INDOOR NAVIGATION FOR VISUALLY IMPAIRED

By

Ng Wei Yu

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

June 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts. Dr. Tan Hung Khoon has given me this incredible opportunity to engage in this indoor navigation project. Thank you for always guiding me and giving me so many ideas for the project. A million thanks to you.

ABSTRACT

The increasing prevalence of visual impairments globally has highlighted the importance of effective assistive technologies to enable independent navigation for the visually impaired in indoor environments. Traditional indoor navigation systems often suffer from high costs, signal interference, and reliance on pre-installed infrastructure, making them inaccessible to many. This project proposes a cost-effective and infrastructure-free indoor navigation system that leverages a smartphone's camera and advanced computer vision. The system's core is a novel localization approach that uses a DINOv2 vision transformer to generate robust semantic embeddings from real-time images. For superior accuracy, initial embedding comparisons are verified by a Vision-Language Model (VLM), which provides contextual understanding of the scene. To handle dynamic environments, the system integrates YOLOv8-seg and Stable Diffusion 2.0 to detect and remove people from the camera feed, ensuring reliable performance. Implemented as a Flutter mobile application with a high-performance FastAPI and Supabase backend, the system provides users with step-by-step guidance along pre-recorded visual routes, delivering clear instructions via audio feedback and a hands-free voice assistant. By integrating state-of-the-art deep learning models into an accessible platform, this project addresses the limitations of existing solutions and contributes meaningfully to advancing assistive technology for the visually impaired community.

Area of Study (Maximum 2): Artificial Intelligence, Computer Vision

Keywords (Maximum 5): Indoor Navigation, Visual Place Recognition, Mobile Application, Visually Impaired, Embedding Comparison

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ViNT	Visual Navigation Transformer
CNN	Convolution Neural Network
GPS	Global Positioning System
IMU	Inertial Measurement Unit
AR	Augmented Reality
PDR	Pedestrian Dead Reckoning
IR	Infrared red
RFID	Radio Frequency Identification
NFC	Near Field Communication
BLE	Bluetooth Low Energy

CHAPTER 1 INTRODUCTION

This chapter presents the project's background information, problem statement, motivation, and contributions to the indoor navigation field.

According to data from the World Health Organization (WHO) in October 2018, there is an estimated global population of almost 1.3 billion individuals with some vision impairment, with 36 million completely blind [1]. It is predicted that the number of blind individuals will reach 38.5 million by 2020 and 115 million by 2050 [2]. Undoubtedly, the demand for navigation guidance will grow in the coming years. Individuals with vision loss face significant challenges in independently performing daily tasks, such as reading, driving, walking, etc. Moreover, they are at higher risk of accidents in both indoor and outdoor environments, such as collisions, falls, getting lost, and so on. Consequently, there is a pressing need for assistive devices that are low-cost, user-friendly, and accurate [11].

Nowadays, indoor navigation offers many benefits to users. It speeds up finding locations within large or complex indoor spaces, such as airports, malls, or office buildings and helps to save a lot of time and energy. For individuals with visual impairments, it helps to prevent accidents and injuries by guiding users around obstacles, stairs, and hazards. Furthermore, the system also provides audio feedback to help them navigate indoor spaces independently.

However, indoor navigation presents some challenges compared to outdoor navigation. The most critical and challenging task in an indoor navigation system is estimating or localising the user's location in real-time [3]. Global Positioning Systems (GPS) track the user's location in outdoor environments. However, GPS is inefficient indoors due to non-line-of-sight issues [4]. It is hard to ensure precise location tracking due to factors like signal interference, structural obstacles, and variations in indoor environments. In addition, the computational cost is another challenging issue, especially for large-scale buildings.

There are three significant methods of indoor navigation systems such as computer vision-based methods, wireless technologies, and dead reckoning technologies [11]. Computer vision methods can be categorised into markerless and marker-based methods [5]. Fiducial markers are visual representations that can be detected and identified by a video camera. Diverse toolkits [6] [7] are utilised to create and identify AR markers. Using sensors such as accelerometers, compasses, gyroscopes, and magnetometers plays a crucial role in implementing dead reckoning systems. Various alternative solutions for indoor navigation encompass wireless methodologies like NFC [17], infrared (IR) [8], radio frequency identification RFID [9], Bluetooth low energy BLE beacons [10], and Wifi [11].

1.1 Problem Statement and Motivation

High Costs and Maintenance of Traditional Methods

Traditional indoor navigation systems, such as Bluetooth beacons, are often expensive to deploy and maintain. These systems require significant investment in hardware and regular upkeep to ensure they function correctly. The high cost and maintenance burden can limit accessibility and scalability, particularly for extensive or budget-constrained facilities. Moreover, frequent maintenance and battery replacement can disrupt service and affect user experience, making it challenging for organisations to justify the ongoing expenses.

Signal Interference

Indoor navigation systems relying on signal-based technologies like Bluetooth or Wi-Fi often face signal strength and reliability challenges. Signal interference from building materials, attenuation through walls, and multi-path effects can lead to inconsistent or inaccurate location tracking. Users may experience incorrect positioning and inefficient routing, leading to frustration and extended time navigating indoor spaces.

Lack of Effective Tactile and Auditory Feedback

Many indoor navigation systems do not provide sufficient tactile or auditory feedback, which is essential for visually impaired users to navigate effectively. These systems often rely on visual cues that are inaccessible to users with visual impairments. This lack of adequate feedback can lead to difficulties understanding their surroundings, increased risk of accidents, and a decreased ability to navigate indoor environments independently.

Complex Setup of Devices

Existing indoor navigation devices for the visually impaired often involve complex setups and require specialised and expensive hardware. The need for installation software or tools and complicated procedures can discourage individuals from using the technology.

The thesis aims to propose innovative solutions for enhancing indoor navigation systems by addressing critical issues related to high costs, signal interference, ineffective feedback, and complex setups. By developing more cost-effective and low-maintenance navigation solutions, this project seeks to reduce financial constraints to make them more accessible and sustainable for many facilities and users. Besides, the project aims to enhance signal reliability and consistency to ensure more accurate and efficient navigation. Furthermore, this research is motivated to improve navigation technologies to provide better guidance and support, enabling safer and more independent navigation for visually impaired individuals.

1.2 Project Scope

The project's scope includes developing a smartphone-based indoor navigation system to assist blind individuals in navigating unfamiliar environments. The system leverages an embedding model to perform localisation using only the smartphone camera, without relying on external markers. A mobile application compatible with Android platforms will be developed, capable of real-time place recognition by matching camera images with a pre-recorded embedding database. An accessible user interface will be implemented to provide clear navigation instructions through audio cues, compass feedback, and vibration alerts. The application will also determine the relative direction (front, left, right, behind) based on the user’s facing orientation using the smartphone’s compass sensor.

1.3 Project Objectives

The main objective of this project is to develop a smartphone-based indoor navigation system that provides precise and real-time guidance for blind and visually impaired users. The system uses only a smartphone camera and built-in sensors to help users navigate unfamiliar indoor environments safely and independently.

Sub-Objectives:

- To design a visual localization system that accurately determines the user’s position using real-time camera input and stored image embeddings.
- To build a route-based navigation system that guides users step-by-step using pre-recorded visual routes, compass data, and real-time scene matching.
- To implement a user-friendly mobile interface that delivers clear and accessible navigation feedback through voice prompts, compass orientation, and vibration cues.
- To ensure the system functions effectively without relying on additional sensors or external infrastructure, making it portable and scalable for real-world indoor environments.

1.4 Contributions

This project introduces a user-friendly indoor navigation system with a clean and intuitive interface to ensure all users and visually impaired individuals can easily operate it. Furthermore, this project will improve tactile and auditory feedback mechanisms explicitly tailored for visually impaired users. Also, the project offers new solutions to overcome signal interference and reliability issues in indoor navigation systems. Improving signal processing techniques enhances the accuracy and consistency of location tracking. Finally, this project will develop an affordable and cheap application suitable for indoor environments.

CHAPTER 2: Literature Review

2.1 Wireless Networking Method

Satellite GPS signals, Near Field Communication (NFC), Infrared (IR), Radio Frequency Identification (RFID), and Bluetooth/WiFi are all examples of wireless networking technologies. Several researchers have employed wireless approaches to aid people in navigating indoor spaces. Their work involves deploying networking infrastructures such as access points, beacons, and other sensors. Bluetooth, WiFi, and RFID sensors are utilised to guide visually impaired people indoors using auditory instructions [12,13]. Another option is an NFC-based indoor navigation system [14], which maintains map data on a server. Touching an NFC tag with a smartphone retrieves location information. The key restriction of this method is that the user must manually locate and contact the next NFC tag on the path.

Infrared and magnetic sensors are also employed to help users navigate. These sensors detect a unique location-based code on the ceiling [15]. The map is downloaded to a handheld infrared device via Bluetooth when the user enters a building. It employs voice commands to obtain location data from the system. The method is based on a huge IR sensor infrastructure deployed within a building. However, users struggle with their movements because they must carry large IR gadgets when walking in the environment. Similarly, GPS signals track users in vast indoor areas such as single-story homes, schools, and tall skyscrapers. GPS can cause inaccurate tracking if nonstandard building materials are used, a low signal-to-noise ratio, or poor satellite signal reception [16].

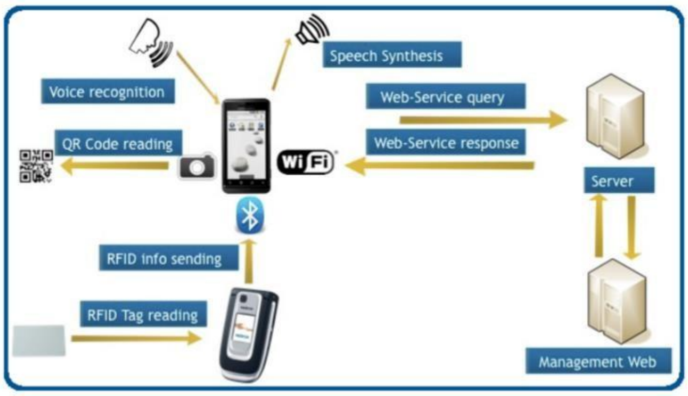


Figure 2.1.1 BlindShopping Distributed Architecture [17]

The "BlindShopping" platform assists visually impaired individuals with indoor navigation and product recognition during shopping. It utilizes a map-based system that identifies RFID tags embedded in specific supermarket areas to guide users and locate products. The platform supports the shopping process, providing navigation and product recognition features. Based on smartphone input, the navigation system uses audio messages to direct users to the passage of their desired products. The product recognition function allows the scanning of QR codes or UPC barcodes attached to products.

As shown in Figure 2.1.1, the system operates on a smartphone with a Java ME application that reads RFID tags and communicates with an Android app via Bluetooth. The system includes several key components: a smartphone with QR code scanning and voice recognition, an RFID reader connected via Bluetooth, and a server connection through Wi-Fi for data retrieval and voice synthesis. Users can issue commands by drawing gestures or using voice commands to navigate within a supermarket, with the smartphone guiding them to specific product categories by scanning QR codes or reading RFID tags on the floor. This setup enables real-time assistance, allowing blind individuals to shop independently by providing location-based guidance and product information.

2.2 Dead Reckoning Method

The Deck Reckoning method utilizes data from the accelerometer, gyroscope, and magnetometer to compute the user's position. Dead reckoning techniques are used for user localization and navigation by calculating user steps and direction toward a particular destination using the smartphone's accelerometer and compass [18]. One of the dead reckoning-based methods is MEMS inertial measurement unit (IMU) based pedestrian dead reckoning (PDR).

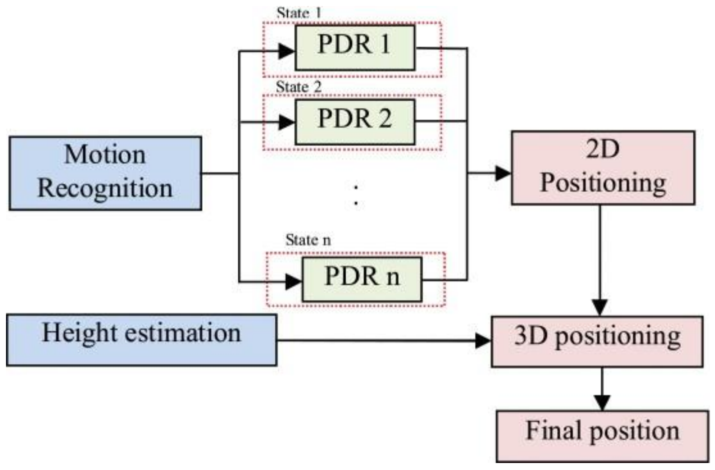


Figure 2.2.1 System Diagram of IMU-based PDR system [19]

Figure 2.2.1 illustrates that this system first uses a motion detection algorithm utilizing smartphone sensors to ascertain if the user is engaged in walking or other forms of movement. Subsequently, a range of PDR techniques are utilized to examine the identified motion patterns, including the duration of each step and the direction taken during each step. The smartphone's barometric pressure sensor determines the user's height. This information is then merged with the 2D position obtained from motion recognition-based PDR (Pedestrian Dead Reckoning) to calculate the user's 3D position.

Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are suitable for motion recognition tasks. However, ANN is considered more user-friendly in terms of implementation than SVM. Furthermore, Artificial Neural Networks (ANN) are highly efficient in extracting features and fine-tuning parameters. The positioning algorithm of the PDR consists of three components: step detection, step length estimate, and heading estimation.

The accelerometer produces the Root Mean Square (RMS) value to detect steps and creates a sinusoidal wave using a moving averaging filter to improve the signal. Subsequently, peak detection is utilized to ascertain the peak in the sinusoidal wave for step detection. Various techniques are employed to estimate step length by analyzing the detected movement. The Kalman Filter (KF) is utilized to estimate certain motion states, while Artificial Neural Networks (ANN) are employed to estimate others in heading estimation. The Kalman Filter is regularly updated with gyroscope data, and the measurements are enhanced using a magnetometer.

IMU-based Pedestrian Dead Reckoning (PDR) systems offer several advantages and limitations. One of their primary strengths is their independence from external infrastructure, making them highly suitable for indoor environments where GPS and other signals may be weak or unavailable. These systems are also easily accessible and portable, often embedded in smartphones, allowing real-time tracking of a pedestrian's movements. Also, they are cheaper due to reliance on phone sensors.

However, these systems also face significant challenges. Users must keep the smartphone in a consistent orientation to avoid interference with motion detection. Additionally, the motion classifier's accuracy decreases during transitions between different types of movement, which can impact performance during rapid movement changes. The precision of height sensors also depends on the availability and accuracy of external data sources. Furthermore, this method depends on smartphone sensors, potentially limiting its applicability to devices with compatible hardware.

2.3 Visual-Based Method

Visual-based applications are available in marker-based or markerless indoor navigation and positioning methods. Plain markers are fixed on the indoor environment's floor, ceiling, or walls. The markers are then captured using a video stream from a camera device. Markerless techniques gather features from a video stream, like corners, walls, and objects, and compute paths based on those features.

2.3.1 Marker-Based Method

The concept behind marker-based methods is straightforward: placing markers in key locations. Then, the system can use them as reference points to determine the user’s position and orientation. This makes it possible to provide users with real-time directions and navigation to a specific destination. Toolkits like ARToolKit, ArUco, AprilTag, and Vuforia add functionality to this approach. Marker-based tracking techniques are more robust and work under different light conditions than markerless [20]. Each toolkit offers different benefits depending on the application-specific needs. For instance, Vuforia has high performance, but it is not open source, while ArUco and AprilTag are open source and cost-effective but require more customization [21]. Khan et al. [22] employed ARToolkit markers for indoor navigation using only cameras. It makes this method cost-effective and easy to implement in various indoor environments. However, the building administrators are not satisfied with the marker placement. In addition, the difficulty of handling a smartphone while navigating is also a problem. Future improvements could include markerless methods or hidden markers. A 2011 study [23] that used a laptop camera for ARToolKit marker detection also faced laptop portability and adaptability issues. This method provides directions via AR overlays and audio but lacks flexibility and generalisation. Furthermore, Kim et al. [24] developed an application using ARToolKit markers to transmit image sequences for marker detection and user positioning but did not include the shortest path mechanism. Their results included an analysis of marker detection, image matching, and location recognition but lacked a user study evaluation. Another study [25] used a laptop with ARToolKit markers, an inertial tracker, and a wrist-mounted touch screen for guidance. Although it provided accurate results under normal lighting, it was bulky and required manual map editing.

