wong Yuan Zhen
19			1.1.1.1.5.1	Existing Framework Review	4 days?	Wed 26/2/25	Sat 1/3/25		Wong Yuan Zhen
20			1.1.1.1.5.2	Comparison and analysis among each framework	1 day?	Sat 1/3/25	Sat 1/3/25		Wong Yuan Zhen
21			1.1.1.2	Define Research Gap	3 days?	Sun 2/3/25	Tue 4/3/25		Wong Yuan Zhen
22			1.1.1.2.1	Compare existing similar application	3 days?	Sun 2/3/25	Tue 4/3/25	20	Wong Yuan Zhen
23			1.1.1.2.2	Justify the importance of this project	1 day?	Tue 4/3/25	Tue 4/3/25		Wong Yuan Zhen
24			1.2	Objectives and Scope	3 days?	Tue 4/3/25	Thu 6/3/25		Wong Yuan Zhen
25			1.2.1	Draft Objectives	2 days?	Tue 4/3/25	Wed 5/3/25		Wong Yuan Zhen
26			1.2.1.1	Define problem statement and specific objectives	2 days?	Tue 4/3/25	Wed 5/3/25		Wong Yuan Zhen
27			1.2.2	Define project scope	2 days?	Wed 5/3/25	Thu 6/3/25		Wong Yuan Zhen
28			1.2.2.1	List the project scope	2 days?	Wed 5/3/25	Thu 6/3/25		Wong Yuan Zhen

Figure 3.2: Project initial Planning Part 1



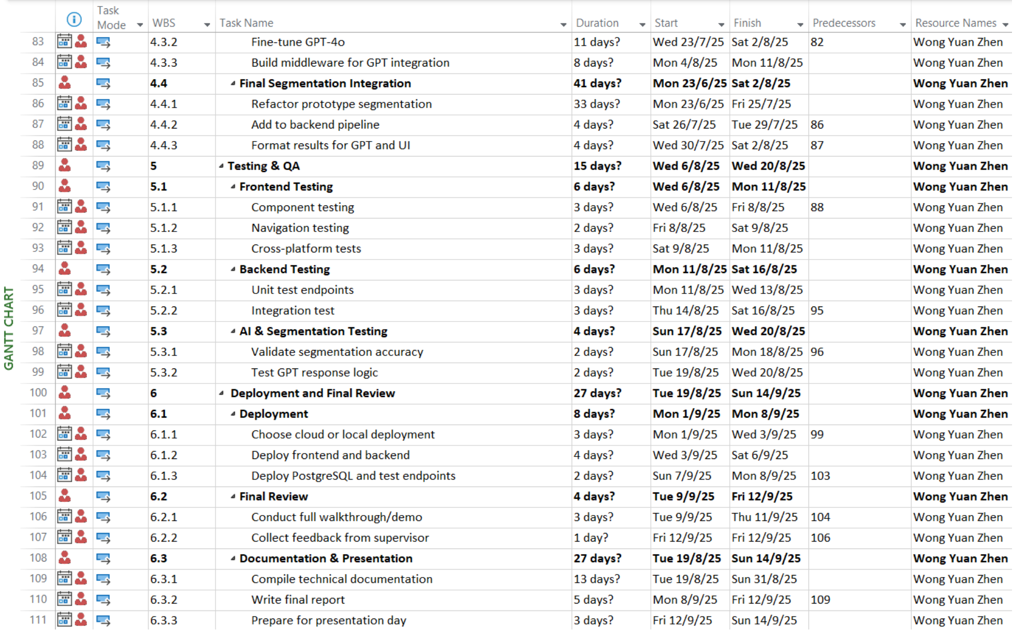
	ⓘ	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names
28			1.2.2.1	List the project scope	2 days?	Wed 5/3/25	Thu 6/3/25		Wong Yuan Zhen
29			1.2.2.2	Identify boundaries and exclusions	1 day?	Thu 6/3/25	Thu 6/3/25		Wong Yuan Zhen
30			1.3	Primary data collection	46 days?	Wed 5/3/25	Sat 19/4/25		Wong Yuan Zhen
31			1.3.1	Design Questionnaire	4 days?	Wed 5/3/25	Sat 8/3/25		Wong Yuan Zhen
32			1.3.1.1	Draft questions	2 days?	Wed 5/3/25	Thu 6/3/25		Wong Yuan Zhen
33			1.3.1.2	Review and revise	2 days?	Fri 7/3/25	Sat 8/3/25	32	Wong Yuan Zhen
34			1.3.2	Distribute Questionnaire	1 day?	Tue 11/3/25	Tue 11/3/25		Wong Yuan Zhen
35			1.3.2.1	Decide distribute platform	1 day?	Tue 11/3/25	Tue 11/3/25	33	Wong Yuan Zhen
36			1.3.3	Analyse survey data	4 days	Wed 16/4/25	Sat 19/4/25		Wong Yuan Zhen
37			1.3.3.1	Use pie chart and bar chart to visualize	3 days	Wed 16/4/25	Fri 18/4/25		Wong Yuan Zhen
38			1.3.3.2	Summarize findings	2 days	Fri 18/4/25	Sat 19/4/25		Wong Yuan Zhen
39			2	Planning and Requirements	49 days?	Wed 12/3/25	Tue 29/4/25		Wong Yuan Zhen
40			2.1	Requirement Analysis	2 days?	Mon 21/4/25	Tue 22/4/25		Wong Yuan Zhen
41			2.1.1	Extract user needs from survey	2 days?	Mon 21/4/25	Tue 22/4/25	38	Wong Yuan Zhen
42			2.2	User Requirement Specification (URS)	38 days?	Fri 21/3/25	Sun 27/4/25		Wong Yuan Zhen
43			2.2.1	Draft URS document	35 days?	Fri 21/3/25	Thu 24/4/25		Wong Yuan Zhen
44			2.2.1.1	Functional requirements	4 days?	Fri 21/3/25	Mon 24/3/25		Wong Yuan Zhen
45			2.2.1.2	Non-functional requirements	1 day?	Thu 24/4/25	Thu 24/4/25		Wong Yuan Zhen
46			2.2.2	Review and finalize	2 days?	Sat 26/4/25	Sun 27/4/25	45	Wong Yuan Zhen
47			2.3	Use Case Diagram and Descriptions	5 days?	Fri 25/4/25	Tue 29/4/25		Wong Yuan Zhen
48			2.3.1	Draft diagram in Enterprise Architecture	2 days?	Fri 25/4/25	Sat 26/4/25		Wong Yuan Zhen
49			2.3.2	Write use case descriptions	3 days?	Sun 27/4/25	Tue 29/4/25	48	Wong Yuan Zhen
50			2.4	Tech Stack Selection	9 days?	Wed 12/3/25	Thu 20/3/25		Wong Yuan Zhen
51			2.4.1	Evaluate frontend frameworks	2 days?	Wed 12/3/25	Thu 13/3/25		Wong Yuan Zhen
52			2.4.2	Compare backend options	2 days?	Fri 14/3/25	Sat 15/3/25	51	Wong Yuan Zhen
53			2.4.3	Choose Large Language Model	3 days?	Sun 16/3/25	Tue 18/3/25		Wong Yuan Zhen
54			2.4.4	Choose database	3 days?	Tue 18/3/25	Thu 20/3/25		Wong Yuan Zhen
55			3	Prototype Development	38 days?	Fri 7/3/25	Sun 13/4/25		Wong Yuan Zhen
56			3.1	Code-Based Segmentation Prototype	33 days?	Fri 7/3/25	Tue 8/4/25		Wong Yuan Zhen

Figure 3.3: Project Initial Planning Part 2, Planning and Requirements and Prototype Development Part 1



	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names
56	**3.1**	**Code-Based Segmentation Prototype**	**33 days?**	**Fri 7/3/25**	**Tue 8/4/25**		**Wong Yuan Zhen**
57	3.1.1	Collect sample input data	3 days?	Fri 7/3/25	Sun 9/3/25		Wong Yuan Zhen
58	**3.1.2**	**Implement initial segmentation code**	**5 days?**	**Wed 26/3/25**	**Sun 30/3/25**		**Wong Yuan Zhen**
59	3.1.2.1	Try 2 segmentation techniques	5 days?	Wed 26/3/25	Sun 30/3/25	57	Wong Yuan Zhen
60	3.1.3	Test and benchmark segmentation logic	4 days?	Sat 5/4/25	Tue 8/4/25	59	Wong Yuan Zhen
61	**3.2**	**Integration Planning**	**5 days?**	**Wed 9/4/25**	**Sun 13/4/25**		**Wong Yuan Zhen**
62	3.2.1	Sketch component-level architecture	2 days?	Wed 9/4/25	Thu 10/4/25		Wong Yuan Zhen
63	3.2.2	Plan how segmentation, GPT, frontend, and backend will link	3 days?	Fri 11/4/25	Sun 13/4/25	62	Wong Yuan Zhen
64	**4**	**Development Phase**	**50 days?**	**Mon 23/6/25**	**Mon 11/8/25**		**Wong Yuan Zhen**
65	**4.1**	**Frontend (React Native & Web)**	**11 days?**	**Mon 23/6/25**	**Thu 3/7/25**		**Wong Yuan Zhen**
66	4.1.1	Setup dev environment (React Native + Web support)	2 days?	Mon 23/6/25	Tue 24/6/25		Wong Yuan Zhen
67	**4.1.2**	**Create UI components**	**7 days?**	**Tue 24/6/25**	**Mon 30/6/25**		**Wong Yuan Zhen**
68	4.1.2.1	Navigation	3 days?	Tue 24/6/25	Thu 26/6/25		Wong Yuan Zhen
69	4.1.2.2	Input and result pages	2 days?	Fri 27/6/25	Sat 28/6/25	68	Wong Yuan Zhen
70	4.1.2.3	Display segmented data	3 days?	Sat 28/6/25	Mon 30/6/25		Wong Yuan Zhen
71	4.1.3	Connect to API	4 days?	Mon 30/6/25	Thu 3/7/25		Wong Yuan Zhen
72	**4.2**	**Backend (Laravel)**	**19 days?**	**Tue 8/7/25**	**Sat 26/7/25**		**Wong Yuan Zhen**
73	4.2.1	Setup backend project structure	4 days?	Tue 8/7/25	Fri 11/7/25	71	Wong Yuan Zhen
74	**4.2.2**	**Implement API endpoints**	**7 days?**	**Fri 11/7/25**	**Thu 17/7/25**		**Wong Yuan Zhen**
75	4.2.2.1	POST user input	2 days?	Fri 11/7/25	Sat 12/7/25		Wong Yuan Zhen
76	4.2.2.2	GET segmentation results	2 days?	Sun 13/7/25	Mon 14/7/25		Wong Yuan Zhen
77	4.2.2.3	POST to GPT module	3 days?	Tue 15/7/25	Thu 17/7/25		Wong Yuan Zhen
78	**4.2.3**	**Integrate PostgreSQL**	**7 days?**	**Sun 20/7/25**	**Sat 26/7/25**		**Wong Yuan Zhen**
79	4.2.3.1	Design schema	5 days?	Sun 20/7/25	Thu 24/7/25	77	Wong Yuan Zhen
80	4.2.3.2	Handle CRUD ops	4 days?	Wed 23/7/25	Sat 26/7/25		Wong Yuan Zhen
81	**4.3**	**GPT-4o Fine-Tuning**	**27 days?**	**Wed 16/7/25**	**Mon 11/8/25**		**Wong Yuan Zhen**
82	4.3.1	Prepare dataset for fine-tuning	7 days?	Wed 16/7/25	Tue 22/7/25		Wong Yuan Zhen
83	4.3.2	Fine-tune GPT-4o	11 days?	Wed 23/7/25	Sat 2/8/25	82	Wong Yuan Zhen
84	4.3.3	Build middleware for GPT integration	8 days?	Mon 4/8/25	Mon 11/8/25		Wong Yuan Zhen

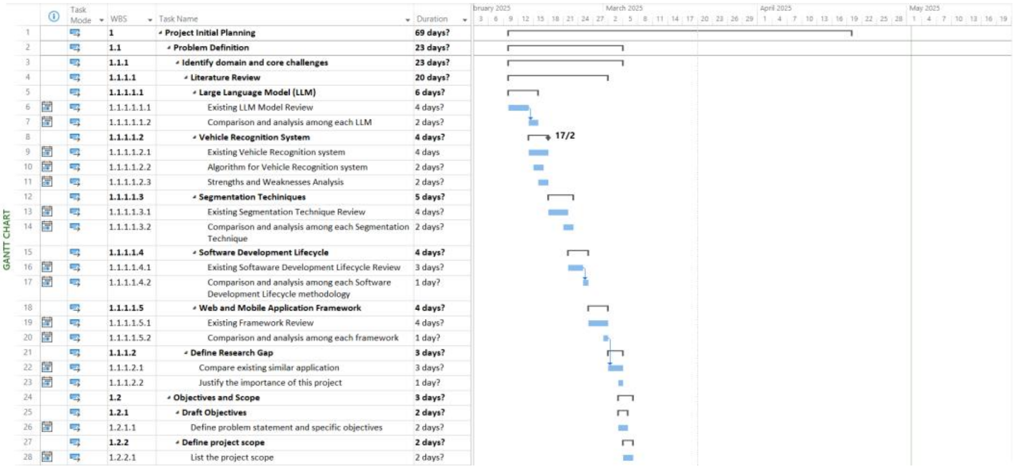
Figure 3.4: Prototype Development Part 2 and Development Phase Part 1



	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names
83	4.3.2	Fine-tune GPT-4o	11 days?	Wed 23/7/25	Sat 2/8/25	82	Wong Yuan Zhen
84	4.3.3	Build middleware for GPT integration	8 days?	Mon 4/8/25	Mon 11/8/25		Wong Yuan Zhen
85	**4.4**	**Final Segmentation Integration**	**41 days?**	**Mon 23/6/25**	**Sat 2/8/25**		**Wong Yuan Zhen**
86	4.4.1	Refactor prototype segmentation	33 days?	Mon 23/6/25	Fri 25/7/25		Wong Yuan Zhen
87	4.4.2	Add to backend pipeline	4 days?	Sat 26/7/25	Tue 29/7/25	86	Wong Yuan Zhen
88	4.4.3	Format results for GPT and UI	4 days?	Wed 30/7/25	Sat 2/8/25	87	Wong Yuan Zhen
89	**5**	**Testing & QA**	**15 days?**	**Wed 6/8/25**	**Wed 20/8/25**		**Wong Yuan Zhen**
90	**5.1**	**Frontend Testing**	**6 days?**	**Wed 6/8/25**	**Mon 11/8/25**		**Wong Yuan Zhen**
91	5.1.1	Component testing	3 days?	Wed 6/8/25	Fri 8/8/25	88	Wong Yuan Zhen
92	5.1.2	Navigation testing	2 days?	Fri 8/8/25	Sat 9/8/25		Wong Yuan Zhen
93	5.1.3	Cross-platform tests	3 days?	Sat 9/8/25	Mon 11/8/25		Wong Yuan Zhen
94	**5.2**	**Backend Testing**	**6 days?**	**Mon 11/8/25**	**Sat 16/8/25**		**Wong Yuan Zhen**
95	5.2.1	Unit test endpoints	3 days?	Mon 11/8/25	Wed 13/8/25		Wong Yuan Zhen
96	5.2.2	Integration test	3 days?	Thu 14/8/25	Sat 16/8/25	95	Wong Yuan Zhen
97	**5.3**	**AI & Segmentation Testing**	**4 days?**	**Sun 17/8/25**	**Wed 20/8/25**		**Wong Yuan Zhen**
98	5.3.1	Validate segmentation accuracy	2 days?	Sun 17/8/25	Mon 18/8/25	96	Wong Yuan Zhen
99	5.3.2	Test GPT response logic	2 days?	Tue 19/8/25	Wed 20/8/25		Wong Yuan Zhen
100	**6**	**Deployment and Final Review**	**27 days?**	**Tue 19/8/25**	**Sun 14/9/25**		**Wong Yuan Zhen**
101	**6.1**	**Deployment**	**8 days?**	**Mon 1/9/25**	**Mon 8/9/25**		**Wong Yuan Zhen**
102	6.1.1	Choose cloud or local deployment	3 days?	Mon 1/9/25	Wed 3/9/25	99	Wong Yuan Zhen
103	6.1.2	Deploy frontend and backend	4 days?	Wed 3/9/25	Sat 6/9/25		Wong Yuan Zhen
104	6.1.3	Deploy PostgreSQL and test endpoints	2 days?	Sun 7/9/25	Mon 8/9/25	103	Wong Yuan Zhen
105	**6.2**	**Final Review**	**4 days?**	**Tue 9/9/25**	**Fri 12/9/25**		**Wong Yuan Zhen**
106	6.2.1	Conduct full walkthrough/demo	3 days?	Tue 9/9/25	Thu 11/9/25	104	Wong Yuan Zhen
107	6.2.2	Collect feedback from supervisor	1 day?	Fri 12/9/25	Fri 12/9/25	106	Wong Yuan Zhen
108	**6.3**	**Documentation & Presentation**	**27 days?**	**Tue 19/8/25**	**Sun 14/9/25**		**Wong Yuan Zhen**
109	6.3.1	Compile technical documentation	13 days?	Tue 19/8/25	Sun 31/8/25		Wong Yuan Zhen
110	6.3.2	Write final report	5 days?	Mon 8/9/25	Fri 12/9/25	109	Wong Yuan Zhen
111	6.3.3	Prepare for presentation day	3 days?	Fri 12/9/25	Sun 14/9/25		Wong Yuan Zhen

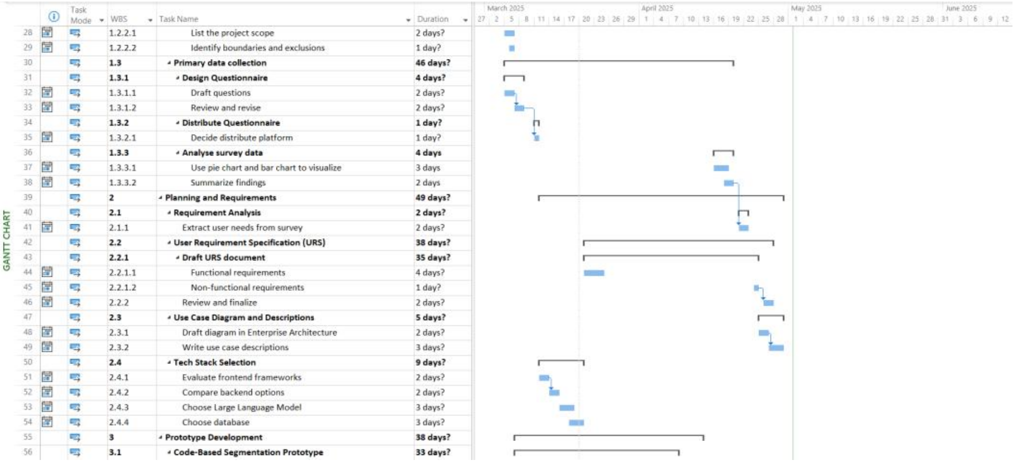
Figure 3.5: Development Phase Part 2, Testing and QA and Deployment and Final Review

3.3.2 Gantt Chart



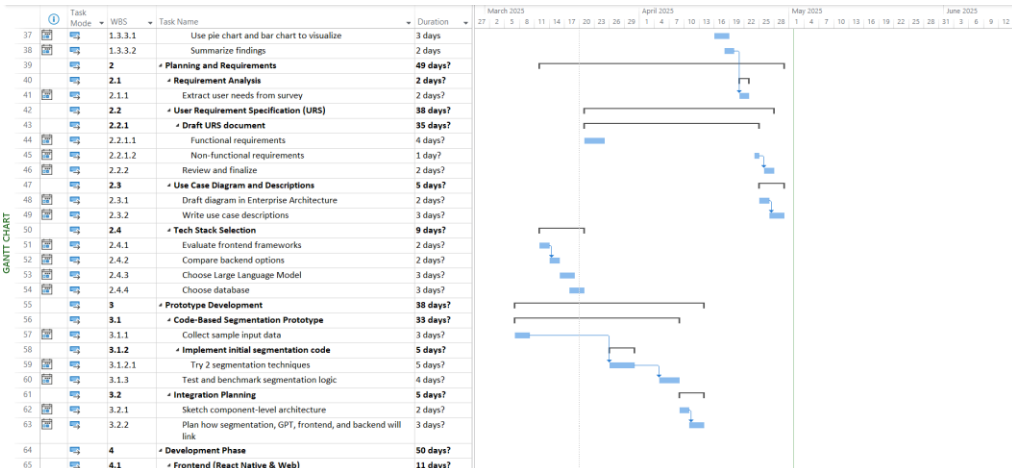
	ⓘ	Task Mode	WBS	Task Name	Duration
1			**1**	**Project Initial Planning**	**69 days?**
2			**1.1**	**Problem Definition**	**23 days?**
3			**1.1.1**	**Identify domain and core challenges**	**23 days?**
4			**1.1.1.1**	**Literature Review**	**20 days?**
5			**1.1.1.1.1**	**Large Language Model (LLM)**	**6 days?**
6			1.1.1.1.1.1	Existing LLM Model Review	4 days?
7			1.1.1.1.1.2	Comparison and analysis among each LLM	2 days?
8			**1.1.1.1.2**	**Vehicle Recognition System**	**4 days?**
9			1.1.1.1.2.1	Existing Vehicle Recognition system	4 days
10			1.1.1.1.2.2	Algorithm for Vehicle Recognition system	2 days?
11			1.1.1.1.2.3	Strengths and Weaknesses Analysis	2 days?
12			**1.1.1.1.3**	**Segmentation Techiniques**	**5 days?**
13			1.1.1.1.3.1	Existing Segmentation Technique Review	4 days?
14			1.1.1.1.3.2	Comparison and analysis among each Segmentation Technique	2 days?
15			**1.1.1.1.4**	**Software Development Lifecycle**	**4 days?**
16			1.1.1.1.4.1	Existing Softaware Development Lifecycle Review	3 days?
17			1.1.1.1.4.2	Comparison and analysis among each Software Development Lifecycle methodology	1 day?
18			**1.1.1.1.5**	**Web and Mobile Application Framework**	**4 days?**
19			1.1.1.1.5.1	Existing Framework Review	4 days?
20			1.1.1.1.5.2	Comparison and analysis among each framework	1 day?
21			**1.1.1.2**	**Define Research Gap**	**3 days?**
22			1.1.1.2.1	Compare existing similar application	3 days?
23			1.1.1.2.2	Justify the importance of this project	1 day?
24			**1.2**	**Objectives and Scope**	**3 days?**
25			**1.2.1**	**Draft Objectives**	**2 days?**
26			1.2.1.1	Define problem statement and specific objectives	2 days?
27			**1.2.2**	**Define project scope**	**2 days?**
28			1.2.2.1	List the project scope	2 days?

Figure 3.6: Project initial Planning – Part 1



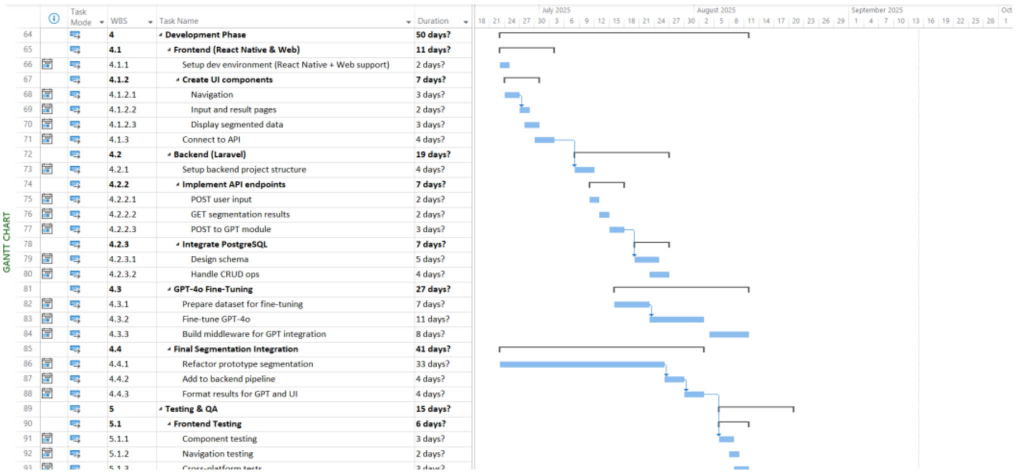
	ⓘ	Task Mode	WBS	Task Name	Duration
28			1.2.2.1	List the project scope	2 days?
29			1.2.2.2	Identify boundaries and exclusions	1 day?
30			**1.3**	**Primary data collection**	**46 days?**
31			**1.3.1**	**Design Questionnaire**	**4 days?**
32			1.3.1.1	Draft questions	2 days?
33			1.3.1.2	Review and revise	2 days?
34			**1.3.2**	**Distribute Questionnaire**	**1 day?**
35			1.3.2.1	Decide distribute platform	1 day?
36			**1.3.3**	**Analyse survey data**	**4 days**
37			1.3.3.1	Use pie chart and bar chart to visualize	3 days
38			1.3.3.2	Summarize findings	2 days
39			**2**	**Planning and Requirements**	**49 days?**
40			**2.1**	**Requirement Analysis**	**2 days?**
41			2.1.1	Extract user needs from survey	2 days?
42			**2.2**	**User Requirement Specification (URS)**	**38 days?**
43			**2.2.1**	**Draft URS document**	**35 days?**
44			2.2.1.1	Functional requirements	4 days?
45			2.2.1.2	Non-functional requirements	1 day?
46			2.2.2	Review and finalize	2 days?
47			**2.3**	**Use Case Diagram and Descriptions**	**5 days?**
48			2.3.1	Draft diagram in Enterprise Architecture	2 days?
49			2.3.2	Write use case descriptions	3 days?
50			**2.4**	**Tech Stack Selection**	**9 days?**
51			2.4.1	Evaluate frontend frameworks	2 days?
52			2.4.2	Compare backend options	2 days?
53			2.4.3	Choose Large Language Model	3 days?
54			2.4.4	Choose database	3 days?
55			**3**	**Prototype Development**	**38 days?**
56			**3.1**	**Code-Based Segmentation Prototype**	**33 days?**

Figure 3.7: Project Initial Planning Part 2, Planning and Requirements and Prototype Development Part 1



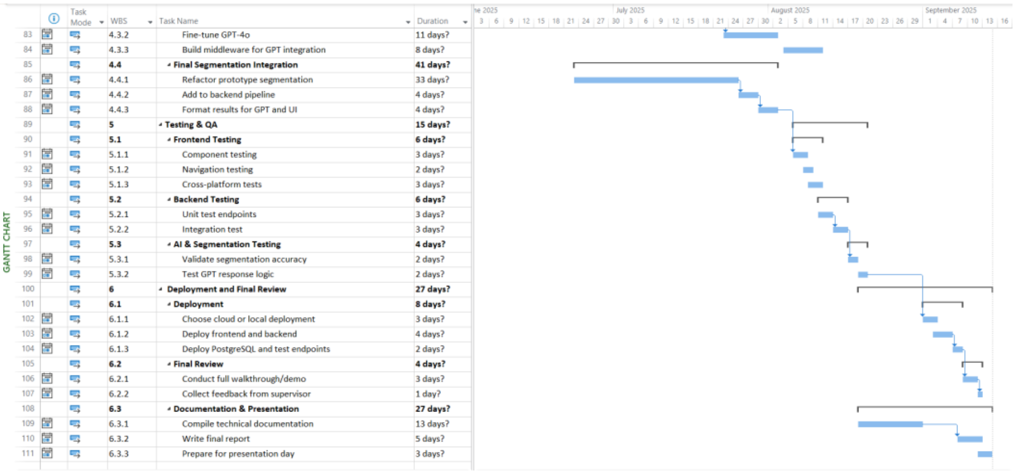
	ⓘ	Task Mode	WBS	Task Name	Duration
37			1.3.3.1	Use pie chart and bar chart to visualize	3 days
38			1.3.3.2	Summarize findings	2 days
39			**2**	**Planning and Requirements**	**49 days?**
40			**2.1**	**Requirement Analysis**	**2 days?**
41			2.1.1	Extract user needs from survey	2 days?
42			**2.2**	**User Requirement Specification (URS)**	**38 days?**
43			**2.2.1**	**Draft URS document**	**35 days?**
44			2.2.1.1	Functional requirements	4 days?
45			2.2.1.2	Non-functional requirements	1 day?
46			2.2.2	Review and finalize	2 days?
47			**2.3**	**Use Case Diagram and Descriptions**	**5 days?**
48			2.3.1	Draft diagram in Enterprise Architecture	2 days?
49			2.3.2	Write use case descriptions	3 days?
50			**2.4**	**Tech Stack Selection**	**9 days?**
51			2.4.1	Evaluate frontend frameworks	2 days?
52			2.4.2	Compare backend options	2 days?
53			2.4.3	Choose Large Language Model	3 days?
54			2.4.4	Choose database	3 days?
55			**3**	**Prototype Development**	**38 days?**
56			**3.1**	**Code-Based Segmentation Prototype**	**33 days?**
57			3.1.1	Collect sample input data	3 days?
58			**3.1.2**	**Implement initial segmentation code**	**5 days?**
59			3.1.2.1	Try 2 segmentation techniques	5 days?
60			3.1.3	Test and benchmark segmentation logic	4 days?
61			**3.2**	**Integration Planning**	**5 days?**
62			3.2.1	Sketch component-level architecture	2 days?
63			3.2.2	Plan how segmentation, GPT, frontend, and backend will link	3 days?
64			**4**	**Development Phase**	**50 days?**
65			**4.1**	**Frontend (React Native & Web)**	**11 days?**

Figure 3.8: Prototype Development Part 2



	Task Mode	WBS	Task Name	Duration
64		**4**	**Development Phase**	**50 days?**
65		**4.1**	**Frontend (React Native & Web)**	**11 days?**
66		4.1.1	Setup dev environment (React Native + Web support)	2 days?
67		**4.1.2**	**Create UI components**	**7 days?**
68		4.1.2.1	Navigation	3 days?
69		4.1.2.2	Input and result pages	2 days?
70		4.1.2.3	Display segmented data	3 days?
71		4.1.3	Connect to API	4 days?
72		**4.2**	**Backend (Laravel)**	**19 days?**
73		4.2.1	Setup backend project structure	4 days?
74		**4.2.2**	**Implement API endpoints**	**7 days?**
75		4.2.2.1	POST user input	2 days?
76		4.2.2.2	GET segmentation results	2 days?
77		4.2.2.3	POST to GPT module	3 days?
78		**4.2.3**	**Integrate PostgreSQL**	**7 days?**
79		4.2.3.1	Design schema	5 days?
80		4.2.3.2	Handle CRUD ops	4 days?
81		**4.3**	**GPT-4o Fine-Tuning**	**27 days?**
82		4.3.1	Prepare dataset for fine-tuning	7 days?
83		4.3.2	Fine-tune GPT-4o	11 days?
84		4.3.3	Build middleware for GPT integration	8 days?
85		**4.4**	**Final Segmentation Integration**	**41 days?**
86		4.4.1	Refactor prototype segmentation	33 days?
87		4.4.2	Add to backend pipeline	4 days?
88		4.4.3	Format results for GPT and UI	4 days?
89		**5**	**Testing & QA**	**15 days?**
90		**5.1**	**Frontend Testing**	**6 days?**
91		5.1.1	Component testing	3 days?
92		5.1.2	Navigation testing	2 days?
93		5.1.3	Cross-platform tests	3 days?

Figure 3.9: Development Phase Part 2 and Testing and QA Part 1



	Task Mode	WBS	Task Name	Duration
83		4.3.2	Fine-tune GPT-4o	11 days?
84		4.3.3	Build middleware for GPT integration	8 days?
85		**4.4**	**Final Segmentation Integration**	**41 days?**
86		4.4.1	Refactor prototype segmentation	33 days?
87		4.4.2	Add to backend pipeline	4 days?
88		4.4.3	Format results for GPT and UI	4 days?
89		**5**	**Testing & QA**	**15 days?**
90		**5.1**	**Frontend Testing**	**6 days?**
91		5.1.1	Component testing	3 days?
92		5.1.2	Navigation testing	2 days?
93		5.1.3	Cross-platform tests	3 days?
94		**5.2**	**Backend Testing**	**6 days?**
95		5.2.1	Unit test endpoints	3 days?
96		5.2.2	Integration test	3 days?
97		**5.3**	**AI & Segmentation Testing**	**4 days?**
98		5.3.1	Validate segmentation accuracy	2 days?
99		5.3.2	Test GPT response logic	2 days?
100		**6**	**Deployment and Final Review**	**27 days?**
101		**6.1**	**Deployment**	**8 days?**
102		6.1.1	Choose cloud or local deployment	3 days?
103		6.1.2	Deploy frontend and backend	4 days?
104		6.1.3	Deploy PostgreSQL and test endpoints	2 days?
105		**6.2**	**Final Review**	**4 days?**
106		6.2.1	Conduct full walkthrough/demo	3 days?
107		6.2.2	Collect feedback from supervisor	1 day?
108		**6.3**	**Documentation & Presentation**	**27 days?**
109		6.3.1	Compile technical documentation	13 days?
110		6.3.2	Write final report	5 days?
111		6.3.3	Prepare for presentation day	3 days?

Figure 3.10: Testing and QA Part 2 and Deployment and Final Review

3.4 Development Tools

The IDE tool Visual Studio Code, the frontend frameworks React Native and React Native Web, the backend framework Next.js, the database PostgreSQL, the backend database management service Supabase, the deployment tool Vercel, the vehicle recognition algorithm, and the datasets are all included in this section as necessary to build the implemented system.

3.4.1 Visual Studio Code IDE

Visual Studio Code was the main coding tool utilized for this project. It offered the advantage of tool extensibility which enabling developers to add snippets to make coding easier. VS Code could edit a variety of programming languages, including HTML, Tailwind CSS, TypeScript, and others after installing the programming language CLI. Additionally, syntax highlighting was made possible with the addition of Laravel extensions to Visual Studio Code which aided in the quicker detection of syntax issues.

3.4.2 Enterprise Architecture

With a wide range of features and capabilities for software modeling, design, and analysis, Enterprise Architect (EA) is a strong and adaptable UML analys is tool. It is extensively employed in many different fields including engineering, architecture and software development. The use case diagrams for every feature offered by the multimodal vehicle recognition access control system were created using EA in this project. By clearly illustrating how the system behaves from the viewpoint of its users, use case diagrams were intended to aid in the identification of the system's functional needs.

3.4.3 React Native

The multimodal LLM-based vehicle detection system's interface was developed using React Native to provide cross-platform interoperability and a consistent user experience on mobile and web devices. React Native's component-based architecture and hot-reload features greatly sped up development and testing because the system necessitates real-time user involvement for tasks like uploading car pictures, examining segmentation results and obtaining GPT-4o feedback. Its cross-platform code sharing capability guaranteed a consistent user interface and decreased redundancy. When this method combined with React Native for Web, it is made possible to retain excellent performance and responsiveness in both environments while streamlining the development process.

3.4.4 Tailwind CSS

For the multimodal LLM-based vehicle detection system, Tailwind CSS was selected as the style framework because of its utility-first methodology which facilitates quick user interface development with clear and maintainable code. In order to display complicated outputs like segmented photos and GPT-4o replies, a responsive and user-friendly user interface was required. Tailwind's preset utility classes made it possible to quickly alter layouts and style them consistently without using cumbersome custom CSS. The application was a perfect fit for the system's web and mobile interfaces because of its mobile-first design philosophy which also made sure that the user experience was consistent across screen sizes.

3.4.5 Next.js

Next.js was chosen as the backend framework for the multimodal LLM-based vehicle detection system due to its integrated API routes, modular design, and smooth interaction with Node.js modules. For handling backend operations like processing user inputs, handling image segmentation findings, and synchronizing GPT-4o outputs, Next.js offered a solid basis. User authentication, optimized model interactions, and the effective execution of segmentation requests were made possible by its support for serverless operations and middleware. Through ORMs like Prisma, Next.js integrated with PostgreSQL to offer dependable database operations while preserving performance and scalability. Next.js was a good fit for our AI-driven recognition project because it makes backend development easier, guarantees safe API handling, and works with contemporary deployment settings.

3.4.6 PostgreSQL

Because of its sophisticated data handling capabilities, support for complicated queries, and dependability, PostgreSQL was chosen as the database administration solution for the multimodal LLM-based vehicle recognition project. Structured data including segmented vehicle attributes, user inputs, model predictions and system logs must be efficiently stored and retrieved by the system. PostgreSQL was ideally suited for handling the varied and expanding datasets that are common in AI-integrated systems because of its support for JSON, full-text search and indexing methods. Furthermore, data integrity and scalability were ensured by its strong ACID compliance and good performance under concurrent access, both of which are essential for the recognition system's smooth operation on online and mobile platforms.

3.4.7 Supabase

Because of its strong PostgreSQL basis and cutting-edge capabilities that directly match the system's requirements, Supabase was selected as the backend platform for database management of this project. While JSON/JSONB support offers flexibility for semi-structured data, it facilitates the effective storing and retrieval of structured data which including segmented vehicle attributes, user inputs, model predictions, and system logs. Realtime subscriptions via PostgreSQL's LISTEN/NOTIFY allow clients to get updates instantly, guaranteeing that any new alerts or recognition results are sent out right away. While scalability and managed hosting relieve the strain of managing a database cluster, strong ACID compliance protects data integrity under concurrent access. Supabase takes care of scaling, backups, and maintenance automatically. All of these characteristics combine to make Supabase the perfect, dependable, and expandable system backbone.

3.4.8 Vercel

Vercel was selected as the hosting and deployment platform for this project due to its smooth integration with Next.js and emphasis on backend performance. It ensures that the vehicle recognition system runs effectively and dependably by offering an optimum environment for server-side rendering and API execution. Automatic updates from the GitHub repository are made possible by Vercel's integrated support for continuous integration and deployment (CI/CD), which lowers deployment overhead and boosts development productivity. Low-latency API call replies are further guaranteed by its global content delivery network (CDN), and the system's autonomous scalability enables it to accommodate changing workloads without the need for manual infrastructure maintenance. Vercel is a scalable and reliable option for hosting the project's backend services because of these features.

3.4.9 GPT-4o Model

With the help of segmented picture data and human input, GPT-4o was included into the multimodal vehicle identification system to deliver intelligent and adaptable replies. It was well-suited for understanding vehicle attributes and providing contextual insights or classifications because of its multimodal capabilities which allow it to handle both text and visual information. The system was able to improve its ability to enable real-time recognition and feedback by fine-tuning GPT-4o with domain-specific data which resulted in higher output accuracy and relevance. Because of its adaptability, GPT-4o was also able to integrate with the backend with ease and utilizing its language and reasoning capabilities to enhance user engagement and application decision-making.

3.4.10 SegementAnything

Because Segment Anything can generalize across various object categories with little task-specific training, it was used as the main image segmentation module in the multimodal vehicle detection system. Regardless of different surroundings or viewpoints, the system was able to precisely extract vehicle components including license plates, headlights or logos due to to its prompt-based segmentation technique. For the refined GPT-4o model to receive organized visual input and perform accurate analysis and interpretation, this high-quality segmentation was crucial. Segment Anything was integrated into the system to create a strong preprocessing phase that greatly improved the overall precision and adaptability of the vehicle recognition procedure.
(Note: The segmentation technique was changed to YOLO in later chapter 6.2.3)

3.5 Summary

To guarantee an organized and effective workflow, the Project Life Cycle (PLC) and the Software Development Life Cycle (SDLC) both served as development process guidelines for this project. In accordance with the project's six-month timeframe, the PLC was split into discrete phases which included initiation, planning, execution, monitoring and closing. Foundational tasks like the literature study, problem identification, requirement analysis and prototype creation were finished during the beginning and planning stages. System implementation was the main emphasis of the execution phase, which included GPT-4o and Segment Anything integration, PostgreSQL integration, frontend interface development using React Native and Tailwind CSS, and backend development using Laravel. The monitoring and closure phases included testing, deployment and final delivery.

In accordance with the Agile-based SDLC methodology, tasks were logically arranged across documentation, development and deployment utilizing a multi-level Work Breakdown Structure (WBS) and divided into 2-

week sprints. The timetable and dependencies were visualized using a Gantt chart which also ensured simultaneous work in several components like UI development and segmentation fine-tuning by indicating overlapping jobs. Project planning, development and tracking were supported by tools including VS Code, Postman, GitHub, and Microsoft Project. Iterative development and continuous integration were supported while a clear deadline was maintained because of this systematic combination of PLC and SDLC.

CHAPTER 4

PRELIMINARY RESULTS

4.1 Introduction

This chapter typically creates the project specification after gathering and eliciting requirements. In order to learn about the current workflow, the first stage is fact-finding which involves identifying and comprehending requirements utilizing an online survey questionnaire. To illustrate how actors engage with the multimodal vehicle recognition access control system, a use case description and diagram were then produced. In order to better comprehend the system's designs and functionalities, a prototype was finally made to illustrate the user interfaces.

4.2 Fact Finding

A total of 30 responses was collected from the intended target audience. The intended target audience provided a total of 30 responses. The questions in this survey were divided into three parts. Demographic data was gathered in Section A, while user experiences with the current vehicle access system were obtained in Section B. Data regarding user expectations for a new system is collected for Section C. Finally, two open-ended questions allow users to express any ideas or concerns that aren't covered in the closed questions.

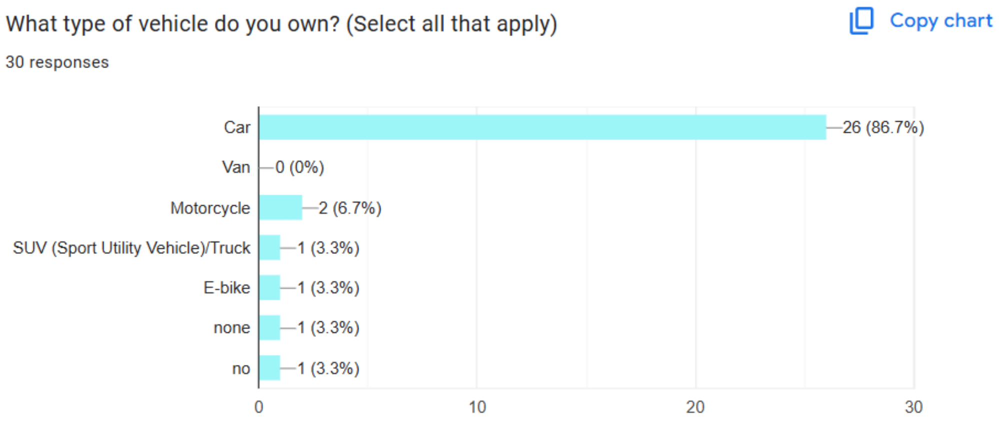
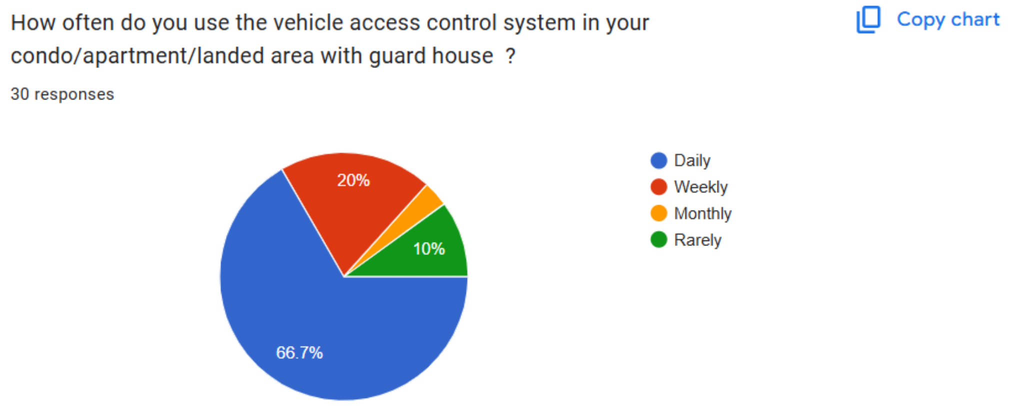


Figure 4.1: Type of vehicle owned

A data visualization of a survey on vehicle ownership with 30 participants is shown in the image. Each vehicle type which included car, van, motorcycle, SUV/truck, e-bike and other (none, no) as well as the proportion of responses are displayed in a horizontal bar chart format. With lengths proportional to their values, the bars extend rightward. According to the bar chart, "Car" has the longest bar in which showing the highest number of responses. The scale is indicated by numerical labels (0–30) at the bottom while percentages are shown next to each bar.

The distribution of vehicle ownership among respondents is clearly shown in this image, with cars being the most prevalent. Although the percentages such as 3.3% for "Motorcycle" aid in placing the data in context, the chart might be enhanced by clearly marking each bar's precise count and eliminating unnecessary choices which included "none" and "no" that come from other. Overall, it accomplishes its goal of demonstrating relative popularity.



How often do you use the vehicle access control system in your condo/apartment/landed area with guard house ?

30 responses

Figure 4.2: Utilization range of vehicle access control system

The image displays a pie chart that visualizes usage frequency statistics for a vehicle access control system. Different usage frequencies are represented by the four segments of the chart which included daily, weekly, monthly, and rarely. It is instantly clear that the "Daily" category dominates with 66.7%, but usage rates in the other categories are steadily declining. This graphic successfully highlights the significant dependence on everyday use, demonstrating that the access system is an essential part of residents' daily lives. T his has highlighted the necessity of reliability and effectiveness in its architecture.

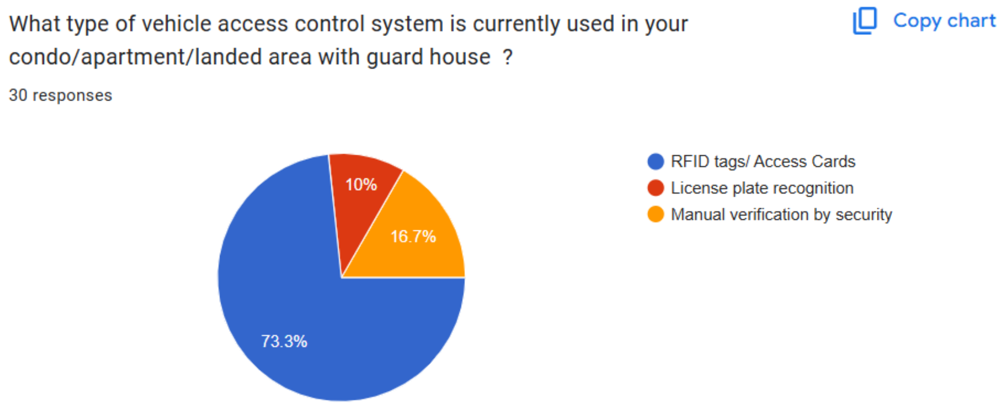


Figure 4.3: Type of vehicle access control system

The data shows that human verification is less widespread (16.7%), while RFID technology dominates current systems (73.3%). A substantial possibility for technical advancement in access control is highlighted by the low adoption rate of license plate recognition (10%). Possibly as a result of their reliability and simplicity of use, the graphic successfully conveys the present market preference for RFID technologies.

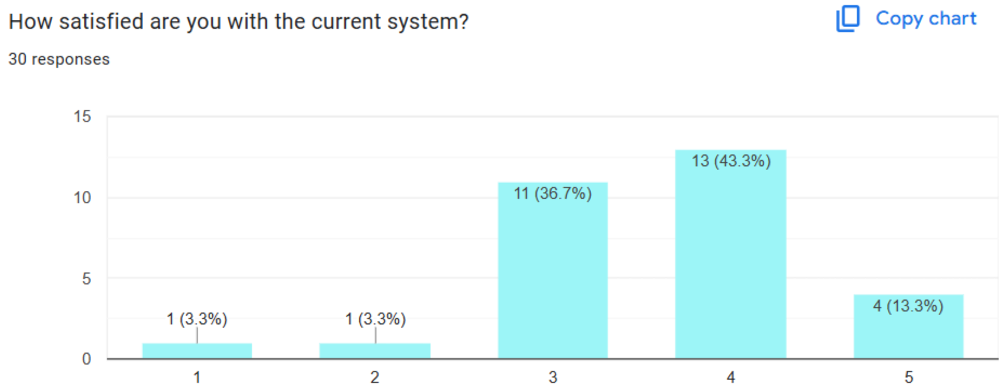


Figure 4.4: Satisfaction on current system

While the majority of users are moderately satisfied (levels 3–4), relatively few are absolutely content (level 5 at 13.3%) or totally dissatisfied (levels 1–2 at 6.6% combined) as this visualization skillfully illustrates. T he chart is successful in pointing out areas that require improvement, especially in terms of converting mid-tier satisfaction (36.7% at level 3) to higher ratings, it could be improved by emphasizing actionable insights. This data indicates to stakeholders that although the existing system works well, major improvements might significantly increase the satisfaction of users.

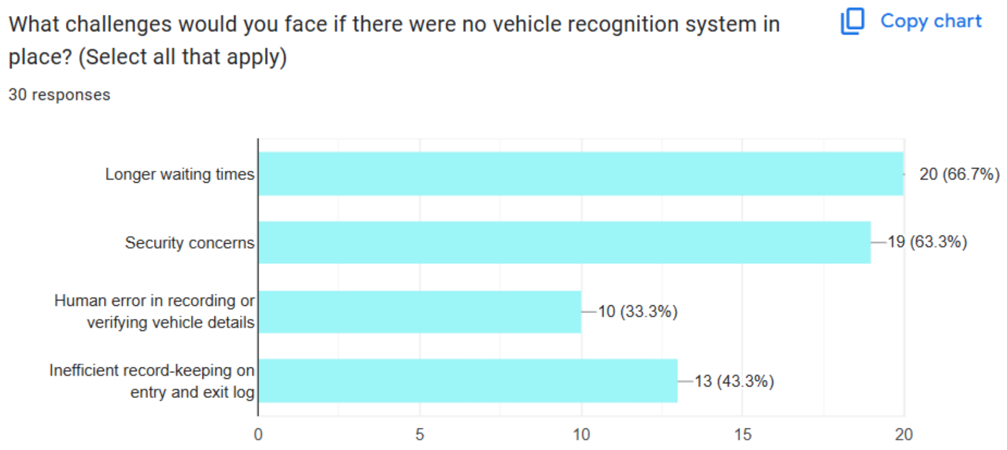


Figure 4.5: Challenges without vehicle recognition system

According to this statistics, 66.7% of people were concerned about delays, indicating that they primarily value the current system for time savings. In addition, 63.3% of people are concerned about security. The significant gap between these primary concerns and secondary ones such as record-keeping (43.3%), indicates that speed and security should come first in any new system.

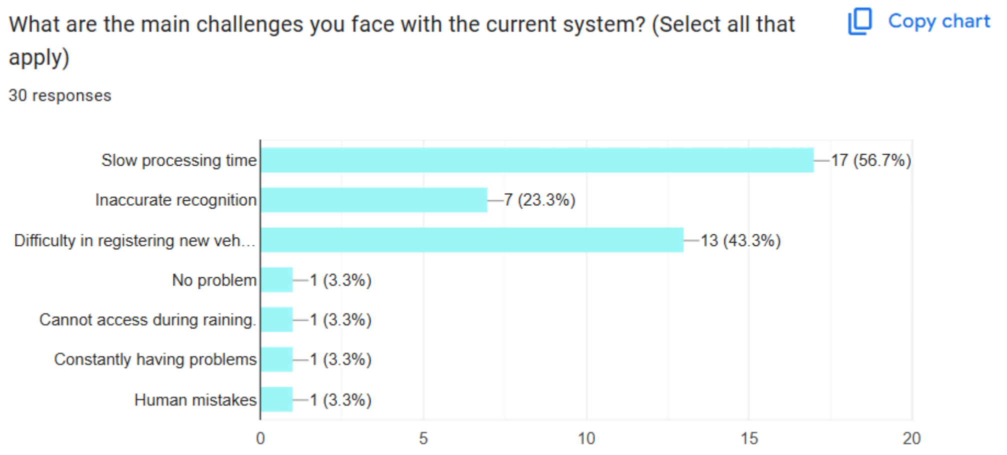


Figure 4.6: Main challenges in current system

Based on 30 survey results, the horizontal bar chart illustrates the main issues residents have with their current vehicle access system. Rapid recognition and gate response times should be the top priorities of any improvement according to the majority (56.7%) who list delayed processing as their worst concern. Registration of new vehicle issues are the second most frequent problem (43.3%) whixh suggested that car registration procedures need to be made more efficient. The fact that nearly a quarter (23.3%) report recognition issues points to areas where reliability of the system could be improved by using stronger hardware or algorithms.

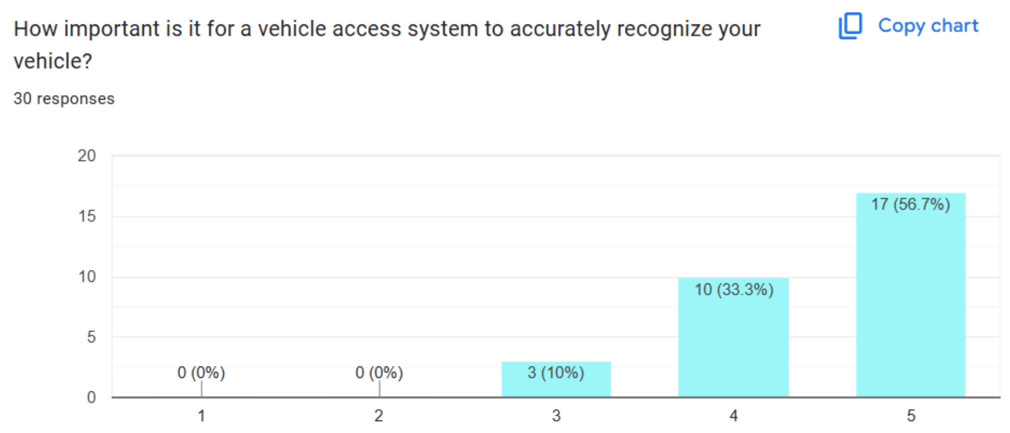


Figure 4.7: Importance of accuracy of vehicle access system

Based on 30 survey results, the bar chart calculates the the residents' assessed importance of accurate vehicle recognition on a 5-point scale ranging from "Not Important" to "Very Important". This is the top priority for system enhancements because most people (90 percent combined for ratings 4-5) believe that exact recognition is essential. Residents appear to have little tolerance for recognizing failures as seen by the total lack of bad ratings (0% for 1-2). The system must function flawlessly for the majority of users as indicated by the 56.7% rating at the top.

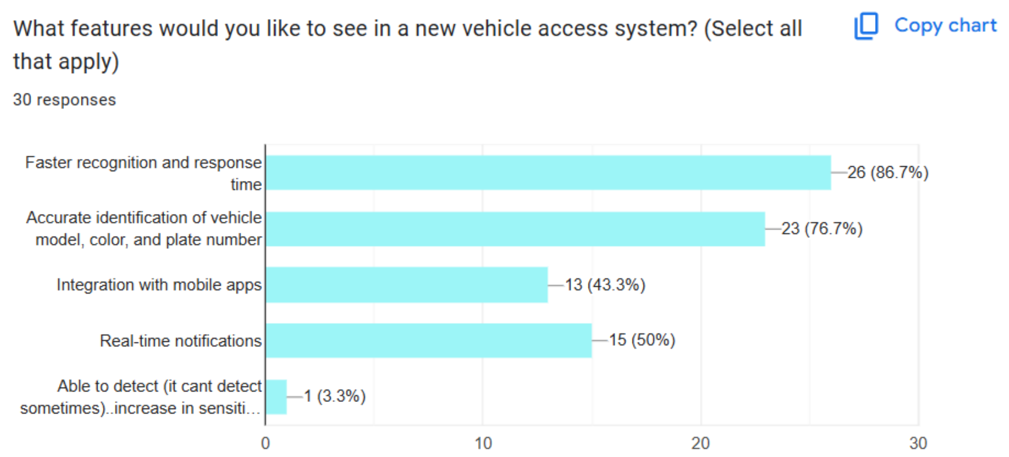


Figure 4.8: New features for future vehicle access system

According to the survey results from 30 participants, there are definite goals for a new vehicle access system. Specifically, 76.7% of respondents emphasize the necessity for precise vehicle information identification, and 86.7% require faster recognition and response times. Real-time notifications (50%) and mobile app integration (43.3%) are significant secondary preferences, while a small 3.3% expressed doubts regarding the accuracy of detection. These results highlight customers' preference for speed and accuracy above convenience features, which directs development efforts toward improving response efficiency and recognition algorithms. For optimal effect, the system should prioritize meeting these core needs before adding digital capabilities like app integration and notifications as supplementary features.

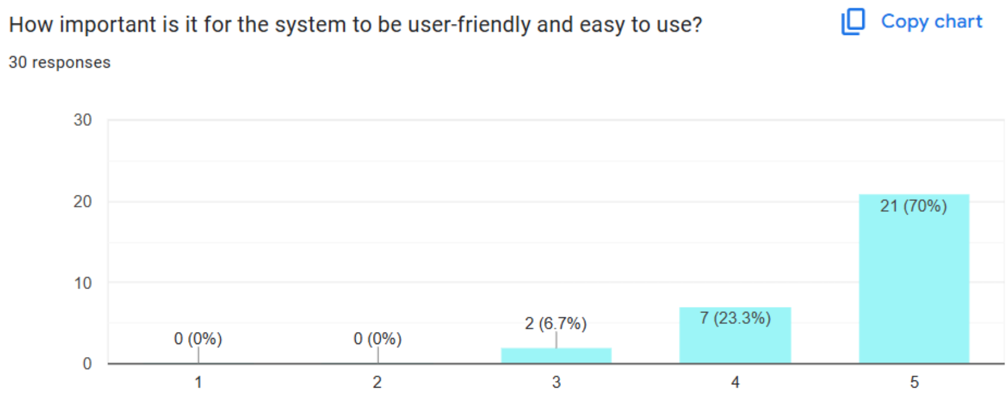


Figure 4.9: Importance of having user-friendly system

According to the survey's findings, user-friendliness is rated as critically important (5/5) by 70% of respondents (21 out of 30), and important (4/5) by another 23.3% (7 respondents). 93.3% of users believe that ease of use is a high-priority feature overall, compared to just 6.7% (2 respondents) who gave it a neutral assessment (3/5) and no one who thought it was unimportant (0% for 1-2). With almost all responders highlighting this component as being just as crucial as technical performance elements like identification accuracy, it is evident that any new vehicle access system must prioritize user-friendly design and easy operation in order to fulfill user expectations. According to the statistics, in order to achieve broad acceptance and user satisfaction, development efforts should strike a balance between advanced functionality and easily navigable interfaces.

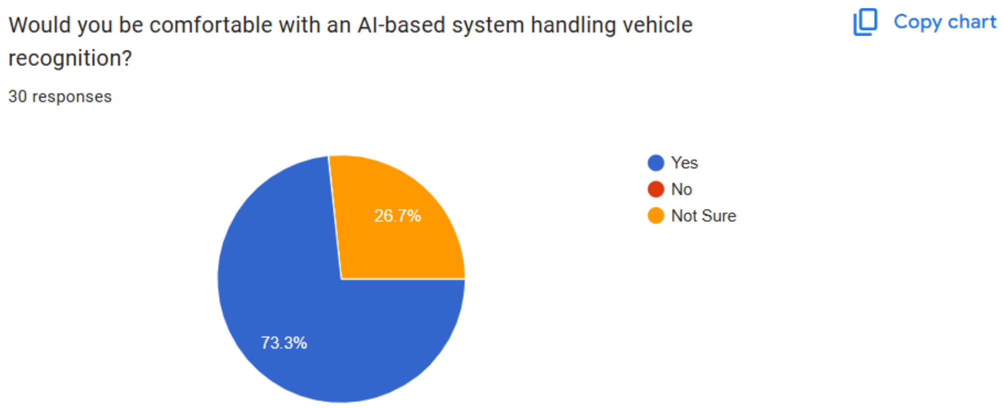


Figure 4.10: Acceptance of AI-based vehicle recognition system

The results of the poll indicate that AI-based car recognition is widely accepted as there is 73.3% of respondents (22/30) said they were comfortably using the system, while 26.7% (8/30) said they were unsure. Importantly, ther e is 0% said they were against the technology. Although the remaining uncertain quarter suggests the need for targeted education demonstrating the system's reliability through test cases, clear refuse mechanisms during rollout to build trust, and visible performance metrics to convince afraid users, the overwhelming positive response suggests high receptiveness to AI implementation. As long as openness is maintained at the forefront of deployment, the total lack of "No" votes presents a unique chance to test AI solutions with minimal neutral opinions.

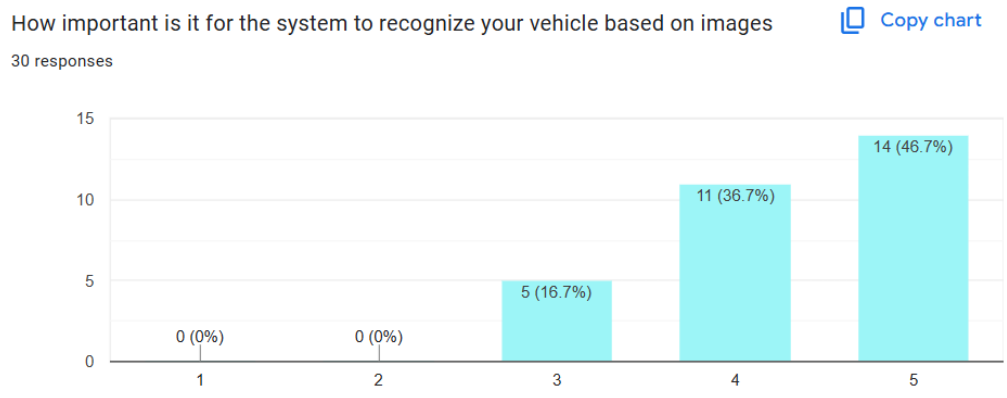


Figure 4.11: Importance of recognisation of vehicle using image

According to the survey, image-based vehicle recognition has been given an extremely high degree of importance with 46.7% of respondents (14/30) rated it as absolutely crucial (5/5) and 36.7% of respondents (11/30) rated it as very significant (4/5). Together, 83.4% of users rated this feature as essential (4+), 16.7% (5/30) gave it a neutral grade (3/5), and 0% downplayed its significance (no 1-2 ratings). This research emphasizes that most users believe reliable picture recognition which requires high-accuracy implementation.

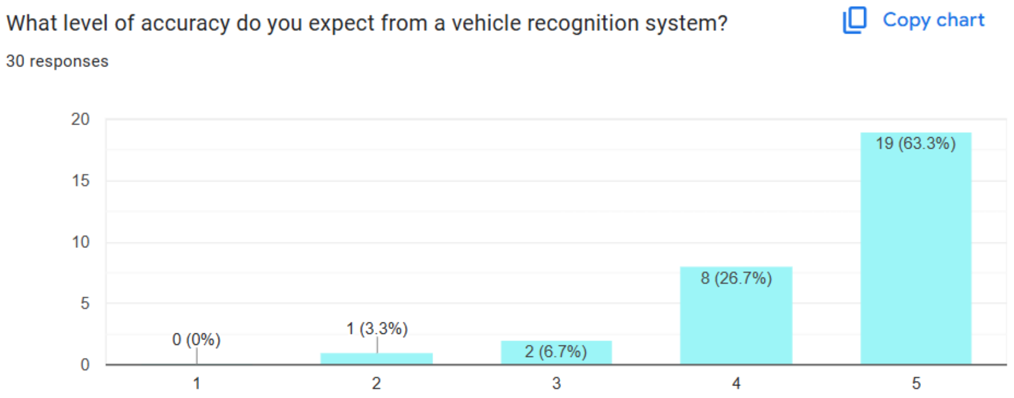
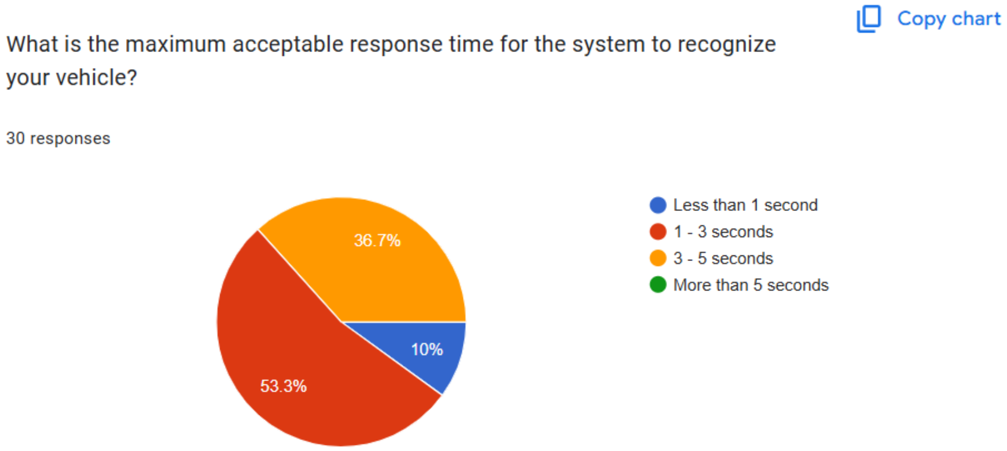


Figure 4.12: Expectation of accuracy on vehicle recognition system

With 63.3% of respondents (19/30) expecting flawless performance (rating 5/5) and another 26.7% (8/30) expecting near-perfect operation (4/5), the survey results show that 90% of users have strict accuracy expectations for vehicle identification systems. Users consider precision to be non-negotiable as evidenced by the fact that only 10% (3 respondents) were fairly tolerant (ratings 2-3) and 0% would tolerate subpar performance. In order to meet the 63.3% who expect zero errors, these findings require the adoption of redundant verification mechanisms, thorough real-world testing, and transparent accuracy reporting. This is because even small recognition failures have the potential to damage systemic trust. According to the research, accuracy is the most important factor that determines user acceptance, compared to all other aspects.



What is the maximum acceptable response time for the system to recognize your vehicle?

30 responses

Figure 4.13: Maximum acceptable time for vehicle recognisation

According to the poll, 90% of users (27 out of 30 respondents) want almost instantaneous vehicle detection, while 36.7% (11 users) accept responses of 3-5 seconds and 53.3% (16 users) are okay with 1-3 seconds. Only 10% (3 users) would need less than a second, and 0% would allow delays longer than five seconds. This establishes a strict 3-second performance criteria for implementation, recommending that the needs of optimize ALPR algorithms for processing in real-time and test load during periods of high traffic. The total lack of tolerance for sluggish answers (>5s) suggests that user pleasure depends on speed just as much as accuracy, demanding a balanced investment in both software and hardware.

4.3 User Requirements Specifications (URS)

This section provides a representation of the user requirement specification, which may be broken down into the two primary categories of "functional requirements" and "non-functional requirements."

4.3.1 Functional Requirements

Role	ID	Module	Functional Requirements
Residents	SRS001	Registration	The system shall allow residents to register an account with their personal information such as house number, name, email and password.

	SRS002	Login	The system shall allow residents to log in by email and password.
	SRS003	Manage Profile	The system shall allow residents to update their profile information such as name, phone number, address and profile image.
	SRS004	Manage Vehicle	The system shall allow residents to register their vehicle with details like plate number, colour, model, and manufacturer.
	SRS005		The system shall allow residents to update their registered vehicle information such as plate number, colour, and model.
	SRS006		The system shall allow residents to delete their registered vehicle.
	SRS007	Notifications	The system shall send notification to residents when their vehicle enters/exits the premises.
	SRS008	Suspicious Activity Alerts	The system shall alert residents if a suspicious event such as clone vehicle plate with different colour or model is detected.
	SRS09	Manage Visitor Pass	The system shall allow residents to generate a visitor pass with different time-limited for visitor registration.
	SRS010		The system shall allow residents to update a visitor pass for editing incorrect information or activate again the visitor pass.