One of the key advantages of marker-based methods is flexibility and simplicity. More markers can be added and their placement can be adjusted. Also, the markers themselves can be designed in different sizes and shapes. Furthermore, it is a low-cost implementation because it requires minimal hardware, including the markers and a camera-equipped device. However, there are some limitations, such as the placement of markers, the effect of light intensity and its direction that can cause issues, and the occlusion of the markers. Hidden markers or markless methods could be the solution to this issue.

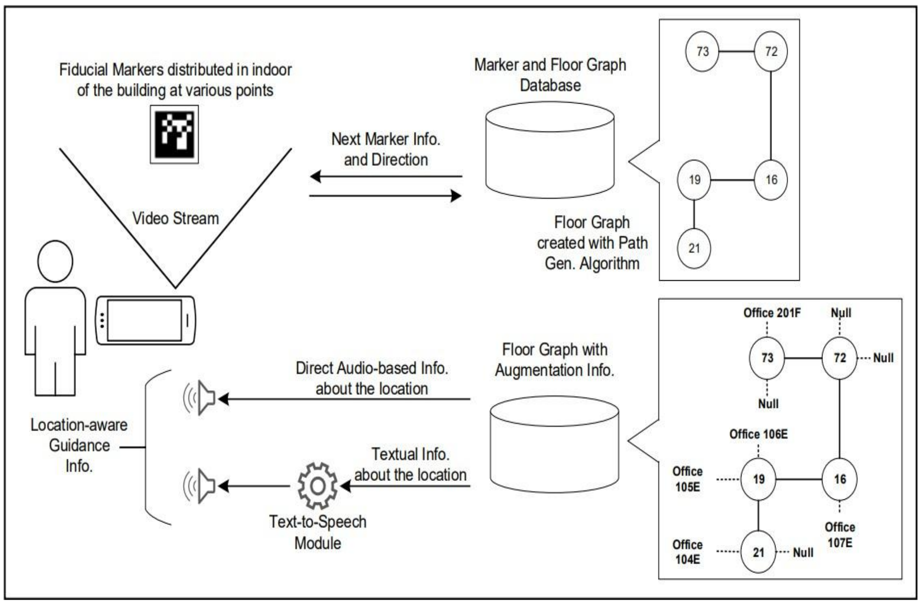

Fiducial Markers distributed in indoor of the building at various points
Video Stream
Next Marker Info. and Direction
Marker and Floor Graph Database
Floor Graph created with Path Gen. Algorithm
73
72
19
16
21
Location-aware Guidance Info.
Direct Audio-based Info. about the location
Textual Info. about the location
Text-to-Speech Module
Floor Graph with Augmentation Info.
Office 201F
Null
Office 106E
Office 105E
Office 104E
Office 107E

Figure 2.3.1.1: Block Diagram of the Indoor Navigation Using Fiducial Marker [22]

The block diagram illustrates an indoor navigation system that leverages fiducial markers, likely similar to ArUco, to help users navigate through a building. These markers are strategically placed at various points within the building, serving as reference points that the system detects through a video stream captured by a user's device, such as a smartphone. The system uses this video stream to recognise the user's location by identifying the markers. Information about all markers and the building's layout is stored in a marker and floor graph database, which the system queries to determine the next marker in the user's path and the direction they should move toward.

Once the next marker and direction are identified, the system provides the user with location-aware guidance, including visual instructions on the device's screen or audio directions. The system also offers direct audio-based information about the user's location to enhance the navigation experience. A more detailed floor graph augmented with extra information can provide further contextual data or navigational cues. The text-to-speech module is crucial in converting textual details on the location into spoken words, ensuring the navigation experience is hands-free and accessible. Throughout the navigation process, the system continuously updates the user with the necessary information to guide them effectively through the indoor environment, integrating real-time video processing, marker data, and audio guidance for a comprehensive and user-friendly navigation experience.

2.3.2 Markerless Based Method

Markerless-based methods in indoor navigation can trace the user's location and navigate the user without fiducial markers.

Firstly, Convolutional Neural Networks have revolutionized many areas of computer vision, including indoor navigation. CNNs can be used for various tasks within navigation systems, such as feature extraction, pose estimation, and scene understanding. Kendall et al. proposed PoseNet, a CNN-based approach for camera relocalization. Their method demonstrated the ability to estimate the 6-DOF camera pose from a single RGB image, making it suitable for indoor navigation scenarios where traditional feature-based methods might struggle [26]. Building on this work, Walch et al. introduced an improved CNN architecture for pose regression. Their approach, which incorporated Long Short-Term Memory (LSTM) units, enhanced performance in challenging indoor environments with repetitive structures [27].

Furthermore, Visual Navigation Transformers (VinT) is a foundation model that aims to bring the success of general-purpose pre-trained models to vision-based robotic navigation. ViNT is trained with a general goal-reaching objective that can be used with any navigation dataset and employs a flexible Transformer-based architecture to learn navigational affordances and enable efficient adaptation to various downstream navigational tasks. Dosovitskiy et al. introduced the Vision Transformer (ViT), demonstrating that a pure transformer applied directly to sequences of image patches can perform remarkably well on image classification tasks [28]. This work laid the foundation for applying transformers to various computer vision tasks relevant to indoor navigation. Building on this, Chen et al. proposed the Indoor Transformer (INDOOR), a visual transformer-based method specifically designed for indoor scene understanding and navigation. Their approach improved performance in tasks such as layout estimation and navigation planning compared to CNN-based methods [29].

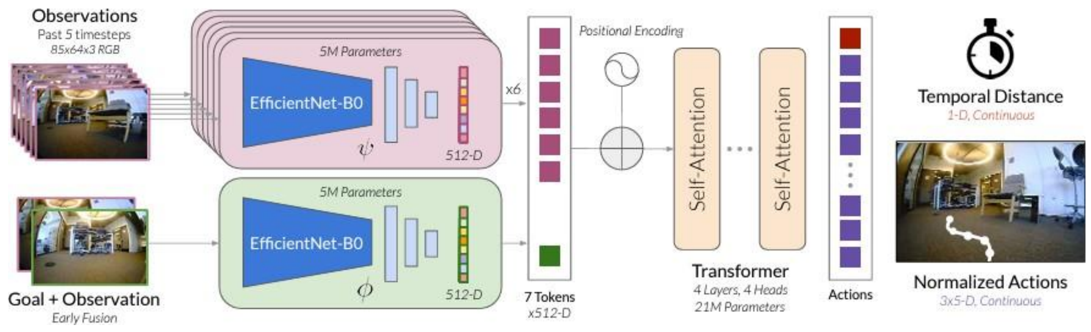


Figure 2.3.2.1 ViNT Model Architecture [30]

Figure 2.3.2.1 shows that the model begins by processing visual input through two EfficientNet-B0 encoders: one for the robots' current visual observations and another for the goal image. These encoders transform the visual data into feature embeddings that are the foundation for further processing. The model then utilises self-attention layers, which enable it to focus on relevant parts of the input image. This helps it understand spatial relationships crucial for navigation.

The core of the ViNT model is a Transformer decoder, which predicts the robot's next actions by processing the encoded visual features. This architecture is particularly effective for sequential tasks like navigation, where understanding the order of actions is essential. The model's output consists of the temporal distance to the goal, indicating how far the robot is from its destination in terms of time and a sequence of future actions the robot should take to reach the goal.

Additionally, the input data undergoes tokenization and positional encoding, which help the model comprehend visual information's spatial and sequential nature. Key parameters such as the number of Transformer layers, self-attention heads, and the dimensionality of the tokens define the model's capacity and complexity. Overall, Figure 2.3.2.1 encapsulates how the ViNT model processes visual information to generate actionable steps for efficient navigation in complex environments.

Both methods are good at extracting relevant visual features. CNNs are strong in identifying spatial hierarchies, while ViNT captures complex relationships across the visual field. Besides, both models can scale to handle large datasets and complex tasks. CNNs are widely used in large-scale image recognition, and ViNT adapts well to various robotic platforms. However, both models require significant computational resources and extensive training data to achieve high performance. These limitations can be overcome by deploying these models on specialised hardware like TPUs or GPUs and using data augmentation techniques to increase the diversity of the training data.

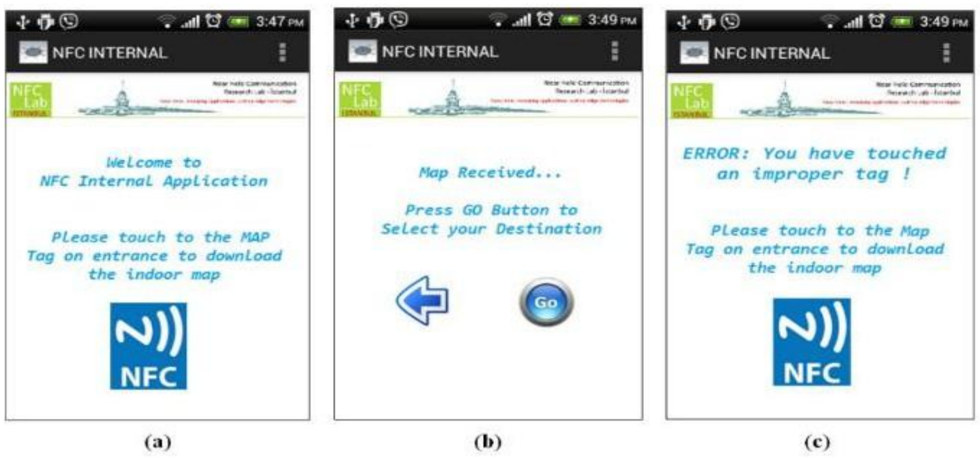
Method	**Advantages**	**Disadvantages**
Dead Reckoning	- Independence from external infrastructure - Easily accessible and portable - Cost-effective due to reliance on smartphone sensors	-Requires consistent smartphone orientation - Accuracy issues during rapid movement changes - Dependent on smartphone sensor quality
Wireless Networking	- Wide range of applications (GPS, NFC, RFID, Bluetooth, WiFi) - Suitable for large areas and complex environments	- Requires external infrastructure (access points, beacons, sensors) - Signal interference and inaccuracies in specific environments
Marker-based	- Simple and flexible to implement - Cost-effective with minimal hardware required - Robust under various light conditions with the use of appropriate toolkits	- Requires physical markers to be placed in the environment - Light intensity and direction can affect performance - Potential occlusion of markers
Markerless based	- High performance in visual tasks - Effective at feature extraction - High scalability to handle complex tasks and large datasets	- Requires extensive training data - Performance may vary depending on the environment and computational power required

Table 2.3.3.1 Comparison between Indoor Navigation Method

2.4 Review of Existing Indoor Navigation Application

2.4.1 NFC Internal Indoor Navigation Application [31]

The NFC Internal system is an innovative indoor navigation application that assists users in navigating complex indoor environments, such as shopping malls, hospitals, and university campuses, using Near Field Communication (NFC) technology. The system lets users touch their NFC-enabled mobile devices to place NFC tags within a building strategically. Then, it updates the user's location and provides directions to their desired destination. The system's key components include Map Tags, Location Tags, and the NFC Internal Application, all working together to guide users accurately and efficiently through indoor spaces.



Welcome to NFC Internal Application

Please touch to the MAP Tag on entrance to download the indoor map

Map Received...

Press GO Button to Select your Destination

ERROR: You have touched an improper tag !

Please touch to the Map Tag on entrance to download the indoor map

Figure 2.4.1.1 Initiating NFC Internal – Shopping Mall

To use the application, users begin by touching their NFC-enabled smartphone to a Map Tag located at the entrance. This tag contains the map data for the building. If the NFC application is not installed, the user is directed to the application's download page. Once installed, the map is automatically loaded onto the smartphone, as shown in Figures 14a and 14b. Suppose the user accidentally touches an incorrect or improper tag. In that case, the application will display an error message, prompting the user to touch the correct Map Tag to load the indoor map, as shown in Figure 14c.

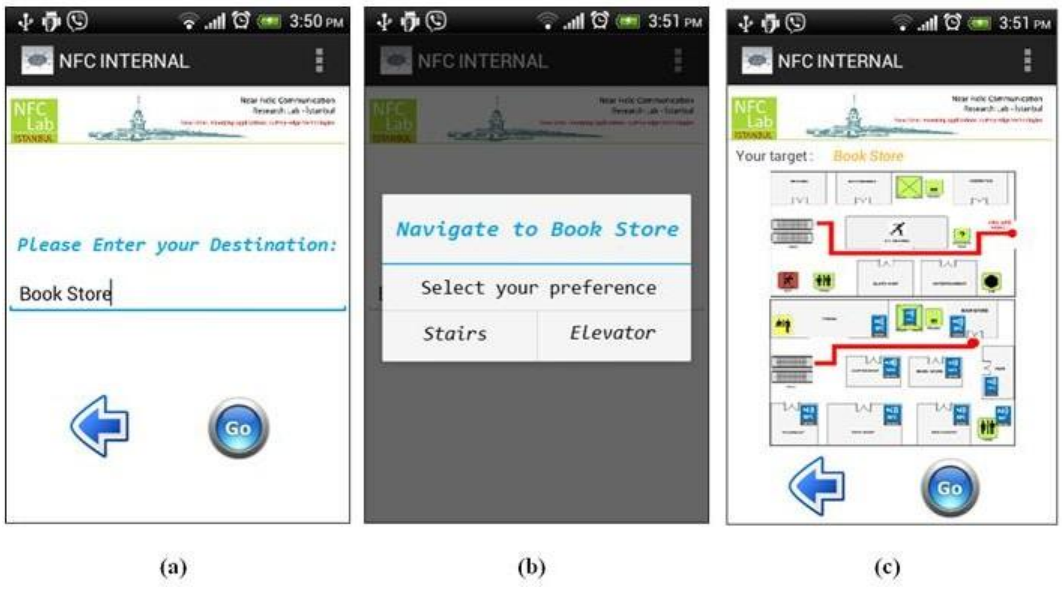


Figure 2.4.1.2 Navigate to Book Store – Shopping Mall

After the map is loaded, the user enters the name of the desired destination within the application, such as "Book Store", as shown in Figure 2.4.1.2(a). The application then prompts the user to select their preference for navigation, such as between stairs or an elevator, as shown in Figure 2.4.1.2(b). Once the preference is selected, the application computes the best route to the bookstore on the second floor and displays the directions, as shown in Figure 2.4.1.2(c).

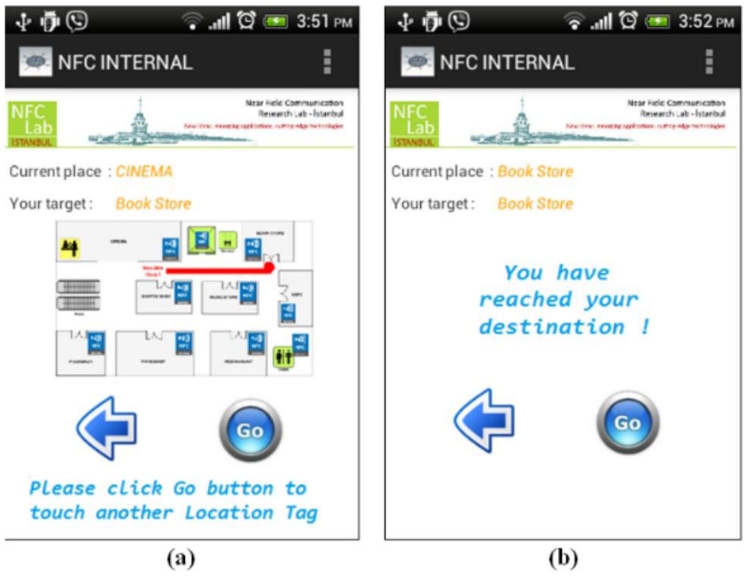

NFC INTERNAL
Current place : CINEMA
Your target : Book Store
Go
Please click Go button to touch another Location Tag
Current place : Book Store
Your target : Book Store
You have reached your destination !
(a)
(b)

Figure 2.4.1.3 Instructions Given By Location Tags

As the user moves through the building, they touch Location Tags placed at various points, such as near shops or key locations like elevators. Each time a Location Tag is touched, the application updates the user's current position on the map and provides real-time instructions to continue along the correct path, as shown in Figure 2.4.1.3(a). When the user reaches the destination, they confirm their arrival by touching the Location Tag placed at the destination point. Then, the application notifies them that they have reached their destination, as shown in Figure 2.4.1.3 (b).