	SRS011		The system shall allow residents to delete a registered visitor pass.
	SRS012	Reset Password	The system shall allow residents to reset their account password.
	SRS013	Vehicle Logs	The system shall allow residents to view a history of their registered vehicle’s entries/exits.
	SRS014		The system shall allow residents to search for history of their registered vehicle’s entries/exits by record ID.
	SRS015		The system shall allow residents to filter the log records by event type and suspiciousness.
	SRS016		The system shall allow residents to sort their registered vehicle’s log records by alphabetical characters in ascending and decreasing order.
Security Guards	SRS017	Login	The system shall allow security guard to log in by guard ID and password.
	SRS018	Reset Password	The system shall allow security guard to reset their account password.
	SRS019	Dashboard	The system shall display real-time data analysis from the log records of vehicle entries or exits.
	SRS020	Vehicle Logs	The system shall allow security guards to search for history of vehicle’s entries/exits by record

			ID, vehicle colour, and vehicle model.
	SRS021		The system shall allow security guards to filter the log records by record status.
	SRS022		The system shall allow security guards to sort the log records by alphabetical characters in ascending and decreasing order.
	SRS023	Suspicious Events	The system shall highlight suspicious vehicles like duplicated license plate for manual checking.
	SRS024	Real-Time Alerts	The system shall notify guards of flagged vehicles via audio/visual alerts on the dashboard.
	SRS025	Manage Profile	The system shall allow security guards to update their profile information such as name and profile image.

4.3.2 Non-Functional Requirements

Category	ID	Non-functional requirement
Performance	NFR001	The system shall return search results within 2 seconds for 90% of queries under normal load.
	NFR002	Real-time notifications shall be displayed on the user interface within 3 seconds of detection.
Availability	NFR003	The system shall be available 90% of the time over a 30-day period.

Security	NFR004	User passwords shall be stored using AES-256 or bcrypt hashing, and never in plain text.
	NFR005	All HTTP requests must be served over HTTPS with TLS encryption.
Usability	NFR006	First-time users shall be able to register and log in within 3 minutes on average.
	NFR007	The user interface shall display and function correctly on modern web browsers.
Reliability	NFR008	The system shall gracefully handle unexpected input or component failure by providing meaningful error messages instead of crashing.
	NFR009	The system shall retry failed communication with the database or external services up to 3 times before notifying the user.
	NFR010	The system shall log all critical errors to a centralized log file or monitoring service with timestamps and error severity level.
	NFR011	The system shall log all user login attempts and key actions with timestamps for auditing and recovery purposes.

4.4 System Use Case

Use cases are a collection of behaviors that explain how users including staff members, event coordinators, kids and business owners interact with systems that have been put into place. System use cases gave users specific instructions on how to achieve their goals within the real system. It is used in the analysis phase to find, specify and make apparent the functional needs from the viewpoint of the end users, as well as the interdependencies between use cases. A use case diagram and description will be included in this section.

4.4.1 Use Case Diagram

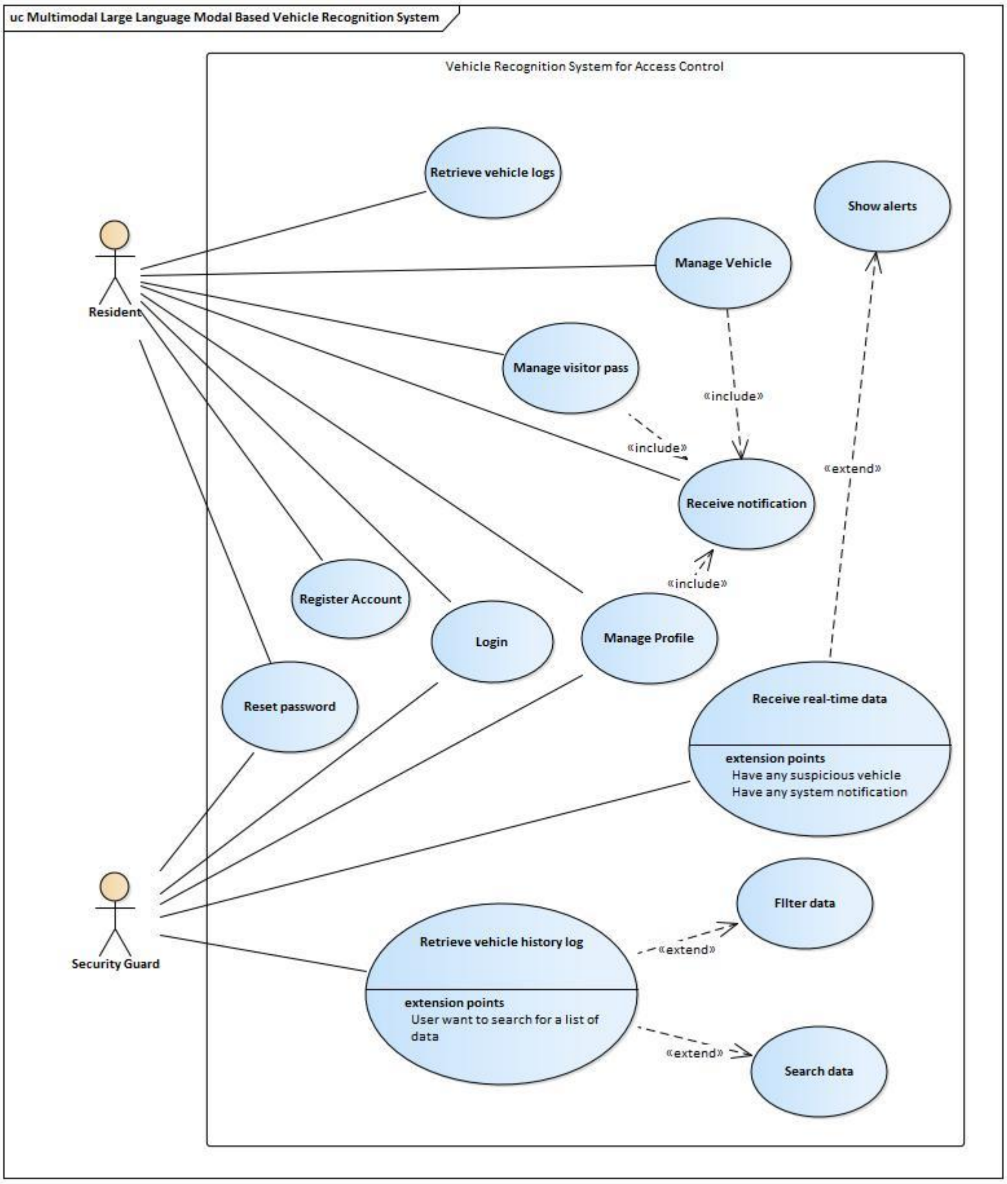


Figure 4.14: Use Case Diagram

4.4.2 Use Case Description

Table 4. 1 : Use Case Description for Retrieve Vehicle Logs

Use Case Name: **Retrieve Vehicle Logs**	ID: **USC001**	Importance Level: **High**
Primary Actor: **Resident**	Use Case Type: **Detailed, real**	
Stakeholders and Interests: **N/A**		
Brief Description: **This use case describes the process of retrieving a list of vehicle logs owned by resident by sorting the data based on vehicle plate, vehicle make and model, event type, activity status or date by the resident.**		
Trigger: **The resident wants to retrieve a collection or a list of his/her vehicle logs.**		
Relationships: Association : **Resident** Include : **N/A** Extend : **N/A** Generalization : **N/A**		
Normal Flow of Events: 1. The system retrives all the vehicle logs that owned by the vehicles registered of resident. 2. The system displays all the information of the vehicle logs. 3. The user input partial information of vehicle logs. Continue to S-1. 4. The system display the search result. Continue to S-2.		

Sub-flows**:**

S-1 Perform 3.1 or 3.2 or 3.3 or 3.4 or 3.5 or 3.6

3.1 If user choose to search by vehicle plate:

3.1.1 The system allow user to key in the vehicle plate. Continue to flow 4.

3.2 If user choose to search by vehicle make and model:

3.2.1 The system allow user to key in the vehicle make and model. Continue to flow 4.

3.3 If user choose to filter by event type entry:

3.3.1 The system allow user to select the event type entry. Continue to flow 4.

3.4 If user choose to search by event type exit:

3.4.1 The system allow user to select the event type exit. Continue to flow 4.

3.5 If user choose to search by all activity:

3.5.1 The system allow user to select the all activity. Continue to flow 4.

3.6 If user choose to search by suspicious activity:

3.6.1 The system allow user to select the suspicious activity. Continue to flow 4.

3.7 If user choose to search by date:

3.7.1 The system allow user to select the date. Continue to flow 4.

S-2 Perform 4.1 or 4.2

4.1 If at least one vehicle log is found:

4.1.1 The system displays all the brief information of the list of vehicle logs. Continue to flow 5.

4.2 If no vehicle log is found:

4.2.1 The system displays no log found message.

Alternate/Exceptional Flows:

Table 4. 2 : Use Case Description for Manage Visitor Pass

<table>
<tr><td>Use Case Name:
<b>Manage Visitor Pass</b></td><td>ID<b>: USC002</b></td><td>Importance Level<b>: High</b></td></tr>
<tr><td>Primary Actor<b>: Resident</b></td><td colspan="2">Use Case Type<b>: Detailed, real</b></td></tr>
<tr><td colspan="3">Stakeholders and Interests<b>: N/A</b></td></tr>
<tr><td colspan="3">Brief Description<b>: This use case describes the process of resident retrieving a list of visitor pass.</b></td></tr>
<tr><td colspan="3">Trigger<b>: The resident wants to retrieve a list of visitor pass.</b></td></tr>
<tr><td colspan="3">Relationships:
Association : <b>Resident</b>
Include : <b>Receive notification</b>
Extend : <b>N/A</b>
Generalization : <b>N/A</b></td></tr>
<tr><td colspan="3">Normal Flow of Events<b>:</b>
1. The system retrives all the visitor access pass.
2. The system displays all the information of the visitor pass. Continue to S-1.
3. The user take action on the visitor pass. Continue to S-2.</td></tr>
<tr><td colspan="3">Sub-flows<b>:</b>
<b>S-1 Perform 2.1 or 2.2</b>
2.1 If the visitor pass is expired:
2.1.1 The system display the visitor pass with a activate again button and delete button. Continue to flow 3.
2.2 If the visitor pass is still valid:
2.2.1 The system display the visitor pass with a edit button and delete button. Continue to flow 3.
<b>S-2 Perform 3.1 or 3.2 or 3.3</b>
3.1 If the user choose to delete a visitor pass:</td></tr>
</table>

3.1.1 The system displays confirmation delete message 3.1.2 The user delete the visitor pass by selecting confirm delete. 3.1.3 The system prompt a delete successful message to user. Continue to USC006. 3.2 If the user choose to update a visitor pass: 3.2.1 The system displays the update visitor pass page with its information to the user. 3.2.2 The user edit the visitor pass information and click on update visitor pass button. 3.2.3 The system save the new visitor pass information and prompt a update successful message to user. Continue to USC006. 3.3 If the user want to add new visitor pass: 3.3.1 The user click on the add visitor button. 3.3.2 The system route user to add visitor page and allow user to enter the visitor name, phone number, vehicle license plate, color, make, model year and date range. 3.3.3 The system save the new visitor pass information and prompt a update successful message to user. Continue to USC006.
Alternate/Exceptional Flows:

Table 4. 3 : Use Case Description for Reset Password

Use Case Name**:** **Reset Password**	ID**: USC003**	Importance Level**: High**
Primary Actor**: Resident, Security Guard**	Use Case Type**: Detailed, real**	
Stakeholders and Interests**: N/A**		
Brief Description**: This use case describes the process of reset the account password.**		

Trigger**: The user clicked the “Forgot Password” link on the login page.**

Relationships:

Association : **Resident, Security Guard**

Include : **N/A**

Extend : **N/A**

Generalization : **N/A**

Normal Flow of Events**:**

1. The user clicked the “Forgot Password” link on the login page.
2. The system displayed a page prompting the user to enter their registered email address.
3. The user entered their email address and submitted the request. Continue to S-1
4. The user opened the reset email and clicked the password reset link.
5. The system redirected the user to the reset password page.
6. The user entered a new password and confirmed it. Continue to S-2.

Sub-flows**:**

S-1 Perform 3.1 or 3.2

3.1 If the email input are not valid:

3.1.1 The system display error message. Continue to flow 3.

3.2 If the email input are valid:

3.2.1 The system sent a reset password link and prompt an alert to user about the reset password email has been sent. Continue to flow 4.

S-2 Perform 6.1 or 6.2

6.1 If the new password not meet security requirement:

6.1.1 The system displayed an error message. Continue to flow 6.

6.2 If the new password meet security requirement::

6.2.1 The system save the new password and route the user back to the login page.

Alternate/Exceptional Flows:

Table 4. 4 : Use Case Description for Login

<table>
<tr><td>Use Case Name: **Login**</td><td>ID: **USC004**</td><td>Importance Level: **High**</td></tr>
<tr><td colspan="2">Primary Actor: **Resident, Security Guard**</td><td>Use Case Type: **Detailed, real**</td></tr>
<tr><td colspan="3">Stakeholders and Interests: **N/A**</td></tr>
<tr><td colspan="3">Brief Description: **This use case describes the process of resident and security guard login the web vehicle recognition access control system.**</td></tr>
<tr><td colspan="3">Trigger: **The resident or security guard wants to access the vehicle recognition access control system.**</td></tr>
<tr><td colspan="3">Relationships:
Association : **Resident, security guard**
Include : **N/A**
Extend : **N/A**
Generalization : **N/A**</td></tr>
<tr><td colspan="3">Normal Flow of Events:
1. The user navigates to the login page.
2. The system allow user to key in their email and password.
3. The user enter their email and password. Perform S-1.
4. The system check the presence of any login before. Perform S-2.
5. The system prompt a login successful message to user.</td></tr>
<tr><td colspan="3">Sub-flows:
S-1 Perform 3.1 or 3.2
3.1 If the user enter correct credentials:
3.1.1 The user enter their email and password. Continue to flow 4.
3.2 If user user enter incorrect credentials:
3.2.1 The system prompt a login error message. Continue to flow 3.</td></tr>
</table>

S-2 Perform 4.1 or 4.2 4.1 If there is login record before: 4.1.1 The system kill the login session before and create a new one. Continue to flow 5. 4.2 If no login record before: 4.2.1 The system create a new login session. Continue to flow 5.
Alternate/Exceptional Flows:

Table 4. 5 : Use Case Description for Register Account

Use Case Name**:** **Register Account**	ID**: USC005**		Importance Level**: High**
Primary Actor**: Resident**		Use Case Type**: Detailed, real**	
Stakeholders and Interests**: N/A**			
Brief Description**: This use case describes the process of resident register a new account in vehicle recognition access control system.**			
Trigger**: The resident wants to register a new account in the vehicle recognition access control system.**			
Relationships: Association **: Resident** Include **: N/A** Extend **: N/A** Generalization **: N/A**			
Normal Flow of Events**:** 1. The user navigates to the register page. 2. The system allow user to key in their name, email and password. 3. The user enter their credentials. Perform S-1.			

4. The system check the presence of any account with same email. Perform S-2.
Sub-flows**:** **S-1 Perform 3.1 or 3.2** 3.1 If the user enter matched password: 3.1.1 The system pass the credentials to backend. Continue to flow 4. 3.2 If user user enter unmatched password: 3.2.1 The system prompt an error message for different password. Continue to flow 3. **S-2 Perform 4.1 or 4.2** 4.1 If there is similar email account before: 4.1.1 The system prompt the error message that the email is registered before. 4.2 If no similar email account before: 4.2.1 The system prompt a message about verify email and sent verification email to user. 4.2.2 The user click on the verification email. 4.2.3 The system register the user.
Alternate/Exceptional Flows:

Table 4. 6 : Use Case Description for Receive Notification

Use Case Name**:** **Receive Notification**	ID**: USC006**	Importance Level**: High**
Primary Actor**: Resident**	Use Case Type**: Detailed, real**	
Stakeholders and Interests**: N/A**		
Brief Description**: This use case describes the process of resident receiving notification in vehicle recognition access control system.**		

Trigger**: The resident wants to view the new notification in the vehicle recognition access control system.**
Relationships: Association **: Resident** Include **: N/A** Extend **: N/A** Generalization **: N/A**
Normal Flow of Events**:** 1. The system generate new notification on activity. 2. The system formats the notification content like title and description. 3. The system push the notification to the user. 4. The user can choose to view or dismiss the notification. Perform S-1.
Sub-flows**:** **S-1 Perform 4.1 or 4.2** 4.1 If the user choose to view the notification: 4.1.1 The application system will be open. 4.2 If the user choose not to view the notification: 4.2.1 The notification will be store in the notification page of application.
Alternate/Exceptional Flows:

Table 4. 7 : Use Case Description for Manage Vehicle

Use Case Name**:** **Manage Vehicle**	ID**: USC007**	Importance Level**: High**
Primary Actor**: Resident**	Use Case Type**: Detailed, real**	
Stakeholders and Interests**: N/A**		

Brief Description**: This use case describes the process of resident manage vehicles in vehicle recognition access control system.**

Trigger**: The resident wants to add, delete or edit the vehicles in the vehicle recognition access control system.**

Relationships:

Association : **Resident**
Include : **N/A**
Extend : **N/A**
Generalization : **N/A**

Normal Flow of Events**:**

1. The user navigate to the vehicle page.
2. The system display a list of registered vehicles.
3. The system display options that can be performed on vehicles.
4. The user chooses the option to add, delete or edit the vehicles. Continue to S-1.

Sub-flows**:**

S-1 Perform 4.1 or 4.2

4.1 If the user choose to add new vehicle:

4.1.1 The system allow user to enter new vehicle information like license plate, make, model, color and year.

4.1.2 The user enter the information needed for new vehicles. Continue to S-2.

4.2 If the user choose to edit vehicle:

4.2.1 The system displays the vehicle's information for the user to select the information that would be edited.

4.2.2 The user enters the edited vehicle information. Continue to S-3.

4.3 If the user choose to delete vehicle:

4.3.1 The system will prompt conformation delete message to user. Continue to S-4. **S-2 Perform 4.1.2.1 or 4.1.2.2** 4.1.2.1 If the user input is valid: 4.1.2.1.1 The system will add the new vehicle. Continue to USC006. Continue to flow 2. 4.1.2.2 If the user input is not valid: 4.1.2.2.1 The system prompts error message. 4.1.2.2.2 The user re-enter the new vehicle information. Continue to flow 4.1.2.1. **S-3 Perform 4.2.2.1 or 4.2.2.2** 4.2.2.1 If the user input is valid: 4.2.2.1.1 The system will update the vehicle information. Continue to USC006. Continue to flow 2. 4.2.2.2 If the user input is not valid: 4.2.2.2.1 The system prompts error message. 4.2.2.2.2 The user re-enter the edited vehicle information. Continue to flow 4.2.2.1. **S-4 Perform 4.3.1.1 or 4.3.1.2** 4.3.1.1 If the user confirm to delete vehicle: 4.3.1.1.1 The system will delete the vehicle selected by user. Continue to USC006. Continue to flow 2. 4.3.1.2 If the user does not confirm to delete vehicle: 4.3.1.2.1 The system will not execute the delete process. Continue to flow 2.
Alternate/Exceptional Flows:

Table 4. 8 : Use Case Description for Manage Profile

Use Case Name**:** **Manage Profile**	ID**: USC008**	Importance Level**: High**

Primary Actor**: Resident, security guard**	Use Case Type**: Detailed, real**
Stakeholders and Interests**: N/A**	
Brief Description**: This use case describes the process of resident and security guard manage their profiles in vehicle recognition access control system.**	
Trigger**: The resident or security guard wants to edit the profile informations in the vehicle recognition access control system.**	
Relationships: Association **: Resident, security guard** Include **: Receive notification** Extend **: N/A** Generalization **: N/A**	
Normal Flow of Events**:** 1. The user navigate to the profile page. 2. The system display the profile information. 3. The system display edit option that can be performed on profile. 4. The user chooses the option to edit the profile. 5. The system displays the profile's information for the user to select the information that would be edited. 6. The user enters the edited profile information. Continue to S-1.	
Sub-flows**:** **S-1 Perform 6.1 or 6.2** 6.1 If the user input is valid: 6.1.1 The system will update the profile information. Continue to USC006. Continue to flow 2. 6.2 If the user input is not valid: 6.2.1 The system prompts error message. 6.2.2 The user re-enter the edited profile information. Continue to flow 6.1.	

Alternate/Exceptional Flows:

Table 4. 9 : Use Case Description for Retrieve Vehicle History Log

Use Case Name: **Retrieve Vehicle History Log**	ID**: USC009**	Importance Level**: High**
Primary Actor**: Security Guard**		Use Case Type**: Detailed, real**
Stakeholders and Interests**: N/A**		
Brief Description**: This use case describes the process of security guard search vehicle history log in vehicle recognition access control system.**		
Trigger**: The security guard wants to search or filter the vehicle history log in the vehicle recognition access control system.**		
Relationships: Association **: Security Guard** Include **: N/A** Extend **: Filter data, Search data** Generalization **: N/A**		
Normal Flow of Events**:** 1. The user navigate to the vehicle logs page. 2. The system display a list of vehicle history logs. 3. The user can manage a particular vehicle logs by searching license plate or make or model or filtering through event type and activity status. Continue to E-1.		

Sub-flows:

S-1 Perform 3.1.2.1 or 3.1.2.2

3.1.2.1 If at least one vehicle history log is found:

3.1.2.1.1 The system display the search results in the page. Continue to flow 3.

3.1.2.2 If no vehicle history log is found:

3.1.2.2.2 The system displays the message no vehicle history log is found. Continue to flow 3.

Alternate/Exceptional Flows:

E-1 Perform 3.1 or 3.2 or 3.3

3.1 If the user choose to vehicle history logs:

3.1.1 The system allow user to enter the search information.

3.1.2 The user enter the search information needed. Continue to S-1.

3.2 If the user choose to filter vehicle history logs:

3.2.1 The system allow user to choose the filter categories.

3.2.2 The user choose the filter categories. Continue to S-1.

3.3 If the user choose not to search or filter vehicle history logs:

3.3.1 The system allow user to not enter any prompt or select any category. Continue to flow 2.

Table 4.10: Use Case Description for Retrieve Real-Time Data

<table>
<tr><td>Use Case Name:
<b>Retrieve Real-Time data</b></td><td colspan="2">ID: <b>USC010</b></td><td>Importance Level: <b>High</b></td></tr>
<tr><td colspan="2">Primary Actor: <b>Security Guard</b></td><td colspan="2">Use Case Type: <b>Detailed, real</b></td></tr>
<tr><td colspan="4">Stakeholders and Interests: <b>N/A</b></td></tr>
<tr><td colspan="4">Brief Description: <b>This use case describes the process of security guard retrieve real-time data and receive alerts in vehicle recognition access control system.</b></td></tr>
</table>

Trigger**: The security guard wants to retrieve real-time data and receive alerts in the vehicle recognition access control system.**

Relationships:

Association	**: Security guard**
Include	**: N/A**
Extend	**: Show alerts**
Generalization	**: N/A**

Normal Flow of Events**:**

1. The system retrieve the real-time data in database.
2. The system generate new alerts based on retrieve real-time data.
3. The system formats the alerts content like title and description.
4. The system push the alerts to the user.
5. The user can choose to view or dismiss the alerts. Perform E-1.

Sub-flows**:**

Alternate/Exceptional Flows:

E-1 Perform 5.1 or 5.2

5.1 If the user choose to view the alert:

5.1.1 The system will redirect the user to the alerts page.

5.2 If the user choose to not view the alert:

5.2.1 The alert will be store in the alerts page of web application.

4.5 Proposed System Flow & Prototype

This section will focus on proposed system flow whcihc represented by user interface flow diagram and the prototype screenshot for visualization purpose.

4.5.1 User Interface Flow Diagram

The flow of web and mobile application with same codebase for security guard and residents are shown in below user interface flow diagrams.

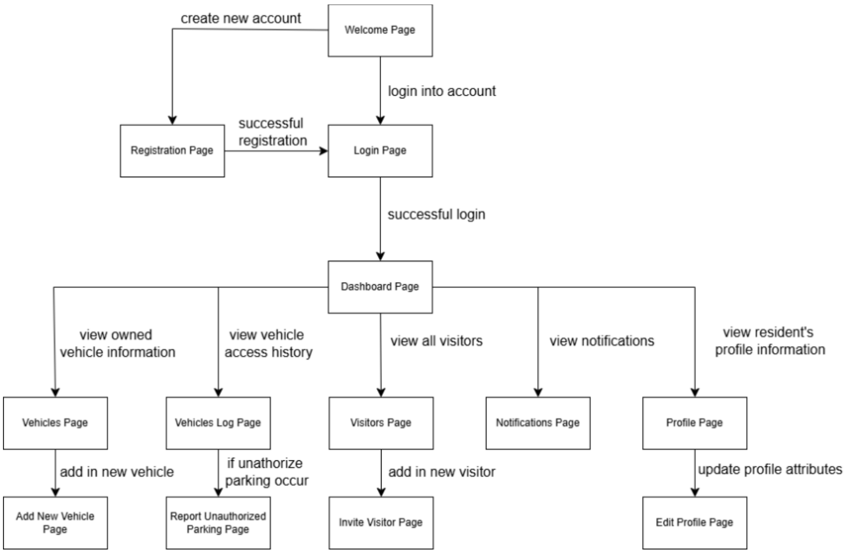


Figure 4.15: User interface flow diagram for residents

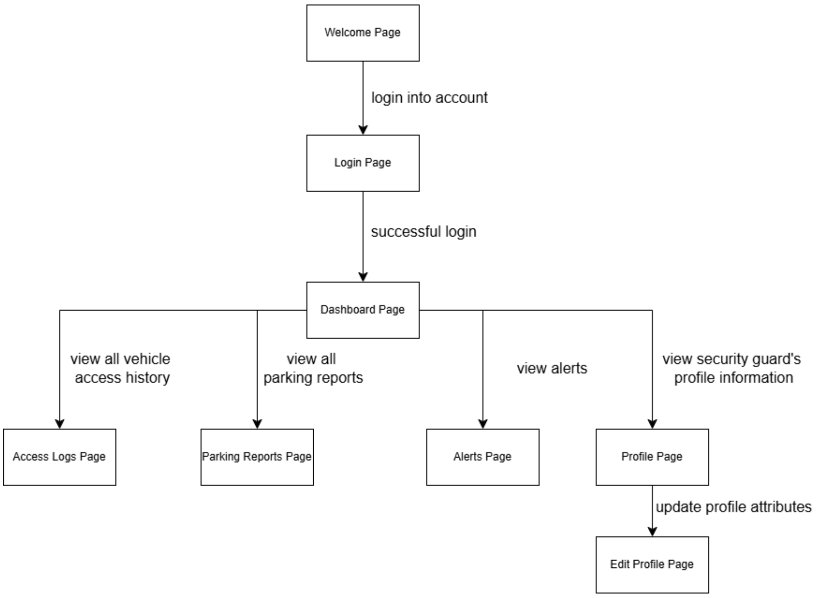


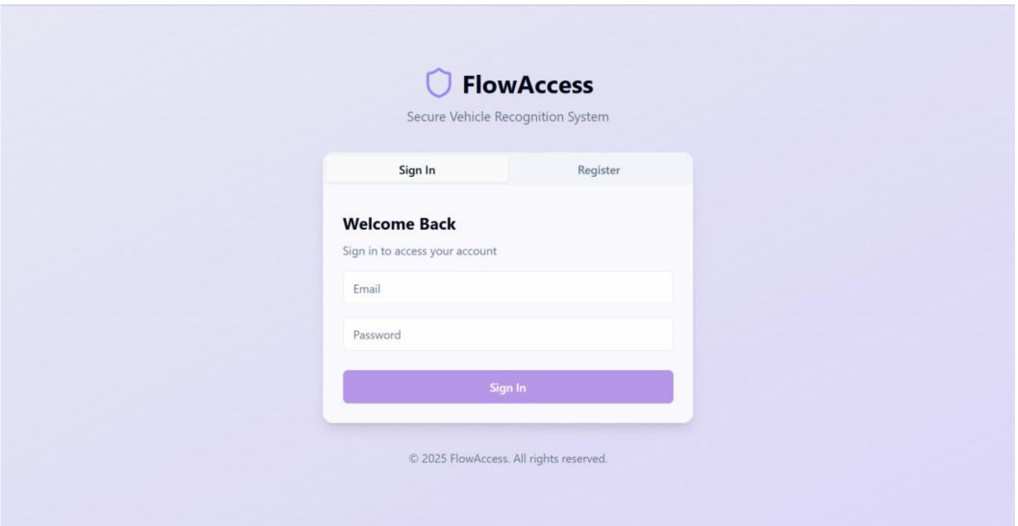
Figure 4.16: User interface flow diagram for security guards

4.6 Prototype

This section show all the web and mobile application's prototype screenshots with subsection, login or register, resident's page and security guard's page.

4.6.1 Login or register

4.6.1.1 Web View



FlowAccess

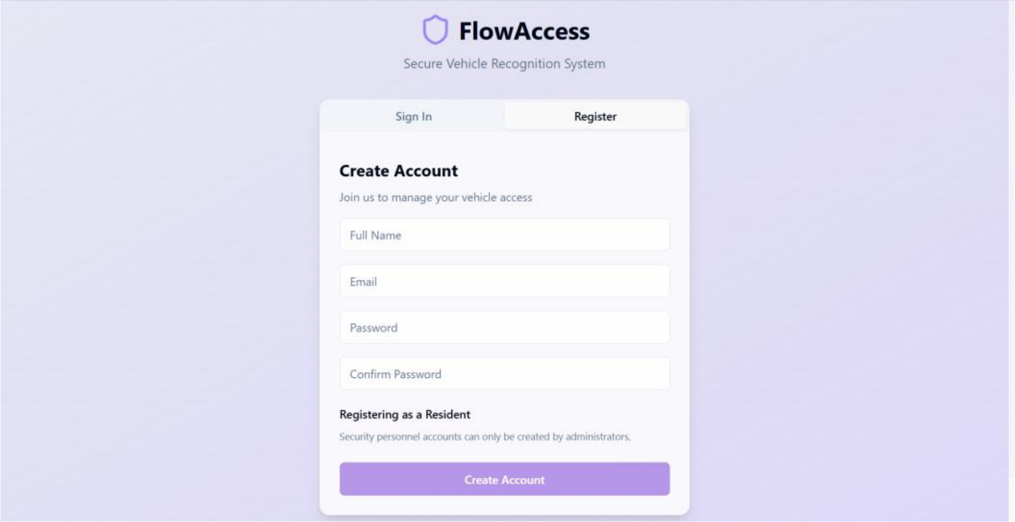
Secure Vehicle Recognition System

Welcome Back

Sign in to access your account

© 2025 FlowAccess. All rights reserved.

Figure 4.17: Login Web View



FlowAccess

Secure Vehicle Recognition System

Create Account

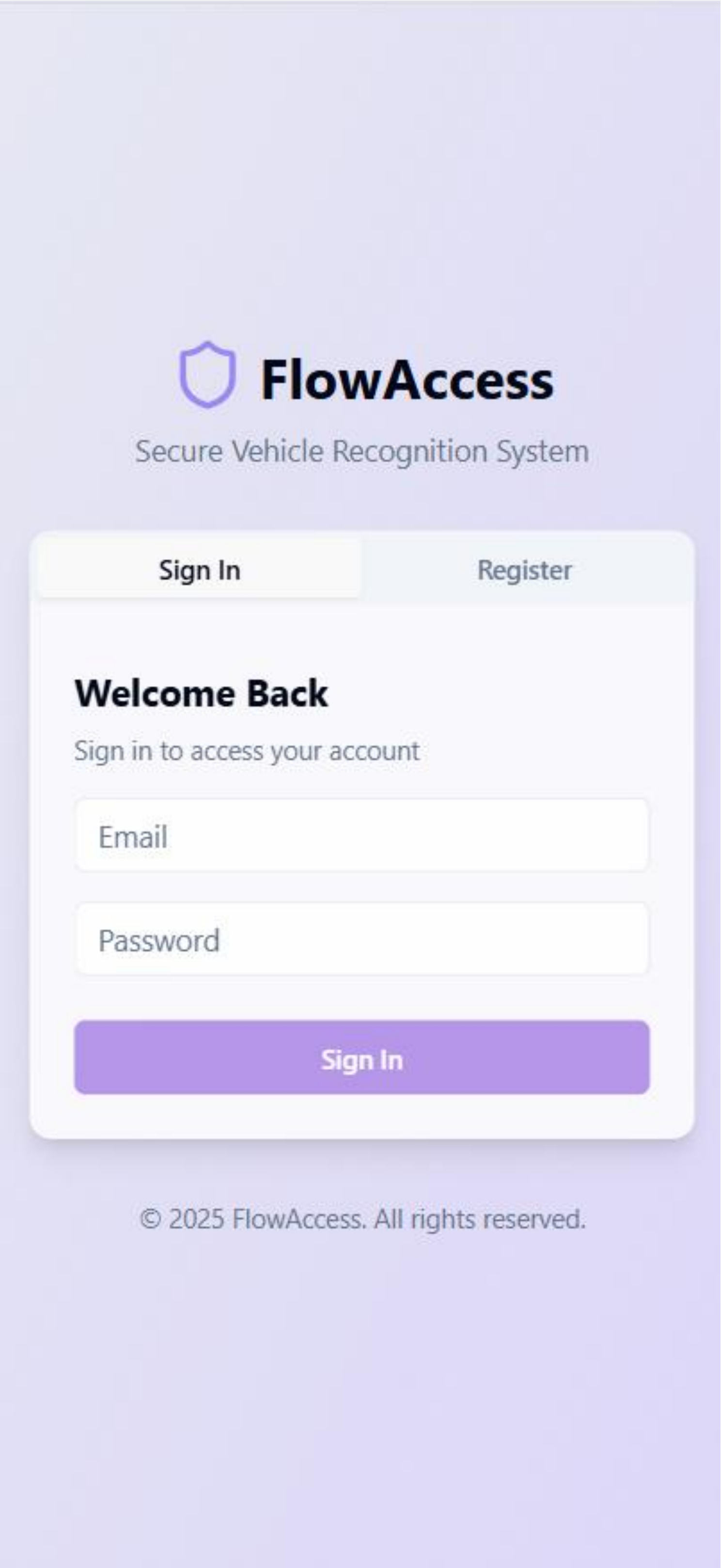
Join us to manage your vehicle access

Registering as a Resident

Security personnel accounts can only be created by administrators.

Figure 4.18: Register Web View

4.6.1.2 Mobile View



FlowAccess

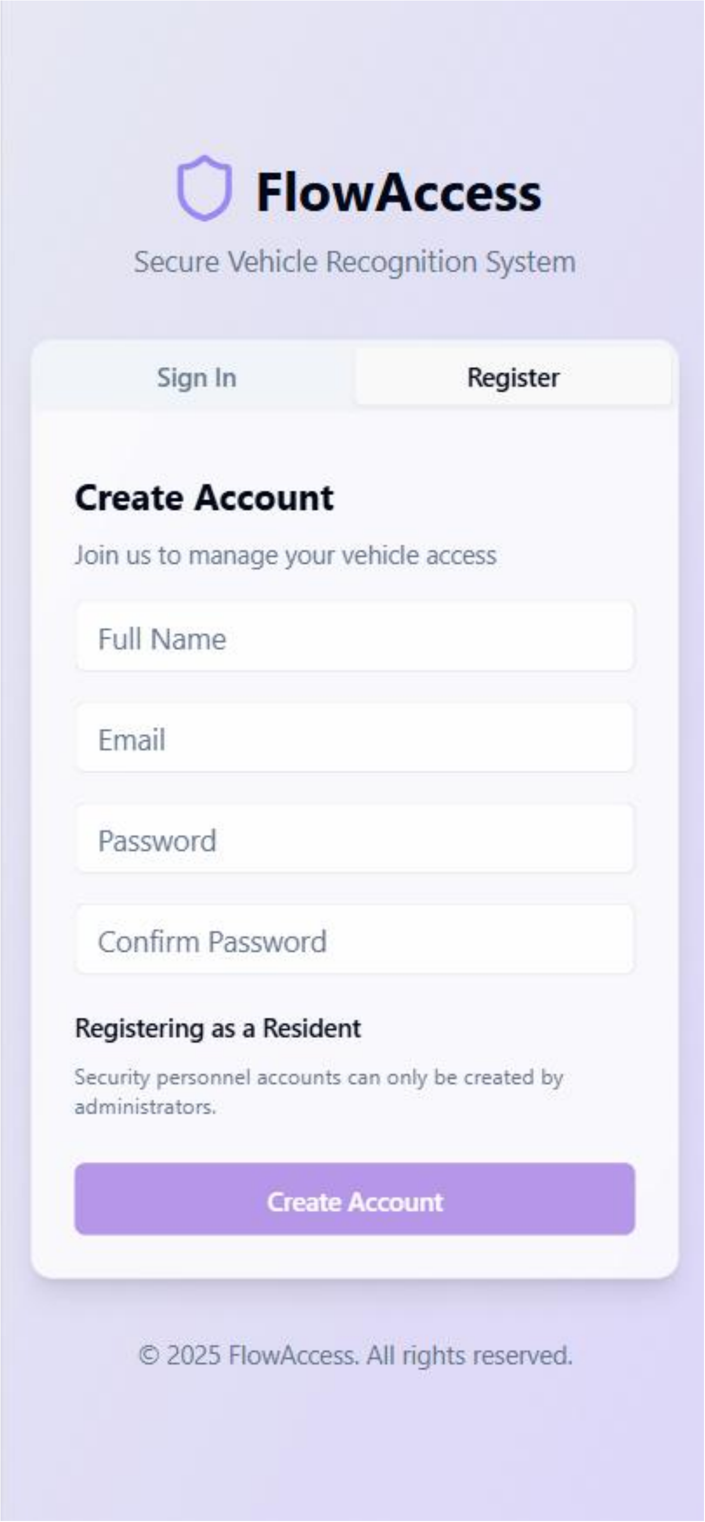
Secure Vehicle Recognition System

Welcome Back

Sign in to access your account

© 2025 FlowAccess. All rights reserved.

Figure 4.19: Login Mobile View



FlowAccess

Secure Vehicle Recognition System

Create Account

Join us to manage your vehicle access

Full Name

Email

Password

Confirm Password

Registering as a Resident

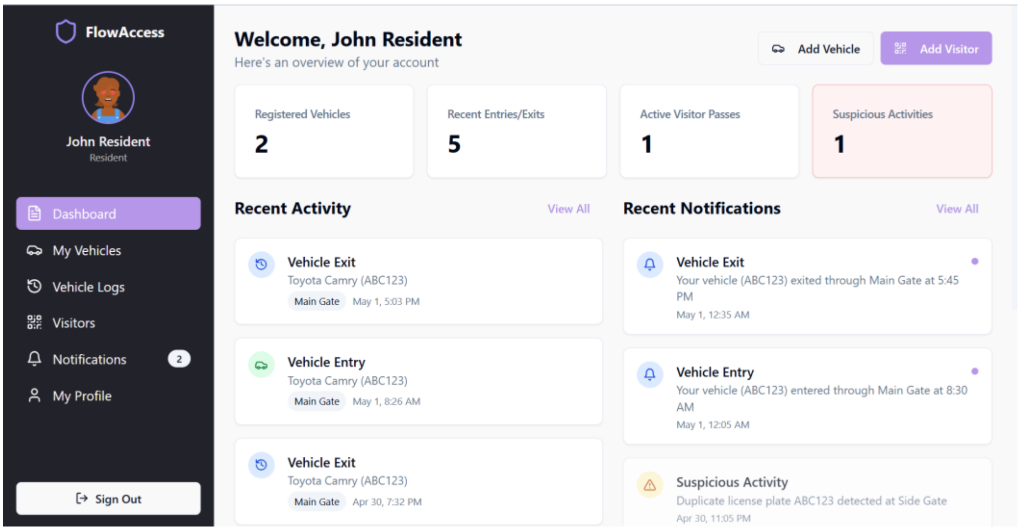
Security personnel accounts can only be created by administrators.

© 2025 FlowAccess. All rights reserved.

Figure 4.20: Register Mobile View

4.6.2 Resident's Page

4.6.2.1 Web View



Welcome, John Resident

Here's an overview of your account

Registered Vehicles	Recent Entries/Exits	Active Visitor Passes	Suspicious Activities
2	5	1	1

Recent Activity

Vehicle Exit
Toyota Camry (ABC123)
Main Gate May 1, 5:03 PM

Vehicle Entry
Toyota Camry (ABC123)
Main Gate May 1, 8:26 AM

Vehicle Exit
Toyota Camry (ABC123)
Main Gate Apr 30, 7:32 PM

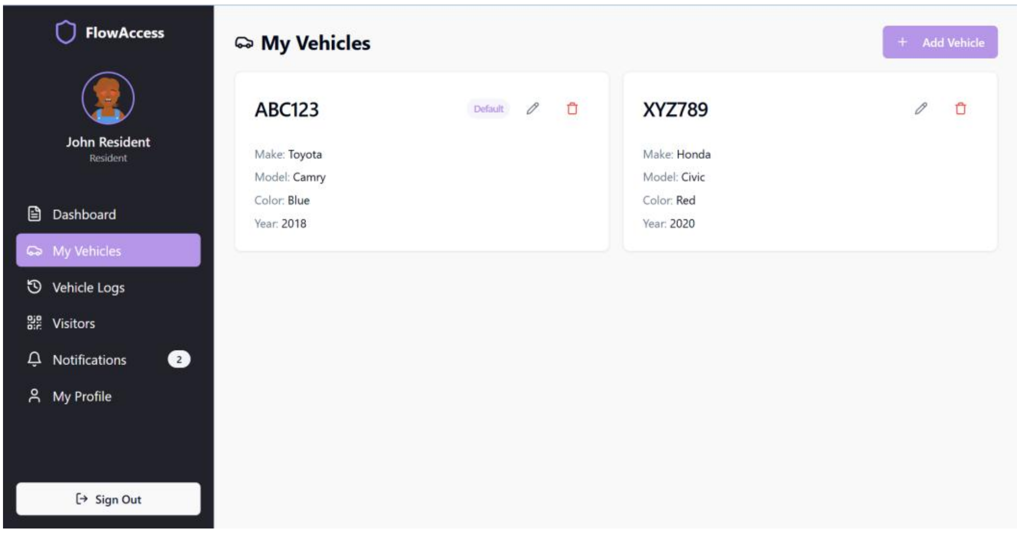
Recent Notifications

Vehicle Exit
Your vehicle (ABC123) exited through Main Gate at 5:45 PM
May 1, 12:35 AM

Vehicle Entry
Your vehicle (ABC123) entered through Main Gate at 8:30 AM
May 1, 12:05 AM

Suspicious Activity
Duplicate license plate ABC123 detected at Side Gate
Apr 30, 11:05 PM

Figure 4.21: Resident’s Dashboard Page Web View

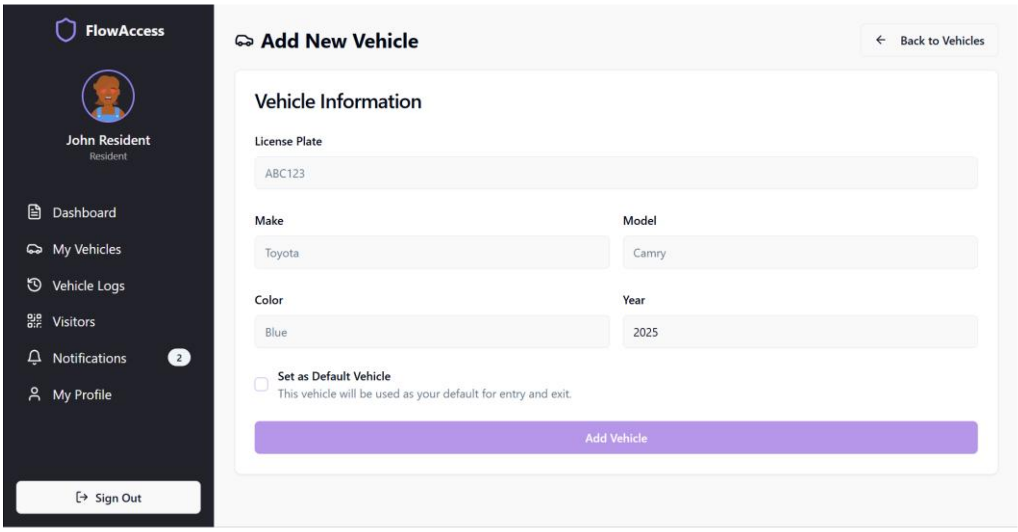


My Vehicles

ABC123
Make: Toyota
Model: Camry
Color: Blue
Year: 2018

XYZ789
Make: Honda
Model: Civic
Color: Red
Year: 2020

Figure 4.22: Resident's Vehicles Page Web View



Add New Vehicle

Vehicle Information

License Plate: ABC123

Make: Toyota

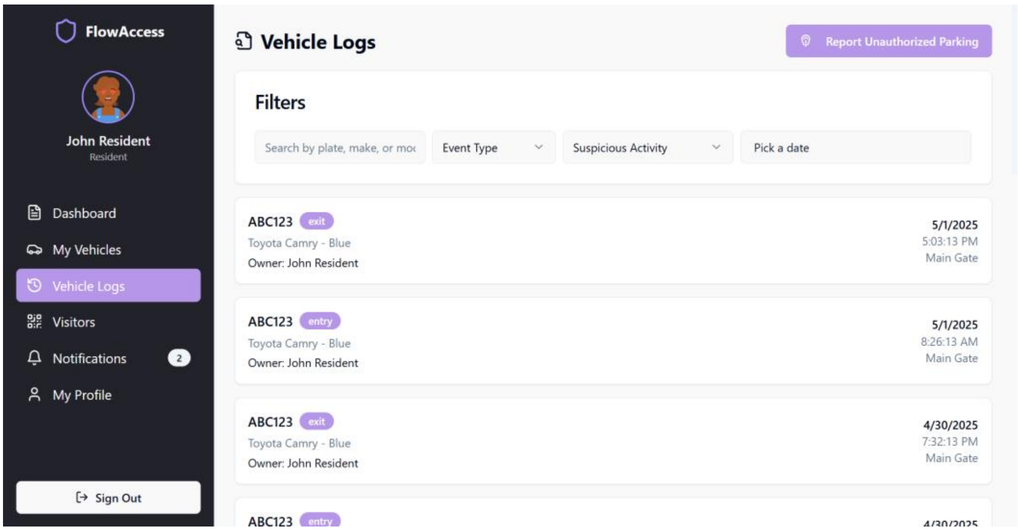
Model: Camry

Color: Blue

Year: 2025

□ Set as Default Vehicle
This vehicle will be used as your default for entry and exit.

Figure 4.23: Resident's Add Vehicle Page Web View



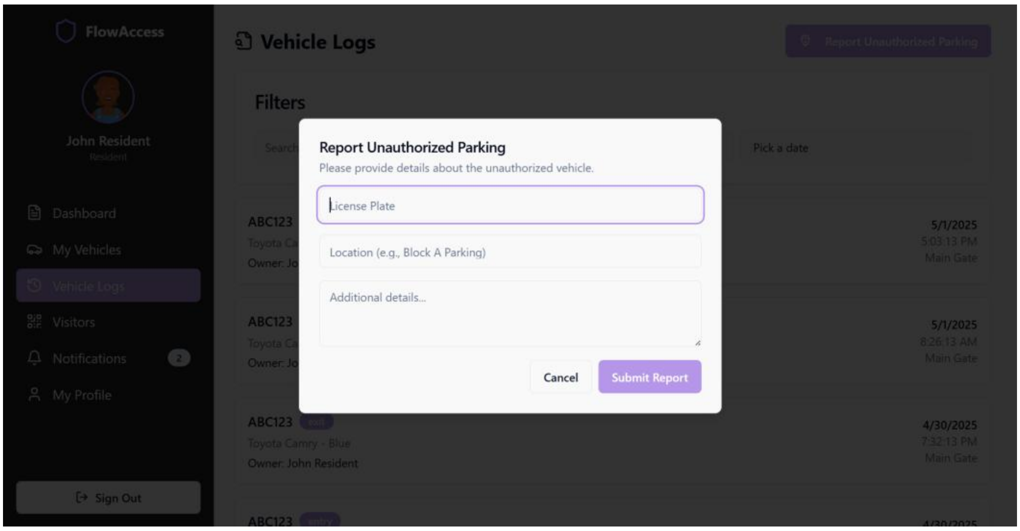
ABC123 exit
Toyota Camry - Blue
Owner: John Resident
5/1/2025
5:03:13 PM
Main Gate

ABC123 entry
Toyota Camry - Blue
Owner: John Resident
5/1/2025
8:26:13 AM
Main Gate

ABC123 exit
Toyota Camry - Blue
Owner: John Resident
4/30/2025
7:32:13 PM
Main Gate

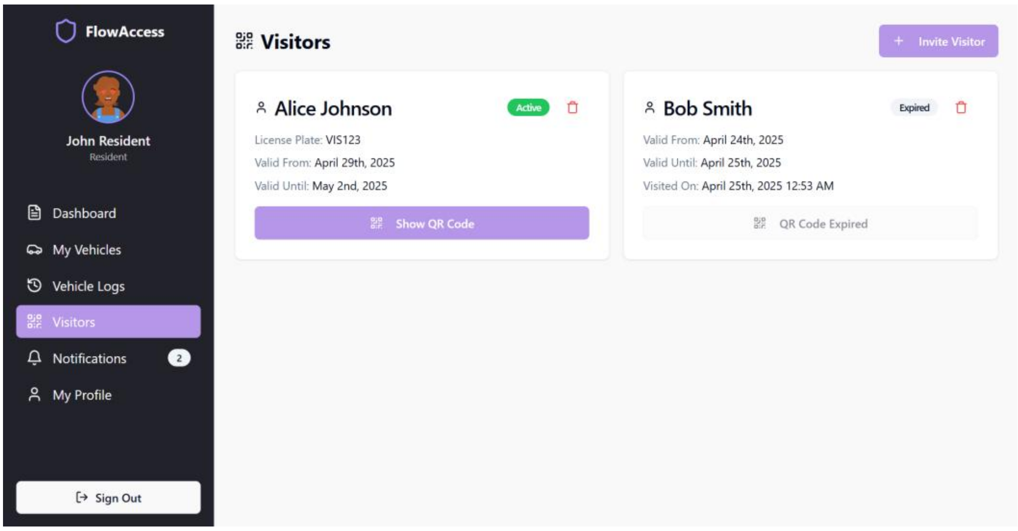
ABC123 entry
4/30/2025

Figure 4.24: Resident's Vehicle Logs Page Web View



Report Unauthorized Parking
Please provide details about the unauthorized vehicle.

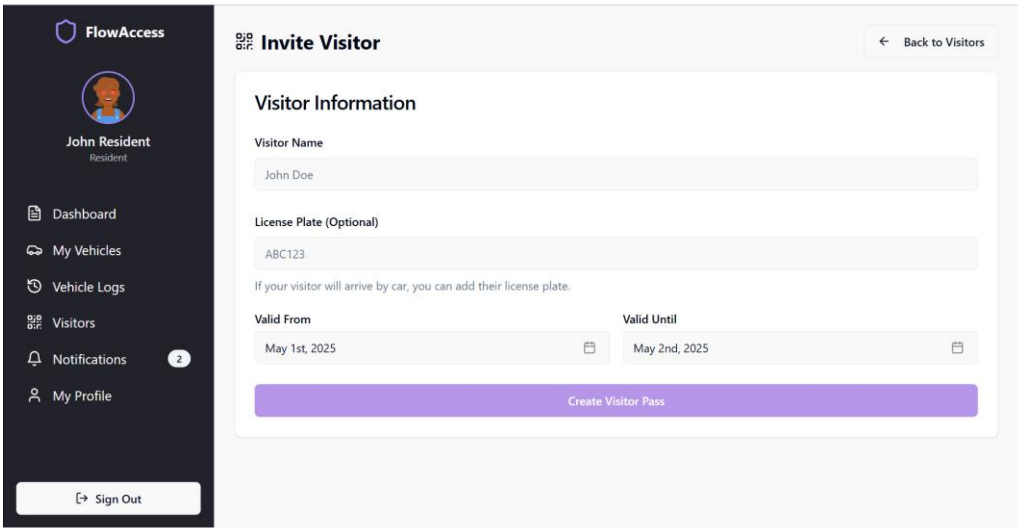
Figure 4.25: Resident's Report Unauthorized Parking Page Web View



Alice Johnson Active
License Plate: VIS123
Valid From: April 29th, 2025
Valid Until: May 2nd, 2025

Bob Smith Expired
Valid From: April 24th, 2025
Valid Until: April 25th, 2025
Visited On: April 25th, 2025 12:53 AM

Figure 4.26: Resident's Visitor Page Web View



Visitor Information

Visitor Name: John Doe

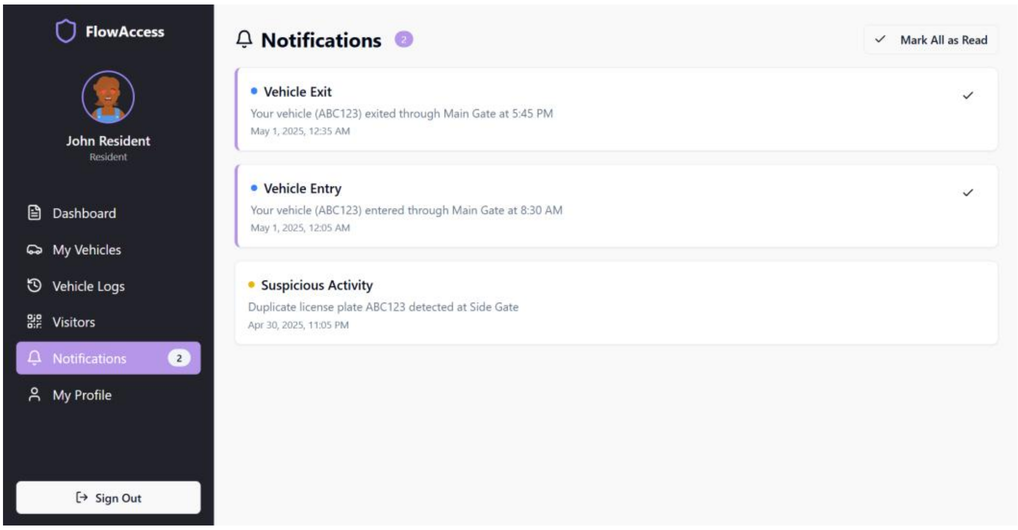
License Plate (Optional): ABC123

If your visitor will arrive by car, you can add their license plate.

Valid From: May 1st, 2025

Valid Until: May 2nd, 2025

Figure 4.27: Resident's Invite Visitor Page Web View

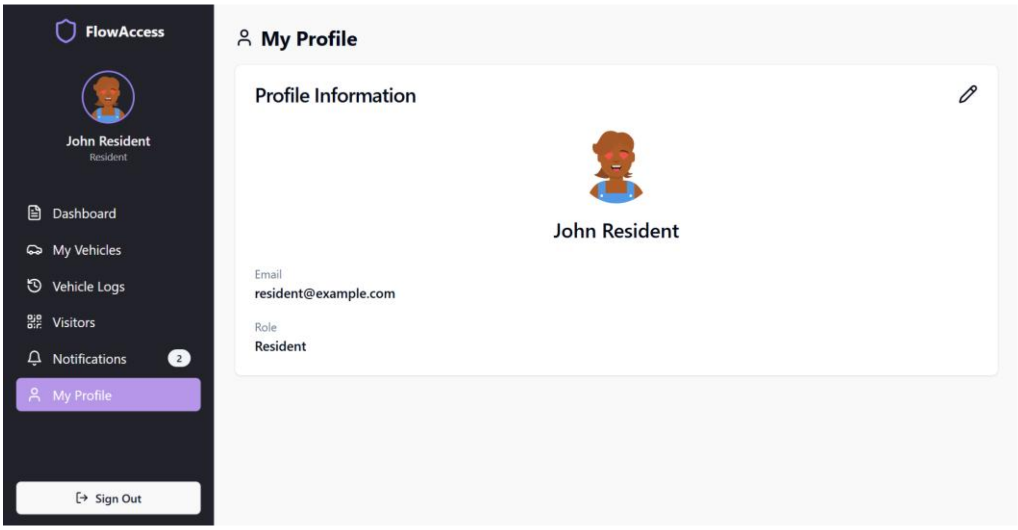


Vehicle Exit
Your vehicle (ABC123) exited through Main Gate at 5:45 PM
May 1, 2025, 12:35 AM

Vehicle Entry
Your vehicle (ABC123) entered through Main Gate at 8:30 AM
May 1, 2025, 12:05 AM

Suspicious Activity
Duplicate license plate ABC123 detected at Side Gate
Apr 30, 2025, 11:05 PM

Figure 4.28: Resident’s Notification Page Web View



Profile Information

John Resident

Email: resident@example.com

Role: Resident

Figure 4.29: Resident's Profile Page Web View

4.6.2.2 Mobile View

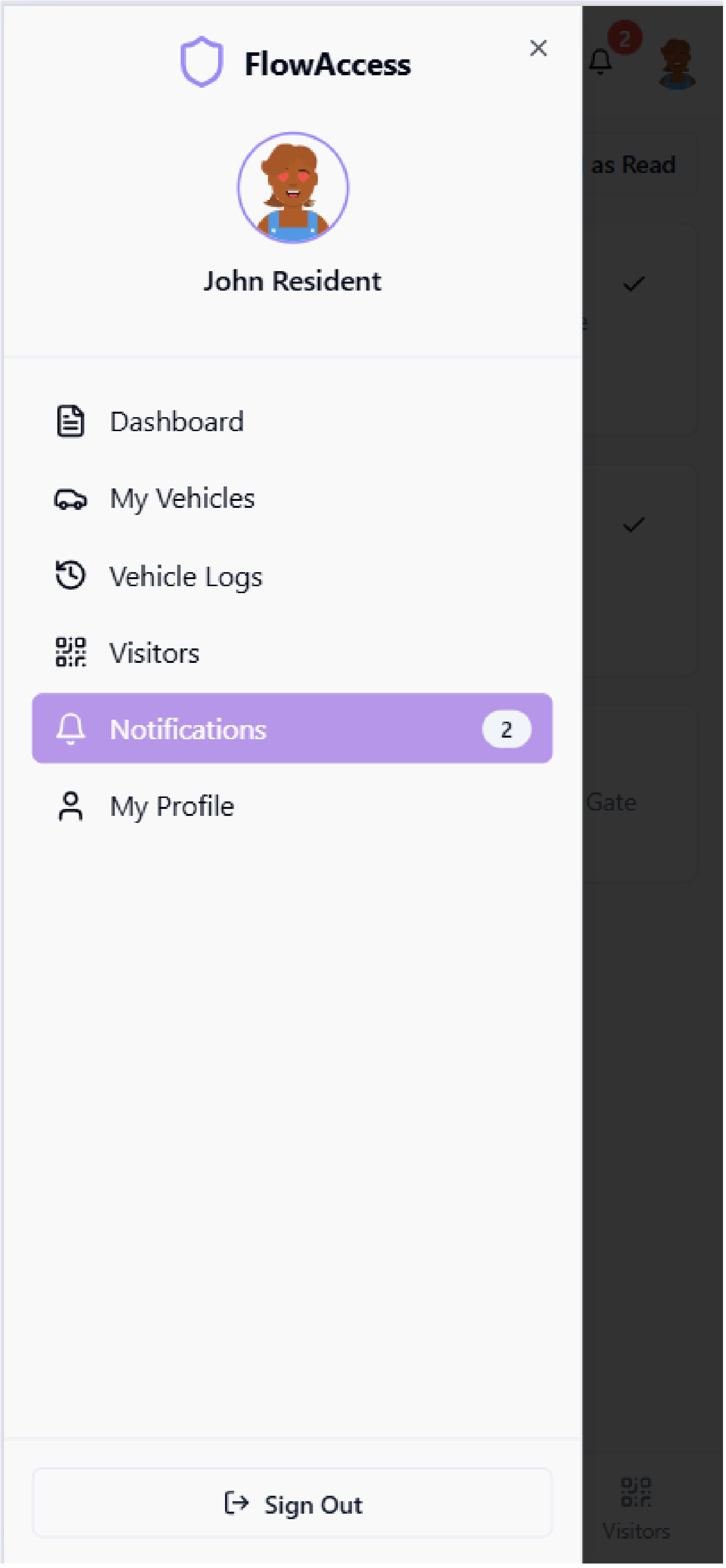
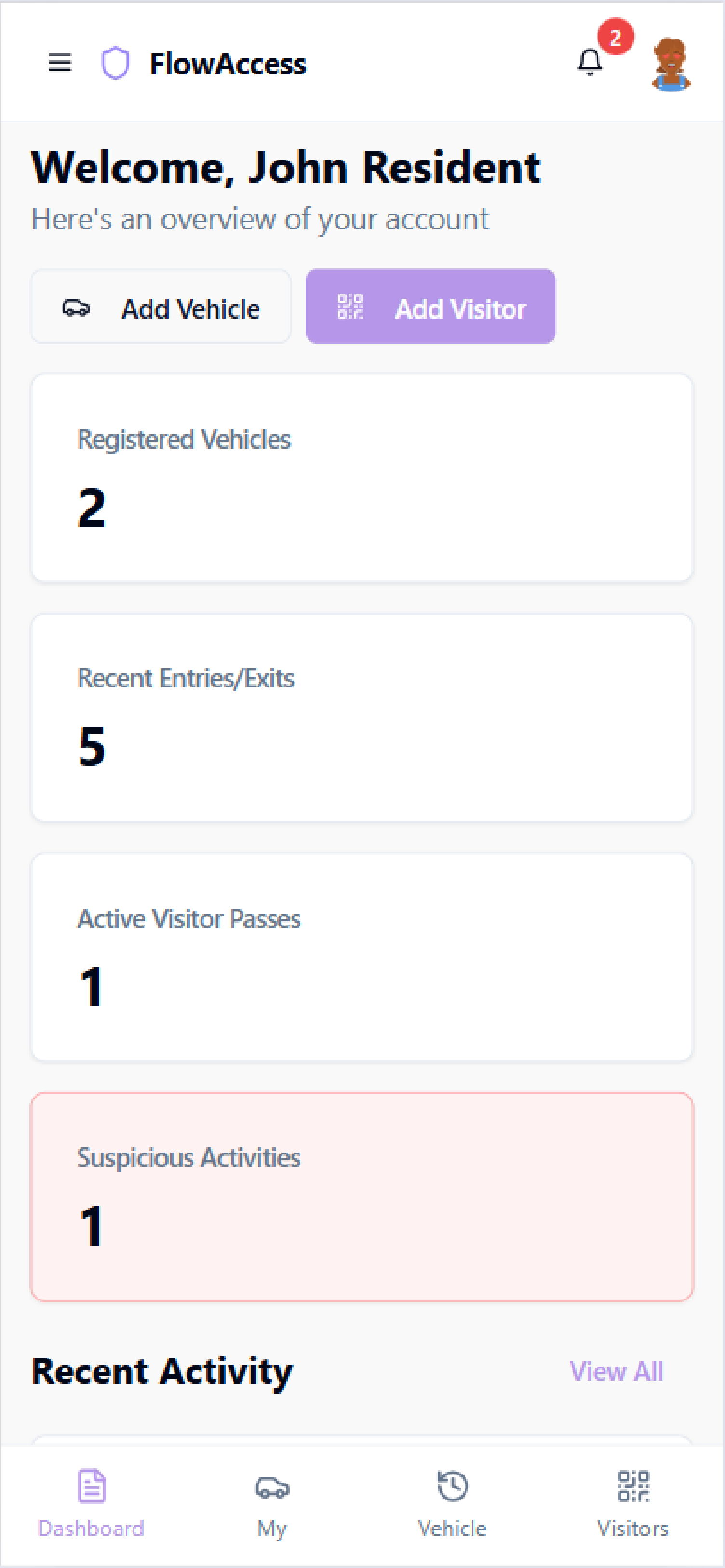


Figure 4.30: Resident’s Sidebar Page Mobile View



FlowAccess

2

Welcome, John Resident

Here's an overview of your account

Registered Vehicles

2

Recent Entries/Exits

5

Active Visitor Passes

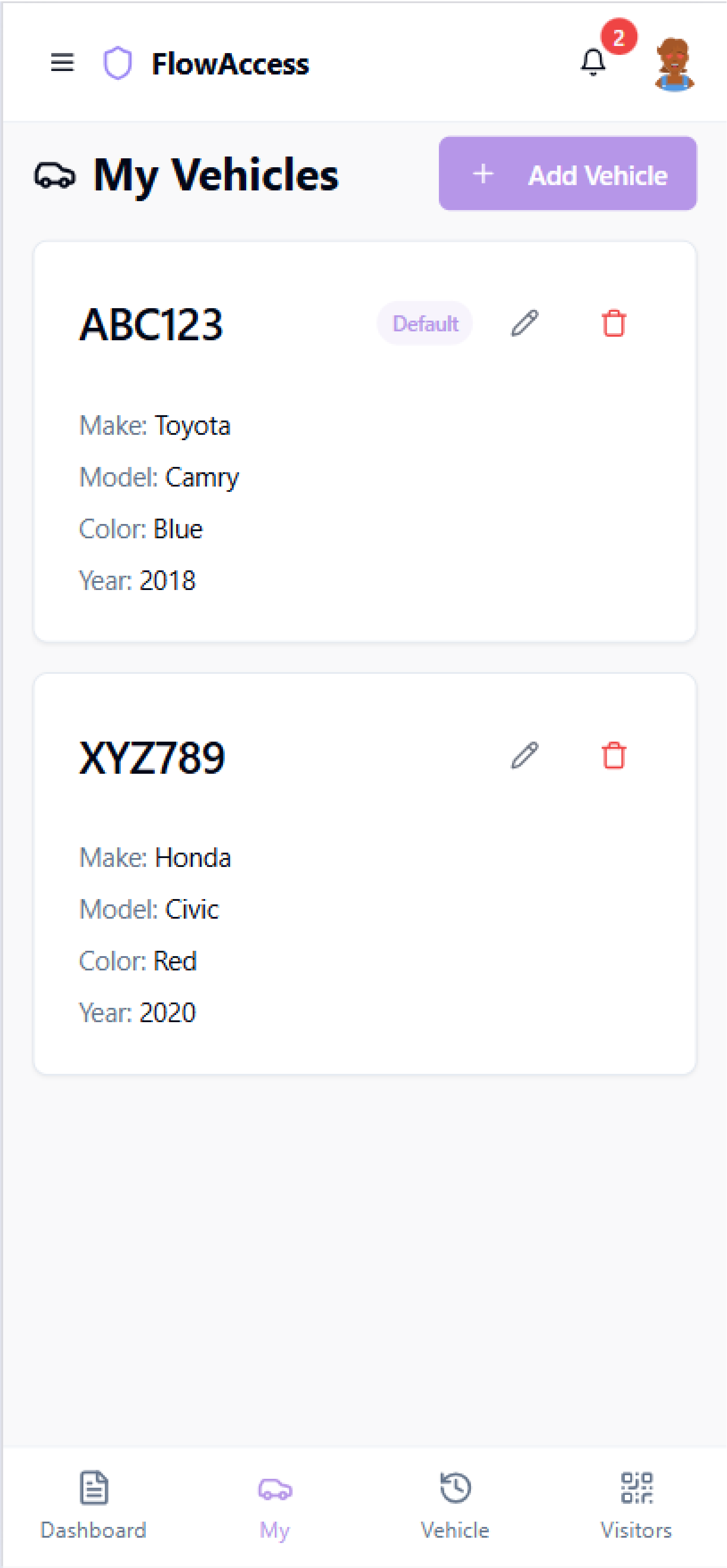
1

Suspicious Activities

1

Recent Activity

Figure 4.31: Resident's Dashboard Page Mobile View



My Vehicles

ABC123

Make: Toyota

Model: Camry

Color: Blue

Year: 2018

XYZ789

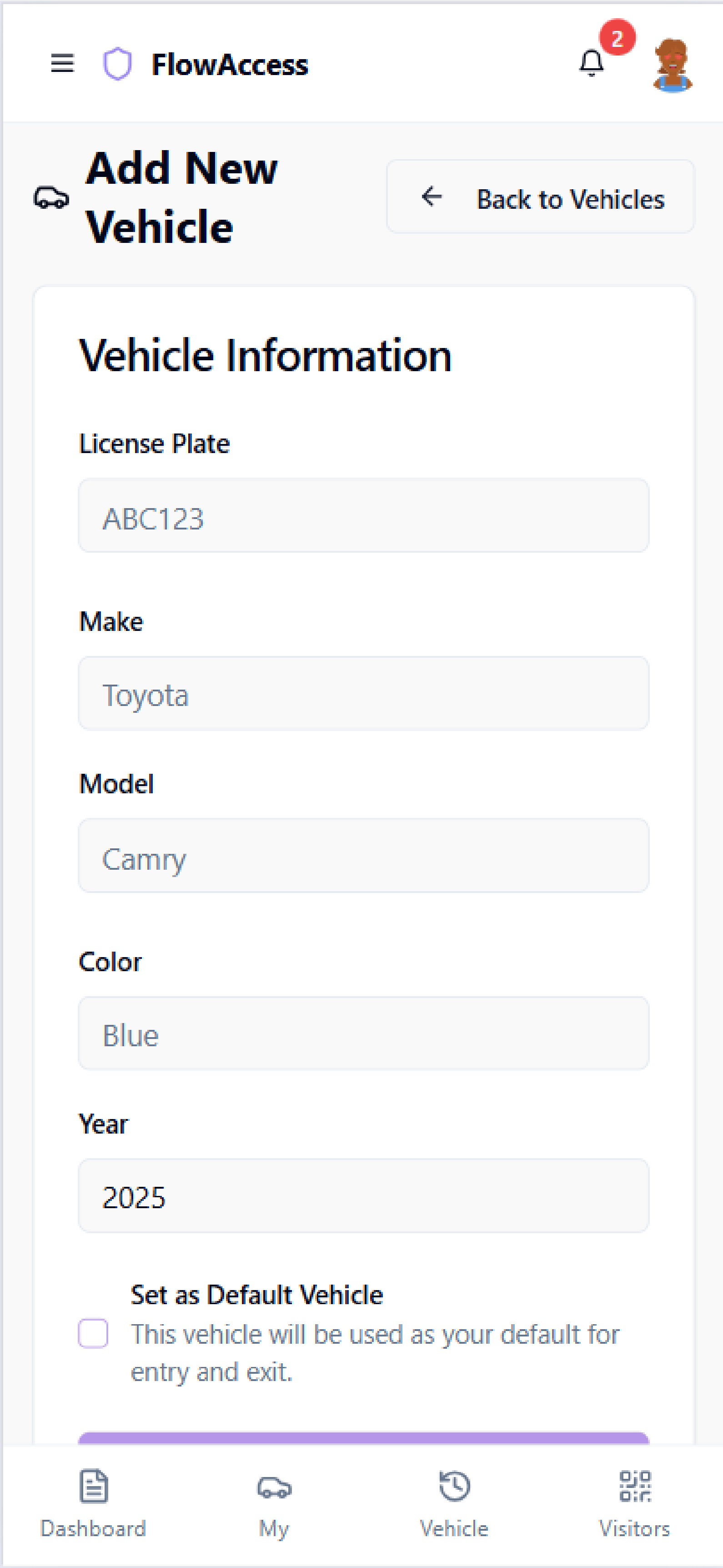
Make: Honda

Model: Civic

Color: Red

Year: 2020

Figure 4.32: Resident's Vehicle Page Mobile View



Vehicle Information

License Plate

ABC123

Make

Toyota

Model

Camry

Color

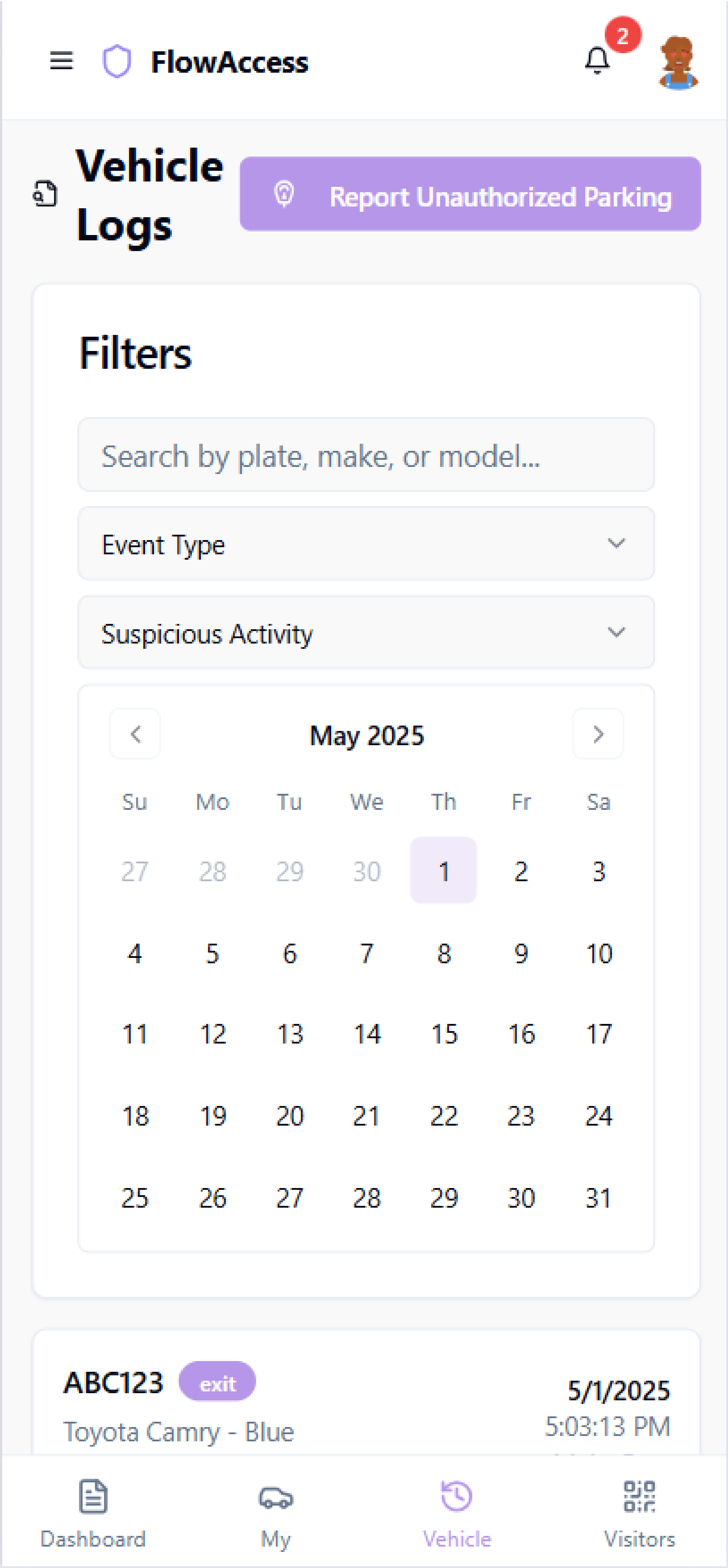
Blue

Year

2025

□ Set as Default Vehicle
This vehicle will be used as your default for entry and exit.

Figure 4.33: Resident's Add Vehicle Page Mobile View



FlowAccess

2

Vehicle Logs

Report Unauthorized Parking

Filters

Search by plate, make, or model...

Event Type

Suspicious Activity

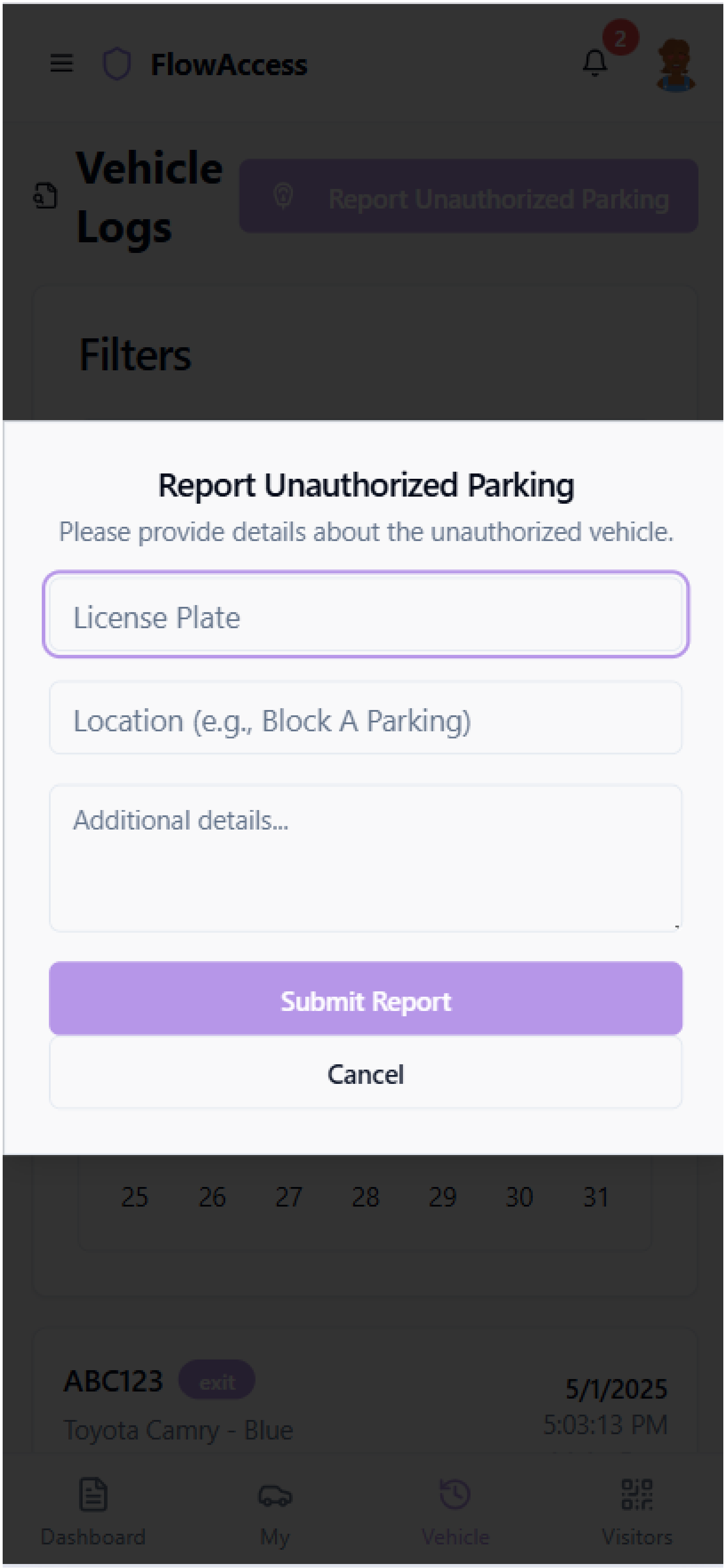
May 2025

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

ABC123 exit 5/1/2025

Toyota Camry - Blue 5:03:13 PM

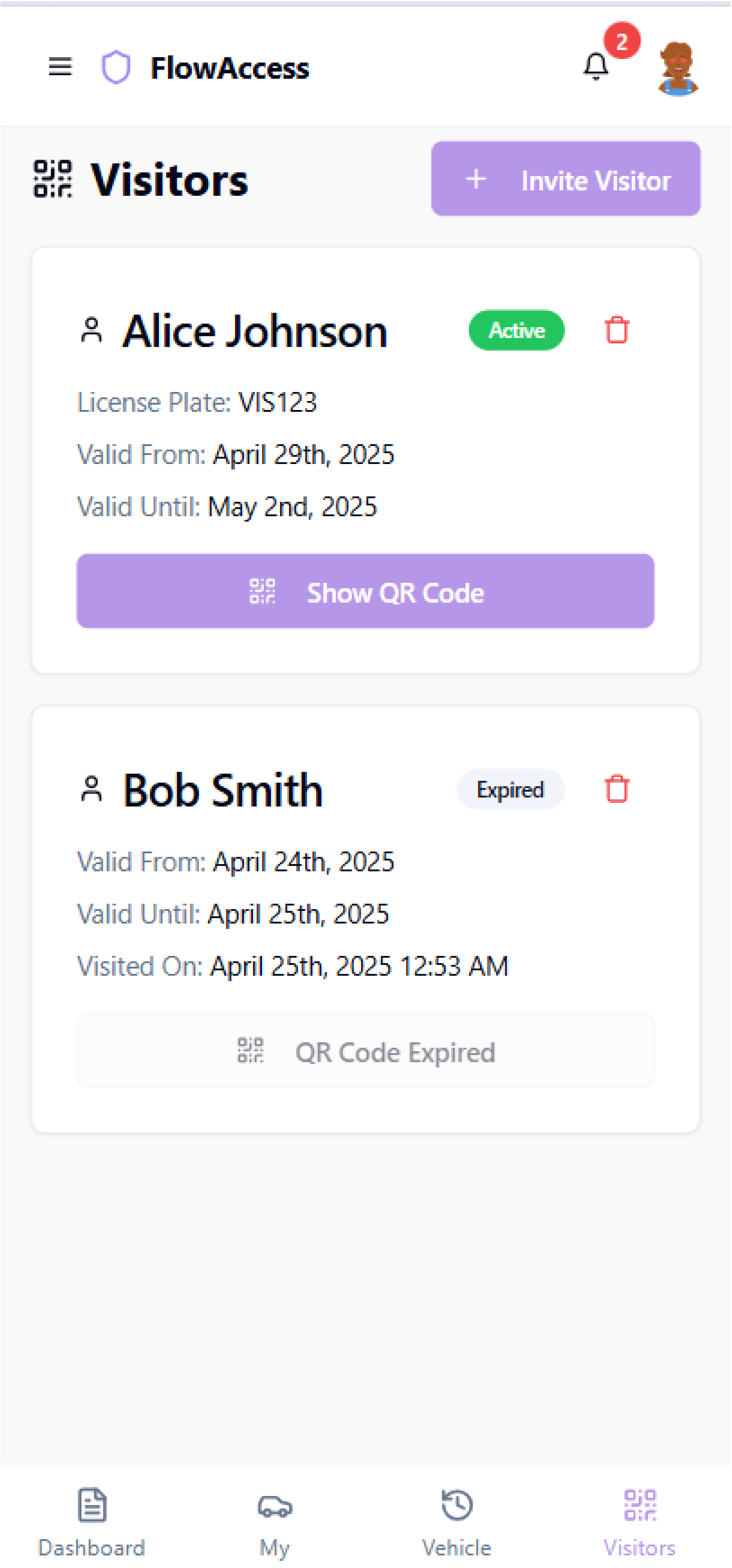
Figure 4.34: Resident's Vehicle Logs Page Mobile View



Report Unauthorized Parking

Please provide details about the unauthorized vehicle.

Figure 4.35: Resident's Report Unauthorized Parking Page Mobile View



FlowAccess

2

Visitors

+ Invite Visitor

Alice Johnson

Active

License Plate: VIS123

Valid From: April 29th, 2025

Valid Until: May 2nd, 2025

Show QR Code

Bob Smith

Expired

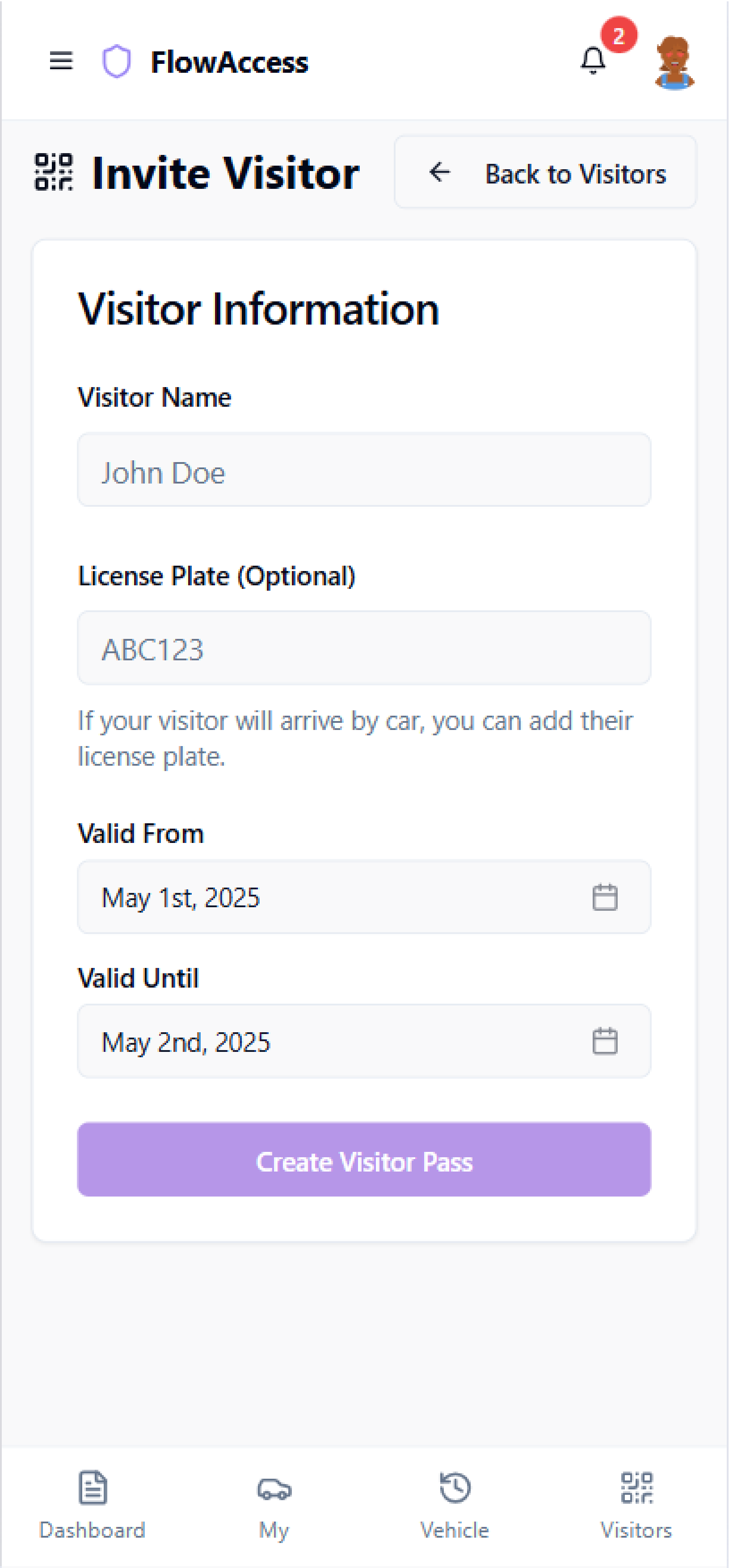
Valid From: April 24th, 2025

Valid Until: April 25th, 2025

Visited On: April 25th, 2025 12:53 AM

QR Code Expired

Figure 4.36: Resident's Visitor Page Mobile View



Invite Visitor

Visitor Information

Visitor Name

John Doe

License Plate (Optional)

ABC123

If your visitor will arrive by car, you can add their license plate.

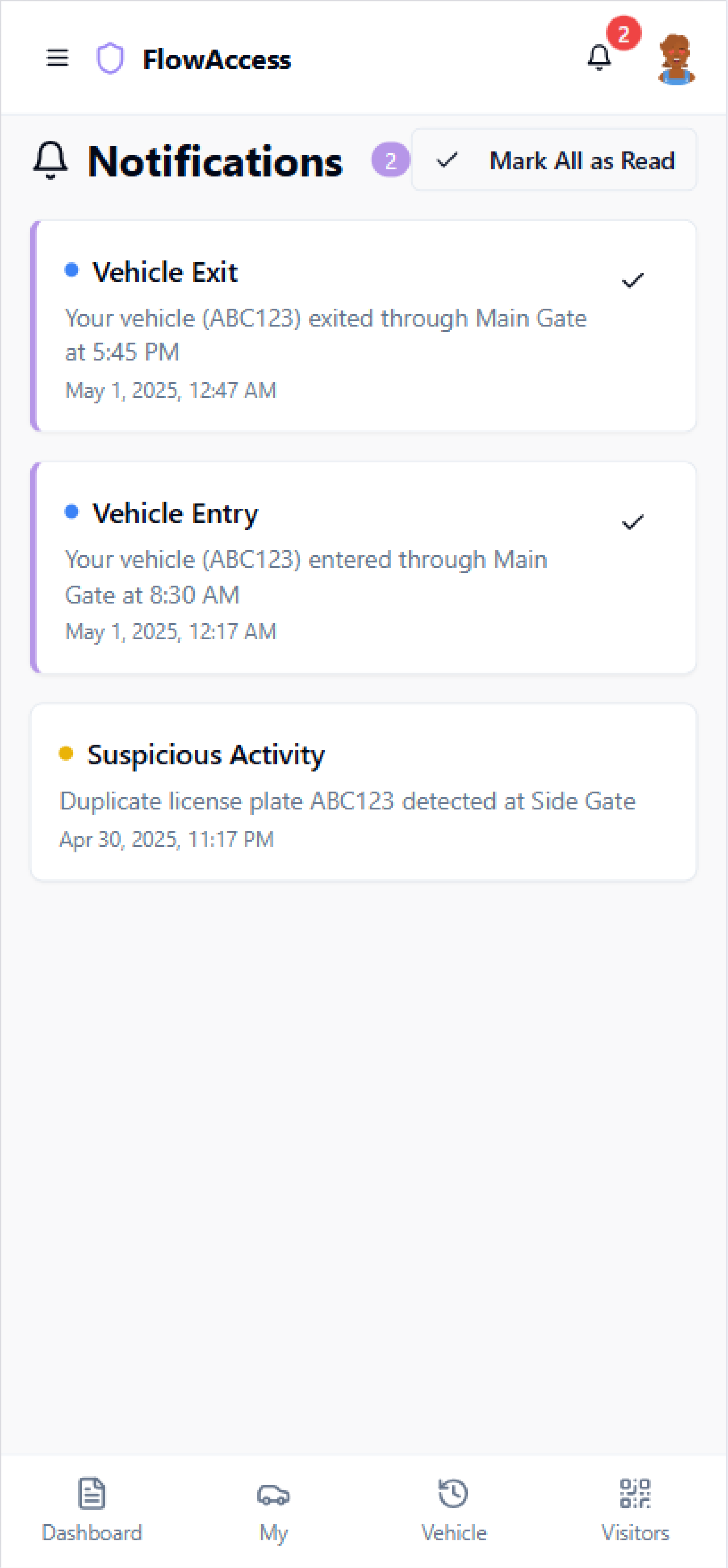
Valid From

May 1st, 2025

Valid Until

May 2nd, 2025

Figure 4.37: Resident's Invite Visitor Page Mobile View



● Vehicle Exit ✓

Your vehicle (ABC123) exited through Main Gate at 5:45 PM

May 1, 2025, 12:47 AM

● Vehicle Entry ✓

Your vehicle (ABC123) entered through Main Gate at 8:30 AM

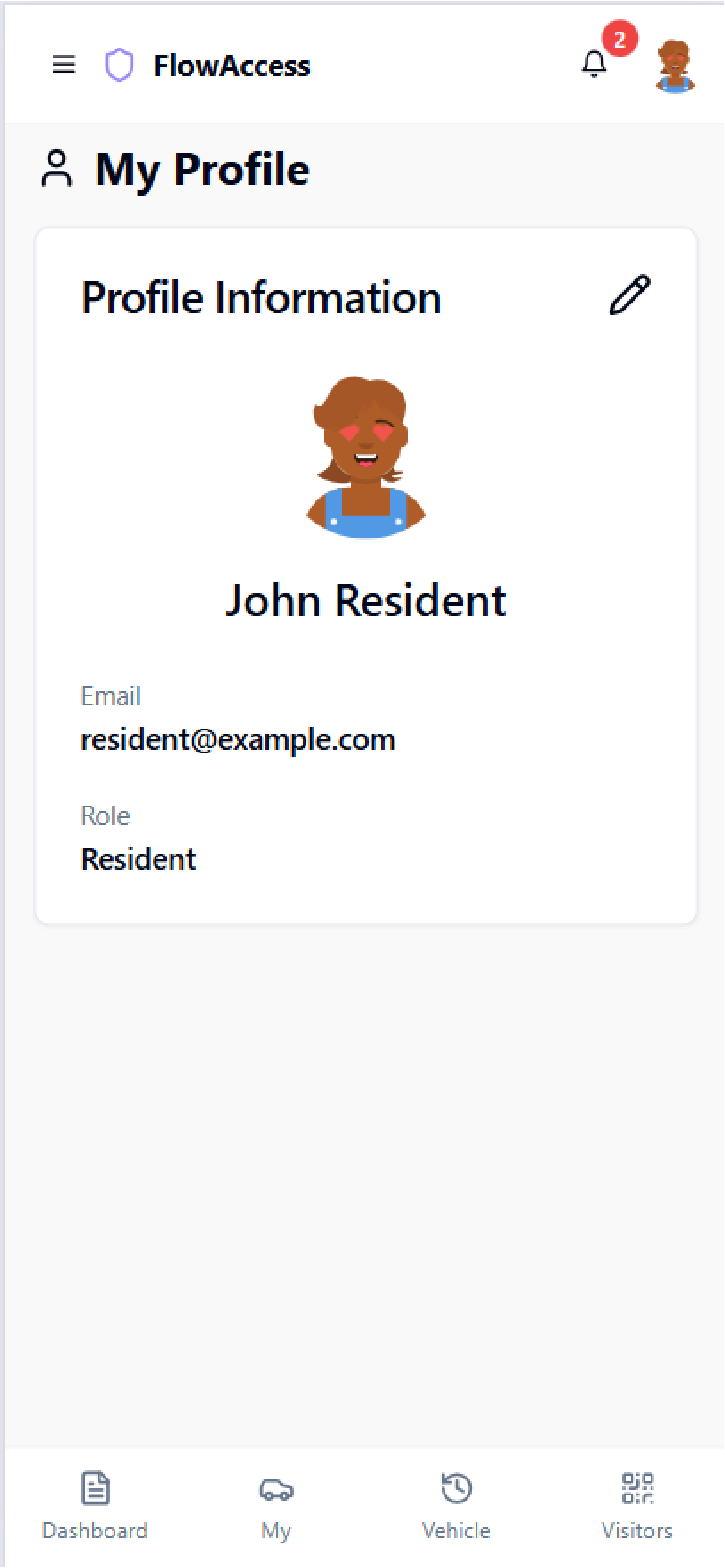
May 1, 2025, 12:17 AM

● Suspicious Activity

Duplicate license plate ABC123 detected at Side Gate

Apr 30, 2025, 11:17 PM

Figure 4.38: Resident’s Notification Page Mobile View



My Profile

Profile Information

John Resident

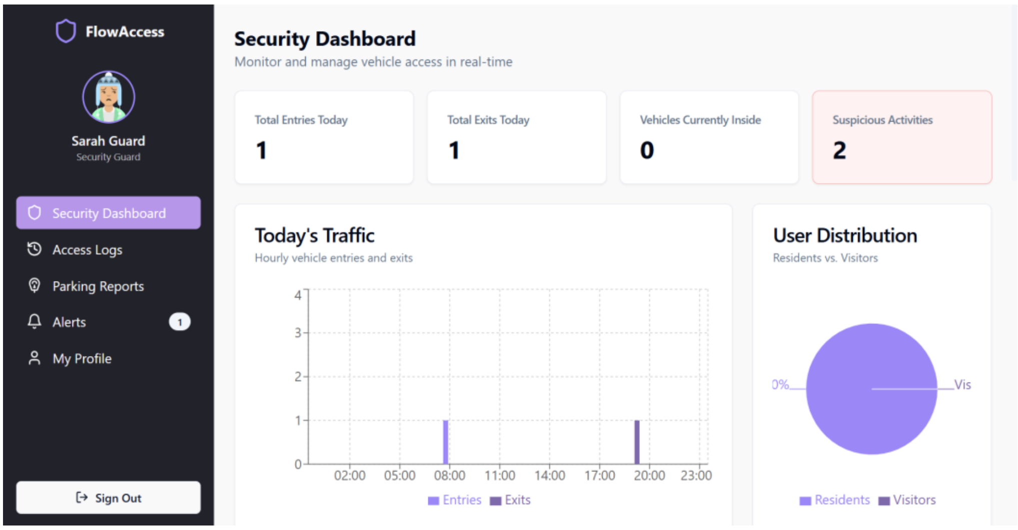
Email
resident@example.com

Role
Resident

Figure 4.39: Resident's Profile Page Mobile View

4.6.3 Security Guard’s Page

4.6.3.1 Web View



Security Dashboard

Monitor and manage vehicle access in real-time

Total Entries Today	Total Exits Today	Vehicles Currently Inside	Suspicious Activities
1	1	0	2

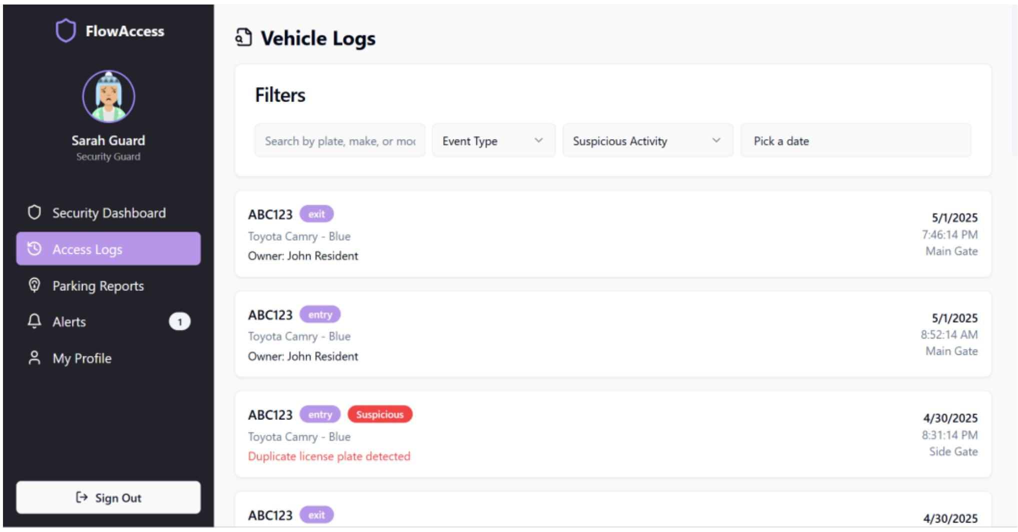
Today's Traffic

Hourly vehicle entries and exits

User Distribution

Residents vs. Visitors

Figure 4.40: Security Guard's Dashboard Page Web View



Vehicle Logs

Filters

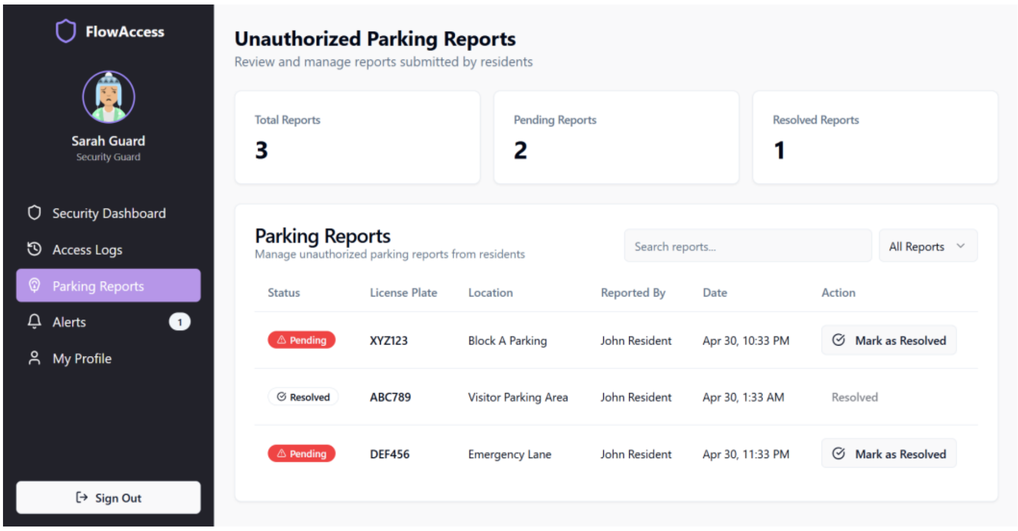
ABC123 exit
Toyota Camry - Blue
Owner: John Resident
5/1/2025
7:46:14 PM
Main Gate

ABC123 entry
Toyota Camry - Blue
Owner: John Resident
5/1/2025
8:52:14 AM
Main Gate

ABC123 entry Suspicious
Toyota Camry - Blue
Duplicate license plate detected
4/30/2025
8:31:14 PM
Side Gate

ABC123 exit
4/30/2025

Figure 4.41: Security Guard's Access Logs Page Web View



Unauthorized Parking Reports

Review and manage reports submitted by residents

Total Reports 3

Pending Reports 2

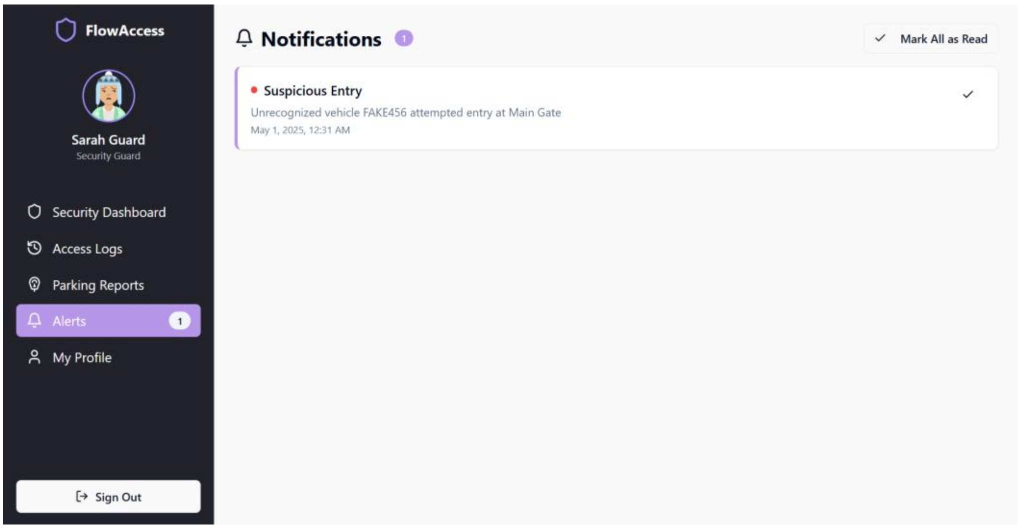
Resolved Reports 1

Parking Reports

Manage unauthorized parking reports from residents

Status	License Plate	Location	Reported By	Date	Action
Pending	XYZ123	Block A Parking	John Resident	Apr 30, 10:33 PM	Mark as Resolved
Resolved	ABC789	Visitor Parking Area	John Resident	Apr 30, 1:33 AM	Resolved
Pending	DEF456	Emergency Lane	John Resident	Apr 30, 11:33 PM	Mark as Resolved

Figure 4.42: Security Guard's Parking Reports Page Web View



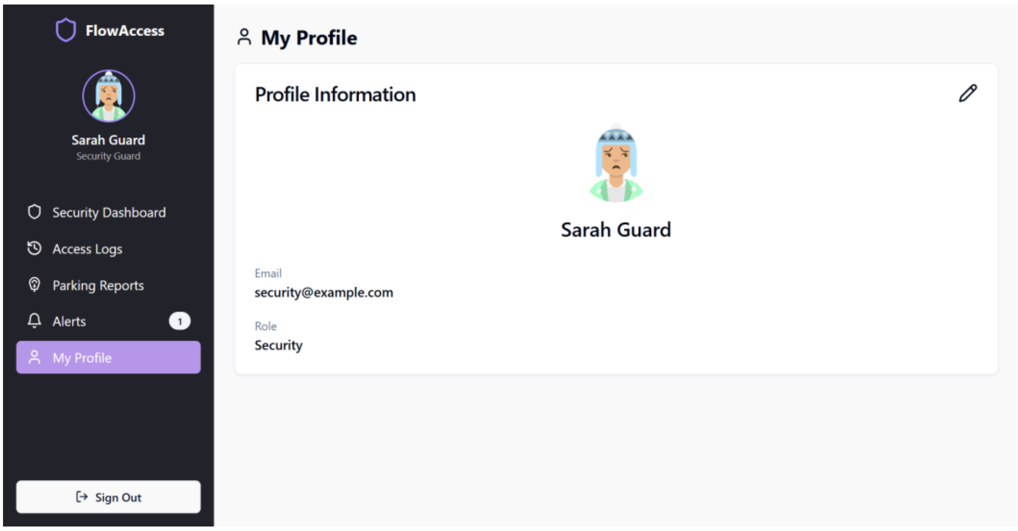
Notifications 1

Suspicious Entry

Unrecognized vehicle FAKE456 attempted entry at Main Gate

May 1, 2025, 12:31 AM

Figure 4.43: Security Guard's Alerts Page Web View



My Profile

Profile Information

Sarah Guard

Email

security@example.com

Role

Security

Figure 4.44: Security Guard's Profile Page Web View

4.6.3.2 Mobile View

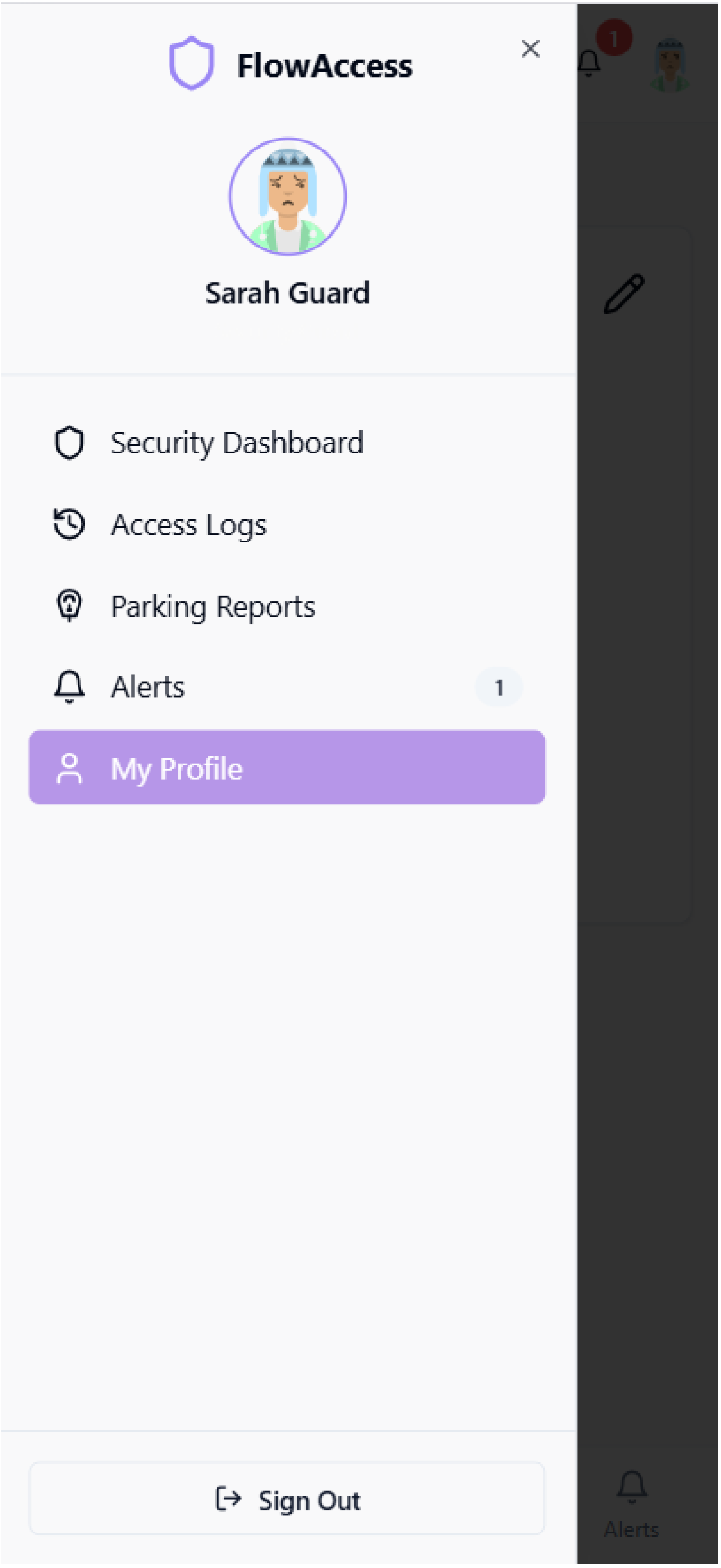
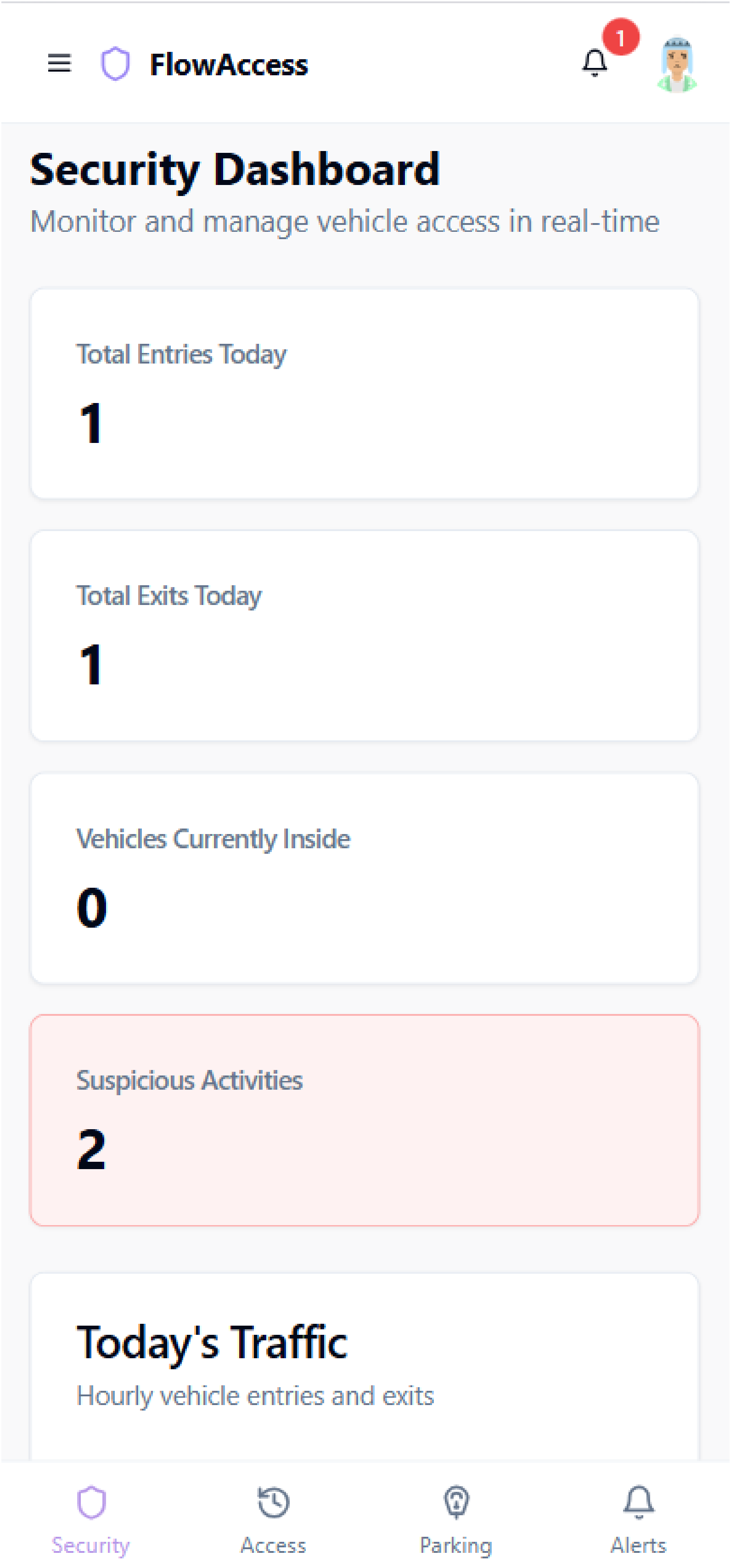


Figure 4.45: Security Guard's Sidebar Page Mobile View



Security Dashboard

Monitor and manage vehicle access in real-time

Total Entries Today

1

Total Exits Today

1

Vehicles Currently Inside

0

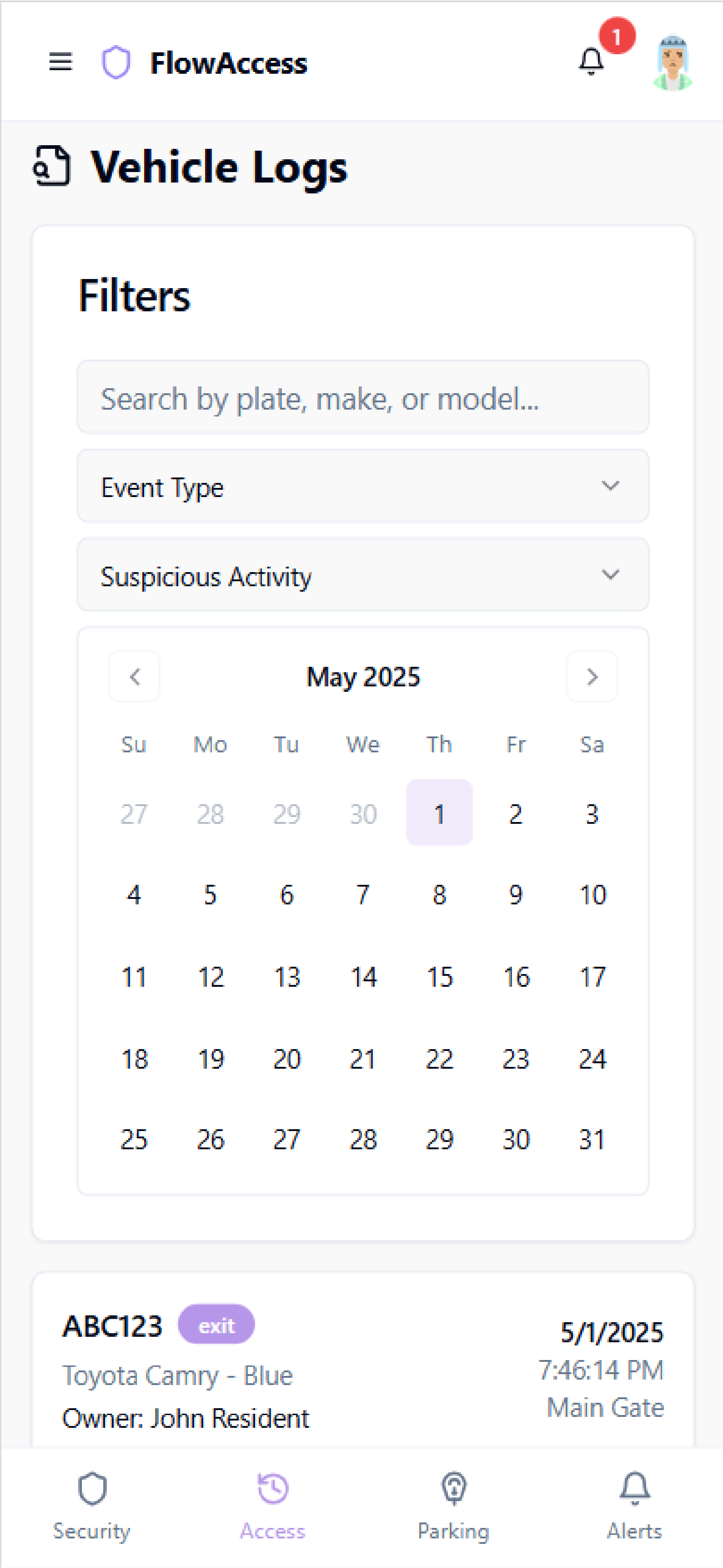
Suspicious Activities

2

Today's Traffic

Hourly vehicle entries and exits

Figure 4.46: Security Guard's Dashboard Page Mobile View



Vehicle Logs

Filters

Search by plate, make, or model...

Event Type

Suspicious Activity

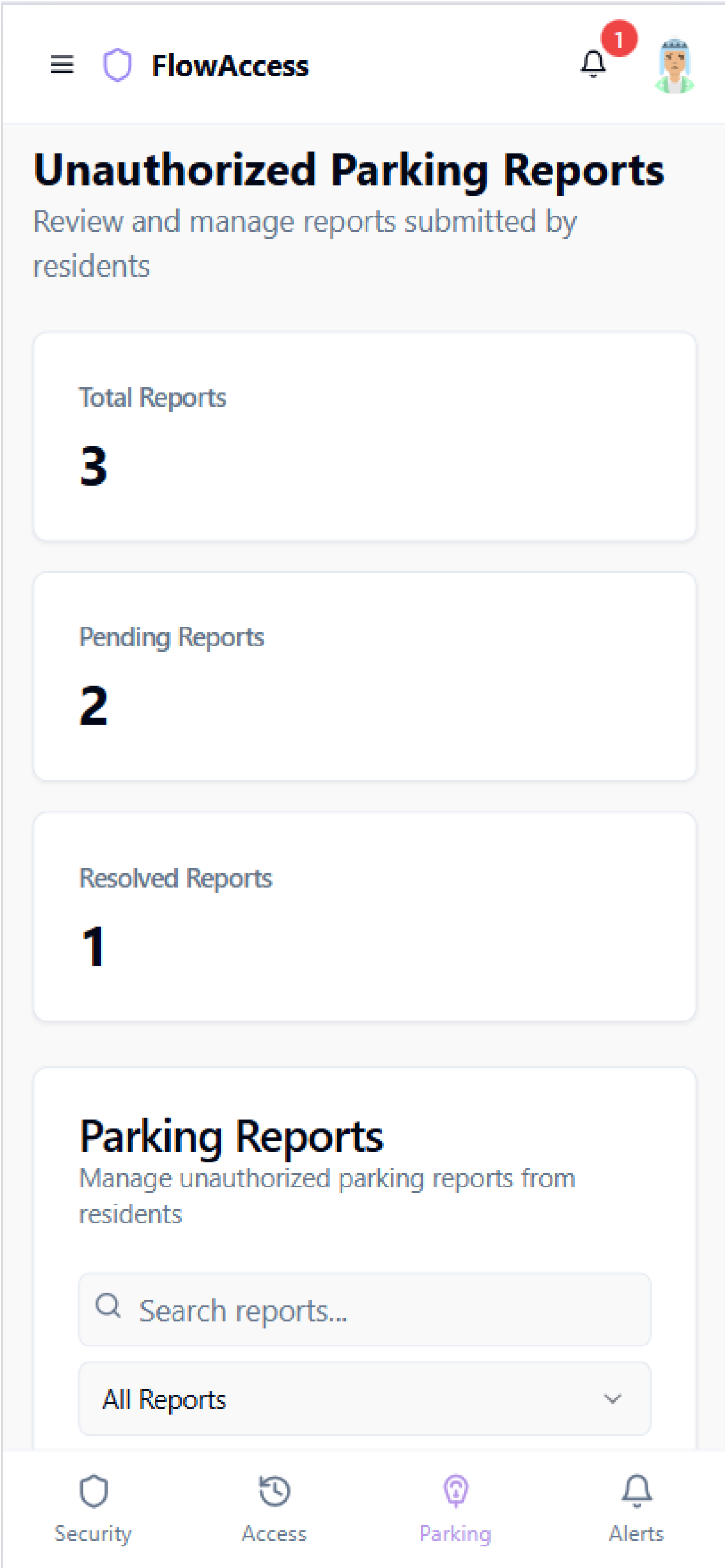
May 2025

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

ABC123 exit
Toyota Camry - Blue
Owner: John Resident

5/1/2025
7:46:14 PM
Main Gate

Figure 4.47: Security Guard's Access Logs Mobile View



Unauthorized Parking Reports

Review and manage reports submitted by residents

Total Reports

3

Pending Reports

2

Resolved Reports

1

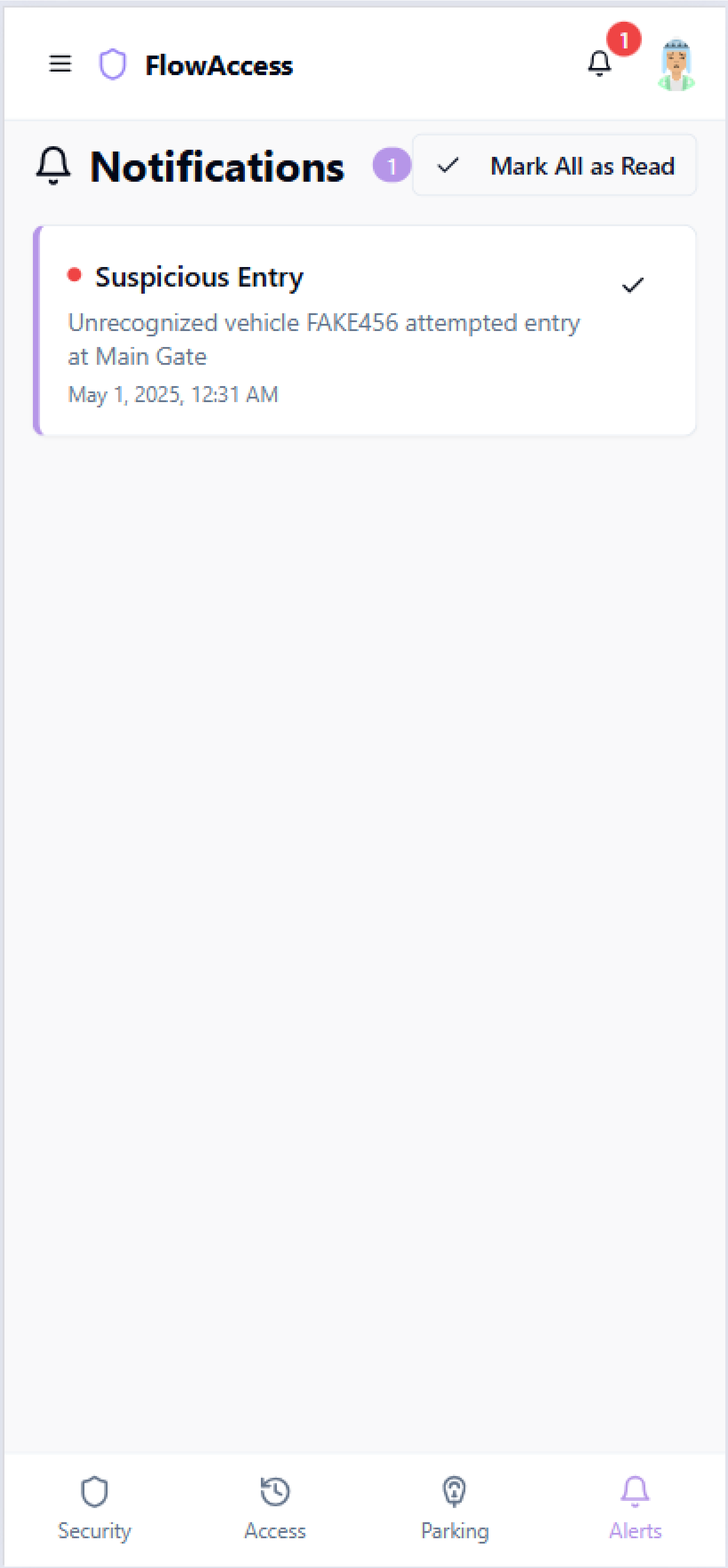
Parking Reports

Manage unauthorized parking reports from residents

Search reports...

All Reports

Figure 4.48: Security Guard's Unauthorized Parking Reports Page Mobile View

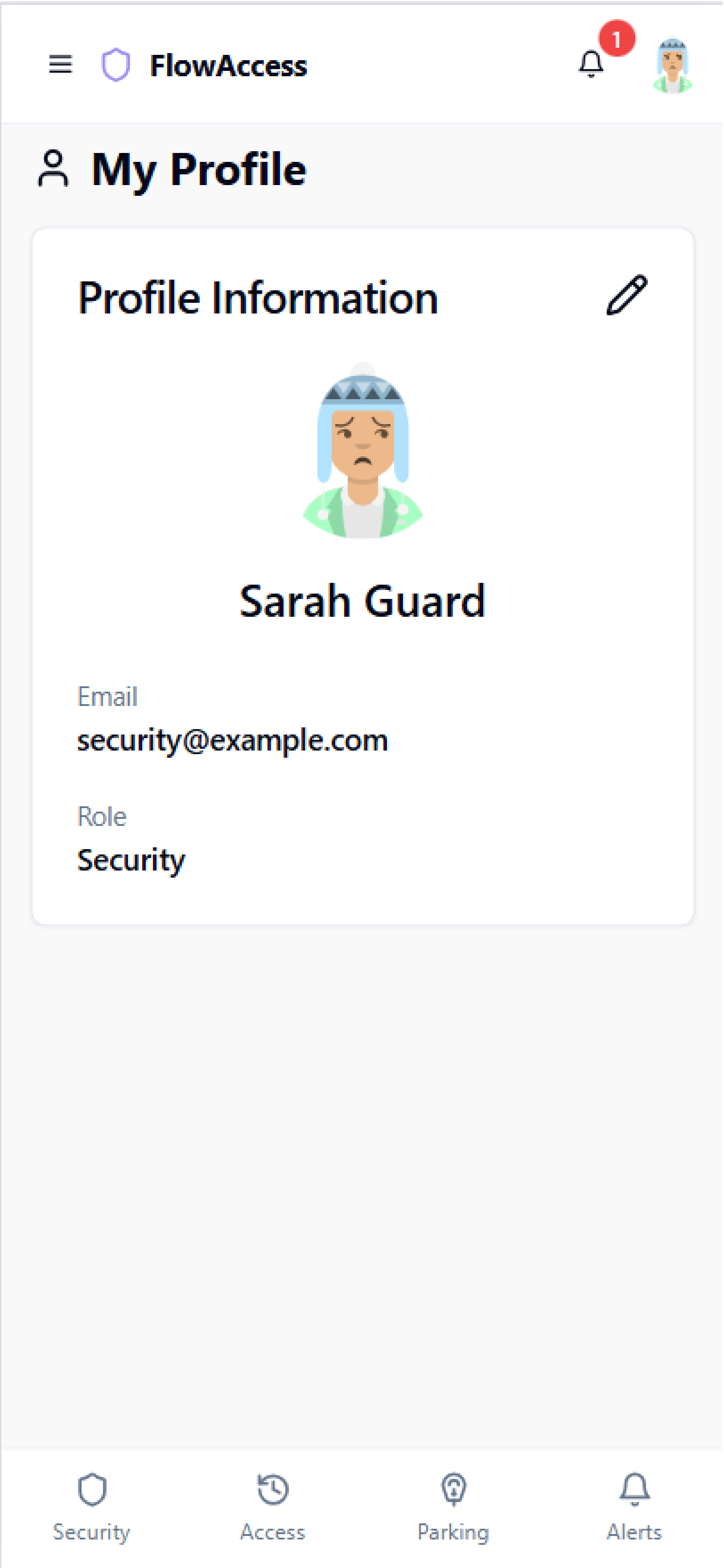


Suspicious Entry

Unrecognized vehicle FAKE456 attempted entry at Main Gate

May 1, 2025, 12:31 AM

Figure 4.49: Security Guard's Alerts Page Mobile View



My Profile

Profile Information

Sarah Guard

Email
security@example.com

Role
Security

Figure 4.50: Security Guard's Profile Page Mobile View

4.7 Preliminary Code run on SegmentAnything and Yolov8-seg

The preliminary code run on Segment Anything (SAM) and YOLOv8-Seg is vital for understanding how each segmentation model functions in the context of vehicle recognition especially when incorporated into a multimodal vehicle recognition system for vehicle access control. The accuracy, effectiveness and usability of several segmentation approaches in practical situations may be compared due to this early implementation. The preliminary study offers insights into the strengths and drawbacks of SAM and YOLOv8-Seg by testing them both on the same dataset. This serves as a basis for choosing the best model for the project. Additionally, it establishes the foundation for future system improvement and refinement, guaranteeing that the finished vehicle recognition framework can accurately and instantly handle a variety of scenarios.

4.7.1 Overview

This preliminary code compares two vehicle segmentation techniques, Segment Anything Model (SAM) and YOLOv8-Segmentation, by evaluating their performance on the same image. YOLOv8 detects vehicle bounding boxes, which are passed to SAM for segmentation, while YOLOv8-Segmentation directly performs end-to-end segmentation. The segmentation results from both models are compared using metrics like Intersection over Union (IoU), Dice Score, and Pixel Accuracy. The performance is also assessed in terms of processing time. Finally, the original image and the segmentation masks from both models are visualized side by side for qualitative comparison.

4.7.2 Detailed Steps

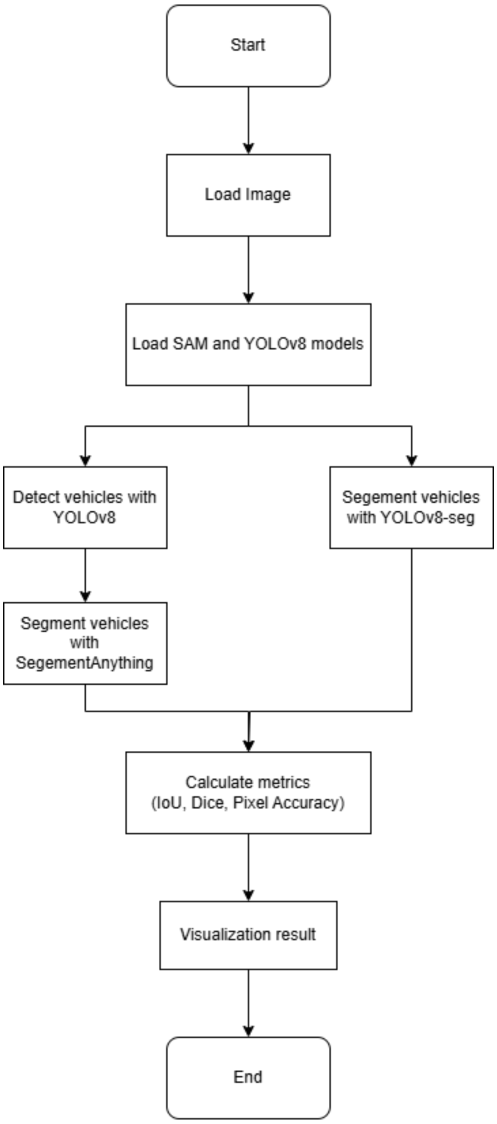

Start
Load Image
Load SAM and YOLOv8 models
Detect vehicles with YOLOv8
Segement vehicles with YOLOv8-seg
Segment vehicles with SegementAnything
Calculate metrics (IoU, Dice, Pixel Accuracy)
Visualization result
End

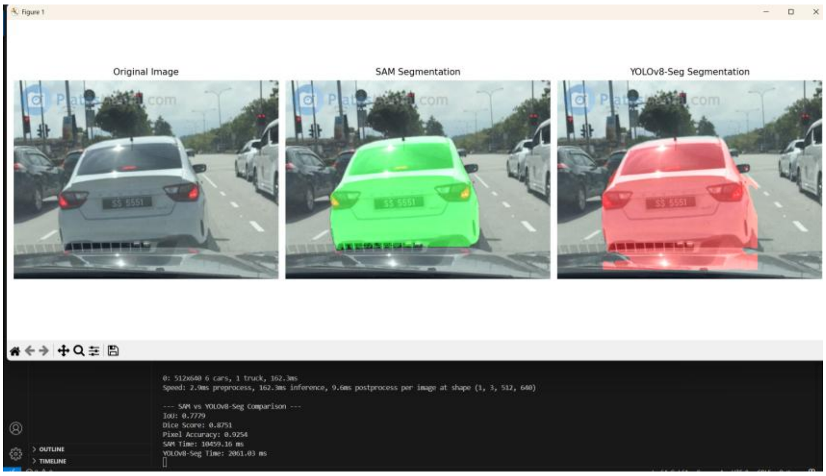
Figure 4.51: Flowchart of preliminary code

The vehicle segmentation pipeline shown in this flowchart combines Segment Anything Model (SAM) for accurate segmentation with YOLOv8 for object recognition. First, both models are initialized and an input image is loaded. In order to construct detailed pixel-level masks, SAM uses the bounding boxes that YOLOv8 creates after initially detecting cars. For comparison, the system also uses YOLOv8's built-in segmentation (YOLOv8-seg) to segment automobiles. The results are then visualized after metrics such as pixel accuracy, Dice coefficient, and intersection over union (IoU) are computed to assess the effectiveness of both segmentation techniques.

This workflow's goal is to ensure maximum accuracy and efficiency for vehicle segmentation jobs by comparing SAM to YOLOv8-seg. This pipeline can be used for this proposed vehicle access control system because the displayed results aid in validating the models' outputs.