This application offers several strengths, such as high accuracy due to real-time updates when users touch Location Tags. It is a user-friendly interface that leverages the natural action of touching objects. It is cost-effective and has low installation and maintenance costs. Furthermore, NFC Internal ensures user privacy, as location data is only processed on the user's device. It boasts performance with NFC tags with nearly unlimited lifespan and fault resistance.

However, this application has some weaknesses. It relies on users having NFC-enabled mobile devices, which may not be universally available. It also does not provide continuous real-time positioning between tags relying on dead reckoning (DR) technology, which can lead to errors. Moreover, there is a potential for queues forming at Location Tags, especially in busy areas like elevators. Solutions such as providing NFC-enabled devices for loans, increasing the density of Location Tags to improve real-time positioning, and installing redundant tags in high-traffic areas to prevent queues can be considered to address these issues.

2.4.2 Dead Reckoning Indoor Positioning Application

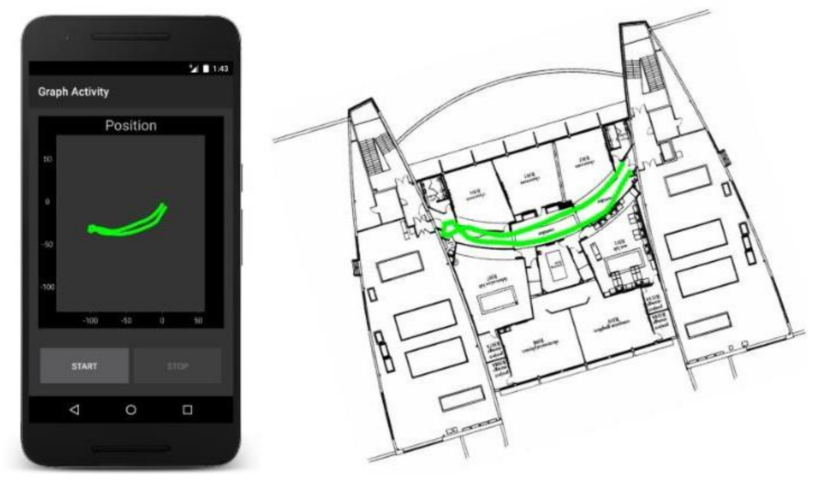


Figure 2.4.2.1 Dead Reckoning Location Tracking Application [32]

The project addresses the limitations of GPS in locations where it is unreliable or unavailable, such as indoors or near large structures. It proposes a solution using dead reckoning, which relies on smartphone inertial sensors to track location. It makes the system self-sufficient and independent of GPS. The app was developed using the Android Software Development Kit (SDK) and tested on a Google Nexus 5 smartphone. It includes a step counter, compass, data collector, QR code scanner, calibration mode, and an interactive graph. The app determines the user's initial orientation relative to Earth by analyzing magnetic fields and gravity data via a Direction Cosine Matrix (DCM). As the user walks, the app detects steps, calculates distance, and monitors heading changes using gyroscope data. This data is then used to calculate the current location, plotted on a screen and stored. The results indicated that while the app could track location, it accumulated errors over time. For instance, after a 450-meter walk, the position error reached 35 meters. The app requires further refinement for navigation, particularly in reducing errors over time. Future improvements could involve better sensor integration and data fusion techniques to minimize errors further and enhance accuracy.

2.4.3 Marker-Based Indoor Navigation Application

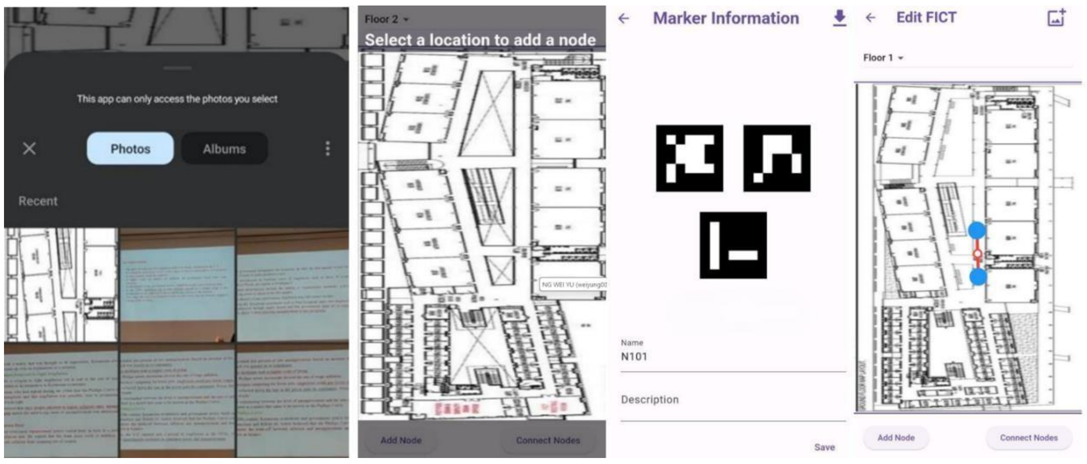


Figure 2.4.3.1 ArUco-Based Indoor Navigation Application [33]

Based on Figure 2.4.3.1, the administrator first uploads the building's floor plan to the system via the mapping module. This floor plan will serve as the base for creating the digital representation of the indoor environment. Then, the administrator selects specific locations on the floor plan where navigation points (nodes) are needed. Each node represents a physical location within the building, such as a room or toilet. Once a node is added, the system automatically generates a corresponding ArUco marker. This unique marker contains encoded information such as the building ID, floor ID, and marker ID. After that, the administrator downloads the generated ArUco markers and prints them out. The printed markers are then placed at the corresponding physical locations in the building.

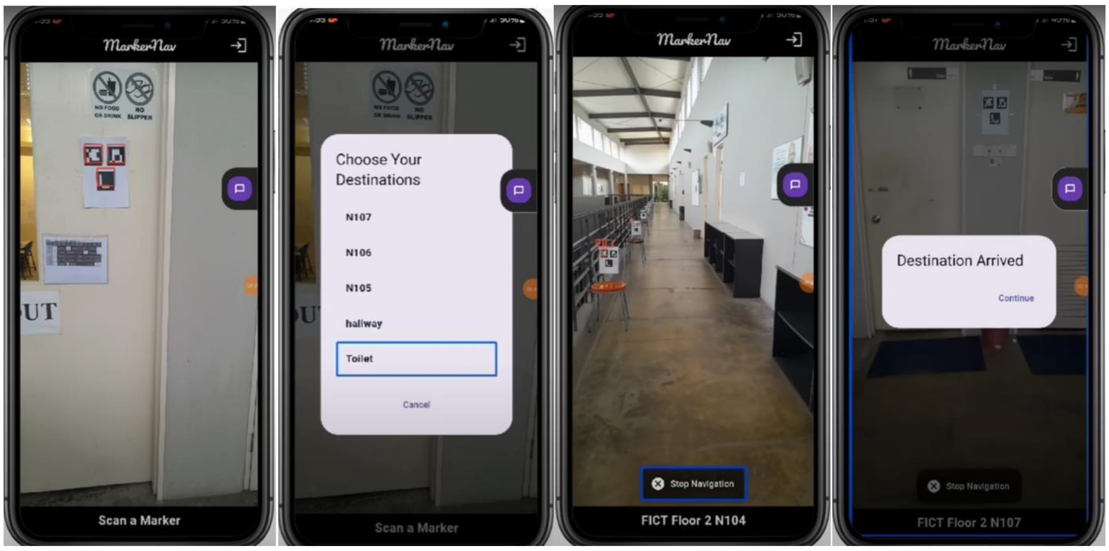


Figure 2.4.3.2: ArUco-Based Indoor Navigation Application

According to Figure 2.4.3.2, the user starts by scanning an ArUco marker with their smartphone to determine their current location within the Faculty of Information Technology. After the marker is detected, the application lists the destinations. The selection will trigger the system to calculate the shortest path using Dijkstra's algorithm. In this case, the user selects the "Toilet" option. The system continuously scans for ArUco markers along the path as the user moves. Each time a marker is detected, the system provides immediate audio feedback to guide the user. The mobile will vibrate if the user goes the wrong way. Finally, the application displays a "Destination Arrived" and an audio message showing the user has reached their destination.

The application is designed to be cost-effective, easy to deploy, and flexible. Besides, the system provides continuous real-time audio feedback as users navigate. This enhances the user experience, especially for individuals with visual impairments. However, this application relies heavily on the visibility of ArUco markers. If markers are damaged, the system will fail to guide the user accurately. This problem can be solved by establishing a routine maintenance protocol that regularly checks the condition of the marker. Furthermore, the lighting conditions, such as low or overly bright, may affect the camera's ability to detect ArUco markers. This will lead to potential navigation errors. One solution is to integrate additional technologies, such as Bluetooth beacons or Wi-Fi triangulation, as a backup method for localization when markers cannot be detected.

Existing Application	Advantages	Disadvantages
NFC Internal Indoor Navigation Application	-High accuracy due to real-time updates. - Ensure user privacy. - Cost-effective with inexpensive and durable tags.	- Dependent on the density and placement of NFC tags. - Limited by the need for physical interaction with NFC tags. - Potential challenges in large or complex environments where many tags are needed.
Dead Reckoning Indoor Positioning Application	- Self-sufficient and independent of GPS. - Cost-effective as it uses existing smartphone hardware.	- Accumulates errors over time, reducing accuracy, especially in long-distance navigation. - Sensor inaccuracies and environmental factors impact performance.
ArUco Based Indoor Navigation Application	- Cost-effective, flexible, and easy to deploy. - Real-time guidance through audio, vibration, and visual cues. - Automatically generates markers for easy setup.	- Performance can be affected by environmental factors such as lighting conditions. - Rely heavily on the visibility of ArUco marker.

Table 2.4.3.1: Comparison Between Existing Indoor Navigation Applications

CHAPTER 3 SYSTEM METHODOLOGY/APPROACH

3.1 Overview of Solution

The proposed solution for the indoor navigation app for visually impaired users is a vision-based system implemented as a Flutter mobile application integrated with a Python backend. The backend is hosted using FastAPI, which provides a lightweight and efficient framework for serving deep learning models through RESTful APIs. The mobile app leverages the device's camera to capture real-time images, which are sent to the FastAPI server for processing. The server generates semantic embeddings using **DINOv2** and compares them via cosine similarity to stored waypoint embeddings for accurate localisation and navigation. To handle dynamic environments, **YOLOv8-seg large** is used for people detection and segmentation, with unwanted regions reconstructed using **Stable Diffusion 2.0**. The system also incorporates compass integration for directional guidance and text-to-speech (TTS) for audio feedback. Paths, embeddings, and metadata are stored and managed in **Supabase**, while the app enables users to record navigation paths by capturing images at key waypoints and storing their embeddings for real-time matching during navigation.

Flutter was chosen to build the Android application because it combines high performance with a rich ecosystem of packages that simplify development. Packages such as camera, flutter_compass, flutter_tts, and http make it straightforward to integrate core functions like camera streaming, compass readings, text-to-speech feedback, and API communication without needing extensive low-level coding. This wide package support allows rapid prototyping and reliable implementation of features essential for indoor navigation. Additionally, Flutter offers smooth integration with Android's native capabilities while maintaining responsive performance, ensuring the app can handle real-time inputs from the camera and sensors. By using Flutter for the front end and offloading heavy tasks such as embedding generation and people detection to the Python backend, the system remains efficient and accessible.

An **embedding** is a numerical representation of data that captures its most important semantic features in a high-dimensional vector space. For images, this means converting visual information into a compact vector that reflects patterns such as shapes, textures, and spatial arrangements rather than raw pixel values. As a result, two images of the same location, even under different lighting conditions or viewing angles, will generate embeddings that are close together, while images of different places will be farther apart. To generate these embeddings, this system employs the **DINOv2 vision transformer model**, a state-of-the-art approach in self-supervised learning that produces highly robust and generalizable visual representations. We chose DINOv2 because its embeddings remain consistent across variations common in indoor environments, such as changes in illumination, occlusions, or dynamic objects. Compared to classical feature-based techniques like SIFT or SURF, DINOv2 does not rely on handcrafted features and instead learns richer semantic representations directly from large-scale image data. Similarly, while lightweight CNN-based models such as MobileNet can provide faster results, they often sacrifice accuracy and robustness, making them less suitable for safety-critical tasks like navigation for visually impaired users.

To handle dynamic indoor environments, the system employs **YOLOv8-seg large** for people detection and segmentation, combined with **Stable Diffusion 2.0** for image inpainting. YOLOv8-seg large was selected because it provides state-of-the-art accuracy in both detecting and segmenting people, outperforming earlier models such as Mask R-CNN or DeepLab in terms of speed and precision. Its ability to generate precise segmentation masks ensures that moving people can be cleanly removed from the scene without leaving irregular artifacts. Once segmented, **Stable Diffusion 2.0** is applied to reconstruct the missing regions. We chose Stable Diffusion over other inpainting models like LaMa or traditional GAN-based approaches because it produces more realistic and context-aware reconstructions, particularly in complex indoor scenes with textures and walls.

The backend of the system is built with **FastAPI**, chosen because models such as **DINOv2** and **Stable Diffusion 2.0** are large and require GPU acceleration to run efficiently in real time. Instead of attempting to convert these models into lightweight formats like TensorFlow Lite, it would sacrifice accuracy and still be too resource-intensive for mobile devices. For data management, the system integrates **Supabase**, a PostgreSQL-based platform with built-in user authentication for secure and personalised navigation. Supabase also supports the pgvector extension, enabling efficient storage and similarity search of high-dimensional embeddings. Together, FastAPI and Supabase form a robust backend pipeline that combines high-performance inference with scalable, secure data management, ensuring reliable navigation support for visually impaired users.

3.2 System Architecture Diagram

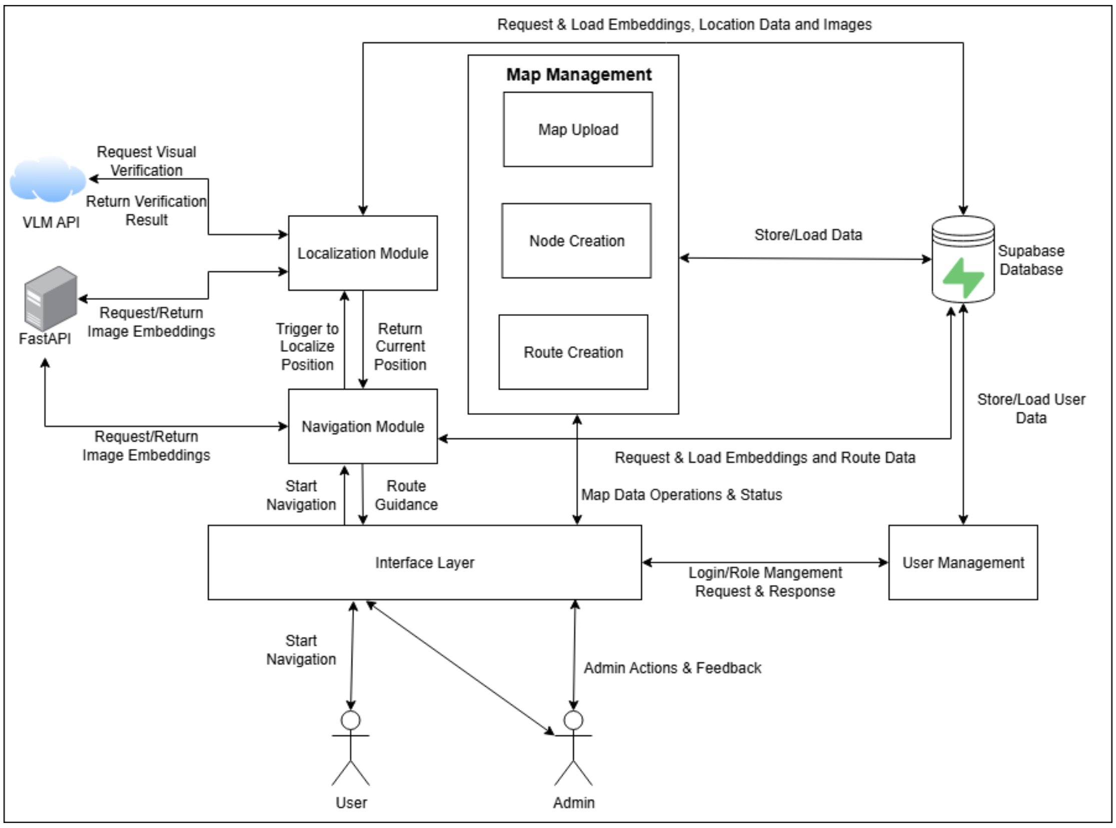

Request & Load Embeddings, Location Data and Images
Map Management
Map Upload
Node Creation
Route Creation
Request Visual Verification
Return Verification Result
VLM API
Localization Module
Store/Load Data
Supabase Database
Request/Return Image Embeddings
FastAPI
Trigger to Localize Position
Return Current Position
Navigation Module
Request/Return Image Embeddings
Store/Load User Data
Request & Load Embeddings and Route Data
Start Navigation
Route Guidance
Map Data Operations & Status
Interface Layer
Login/Role Mangement Request & Response
User Management
Start Navigation
Admin Actions & Feedback
User
Admin

Figure 3.2.1 System Architecture Diagram

This system design diagram illustrates a comprehensive indoor navigation and mapping system, architected to support real-time localization, route guidance, and detailed map administration. The system serves two primary roles which are administrators who manage the environment and regular users who navigate within it.

For a regular user, initiating navigation triggers a sophisticated localization process. The Localization Module is the core of this operation. To pinpoint the user's exact position, it first captures a real-time image and sends it to a dedicated FastAPI server, which converts the image into a vector embedding. This embedding is then used to query the Supabase Database, which searches for the most similar stored image and retrieves its associated location data. For final confirmation, both the live image and the retrieved database image are sent to a VLM API for a visual verification, ensuring high accuracy. Once the location is confirmed, it is passed to the Navigation Module, which then provides clear Route Guidance to the user through the Interface Layer.

For administrators, the system provides a powerful suite of tools through the Map Management module. They can upload new floor plans, create and edit location nodes, and define the routes that connect them. The User Management module allows administrators to handle user roles, permissions, and secure login access.

All system data is centralized in the Supabase Database. It acts as the single source of truth, storing everything from map layouts, image embeddings, and route data to user credentials. This ensures that both the user-facing navigation features and the back-end administrative tools operate on consistent and up-to-date information.

3.3 User Requirements

The user requirements define what the end-users expect from the system in terms of functionality, performance, and usability. These requirements guide the development process to ensure that the final product aligns with user needs and enhances the workflows. The requirements are divided into functional requirements and nonfunctional requirements.

3.3.1 Functional Requirements

FR1: User Authentication & Account Management

- Allow users (admin or VIP) to register an account.
- Allow users to log in with valid credentials.
- Allow registered users to reset their passwords.
- Differentiate between admin and VIP accounts during registration.
- Restrict admin functions to authenticated admin users only.

FR2: Organisation Management

- Allow admin to create an organisation.
- Allow admin to delete an organisation.
- Allow admin to invite VIP users to an organisation via email.
- Allow admin to remove VIP users from an organisation.

FR3: Floor Map & Node Management

- Allow admin to upload a floor map.
- Allow admin to delete a floor map.
- Allow admin to place nodes (waypoints) on the floor map.
- Allow admin to edit or delete existing nodes.

FR4: Edge & Path Management

- Allow admin to connect nodes by recording a path.
- Automatically capture frames and compass readings during path recording.
- Convert captured images into embeddings and store them in the database.
- Create and store edges between nodes based on recorded paths.
- Allow admin to edit or delete existing edges.

FR5: Localisation

- Convert captured frames into embeddings and compare with stored embeddings in Supabase to choose the best similarity for each frame.
- The frames are compared using VLM for location verification.

FR 6: Navigation

- Allow VIP users to capture frames in real time during navigation.
- Identify the user's current position based on similarity search.
- Provide navigation guidance (e.g., turn left, turn right, go straight) to VIP users.
- Provide audio feedback for visually impaired users during navigation.

FR7: Error Handling

- Notify users if login, registration, or authentication fails.
- Notify admin if floor map, node, or edge upload fails.
- Retry saving embeddings and paths if network failure occurs.

3.3.2 Non-Functional Requirements

NFR1: Performance

- The system should localise the user's position within 2 seconds after capturing a frame.
- Path recording should not drop frames during continuous capture.

NFR2: Security

- User credentials should be stored using encryption.
- Only authenticated users can access the system.
- Admin functions must be restricted to admin accounts only.

NFR3: Scalability

- The system should support at least 100 concurrent users.
- Supabase vector search should scale with large embedding datasets.

NFR4: Reliability

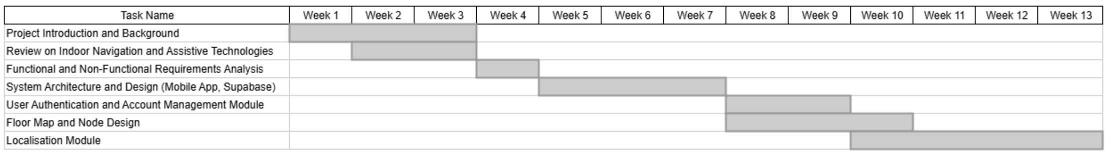
- The system should handle errors gracefully (e.g., failed uploads, network disconnections).
- Captured embeddings and routes must be saved persistently in the database.

NFR5: Maintainability

- The system should have modular code for easy updates.
- Documentation should be provided for developers to maintain and extend the system.

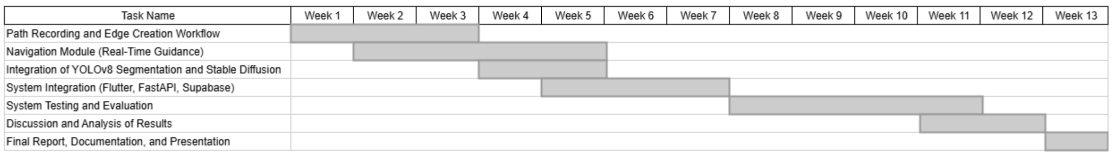
3.4 Timeline and Gantt Chart

Both figures below illustrate a Gantt Chart for this project in Project I and Project II.



Task Name	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Project Introduction and Background	▨	▨	▨										
Review on Indoor Navigation and Assistive Technologies		▨	▨										
Functional and Non-Functional Requirements Analysis				▨									
System Architecture and Design (Mobile App, Supabase)					▨	▨	▨						
User Authentication and Account Management Module								▨	▨				
Floor Map and Node Design								▨	▨	▨			
Localisation Module										▨	▨	▨	▨

Figure 3.4.1 Gantt Chart of FYP1 Timeline



Task Name	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Path Recording and Edge Creation Workflow	▨	▨	▨										
Navigation Module (Real-Time Guidance)		▨	▨	▨	▨								
Integration of YOLOv8 Segmentation and Stable Diffusion				▨	▨								
System Integration (Flutter, FastAPI, Supabase)					▨	▨	▨						
System Testing and Evaluation								▨	▨	▨	▨		
Discussion and Analysis of Results											▨	▨	
Final Report, Documentation, and Presentation													▨

Figure 3.4.2 Gantt Chart of FYP2 Timeline

CHAPTER 4: SYSTEM DESIGN

4.1 System Flowchart

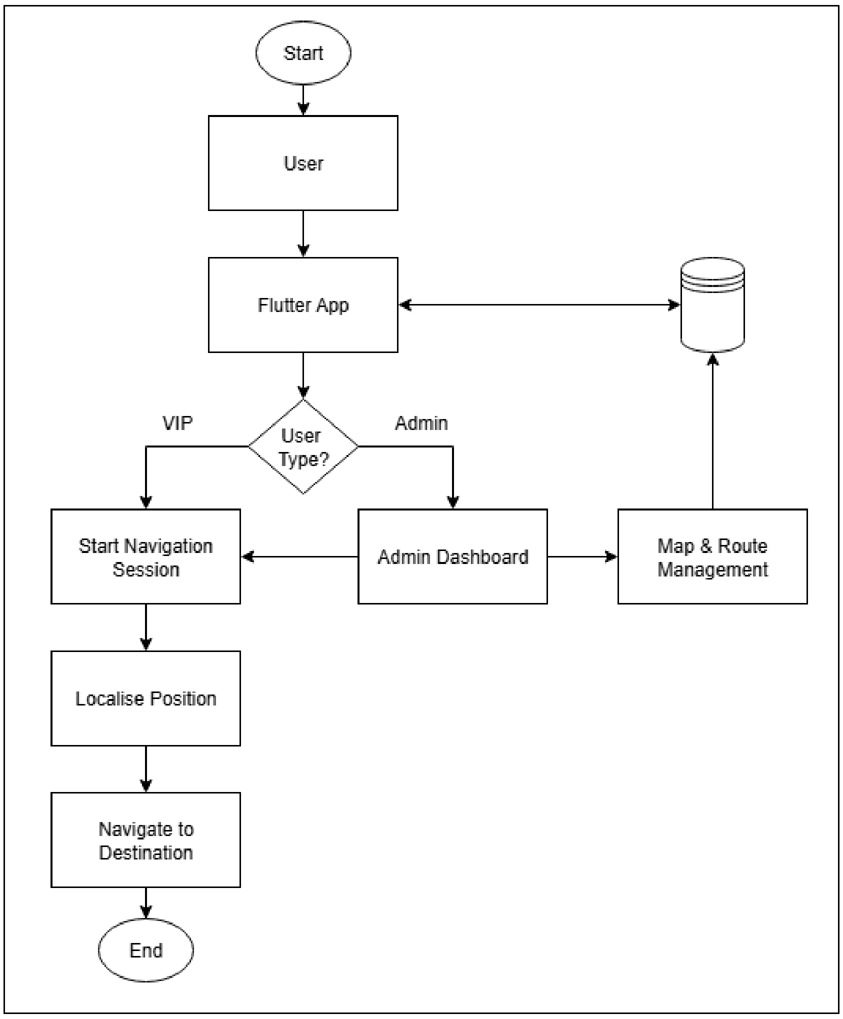


Figure 4.1.1 System Flowchart

This flowchart illustrates the operational workflow of a role-based indoor navigation system built with Flutter. The process begins with a user interacting with the Flutter application. The system then determines the user's role. If the user is an Admin, they are directed to an Admin Dashboard. From here, they can access the "Map & Route Management" module to create, update, or delete navigation data, which is stored in and retrieved from a database. The flowchart also indicates that an Admin can initiate a "Start Navigation Session," granting them access to the same navigation features as a VIP user. If the user is a VIP, their path is streamlined for navigation. They directly enter a "Start Navigation Session," where the application first works to "Localise Position" to determine their current location. Once the position is established, the system proceeds to "Navigate to Destination," guiding until the process ends.

4.2 User Authentication and Account Management

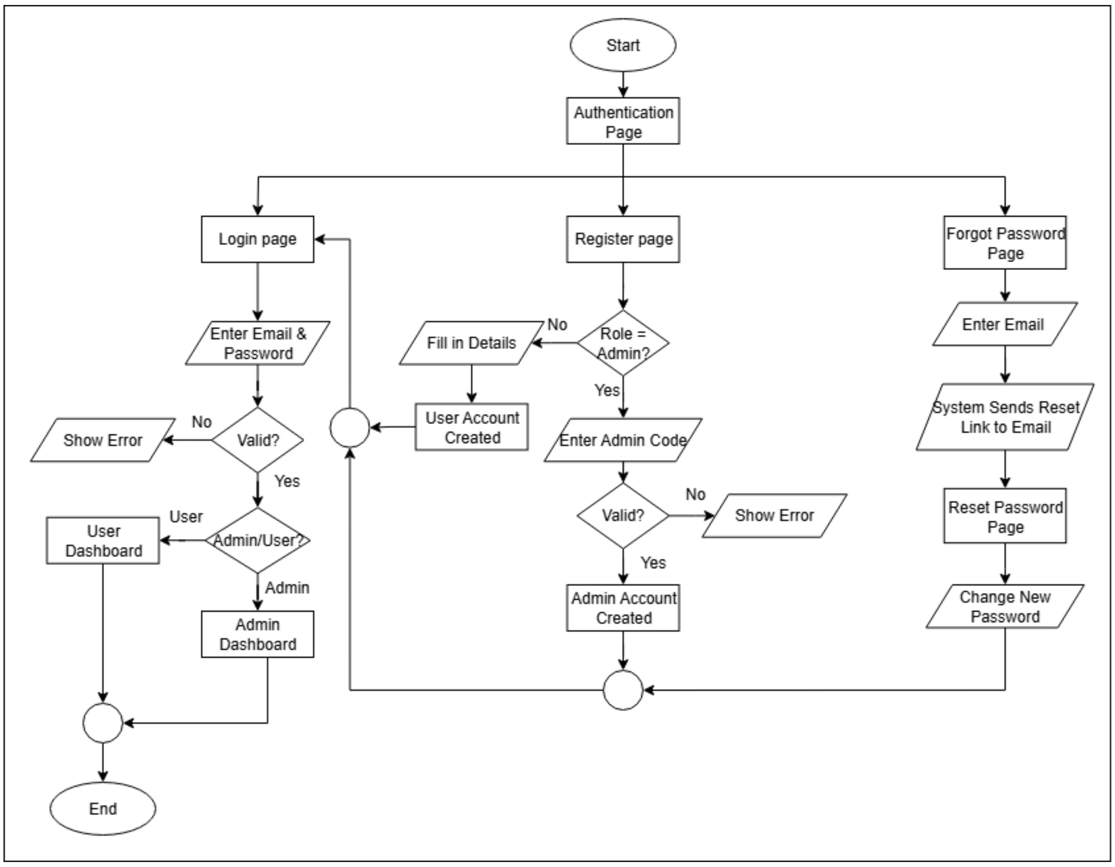


Figure 4.2.1.1 User Authentication and Account Management Flowchart

The process begins at the authentication page, where users can either register a new account, log in, or recover their password. For registration, the user fills in the required details and selects their role. If the role is a normal user, the system directly creates a user account. If the role is admin, the user is required to enter a verification code (ADMIN123). If the code is valid, an admin is created. Otherwise, an error is displayed and the user is redirected to reattempt registration.

For login, users enter their email and password, which the system validates. If the credentials are incorrect, an error message is shown, prompting the user to retry. If the credentials are valid, the system determines the user's role. Normal users are directed to the user dashboard, while admins are directed to the admin dashboard.

If the user selects "Forgot Password," they must enter their email, after which the system sends a reset link to the email. The user can then proceed to the reset password screen, change their new password, and return to the login page. The process ensures secure access control by differentiating between admin and normal user roles, enforcing role-specific validation, and supporting password recovery.