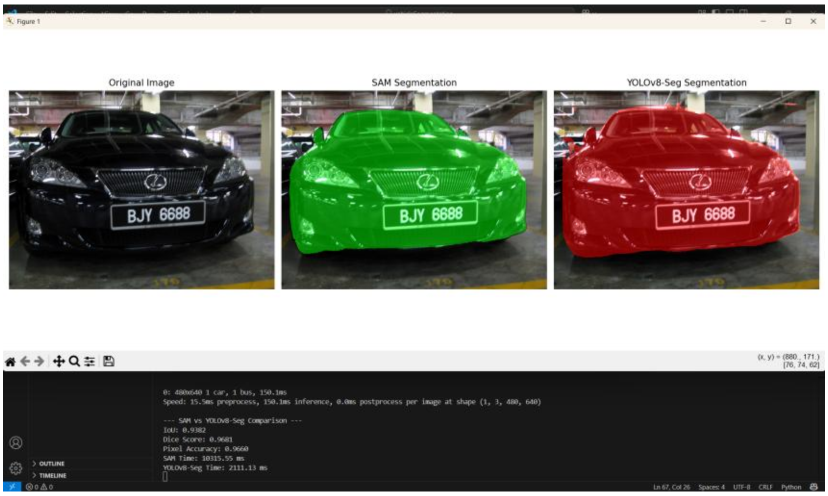
4.7.3 Experimental Setup and Result

The data collected is randomly selected from internet resources. For analyzing the results, the visualization results and the other metrics such as IoU and Dice are shown in the below figures and table:



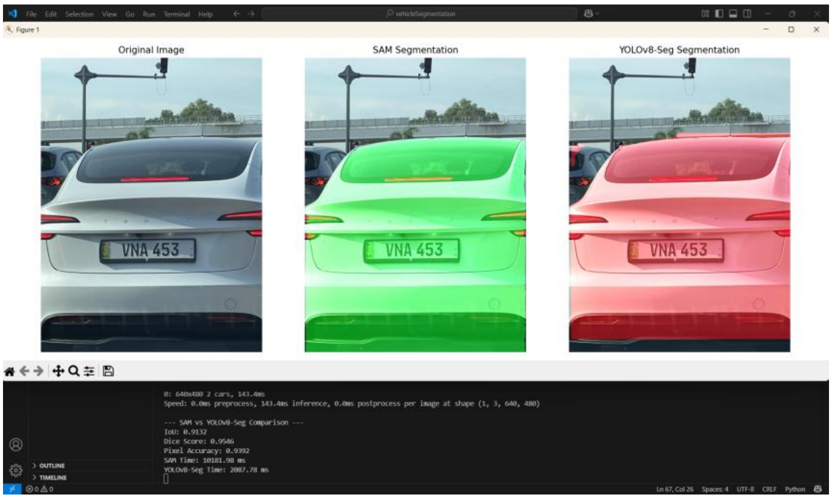
```
0: 512x640 6 cars, 1 truck, 162.3ms
Speed: 2.9ms preprocess, 162.3ms inference, 9.6ms postprocess per image at shape (1, 3, 512, 640)

--- SAM vs YOLOv8-Seg Comparison ---
IoU: 0.7779
Dice Score: 0.8751
Pixel Accuracy: 0.9254
SAM Time: 10459.16 ms
YOLOv8-Seg Time: 2061.03 ms
```

Figure 4.52: Visualization result of car 1



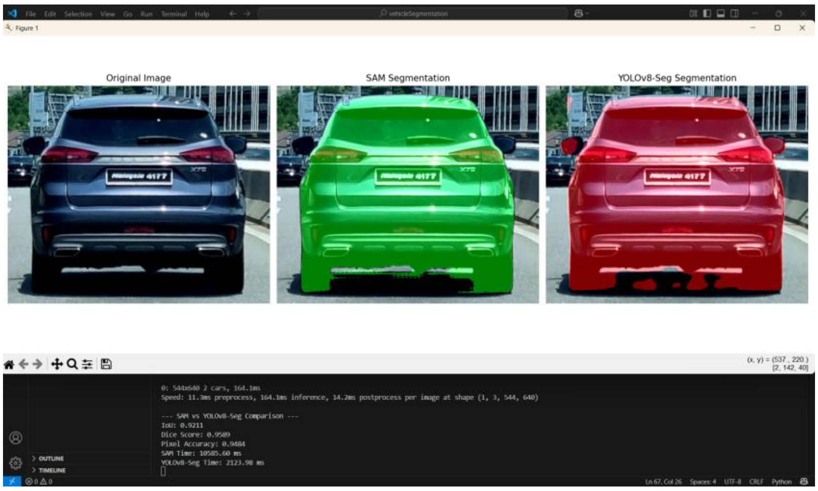
```
0: 480x640 1 car, 1 bus, 150.1ms
Speed: 15.5ms preprocess, 150.1ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

--- SAM vs YOLOv8-Seg Comparison ---
IoU: 0.9382
Dice Score: 0.9681
Pixel Accuracy: 0.9660
SAM Time: 10315.55 ms
YOLOv8-Seg Time: 2111.13 ms
```

Figure 4.53: Visualization result of car 2



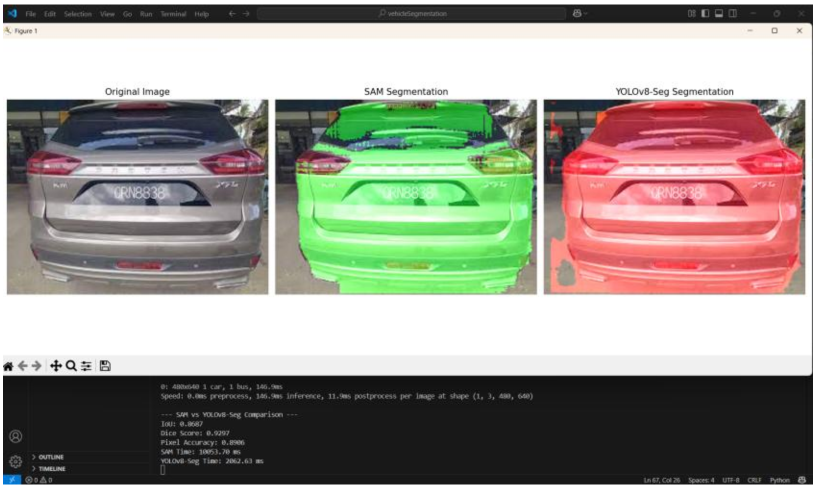
```
0: 640x480 2 cars, 143.4ms
Speed: 0.0ms preprocess, 143.4ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 480)

--- SAM vs YOLOv8-Seg Comparison ---
IoU: 0.9132
Dice Score: 0.9546
Pixel Accuracy: 0.9392
SAM Time: 10181.98 ms
YOLOv8-Seg Time: 2087.78 ms
```

Figure 4.54: Visualization result of car 3



```
0: 544x640 2 cars, 164.1ms
Speed: 11.3ms preprocess, 164.1ms inference, 14.2ms postprocess per image at shape (1, 3, 544, 640)

--- SAM vs YOLOv8-Seg Comparison ---
IoU: 0.9211
Dice Score: 0.9589
Pixel Accuracy: 0.9484
SAM Time: 10585.60 ms
YOLOv8-Seg Time: 2123.98 ms
```

Figure 4.55: Visualization result of car 4



```
0: 480x640 1 car, 1 bus, 146.9ms
Speed: 0.0ms preprocess, 146.9ms inference, 11.9ms postprocess per image at shape (1, 3, 480, 640)

--- SAM vs YOLOv8-Seg Comparison ---
IoU: 0.8687
Dice Score: 0.9297
Pixel Accuracy: 0.8906
SAM Time: 10053.70 ms
YOLOv8-Seg Time: 2062.63 ms
```

Figure 4.56: Visualization result of car 5

Table 4.１１: Experimental result of SAM and Yolov8-seg

Images	IoU	Dice	Pixel Accuracy	SAM Time (ms)	Yolov8-seg Time (ms)
1	0.7779	0.8751	0.9254	10459.16	2061.03
2	0.9382	0.9681	0.9660	10315.55	2111.13
3	0.9132	0.9546	0.9392	10181.98	2087.78
4	0.9211	0.9589	0.9484	10585.60	2123.98
5	0.8687	0.9297	0.8906	10053.70	2062.63

According to table 4.13, the experimental results comparing Segment Anything (SAM) and YOLOv8-Seg are shown in Table 4.13, with a focus on the following important performance metrics including processing times, intersection over union (IoU), dice coefficient and pixel accuracy. High accuracy was shown by both models as SAM obtained IoU values between 0.7779 and 0.9382 while YOLOv8-Seg yielded comparable outcomes. This implies that both models can produce segmentations that are extremely accurate. The Dice coefficient which measures the overlap between the ground truth and predicted masks was also high for both models with YOLOv8-Seg ranging from 0.8906 to 0.9660 and SAM scoring between 0.8751 and 0.9681.

Both SAM and YOLOv8-Seg performed well while assessing Pixel Accuracy as SAM received scores ranging from 0.8906 to 0.9254 while YOLOv8-Seg received ratings between 0.9660 to 0.8906. This shows that both models successfully and accurately label individual pixels in their segmentation tasks. However, the processing time of the two models is a crucial distinction between them. Compared to YOLOv8-Seg, which processed photos between 2061.03 ms and 2123.98 ms, SAM's processing time ranged roughly from 10053.7 ms to 10585.6 ms.

The findings point to a trade-off. SAM is better suited for applications where accuracy is more important than speed because it consumes more computing time even though it produces higher segmentation quality. However, YOLOv8-Seg has a little lower segmentation accuracy but delivers faster speed which making it perfect for real-time applications. Despite its higher processing time, SAM is the recommended option due of its superior segmentation accuracy which is crucial for vehicle identification systems.

(Note: The segmentation technique was changed to YOLO in later chapter 6.2.3)

4.8 Chapter Summary

The project's initial findings are presented in this chapter which provides information on the system's development process and early results. A summary of the work completed so far opens the chapter which is followed by a fact-finding phase that examined important data relevant to the project. Understanding the goals and expectations of the system requires knowledge of both functional and non-functional needs which are described in the User needs Specifications (URS) section.

System use cases are also included in this chapter, along with a use case diagram and descriptions that provide readers a clear idea of how the system will work in practical situations. Additionally, the prototype and suggested system flow are presented, showcasing the expected user interface flow and preliminary design components.

The initial code runs of the SegmentAnything and YOLOv8-seg models which are essential to the vehicle detection system under development are specifically covered in Section 4.6. This part contains an examination of the experimental setup and outcomes, a thorough description of the procedures followed and a summary of the models' performance. Insights on the advantages and disadvantages of the selected segmentation models are provided by this early testing, which is essential for improving the system's performance in later phases of development.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter gave a summary of the system's design and included a number of diagrams and the system architecture that showed the system's structure. The flow of data or information through the system was represented in a data flow diagram (DFD). The use cases covered in earlier chapters served as the foundation for the processes in the data flow diagram. Level-0 DFD and a context diagram were also included in this chapter. Lastly, the screenshots of the developed system's user interface flow design were displayed. For easier reading and understanding, all the screenshots were arranged according to the system modules.

5.2 System Architecture Design

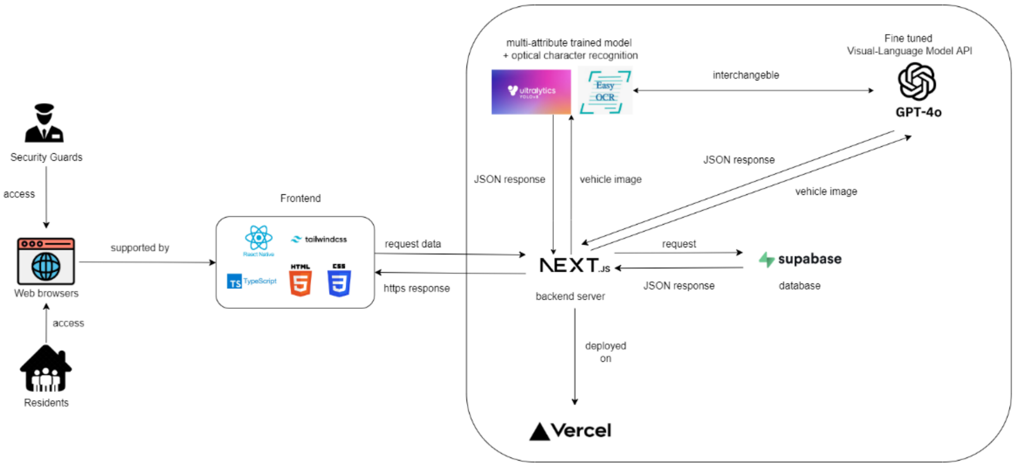


Figure 5.1: System Architecture Diagram

The figure above showed the system architecture of the proposed vehicle recognition and monitoring system which combined the web-based and artificial intelligence components. Residents and security personnel were the primary users of the architecture which interacting with the system via the web browsers. Modern web programming languages including React Native, TypeScript, HTML, CSS, and TailwindCSS were used in the development of the application's frontend as they offered a responsive and user-friendly interface for managing the residents, visitors, vehicles and notifications. To ensure the safe data flow throughout the system, HTTPS requests and replies were used to establish communication between the frontend and the backend.

The Next.js-powered backend server which served as the primary processing unit for managing the user requests was at the heart of the architecture. The database layer and the AI recognition services were the two primary components that the backend communicated with. Supabase managed all the structured data, including user profiles, residents, visitors, vehicles, records and notifications. It also integrated with the auth.users to offer authentication and authorization. In the meantime, the backend sent vehicle images to the AI modules for tasks such as recognition.

Following the processing, the Next.js backend received the JSON replies from the Supabase and the AI modules. It then compiled and formatted the data before returning it to the frontend. This made it possible for the users especially the security personnel to effectively monitor the access in real time, identify questionable records and confirm the identities of vehicles. Vercel was used to deploy the full backend system and this had guaranted the scalability, dependability, and cloud-based accessibility. In conclusion, this architecture supported the intelligent monitoring and effective communication within the system by facilitating a smooth integration between the frontend interfaces, backend logic, safe data storage and AI-powered recognition.

5.3 System Design Models

This section presented a entity relationship illustration that described the structured view of the database concepts and their relationships.

5.3.1 Entity Relationship Diagram (ERD)

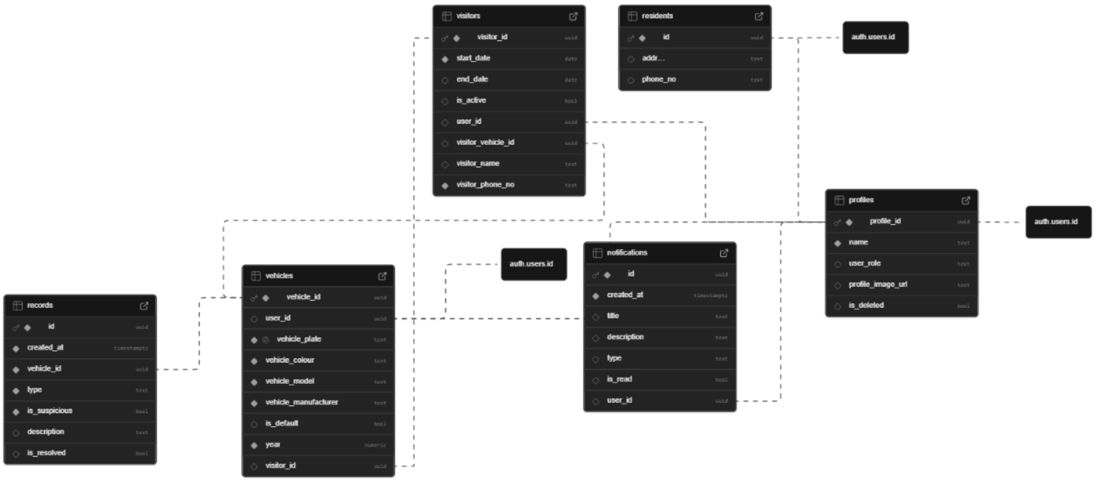


Figure 5.2: Entity Relationship Diagram (ERD) of system

Figure 5.2 above illustrated the Entity Relationship Diagram (ERD) of the system that includes seven primary tables which are auth.users, profiles, residents, visitors, cars, records, and notifications. The auth.users table is the authentication layer provided by Supabase. The function of the auth users table is to store each system user's core login credentials like email and password. It is expanded by the profiles table, which serves as the main link between other modules and stores extra user information like name, role, and profile picture url. By collecting resident-specific data like address and phone number, the residents table extends profiles even further. Meanwhile, the visitors table keeps track of temporary users' contact information, their duration of visit, and optional vehicle linkages. Vehicle-related data, including license plate number, model, manufacturer, color, and year, are stored in the vehicle's table and are associated with either a resident through profiles table or a visitor. Vehicle-related occurrences and events are recorded in the records table along with the type, description, and status of the case which is suspicious or resolved. Last but not least, the notifications table provides users with system-generated messages and alerts based on their events or behaviors. Secure authentication through auth.users, smooth resident and visitor linkage through profiles, and effective vehicle, incident, and system-wide communication monitoring are all guaranteed by this schema design.

5.3.2 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) was used to represent the flow of information within the system. It illustrated how the data entered the system, the processes that transformed the data and also the outputs generated. By providing a clear and structured view of the interactions between users, processes, and data stores, the DFD helped in understanding the overall functionality of the system at different levels of the abstraction.

5.3.2.1 Context Diagram

The context diagram provided in this section illustrated the system at the highest level of the abstraction. The diagram outlined the system's boundaries and demonstrated the flow of the information between the users and the system without detailing the internal processes.

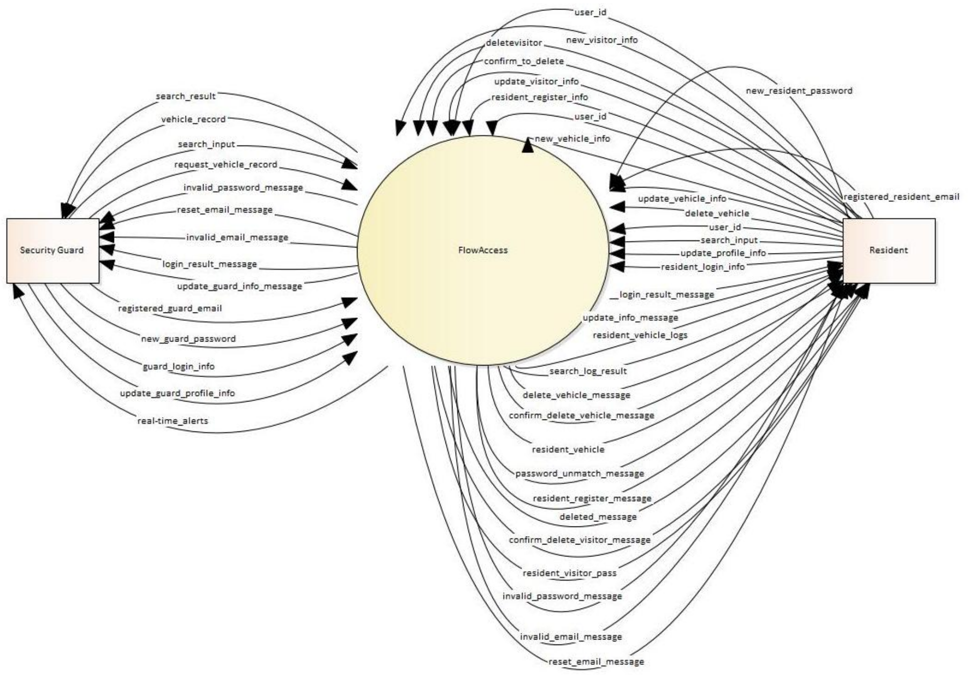


Figure 5.3: Context Diagram

5.3.2.2 DFD Level – 0

The Level 0 DFD provides a high-level overview of the system. It showed the main processes, external entities and also the data flows. It defines the system boundaries and illustrates how the data moves between the users, the system, and data stores. In other words, it highlighted the core functions such as vehicle registration, recognition, and data storage in this project.

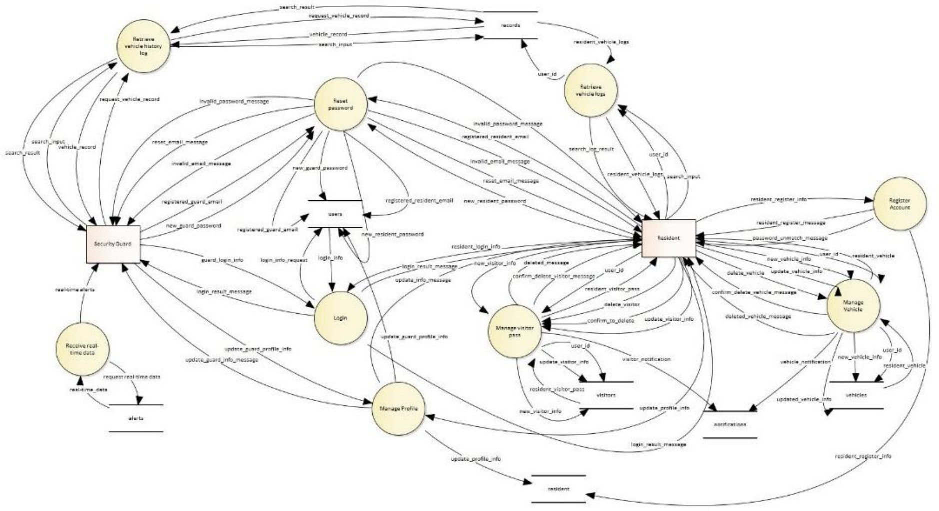


Figure 5.4: Data Flow Diagram Level - 0

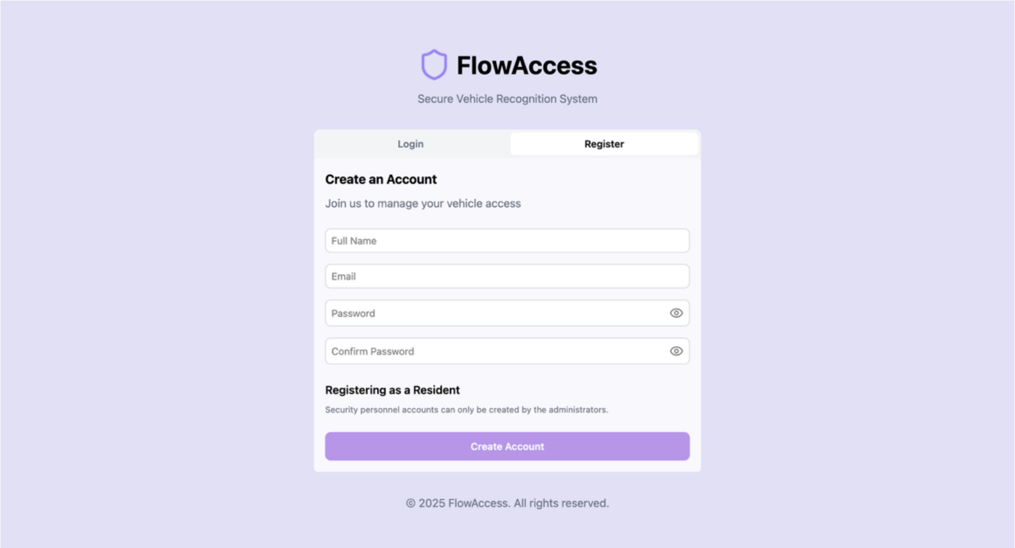
5.4 User Interface Design

User Interface (UI) Design defines how the users interact with the vehicle recognition system. It focuses on creating the intuitive, user-friendly screens for both residents and security guards and also ensuring that all system functions such as the vehicle registration, visitor pass generation, recognition results and access management are easily accessible and efficiently navigable to the user. Good UI design improves the user experience, reduces errors and supports the smooth operation of the system on the web platform.

5.4.1 Resident Side

Resident Side UI Design focuses on creating an intuitive and user-friendly interface for the residents to interact with the system.

5.4.1.1 Registration



FlowAccess

Secure Vehicle Recognition System

Login | Register

Create an Account

Join us to manage your vehicle access

Full Name

Email

Password

Confirm Password

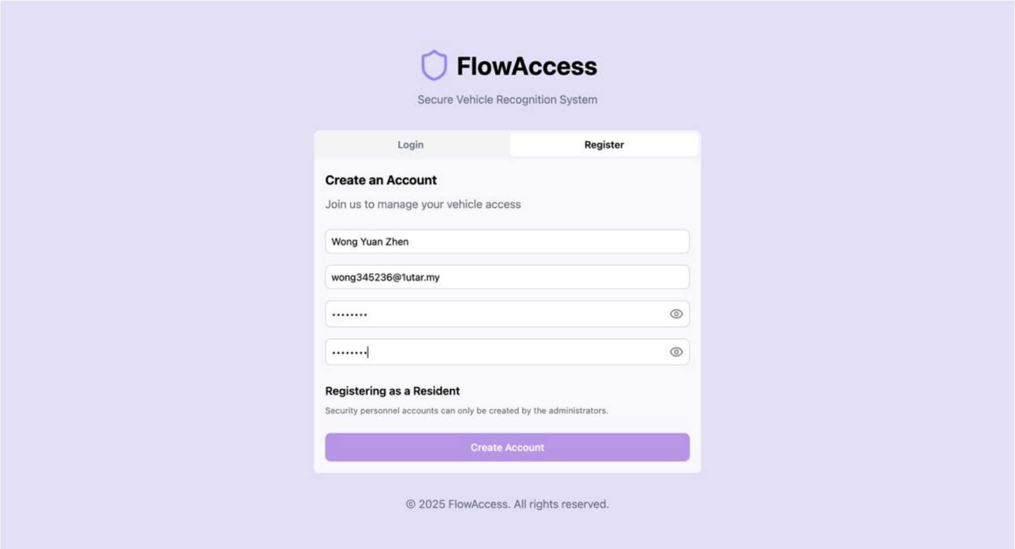
Registering as a Resident

Security personnel accounts can only be created by the administrators.

Create Account

© 2025 FlowAccess. All rights reserved.

Figure 5.5: Resident Register UI



FlowAccess

Secure Vehicle Recognition System

Login | Register

Create an Account

Join us to manage your vehicle access

Wong Yuan Zhen

wong345236@1utar.my

••••••••

••••••••

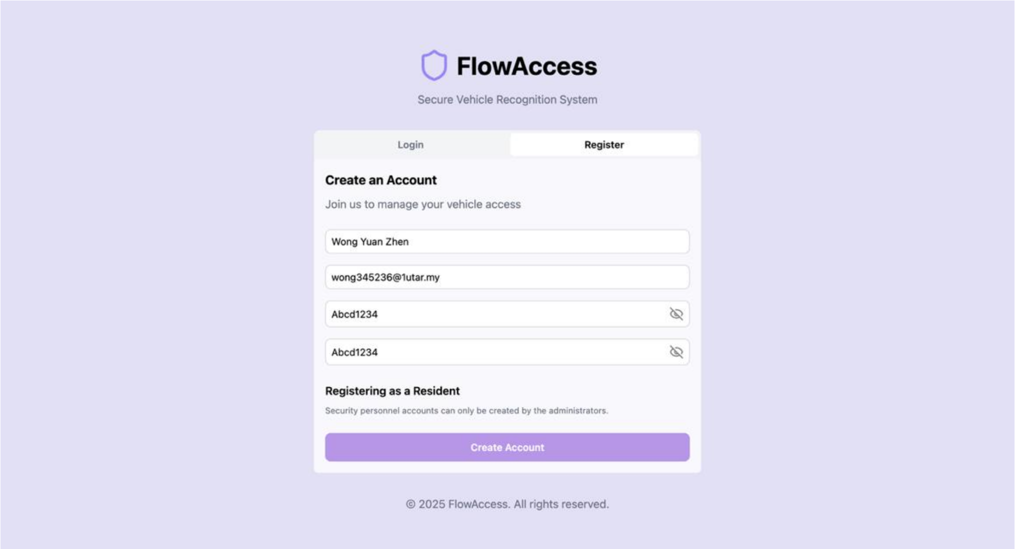
Registering as a Resident

Security personnel accounts can only be created by the administrators.

Create Account

© 2025 FlowAccess. All rights reserved.

Figure 5.6: Resident fill in Registration UI without readability on password



FlowAccess

Secure Vehicle Recognition System

Create an Account

Join us to manage your vehicle access

Wong Yuan Zhen

wong345236@1utar.my

Abcd1234

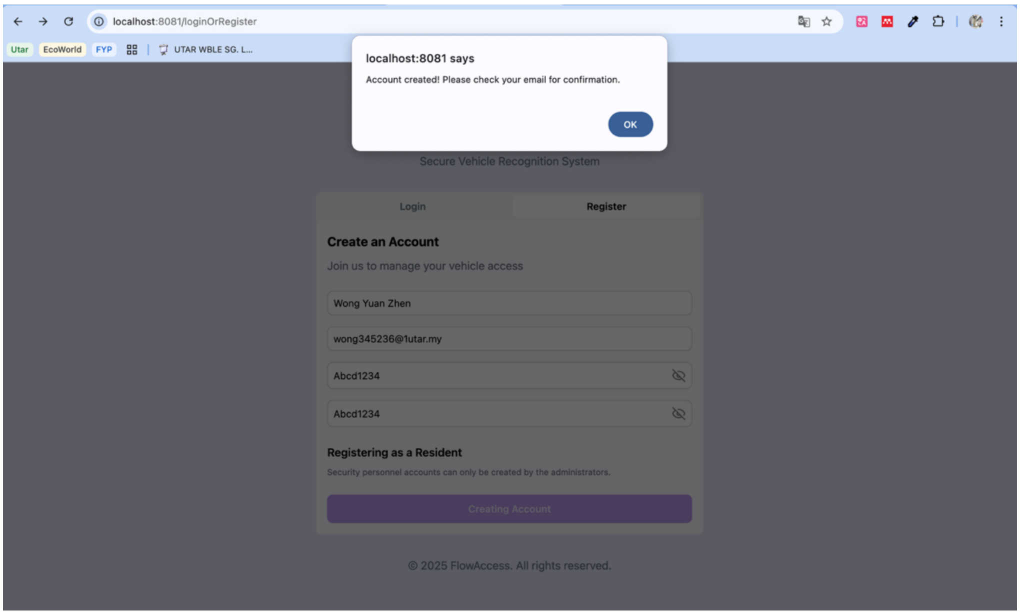
Abcd1234

Registering as a Resident

Security personnel accounts can only be created by the administrators.

© 2025 FlowAccess. All rights reserved.

Figure 5.7: Resident fill in Registration UI with readability on password



localhost:8081 says

Account created! Please check your email for confirmation.

Secure Vehicle Recognition System

Create an Account

Join us to manage your vehicle access

Wong Yuan Zhen

wong345236@1utar.my

Abcd1234

Abcd1234

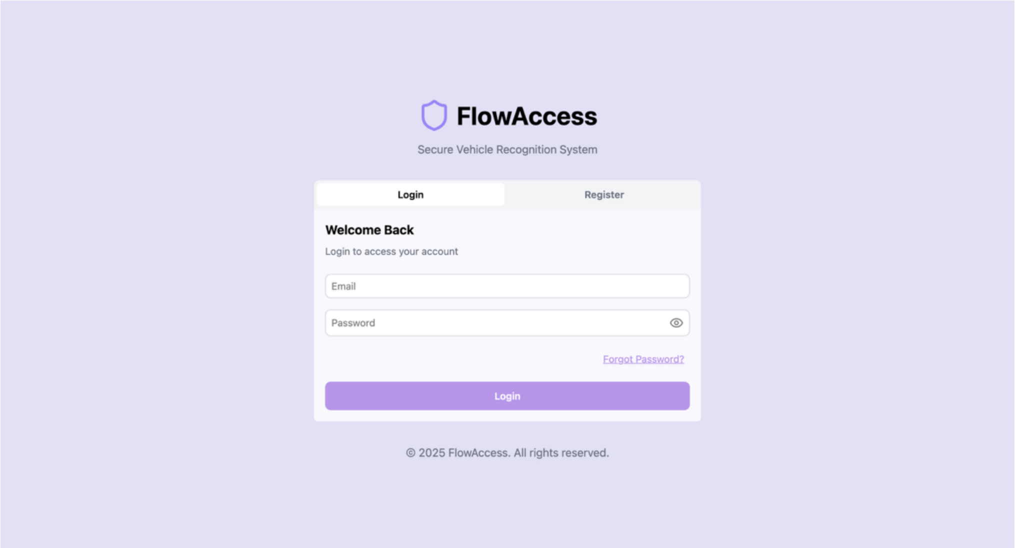
Registering as a Resident

Security personnel accounts can only be created by the administrators.

© 2025 FlowAccess. All rights reserved.

Figure 5.8: Resident Registration Success UI

5.4.1.2 Login

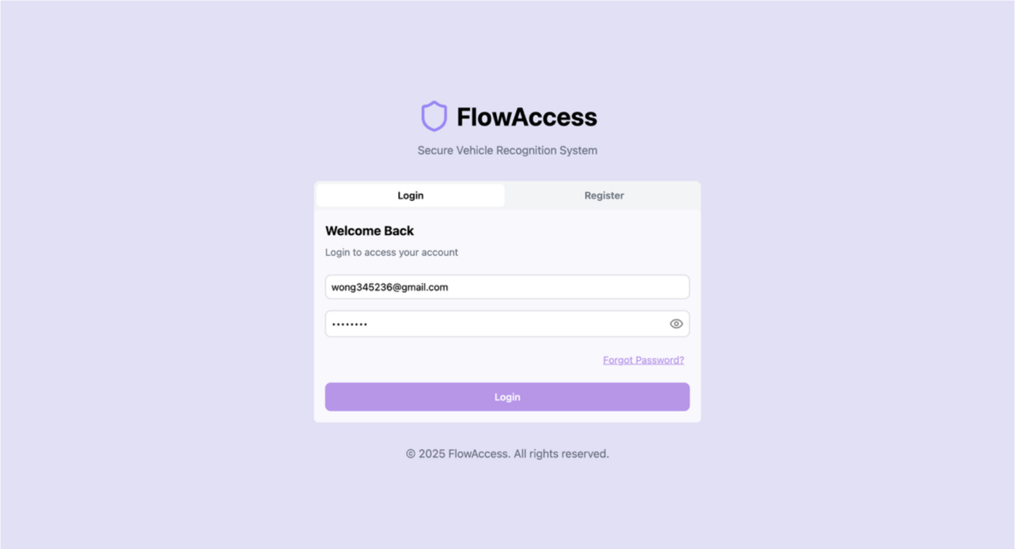


Welcome Back

Login to access your account

© 2025 FlowAccess. All rights reserved.

Figure 5.9: Resident Login UI



Welcome Back

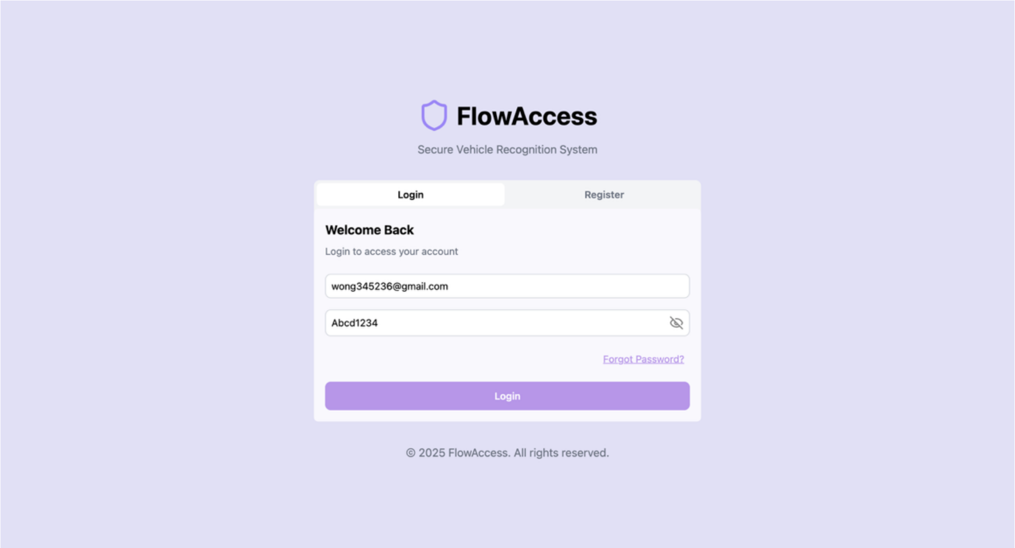
Login to access your account

wong345236@gmail.com

••••••••

© 2025 FlowAccess. All rights reserved.

Figure 5.10: Resident Login UI without readability on password



FlowAccess

Secure Vehicle Recognition System

Welcome Back

Login to access your account

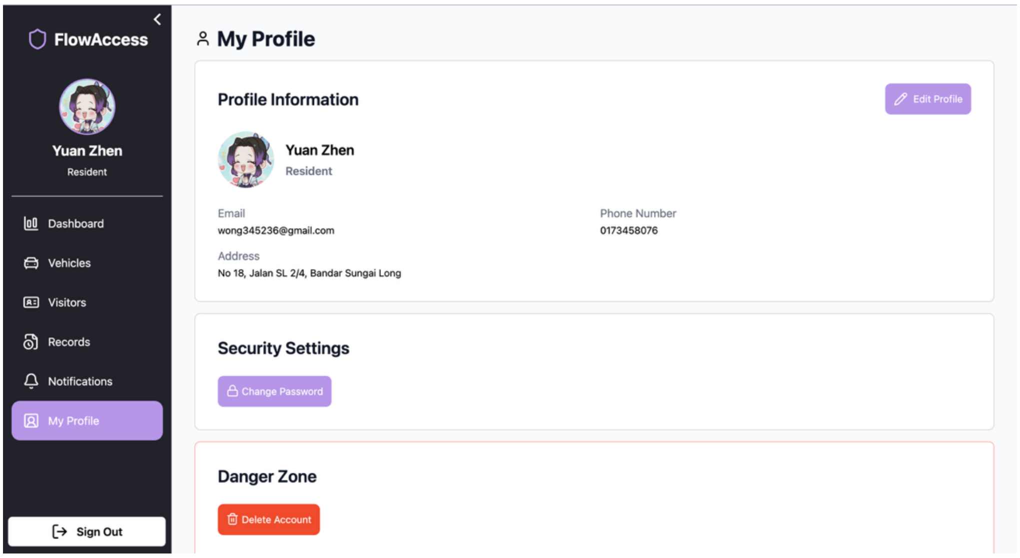
wong345236@gmail.com

Abcd1234

© 2025 FlowAccess. All rights reserved.

Figure 5.11: Resident Login UI with readability on password

5.4.1.3 Manage Profile



My Profile

Profile Information

Yuan Zhen
Resident

Email
wong345236@gmail.com

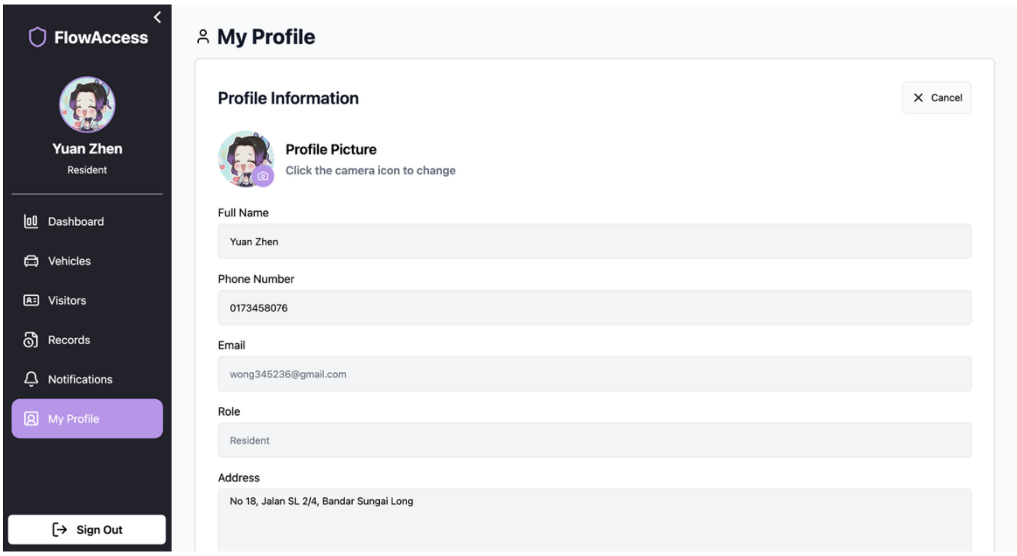
Phone Number
0173458076

Address
No 18, Jalan SL 2/4, Bandar Sungai Long

Security Settings

Danger Zone

Figure 5.12: Resident Profile Page UI



My Profile

Profile Information

Profile Picture
Click the camera icon to change

Full Name
Yuan Zhen

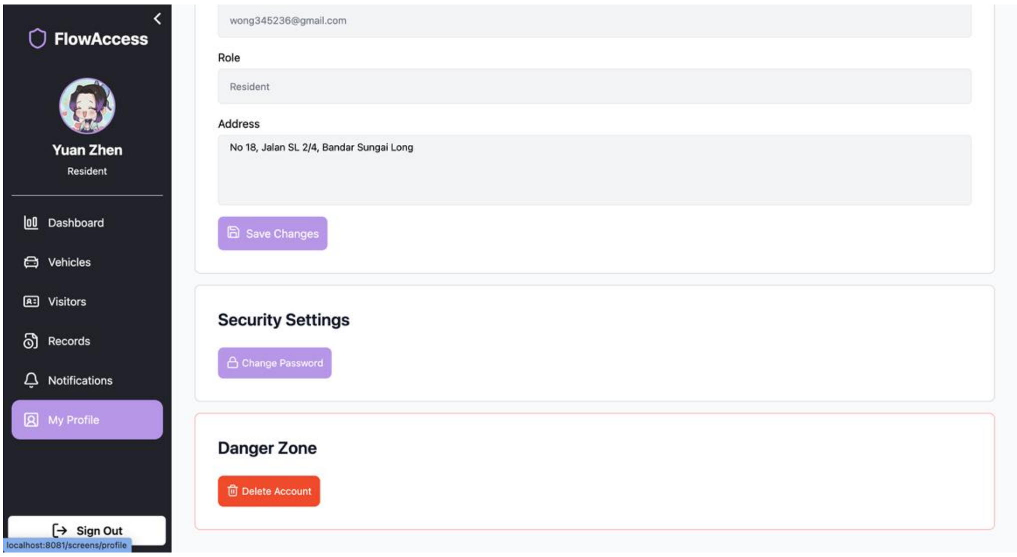
Phone Number
0173458076

Email
wong345236@gmail.com

Role
Resident

Address
No 18, Jalan SL 2/4, Bandar Sungai Long

Figure 5.13: Resident Update Profile Page UI Part 1



wong345236@gmail.com

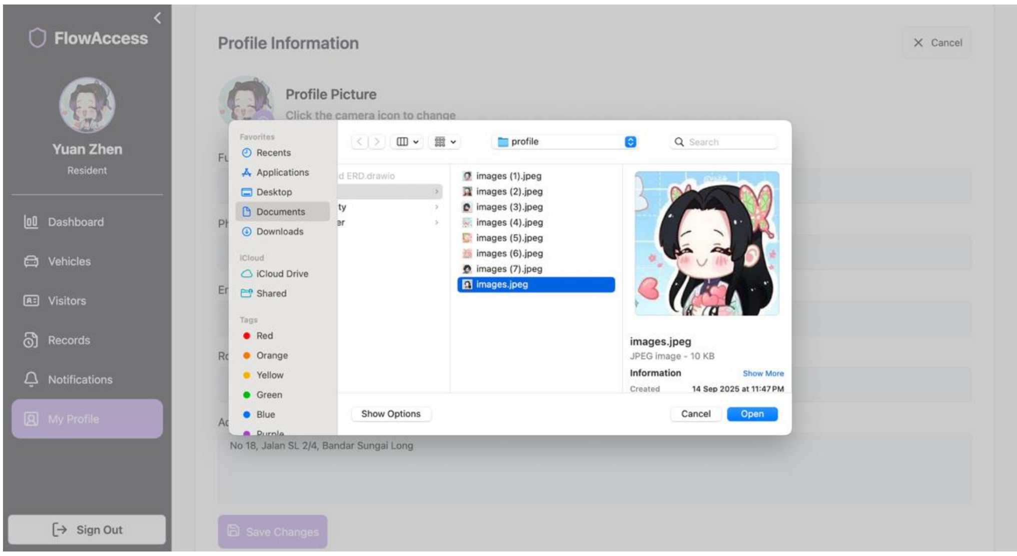
Role
Resident

Address
No 18, Jalan SL 2/4, Bandar Sungai Long

Security Settings

Danger Zone

Figure 5.14: Resident Update Profile Page UI Part 2



Profile Information

Profile Picture

Click the camera icon to change

images.jpeg

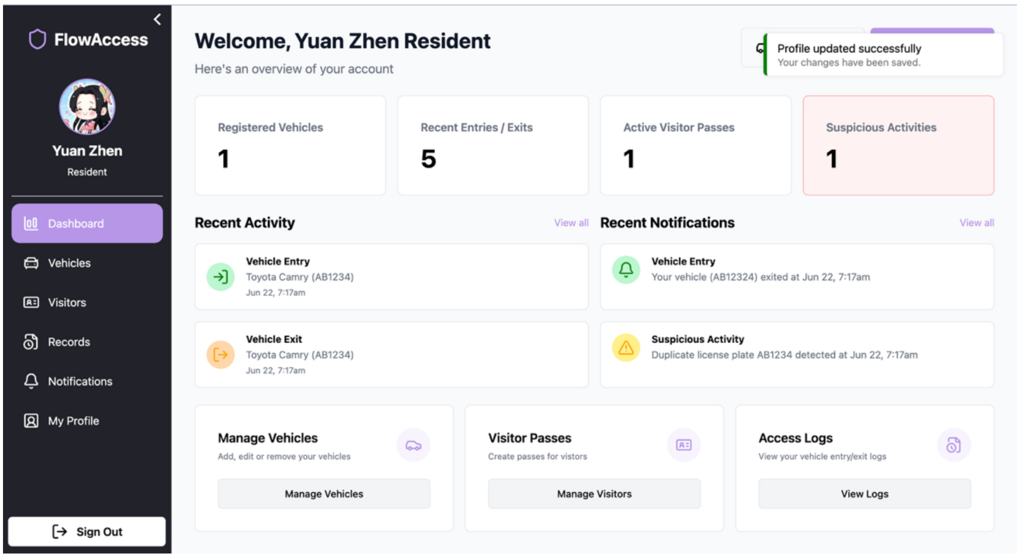
JPEG image - 10 KB

Information Show More

Created 14 Sep 2025 at 11:47 PM

No 18, Jalan SL 2/4, Bandar Sungai Long

Figure 5.15: Resident Upload Profile Image UI



Welcome, Yuan Zhen Resident

Here's an overview of your account

Profile updated successfully
Your changes have been saved.

Registered Vehicles	Recent Entries / Exits	Active Visitor Passes	Suspicious Activities
1	5	1	1

Recent Activity

Vehicle Entry
Toyota Camry (AB1234)
Jun 22, 7:17am

Vehicle Exit
Toyota Camry (AB1234)
Jun 22, 7:17am

Recent Notifications

Vehicle Entry
Your vehicle (AB12324) exited at Jun 22, 7:17am

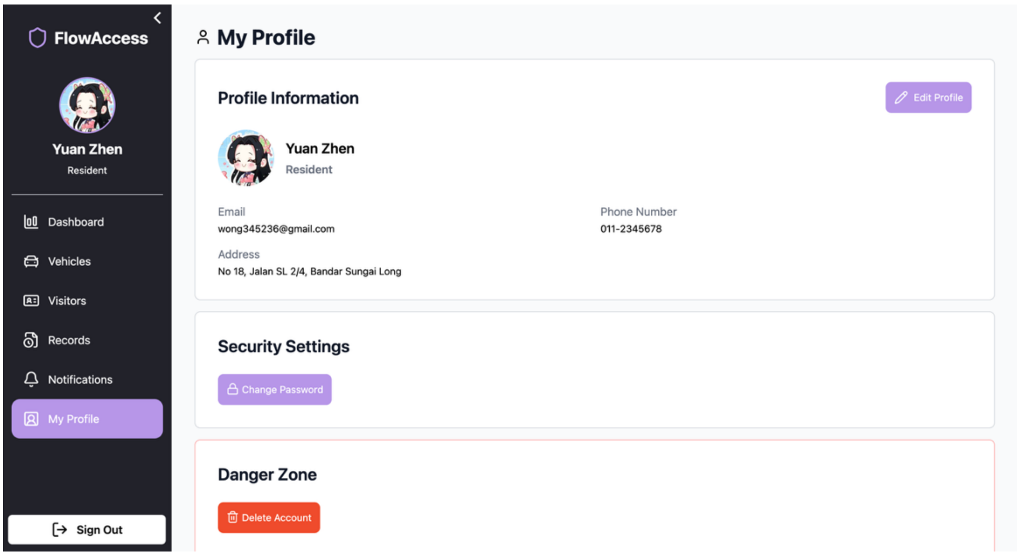
Suspicious Activity
Duplicate license plate AB1234 detected at Jun 22, 7:17am

Manage Vehicles
Add, edit or remove your vehicles

Visitor Passes
Create passes for vistors

Access Logs
View your vehicle entry/exit logs

Figure 5.16: Resident Update Profile Successful UI



Profile Information

Yuan Zhen
Resident

Email
wong345236@gmail.com

Phone Number
011-2345678

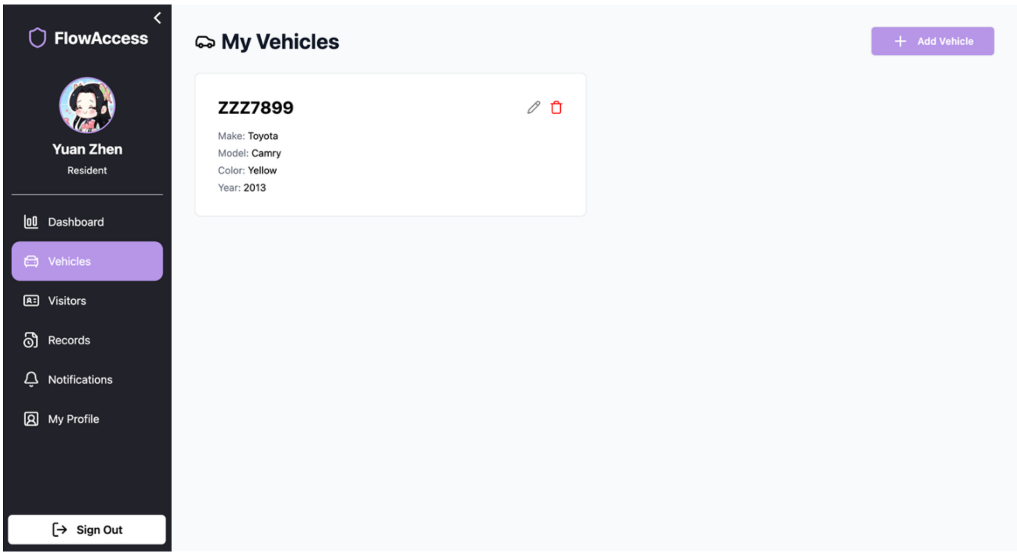
Address
No 18, Jalan SL 2/4, Bandar Sungai Long

Security Settings

Danger Zone

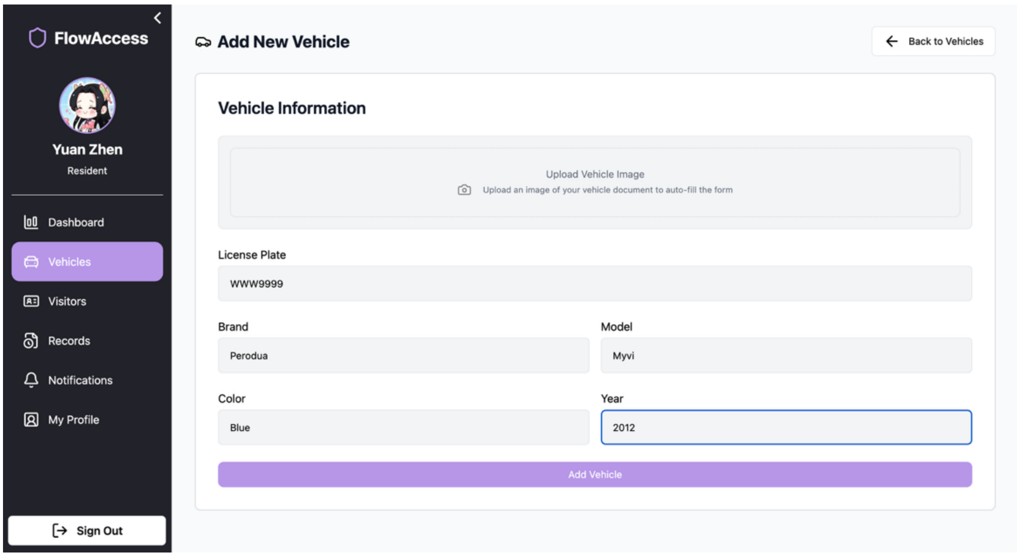
Figure 5.17: Resident New Updated Profile Page UI

5.4.1.4 Manage Vehicle



ZZZ7899
Make: Toyota
Model: Camry
Color: Yellow
Year: 2013

Figure 5.18: Resident Vehicle Page UI



Add New Vehicle

Vehicle Information

License Plate: WWW9999

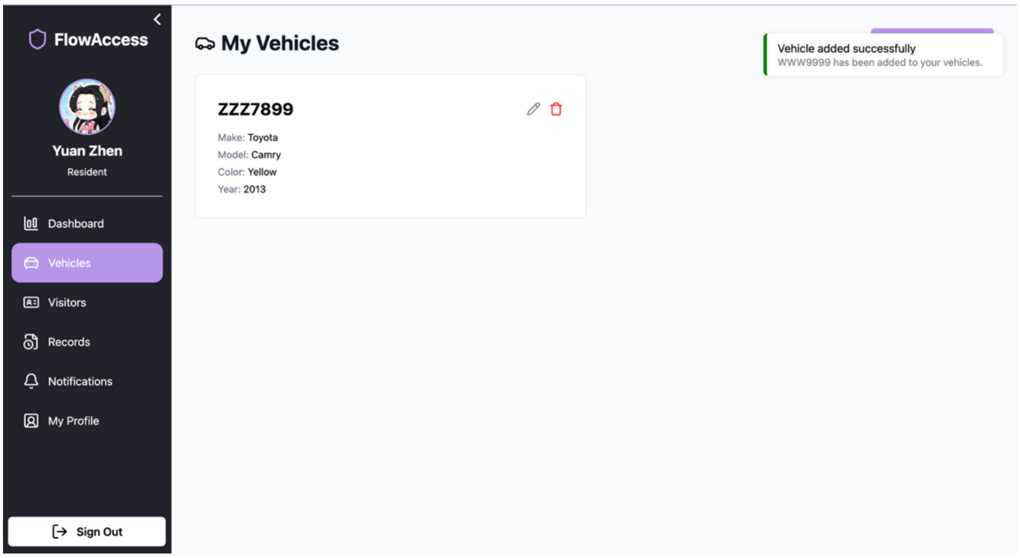
Brand: Perodua

Model: Myvi

Color: Blue

Year: 2012

Figure 5.19: Resident Add New Vehicle Page UI



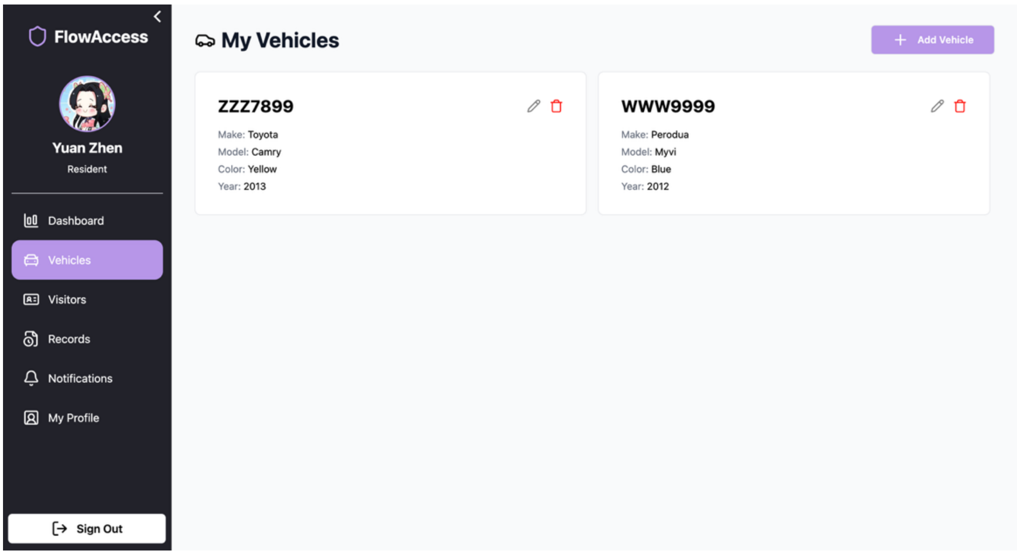
My Vehicles

Vehicle added successfully
WWW9999 has been added to your vehicles.

ZZZ7899

Make: Toyota
Model: Camry
Color: Yellow
Year: 2013

Figure 5.20: Resident Add New Vehicle Successfully Page UI



My Vehicles

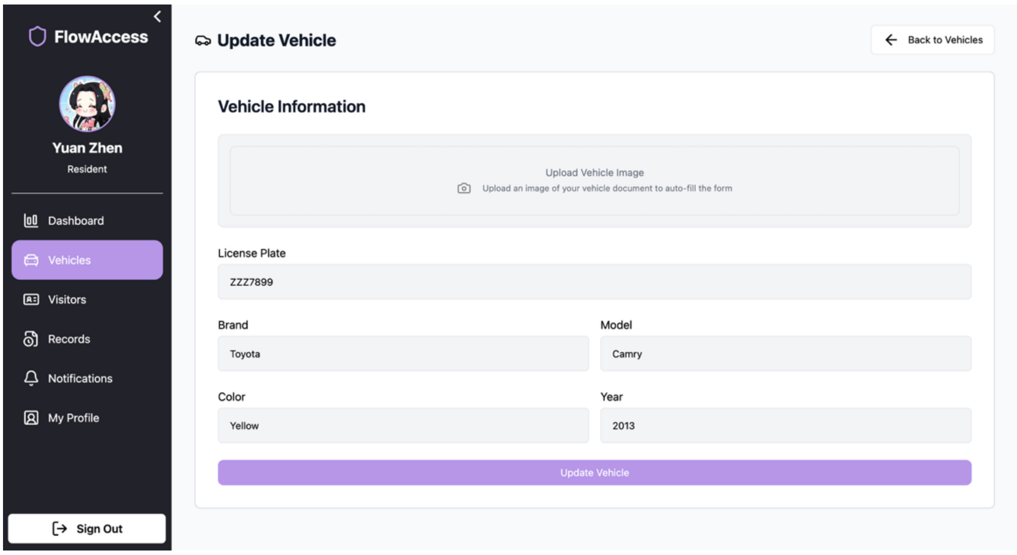
ZZZ7899

Make: Toyota
Model: Camry
Color: Yellow
Year: 2013

WWW9999

Make: Perodua
Model: Myvi
Color: Blue
Year: 2012

Figure 5.21: Resident New Added Vehicle Page UI



Update Vehicle

Vehicle Information

Upload Vehicle Image
Upload an image of your vehicle document to auto-fill the form

License Plate: ZZZ7899

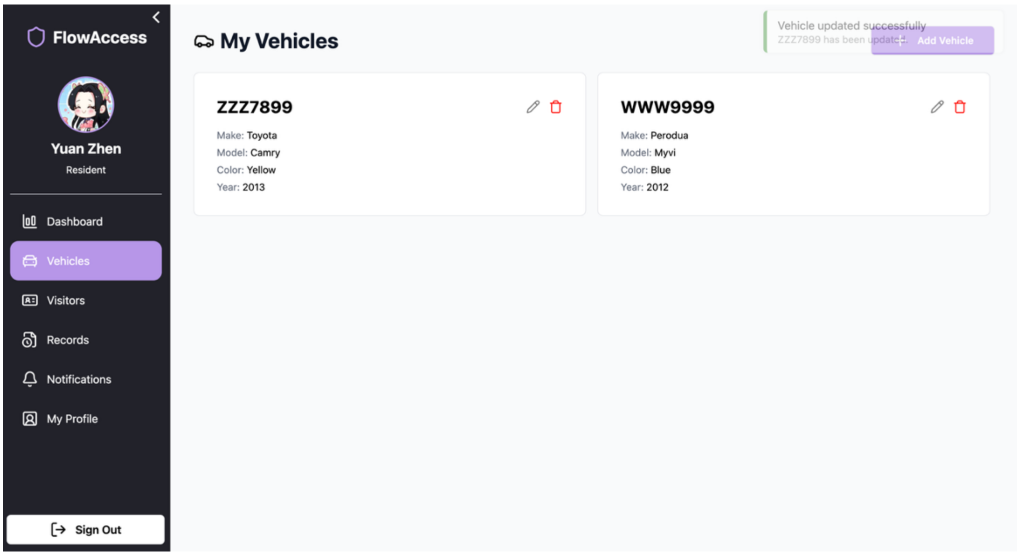
Brand: Toyota

Model: Camry

Color: Yellow

Year: 2013

Figure 5.22: Resident Update Vehicle Page UI



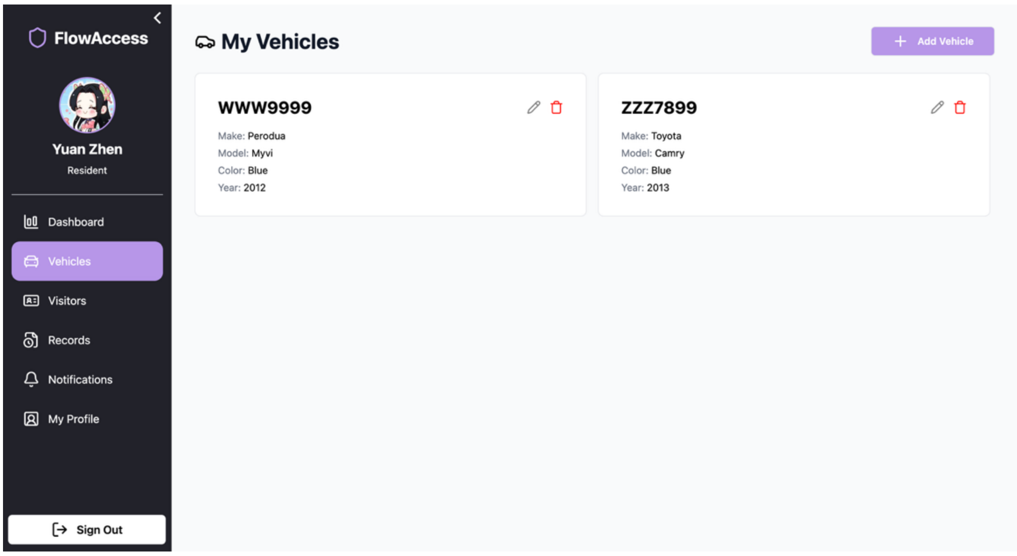
My Vehicles

Vehicle updated successfully
ZZZ7899 has been updated.

ZZZ7899
Make: Toyota
Model: Camry
Color: Yellow
Year: 2013

WWW9999
Make: Perodua
Model: Myvi
Color: Blue
Year: 2012

Figure 5.23: Resident Update Vehicle Successfully Page UI

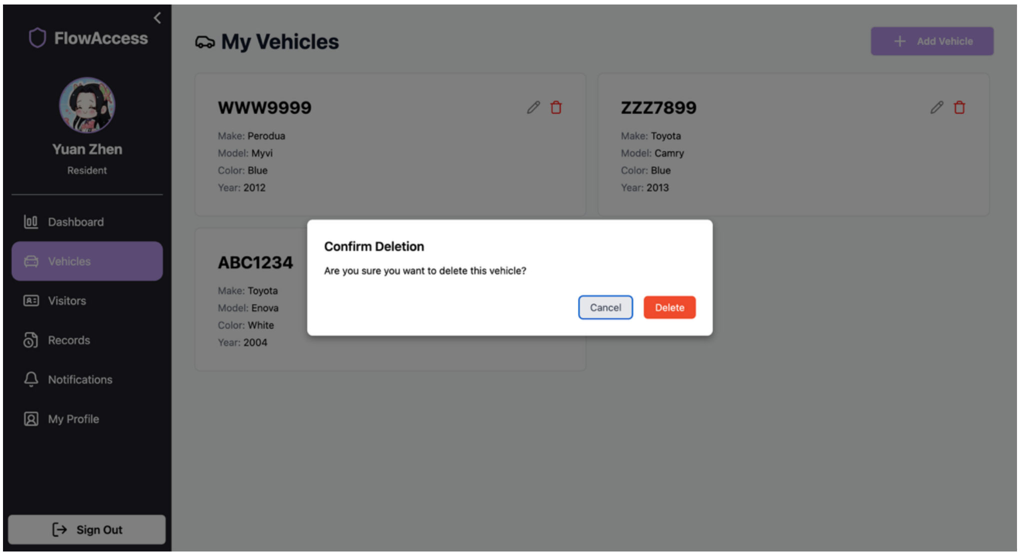


My Vehicles

WWW9999
Make: Perodua
Model: Myvi
Color: Blue
Year: 2012

ZZZ7899
Make: Toyota
Model: Camry
Color: Blue
Year: 2013

Figure 5.24: Resident New Updated Vehicle Page UI



WWW9999
Make: Perodua
Model: Myvi
Color: Blue
Year: 2012

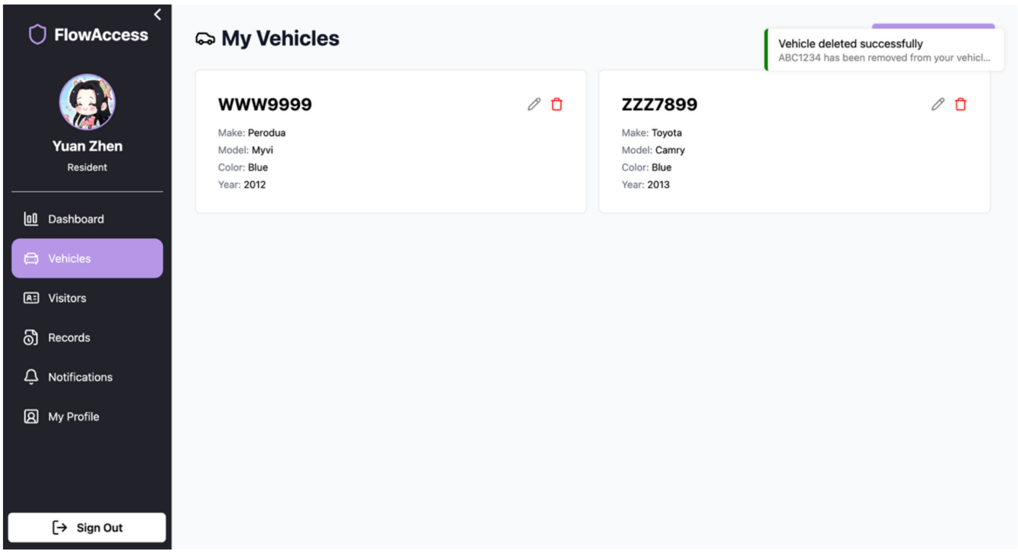
ZZZ7899
Make: Toyota
Model: Camry
Color: Blue
Year: 2013

ABC1234
Make: Toyota
Model: Enova
Color: White
Year: 2004

Confirm Deletion

Are you sure you want to delete this vehicle?

Figure 5.25: Resident Delete Vehicle Page UI



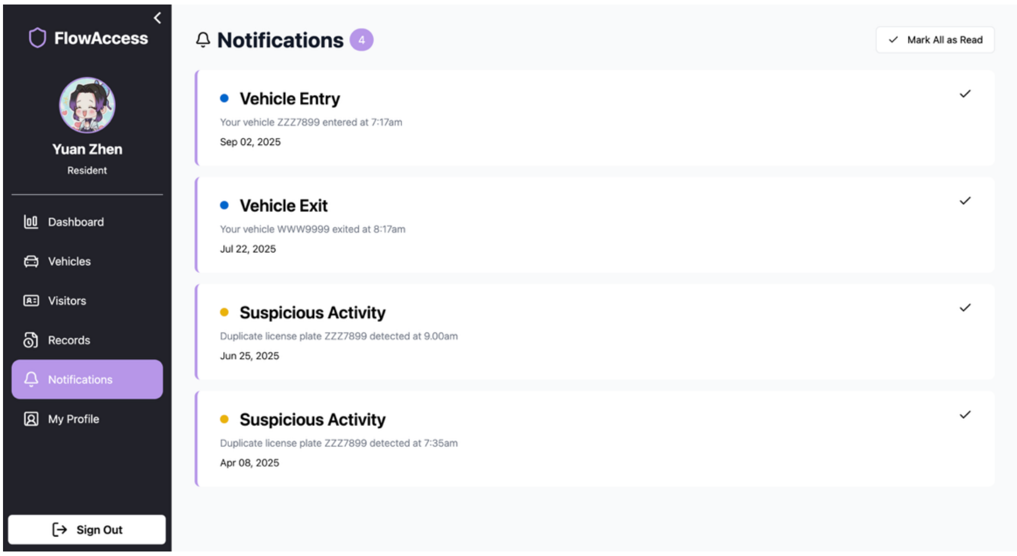
Vehicle deleted successfully
ABC1234 has been removed from your vehicl...