4.3 Admin and VIP Role Module

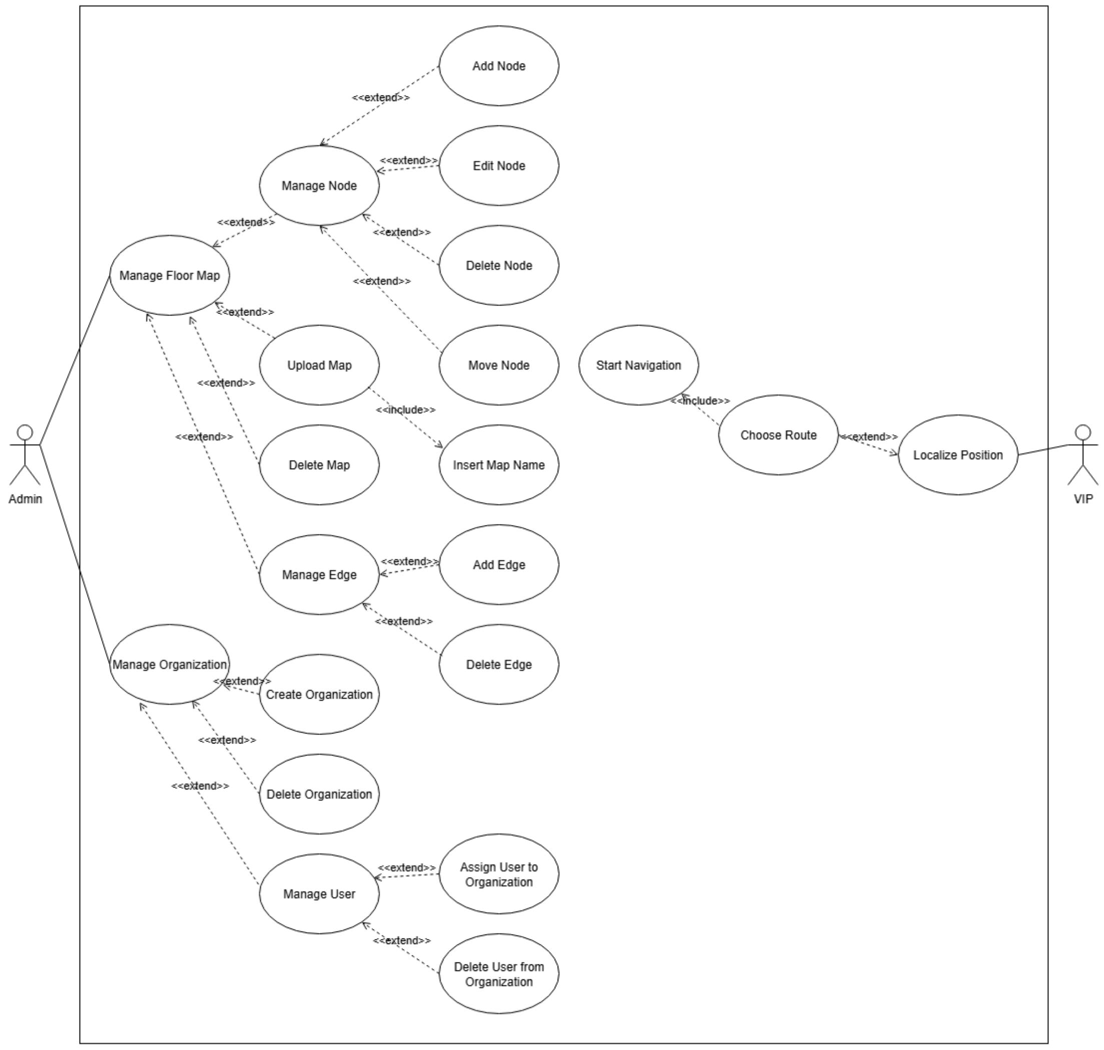


Figure 4.3.2.1 Admin and VIP Role Use Case Diagram

The map management dashboard will only be displayed when the user is logged in as an admin. The user can upload a new map from the album and assign it a relevant name. They can also view available maps and delete existing maps with a confirmation prompt. Once a map is uploaded, the admin can interact with it by adding, editing, deleting, or repositioning nodes. For example, admins can edit a node to change its name or re-record the video if needed. Nodes can also be deleted or moved to a more accurate position on the map. In addition, admins can manage edges between nodes. They can add new edges to create a route between locations and also delete existing edges. Furthermore, admins can manage organisations within the application. This includes adding new organisations and deleting organisations. After creating an organisation, the admin can assign the users to organisations by inserting the VIP's email address and also delete the VIP from the organisation. The purpose of inviting the VIP to the organisation is to provide them with access to the indoor navigation services within that specific organisation. Once the VIP has been successfully added, they will be able to perform localisation by recognising their position and navigate based on the predefined routes or nodes set up by the corresponding admin account. This ensures that each organisation can maintain its own customised navigation environment.

4.3.1 Node Creation

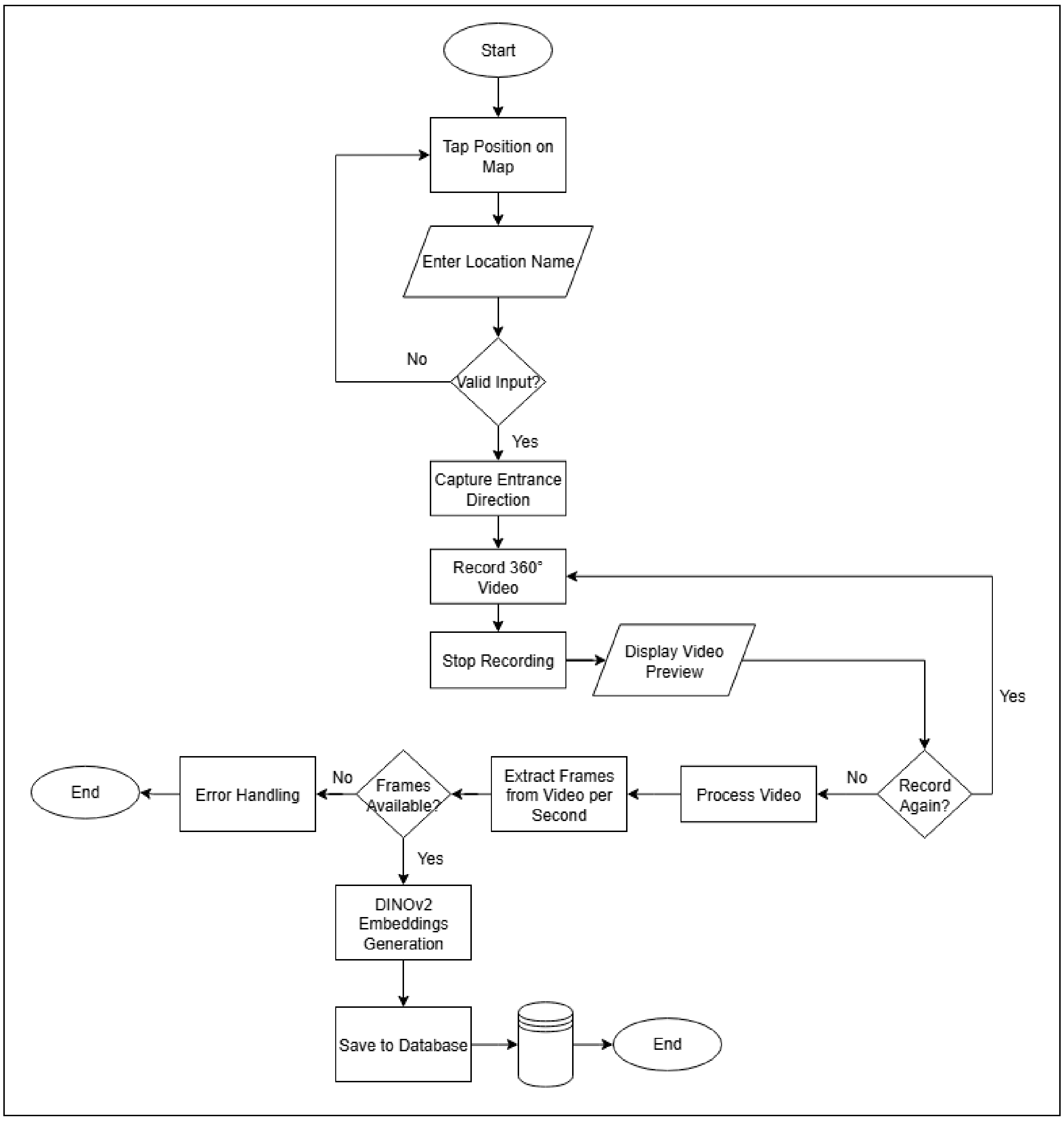


Figure 4.3.1.1 Node Creation Procedure

This flowchart illustrates a process for creating nodes, which begins with the admin tapping on the floor map and providing a name for the new location. After naming, the user is asked to stand still and face the location to capture the current compass direction. It will later be used during recognition to determine the relative position of the user with the saved location. After that, the user can record 360-degree videos of the surroundings. If they are not satisfied with the recording, they can re-record. Upon confirmation, the captured video is extracted frame by frame per second. Then, each frame is preprocessed through the Dinov2 model to generate semantic embeddings, which have 768-dimensional vectors that capture visual features for place recognition. Besides, each frame will be stored in the bucket for comparison between the stored image and the captured image used by VLM for localisation verification. Lastly, all the information will be stored in the Supabase vector database.

4.3.2 Edge Creation

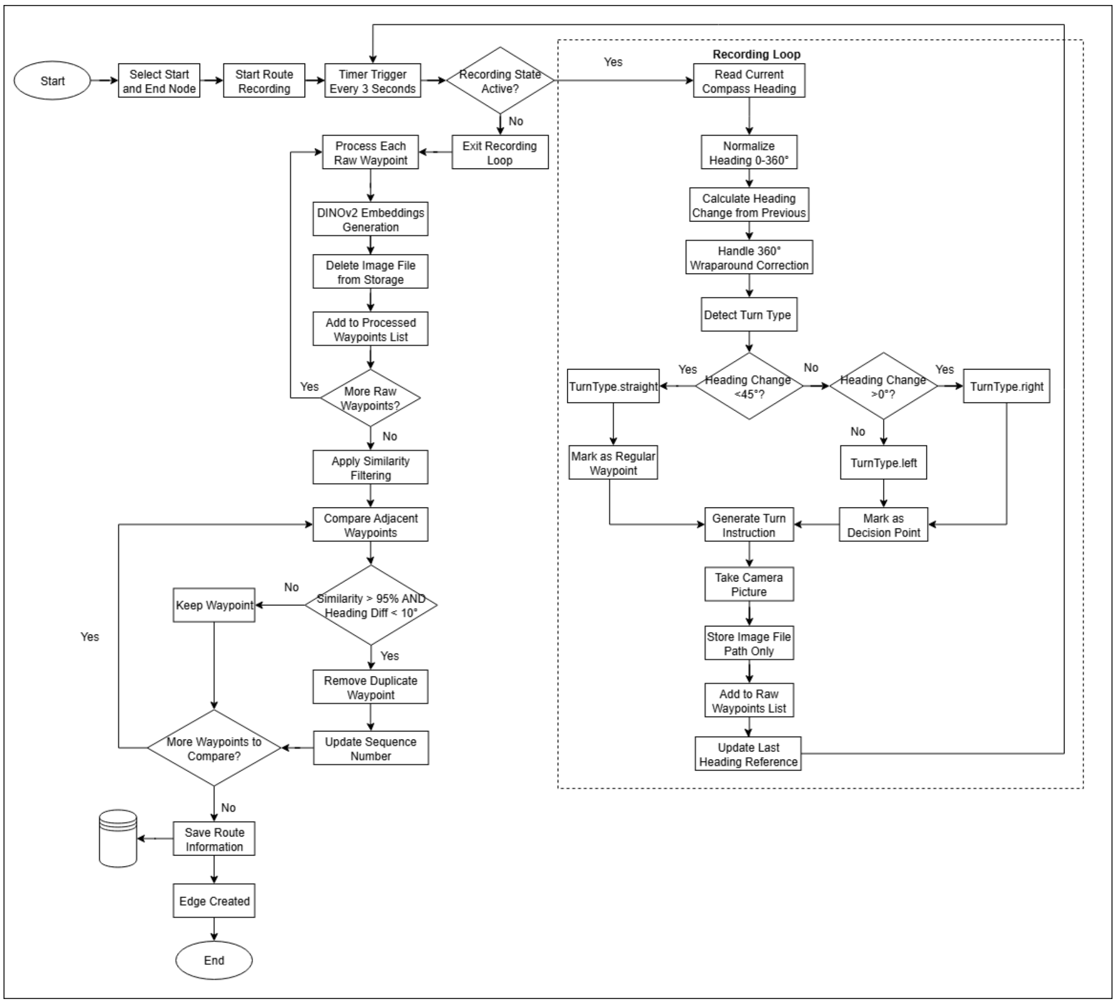

Start
Select Start and End Node
Start Route Recording
Timer Trigger Every 3 Seconds
Recording State Active?
Yes
No
Recording Loop
Read Current Compass Heading
Normalize Heading 0-360°
Calculate Heading Change from Previous
Handle 360° Wraparound Correction
Detect Turn Type
Heading Change <45°?
Heading Change >0°?
TurnType.straight
TurnType.right
TurnType.left
Mark as Regular Waypoint
Mark as Decision Point
Generate Turn Instruction
Take Camera Picture
Store Image File Path Only
Add to Raw Waypoints List
Update Last Heading Reference
Exit Recording Loop
Process Each Raw Waypoint
DINOv2 Embeddings Generation
Delete Image File from Storage
Add to Processed Waypoints List
More Raw Waypoints?
Apply Similarity Filtering
Compare Adjacent Waypoints
Similarity > 95% AND Heading Diff < 10°
Keep Waypoint
Remove Duplicate Waypoint
Update Sequence Number
More Waypoints to Compare?
Save Route Information
Edge Created
End

Figure 4.3.2.1 Edge Creation Flowchart

Figure 4.3.2.1 illustrates the workflow for edge creation in the proposed indoor navigation system. The process begins when the administrator selects a start node and an end node, followed by initiating the route recording. Once recording is activated, the system enters a timed loop that is triggered every three seconds to capture both a camera frame and the corresponding compass heading.

Within the recording loop, the compass heading is normalized to a 0°–360° range, and the change relative to the previous heading is calculated. A wrap-around correction is applied to ensure that the heading difference always falls within the range of −180° to +180°. Based on this difference, the system classifies the movement into three categories: straight movement if the change is less than 45°, a right turn if the positive change is equal to or greater than 45°, and a left turn if the negative change is equal to or greater than 45°. Straight movements are marked as regular waypoints, while directional changes are marked as decision points. At each waypoint, a turn instruction is generated, a camera picture is taken, and only the file path is stored. The waypoint metadata, including heading, heading difference, turn type, decision-point flag, timestamp, and sequence number, is then added to the raw waypoint list. The last heading reference is updated before the system continues to the next capture. Recording continues until the administrator reaches the destination node or stops the process manually.

Once the recording session ends, the system proceeds to process the raw waypoints. Each captured image is converted into embeddings using the DINOv2 model, after which the image files are deleted from storage to optimize memory usage. The processed waypoints are then subjected to similarity filtering. Adjacent waypoints are compared, and duplicates are removed if their embedding similarity is greater than 95% while their heading difference is less than 10°. The sequence numbers are updated to maintain the correct order of the remaining waypoints.

Finally, after all waypoints are processed, the route information is saved in the Supabase database. An edge is then created to link the selected start and end nodes, which is subsequently displayed on the map. This marks the completion of the edge creation process within the system.

4.4 Localisation Module

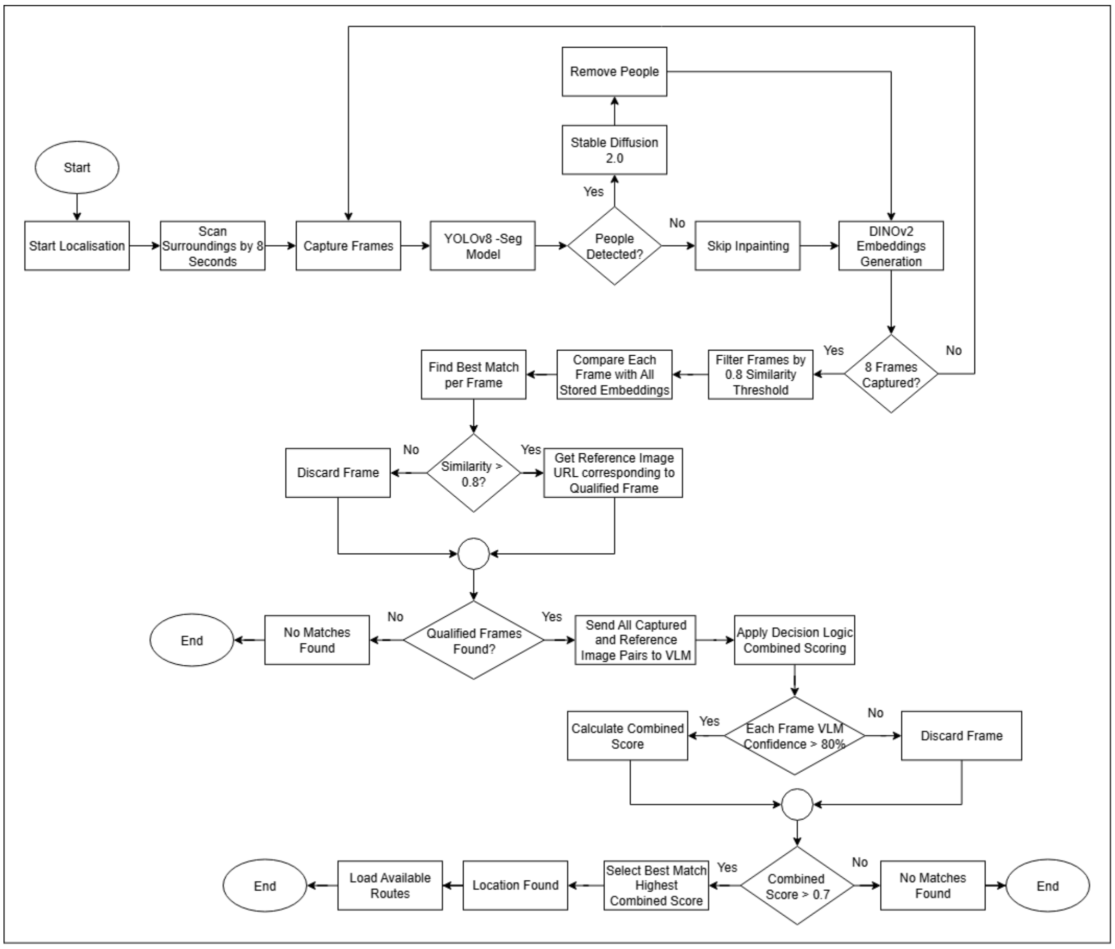

Start
Start Localisation
Scan Surroundings by 8 Seconds
Capture Frames
YOLOv8 -Seg Model
People Detected?
Yes
Stable Diffusion 2.0
Remove People
No
Skip Inpainting
DINOv2 Embeddings Generation
8 Frames Captured?
Yes
No
Filter Frames by 0.8 Similarity Threshold
Compare Each Frame with All Stored Embeddings
Find Best Match per Frame
Similarity > 0.8?
No
Discard Frame
Yes
Get Reference Image URL corresponding to Qualified Frame
Qualified Frames Found?
No
No Matches Found
End
Yes
Send All Captured and Reference Image Pairs to VLM
Apply Decision Logic Combined Scoring
Each Frame VLM Confidence > 80%
Yes
Calculate Combined Score
No
Discard Frame
Combined Score > 0.7
Yes
Select Best Match Highest Combined Score
Location Found
Load Available Routes
End
No
No Matches Found
End

Figure 4.4.1 Localisation Process

The localisation process begins when users start localisation by scanning their surroundings to capture eight consecutive camera frames at one-second intervals. Each frame then undergoes AI-powered preprocessing. A YOLO object detector is applied to identify people in the scene, though this feature can be disabled to improve speed. If people are detected, the system applies an inpainting technique to remove them, producing a clean and reliable visual representation of the environment. These preprocessed frames are then passed through the DINOv2 vision transformer, which generates a 768-dimensional semantic embedding that captures the unique visual features of the indoor setting.

To identify the user's location, the system compares each embedding with all stored reference locations in the Supabase database using cosine similarity. Frames that achieve similarity scores above 0.8 are flagged as qualified frames and sent for further verification using a Vision-Language Model (VLM). At this stage, the system batches the candidate frames with reference images and analyzes them with GPT-4-mini, which evaluates whether the frames semantically match the stored references. Only frames with a VLM confidence score above 80% move forward.

The system then calculates a combined confidence score for each verified frame, weighing the embedding similarity at 60% and the VLM confidence at 40%. To confirm localisation, the highest combined score must reach at least 70%. Once this threshold is met, the system identifies the user's most likely location, displays the result with detailed confidence metrics, and retrieves available navigation routes from that point. Users can then select their destination and begin turn-by-turn indoor guidance.

4.5 Navigation Module

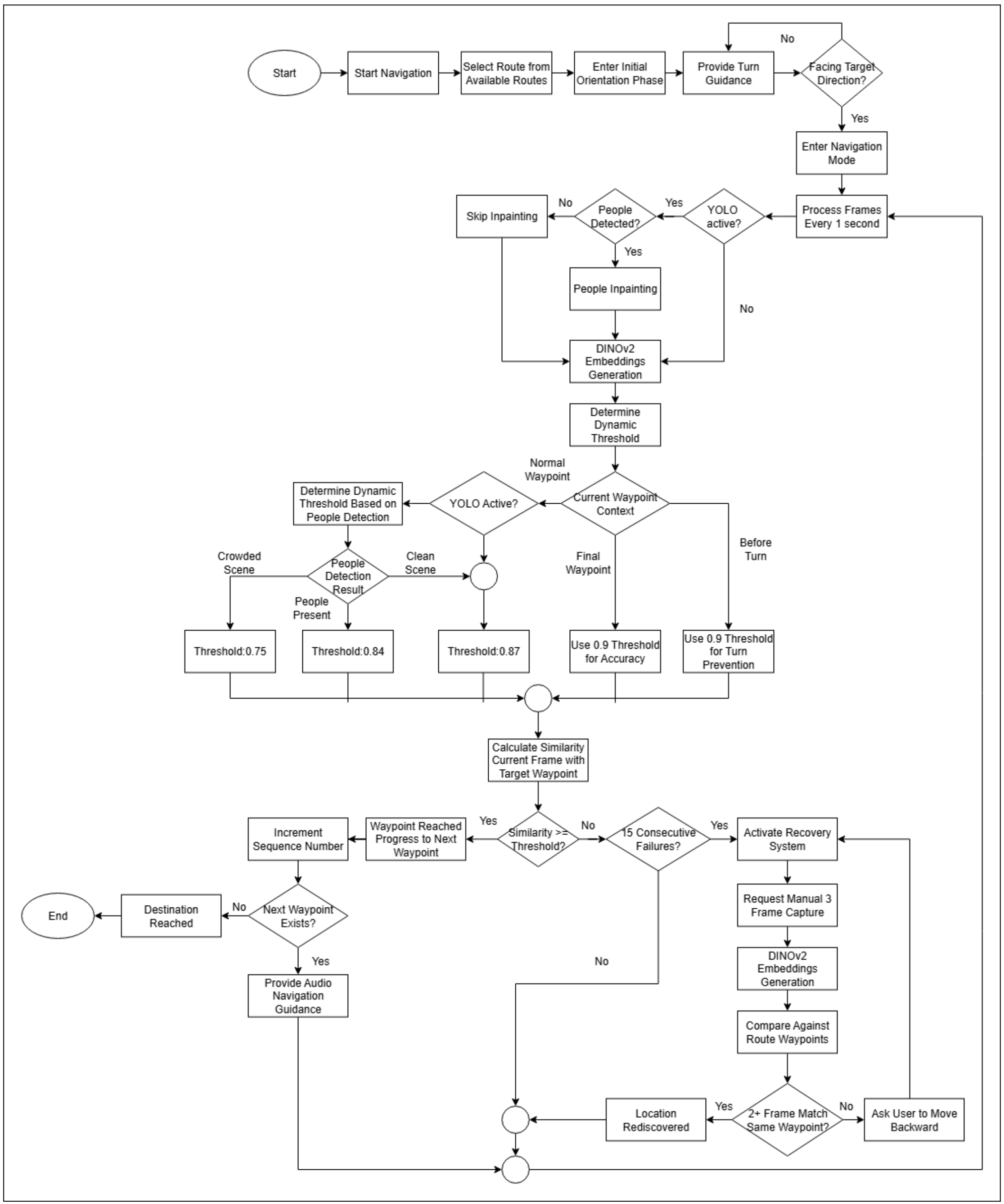

Start
Start Navigation
Select Route from Available Routes
Enter Initial Orientation Phase
Provide Turn Guidance
Facing Target Direction?
No
Yes
Enter Navigation Mode
Process Frames Every 1 second
YOLO active?
Yes
People Detected?
No
Skip Inpainting
Yes
People Inpainting
No
DINOv2 Embeddings Generation
Determine Dynamic Threshold
Current Waypoint Context
Normal Waypoint
YOLO Active?
Determine Dynamic Threshold Based on People Detection
People Detection Result
Crowded Scene
Clean Scene
People Present
Final Waypoint
Before Turn
Threshold:0.75
Threshold:0.84
Threshold:0.87
Use 0.9 Threshold for Accuracy
Use 0.9 Threshold for Turn Prevention
Calculate Similarity Current Frame with Target Waypoint
Similarity >= Threshold?
Yes
Waypoint Reached Progress to Next Waypoint
Increment Sequence Number
Next Waypoint Exists?
No
Destination Reached
End
Yes
Provide Audio Navigation Guidance
No
15 Consecutive Failures?
Yes
No
Activate Recovery System
Request Manual 3 Frame Capture
DINOv2 Embeddings Generation
Compare Against Route Waypoints
2+ Frame Match Same Waypoint?
Yes
Location Rediscovered
No
Ask User to Move Backward

Figure 4.5.1 Navigation Module Flowchart

The navigation process begins when users select a route from available destinations, triggering the system to load the corresponding waypoints and establish the navigation sequence. Before commencing actual navigation, the system enters an initial orientation phase that utilizes the device's compass system to ensure users are facing the correct direction for the first waypoint. If compass readings are available, the system continuously monitors the user's heading and provides audio guidance to turn left, right, or continue straight until the target orientation is achieved.

Once properly oriented, the system transitions to active navigation mode, processing camera frames at one-second intervals to maintain real-time position tracking. Each frame undergoes YOLO-based person detection, with inpainting techniques applied when people are detected to remove dynamic elements that could interfere with visual matching. The system then generates DINOv2 embeddings for scene analysis and applies a sophisticated dynamic threshold system that adapts based on waypoint context and environmental conditions. For waypoints preceding turns or serving as final destinations, the system employs a 90% similarity threshold to prevent premature turn instructions and ensure destination accuracy, while normal waypoints utilize dynamic thresholds ranging from 87% for clean scenes to 75% for crowded environments.

When waypoint similarity meets or exceeds the applicable threshold, the system progresses to the next waypoint in sequence, providing clear audio guidance commands such as "continue straight," "turn left," or "turn right." The system tracks consecutive navigation failures, activating a recovery system after five successive failures that requires manual capture of three directional frames for location rediscovery through majority voting analysis. Upon reaching the final destination, the system calculates relative directional positioning and provides comprehensive audio announcements before automatically terminating navigation and resetting all operational states.

4.6 Voice Assistant

In this project, the voice assistant is implemented to operate as a sophisticated hands-free interface that enables natural speech control of the navigation app. The system begins with continuous background wake word detection using Picovoice Porcupine, listening specifically for the custom wake word "Hey Navi". When detected, it provides haptic feedback and switches from wake word detection to active speech recognition using Flutter's speech_to_text package, with an 8-second listening timeout and real-time partial result feedback.

Once speech is captured, the system employs OpenAI's GPT-4.1-nano for intelligent natural language understanding, parsing commands like "Where am I?", "Find my location", "Take me to the library", "What routes are available?", or “Why do you think this is the location?” and mapping them to specific intents such as localization, navigation, or explanation requests. If GPT processing fails, it gracefully falls back to keyword-based parsing for reliability. The recognized commands are then executed through coordination with core services, including the embeddings service for image processing, the Supabase service for database access, the navigation service for route guidance, and the position localisation service for location identification.

During execution, the system maintains various session states from idle wake word detection through listening, processing, executing, and speaking phases, providing audio feedback through Flutter TTS with a deliberately slow 0.5x speech rate for clarity. Users can inquire about their current location, request relocalization through enhanced camera scanning, navigate to destinations by voice, list available routes, or ask for explanations of location identification reasoning. The assistant handles various error conditions like network issues, microphone permissions, timeouts, and device states through comprehensive error handling with automatic retry mechanisms and clear audio recovery instructions.

4.7 Database Design

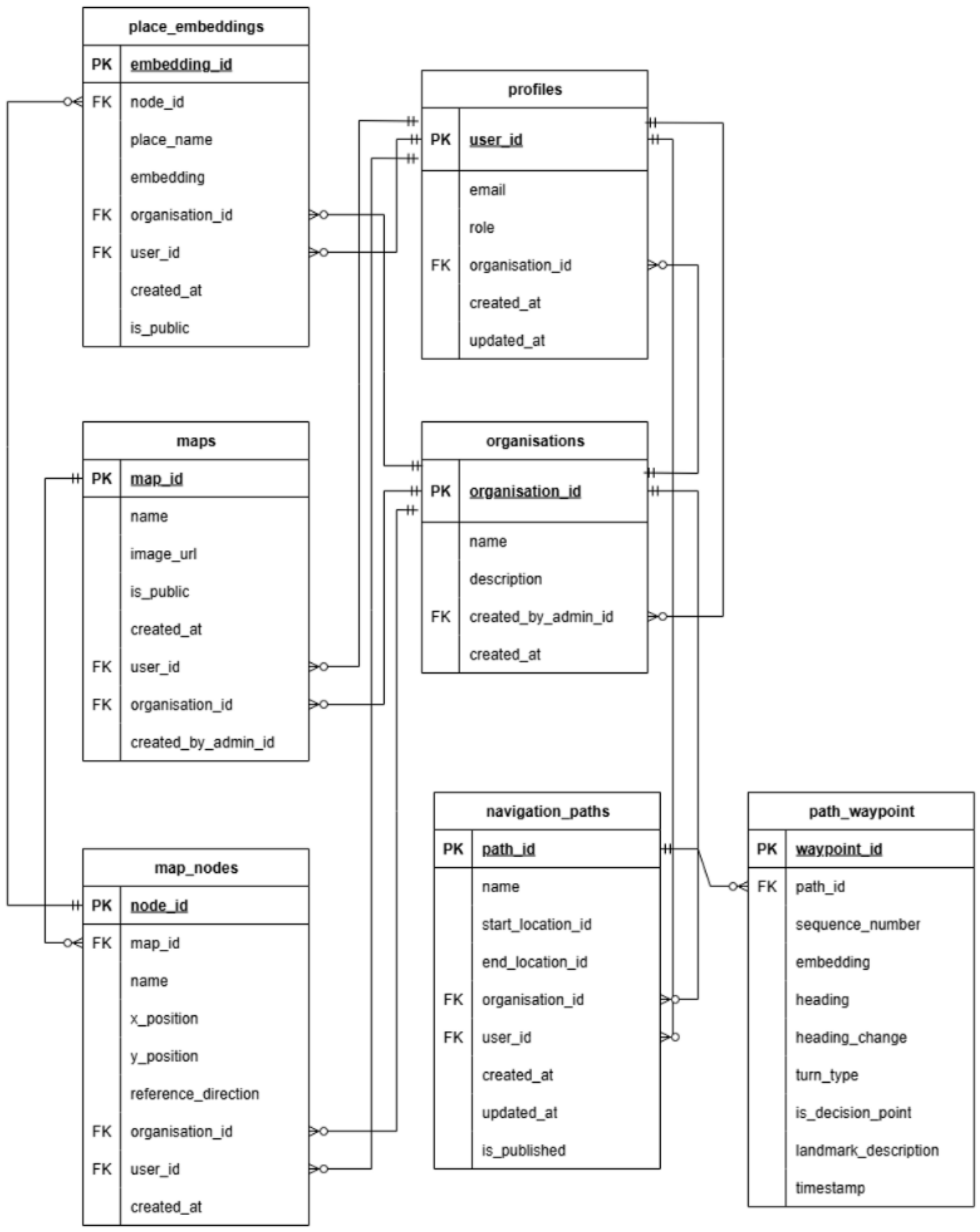


Figure 4.7.1 Database ERD Diagram

Figure 4.7.1 presents the Entity Relationship Diagram (ERD) for the indoor navigation system database. The design captures the essential entities, attributes, and relationships needed to support route recording, place recognition, and navigation functionalities.