WWW9999
Make: Perodua
Model: Myvi
Color: Blue
Year: 2012

ZZZ7899
Make: Toyota
Model: Camry
Color: Blue
Year: 2013

Figure 5.26: Resident Delete Vehicle Successful Page UI

5.4.1.5 Notifications



Notifications 4

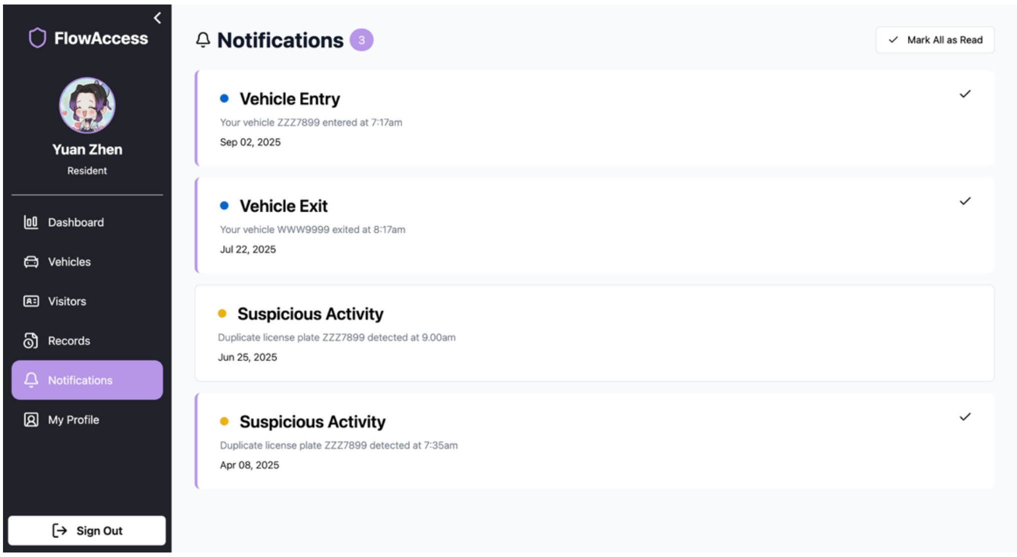
Vehicle Entry
Your vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025

Vehicle Exit
Your vehicle WWW9999 exited at 8:17am
Jul 22, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025

Figure 5.27: Resident Notification Page UI



Notifications 3

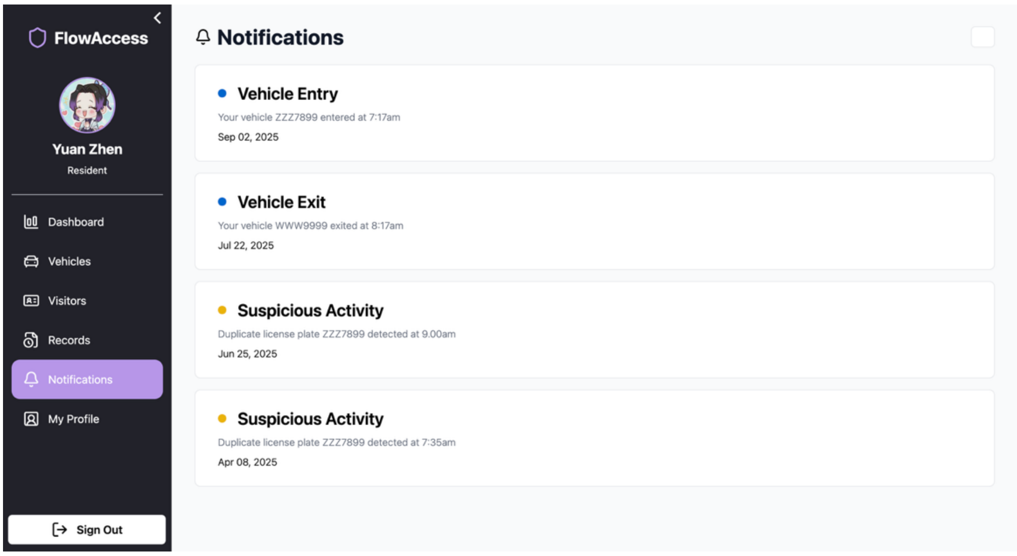
Vehicle Entry
Your vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025

Vehicle Exit
Your vehicle WWW9999 exited at 8:17am
Jul 22, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025

Figure 5.28: Resident Mark 1 Notification as Read UI



Notifications

Vehicle Entry
Your vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025

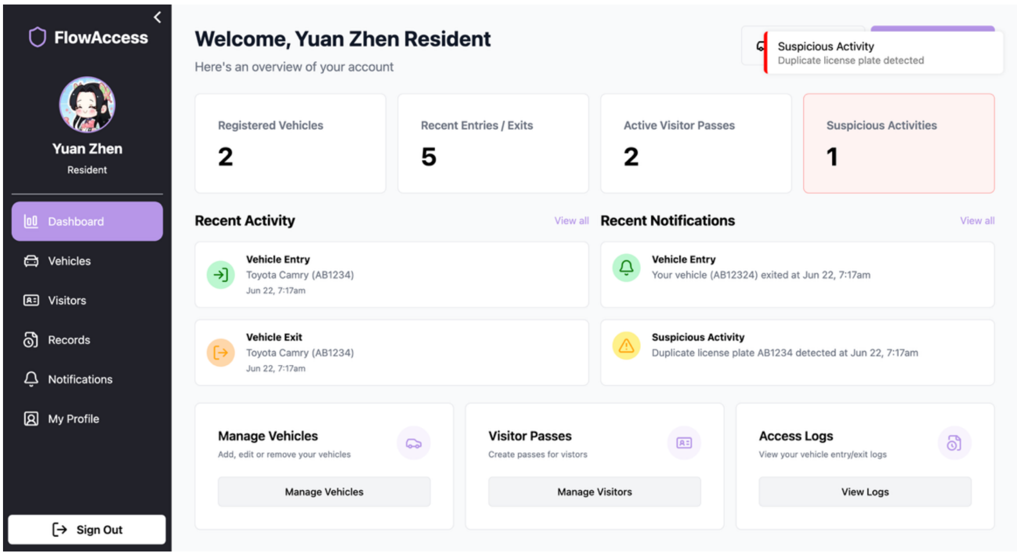
Vehicle Exit
Your vehicle WWW9999 exited at 8:17am
Jul 22, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025

Figure 5.29: Resident Mark All Notification as Read Page UI

5.4.1.6 Suspicious Activity Alerts



Welcome, Yuan Zhen Resident
Here's an overview of your account

Suspicious Activity
Duplicate license plate detected

Registered Vehicles 2

Recent Entries / Exits 5

Active Visitor Passes 2

Suspicious Activities 1

Recent Activity — View all

Vehicle Entry
Toyota Camry (AB1234)
Jun 22, 7:17am

Vehicle Exit
Toyota Camry (AB1234)
Jun 22, 7:17am

Recent Notifications — View all

Vehicle Entry
Your vehicle (AB12324) exited at Jun 22, 7:17am

Suspicious Activity
Duplicate license plate AB1234 detected at Jun 22, 7:17am

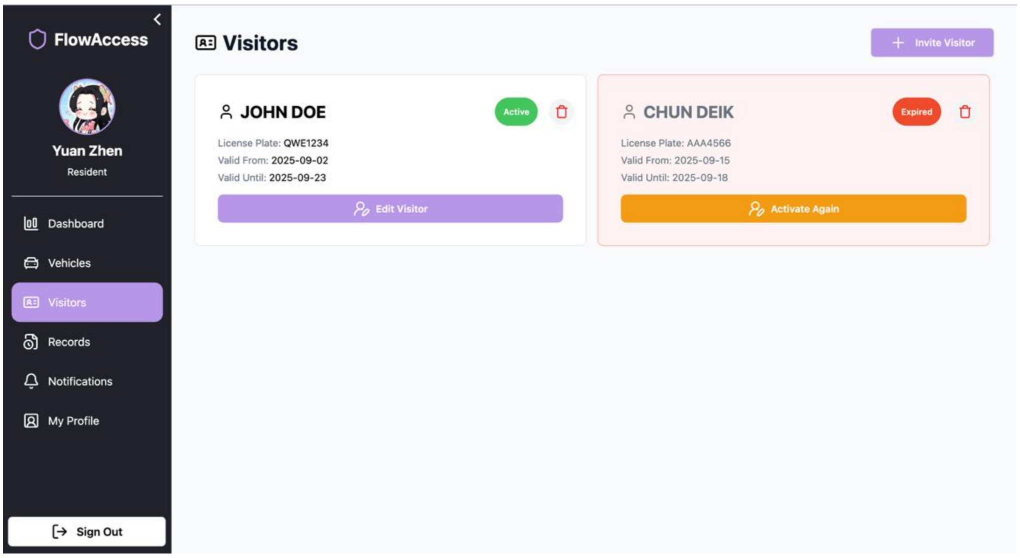
Manage Vehicles
Add, edit or remove your vehicles
Manage Vehicles

Visitor Passes
Create passes for vistors
Manage Visitors

Access Logs
View your vehicle entry/exit logs
View Logs

Figure 5.30: Resident Receive Suspicious Alert Page UI

5.4.1.7 Manage Visitor Pass

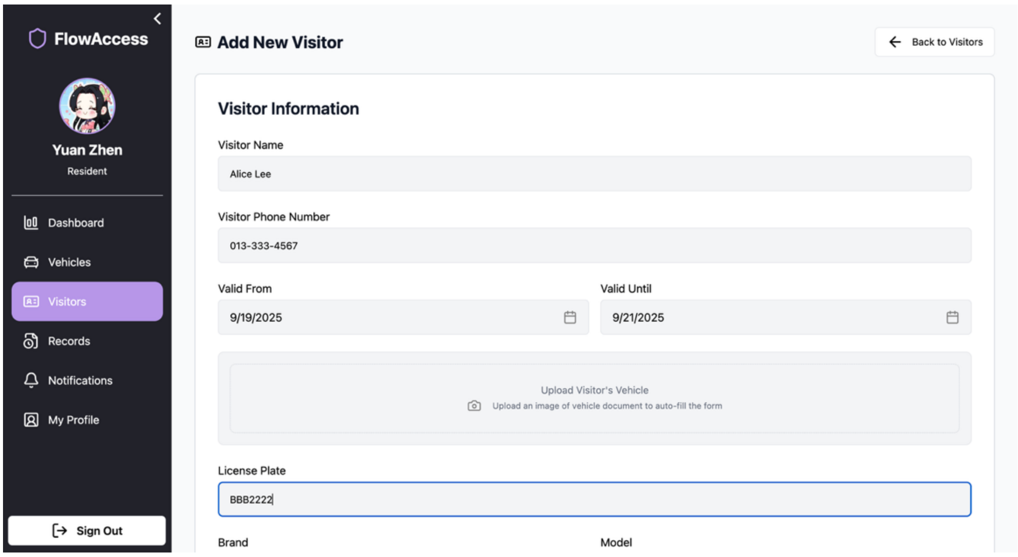


Visitors

JOHN DOE
License Plate: QWE1234
Valid From: 2025-09-02
Valid Until: 2025-09-23

CHUN DEIK
License Plate: AAA4566
Valid From: 2025-09-15
Valid Until: 2025-09-18

Figure 5.31: Resident Visitor Page UI



Add New Visitor

Visitor Information

Visitor Name: Alice Lee

Visitor Phone Number: 013-333-4567

Valid From: 9/19/2025

Valid Until: 9/21/2025

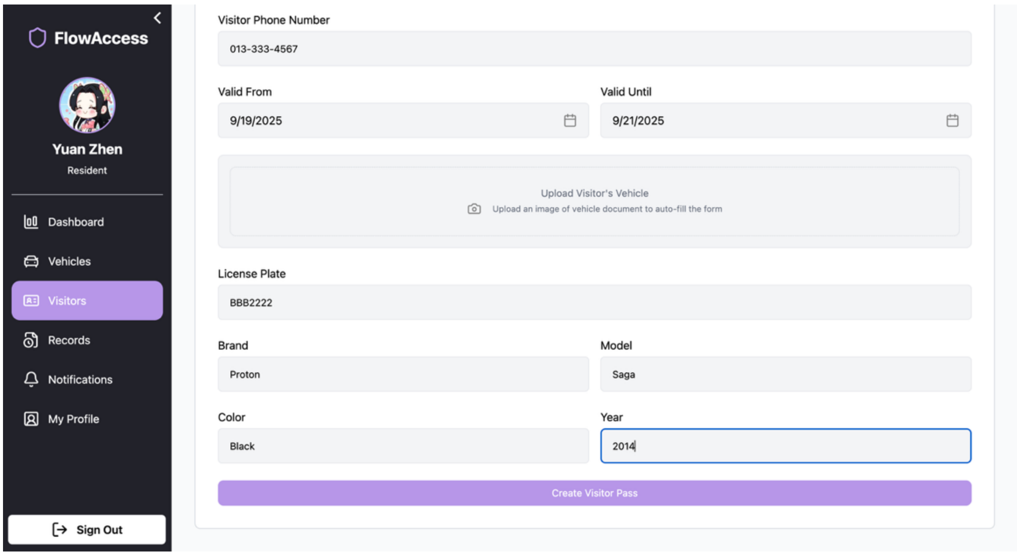
Upload Visitor's Vehicle
Upload an image of vehicle document to auto-fill the form

License Plate: BBB2222

Brand

Model

Figure 5.32: Resident Add New Visitor Page UI Part 1



Visitor Phone Number: 013-333-4567

Valid From: 9/19/2025

Valid Until: 9/21/2025

Upload Visitor's Vehicle

Upload an image of vehicle document to auto-fill the form

License Plate: BBB2222

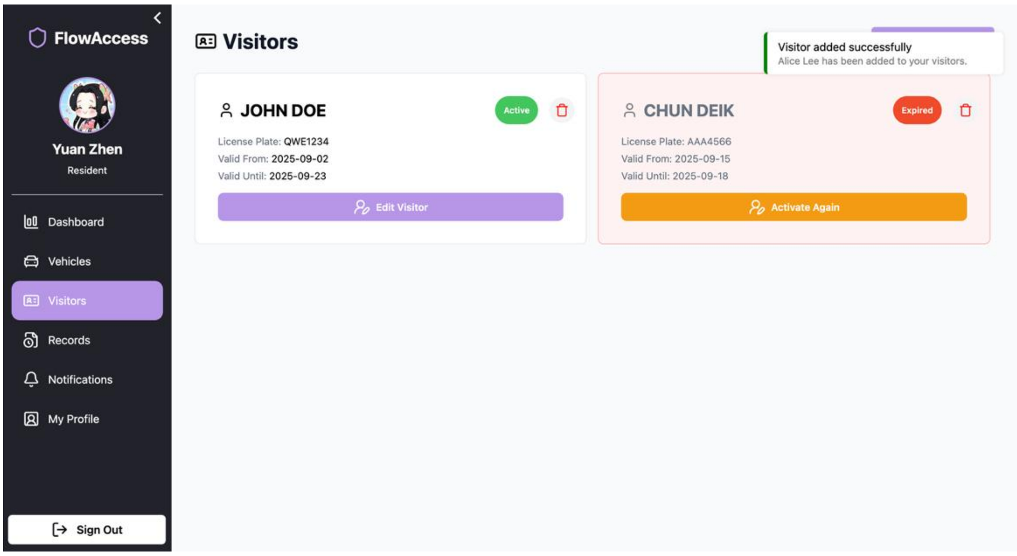
Brand: Proton

Model: Saga

Color: Black

Year: 2014

Figure 5.33: Resident Add New Visitor Page UI Part 2



Visitors

Visitor added successfully

Alice Lee has been added to your visitors.

JOHN DOE — Active

License Plate: QWE1234

Valid From: 2025-09-02

Valid Until: 2025-09-23

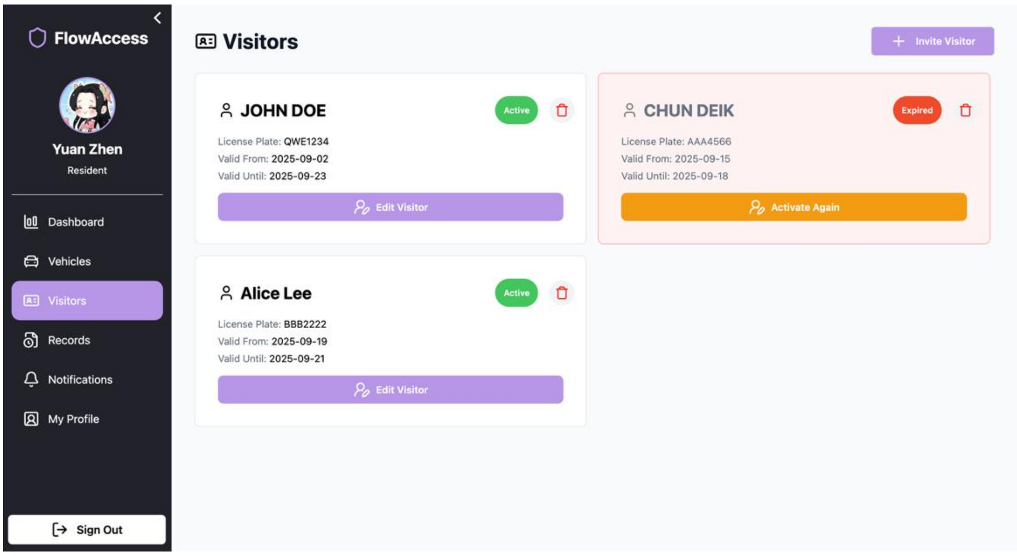
CHUN DEIK — Expired

License Plate: AAA4566

Valid From: 2025-09-15

Valid Until: 2025-09-18

Figure 5.34: Resident Add New Visitor Successful Page UI



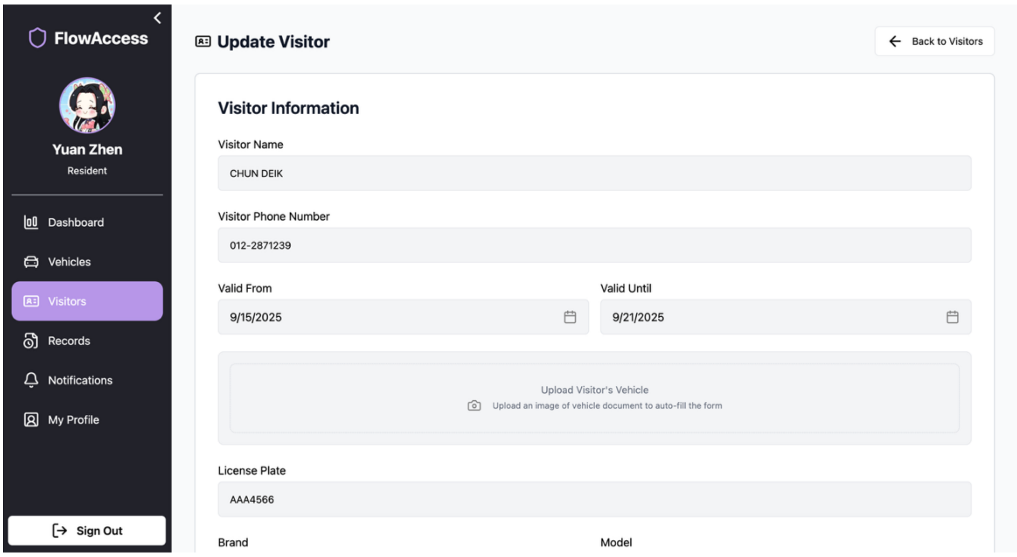
Visitors

JOHN DOE — Active
License Plate: QWE1234
Valid From: 2025-09-02
Valid Until: 2025-09-23

CHUN DEIK — Expired
License Plate: AAA4566
Valid From: 2025-09-15
Valid Until: 2025-09-18

Alice Lee — Active
License Plate: BBB2222
Valid From: 2025-09-19
Valid Until: 2025-09-21

Figure 5.35: Resident New Add Visitor Page UI



Update Visitor

Visitor Information

Visitor Name: CHUN DEIK

Visitor Phone Number: 012-2871239

Valid From: 9/15/2025

Valid Until: 9/21/2025

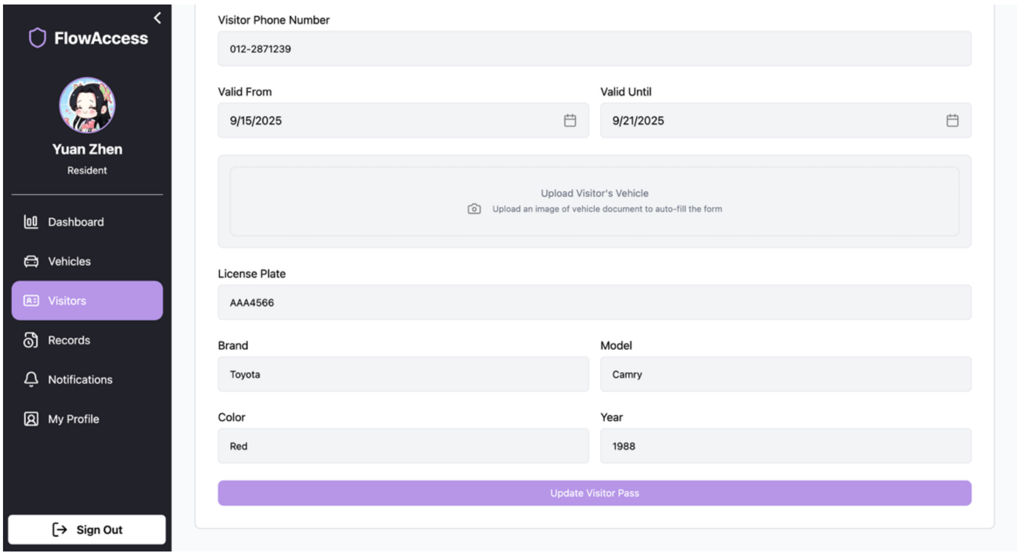
Upload Visitor's Vehicle
Upload an image of vehicle document to auto-fill the form

License Plate: AAA4566

Brand

Model

Figure 5.36: Resident Update Visitor Page UI Part 1



Visitor Phone Number: 012-2871239

Valid From: 9/15/2025

Valid Until: 9/21/2025

License Plate: AAA4566

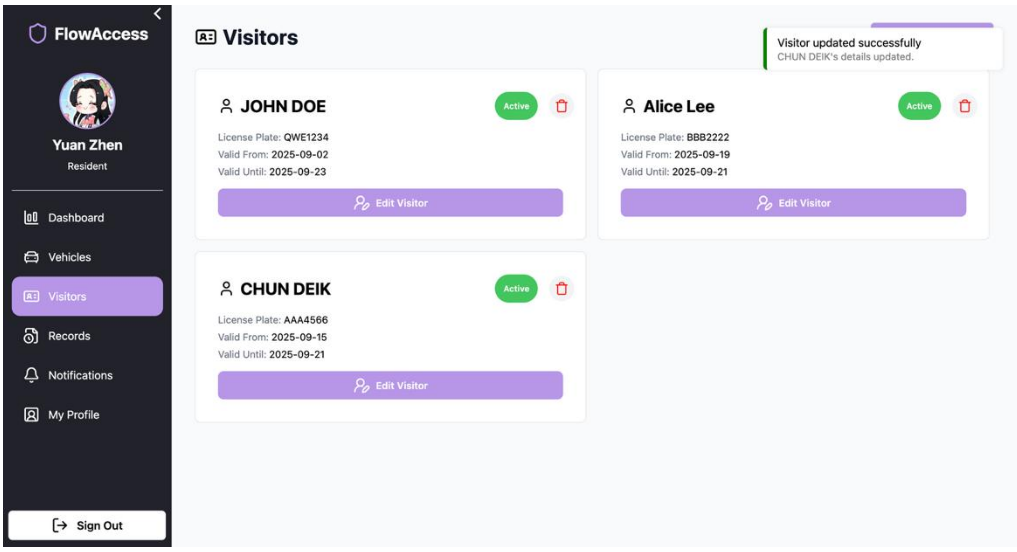
Brand: Toyota

Model: Camry

Color: Red

Year: 1988

Figure 5.37: Resident Update Visitor Page UI Part 2



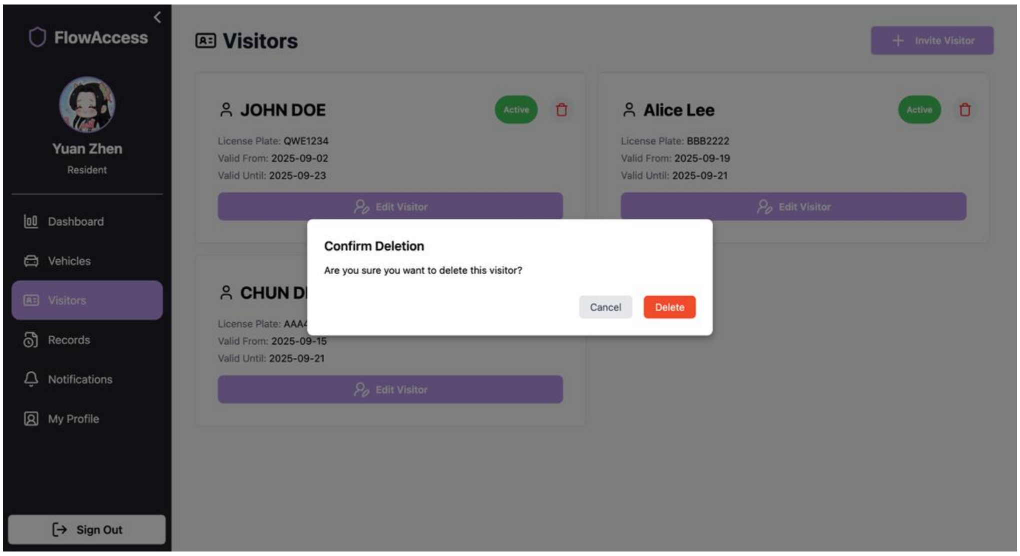
Visitor updated successfully
CHUN DEIK's details updated.

JOHN DOE
License Plate: QWE1234
Valid From: 2025-09-02
Valid Until: 2025-09-23

Alice Lee
License Plate: BBB2222
Valid From: 2025-09-19
Valid Until: 2025-09-21

CHUN DEIK
License Plate: AAA4566
Valid From: 2025-09-15
Valid Until: 2025-09-21

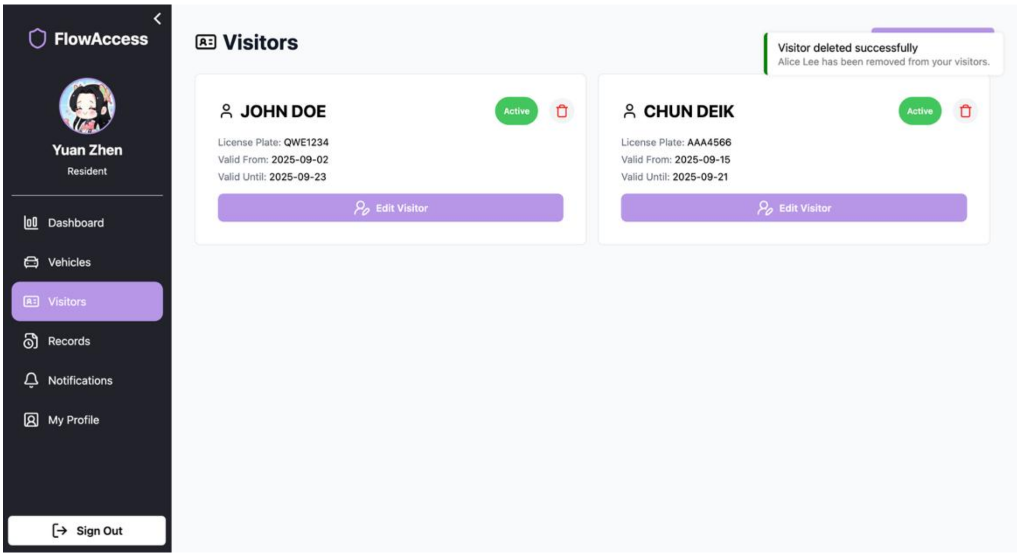
Figure 5.38: Resident Update Visitor Success Page UI



Confirm Deletion

Are you sure you want to delete this visitor?

Figure 5.39: Resident Delete Visitor Page UI



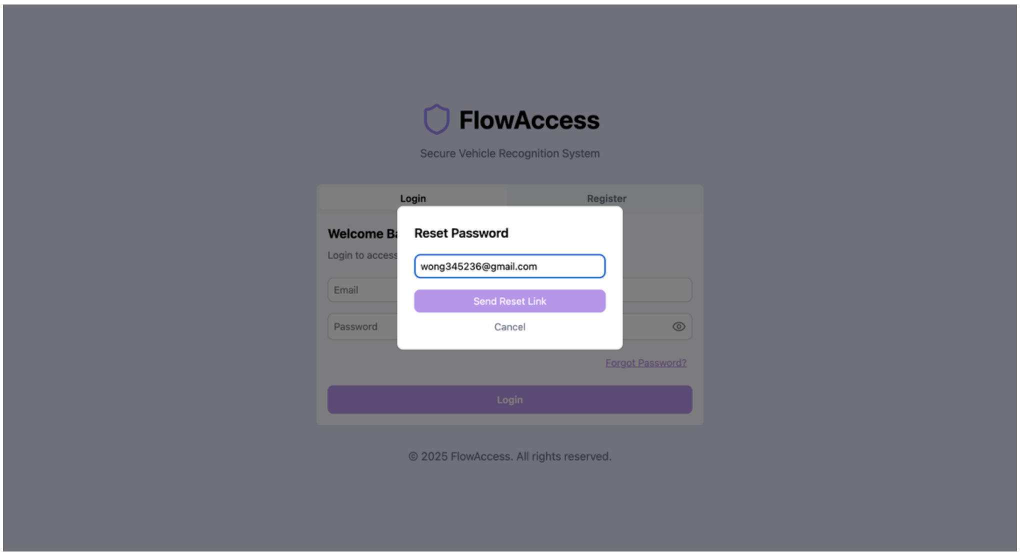
Visitor deleted successfully
Alice Lee has been removed from your visitors.

JOHN DOE
License Plate: QWE1234
Valid From: 2025-09-02
Valid Until: 2025-09-23

CHUN DEIK
License Plate: AAA4566
Valid From: 2025-09-15
Valid Until: 2025-09-21

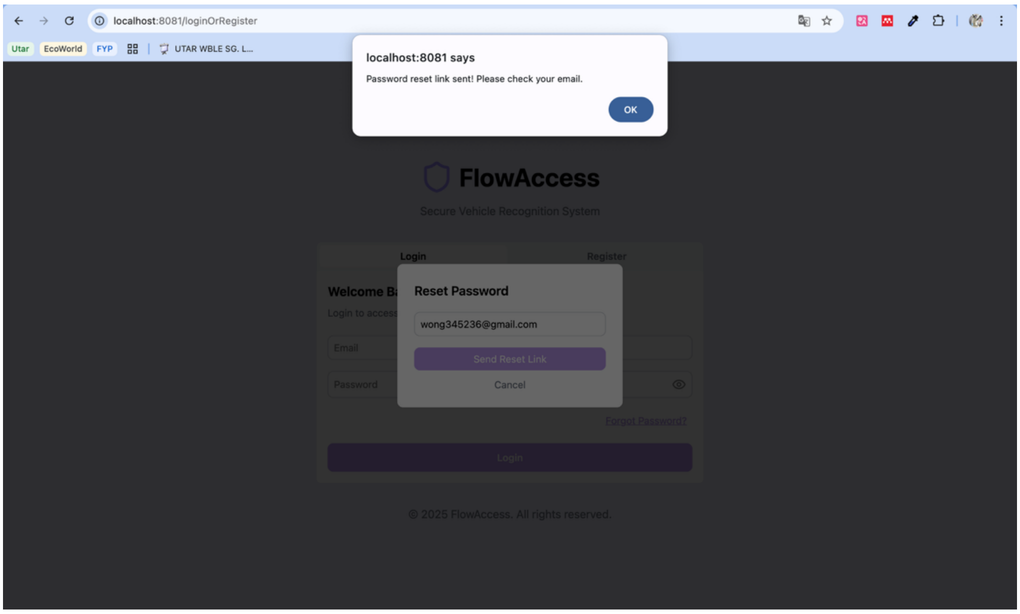
Figure 5.40: Resident Delete Visitor Success Page UI

5.4.1.8 Reset Password



wong345236@gmail.com

Figure 5.41: Resident Reset Password Page UI



localhost:8081/loginOrRegister

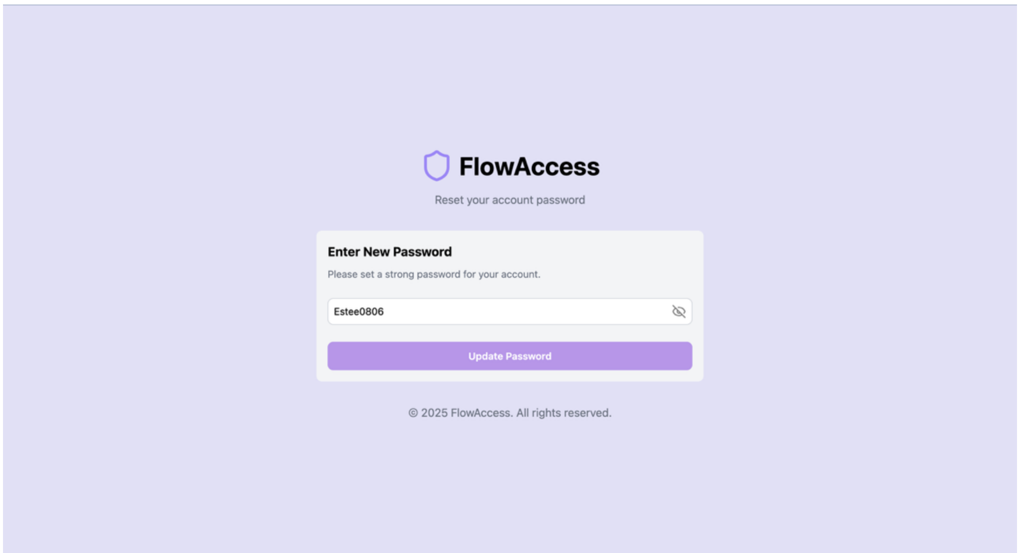
localhost:8081 says

Password reset link sent! Please check your email.

Reset Password

wong345236@gmail.com

Figure 5.42: Resident Reset Password Accept Page UI



Enter New Password

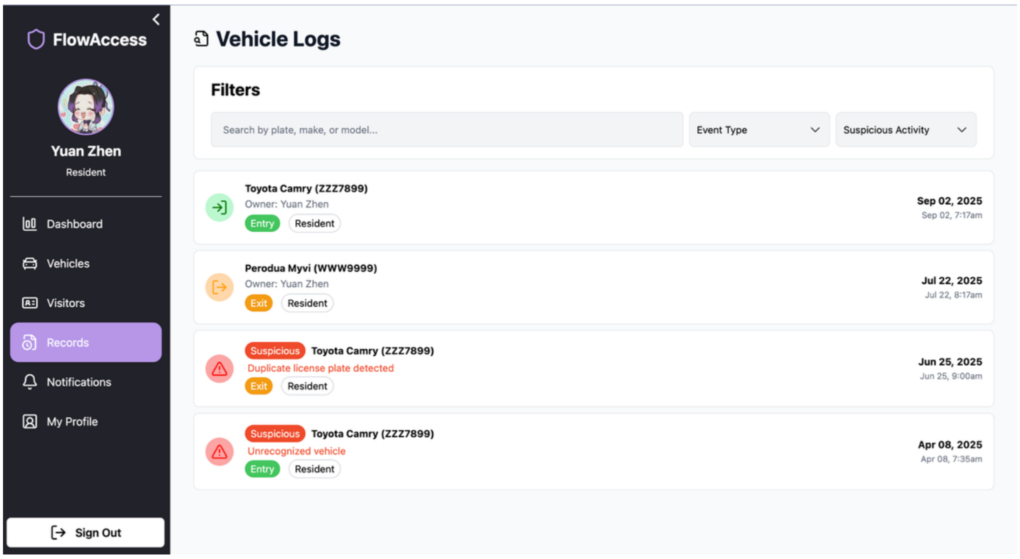
Please set a strong password for your account.

Estee0806

FlowAccess
Reset your account password
Update Password
© 2025 FlowAccess. All rights reserved.

Figure 5.43: Resident Reset Password Page

5.4.1.9 Vehicle Logs



Vehicle Logs

Filters

Search by plate, make, or model...

Event Type

Suspicious Activity

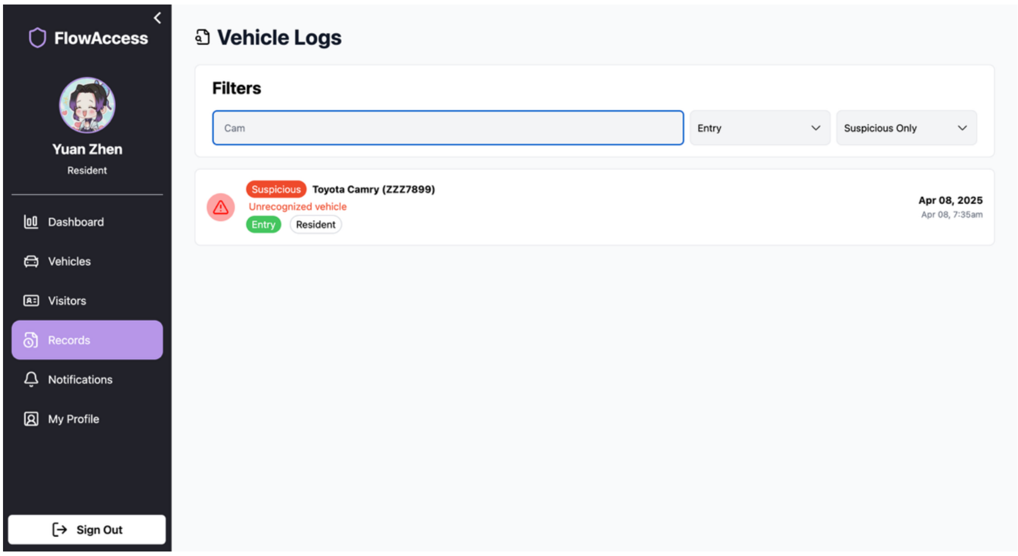
Toyota Camry (ZZZ7899)
Owner: Yuan Zhen
Entry Resident
Sep 02, 2025
Sep 02, 7:17am

Perodua Myvi (WWW9999)
Owner: Yuan Zhen
Exit Resident
Jul 22, 2025
Jul 22, 8:17am

Suspicious Toyota Camry (ZZZ7899)
Duplicate license plate detected
Exit Resident
Jun 25, 2025
Jun 25, 9:00am

Suspicious Toyota Camry (ZZZ7899)
Unrecognized vehicle
Entry Resident
Apr 08, 2025
Apr 08, 7:35am

Figure 5.44: Resident Vehicle Log Page UI



Vehicle Logs

Filters

Cam

Entry

Suspicious Only

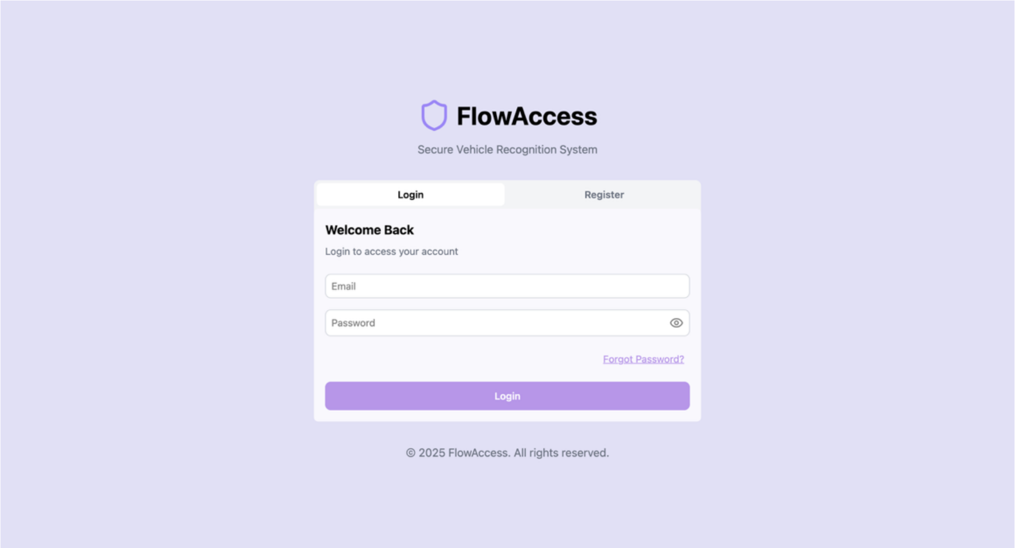
Suspicious Toyota Camry (ZZZ7899)
Unrecognized vehicle
Entry Resident
Apr 08, 2025
Apr 08, 7:35am

Figure 5.45: Resident Vehicle Log Apply Searach and Filter Page UI

5.4.2 Security Guard Side

Security Guard Side UI Design focuses on creating an intuitive and user-friendly interface for security guard to interact with the system.

5.4.2.1 Login



FlowAccess

Secure Vehicle Recognition System

Login | Register

Welcome Back

Login to access your account

Email

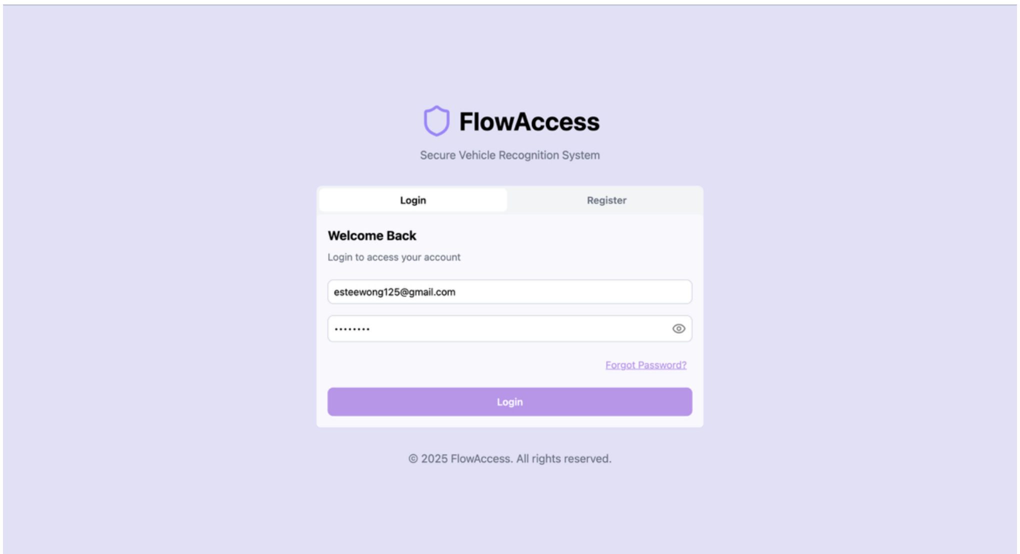
Password

Forgot Password?

Login

© 2025 FlowAccess. All rights reserved.

Figure 5.46: Security Guard Login Page UI



FlowAccess

Secure Vehicle Recognition System

Login | Register

Welcome Back

Login to access your account

esteewong125@gmail.com

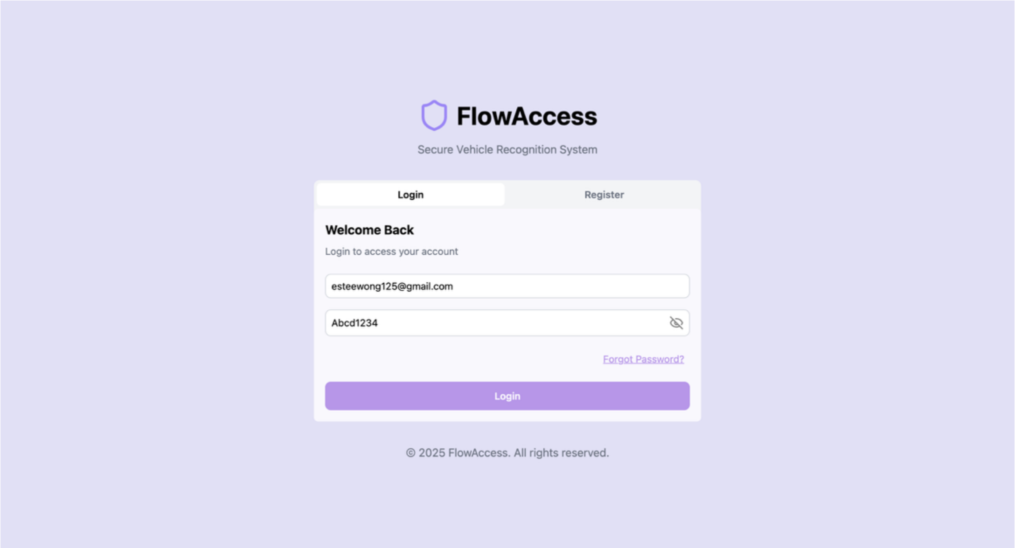
••••••••

Forgot Password?

Login

© 2025 FlowAccess. All rights reserved.

Figure 5.47: Security Guard Login without readability password Page UI


FlowAccess

Secure Vehicle Recognition System

Welcome Back

Login to access your account

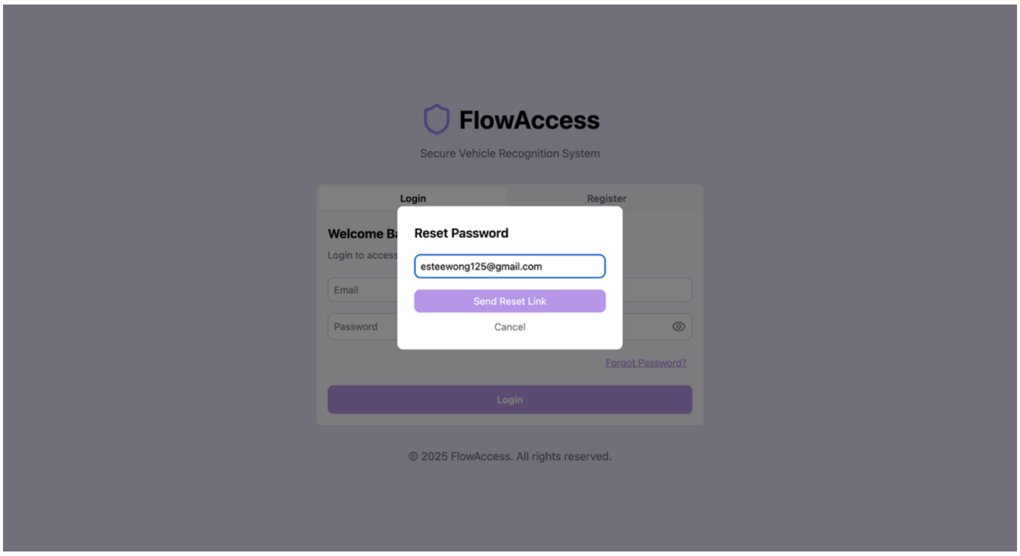
esteewong125@gmail.com

Abcd1234

© 2025 FlowAccess. All rights reserved.

Figure 5.48: Security Guard Login with readability password

5.4.2.2 Reset Password


FlowAccess

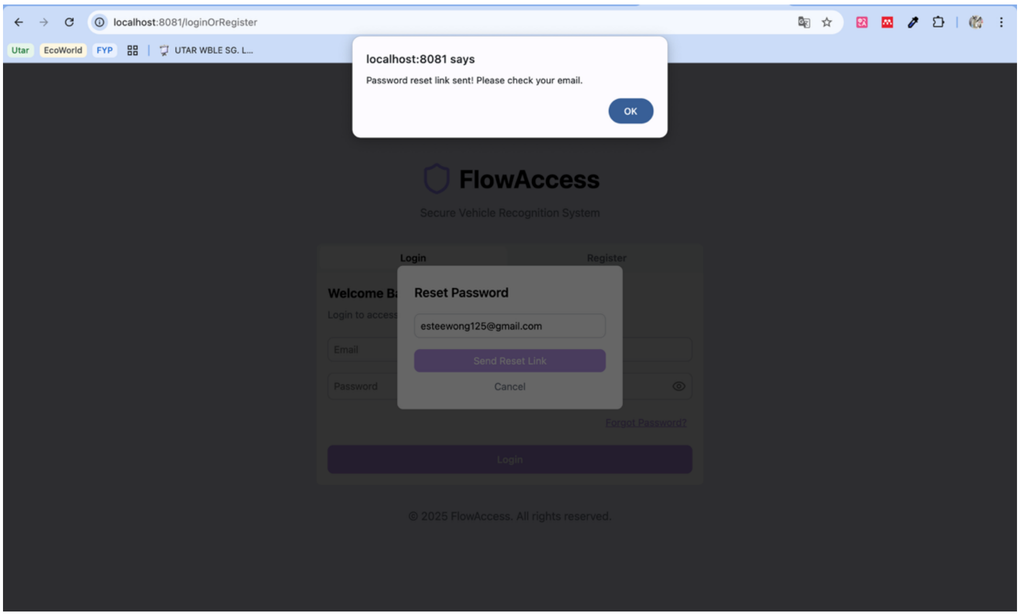
Secure Vehicle Recognition System

Reset Password

esteewong125@gmail.com

© 2025 FlowAccess. All rights reserved.

Figure 5.49: Security Guard Reset Password Page UI



localhost:8081/loginOrRegister

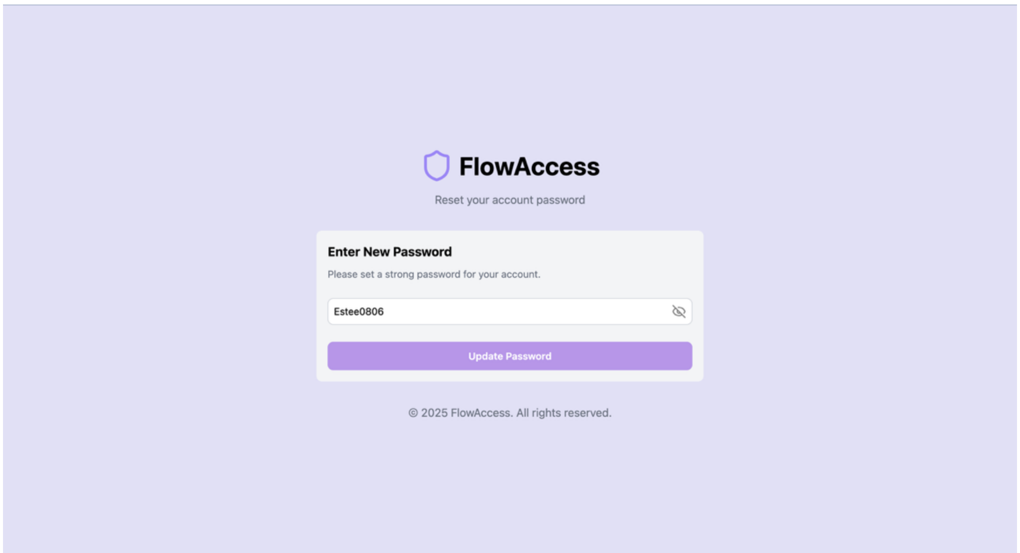
localhost:8081 says

Password reset link sent! Please check your email.

Reset Password

esteewong125@gmail.com

Figure 5.50: Security Guard Reset Password Accept Page UI



Enter New Password

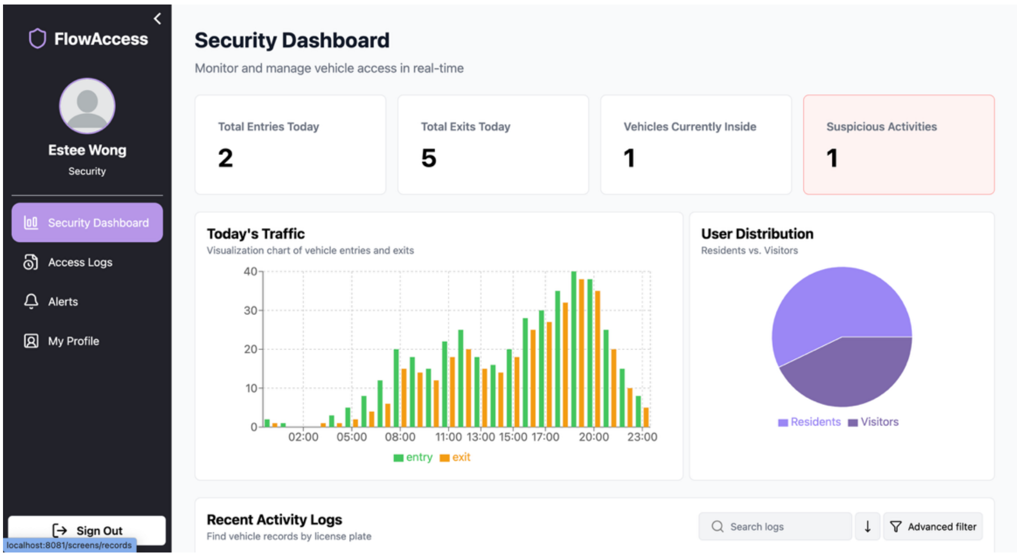
Please set a strong password for your account.

Estee0806

FlowAccess
Reset your account password
Update Password
© 2025 FlowAccess. All rights reserved.

Figure 5.51: Security Guard Reset Password Page UI

5.4.2.3 Dashboard



Security Dashboard

Monitor and manage vehicle access in real-time

Total Entries Today	Total Exits Today	Vehicles Currently Inside	Suspicious Activities
2	5	1	1

Today's Traffic

Visualization chart of vehicle entries and exits

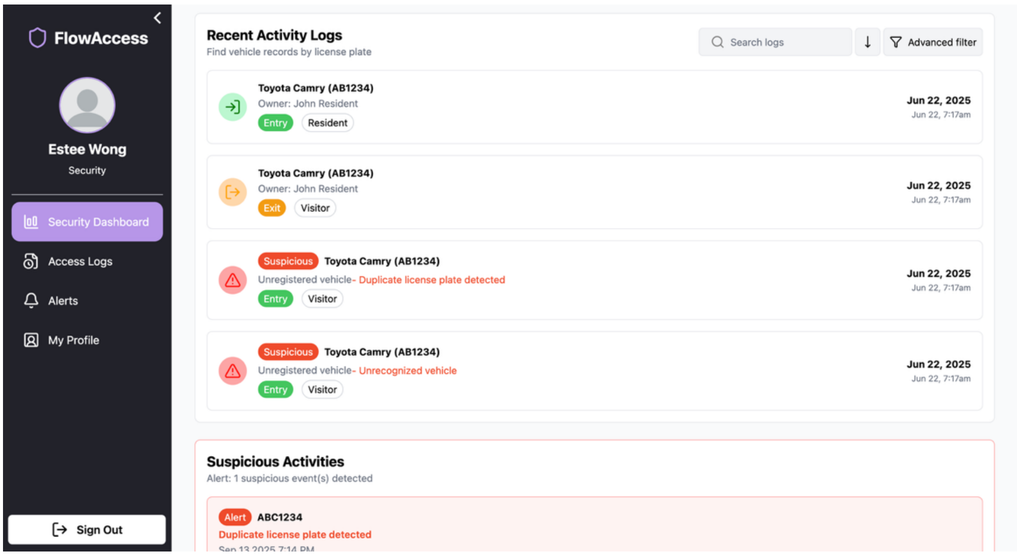
User Distribution

Residents vs. Visitors

Recent Activity Logs

Find vehicle records by license plate

Figure 5.52: Security Guard Dahsboard Page UI Part 1



Recent Activity Logs

Find vehicle records by license plate

Toyota Camry (AB1234)
Owner: John Resident
Entry Resident
Jun 22, 2025
Jun 22, 7:17am

Toyota Camry (AB1234)
Owner: John Resident
Exit Visitor
Jun 22, 2025
Jun 22, 7:17am

Suspicious Toyota Camry (AB1234)
Unregistered vehicle- Duplicate license plate detected
Entry Visitor
Jun 22, 2025
Jun 22, 7:17am

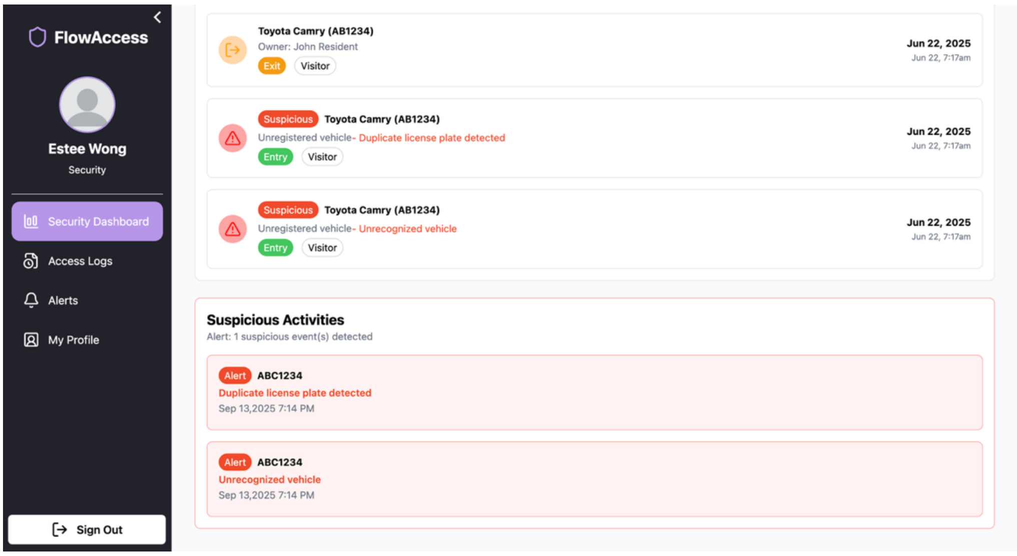
Suspicious Toyota Camry (AB1234)
Unregistered vehicle- Unrecognized vehicle
Entry Visitor
Jun 22, 2025
Jun 22, 7:17am

Suspicious Activities

Alert: 1 suspicious event(s) detected

Alert ABC1234
Duplicate license plate detected
Sep 13 2025 7:14 PM

Figure 5.53: Security Guard Dashboard Page UI Part 2



Toyota Camry (AB1234)
Owner: John Resident
Exit Visitor
Jun 22, 2025
Jun 22, 7:17am

Suspicious Toyota Camry (AB1234)
Unregistered vehicle- Duplicate license plate detected
Entry Visitor
Jun 22, 2025
Jun 22, 7:17am

Suspicious Toyota Camry (AB1234)
Unregistered vehicle- Unrecognized vehicle
Entry Visitor
Jun 22, 2025
Jun 22, 7:17am

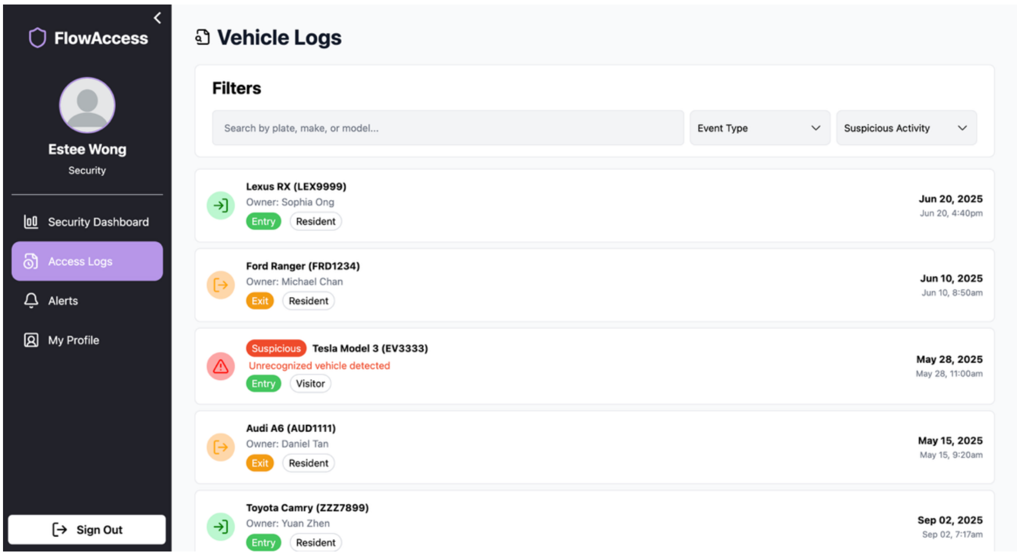
Suspicious Activities
Alert: 1 suspicious event(s) detected

Alert ABC1234
Duplicate license plate detected
Sep 13,2025 7:14 PM

Alert ABC1234
Unrecognized vehicle
Sep 13,2025 7:14 PM

Figure 5.54: Security Guard Dashboard Page UI Part 3

5.4.2.4 Vehicle Logs



Vehicle Logs

Lexus RX (LEX9999)
Owner: Sophia Ong
Entry Resident
Jun 20, 2025
Jun 20, 4:40pm

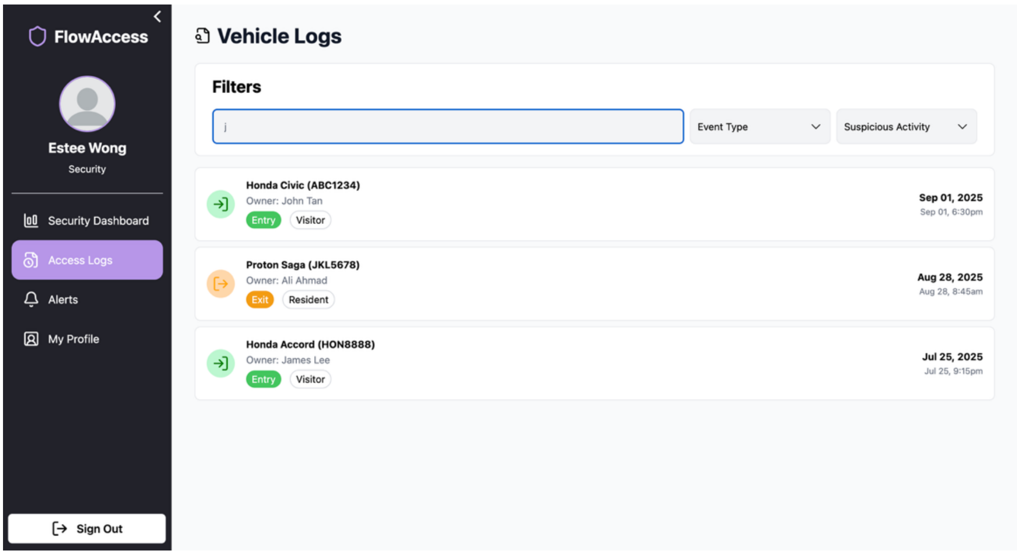
Ford Ranger (FRD1234)
Owner: Michael Chan
Exit Resident
Jun 10, 2025
Jun 10, 8:50am

Suspicious Tesla Model 3 (EV3333)
Unrecognized vehicle detected
Entry Visitor
May 28, 2025
May 28, 11:00am

Audi A6 (AUD1111)
Owner: Daniel Tan
Exit Resident
May 15, 2025
May 15, 9:20am

Toyota Camry (ZZZ7899)
Owner: Yuan Zhen
Entry Resident
Sep 02, 2025
Sep 02, 7:17am

Figure 5.55: Security Guard Vehicle Logs Page UI



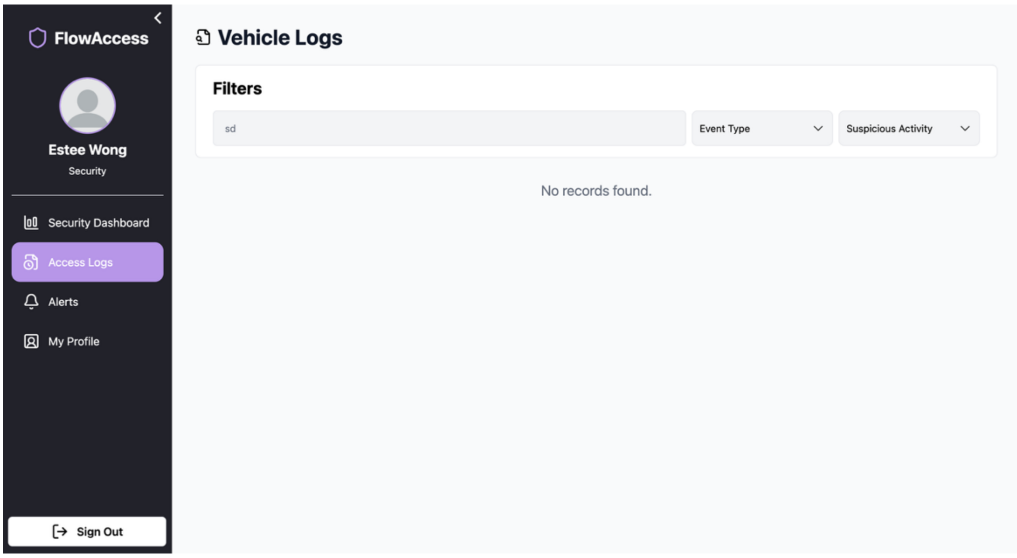
j

Honda Civic (ABC1234)
Owner: John Tan
Entry Visitor
Sep 01, 2025
Sep 01, 6:30pm

Proton Saga (JKL5678)
Owner: Ali Ahmad
Exit Resident
Aug 28, 2025
Aug 28, 8:45am

Honda Accord (HON8888)
Owner: James Lee
Entry Visitor
Jul 25, 2025
Jul 25, 9:15pm

Figure 5.56: Security Guard Vehicle Logs Apply Search and Filter Page UI

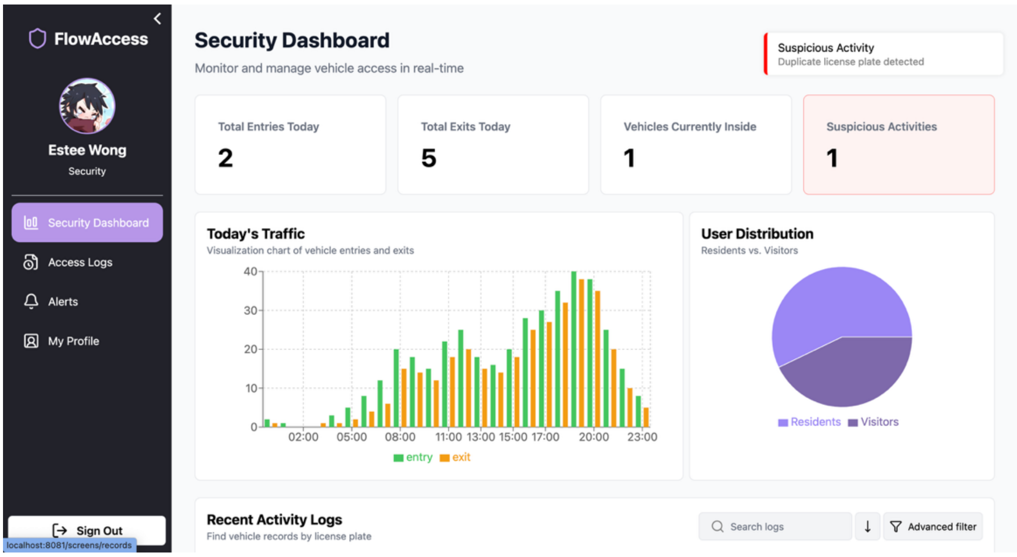


sd

No records found.