The **profiles** table manages user information, where each user is identified by a unique user_id. Attributes include email, role, and references to their associated organisation_id. This allows the system to distinguish between different roles (e.g., administrator or regular user) and maintain organisational ownership of data.

The **organisations** table stores details of each organisation, with organisation_id as the primary key. It contains attributes such as name, description, and created_by_admin_id, ensuring that all maps, nodes, and navigation data can be grouped under organisational control. The relationship between profiles and organisations allows multiple users to belong to the same organisation, while administrative ownership is preserved.

The **maps** table represents uploaded maps of indoor environments. Each map is uniquely identified by map_id and contains attributes such as name, image_url, is_public, and creation metadata (created_at, user_id, organisation_id). Maps are created by administrators and linked to organisations, ensuring controlled access and visibility.

Associated with maps are the **map_nodes**, which define key locations or landmarks within a map. Each node is identified by node_id and contains attributes such as name, spatial coordinates (x_position, y_position), and reference_direction. Nodes are directly linked to maps and organisations, and they form the structural basis for route creation.

The **place_embeddings** table stores vision-based embeddings of places for recognition purposes. Each embedding, identified by embedding_id, is associated with a specific node_id. It contains the embedding vector, place_name, ownership metadata (organisation_id, user_id), and a visibility flag (is_public). These embeddings are generated using the DINOv2 model and are used for matching and localisation during navigation.

The **navigation_paths** table manages the routes between nodes. Each path is uniquely identified by path_id and contains attributes such as name, start_location_id, end_location_id, and ownership metadata (organisation_id, user_id). It also includes status attributes such as created_at, updated_at, and is_published. This enables administrators to create and manage multiple navigation paths within a given environment.

Each navigation path consists of multiple **path_waypoints**, which describe the detailed steps along the route. The waypoint_id serves as the primary key, while attributes such as sequence_number, embedding, heading, heading_change, and turn_type preserve the navigation context. Additional fields such as is_decision_point, landmark_description, and timestamp allow for enhanced guidance and route reconstruction.

CHAPTER 5 SYSTEM IMPLEMENTATION

This section presents the design specifications, including hardware specifications, software specifications, settings and configuration, system operation, and implementation issues and challenges.

5.1 Hardware Setup

Description	Specifications
Model	Acer Predator Helios Neo 16
Processor	Intel Core i7-13600HX
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 4060 8GB GDDR6
Memory	16GB DDR5 RAM
Storage	1TB SSD

Table 5.1.1 Specification of Laptop

Description	Specifications
Model	Oppo Reno 14
Processor	MediaTek Dimensity 8350
Operating System	Android 15, ColorOS15
Graphic	Mali-G615 MC6
Memory	12GB RAM
Storage	256GB SSD

Table 5.1.2 Specification of Mobile Phone

5.2 Software Setup

5.2.1 Development Environment

- Installed **Flutter** SDK 3.32.6 with Dart 3.8.1.
- IDE: **Visual Studio Code** 1.103.2 with Flutter and Dart extensions.

5.2.2 Backend Environment

- Installed Python 3.10+ for running **FastAPI** server.
- Created virtual environment using venv / conda.

5.2.3 Database Setup

- Created **Supabase** project (PostgreSQL backend).
- Enabled pgvector extension for vector search.

5.2.4 Version Control

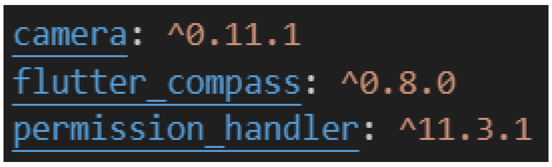
- Installed **Git** 2.45.2 to set up a Git repository for project source code.
- Used **GitHub** for version management.

5.3 Settings and Configuration

5.3.1 Flutter Dependencies

16 Flutter dependencies were added in the **pubspec.yaml** file to support the implementation of the indoor navigation app. The following describes the version and purpose of each dependency:

Core Navigation Functionality

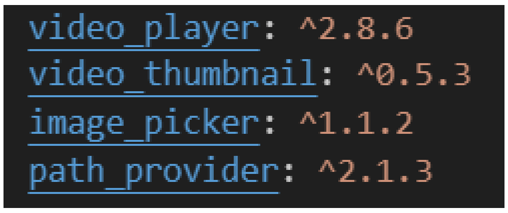


```
camera: ^0.11.1
flutter_compass: ^0.8.0
permission_handler: ^11.3.1
```

Figure 5.3.1.1 Core Navigation Functionality Packages

- camera (0.11.1): enables access to the device camera for real-time frame capture.
- flutter_compass (0.8.0): provides compass heading for orientation detection.
- permission_handler (11.3.1): manages runtime permissions for camera, storage, and sensors.

Media Handling

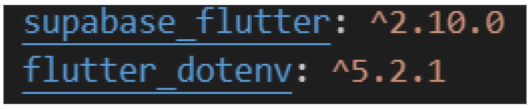


```
video_player: ^2.8.6
video_thumbnail: ^0.5.3
image_picker: ^1.1.2
path_provider: ^2.1.3
```

Figure 5.3.1.2 Media Handling Packages

- video_player (2.8.6): used for video playback within the application.
- video_thumbnail (0.5.3): generates thumbnails from video recordings.
- image_picker (1.1.2): allows selection of images and videos from the gallery or camera.
- path_provider (2.1.3): retrieves device file system directories for storing temporary data.

Backend Integration

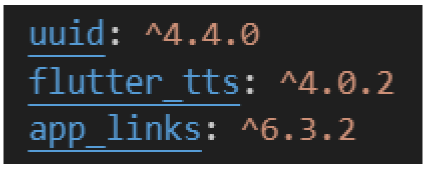


```
supabase_flutter: ^2.10.0
flutter_dotenv: ^5.2.1
```

Figure 5.3.1.3 Backend Packages

- supabase_flutter (2.10.0): connects the application to the Supabase backend for authentication and database operations.
- flutter_dotenv (5.2.1): loads environment variables from a .env file for secure configuration.

Utilities

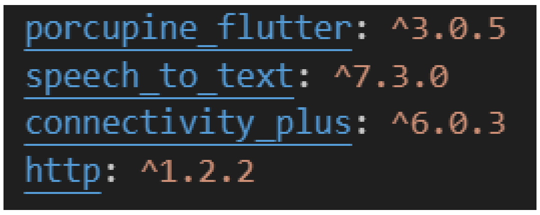


```
uuid: ^4.4.0
flutter_tts: ^4.0.2
app_links: ^6.3.2
```

Figure 5.3.1.4 Utilities Packages

- uuid (4.4.0): generates unique identifiers for records such as nodes and edges.
- flutter_tts (4.0.2): provides text-to-speech support for voice guidance.
- app_links (6.3.2): handles app deep linking and navigation redirection.

Voice Assistant

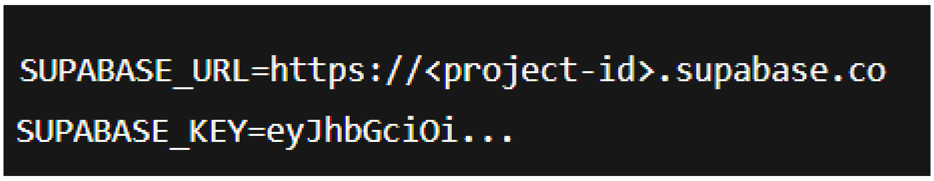


```
porcupine_flutter: ^3.0.5
speech_to_text: ^7.3.0
connectivity_plus: ^6.0.3
http: ^1.2.2
```

Figure 5.3.1.5 Voice Assistant Packages

- porcupine_flutter (3.0.5) → Wake word detection (e.g., "Hey Siri", "OK Google").
- speech_to_text (7.3.0) → Converts spoken words into text.
- connectivity_plus (6.0.3) → Checks internet connection status (WiFi, mobile, offline).
- http (1.2.2) → Makes HTTP requests (GET, POST, etc.) to web servers.

Assets



```
SUPABASE_URL=https://<project-id>.supabase.co
SUPABASE_KEY=eyJhbGciOi...
```

Figure 5.3.1.6. env Format

- .env file: Stores sensitive configuration details such as Supabase project URL and API keys, ensuring security and flexibility across environments.

5.3.2 FastAPI Server Configuration

The backend was implemented using Python 3.10+ with dependencies listed in the **requirements.txt** file. These libraries enable the FastAPI server, AI/ML model integration, and supporting utilities for the indoor navigation system.

Libraries and Dependencies	Version	Description
Core Web & API Infrastructure		
fastapi	0.116.1	Web framework for building REST API endpoints.
uvicorn	0.23.1	ASGI server to run the FastAPI application.

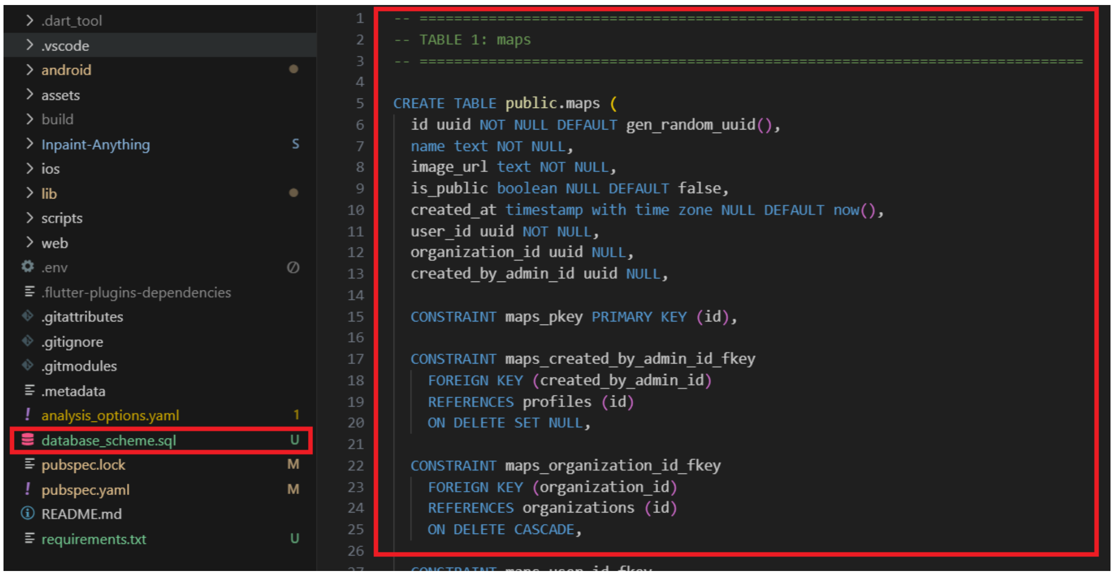
python-multipart	0.0.6	Handles file uploads via multipart form data.
Image Processing and Computer Vision		
PIL (Pillow)	11.1.0	Image processing library for basic manipulations.
numPy	2.0.1	Numerical computations and array handling.
AI/ML Frameworks		
torch	2.5.1	PyTorch framework for running models.
torchvision	0.20.1	PyTorch computer vision utilities.
torchaudio	2.5.1	PyTorch audio processing support.
transformers	4.56.0	For loading Dinov2 models.
ultralytics	8.3.191	YOLOv8 for human detection and segmentation.
diffusers	0.35.1	Stable Diffusion 2.0 for people inpainting
HTTP and Networking		
requests	2.32.5	HTTP requests for API communication.

Table 5.3.2.1 Python Libraries

5.3.3 Supabase Configuration

Step 1: Create Tables with SQL Script

- Log in to the Supabase project.
- Navigate to the SQL Editor in the dashboard.
- Copy the SQL script from database_scheme.sql as shown in Figure 5.3.3.1 and paste it into the SQL editor as shown in Figure 5.3.3.2 and run the SQL script to create the required tables.



```sql
-- ============================================================================
-- TABLE 1: maps
-- ============================================================================

CREATE TABLE public.maps (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL,
  image_url text NOT NULL,
  is_public boolean NULL DEFAULT false,
  created_at timestamp with time zone NULL DEFAULT now(),
  user_id uuid NOT NULL,
  organization_id uuid NULL,
  created_by_admin_id uuid NULL,

  CONSTRAINT maps_pkey PRIMARY KEY (id),

  CONSTRAINT maps_created_by_admin_id_fkey
    FOREIGN KEY (created_by_admin_id)
    REFERENCES profiles (id)
    ON DELETE SET NULL,

  CONSTRAINT maps_organization_id_fkey
    FOREIGN KEY (organization_id)
    REFERENCES organizations (id)
    ON DELETE CASCADE,

```

Figure 5.3.3.1 database scheme.sql

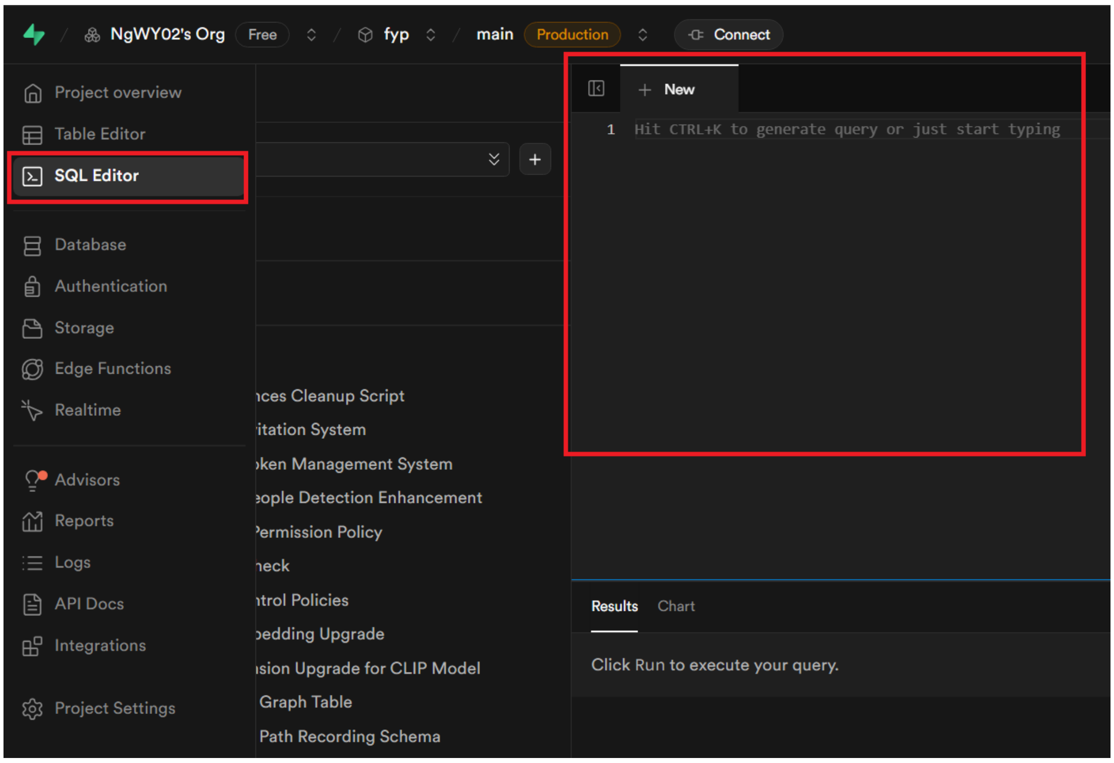
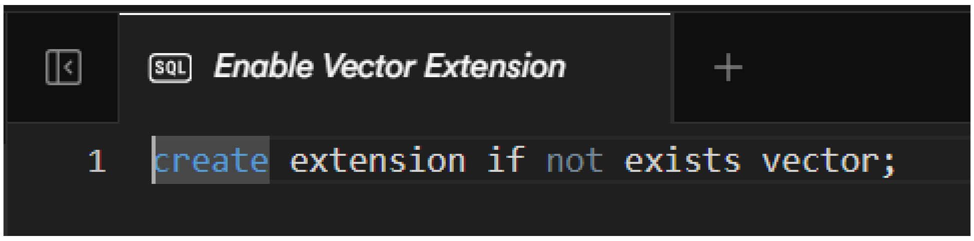


Figure 5.3.3.2 Supabase SQL editor

Step 2: Enable pgvector Extension

- In the SQL editor, run the command:



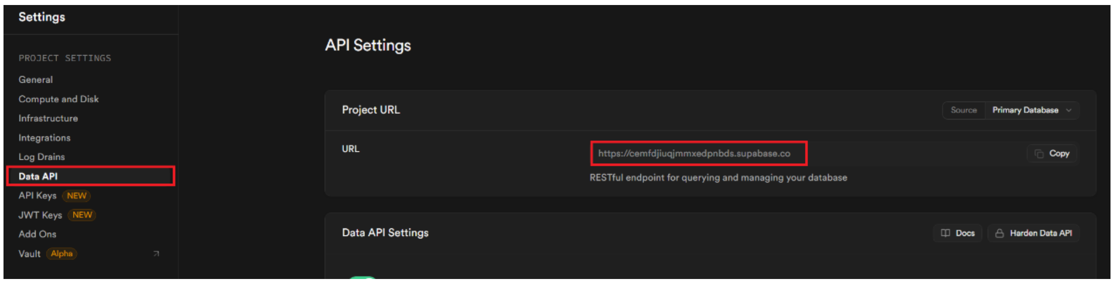
```
create extension if not exists vector;
```

Figure 5.3.3.3 pgvector Extension

- This enables **vector search** for storing and querying image embeddings.