Figure 5.57: Security Guard Vehicle Logs No Searching Result Page UI

5.4.2.5 Suspicious Events



Security Dashboard

Monitor and manage vehicle access in real-time

Suspicious Activity
Duplicate license plate detected

Total Entries Today	Total Exits Today	Vehicles Currently Inside	Suspicious Activities
2	5	1	1

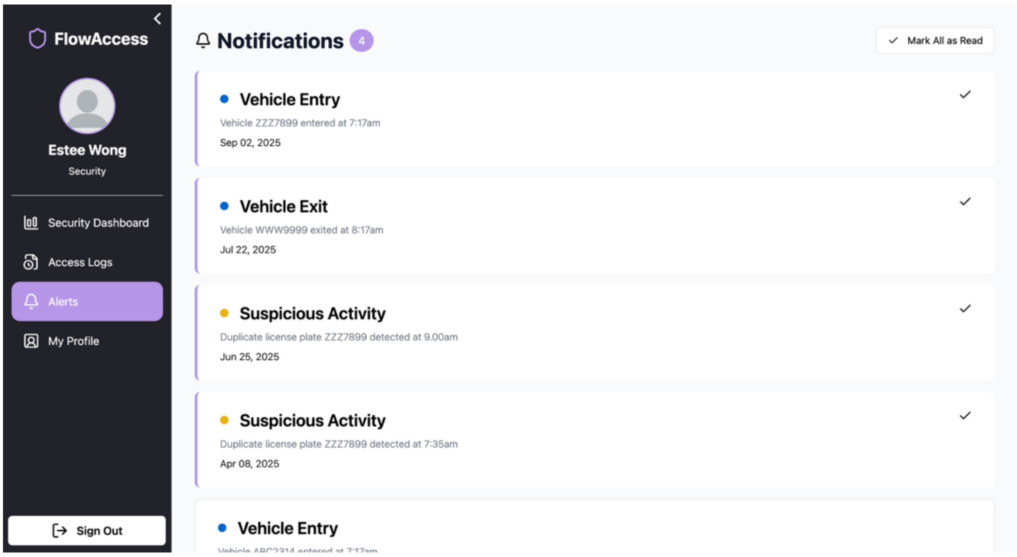
Today's Traffic
Visualization chart of vehicle entries and exits

User Distribution
Residents vs. Visitors

Recent Activity Logs
Find vehicle records by license plate

Figure 5.58: Security Guard Receive Suspicious Alert Page UI

5.4.2.6 Real-Time Alerts



Notifications 4

● **Vehicle Entry**
Vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025 ✓

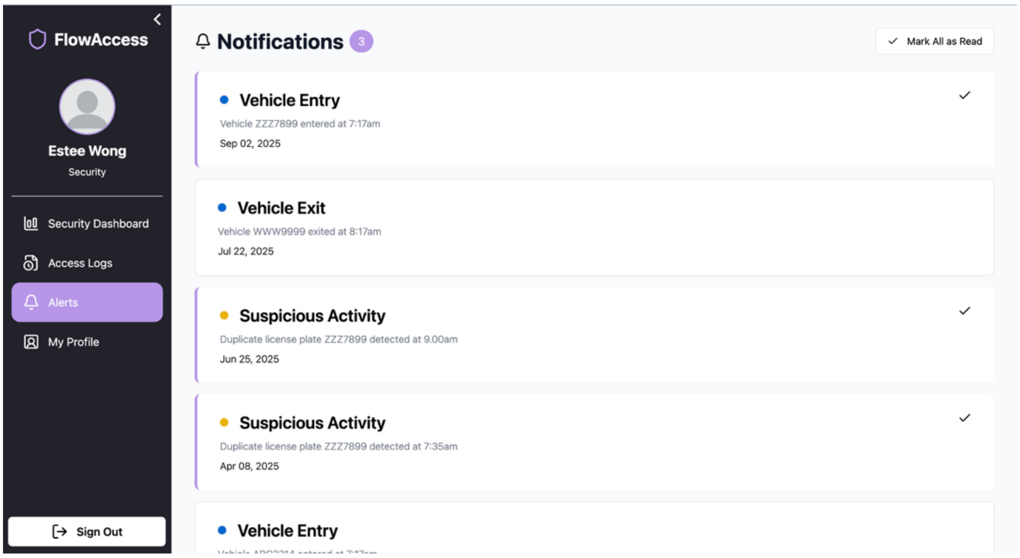
● **Vehicle Exit**
Vehicle WWW9999 exited at 8:17am
Jul 22, 2025 ✓

● **Suspicious Activity**
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025 ✓

● **Suspicious Activity**
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025 ✓

● **Vehicle Entry**
Vehicle ABC2314 entered at 7:17am

Figure 5.59: Security Guard Notification Page UI



Vehicle Entry ✓
Vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025

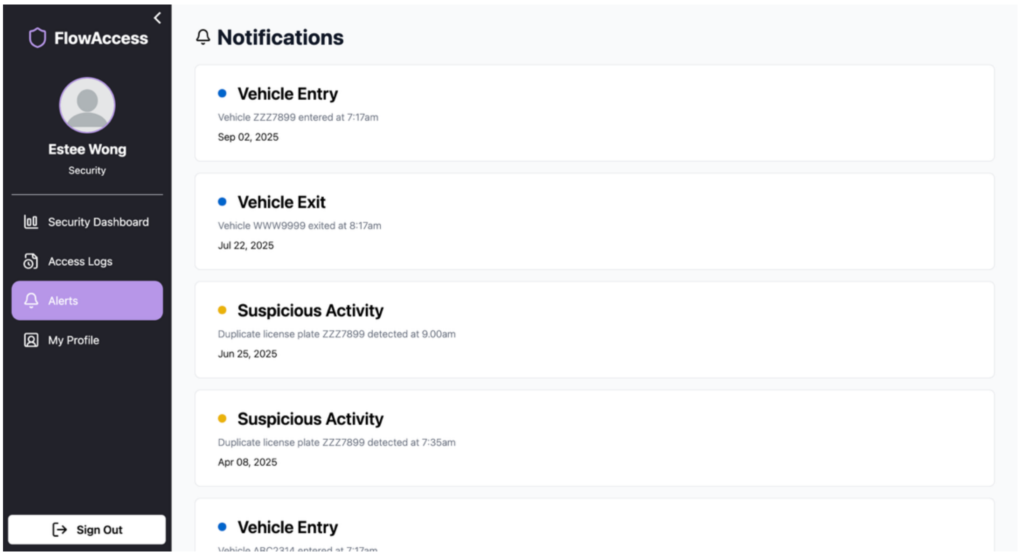
Vehicle Exit
Vehicle WWW9999 exited at 8:17am
Jul 22, 2025

Suspicious Activity ✓
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025

Suspicious Activity ✓
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025

Vehicle Entry
Vehicle ABC2314 entered at 7:17am

Figure 5.60: Security Guard Mark 1 Notificaiton as Read Page UI



Vehicle Entry
Vehicle ZZZ7899 entered at 7:17am
Sep 02, 2025

Vehicle Exit
Vehicle WWW9999 exited at 8:17am
Jul 22, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 9.00am
Jun 25, 2025

Suspicious Activity
Duplicate license plate ZZZ7899 detected at 7:35am
Apr 08, 2025

Vehicle Entry
Vehicle ABC2314 entered at 7:17am

Figure 5.61: Security Guard Mark All Notificaiton as Read Page UI

5.4.2.7 Manage Profile

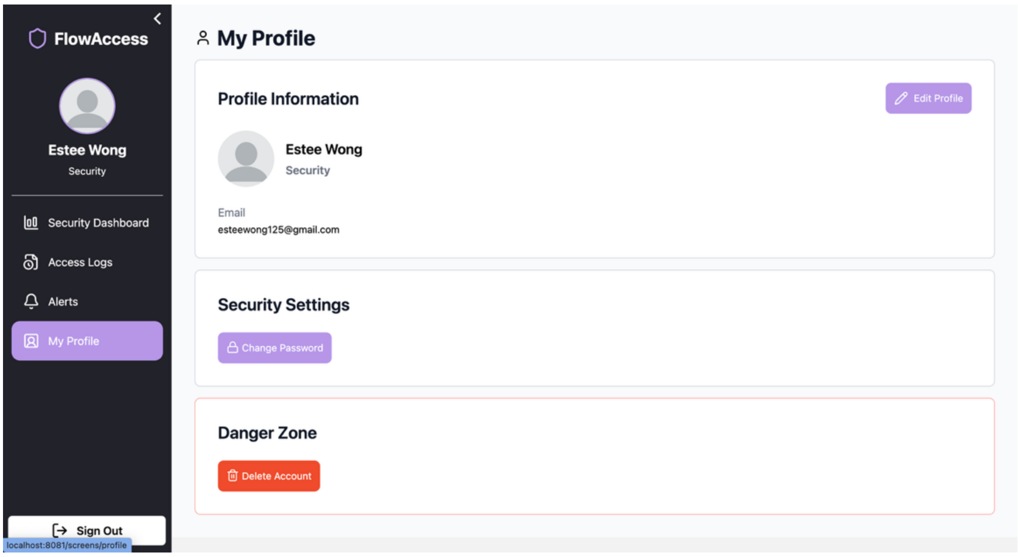


Figure 5.62: Security Guard Profile Page UI

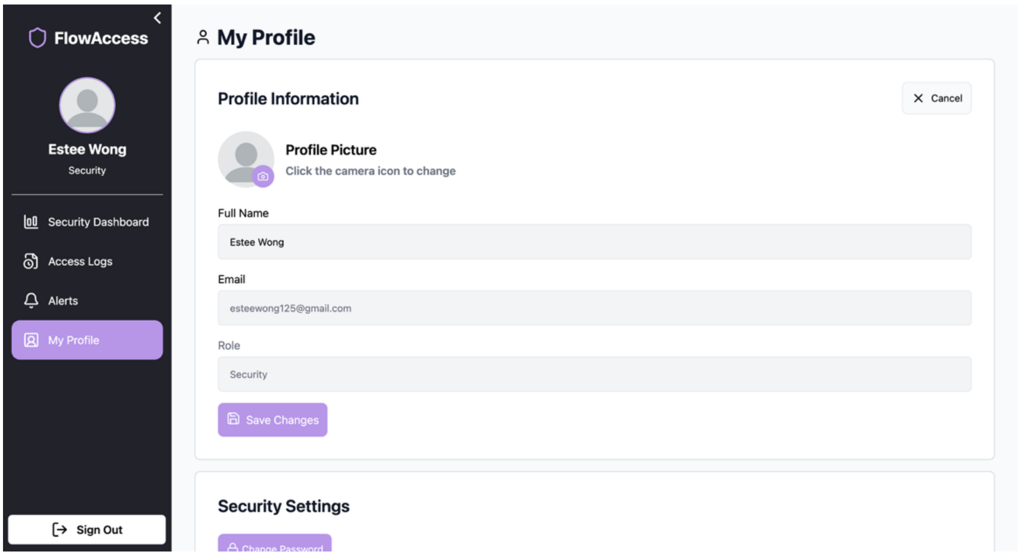
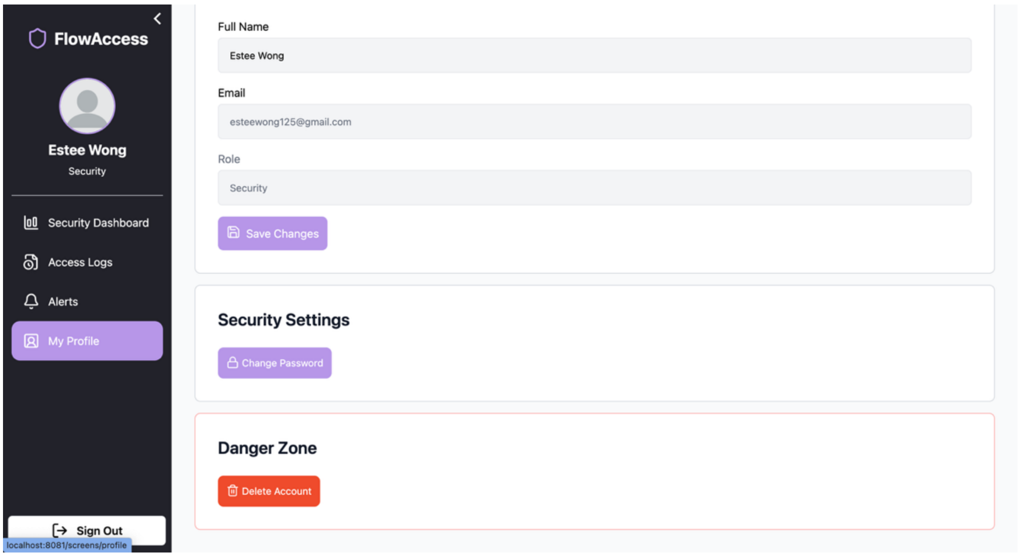


Figure 5.63: Security Guard Update Profile Page UI Part 1



Full Name
Estee Wong

Email
esteewong125@gmail.com

Role
Security

Security Settings

Danger Zone

Figure 5.64: Security Guard Update Profile Page UI Part 2

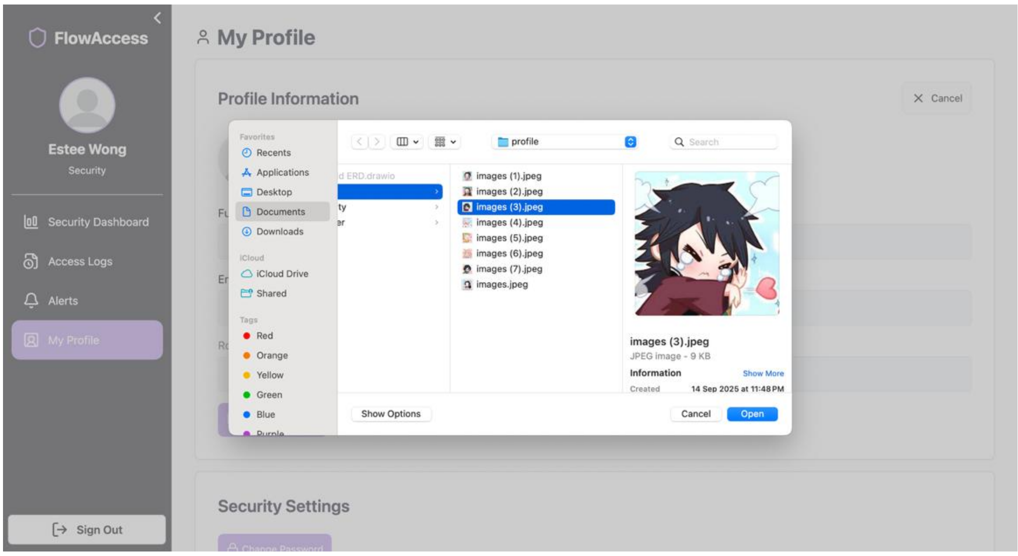
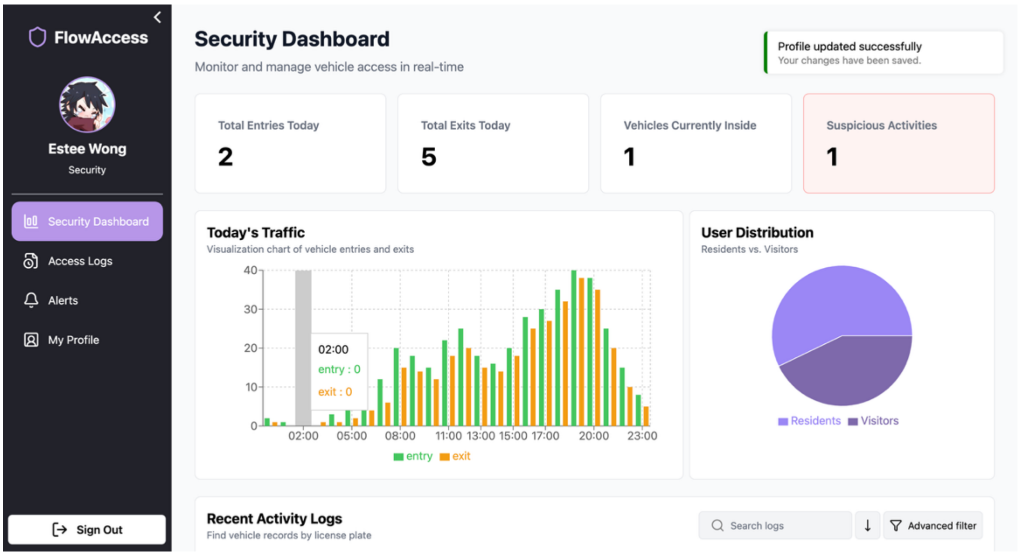


Figure 5.65: Security Guard Update Profile Image Page UI



FlowAccess

Estee Wong

Security

Security Dashboard

Monitor and manage vehicle access in real-time

Profile updated successfully
Your changes have been saved.

Total Entries Today 2

Total Exits Today 5

Vehicles Currently Inside 1

Suspicious Activities 1

Today's Traffic

Visualization chart of vehicle entries and exits

User Distribution

Residents vs. Visitors

Recent Activity Logs

Find vehicle records by license plate

Figure 5.66: Security Guard Update Profile Successful Page UI

CHAPTER 6

SYSTEM DESIGN

6.1 Introduction

This chapter provides a complete overview of the entire system's implementation, including the variety of modules created to meet the unique use cases and functional requirements described in the previous chapter. To ensure an accurate understanding of the system's design and capabilities, each module is carefully studied with a focus on its own set of features and functionalities.

Supabase serves as the system's backend platform in this project. It offers a serverless and scalable infrastructure that includes authentication, database administration, and real-time APIs. Supabase's PostgreSQL database allows for efficient storing and retrieval of processed data, while the built-in authentication service ensures safe access control. In other words, Supabase provides a modern, cloud-native architecture that simplifies data management and allows for seamless interaction between the client application and the backend.

In addition to backend services, YOLOv8 which is a state-of-the-art object detection model is also used to handle vehicle recognition tasks. The workflow begins with data preprocessing, where the raw dataset consisting of vehicle images undergoes cleaning, annotation, and normalization. This step ensures that the input data is consistent and optimized for model training. The preprocessed dataset is then used to train the YOLOv8 model, where vehicle attribute such as color, model, year and manufacturer are fine-tuned to achieve multimodal recognition of vehicle rather than only rely on license plate.

6.2 System Module

Table 6.1 presented below provides a comprehensive list of the modules developed in this project. As a result, to gain a better knowledge of the modules integrated into each system, all modules are categorized according to their intended users, as shown below.

Table 6. 1 : System Module

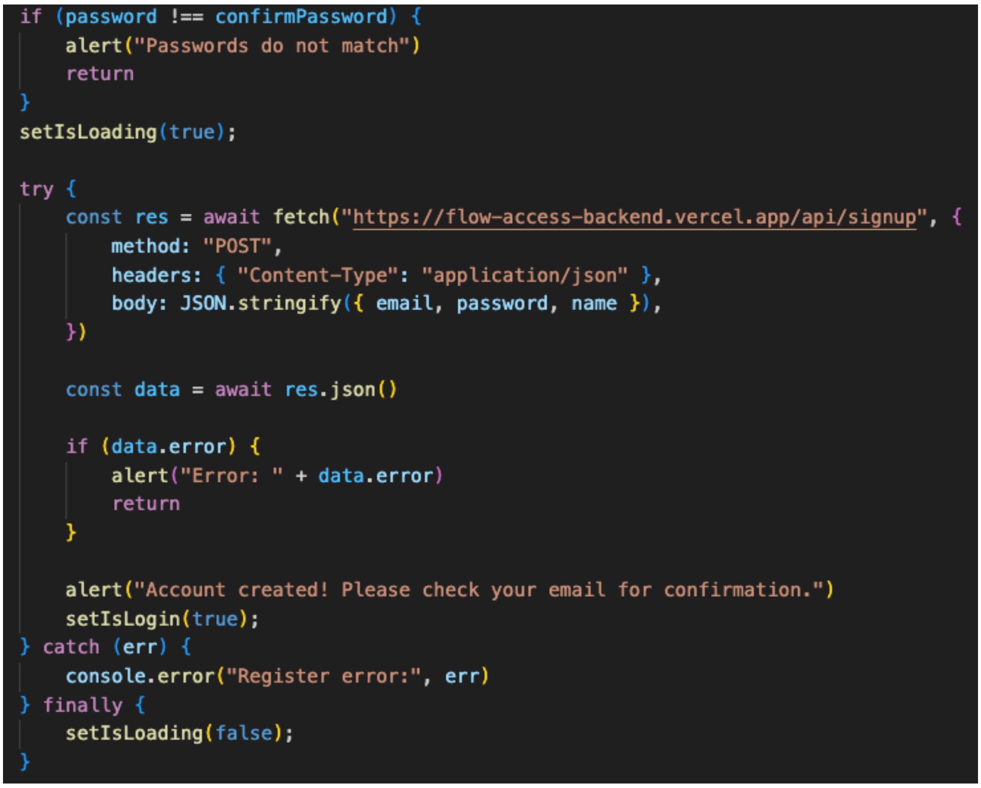
Target User	Module
Residents	Registration
	Login
	Manage Profile
	Manage Vehicle
	Notifications
	Suspicious Activity Alerts
	Manage Visitor Pass
	Reset Password
	Vehicle Logs
Security Guards	Login
	Reset Password
	Dashboard
	Vehicle Logs
	Suspicious Event
	Real-Time Alerts

6.2.1 Resident

The Resident user in the implemented system includes a number of features targeted at improving the user experience and facilitating smooth interaction across the platform. It is primarily intended to meet the demands of those who interact with the system in the position as the residents.

6.2.1.1 Registration

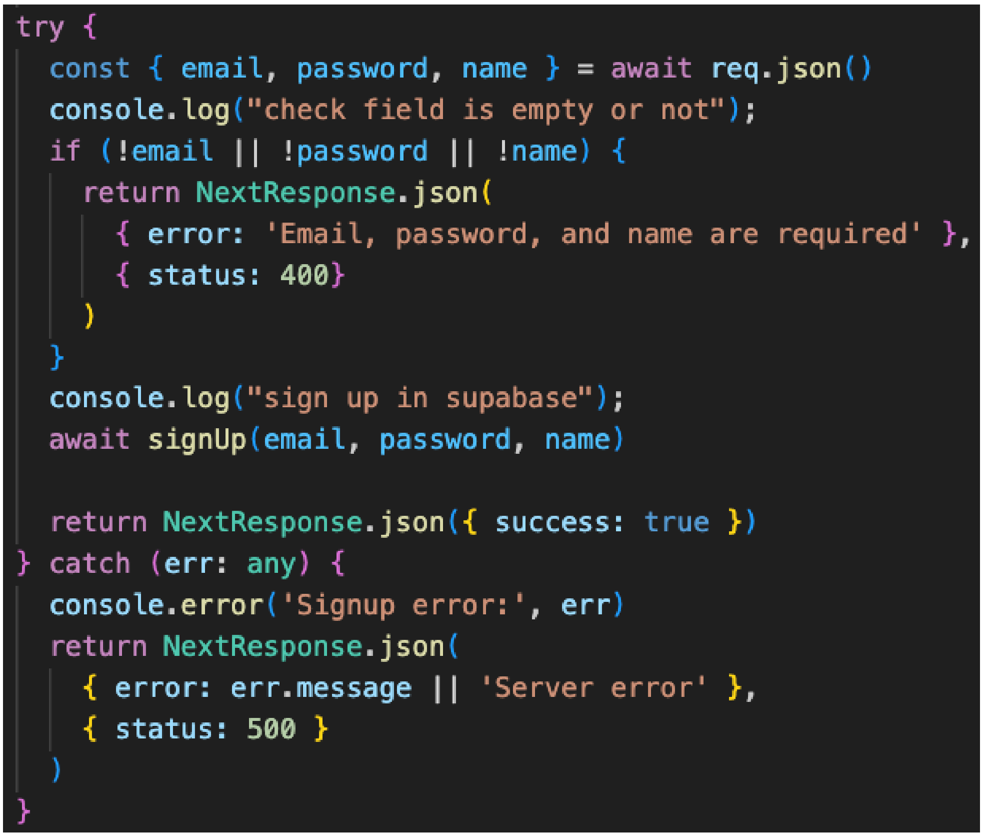
The implemented system provides a secure and easy registration method for residents powered by Supabase authentication which is a modern, cloud-native solution that interfaces directly with PostgreSQL. This solution ensures that user data is handled consistently throughout numerous system modules, in addition to authentication.



```
if (password !== confirmPassword) {
    alert("Passwords do not match")
    return
}
setIsLoading(true);

try {
    const res = await fetch("https://flow-access-backend.vercel.app/api/signup", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email, password, name }),
    })

    const data = await res.json()

    if (data.error) {
        alert("Error: " + data.error)
        return
    }

    alert("Account created! Please check your email for confirmation.")
    setIsLogin(true);
} catch (err) {
    console.error("Register error:", err)
} finally {
    setIsLoading(false);
}
```

Figure 6.1: Registration Frontend Code

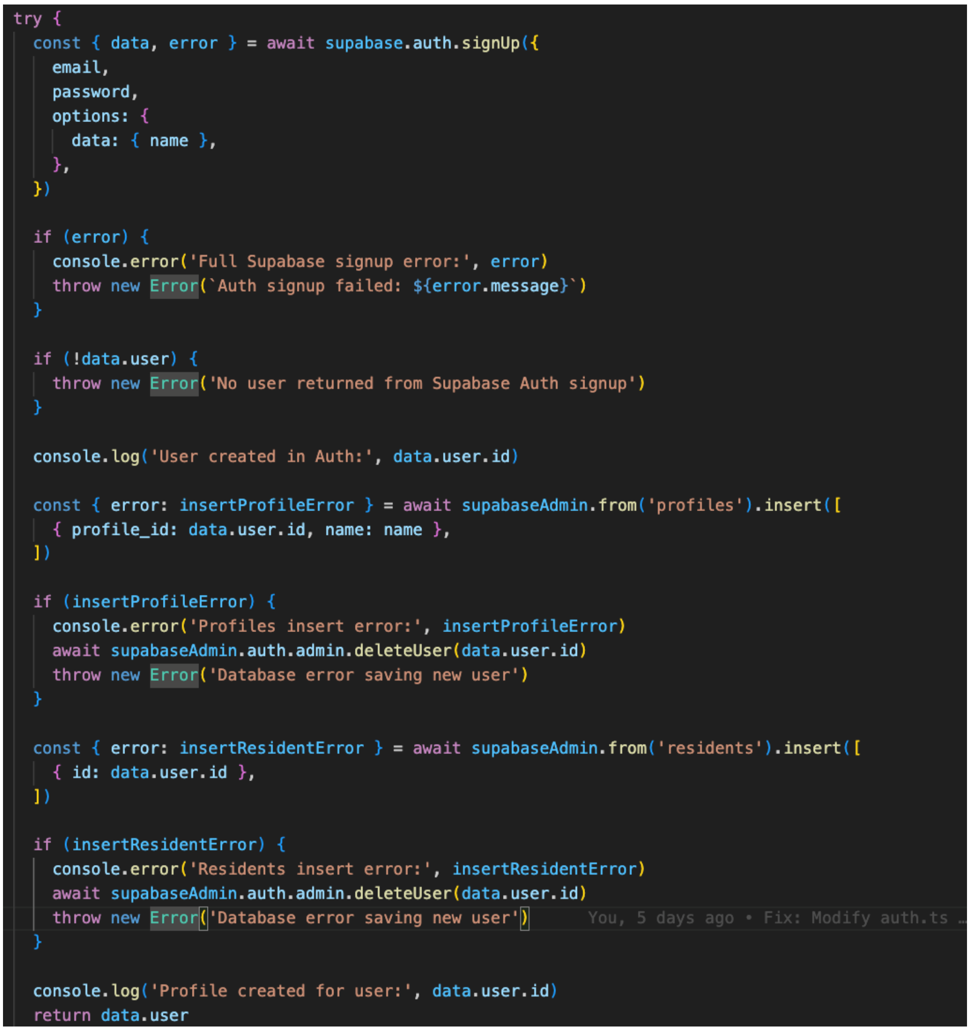
In the code, the register function will firstly ensure the password and confirm password is match before sending the data to backend. While waiting for backend's response, the loading sign which is the "Creating account" words will appear to provide system feedback for user in order to let them know the data is being processed rather than the button is not clicked. Once the account is created successfully, an alert will be prompt to the user for them to click on the email sent to them for activating their account. After that, the system will switch to login part which give convenience to user.



```
try {
  const { email, password, name } = await req.json()
  console.log("check field is empty or not");
  if (!email || !password || !name) {
    return NextResponse.json(
      { error: 'Email, password, and name are required' },
      { status: 400}
    )
  }
  console.log("sign up in supabase");
  await signUp(email, password, name)

  return NextResponse.json({ success: true })
} catch (err: any) {
  console.error('Signup error:', err)
  return NextResponse.json(
    { error: err.message || 'Server error' },
    { status: 500 }
  )
}
```

Figure 6.2: Registration Backend Code Part 1

During registration, the system validates required fields such as email, password, and name before securely creating the user account using Supabase Auth. Passwords are encrypted and email confirmation is required to prevent illegal access. To improve functionality, user information is saved in the profiles database, while a corresponding record in the residents table determines the resident's role in the system.



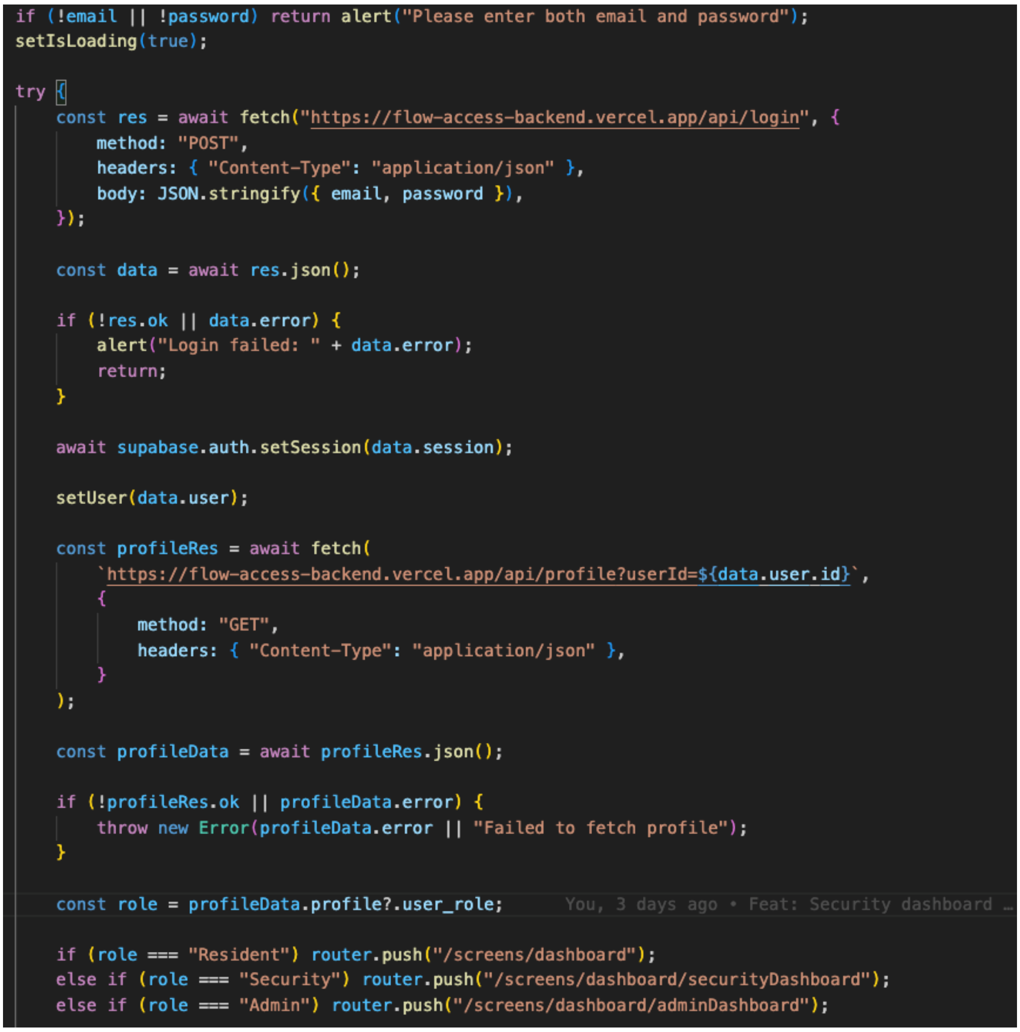
```
try {
  const { data, error } = await supabase.auth.signUp({
    email,
    password,
    options: {
      data: { name },
    },
  })

  if (error) {
    console.error('Full Supabase signup error:', error)
    throw new Error(`Auth signup failed: ${error.message}`)
  }

  if (!data.user) {
    throw new Error('No user returned from Supabase Auth signup')
  }

  console.log('User created in Auth:', data.user.id)

  const { error: insertProfileError } = await supabaseAdmin.from('profiles').insert([
    { profile_id: data.user.id, name: name },
  ])

  if (insertProfileError) {
    console.error('Profiles insert error:', insertProfileError)
    await supabaseAdmin.auth.admin.deleteUser(data.user.id)
    throw new Error('Database error saving new user')
  }

  const { error: insertResidentError } = await supabaseAdmin.from('residents').insert([
    { id: data.user.id },
  ])

  if (insertResidentError) {
    console.error('Residents insert error:', insertResidentError)
    await supabaseAdmin.auth.admin.deleteUser(data.user.id)
    throw new Error('Database error saving new user')      You, 5 days ago • Fix: Modify auth.ts …
  }

  console.log('Profile created for user:', data.user.id)
  return data.user
```

Figure 6.3: Registration Backend Code Part 2

To guarantee data integrity, robust rollback procedures are also provided, ensuring that incomplete or failed insertions do not result in invalid records in the database. By combining Supabase authentication with structured data integration, the solution creates an efficient and secure resident-focused access process that maintains usability and system stability.

6.2.1.2 Login

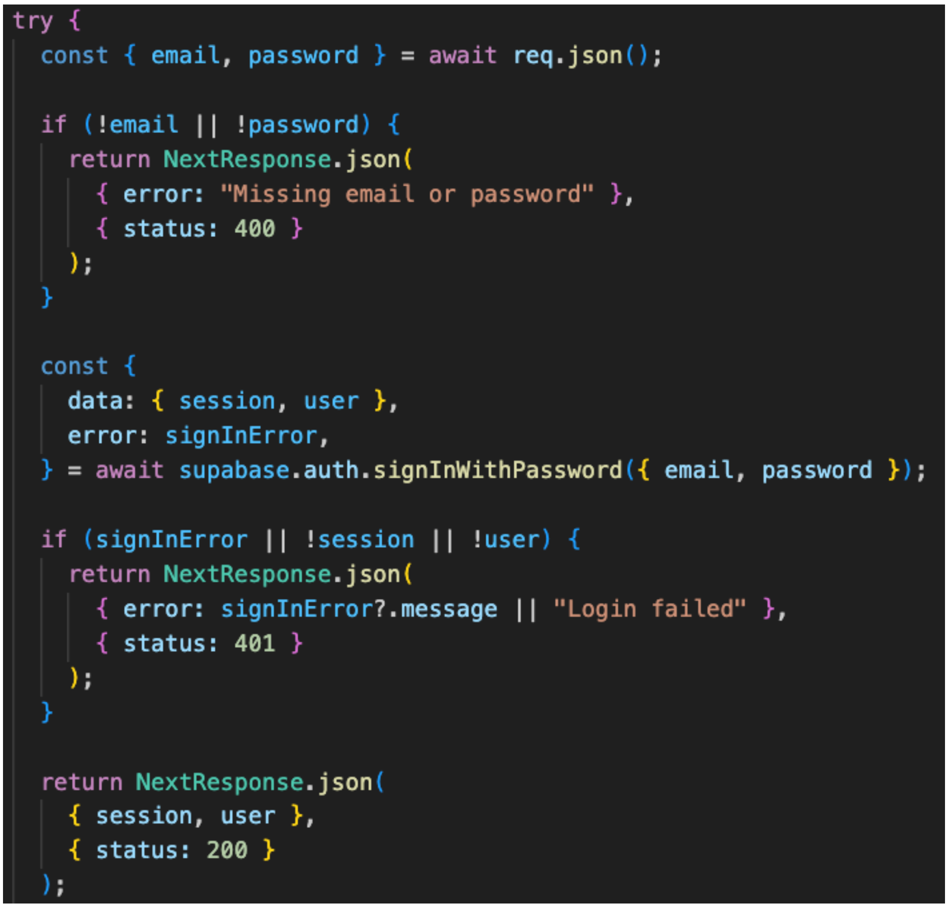
For login module, a secure mechanism was developed to authenticate residents and security guard and then grant them access to role-specific functionalities. The process is powered by Supabase Authentication, which provides robust and scalable user session management while ensuring data security.



```
if (!email || !password) return alert("Please enter both email and password");
setIsLoading(true);

try {
    const res = await fetch("https://flow-access-backend.vercel.app/api/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email, password }),
    });

    const data = await res.json();

    if (!res.ok || data.error) {
        alert("Login failed: " + data.error);
        return;
    }

    await supabase.auth.setSession(data.session);

    setUser(data.user);

    const profileRes = await fetch(
        `https://flow-access-backend.vercel.app/api/profile?userId=${data.user.id}`,
        {
            method: "GET",
            headers: { "Content-Type": "application/json" },
        }
    );

    const profileData = await profileRes.json();

    if (!profileRes.ok || profileData.error) {
        throw new Error(profileData.error || "Failed to fetch profile");
    }

    const role = profileData.profile?.user_role;      You, 3 days ago • Feat: Security dashboard …

    if (role === "Resident") router.push("/screens/dashboard");
    else if (role === "Security") router.push("/screens/dashboard/securityDashboard");
    else if (role === "Admin") router.push("/screens/dashboard/adminDashboard");
```

Figure 6.4: Login Frontend Code

When user wants to sign in, the system first ensures that both the email address and the password are entered before submitting the request. After successful login, the user will be route to different page based on the user role in the profile table.

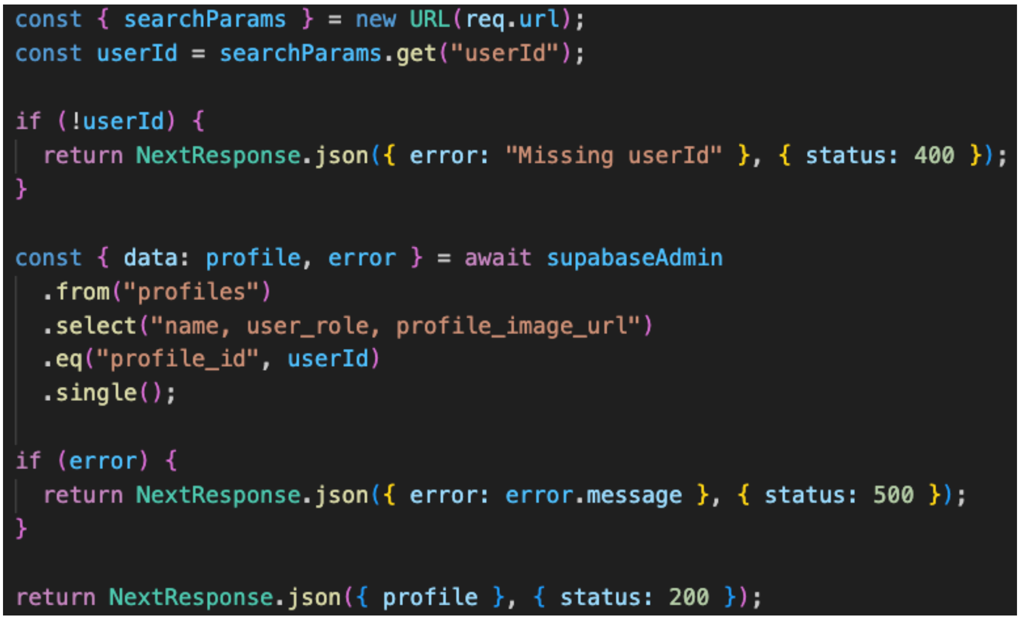


```
try {
  const { email, password } = await req.json();

  if (!email || !password) {
    return NextResponse.json(
      { error: "Missing email or password" },
      { status: 400 }
    );
  }

  const {
    data: { session, user },
    error: signInError,
  } = await supabase.auth.signInWithPassword({ email, password });

  if (signInError || !session || !user) {
    return NextResponse.json(
      { error: signInError?.message || "Login failed" },
      { status: 401 }
    );
  }

  return NextResponse.json(
    { session, user },
    { status: 200 }
  );
```

Figure 6.5: Login Backend Code Part 1

The credentials are then handled by Supabase Auth which checks the user's information. A secure session token will be generated after successful authentication by user. This token is saved locally to keep the user authenticated during their interactions with the system.

In the event of invalid credentials or server-side failures, the system is configured to return clear error messages while blocking unauthorized access. By integrating Supabase's secure session handling with structured profile validation, the login mechanism provides an efficient, role-based authentication procedure that improves system security and user experience.



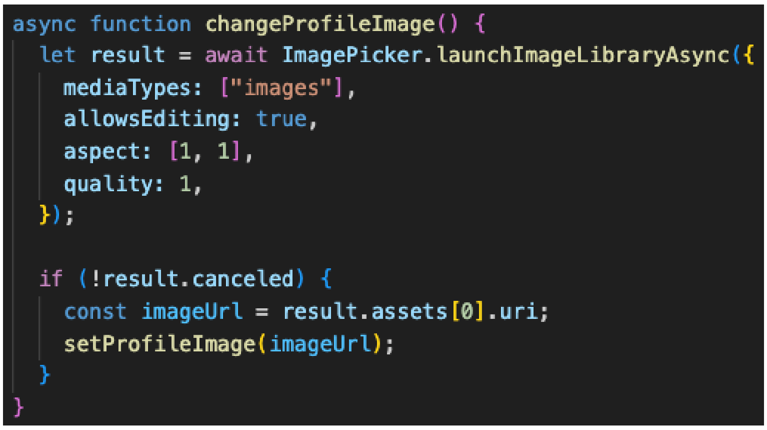
```
const { searchParams } = new URL(req.url);
const userId = searchParams.get("userId");

if (!userId) {
  return NextResponse.json({ error: "Missing userId" }, { status: 400 });
}

const { data: profile, error } = await supabaseAdmin
  .from("profiles")
  .select("name, user_role, profile_image_url")
  .eq("profile_id", userId)
  .single();

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

return NextResponse.json({ profile }, { status: 200 });
```

Figure 6.6: Login Backend Code Part 2

After having the secure session, the profile information to be shown is then read by using searchParam function to pass the unique user id to the database. The profile information is then set to the AuthContext to ease the pass of profile data among each page to get relevant information from database.

6.2.1.3 Manage Profile

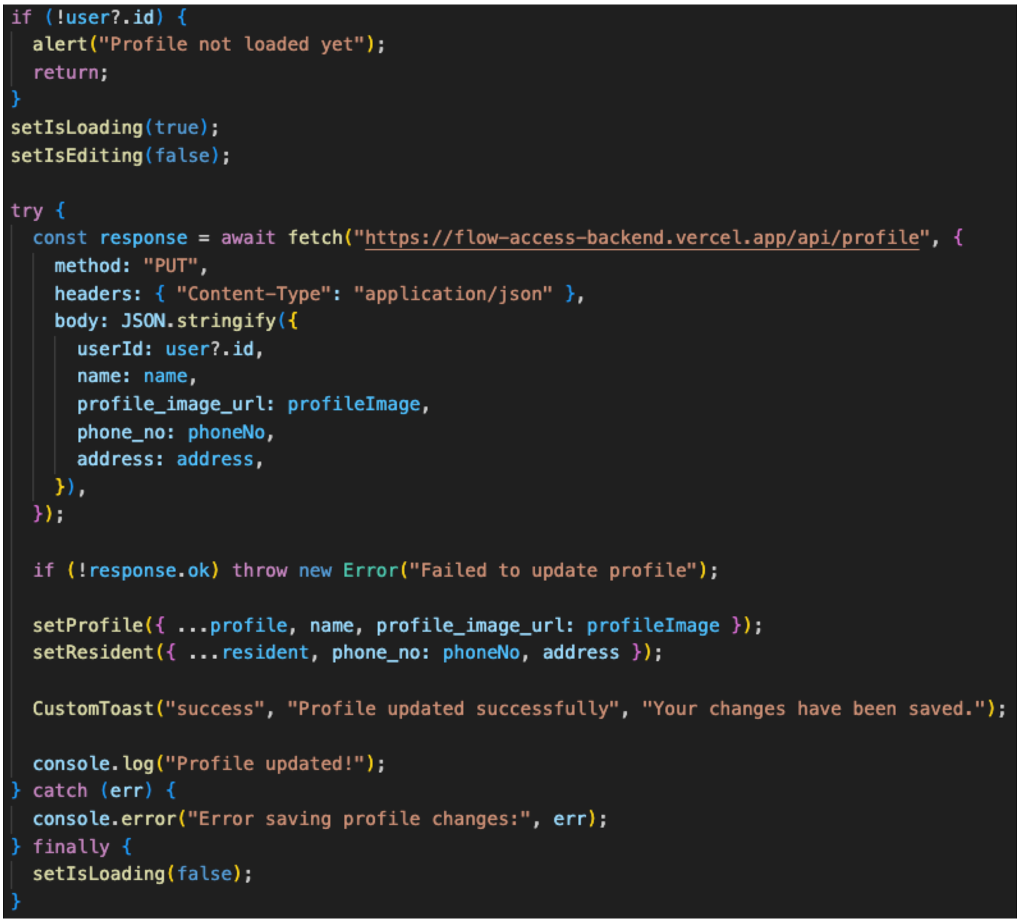
This module optimizes the user experience by allowing updates to details like name, phone number, address, and profile image, while guaranteeing that all changes are securely kept in the system's database via Supabase integration.



```
async function changeProfileImage() {
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ["images"],
    allowsEditing: true,
    aspect: [1, 1],
    quality: 1,
  });

  if (!result.canceled) {
    const imageUrl = result.assets[0].uri;
    setProfileImage(imageUrl);
  }
}
```

Figure 6.7: Change Profile Image Frontend Code

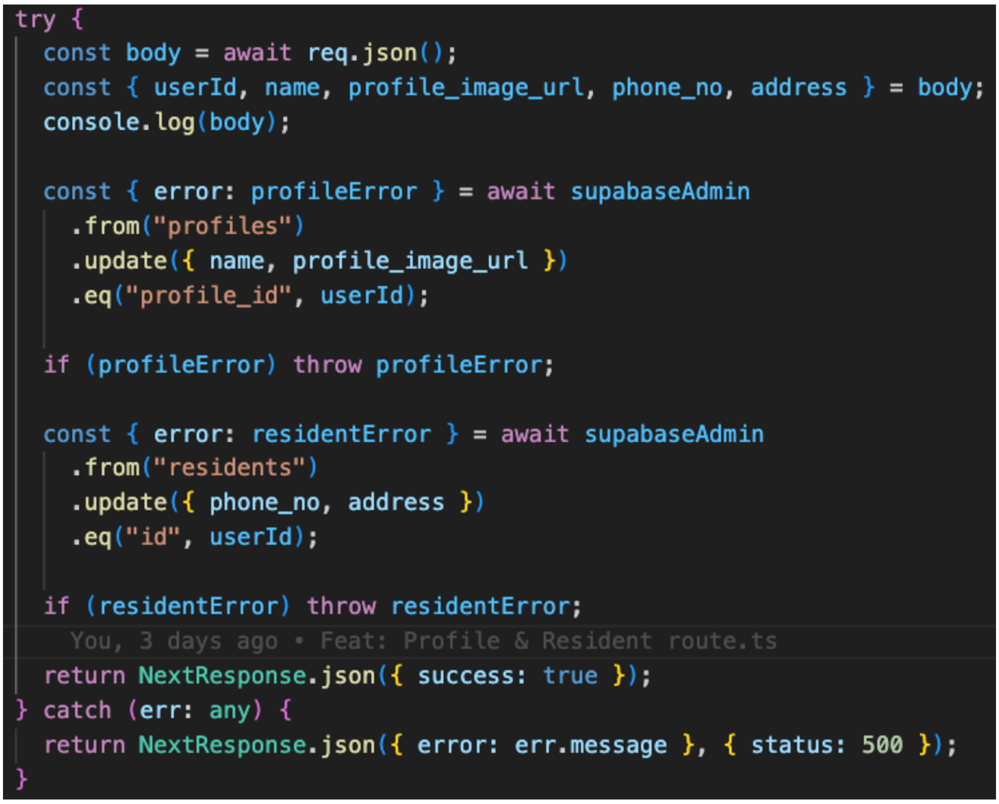
When user want to update their profile image, they starts the procedure by selecting a new profile image from their device's image picker capabilities after clicking the edit button in profile information. When an image is selected, the system briefly adjusts the interface to reflect the new selection.



```
if (!user?.id) {
  alert("Profile not loaded yet");
  return;
}
setIsLoading(true);
setIsEditing(false);

try {
  const response = await fetch("https://flow-access-backend.vercel.app/api/profile", {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      userId: user?.id,
      name: name,
      profile_image_url: profileImage,
      phone_no: phoneNo,
      address: address,
    }),
  });

  if (!response.ok) throw new Error("Failed to update profile");

  setProfile({ ...profile, name, profile_image_url: profileImage });
  setResident({ ...resident, phone_no: phoneNo, address });

  CustomToast("success", "Profile updated successfully", "Your changes have been saved.");

  console.log("Profile updated!");
} catch (err) {
  console.error("Error saving profile changes:", err);
} finally {
  setIsLoading(false);
}
```

Figure 6.8: Upload New Profile Image Frontend Code

When a resident approves and saves their profile updates, the system ensures that a valid user session is active before proceeding with the update request. The new data, which includes the profile image URL, contact number, and address, is subsequently transferred to the backend using a secure API endpoint.



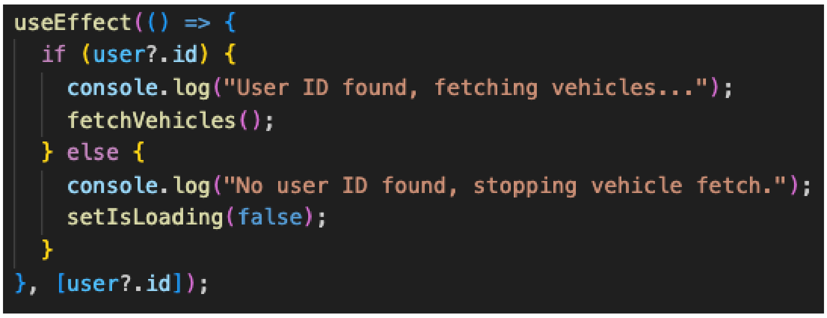
```
try {
  const body = await req.json();
  const { userId, name, profile_image_url, phone_no, address } = body;
  console.log(body);

  const { error: profileError } = await supabaseAdmin
    .from("profiles")
    .update({ name, profile_image_url })
    .eq("profile_id", userId);

  if (profileError) throw profileError;

  const { error: residentError } = await supabaseAdmin
    .from("residents")
    .update({ phone_no, address })
    .eq("id", userId);

  if (residentError) throw residentError;
    You, 3 days ago • Feat: Profile & Resident route.ts
  return NextResponse.json({ success: true });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.9: Update Profile Information Backend Code

On the backend, the system handles the update request by modifying two essential tables which include the profiles table that contains general information such as the resident's name and profile image. The other one is the residents table, which provides role-specific information such as phone number and address that is not suitable for security guard role. This dual update guarantees that all resident-related information is consistent and appropriately formatted throughout the database. In the event of an error during the update, the system gives the user clear feedback while preventing incomplete or incorrect data from being stored.

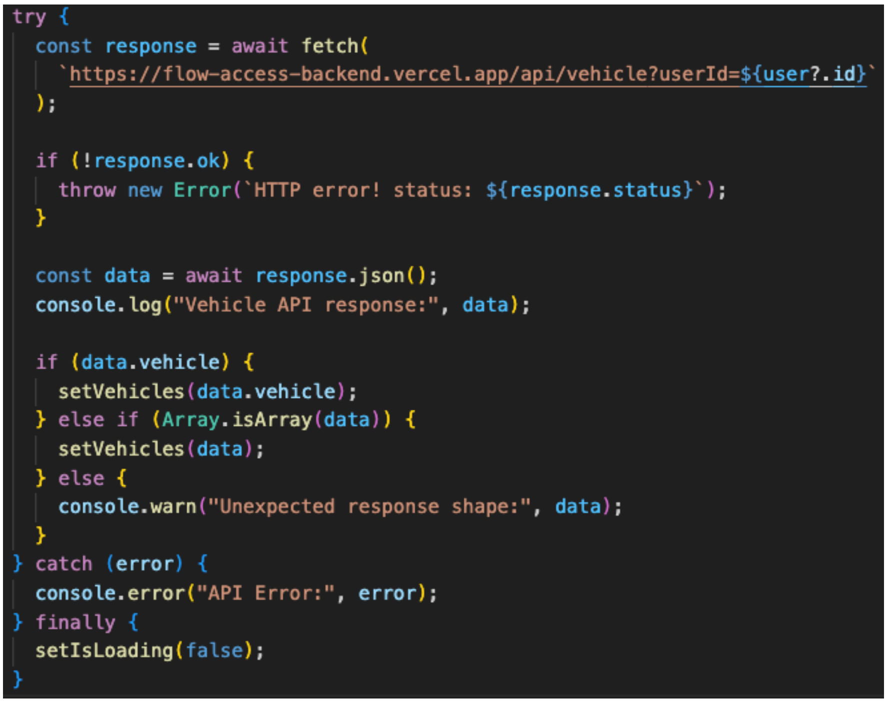
6.2.1.4 Manage Vehicle

This module basically is to allow residents to conveniently register, maintain, and manage their automobiles within the platform. This module guarantees that all vehicle-related data is regularly maintained and securely saved, while also providing residents with a user-friendly interface for managing their records.



```
useEffect(() => {
  if (user?.id) {
    console.log("User ID found, fetching vehicles...");
    fetchVehicles();
  } else {
    console.log("No user ID found, stopping vehicle fetch.");
    setIsLoading(false);
  }
}, [user?.id]);
```

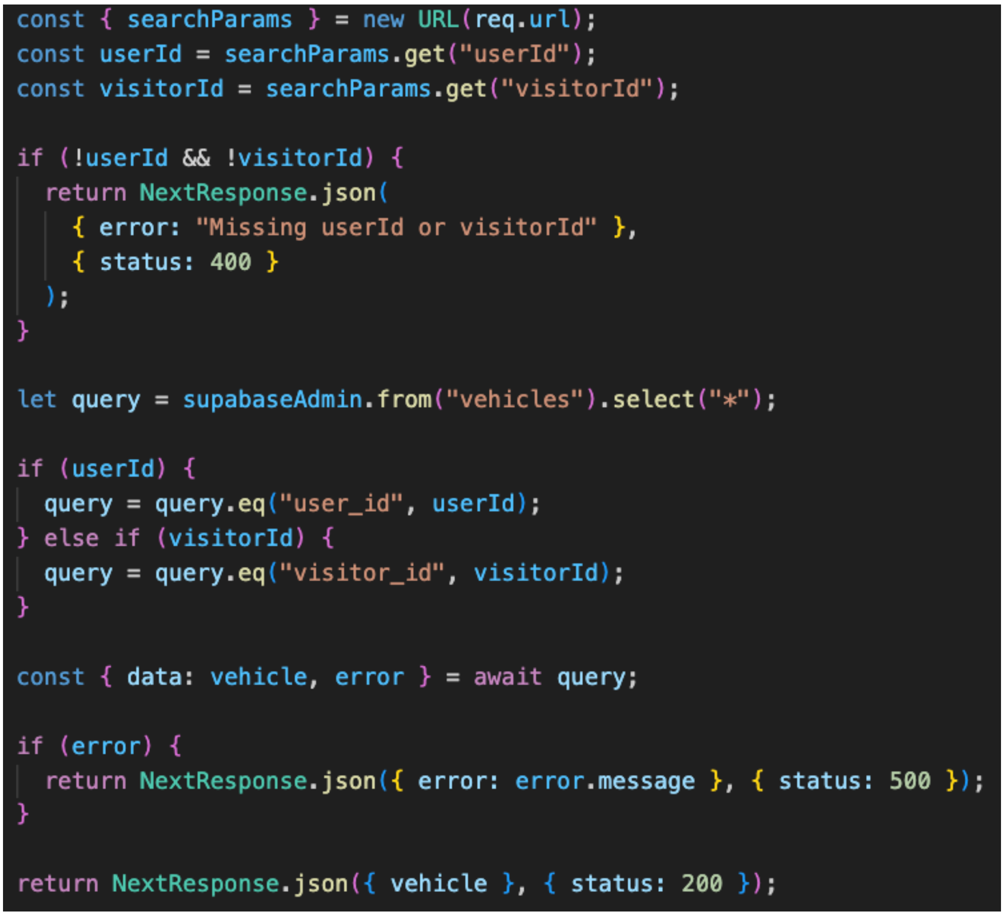
Figure 6.10: Fetch Vehicles on Vehicle Page Frontend Code



```
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/vehicle?userId=${user?.id}`
  );

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  const data = await response.json();
  console.log("Vehicle API response:", data);

  if (data.vehicle) {
    setVehicles(data.vehicle);
  } else if (Array.isArray(data)) {
    setVehicles(data);
  } else {
    console.warn("Unexpected response shape:", data);
  }
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
}
```

Figure 6.11: Fetch Vehicles Frontend Code

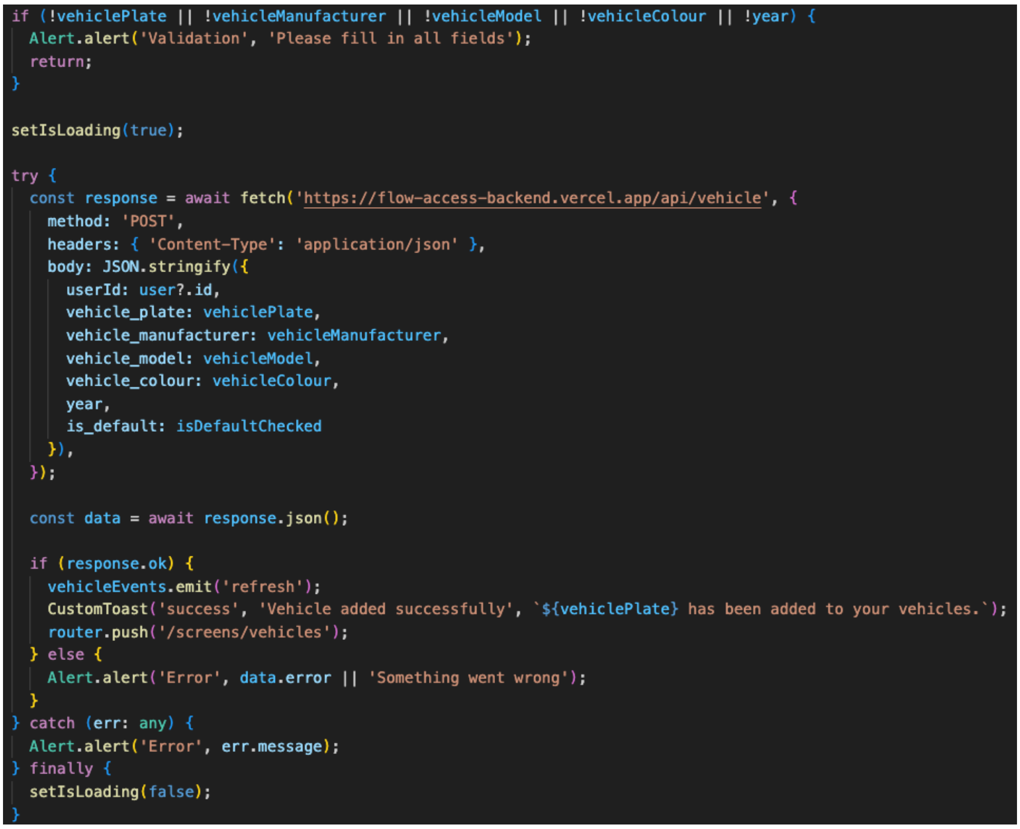
While entering the vehicle page, the system will show the loading page and check whether the user is login or not by checking its id to ensure valid access of vehicle data. Once confirmed, the system will then fetch the vehicle data from backend by using the parameter pass in the endpoint which is the user id. After getting all the vehicle data, they will be shown on the vehicle page and deactivate the loading page. If any error occur during the process, the loading page will also be deactivated.



```
const { searchParams } = new URL(req.url);
const userId = searchParams.get("userId");
const visitorId = searchParams.get("visitorId");

if (!userId && !visitorId) {
  return NextResponse.json(
    { error: "Missing userId or visitorId" },
    { status: 400 }
  );
}

let query = supabaseAdmin.from("vehicles").select("*");

if (userId) {
  query = query.eq("user_id", userId);
} else if (visitorId) {
  query = query.eq("visitor_id", visitorId);
}

const { data: vehicle, error } = await query;

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

return NextResponse.json({ vehicle }, { status: 200 });
```

Figure 6.12: Fetch Vehicles Backend Code

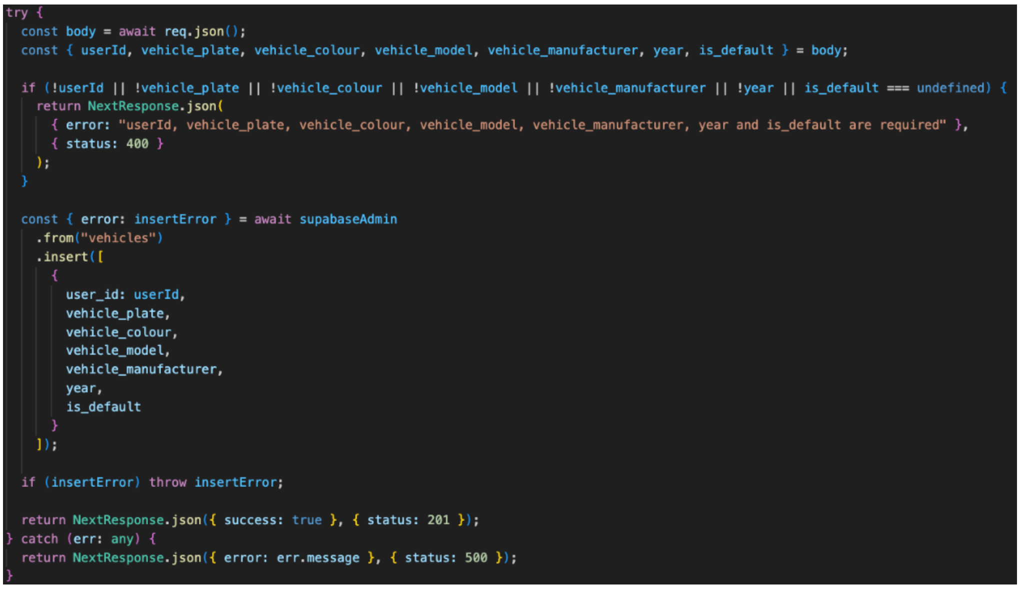
In the backend, it will try to get either the user id or the visitor id to filter out the car that is registered by the login user. It use the supabaseAdmin to access the vehicles table as the row level security of the table is enable. In this case, the supabaseAdmin use the service role key to access the table rather than using supabase that use anon key to access. This ensure restrict rules to get data from the database and increase the security level as the service role key is not exposed to the frontend but only use in backend. After that, the vehicle's are selected accordingly and return as json format with the status code.



```
if (!vehiclePlate || !vehicleManufacturer || !vehicleModel || !vehicleColour || !year) {
  Alert.alert('Validation', 'Please fill in all fields');
  return;
}

setIsLoading(true);

try {
  const response = await fetch('https://flow-access-backend.vercel.app/api/vehicle', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      userId: user?.id,
      vehicle_plate: vehiclePlate,
      vehicle_manufacturer: vehicleManufacturer,
      vehicle_model: vehicleModel,
      vehicle_colour: vehicleColour,
      year,
      is_default: isDefaultChecked
    }),
  });

  const data = await response.json();

  if (response.ok) {
    vehicleEvents.emit('refresh');
    CustomToast('success', 'Vehicle added successfully', `${vehiclePlate} has been added to your vehicles.`);
    router.push('/screens/vehicles');
  } else {
    Alert.alert('Error', data.error || 'Something went wrong');
  }
} catch (err: any) {
  Alert.alert('Error', err.message);
} finally {
  setIsLoading(false);
}
```

Figure 6.13: Add New Vehicle Frontend Code

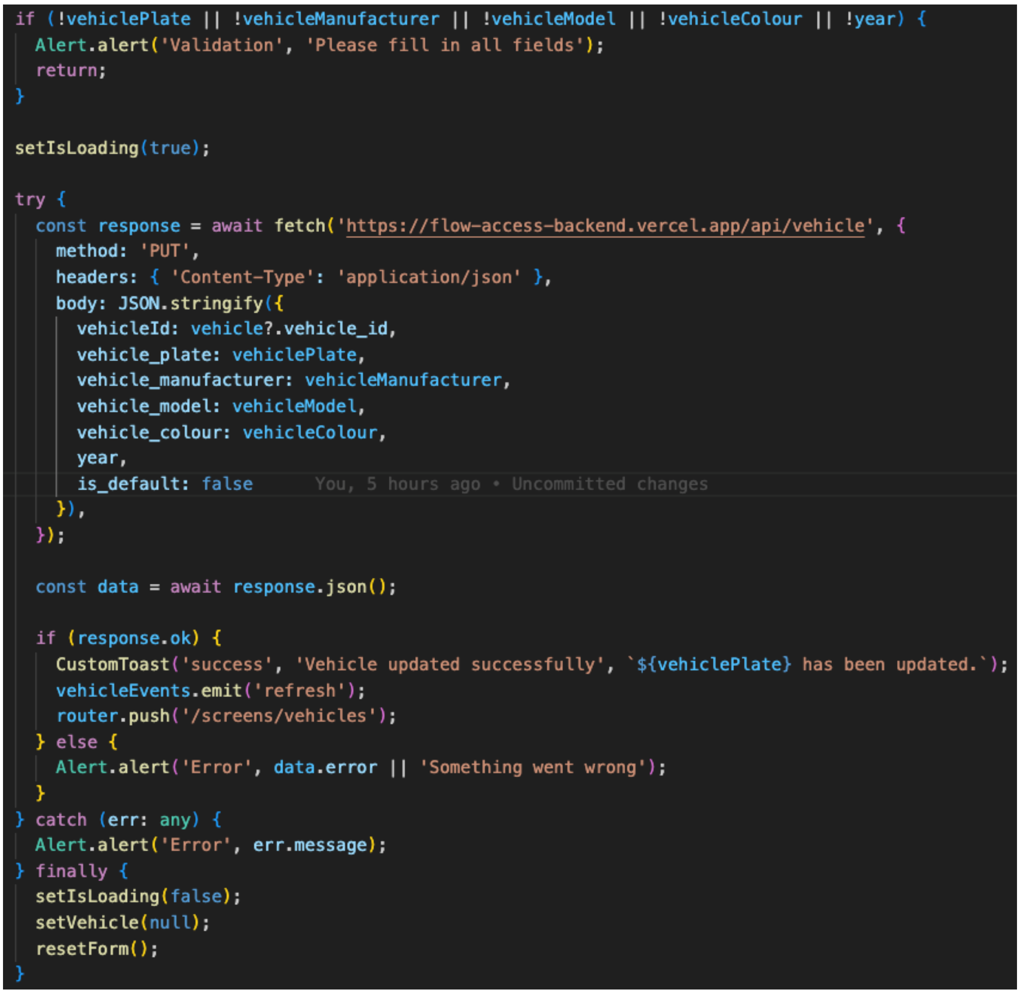
When a resident registers a new vehicle, the system receives information such the plate number, make, model, color and manufacturer year. The field are checking before calling the endpoint to add a new vehicle by using the user id as referencing for making other action on it. A json format data is send along to the endpoint to add new vehicle. Once getting a response from the endpoint. The vehicleEvent will emit a refresh signal to refresh the list of vehicle in the vehicle page to ensure the vehicle list is always up-to-date.



```
try {
  const body = await req.json();
  const { userId, vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year, is_default } = body;

  if (!userId || !vehicle_plate || !vehicle_colour || !vehicle_model || !vehicle_manufacturer || !year || is_default === undefined) {
    return NextResponse.json(
      { error: "userId, vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year and is_default are required" },
      { status: 400 }
    );
  }

  const { error: insertError } = await supabaseAdmin
    .from("vehicles")
    .insert([
      {
        user_id: userId,
        vehicle_plate,
        vehicle_colour,
        vehicle_model,
        vehicle_manufacturer,
        year,
        is_default
      }
    ]);

  if (insertError) throw insertError;

  return NextResponse.json({ success: true }, { status: 201 });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.14: Add New Vehicle Backend Code

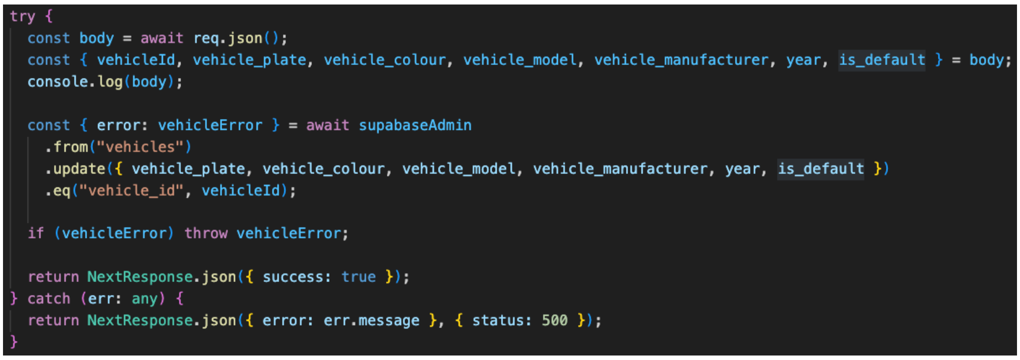
Similar to frontend, backend code also check whether the require field is passed in or not before adding the vehicle. If missing require body, it will directly return the error message with status code as response data. On the other hands, if all the require data is exist, it will pass it to database and return a json response which include the success message and status code 201 which represent created.



```
if (!vehiclePlate || !vehicleManufacturer || !vehicleModel || !vehicleColour || !year) {
  Alert.alert('Validation', 'Please fill in all fields');
  return;
}

setIsLoading(true);

try {
  const response = await fetch('https://flow-access-backend.vercel.app/api/vehicle', {
    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      vehicleId: vehicle?.vehicle_id,
      vehicle_plate: vehiclePlate,
      vehicle_manufacturer: vehicleManufacturer,
      vehicle_model: vehicleModel,
      vehicle_colour: vehicleColour,
      year,
      is_default: false      You, 5 hours ago • Uncommitted changes
    }),
  });

  const data = await response.json();

  if (response.ok) {
    CustomToast('success', 'Vehicle updated successfully', `${vehiclePlate} has been updated.`);
    vehicleEvents.emit('refresh');
    router.push('/screens/vehicles');
  } else {
    Alert.alert('Error', data.error || 'Something went wrong');
  }
} catch (err: any) {
  Alert.alert('Error', err.message);
} finally {
  setIsLoading(false);
  setVehicle(null);
  resetForm();
}
```

Figure 6.15: Update Vehicle Frontend Code

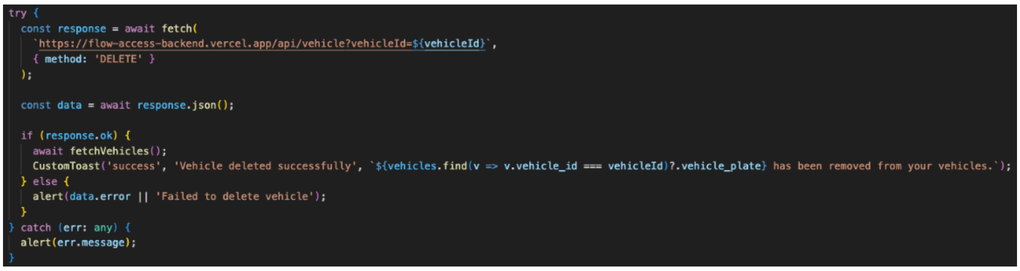
This function updates a vehicle's information in a backend database. It first validates that all required fields which are plate number, manufacturer, model, colour, and year all are filled, showing an alert if any are missing. Once validated, it sets a loading state and sends a PUT request to the backend API with the updated vehicle details that including the vehicle ID and a flag for `is_default`. After receiving a response, it checks if the update was successful: if so, it shows a custom toast notification and the emits a refresh event to update the other components, and navigates the user back to the vehicles screen; if not, it alerts the user of an error. Finally, regardless of success or failure, it resets the loading state, clears the selected vehicle, and resets the form fields.



```
try {
  const body = await req.json();
  const { vehicleId, vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year, is_default } = body;
  console.log(body);

  const { error: vehicleError } = await supabaseAdmin
    .from("vehicles")
    .update({ vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year, is_default })
    .eq("vehicle_id", vehicleId);

  if (vehicleError) throw vehicleError;

  return NextResponse.json({ success: true });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.16: Update Vehicle Backend Code

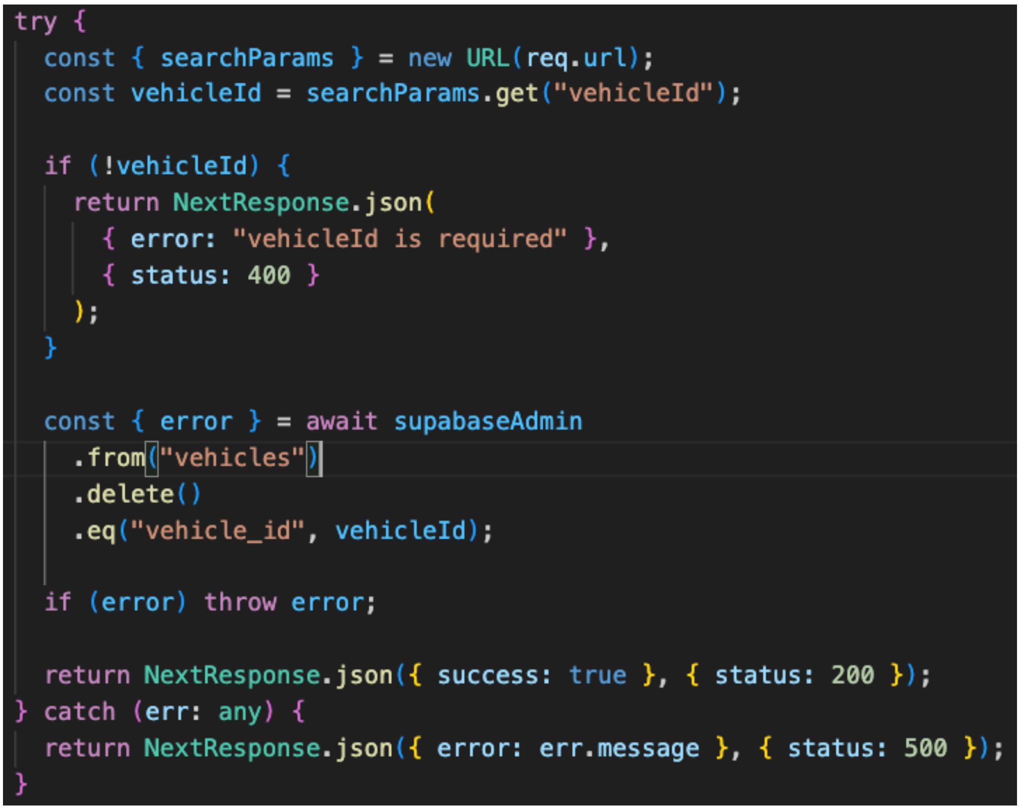
This code defines an asynchronous PUT API handler for updating vehicle records in a Supabase database. It first extracts the JSON body from the incoming request and destructures the vehicle information, including vehicleId, plate, colour, model, manufacturer, year and is_default. Using the Supabase admin client, it updates the corresponding record in the "vehicles" table where the vehicle_id matches the provided vehicleId. If the update succeeds, it returns a JSON response indicating success; if any error occurs during the update, it catches the error and returns a JSON response with the error message and a 500 status code. The console.log(body) line allows debugging by logging the received data.



```
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/vehicle?vehicleId=${vehicleId}`,
    { method: 'DELETE' }
  );

  const data = await response.json();

  if (response.ok) {
    await fetchVehicles();
    CustomToast('success', 'Vehicle deleted successfully', `${vehicles.find(v => v.vehicle_id === vehicleId)?.vehicle_plate} has been removed from your vehicles.`);
  } else {
    alert(data.error || 'Failed to delete vehicle');
  }
} catch (err: any) {
  alert(err.message);
}
```

Figure 6.17: Delete Vehicle Frontend Code

This code snippet is an asynchronous function that deletes a vehicle from the backend. It sends a DELETE request to the API endpoint with the `vehicleId` as a query parameter. After receiving the response, it parses the JSON data and checks if the deletion was successful. If so, it refreshes the vehicle list by calling `fetchVehicles()` and shows a custom toast notification confirming the deletion, including the vehicle plate number. If the request fails, it displays an alert with the error message from the server or a default failure message. Any unexpected errors during the request are caught and displayed using a standard alert.

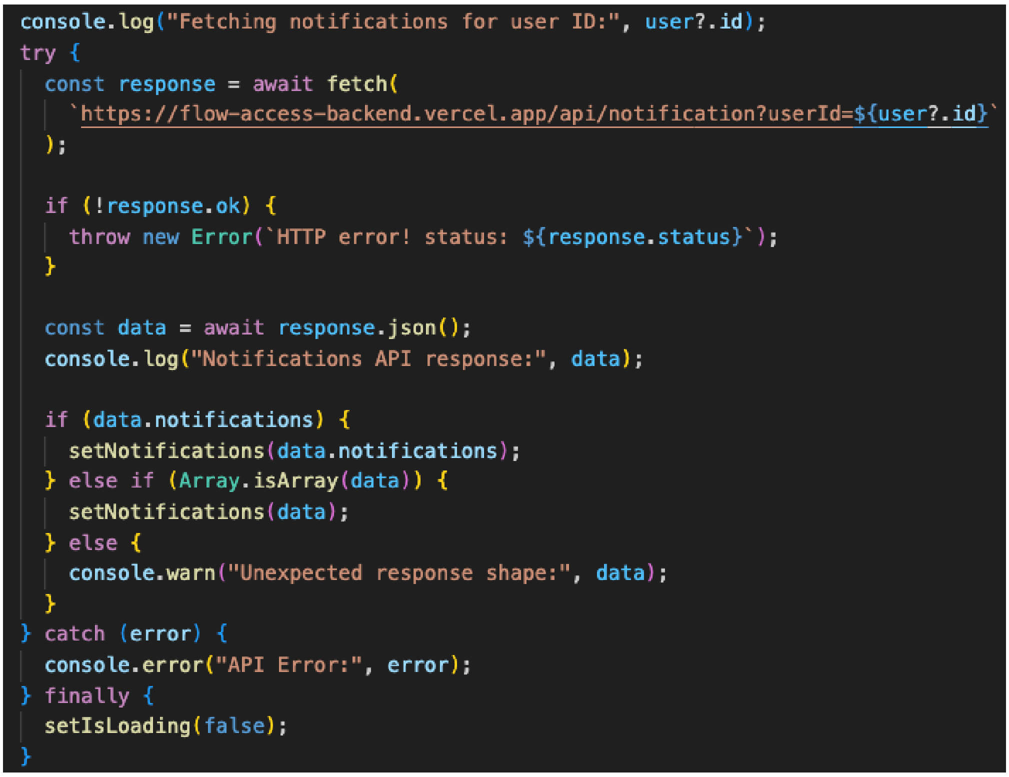


```
try {
  const { searchParams } = new URL(req.url);
  const vehicleId = searchParams.get("vehicleId");

  if (!vehicleId) {
    return NextResponse.json(
      { error: "vehicleId is required" },
      { status: 400 }
    );
  }

  const { error } = await supabaseAdmin
    .from("vehicles")
    .delete()
    .eq("vehicle_id", vehicleId);

  if (error) throw error;

  return NextResponse.json({ success: true }, { status: 200 });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.18: Delete Vehicle Backend Code

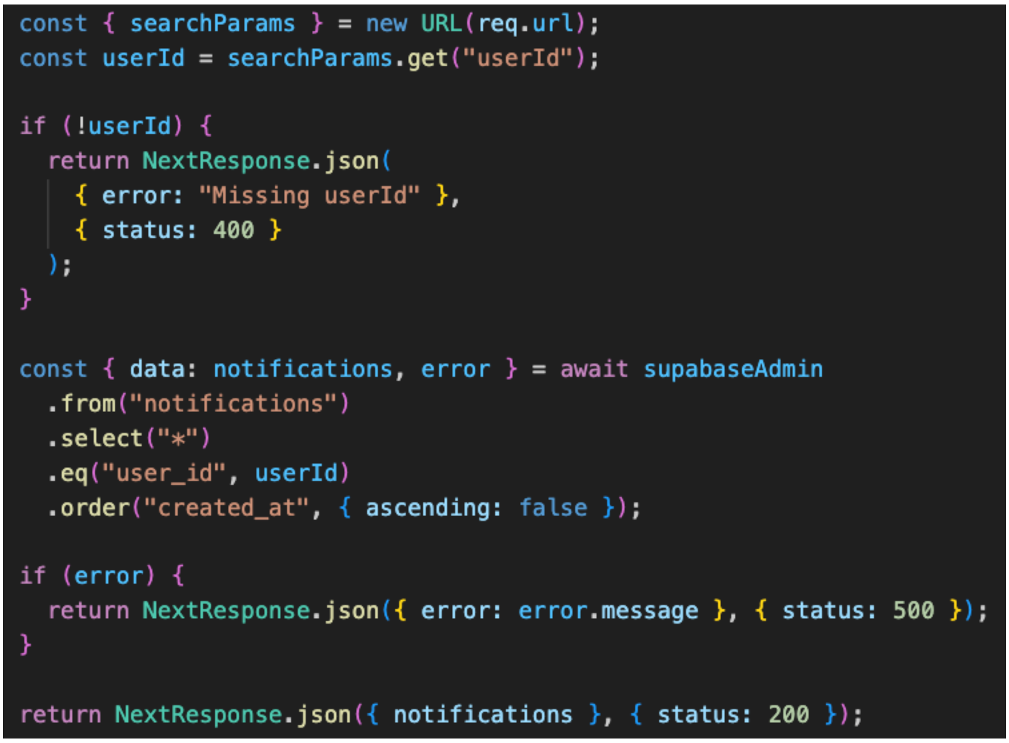
This code defines an asynchronous DELETE API handler for removing a vehicle from the Supabase database. It first extracts the vehicleId from the request's URL query parameters and returns a 400 error if vehicleId is missing. Using the Supabase admin client, it deletes the record in the "vehicles" table where vehicle_id matches the provided ID. If the deletion succeeds, it responds with a JSON object indicating success and a 200 status code. If any error occurs during the process, it catches the error and returns a JSON response with the error message and a 500 status code.

6.2.1.5 Notifications



```
console.log("Fetching notifications for user ID:", user?.id);
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/notification?userId=${user?.id}`
  );

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  const data = await response.json();
  console.log("Notifications API response:", data);

  if (data.notifications) {
    setNotifications(data.notifications);
  } else if (Array.isArray(data)) {
    setNotifications(data);
  } else {
    console.warn("Unexpected response shape:", data);
  }
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
}
```

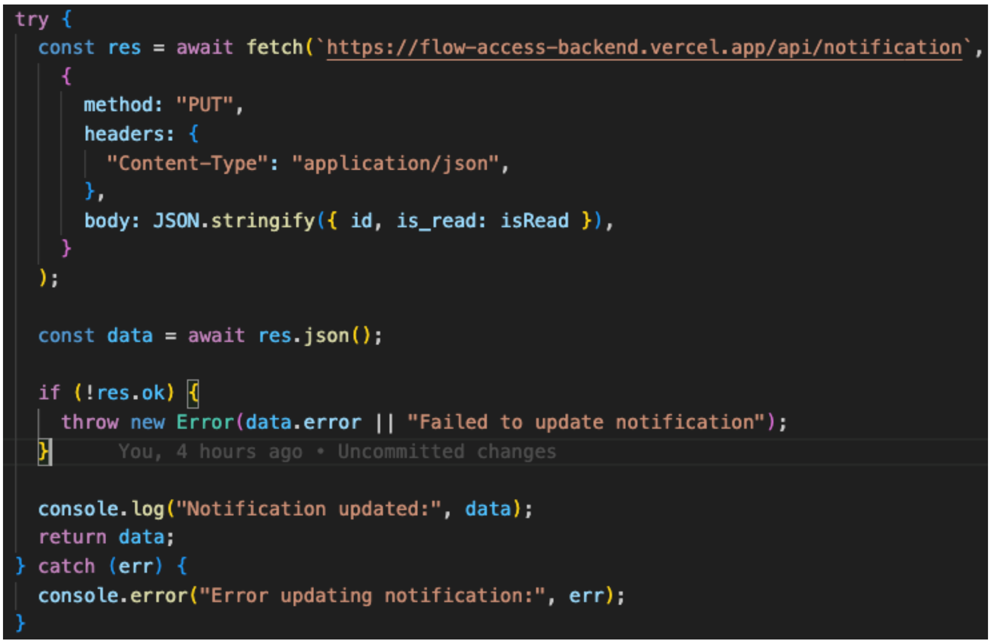
Figure 6.19: Fetch Notifications Frontend Code



```
const { searchParams } = new URL(req.url);
const userId = searchParams.get("userId");

if (!userId) {
  return NextResponse.json(
    { error: "Missing userId" },
    { status: 400 }
  );
}

const { data: notifications, error } = await supabaseAdmin
  .from("notifications")
  .select("*")
  .eq("user_id", userId)
  .order("created_at", { ascending: false });

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

return NextResponse.json({ notifications }, { status: 200 });
```

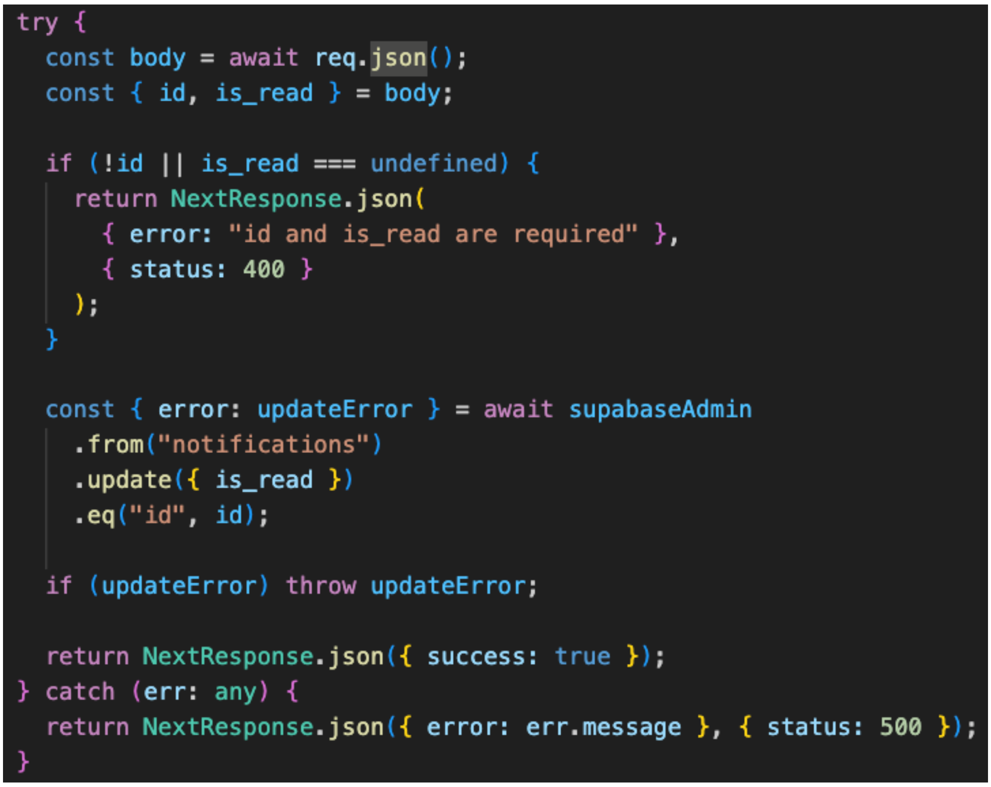
Figure 6.20: Fetch Notifications Backend Code

This code snippet is an asynchronous function that fetches notifications for a specific resident from the backend. It sends a GET request to the notifications API with the user’s ID as a query parameter. If the response is not OK, it throws an error with the HTTP status. Once the response is received, it parses the JSON data and logs it for debugging. The code then checks the structure of the returned data: if `data.notifications` exists, it updates the local `notifications` state; if the data itself is an array, it uses that directly; otherwise, it logs a warning about an unexpected response shape. Any errors during the fetch are caught and logged, and finally, the loading state is set to false.



```
try {
  const res = await fetch(`https://flow-access-backend.vercel.app/api/notification`,
    {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ id, is_read: isRead }),
    }
  );

  const data = await res.json();

  if (!res.ok) {
    throw new Error(data.error || "Failed to update notification");
  }      You, 4 hours ago • Uncommitted changes

  console.log("Notification updated:", data);
  return data;
} catch (err) {
  console.error("Error updating notification:", err);
}
```

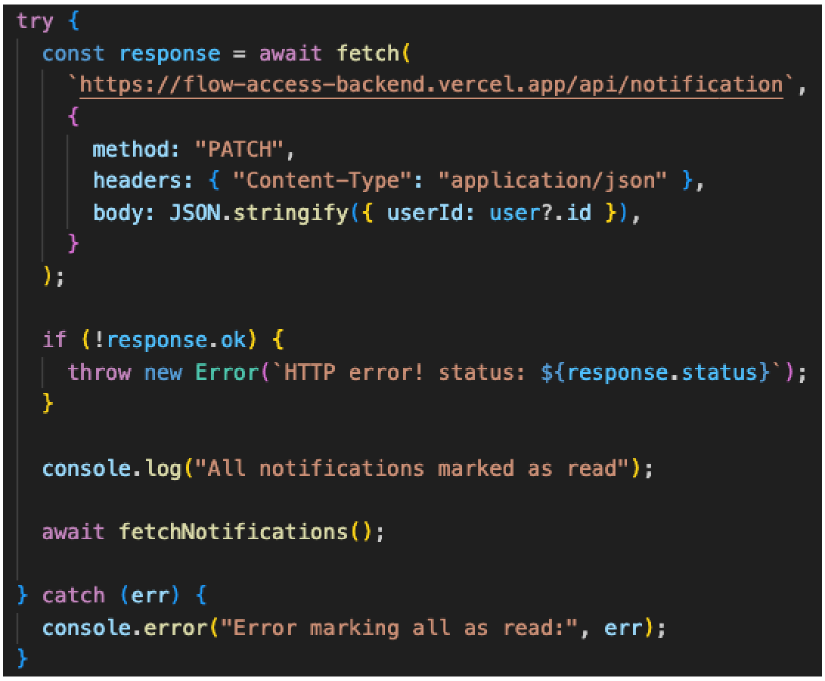
Figure 6.21:	Update 1 Notification as Read Frontend Code



```
try {
  const body = await req.json();
  const { id, is_read } = body;

  if (!id || is_read === undefined) {
    return NextResponse.json(
      { error: "id and is_read are required" },
      { status: 400 }
    );
  }

  const { error: updateError } = await supabaseAdmin
    .from("notifications")
    .update({ is_read })
    .eq("id", id);

  if (updateError) throw updateError;

  return NextResponse.json({ success: true });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

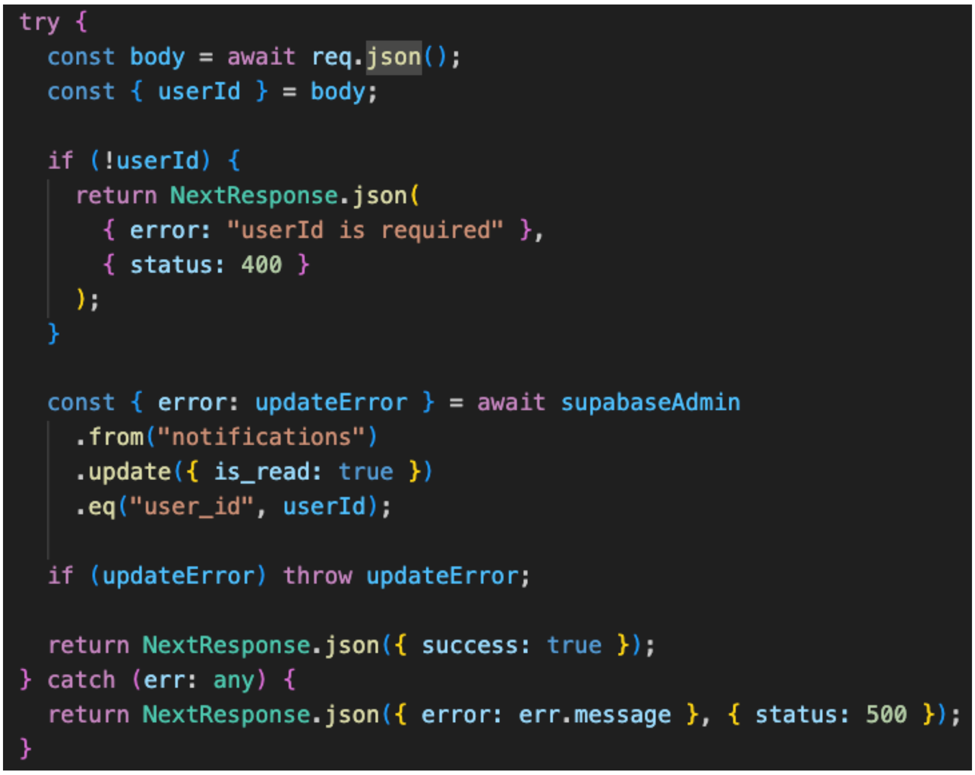
Figure 6.22: Update 1 Notification as Read Backend Code

This code updates a notification’s status in the backend. It sends a PUT request to the notifications API with a JSON body containing the notification id and its updated is_read status. After receiving the response, it parses the JSON data and checks if the request was successful. If not, it throws an error using the returned message or a default error message. On success, it logs the updated notification data for debugging and returns it. Any errors during the request are caught and logged to the console.



```
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/notification`,
    {
      method: "PATCH",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ userId: user?.id }),
    }
  );

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  console.log("All notifications marked as read");

  await fetchNotifications();

} catch (err) {
  console.error("Error marking all as read:", err);
}
```

Figure 6.23: Mark All Notifications as Read Frontend Code

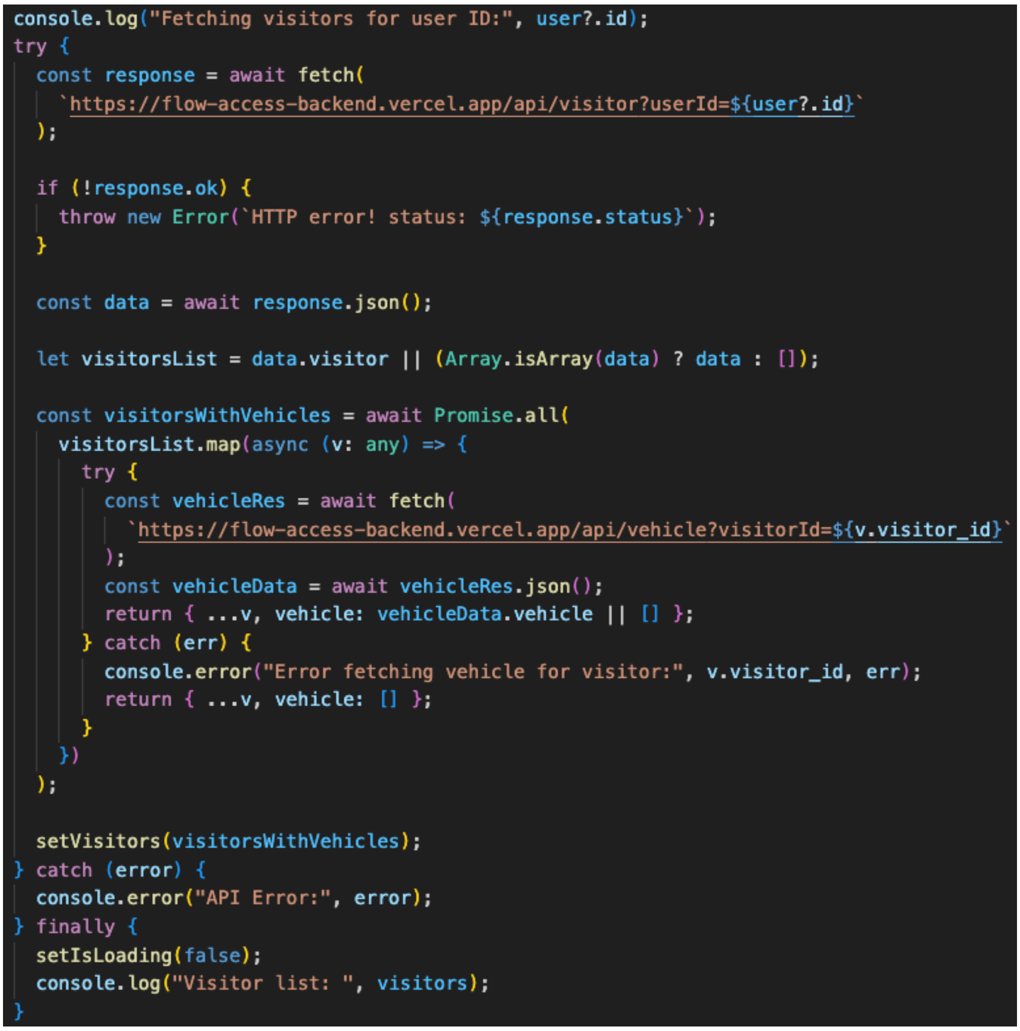


```
try {
  const body = await req.json();
  const { userId } = body;

  if (!userId) {
    return NextResponse.json(
      { error: "userId is required" },
      { status: 400 }
    );
  }

  const { error: updateError } = await supabaseAdmin
    .from("notifications")
    .update({ is_read: true })
    .eq("user_id", userId);

  if (updateError) throw updateError;

  return NextResponse.json({ success: true });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.24: Mark All Notifications as Read Backend Code

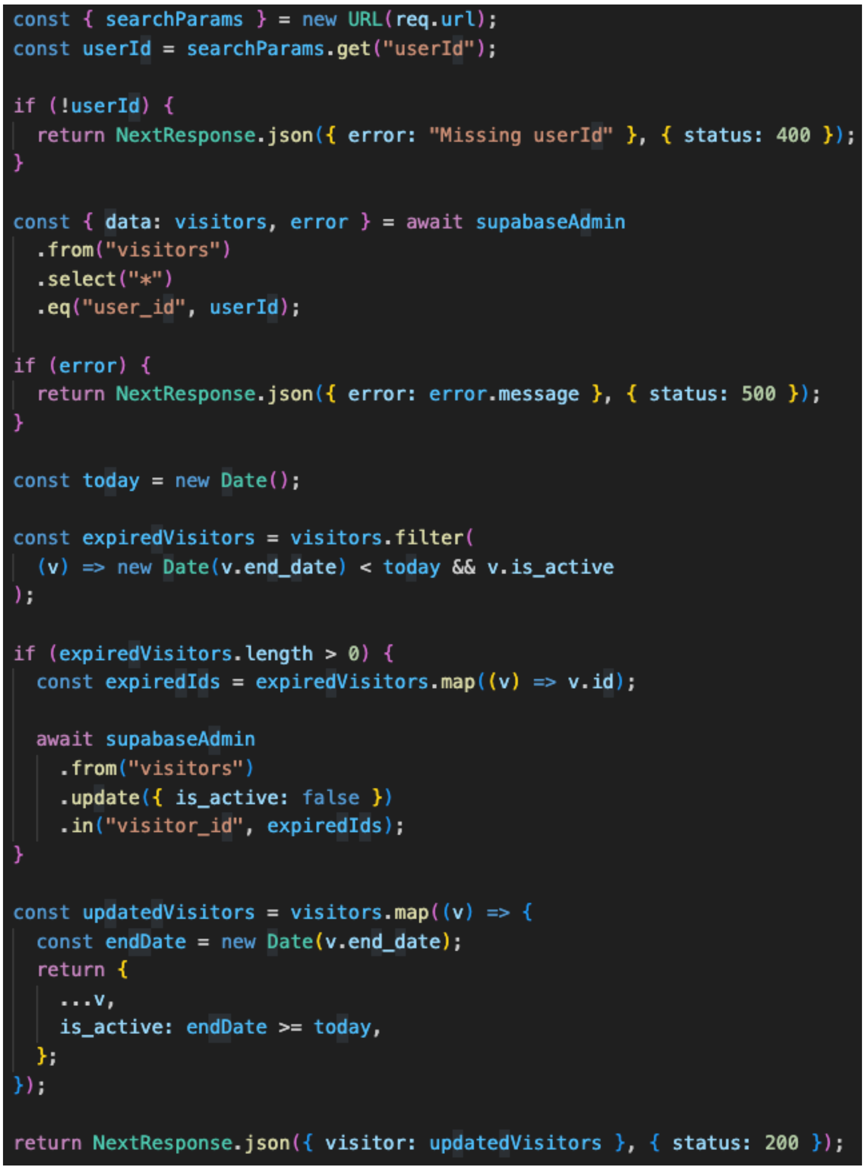
This function marks all notifications for a resident as read. It sends a PATCH request to the notifications API with the user's ID in the request body. If the response is not OK, it throws an error with the HTTP status. Upon a successful response, it logs a confirmation message and calls fetchNotifications() to refresh the local notification state. Any errors during the request are caught and logged to the console for debugging.

6.2.1.6 Manage Visitor Pass



```
console.log("Fetching visitors for user ID:", user?.id);
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/visitor?userId=${user?.id}`
  );

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  const data = await response.json();

  let visitorsList = data.visitor || (Array.isArray(data) ? data : []);

  const visitorsWithVehicles = await Promise.all(
    visitorsList.map(async (v: any) => {
      try {
        const vehicleRes = await fetch(
          `https://flow-access-backend.vercel.app/api/vehicle?visitorId=${v.visitor_id}`
        );
        const vehicleData = await vehicleRes.json();
        return { ...v, vehicle: vehicleData.vehicle || [] };
      } catch (err) {
        console.error("Error fetching vehicle for visitor:", v.visitor_id, err);
        return { ...v, vehicle: [] };
      }
    })
  );

  setVisitors(visitorsWithVehicles);
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
  console.log("Visitor list: ", visitors);
}
```

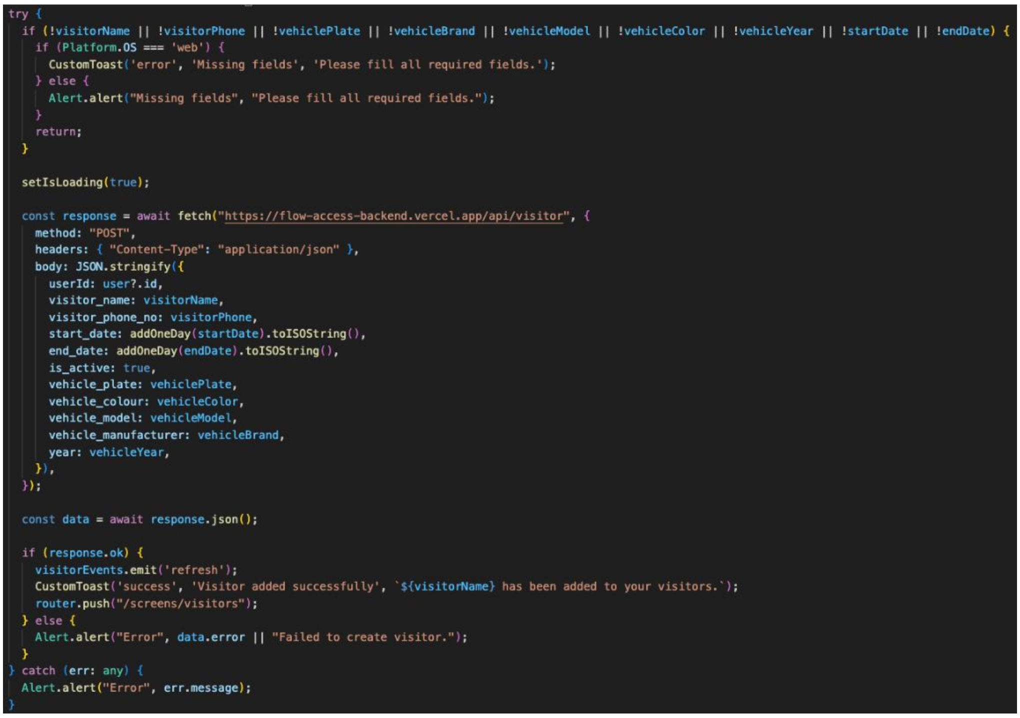
Figure 6.25: Fetch Visitors Frontend Code



```
const { searchParams } = new URL(req.url);
const userId = searchParams.get("userId");

if (!userId) {
  return NextResponse.json({ error: "Missing userId" }, { status: 400 });
}

const { data: visitors, error } = await supabaseAdmin
  .from("visitors")
  .select("*")
  .eq("user_id", userId);

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

const today = new Date();

const expiredVisitors = visitors.filter(
  (v) => new Date(v.end_date) < today && v.is_active
);

if (expiredVisitors.length > 0) {
  const expiredIds = expiredVisitors.map((v) => v.id);

  await supabaseAdmin
    .from("visitors")
    .update({ is_active: false })
    .in("visitor_id", expiredIds);
}

const updatedVisitors = visitors.map((v) => {
  const endDate = new Date(v.end_date);
  return {
    ...v,
    is_active: endDate >= today,
  };
});

return NextResponse.json({ visitor: updatedVisitors }, { status: 200 });
```

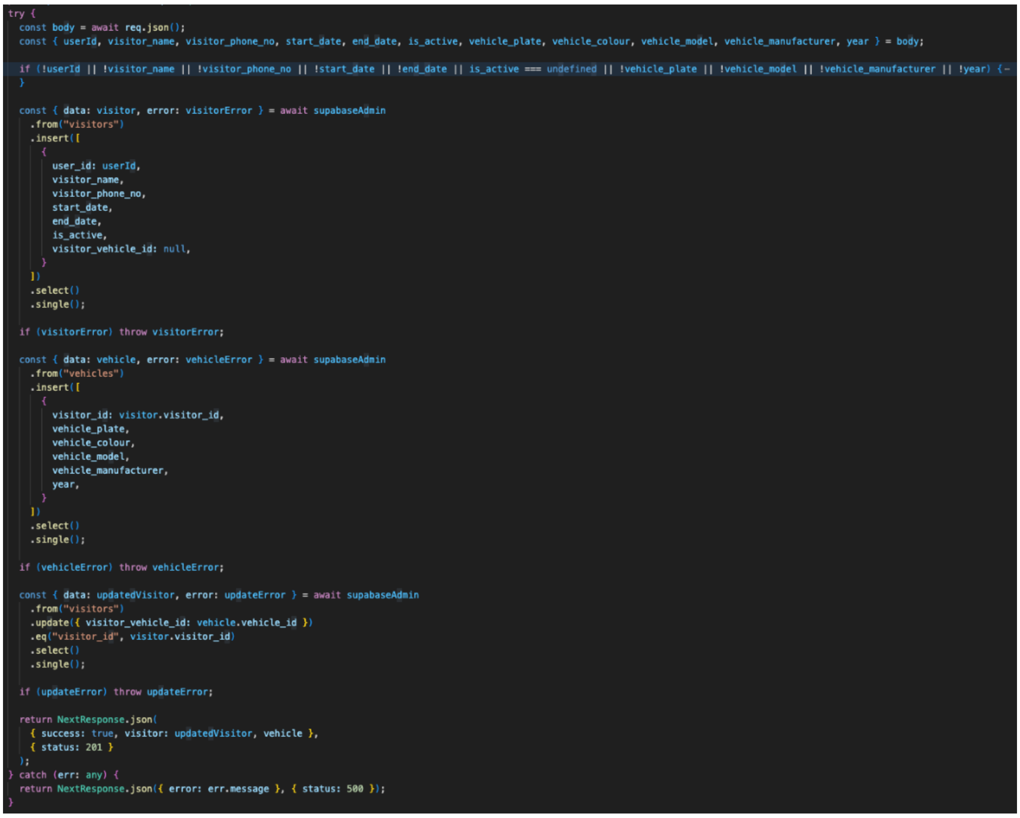
Figure 6.26: Fetch Visitors Backend Code

These code snippets show an asynchronous function that fetches a list of visitors for a specific user and their associated vehicles. It first sends a GET request to the visitor API with the user’s ID. If the response is successful, it parses the JSON data and ensures that visitorsList is an array. Then, for each visitor, it sends a separate GET request to the vehicle API to fetch vehicles linked to that visitor, combining the visitor and vehicle data into a single object. Any errors while fetching individual vehicles are caught and logged, with an empty vehicle array returned for that visitor. Finally, the combined visitor-vehicle data is stored in state via setVisitors(), the loading state is cleared, and the visitor list is logged for debugging.



```
try {
  if (!visitorName || !visitorPhone || !vehiclePlate || !vehicleBrand || !vehicleModel || !vehicleColor || !vehicleYear || !startDate || !endDate) {
    if (Platform.OS === 'web') {
      CustomToast('error', 'Missing fields', 'Please fill all required fields.');
    } else {
      Alert.alert("Missing fields", "Please fill all required fields.");
    }
    return;
  }

  setIsLoading(true);

  const response = await fetch("https://flow-access-backend.vercel.app/api/visitor", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      userId: user?.id,
      visitor_name: visitorName,
      visitor_phone_no: visitorPhone,
      start_date: addOneDay(startDate).toISOString(),
      end_date: addOneDay(endDate).toISOString(),
      is_active: true,
      vehicle_plate: vehiclePlate,
      vehicle_colour: vehicleColor,
      vehicle_model: vehicleModel,
      vehicle_manufacturer: vehicleBrand,
      year: vehicleYear,
    }),
  });

  const data = await response.json();

  if (response.ok) {
    visitorEvents.emit('refresh');
    CustomToast('success', 'Visitor added successfully', `${visitorName} has been added to your visitors.`);
    router.push("/screens/visitors");
  } else {
    Alert.alert("Error", data.error || "Failed to create visitor.");
  }
} catch (err: any) {
  Alert.alert("Error", err.message);
}
```

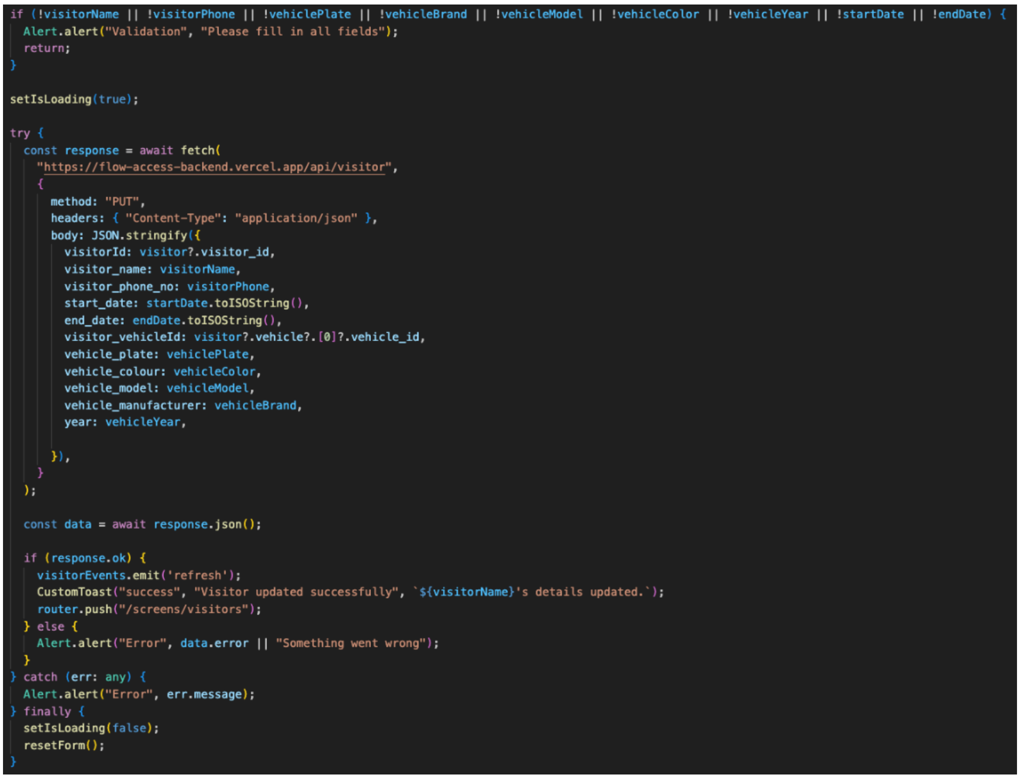
Figure 6.27: Add New Visitor Frontend Code



```
try {
  const body = await req.json();
  const { userId, visitor_name, visitor_phone_no, start_date, end_date, is_active, vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year } = body;

  if (!userId || !visitor_name || !visitor_phone_no || !start_date || !end_date || is_active === undefined || !vehicle_plate || !vehicle_model || !vehicle_manufacturer || !year) {-
  }

  const { data: visitor, error: visitorError } = await supabaseAdmin
    .from("visitors")
    .insert([
      {
        user_id: userId,
        visitor_name,
        visitor_phone_no,
        start_date,
        end_date,
        is_active,
        visitor_vehicle_id: null,
      }
    ])
    .select()
    .single();

  if (visitorError) throw visitorError;

  const { data: vehicle, error: vehicleError } = await supabaseAdmin
    .from("vehicles")
    .insert([
      {
        visitor_id: visitor.visitor_id,
        vehicle_plate,
        vehicle_colour,
        vehicle_model,
        vehicle_manufacturer,
        year,
      }
    ])
    .select()
    .single();

  if (vehicleError) throw vehicleError;

  const { data: updatedVisitor, error: updateError } = await supabaseAdmin
    .from("visitors")
    .update({ visitor_vehicle_id: vehicle.vehicle_id })
    .eq("visitor_id", visitor.visitor_id)
    .select()
    .single();

  if (updateError) throw updateError;

  return NextResponse.json(
    { success: true, visitor: updatedVisitor, vehicle },
    { status: 201 }
  );
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

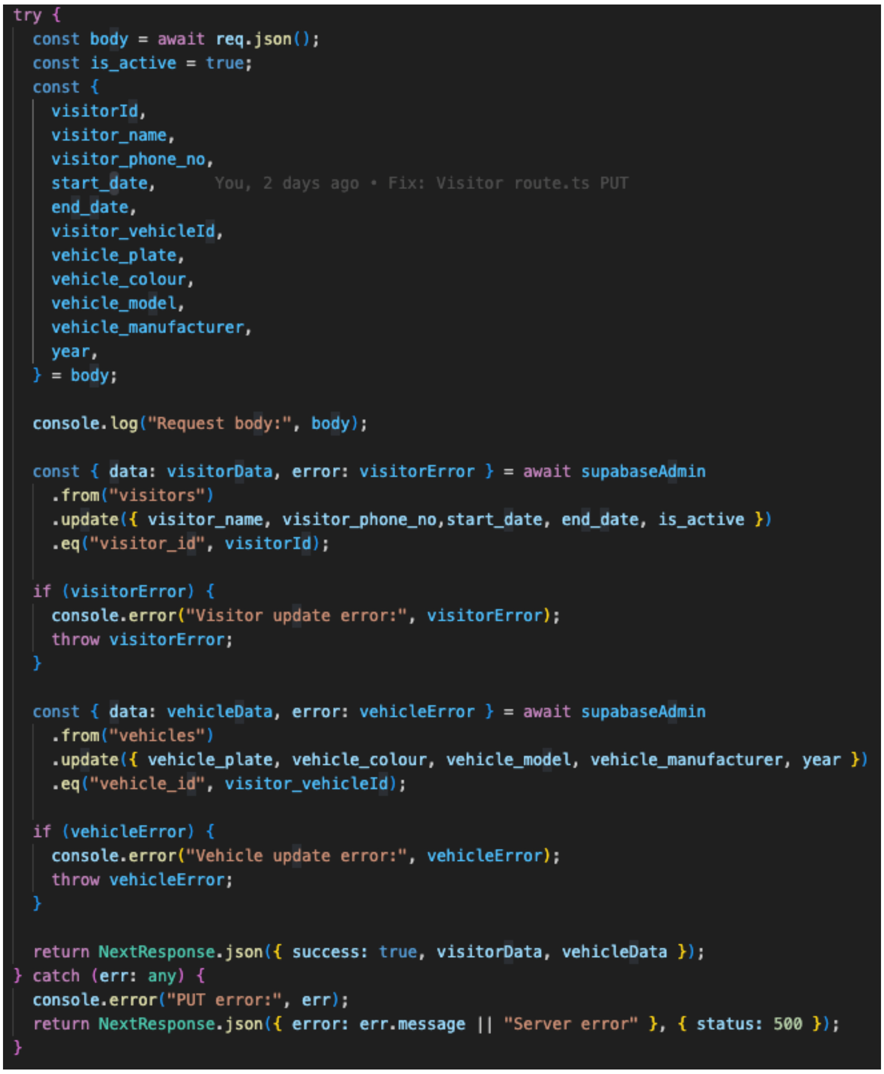
Figure 6.28: Add New Visitor Backend Code

The function handleAddVisitor validates that all required visitor and vehicle fields are filled. If not, it shows an error alert or toast depending on the platform. If validation passes, it sets a loading state and sends a POST request to the backend API with the visitor and vehicle details, including adjusted start and end dates. On success, it emits a refresh event, shows a success toast, and navigates to the visitors screen; if it fails, it shows an error alert.



```
if (!visitorName || !visitorPhone || !vehiclePlate || !vehicleBrand || !vehicleModel || !vehicleColor || !vehicleYear || !startDate || !endDate) {
  Alert.alert("Validation", "Please fill in all fields");
  return;
}

setIsLoading(true);

try {
  const response = await fetch(
    "https://flow-access-backend.vercel.app/api/visitor",
    {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        visitorId: visitor?.visitor_id,
        visitor_name: visitorName,
        visitor_phone_no: visitorPhone,
        start_date: startDate.toISOString(),
        end_date: endDate.toISOString(),
        visitor_vehicleId: visitor?.vehicle?.[0]?.vehicle_id,
        vehicle_plate: vehiclePlate,
        vehicle_colour: vehicleColor,
        vehicle_model: vehicleModel,
        vehicle_manufacturer: vehicleBrand,
        year: vehicleYear,

      }),
    }
  );

  const data = await response.json();

  if (response.ok) {
    visitorEvents.emit('refresh');
    CustomToast("success", "Visitor updated successfully", `${visitorName}'s details updated.`);
    router.push("/screens/visitors");
  } else {
    Alert.alert("Error", data.error || "Something went wrong");
  }
} catch (err: any) {
  Alert.alert("Error", err.message);
} finally {
  setIsLoading(false);
  resetForm();
}
```

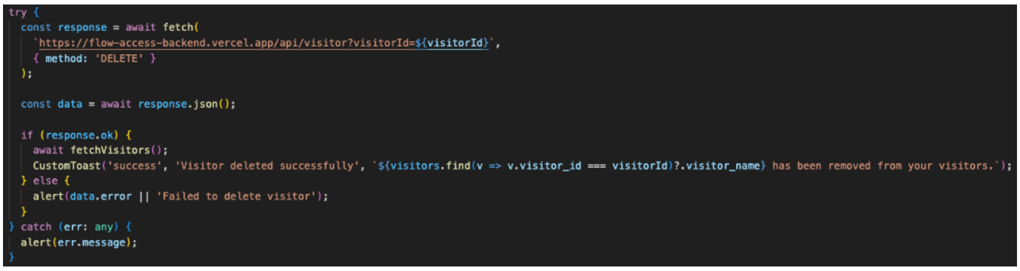
Figure 6.29: Update Visitor Frontend Code



```
try {
  const body = await req.json();
  const is_active = true;
  const {
    visitorId,
    visitor_name,
    visitor_phone_no,
    start_date,      You, 2 days ago • Fix: Visitor route.ts PUT
    end_date,
    visitor_vehicleId,
    vehicle_plate,
    vehicle_colour,
    vehicle_model,
    vehicle_manufacturer,
    year,
  } = body;

  console.log("Request body:", body);

  const { data: visitorData, error: visitorError } = await supabaseAdmin
    .from("visitors")
    .update({ visitor_name, visitor_phone_no,start_date, end_date, is_active })
    .eq("visitor_id", visitorId);

  if (visitorError) {
    console.error("Visitor update error:", visitorError);
    throw visitorError;
  }

  const { data: vehicleData, error: vehicleError } = await supabaseAdmin
    .from("vehicles")
    .update({ vehicle_plate, vehicle_colour, vehicle_model, vehicle_manufacturer, year })
    .eq("vehicle_id", visitor_vehicleId);

  if (vehicleError) {
    console.error("Vehicle update error:", vehicleError);
    throw vehicleError;
  }

  return NextResponse.json({ success: true, visitorData, vehicleData });
} catch (err: any) {
  console.error("PUT error:", err);
  return NextResponse.json({ error: err.message || "Server error" }, { status: 500 });
}
```

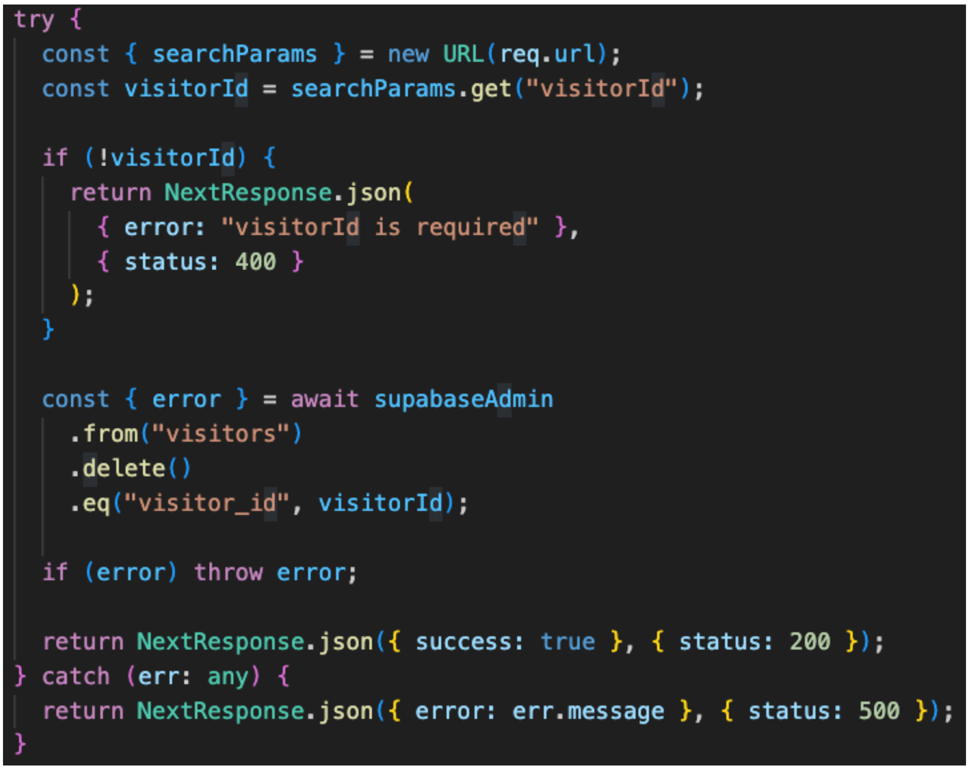
Figure 6.30: Update Visitor Backend Code

The code performs similar validation as above code which add new visitor. Then, it sets the loading state, and sends a PUT request to update an existing visitor and their vehicle details. It includes the visitor ID and the vehicle ID for updating. After receiving the response, it handles success by emitting a refresh event, showing a success toast, and navigating back, or shows an error alert if the update fails. In both cases, the loading state is cleared, and the form is reset after the operation.



```
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/visitor?visitorId=${visitorId}`,
    { method: 'DELETE' }
  );

  const data = await response.json();

  if (response.ok) {
    await fetchVisitors();
    CustomToast('success', 'Visitor deleted successfully', `${visitors.find(v => v.visitor_id === visitorId)?.visitor_name} has been removed from your visitors.`);
  } else {
    alert(data.error || 'Failed to delete visitor');
  }
} catch (err: any) {
  alert(err.message);
}
```

Figure 6.31: Delete Visitor Frontend Code

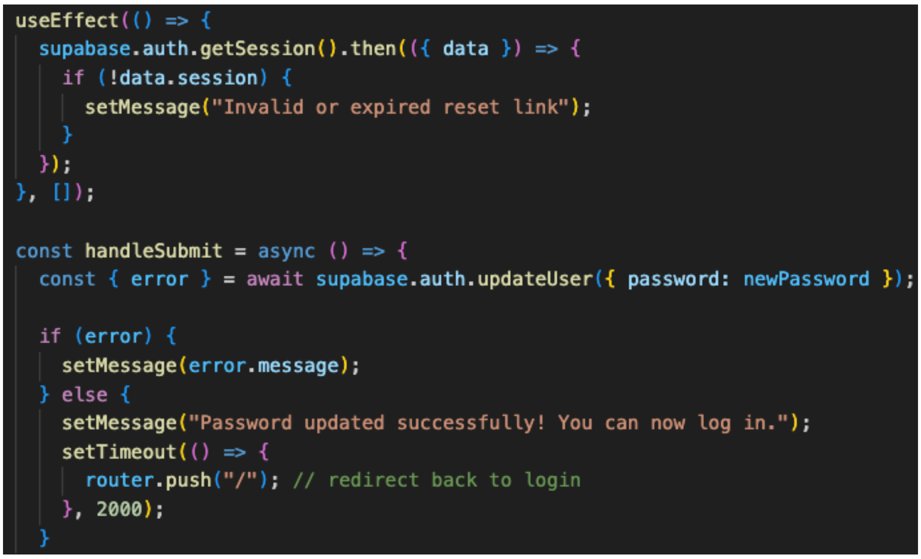


```
try {
  const { searchParams } = new URL(req.url);
  const visitorId = searchParams.get("visitorId");

  if (!visitorId) {
    return NextResponse.json(
      { error: "visitorId is required" },
      { status: 400 }
    );
  }

  const { error } = await supabaseAdmin
    .from("visitors")
    .delete()
    .eq("visitor_id", visitorId);

  if (error) throw error;

  return NextResponse.json({ success: true }, { status: 200 });
} catch (err: any) {
  return NextResponse.json({ error: err.message }, { status: 500 });
}
```

Figure 6.32: Delete Visitor Backend Code

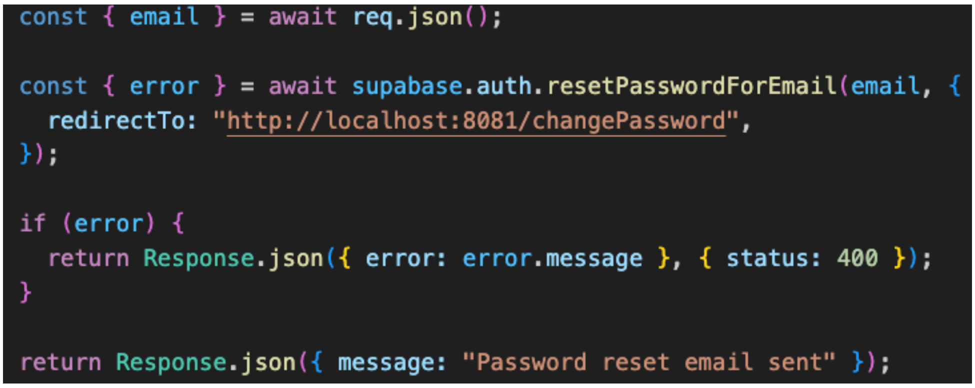
These code snippets show an asynchronous function that deletes a visitor from the backend. It sends a DELETE request to the visitor API with the visitorId as a query parameter. After receiving the response, it parses the JSON data and checks if the deletion was successful. If so, it refreshes the visitor list by calling fetchVisitors() and shows a custom toast notification confirming the deletion, including the visitor’s name. If the deletion fails, it displays an alert with the error message returned from the server or a default failure message. Any unexpected errors during the request are caught and displayed using an alert.

6.2.1.7 Reset Password



```
useEffect(() => {
  supabase.auth.getSession().then(({ data }) => {
    if (!data.session) {
      setMessage("Invalid or expired reset link");
    }
  });
}, []);

const handleSubmit = async () => {
  const { error } = await supabase.auth.updateUser({ password: newPassword });

  if (error) {
    setMessage(error.message);
  } else {
    setMessage("Password updated successfully! You can now log in.");
    setTimeout(() => {
      router.push("/"); // redirect back to login
    }, 2000);
  }
```

Figure 6.33: Reset Password Frontend Code



```
const { email } = await req.json();

const { error } = await supabase.auth.resetPasswordForEmail(email, {
  redirectTo: "http://localhost:8081/changePassword",
});

if (error) {
  return Response.json({ error: error.message }, { status: 400 });
}

return Response.json({ message: "Password reset email sent" });
```

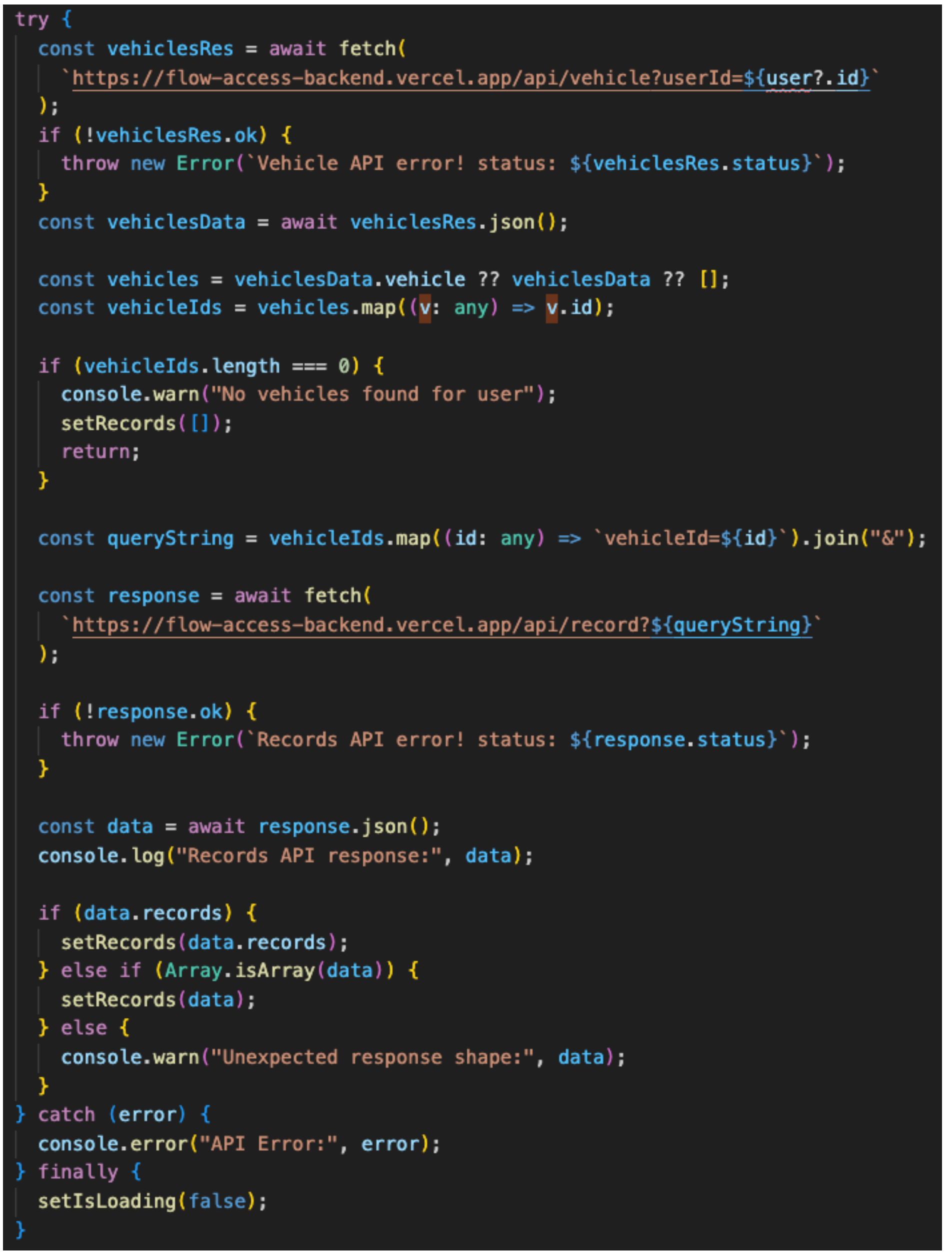
Figure 6.34: Reset Password Backend Code

These code snippet handled password reset functionality using Supabase authentication in a React component.

The useEffect hook runs on component mount to check the current session via supabase.auth.getSession(). If there is no active session, it sets a message indicating that the reset link is invalid or expired.

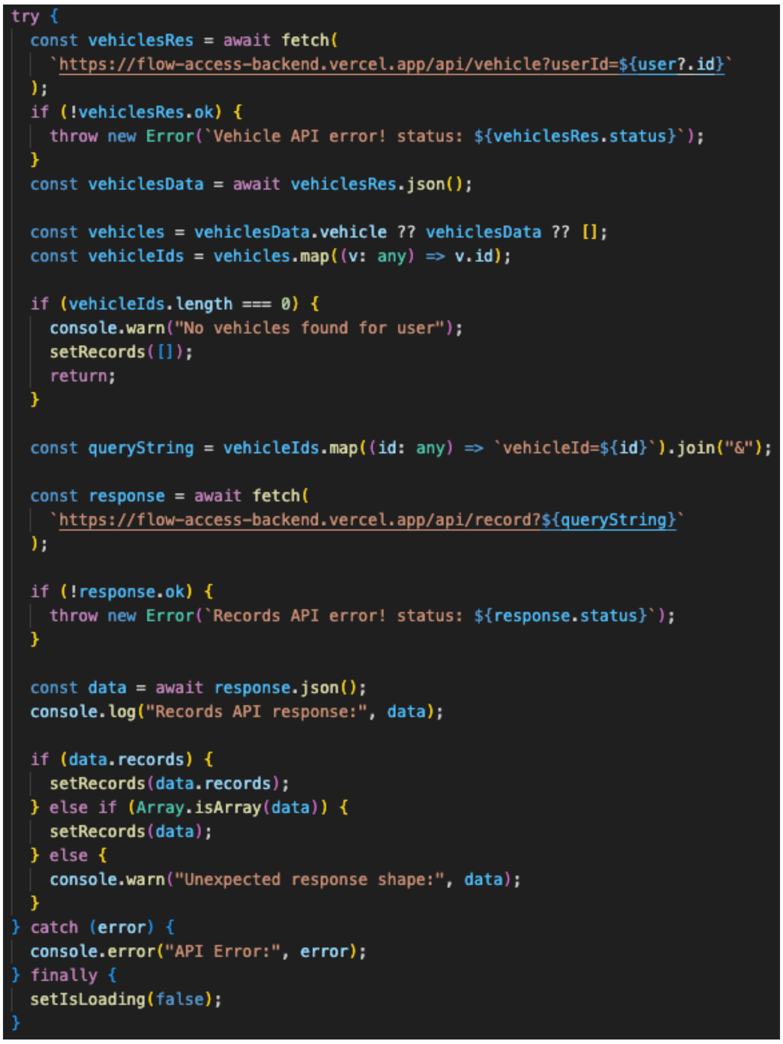
The handleSubmit function updates the user’s password by calling supabase.auth.updateUser() with the new password. If an error occurs, it displays the error message; otherwise, it shows a success message and, after a 2-second delay, redirects the user back to the login page using router.push("/").

6.2.1.8 Vehicle Logs



```
try {
  const vehiclesRes = await fetch(
    `https://flow-access-backend.vercel.app/api/vehicle?userId=${user?.id}`
  );
  if (!vehiclesRes.ok) {
    throw new Error(`Vehicle API error! status: ${vehiclesRes.status}`);
  }
  const vehiclesData = await vehiclesRes.json();

  const vehicles = vehiclesData.vehicle ?? vehiclesData ?? [];
  const vehicleIds = vehicles.map((v: any) => v.id);

  if (vehicleIds.length === 0) {
    console.warn("No vehicles found for user");
    setRecords([]);
    return;
  }

  const queryString = vehicleIds.map((id: any) => `vehicleId=${id}`).join("&");

  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/record?${queryString}`
  );

  if (!response.ok) {
    throw new Error(`Records API error! status: ${response.status}`);
  }

  const data = await response.json();
  console.log("Records API response:", data);

  if (data.records) {
    setRecords(data.records);
  } else if (Array.isArray(data)) {
    setRecords(data);
  } else {
    console.warn("Unexpected response shape:", data);
  }
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
}
```

Figure 6.35: Fetch Vehicle Logs Frontend Code



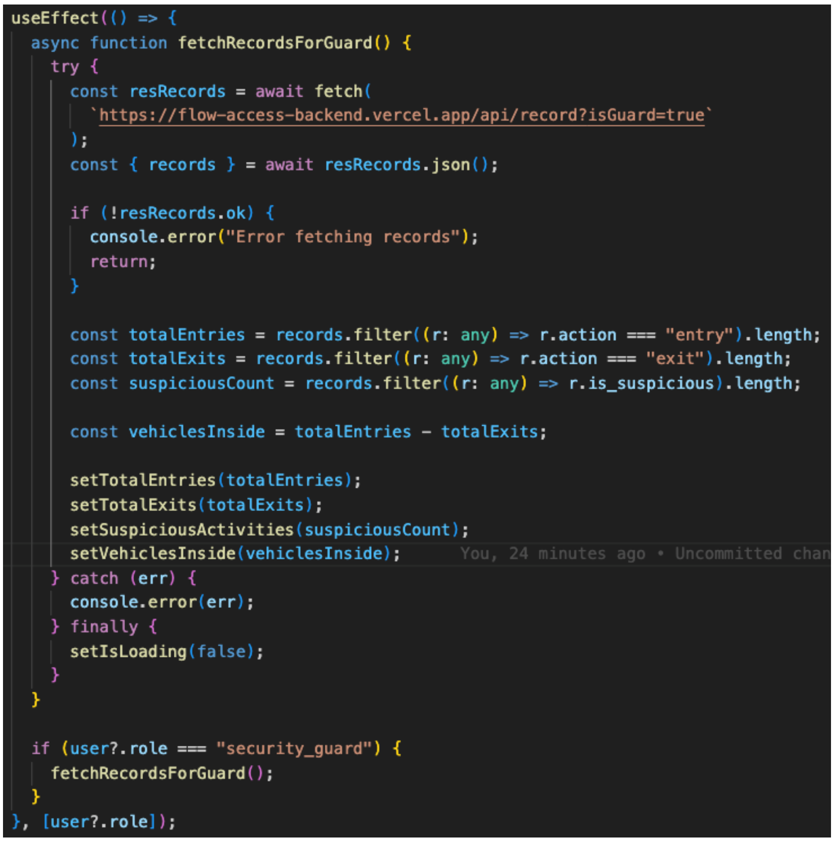
```
try {
  const vehiclesRes = await fetch(
    `https://flow-access-backend.vercel.app/api/vehicle?userId=${user?.id}`
  );
  if (!vehiclesRes.ok) {
    throw new Error(`Vehicle API error! status: ${vehiclesRes.status}`);
  }
  const vehiclesData = await vehiclesRes.json();

  const vehicles = vehiclesData.vehicle ?? vehiclesData ?? [];
  const vehicleIds = vehicles.map((v: any) => v.id);

  if (vehicleIds.length === 0) {
    console.warn("No vehicles found for user");
    setRecords([]);
    return;
  }

  const queryString = vehicleIds.map((id: any) => `vehicleId=${id}`).join("&");

  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/record?${queryString}`
  );

  if (!response.ok) {
    throw new Error(`Records API error! status: ${response.status}`);
  }

  const data = await response.json();
  console.log("Records API response:", data);

  if (data.records) {
    setRecords(data.records);
  } else if (Array.isArray(data)) {
    setRecords(data);
  } else {
    console.warn("Unexpected response shape:", data);
  }
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
}
```

Figure 6.36: Fetch Vehicle Logs Backend Code

These code snippets show an asynchronous function that fetches vehicle access records for a specific resident. It first sends a GET request to the vehicle API using the user's ID to retrieve the list of vehicles. If the request fails, it throws an error. It then extracts vehicle IDs from the response; if no vehicles are found, it logs a warning and clears the records state. Next, it constructs a query string with all vehicle IDs and sends another GET request to the records API to fetch related access records. The response is parsed, and the records state is updated depending on whether the response contains a records field or is directly an array. Any unexpected response shapes are logged as warnings. Errors during the fetch are caught and logged, and the loading state is cleared in the finally block.

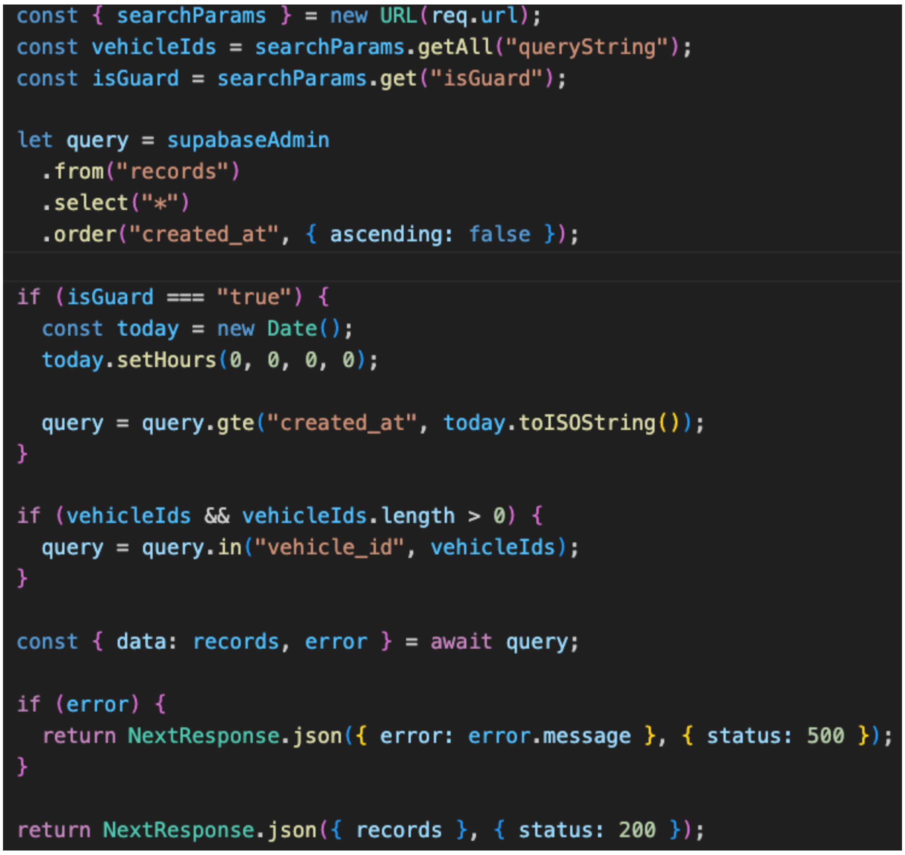
6.2.2 Security Guard

6.2.2.1 Dashboard



```
useEffect(() => {
  async function fetchRecordsForGuard() {
    try {
      const resRecords = await fetch(
        `https://flow-access-backend.vercel.app/api/record?isGuard=true`
      );
      const { records } = await resRecords.json();

      if (!resRecords.ok) {
        console.error("Error fetching records");
        return;
      }

      const totalEntries = records.filter((r: any) => r.action === "entry").length;
      const totalExits = records.filter((r: any) => r.action === "exit").length;
      const suspiciousCount = records.filter((r: any) => r.is_suspicious).length;

      const vehiclesInside = totalEntries - totalExits;

      setTotalEntries(totalEntries);
      setTotalExits(totalExits);
      setSuspiciousActivities(suspiciousCount);
      setVehiclesInside(vehiclesInside);      You, 24 minutes ago • Uncommitted chan
    } catch (err) {
      console.error(err);
    } finally {
      setIsLoading(false);
    }
  }

  if (user?.role === "security_guard") {
    fetchRecordsForGuard();
  }
}, [user?.role]);
```

Figure 6.37: Fetch Dashboard Information Frontend Code

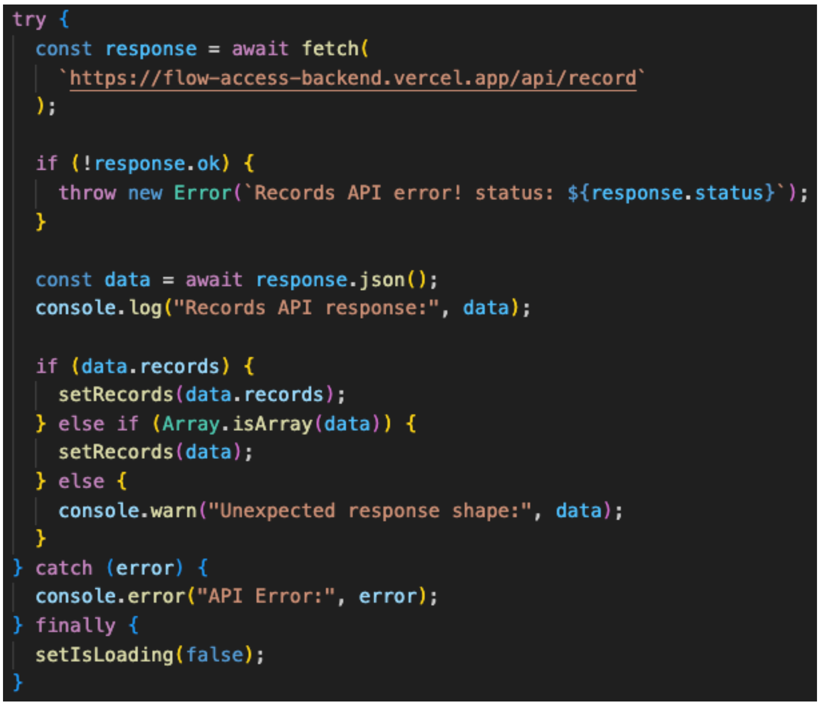


```
const { searchParams } = new URL(req.url);
const vehicleIds = searchParams.getAll("queryString");
const isGuard = searchParams.get("isGuard");

let query = supabaseAdmin
  .from("records")
  .select("*")
  .order("created_at", { ascending: false });

if (isGuard === "true") {
  const today = new Date();
  today.setHours(0, 0, 0, 0);

  query = query.gte("created_at", today.toISOString());
}

if (vehicleIds && vehicleIds.length > 0) {
  query = query.in("vehicle_id", vehicleIds);
}

const { data: records, error } = await query;

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

return NextResponse.json({ records }, { status: 200 });
```

Figure 6.38: Fetch Dashboard Information Backend Code

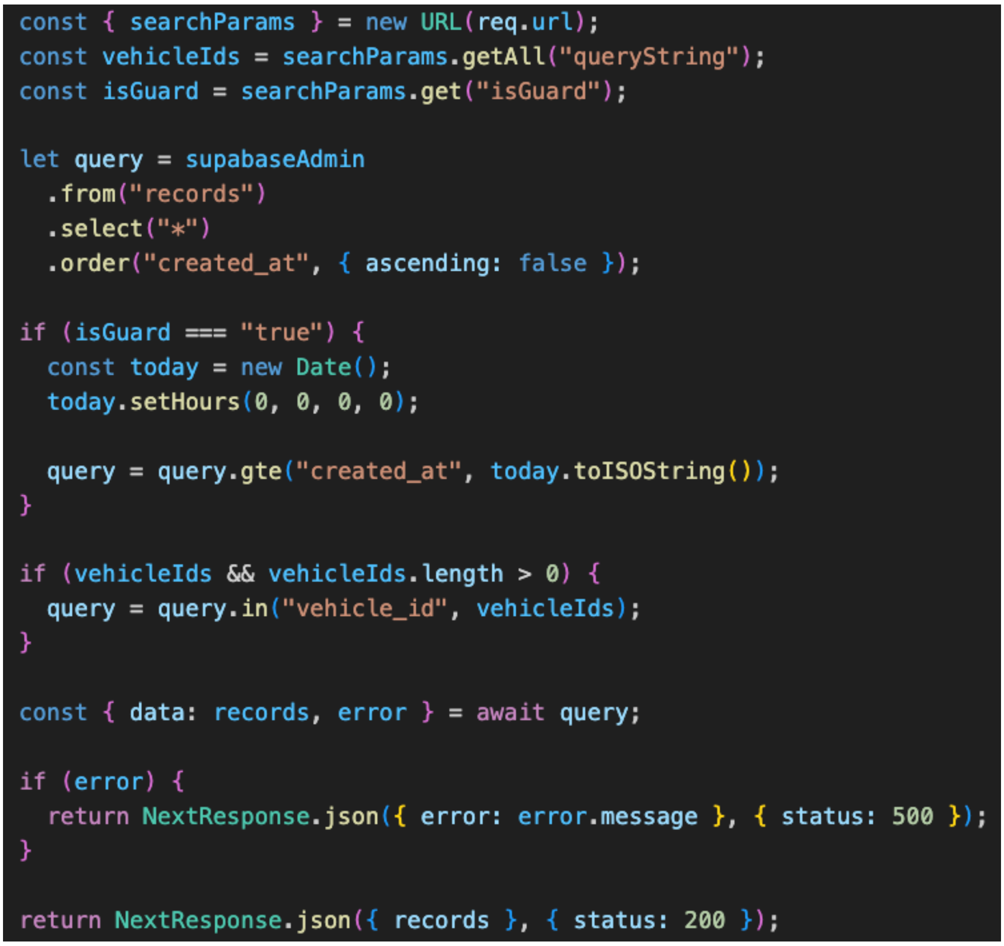
This code snippet defines an asynchronous function, fetchRecordsForGuard, that fetches vehicle access records for security guards. It sends a GET request to the records API with isGuard=true to retrieve relevant records. After parsing the JSON response, it checks for request success; if unsuccessful, it logs an error. The code then calculates key statistics: the total number of entries, total exits, the count of suspicious activities, and the number of vehicles currently inside (entries minus exits). These values are stored in state variables using setTotalEntries, setTotalExits, setSuspiciousActivities, and setVehiclesInside. Any errors during the fetch are caught and logged, and the loading state is cleared in the finally block. Finally, the function is called if the current user's role is "security_guard".

6.2.2.2 Vehicle Logs



```
try {
  const response = await fetch(
    `https://flow-access-backend.vercel.app/api/record`
  );

  if (!response.ok) {
    throw new Error(`Records API error! status: ${response.status}`);
  }

  const data = await response.json();
  console.log("Records API response:", data);

  if (data.records) {
    setRecords(data.records);
  } else if (Array.isArray(data)) {
    setRecords(data);
  } else {
    console.warn("Unexpected response shape:", data);
  }
} catch (error) {
  console.error("API Error:", error);
} finally {
  setIsLoading(false);
}
```

Figure 6.39: Fetch Daily Vehicle Logs Frontend Code

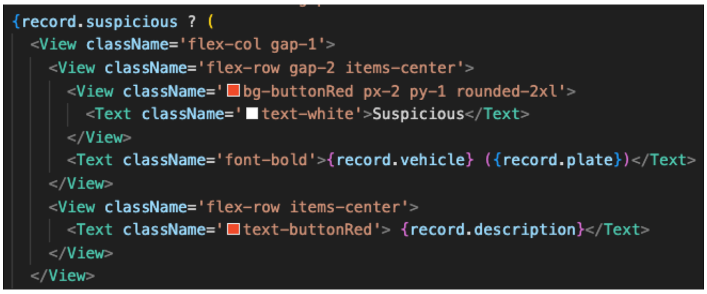


```
const { searchParams } = new URL(req.url);
const vehicleIds = searchParams.getAll("queryString");
const isGuard = searchParams.get("isGuard");

let query = supabaseAdmin
  .from("records")
  .select("*")
  .order("created_at", { ascending: false });

if (isGuard === "true") {
  const today = new Date();
  today.setHours(0, 0, 0, 0);

  query = query.gte("created_at", today.toISOString());
}

if (vehicleIds && vehicleIds.length > 0) {
  query = query.in("vehicle_id", vehicleIds);
}

const { data: records, error } = await query;

if (error) {
  return NextResponse.json({ error: error.message }, { status: 500 });
}

return NextResponse.json({ records }, { status: 200 });
```

Figure 6.40: Fetch Daily Vehicle Logs Backend Code

This code snippet is an asynchronous function that fetches all vehicle access records from the backend. It sends a GET request to the records API and checks if the response is successful, throwing an error if it isn't. After parsing the JSON response, it updates the local records state based on the response shape using data.records if available, or the data itself if it is an array. Any unexpected response structures are logged as warnings. Errors encountered during the fetch are caught and logged, and the loading state is cleared in the finally block.

6.2.2.3 Suspicious Event



```
{record.suspicious ? (
  <View className='flex-col gap-1'>
    <View className='flex-row gap-2 items-center'>
      <View className='🟥bg-buttonRed px-2 py-1 rounded-2xl'>
        <Text className='⬜text-white'>Suspicious</Text>
      </View>
      <Text className='font-bold'>{record.vehicle} ({record.plate})</Text>
    </View>
    <View className='flex-row items-center'>
      <Text className='🟥text-buttonRed'> {record.description}</Text>
    </View>
  </View>
```

Figure 6.41: Show Suspicious Event Frontend Code

This JSX snippet conditionally renders a vehicle access record based on whether it is marked as suspicious. If record.suspicious is true, it highlights the record with a red “Suspicious” badge, displays the vehicle name and plate in bold, and shows the suspicious activity description in red text. If record.suspicious is false, it simply displays the vehicle name and plate in bold along with the owner’s name in smaller, gray text. The layout uses flexbox styling to align and space elements properly, ensuring that suspicious records are visually distinguished from normal ones.

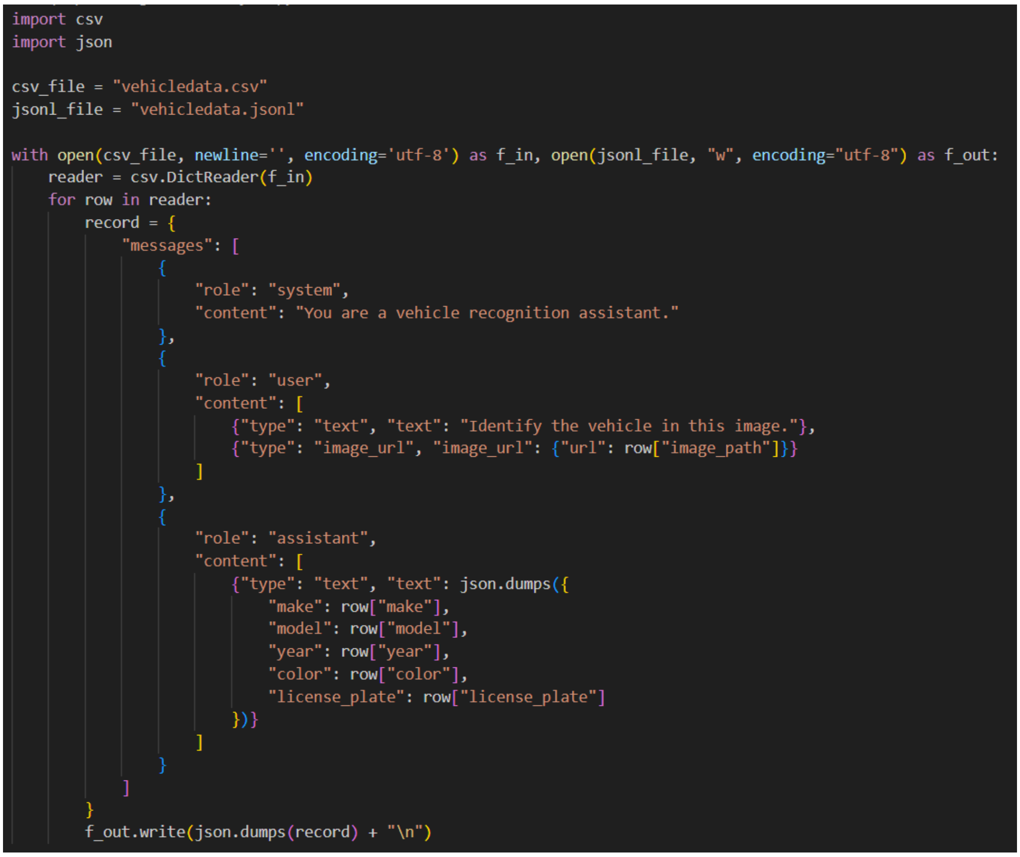
6.2.3 Vehicle Recognition

Vehicle Recognition is responsible for identifying and verifying vehicles entering or exiting the premises. This module involves two separate systems: a YOLOv8 Multi-Attribute model for detecting and classifying vehicle attributes such as brand, model, colour, and year, and a multimodal VLM-based system for analyzing and reasoning about vehicle information. The development of each system includes data preprocessing, model training, and testing to optimize performance. After training, both models are evaluated using relevant metrics to assess their accuracy and reliability, ensuring that the recognition system can effectively cross-reference vehicle information with registered data for precise identification.

Segment Anything (SAM) was first chosen as the main picture segmentation module because of its capacity to accurately extract the whole vehicle component. However, YOLOv8 was chosen to replace SAM since the project requirements include end-to-end vehicle detection and recognition. In other words, YOLOv8 provides a single framework for entire vehicle identification, training, and real-time performance as compared to SAM which is superior at component-level segmentation. It was a more realistic answer for the system's goals because of its great precision and efficiency in directly detecting vehicles in a variety of situations. This change made it possible for the multimodal vehicle detection system to grow efficiently while preserving processing speed and resilience.

6.2.3.1 Data preprocessing

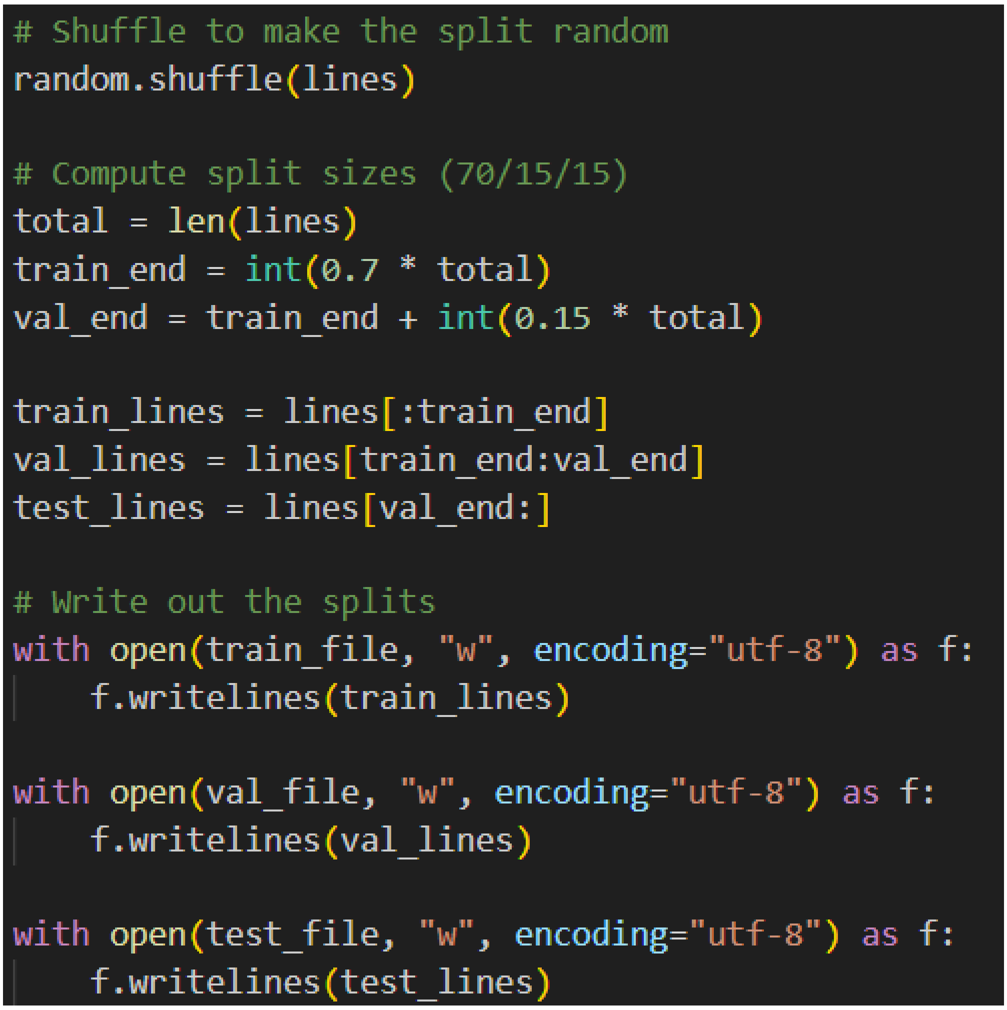
The dataset use to train both of these model are 1 image folder and 1 csv file which includes the image path, vehicle make, model, year, color and plate that act as ground truth. It contains a total of 1202 vehicle images which include different car manufacturer like Kia, Perodua, Proton, Nissan, Honda, Hyundai, Audi, Toyota, BMW and so on. Due to time constraints, each model is just cover with around 10-20 images.



```
import csv
import json

csv_file = "vehicledata.csv"
jsonl_file = "vehicledata.jsonl"

with open(csv_file, newline='', encoding='utf-8') as f_in, open(jsonl_file, "w", encoding="utf-8") as f_out:
    reader = csv.DictReader(f_in)
    for row in reader:
        record = {
            "messages": [
                {
                    "role": "system",
                    "content": "You are a vehicle recognition assistant."
                },
                {
                    "role": "user",
                    "content": [
                        {"type": "text", "text": "Identify the vehicle in this image."},
                        {"type": "image_url", "image_url": {"url": row["image_path"]}}
                    ]
                },
                {
                    "role": "assistant",
                    "content": [
                        {"type": "text", "text": json.dumps({
                            "make": row["make"],
                            "model": row["model"],
                            "year": row["year"],
                            "color": row["color"],
                            "license_plate": row["license_plate"]
                        })}
                    ]
                }
            ]
        }
        f_out.write(json.dumps(record) + "\n")
```

Figure 6.42: CSV to JSONL Conversion Code

This Python script converts a CSV file containing vehicle data into a JSONL format suitable for fine-tuning a multimodal VLM which is the GPT-4o model. For each row in the CSV, which includes columns such as image_path, make, model, year, color, and license_plate, the script creates a training record with three messages: a system message defining the assistant as a vehicle recognition assistant, a user message containing a text prompt to identify the vehicle along with the image URL, and an assistant message providing the ground truth vehicle attributes in JSON format. Each record is written as a separate line in the output JSONL file, preparing the dataset for training the model to accurately identify vehicle attributes from images.



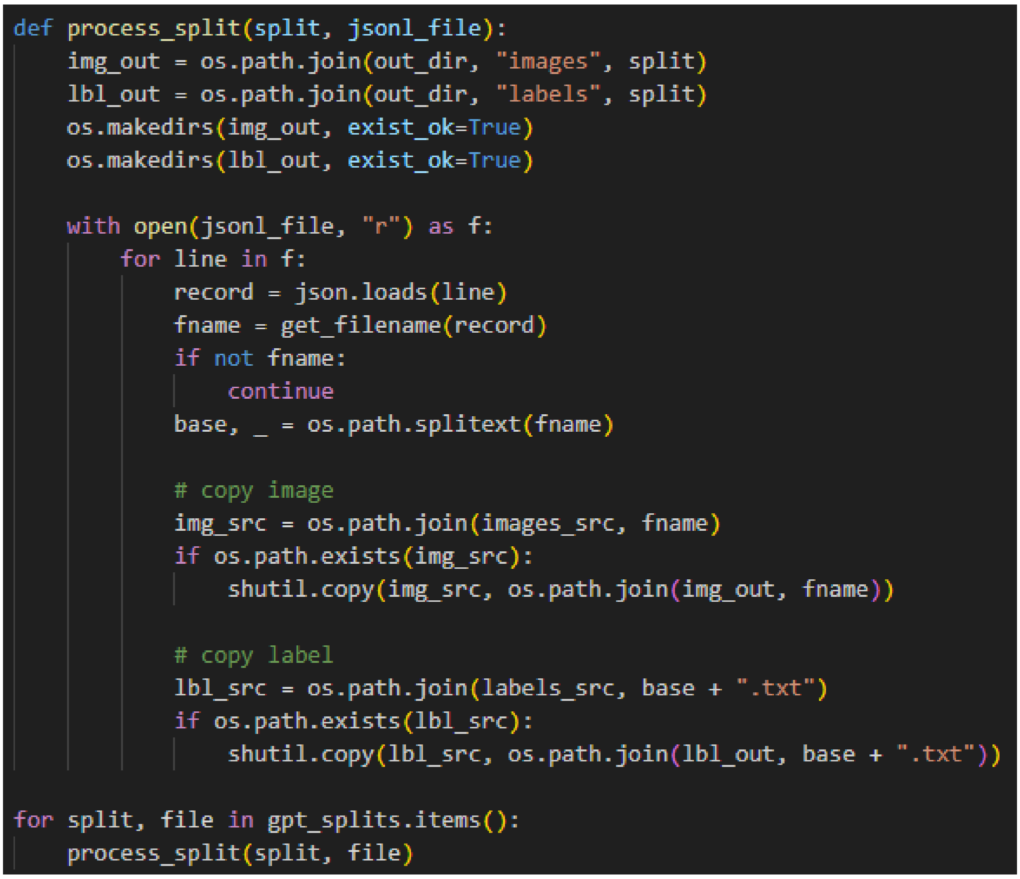
```
# Shuffle to make the split random
random.shuffle(lines)

# Compute split sizes (70/15/15)
total = len(lines)
train_end = int(0.7 * total)
val_end = train_end + int(0.15 * total)

train_lines = lines[:train_end]
val_lines = lines[train_end:val_end]
test_lines = lines[val_end:]

# Write out the splits
with open(train_file, "w", encoding="utf-8") as f:
    f.writelines(train_lines)

with open(val_file, "w", encoding="utf-8") as f:
    f.writelines(val_lines)

with open(test_file, "w", encoding="utf-8") as f:
    f.writelines(test_lines)
```

Figure 6.43: Script to split data for Fine-tuned GPT-4o model

This Python script splits a JSONL dataset into training, validation, and test sets for model fine-tuning. It reads all lines from the input JSONL file, shuffles them randomly to ensure unbiased distribution, and then divides the data into 70% training, 15% validation, and 15% test sets. Each subset is written to separate files (train.jsonl, validation.jsonl, test.jsonl).

6.2.3.1.1 Yolov8

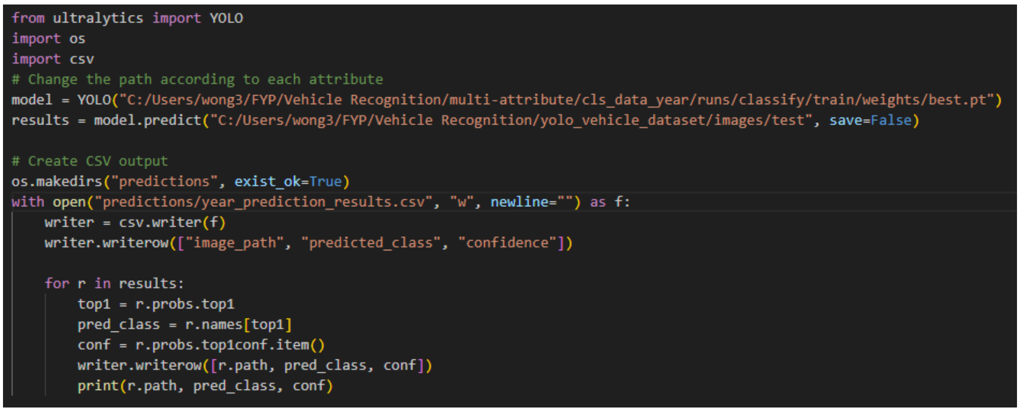
The YOLOv8 preprocessing workflow began with installing the Ultralytics library and running a pretrained YOLOv8 model on the dataset to detect vehicles using the command “yolo detect predict model=yolov8s.pt source=your_dataset/images save_txt=True”.



```python
def process_split(split, jsonl_file):
    img_out = os.path.join(out_dir, "images", split)
    lbl_out = os.path.join(out_dir, "labels", split)
    os.makedirs(img_out, exist_ok=True)
    os.makedirs(lbl_out, exist_ok=True)

    with open(jsonl_file, "r") as f:
        for line in f:
            record = json.loads(line)
            fname = get_filename(record)
            if not fname:
                continue
            base, _ = os.path.splitext(fname)

            # copy image
            img_src = os.path.join(images_src, fname)
            if os.path.exists(img_src):
                shutil.copy(img_src, os.path.join(img_out, fname))

            # copy label
            lbl_src = os.path.join(labels_src, base + ".txt")
            if os.path.exists(lbl_src):
                shutil.copy(lbl_src, os.path.join(lbl_out, base + ".txt"))

for split, file in gpt_splits.items():
    process_split(split, file)
```

Figure 6.44: Script to split data for YOLO model

After detection, irrelevant labels were removed, keeping only the vehicle class, and the largest bounding boxes in each image were extracted along with their corresponding label files. The dataset was then split to align with GPT-4o’s dataset partitioning, ensuring consistency between the two systems for training and evaluation.



```python
from ultralytics import YOLO
import os
import csv
# Change the path according to each attribute
model = YOLO("C:/Users/wong3/FYP/Vehicle Recognition/multi-attribute/cls_data_year/runs/classify/train/weights/best.pt")
results = model.predict("C:/Users/wong3/FYP/Vehicle Recognition/yolo_vehicle_dataset/images/test", save=False)

# Create CSV output
os.makedirs("predictions", exist_ok=True)
with open("predictions/year_prediction_results.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["image_path", "predicted_class", "confidence"])

    for r in results:
        top1 = r.probs.top1
        pred_class = r.names[top1]
        conf = r.probs.top1conf.item()
        writer.writerow([r.path, pred_class, conf])
        print(r.path, pred_class, conf)
```

Figure 6.45: Scripts for running testing on each attribute

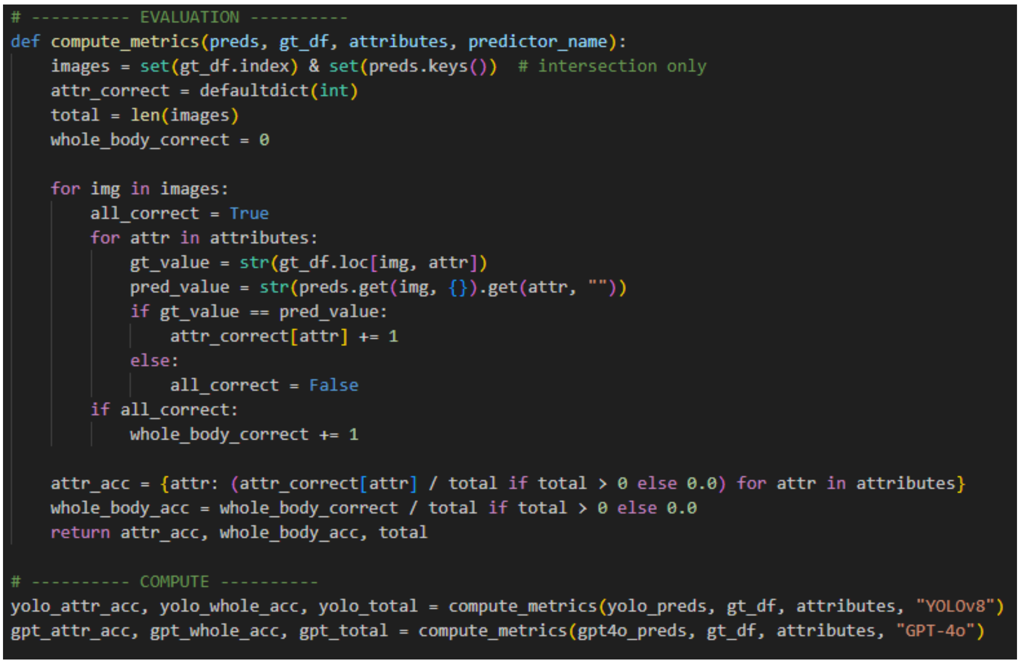
Finally, separate YOLOv8 models were trained for each attribute—color, make, model, and year. Inference tests were run on each attribute, and EasyOCR was used to extract license plates. The predictions were then compared against ground truth labels to compute the evaluation metrics.

6.2.3.1.2 GPT 4-o

The GPT-4o system utilized the same dataset with preprocessing tailored for VLM input. Text-based vehicle attributes and images were paired to create multimodal input suitable for fine-tuning the model. No bounding box extraction was needed since GPT-4o reasons over the entire image.

6.2.3.2 Comparison of Yolo-V8 + EasyOCR and GPT-4o Metric Result agaist ground truth

Once done all the testing, the code below is being run to get the metric for both model.



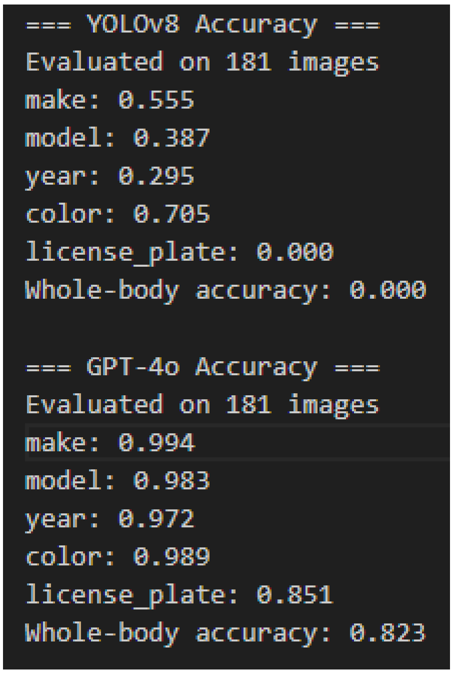
```python
# ---------- EVALUATION ----------
def compute_metrics(preds, gt_df, attributes, predictor_name):
    images = set(gt_df.index) & set(preds.keys())  # intersection only
    attr_correct = defaultdict(int)
    total = len(images)
    whole_body_correct = 0

    for img in images:
        all_correct = True
        for attr in attributes:
            gt_value = str(gt_df.loc[img, attr])
            pred_value = str(preds.get(img, {}).get(attr, ""))
            if gt_value == pred_value:
                attr_correct[attr] += 1
            else:
                all_correct = False
        if all_correct:
            whole_body_correct += 1

    attr_acc = {attr: (attr_correct[attr] / total if total > 0 else 0.0) for attr in attributes}
    whole_body_acc = whole_body_correct / total if total > 0 else 0.0
    return attr_acc, whole_body_acc, total

# ---------- COMPUTE ----------
yolo_attr_acc, yolo_whole_acc, yolo_total = compute_metrics(yolo_preds, gt_df, attributes, "YOLOv8")
gpt_attr_acc, gpt_whole_acc, gpt_total = compute_metrics(gpt4o_preds, gt_df, attributes, "GPT-4o")
```

Figure 6.46: Compute model metric code

This Python script is designed to evaluate and compare the performance of YOLOv8 + EasyOCR and GPT-4o on vehicle recognition tasks using a ground truth dataset. It begins by loading the ground truth CSV file, which contains the actual vehicle attributes, along with YOLOv8 predictions for each attribute (make, model, year, color, license plate) stored in separate CSV files, and GPT-4o predictions stored in a JSONL file. YOLOv8 predictions are mapped to their base image names, removing suffixes from cropped images, while GPT-4o predictions are indexed directly by the image names.

The script uses a compute_metrics function to calculate both attribute-level accuracy and whole-body accuracy, which measures whether all vehicle attributes were correctly predicted for an image. It compares each predicted attribute against the corresponding ground truth, counting correct predictions per attribute and determining if the entire vehicle was correctly recognized. Only images present in both the ground truth and prediction datasets are evaluated to ensure a fair comparison.

Finally, the results are written to an output file, metrics_results.txt, showing the accuracy for each attribute as well as the overall whole-body accuracy for both YOLOv8 + EasyOCR and GPT-4o. This process provides a clear comparison of the two models' capabilities, highlighting the strengths and weaknesses of YOLOv8 for attribute detection and GPT-4o for complete vehicle recognition.



```
=== YOLOv8 Accuracy ===
Evaluated on 181 images
make: 0.555
model: 0.387
year: 0.295
color: 0.705
license_plate: 0.000
Whole-body accuracy: 0.000

=== GPT-4o Accuracy ===
Evaluated on 181 images
make: 0.994
model: 0.983
year: 0.972
color: 0.989
license_plate: 0.851
Whole-body accuracy: 0.823
```

Figure 6.47: Result of metric for both model

The evaluation results show a clear performance difference between YOLOv8 + EasyOCR and GPT-4o in vehicle recognition on 181 test images.

For YOLOv8, the attribute-level accuracies are moderate to low. It correctly predicts color in 70.5% of cases, make in 55.5%, model in 38.7%, and year in 29.5%. However, it completely fails at predicting license plates (0.0%)

and consequently achieves a whole-body accuracy of 0.0%, meaning it never predicts all attributes correctly for any single image. This suggests that YOLOv8 struggles with fine-grained attributes and license plate recognition, likely because of the limited training on partial or small vehicle details and reliance on EasyOCR for plate extraction.

In contrast, GPT-4o achieves very high attribute-level accuracies with make (99.4%), model (98.3%), year (97.2%), color (98.9%), and license plate (85.1%). The whole-body accuracy is 82.3%, indicating that in most images, GPT-4o correctly predicts all vehicle attributes simultaneously. This demonstrates that GPT-4o is far more effective at holistic vehicle recognition, particularly in handling multiple attributes at once and extracting license plate information accurately.

Overall, the metrics highlight that while YOLOv8 is useful for coarse attribute detection, GPT-4o provides significantly more accurate and complete vehicle recognition, making it more reliable for systems requiring precise multi-attribute identification.

6.3 Summary

The system integrates frontend-backend functionalities with AI-based vehicle recognition. Modules for residents and security guards handle vehicles, visitors, notifications, and activity records through Supabase, ensuring validation, real-time updates, and role-specific views.

On the AI side, two recognition systems were developed. YOLOv8 + EasyOCR processed vehicle images with bounding box analysis and attribute-specific models, while GPT-4o used a fine-tuned JSONL dataset for holistic attribute recognition. Evaluation showed GPT-4o outperforms YOLOv8 in both attribute-level and whole-vehicle accuracy, highlighting the VLM-based approach's effectiveness.

Overall, the system combines robust management features with accurate multimodal vehicle recognition for real-world deployment.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

The requirement traceability matrices, use case tables, and functional tables are the main components of this chapter. It also assesses the test cases and their outcomes, as well as the unit and integration testing. In order to make sure the system satisfies both functional and user experience objectives, the chapter concludes by discussing the evaluation of User Acceptance Testing (UAT) and System Usability Scale (SUS).

7.2 Traceability between Use Cases, Functional Requirements and Test Cases

Throughout the project lifecycle, a Requirement Traceability Matrix (RTM) is a tool that helps make sure that all project requirements are recorded, monitored, and met.

Table 7. 1 : Requirement Traceability Matrix

Use Case ID	Use Case Name	Functional Requirement ID(s)	Test Case ID(s)	Test Case Description
USC001	Retrieve Vehicle Logs	SRS013 SRS014 SRS015 SRS016	TC008 TC009	Verify resident can view, search, filter, and sort vehicle logs
USC002	Manage Visitor Pass	SRS009 SRS010 SRS011	TC010 TC011 TC012	Verify resident can create, update, and delete visitor passes
USC003	Reset Password	SRS012 SRS018	TC003	Verify resident and security guard can reset their password

USC004	Login	SRS002 SRS017	TC002	Verify resident and security guard can log in with correct credentials
USC005	Register Account	SRS001	TC001	Verify resident can register a new account with all required info
USC006	Receive Notification	SRS007 SRS008	TC013	Verify residents receive notifications for vehicle entries/exits and suspicious activities
USC007	Manage Vehicle	SRS004 SRS005 SRS006	TC005 TC006 TC007	Verify residents can register, update, and delete vehicle information
USC008	Manage Profile	SRS003 SRS025	TC004	Verify residents and guards can update their profile information
USC009	Retrieve Vehicle History Log	SRS020 SRS021 SRS022 SRS023	TC008 TC009	Verify guards can search, filter, sort, and detect suspicious vehicles in history logs

USC010	Retrieve Real-Time Data	SRS019 SRS024	TC015	Verify guards receive real-time data and alerts for flagged vehicles

7.2.1 Use Case Table

A Use Case Table lists each use case together with its unique identity, major actor or actors, preconditions, main flow, alternate flows, and expected results to give an organized picture of the system's functional requirements. It makes it easier to track how users interact with the system, makes it clear what each actor is responsible for, and guarantees that all functional requirements are recorded so they can be referred to during development, testing, and validation.

Table 7. 2 : Use Case Table

ID	Use Case Name	Description
USC001	Retrieve Vehicle Logs	This use case describes the process of retrieving a list of vehicle logs owned by resident by sorting the data based on vehicle plate, vehicle make and model, event type, activity status or date by the resident.
USC002	Manage Visitor Pass	This use case describes the process of resident retrieving a list of visitor pass.
USC003	Reset Password	This use case describes the process of reset the account password.
USC004	Login	This use case describes the process of resident and security guard login the web vehicle recognition access control system.
USC005	Register Account	This use case describes the process of resident register a new account in vehicle recognition access control system.

USC006	Receive Notification	This use case describes the process of resident receiving notification in vehicle recognition access control system.
USC007	Manage Vehicle	This use case describes the process of resident manage vehicles in vehicle recognition access control system.
USC008	Manage Profile	This use case describes the process of resident and security guard manage their profiles in vehicle recognition access control system.
USC009	Retrieve Vehicle History Log	This use case describes the process of security guard search vehicle history log in vehicle recognition access control system.
USC010	Retrieve Real-Time Data	This use case describes the process of security guard retrieve real-time data and receive alerts in vehicle recognition access control system.

7.2.2 Functional Requirement Table

Functional Requirement table lists the precise features that the system needs to have. A unique ID, a description of the need, its priority or importance level, and any pertinent dependencies or constraints are usually included with each entry. This table guarantees that every system function is precisely specified, traceable, and able to be methodically checked during testing to make sure the system fulfills its intended function.

Table 7. 3 : Functional Requirement Table

Role	ID	Module	Functional Requirements
Residents	SRS001	Registration	The system shall allow residents to register an account with their personal information such as

			house number, name, email and password.
	SRS002	Login	The system shall allow residents to log in by email and password.
	SRS003	Manage Profile	The system shall allow residents to update their profile information such as name, phone number, address and profile image.
	SRS004	Manage Vehicle	The system shall allow residents to register their vehicle with details like plate number, colour, model, and manufacturer.
	SRS005		The system shall allow residents to update their registered vehicle information such as plate number, colour, and model.
	SRS006		The system shall allow residents to delete their registered vehicle.
	SRS007	Notifications	The system shall send notification to residents when their vehicle enters/exits the premises.
	SRS008	Suspicious Activity Alerts	The system shall alert residents if a suspicious event such as clone vehicle plate with different colour or model is detected.
	SRS09	Manage Visitor Pass	The system shall allow residents to generate a visitor pass with different time-limited for visitor registration.
	SRS010		The system shall allow residents to update a visitor pass for editing

			incorrect information or activate again the visitor pass.
	SRS011		The system shall allow residents to delete a registered visitor pass.
	SRS012	Reset Password	The system shall allow residents to reset their account password.
	SRS013	Vehicle Logs	The system shall allow residents to view a history of their registered vehicle’s entries/exits.
	SRS014		The system shall allow residents to search for history of their registered vehicle’s entries/exits by record ID.
	SRS015		The system shall allow residents to filter the log records by event type and suspiciousness.
	SRS016		The system shall allow residents to sort their registered vehicle’s log records by alphabetical characters in ascending and decreasing order.
Security Guards	SRS017	Login	The system shall allow security guard to log in by guard ID and password.
	SRS018	Reset Password	The system shall allow security guard to reset their account password.
	SRS019	Dashboard	The system shall display real-time data analysis from the log records of vehicle entries or exits.
	SRS020	Vehicle Logs	The system shall allow security guards to search for history of

			vehicle’s entries/exits by record ID, vehicle colour, and vehicle model.
	SRS021		The system shall allow security guards to filter the log records by record status.
	SRS022		The system shall allow security guards to sort the log records by alphabetical characters in ascending and decreasing order.
	SRS023	Suspicious Events	The system shall highlight suspicious vehicles like duplicated license plate for manual checking.
	SRS024	Real-Time Alerts	The system shall notify guards of flagged vehicles via audio/visual alerts on the dashboard.
	SRS025	Manage Profile	The system shall allow security guards to update their profile information such as name and profile image.

7.2.3 Test Cases Table of Unit Testing

The unit tests carried out for every single system module are shown in this part. Registration, login, password reset, profile management, vehicle management, visitor pass handling, alerts, and vehicle logs are just a few of the features that unit testing aims to confirm operate properly when used separately. The module or feature being tested, the actions or inputs made, the anticipated output or result, and whether the test passed or failed are all specified in each test case. Before connecting the system's core functionalities with additional modules, these tests make sure they function as intended.

Table 7. 4 : Test Case Table of Unit Testing

Test Case ID	Module / Feature	Description	Input / Action	Expected Output / Result	Pass/Fail
TC001	Registration	Verify resident can register an account	Enter name, email and password	Account created successfully, confirmation message shown	Pass
TC002	Login	Verify login functionality for resident and security guard	Enter email and password	User successfully logged in	Pass
TC003	Reset Password	Verify password reset for resident and guard	Enter new password	Password updated, confirmation message shown	Pass
TC004	Manage Profile	Verify profile update functionality	Update name, phone, address, profile image	Updated profile information is saved	Pass
TC005	Manage Vehicle	Verify vehicle registration	Enter vehicle plate, color, model, manufacturer and year	Vehicle added to the system	Pass
TC006	Manage Vehicle	Verify vehicle update	Update vehicle plate, color, model, manufacturer and year	Updated vehicle details are saved	Pass

TC007	Manage Vehicle	Verify vehicle deletion	Select vehicle to delete	Vehicle removed from the system	Pass
TC008	Vehicle Logs	Verify viewing vehicle history log	Request vehicle logs	Logs displayed correctly, sortable and filterable	Pass
TC009	Vehicle Logs	Verify filtering and sorting logs	Filter by event type, suspiciousness	Logs filtered and sorted correctly	Pass
TC010	Manage Visitor Pass	Verify visitor pass creation	Enter visitor name, phone number, vehicle's attibutes and time slot	Visitor pass generated successfully	Pass
TC011	Manage Visitor Pass	Verify visitor pass update	Update visitor info	Visitor pass updated correctly	Pass
TC012	Manage Visitor Pass	Verify visitor pass deletion	Select visitor pass to delete	Visitor pass removed	Pass
TC013	Notifications	Verify receiving notifications	Vehicle entry/exit occurs	Notification received in app	Pass
TC014	Suspicious Activity Alerts	Verify system highlights suspicious vehicles	Duplicate plate or unusual event	Alerts displayed for residents/guards	Pass
TC015	Real-Time Data Dashboard	Verify dashboard updates	Vehicle enters/exits	Dashboard shows correct counts	Pass

		vehicle entries/exits			

7.2.4 Test Cases Table of Integration Testing

The integration testing that was done to confirm how the various system modules interacted with one another is presented in this section.

Table 7. 5 : Test Cases Table of Integration Testing

Test Case ID	Modules / Features Integrated	Description	Input / Action	Expected Output / Result	Pass/Fail
ITC001	Registration + Login	Verify that a newly registered resident can log in successfully	Register a new account, then attempt login	Resident account created and login successful	Pass
ITC002	Manage Profile + Login	Verify profile updates persist across sessions	Update profile info, log out, log in again	Updated profile information retained	Pass
ITC003	Manage Visitor Pass + Notifications	Verify that creating a visitor pass triggers notifications	Create a visitor pass for a vehicle entry	Notification received in app	Pass
ITC004	Suspicious Activity Alerts + Vehicle Logs	Verify suspicious vehicles are highlighted	Duplicate plate or unusual vehicle entry	Alerts triggered and logs marked as suspicious	Pass

		in logs and alerts			

7.3 User Acceptance Test (UAT)

To make sure the system satisfies the requirements and is prepared for deployment, the User Acceptance Test (UAT) assesses it from the viewpoint of the end user. Usability, functionality, and general satisfaction feedback are gathered. Verifying that the system satisfies business requirements, performs as anticipated in real-world situations, and is suitable for operational usage is the aim of UAT.

7.3.1.1 Test Results of User Acceptance Test

The User Acceptance Test (UAT) results provide an overview of the results of every test case that end users have run to confirm that the system is prepared for deployment. The outcomes highlight any flaws or problems encountered and show whether each functionality operated as intended. With the majority of test cases passing and only minor changes needed for a few functionalities, the UAT overall verified that the system satisfies user requirements, operates dependably, and has an intuitive interface.

7.3.1.2 Resident Side

Table 7. 6 : UAT Result on Resident Side

No.	Question	Average Rating of Likert Scale (1-5)
Q1.	Can you successfully register an account using your house number, name, email, and password?	4.8
Q2.	Are you able to log in using your registered email and password?	5
Q3.	Can you reset your password successfully?	4.8
Q4.	Can you update your profile information such as name, phone number, and profile image?	4.6

Q5.	Are you able to register your vehicle with its plate number, color, year, model, and manufacturer?	4.4
Q6.	Can you update your registered vehicle’s details like plate number, color, and model?	4.6
Q7.	Can you delete your vehicle from the system?	4.4
Q8.	Do you receive a notification when your vehicle enters or exits the premises?	4.6
Q9.	Are you alerted when suspicious activity (e.g., duplicated plate with different model/color) is detected?	4.4
Q10.	Are you able to generate a visitor pass with a time limit?	4.2
Q11.	Can you view your own vehicle's entry/exit log history?	4.6
Q12.	Are you able to search vehicle logs by record ID?	4.6
Q13.	Can you filter log records by event status?	4.6
Q14.	Can you sort vehicle logs?	4.4

Based on participant ratings on a Likert scale, the User Acceptance Test (UAT) results show that the system is very efficient and easy to use. With an average rating of 4.8 for Q1, users were able to properly create accounts and log in with their registered credentials, earning a flawless score of 5. With average scores of 4.8 and 4.6, respectively, the password reset and profile update capabilities were also well evaluated, indicating that important account management tools are dependable and easy to use.

Feedback on vehicle management features was also quite positive. Updates and deletions of vehicle information were marginally higher at 4.6 and 4.4, indicating that the system facilitates easy car record maintenance. Users were able to register their automobiles with comprehensive features such license plate number, color, year, model, and manufacturer, scoring an average of 4.4.

Both alerts for suspicious activity and notifications for vehicle entry and exit performed well, scoring 4.6 and 4.4, respectively, showing that the system can deliver timely updates and uphold security awareness.

With an average rating of 4.2 for creating passes with time limits, visitor pass management had a moderately high score, indicating that while it is functional, there may be some need for improvement in terms of user convenience or clarity. Users successfully viewed, searched, filtered, and sorted their car logs, demonstrating the effectiveness and usability of data retrieval and organization. Log management functionalities worked well, with users average between 4.4 and 4.6. All things considered, the UAT shows that the system satisfies user expectations, is generally easy to use, and offers essential features for residents. For the best user experience, only minor adjustments are advised.

7.3.1.3 Security Guard Side

Table 7. 7 : UAT Result on Security Guard Site

No.	Question	Average Rating of Likert Scale (1-5)
Q1.	Can you log in with your guard ID and password?	5
Q2.	Can you reset your password successfully?	4.6
Q3.	Can you update your profile information such as name, phone number, and password?	4.6
Q4.	Does the dashboard display data of vehicle entries/exits?	4.6
Q5.	Can you search entry/exit records by various fields (e.g., record ID, vehicle model, vehicle color) ?	4.6
Q6.	Can you filter log records by various fields (e.g., record status)?	4.2
Q7.	Can you sort records as required?	4.4

Q8.	Does the system highlight suspicious vehicles like duplicated plates?	4.4
Q9.	Do you receive audio or visual alerts for flagged vehicles?	4.4

The security guard User Acceptance Test (UAT) results demonstrate that the system operates efficiently and satisfies operational requirements. A flawless score of five was obtained for logging in using a guard ID and password, indicating that authentication is simple and trustworthy. Both the profile update and password reset tools received ratings of 4.6, demonstrating how user-friendly and effective account management features are.

The dashboard's ability to show car entry and leave statistics received a score of 4.6 as well, demonstrating how easy and transparent real-time monitoring is. The user experience could be improved with small usability enhancements. Record management capabilities, such as searching by fields like record ID, car model, or color, obtained similarly good scores (4.6), whereas filtering records received a slightly lower score (4.2). Data organization is typically effective, as seen by the 4.4 score for sorting records as needed.

Although there may be some space for improvement in alert visibility or clarity, security features like highlighting suspicious vehicles and sending audio/visual alerts for flagged vehicles received ratings of 4.4, demonstrating that the system effectively supports situational awareness and quick response. All things considered, the UAT verifies that the system is mainly user-friendly, dependable, and useful for security guard operations.

7.4 System Usability Test

A standardized instrument for assessing the general usability of the vehicle recognition access control system is the System Usability Scale (SUS). Ten Likert-scale questions are used to gather user input on usability, effectiveness, learnability, and satisfaction. Both the resident and security guard interfaces' strengths and potential areas for development can be determined with the use of the SUS results, which offer a quantitative assessment of user experience. By ensuring that the system is both functional and easy to use, this evaluation promotes successful adoption and day-to-day operations.

7.4.1 System Usability Scale Template

The template of SUS 10 questions used in this project was shown below:

1. I think I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think I would need the support of a technical person to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

7.4.2 System Usability Testing Result

The System Usability Scale (SUS) is a popular, efficient, and trustworthy instrument for evaluating how user-friendly a system, product, or service is. It is a 10-question survey that alternates between positive and negative statements on a 5-point Likert scale from "Strongly Disagree" to "Strongly Agree." The answers are combined into a single score between 0 and 100, with 68 serving as the standard for average usability. The higher the SUS scores, the better the usability.

Table 7. 8 : SUS Result Table

Tester	Score for each item										Total
	1	2	3	4	5	6	7	8	9	10	
Resident Side											
Tester 1	5	1	5	1	4	1	5	1	5	1	97.5
Tester 2	4	3	3	4	4	2	4	2	4	3	62.5
Tester 3	4	1	4	1	5	1	4	1	5	2	90.0
Tester 4	4	1	5	2	5	1	5	1	4	2	90.0

Tester 5	5	2	4	2	4	2	3	2	4	2	75.0
Security Guard Side											
Tester 1	5	1	5	1	4	1	4	1	5	1	95.0
Tester 2	5	1	5	2	4	2	5	1	4	2	87.5
Tester 3	4	3	4	3	4	3	5	4	5	5	60.0
Tester 4	5	1	5	1	5	1	5	1	5	2	97.5
Tester 5	4	2	4	2	4	2	4	2	4	2	75.0
Average SUS Score											83.0

Five testers from each group participated in the SUS evaluation, which was carried out on the resident and security guard sides of the vehicle recognition access control system. Although one tester reported a lower score of 62.5, indicating some small usability concerns or individual difficulty with some system elements, the resident side's SUS scores varied from 62.5 to 97.5, indicating a generally good level of usability. Most resident testers had scores above 75, indicating that the system is generally easy to use and straightforward for completing essential functions including notifications, vehicle management, and registration.

The SUS scores for security guards varied from 60.0 to 97.5, indicating that the dashboard, record search, and alert features were easy to use for the majority of users. The lowest score of 60.0 might draw attention to certain locations where vehicle log filtering or sorting needs to be improved. In spite of this, the system received great ratings from most security guard testers, demonstrating its excellent usability for tracking suspicious activity and real-time vehicle entries and exits.

All testers' average SUS score was 83.0, which is regarded as a good usability rating. This implies that both residents and security personnel find the system to be well-designed, effective, and satisfactory. Although the system exhibits excellent general user acceptance and operational efficacy, minor improvements should be explored to improve learnability and consistency, especially in more difficult aspects like record filtering or guest pass administration.

7.5 Summary

This chapter provided a thorough analysis of the vehicle recognition access control system, including testing, functional requirements, requirement traceability, and user input. All system features were correctly validated thanks to the requirement traceability matrix, which connected system use cases with functional requirements and matching test cases. In order to verify that essential features including registration, login, vehicle management, notifications, and suspicious behavior warnings worked as intended, unit and integration testing tables recorded the validation of individual modules and their interactions.

Strong user satisfaction was demonstrated by the User Acceptance Test (UAT) findings which showed that residents and security guards were able to successfully complete essential tasks such account administration, car registration, log retrieval, and real-time monitoring. All of these tasks received high Likert scale ratings. With an overall average score of 83.0, the System Usability Scale (SUS) study also showed good usability, suggesting that the system was easy to use and efficient for the security guards' and residents' operating requirements. When taken as a whole, these assessments verify that the system satisfies its functional specifications, is easy to use, and is prepared for implementation in practical situations.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusion

The planning, design, development, and testing stages of the Vehicle Recognition Access Control System project took six months to complete. By combining GPT-4o for adaptive visual-language-based recognition and YOLOv8 for multi-attribute recognition, the project effectively overcame the drawbacks of traditional vehicle recognition systems. In addition to receiving real-time notifications and alerts for suspicious activity, the system enables residents and security personnel to effectively manage accounts, vehicles, and visitor cards. Data preprocessing, model training, inference, and performance evaluation were all part of the iterative cycles of coding, training, and testing that were used to construct YOLOv8 and GPT-4o. High functionality and usability were demonstrated by the test results, with GPT-4o exhibiting higher recognition accuracy across all aspects and YOLOv8 performing well for some attributes but having trouble with whole-body and license plate identification.

8.2 Objective Fulfillment

A thorough literature review and analysis of current access control solutions allowed for the accomplishment of the first goal, which was to look into the main drawbacks of traditional vehicle recognition systems. The study emphasized issues such poor handling of edge cases, limited attribute identification, and imprecise recognition in complex scenarios.

Using EasyOCR for license plate extraction and models for car make, model, color, and year, the second goal—creating a multi-attribute vehicle recognition system with YOLOv8—was achieved. For residents and security personnel, YOLOv8 successfully identified and categorized vehicle attributes.

The third goal was accomplished by fine-tuning a visual language model (GPT-4o) using annotated vehicle data in JSONL format in order to integrate it for adaptive recognition and edge case management. GPT-4o proved its strength in adaptive recognition settings by exhibiting high accuracy across all aspects, including whole-vehicle and license plate identification. The system is easy to use, efficient, and well-liked by both residents and security guards, according to user acceptance testing and system usability review.

8.3 Limitations

The Vehicle Recognition Access Control System was developed successfully, although during the project, a number of drawbacks were found. First, there is currently a limit of one car registration per tourist which may limit flexibility in situations when a visitor has more than one vehicle. Secondly, because there isn't a specific admin interface for managing guard accounts, the administrator must install security guards manually, which could limit scalability and administrative simplicity. Lastly, the requirement that residents and security personnel read all system messages prior to signing out may have an impact on usability and efficiency, particularly in busy settings where prompt access and message handling are crucial.

8.4 Recommendations for future work

Potential areas for further investigation, development or application of the current project or study are highlighted in this part on recommendations for future work. It points out shortcomings that might be fixed, provides improvements to methods and makes recommendations for new features, lines of inquiry, or technological advancements that could be used in the future. By guaranteeing continuity and expansion beyond the current work's purview, this section aids in directing future endeavors.

Table 8. 1 : Recommendations for future work

No	Recommendation	Description
1	Improved YOLOv8 Accuracy	Enhance YOLOv8 performance by increasing dataset size, including partial or small vehicles, and refining bounding box annotations to improve recognition accuracy for challenging scenarios.
2	Enhanced GPT-4o Fine-tuning	Expand the GPT-4o dataset with more diverse vehicle examples and attributes to improve

		adaptive recognition for edge cases and rare vehicle types.
3	Expanded User Testing	Conduct larger-scale usability and acceptance testing with more diverse residents and security personnel to validate system reliability and uncover additional improvements.
4	Real-Time System Optimization	Explore edge computing or server optimization to reduce inference latency for real-time recognition, alerts, and notifications, enhancing overall responsiveness.
5	Mobile-Based Application	Develop a mobile version of the system to allow residents and security guards to access features conveniently on smartphones and tablets. This can improve usability, real-time monitoring, and overall system accessibility.

REFERENCES

Amirkhani, A. and Barshooi, A.H. (2023) 'DeepCar 5.0: Vehicle Make and Model Recognition Under Challenging Conditions', *IEEE Transactions on Intelligent Transportation Systems*, 24(1), pp. 541–553. Available at: https://doi.org/10.1109/TITS.2022.3212921.

Bakshi, A. *et al.* (2023a) 'ALPR - An Intelligent Approach Towards Detection and Recognition of License Plates in Uncontrolled Environments', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Science and Business Media Deutschland GmbH, pp. 253–269. Available at: https://doi.org/10.1007/978-3-031-24848-1_18.

Bakshi, A. *et al.* (2023b) 'ALPR - An Intelligent Approach Towards Detection and Recognition of License Plates in Uncontrolled Environments', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Science and Business Media Deutschland GmbH, pp. 253–269. Available at: https://doi.org/10.1007/978-3-031-24848-1_18.

Cayetano, F. (2024) *Vehicle Access Control: 4 Systems and Trends*. Available at: https://butterflymx.com/blog/vehicle-access-control/ (Accessed: 29 April 2025).

Dave, B. and Cole, S. (2024) *What is Loss Function? | IBM*. Available at: https://www.ibm.com/think/topics/loss-function (Accessed: 13 April 2025).

Ferrer Josep (2024) *How Transformers Work: A Detailed Exploration of Transformer Architecture | DataCamp*. Available at: https://www.datacamp.com/tutorial/how-transformers-work (Accessed: 30 March 2025).

Gao, P. *et al.* (2023) 'LLaMA-Adapter V2: Parameter-Efficient Visual Instruction Model'. Available at: http://arxiv.org/abs/2304.15010.

Hu, M. *et al.* (2023) 'Vehicle color recognition based on smooth modulation neural network with multi-scale feature fusion', *Frontiers of Computer Science*, 17(3). Available at: https://doi.org/10.1007/s11704-022-1389-x.

Kirillov, A. *et al.* (no date) *Segment Anything*. Available at: https://segment-anything.com.

Lai, Z. *et al.* (no date) *CLIPath: Fine-tune CLIP with Visual Feature Fusion for Pathology Image Analysis Towards Minimizing Data Collection Efforts*.

Larman, C. and Vodde, B. (no date) *Large-Scale Scrum*. Available at: https://less.works.

Lee, B. (2024) *Cloned licence plate almost landed man in hot water | The Star*. Available at: https://www.thestar.com.my/news/nation/2024/12/30/cloned-licence-plate-almost-landed-man-in-hot-water (Accessed: 29 April 2025).

Li, B. *et al.* (2024) 'Generative Visual Instruction Tuning'. Available at: http://arxiv.org/abs/2408.03326 (Accessed: 2 May 2025).

Mahalakshmi, M. and Sundararajan, D.M. (2008) *International Journal of Emerging Technology and Advanced Engineering Traditional SDLC Vs Scrum Methodology-A Comparative Study*, *Certified Journal*. Available at: www.ijetae.com.

MingxingTan, Ruoming Pang and Quoc V. Le (no date) *EfficientDet: Scalable and Efficient Object Detection*. Available at: https://github.com/google/.

Mo, S., Xia, J. and Markevych, I. (2023) 'CAVL: Learning Contrastive and Adaptive Representations of Vision and Language'. Available at: https://arxiv.org/pdf/2304.04399 (Accessed: 2 May 2025).

Murugaiyan, D. (2012) 'International Journal of Information Technology and Business Management WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC', 2(1). Available at: www.jitbm.com.

Pargaonkar, S. (2023) 'A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering', *International Journal of Scientific and Research Publications*, 13(8), pp. 120–124. Available at: https://doi.org/10.29322/ijsrp.13.08.2023.p14015.

Paula (no date) *SAFe Scrum Master Roles and Responsibilities - Duties*. Available at: https://premieragile.com/safe-scrum-master-roles-and-responsibilities/ (Accessed: 27 April 2025).

Peng, B. *et al.* (2023) 'Instruction Tuning with GPT-4'. Available at: http://arxiv.org/abs/2304.03277.

Poudel, R.P.K., Liwicki, S. and Cipolla, R. (2019) 'Fast-SCNN: Fast Semantic Segmentation Network'. Available at: http://arxiv.org/abs/1902.04502.

Red Ideas (2025) *JaGaApp - JaGaSolution*. Available at: https://jagasolution.com/jagaapp/ (Accessed: 2 May 2025).

Sinha, A. and Das, P. (2021) 'Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry', in *2021 5th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2021*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/IEMENTech53263.2021.9614779.

Slawek-Polczynska, A. (2020) *Is Agile always the best solution for software development projects? - SolDevelo*. Available at: https://soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/ (Accessed: 30 April 2025).

Stanford CRFM (no date). Available at: https://crfm.stanford.edu/2023/03/13/alpaca.html?utm_source=chatgpt.com (Accessed: 2 May 2025).

TimeTec (2025) *Property Management Ecosystem | iNeighbour*. Available at: https://www.i-neighbour.com/ (Accessed: 2 May 2025).

TimeTec Cloud (2025) *TimeTec VMS*. Available at: https://www.timetecvms.com/ (Accessed: 2 May 2025).

Ultralytics Inc (no date) *Anchor-Based Detectors*. Available at: https://www.ultralytics.com/glossary/anchor-based-detectors (Accessed: 29 March 2025).

VISITORZ TECH PRIVATE LIMITED (no date) *Visitorz | Touchless Visitor Management System*. Available at: https://visitorz.in/ (Accessed: 2 May 2025).

Wang, Y. *et al.* (2022) 'Self-Instruct: Aligning Language Models with Self-Generated Instructions', *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1, pp. 13484–13508. Available at: https://doi.org/10.18653/v1/2023.acl-long.754.

Wei, C. *et al.* (2024) 'Efficient license plate recognition in unconstrained scenarios', *Journal of Visual Communication and Image Representation*, 104. Available at: https://doi.org/10.1016/j.jvcir.2024.104314.

Wu, T., Feng, A. and Zhang, Q. (2024) ‘Engineering Vehicle Object Segmentation Algorithm Based on Improved YOLOv8x-seg’, in *2024 5th International Conference on Intelligent Computing and Human-Computer Interaction, ICHCI 2024*. Institute of Electrical and Electronics Engineers Inc., pp. 47–52. Available at: https://doi.org/10.1109/ICHCI63580.2024.10808149.

Zhou, X., Wang, D. and Krähenbühl, P. (2019) ‘Objects as Points’. Available at: http://arxiv.org/abs/1904.07850.

Zoinla (2019) *MyTaman - Upgrade your taman security today!* Available at: https://hello.mytaman.com/ (Accessed: 2 May 2025).