Step 3: Obtain API Keys

- Go to Project Settings → API in the Supabase dashboard.
- Copy the Project URL and Anon/Public API Key.
- These will be used in the Flutter app (.env file) and the FastAPI backend.
- Figure 5.3.3.4 and Figure 5.3.3.5 show the API key section in the Supabase dashboard.



API Settings

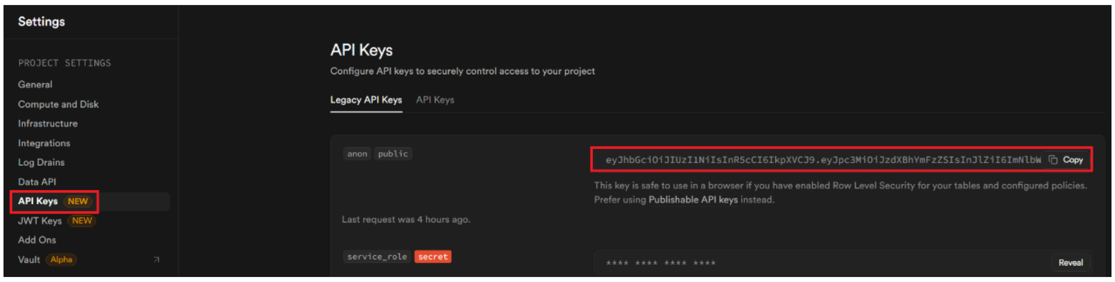
Project URL

URL: https://cemfdjiuqjmmxedpnbds.supabase.co

RESTful endpoint for querying and managing your database

Data API Settings

Figure 5.3.3.4 Project URL



API Keys

Configure API keys to securely control access to your project

anon public: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImNlbW

This key is safe to use in a browser if you have enabled Row Level Security for your tables and configured policies. Prefer using Publishable API keys instead.

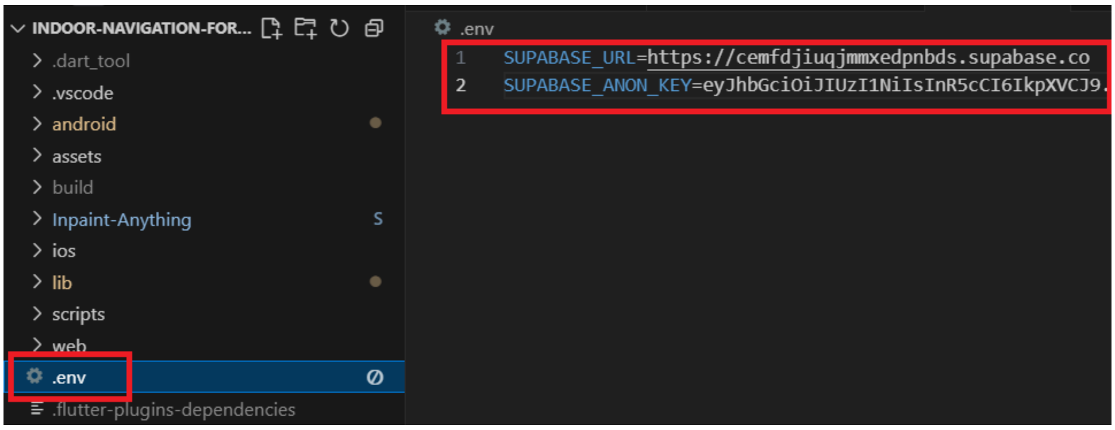
Last request was 4 hours ago.

service_role secret: **** **** **** ****

Figure 5.3.3.5 Public API Key

Step 4: Connect Backend to Database

- Store the Supabase project URL and API key in the .env file of the Flutter app.
- Example .env file configuration:



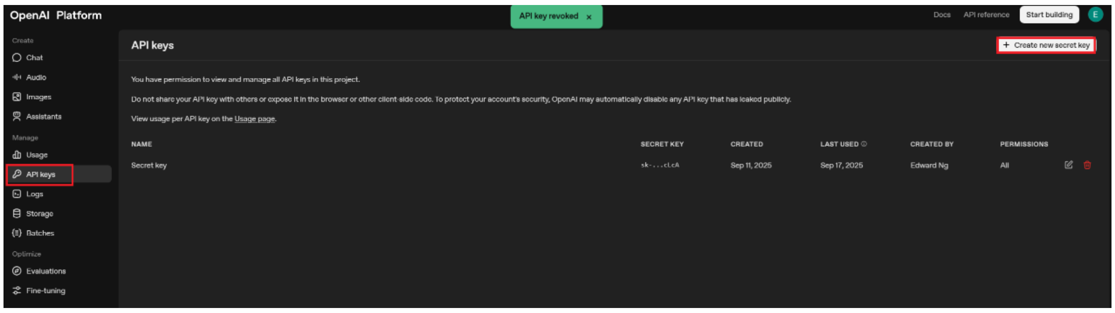
```
SUPABASE_URL=https://cemfdjiuqjmmxedpnbds.supabase.co
SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
```

Figure 5.3.3.6 .env file configuration

5.3.4 VLM GPT Configuration

Step 1: Go to https://platform.openai.com/api-keys

- Click Create New Secret Key button.



API keys

You have permission to view and manage all API keys in this project.

Do not share your API key with others or expose it in the browser or other client-side code. To protect your account's security, OpenAI may automatically disable any API key that has leaked publicly.

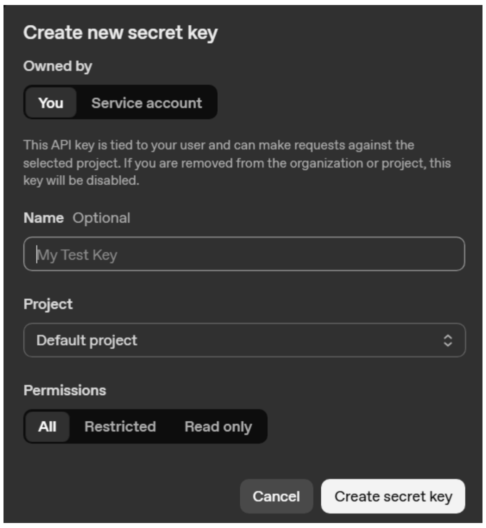
View usage per API key on the Usage page.

NAME	SECRET KEY	CREATED	LAST USED	CREATED BY	PERMISSIONS
Secret key	sk-...cLcA	Sep 11, 2025	Sep 17, 2025	Edward Ng	All

Figure 5.3.4.1 GPT API Web Page

Step 2: Create New Secret Key

- Enter the project name (optional)

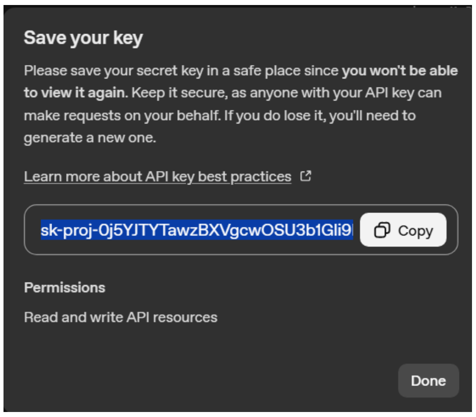


This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.

Figure 5.3.4.2 Create New Secret Key

Step 3: Copy the Secret Key

- The secret key must be saved properly because it won't be able to be viewed again.



Please save your secret key in a safe place since **you won't be able to view it again.** Keep it secure, as anyone with your API key can make requests on your behalf. If you do lose it, you'll need to generate a new one.

sk-proj-0j5YJTYTawzBXVgcwOSU3b1Gli9

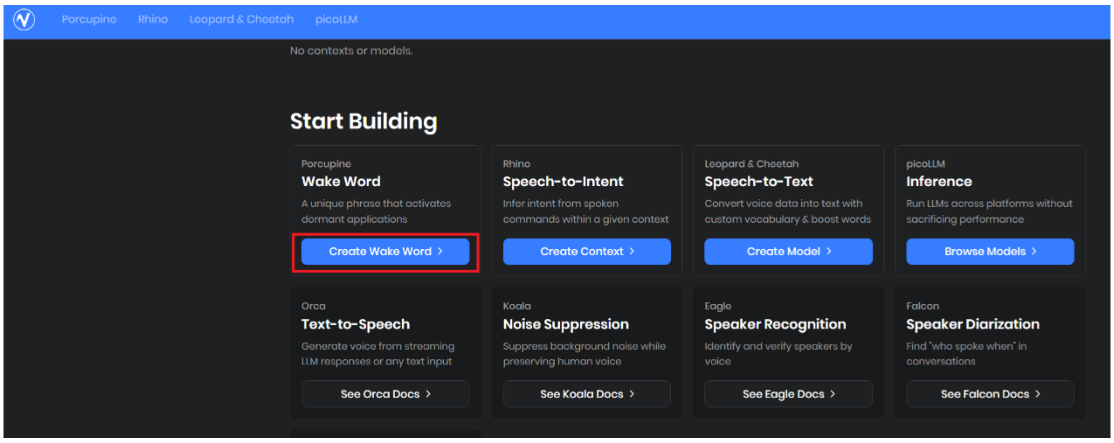
Read and write API resources

Figure 5.3.4.3 Secret Key

5.3.5 Wake Word Configuration

Step 1: Sign up & Get Access

- Go to Picovoice website
- Click Create Wake Word button



No contexts or models.

Start Building

Porcupine
Wake Word
A unique phrase that activates dormant applications

Rhino
Speech-to-Intent
Infer intent from spoken commands within a given context

Leopard & Cheetah
Speech-to-Text
Convert voice data into text with custom vocabulary & boost words

picoLLM
Inference
Run LLMs across platforms without sacrificing performance

Orca
Text-to-Speech
Generate voice from streaming LLM responses or any text input

Koala
Noise Suppression
Suppress background noise while preserving human voice

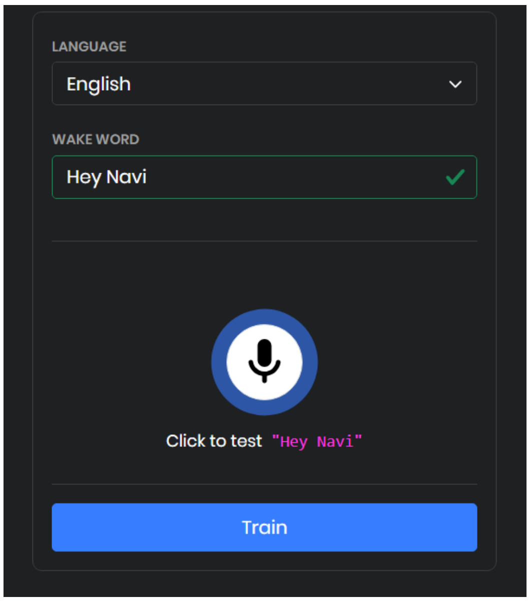
Eagle
Speaker Recognition
Identify and verify speakers by voice

Falcon
Speaker Diarization
Find "who spoke when" in conversations

Figure 5.3.5.1 Picovoice Web Page

Step 2: Create Custom Wake Word

- Choose English Language
- Type the wake word "Hey Navi"
- Click the Train button



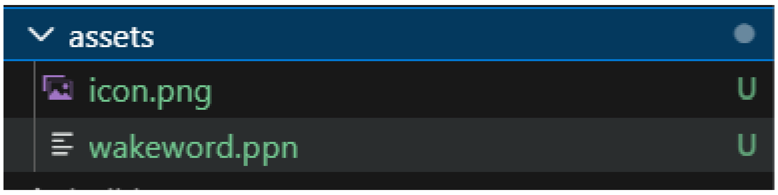
LANGUAGE: English

WAKE WORD: Hey Navi

Figure 5.3.5.2 Create Custom Wake Word

Step 3: Download wake word .ppn

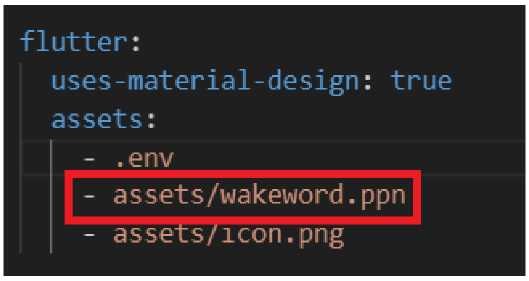
- Put the .ppn file into the assets folder



```
∨ assets
    icon.png          U
    wakeword.ppn      U
```

Figure 5.3.5.3 .ppn File in Assets Folder

Step 4: Put this line into the pubspec.yaml file



```
flutter:
  uses-material-design: true
  assets:
    - .env
    - assets/wakeword.ppn
    - assets/icon.png
```

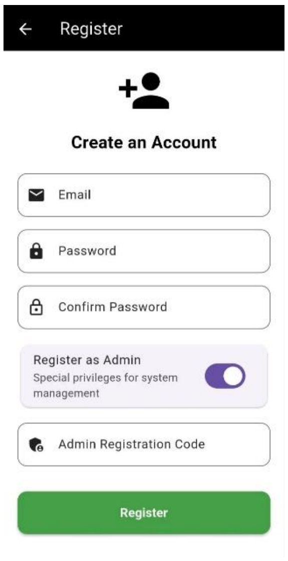
Figure 5.3.5.4 pubspec.yaml File

5.4 System Operation

This section describes the system's operational aspects regarding the graphical user interface (GUI) design and the functionality of various screens.

5.4.1 User Authentication and Account Management

Register Page



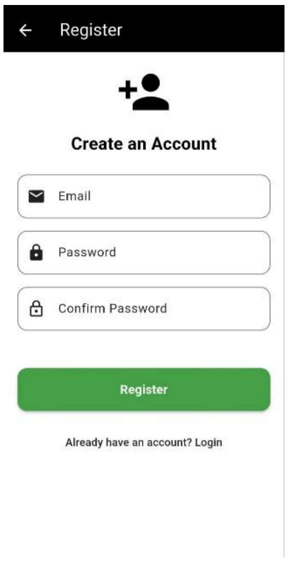


Figure 5.4.1.1 Admin Register Screen　　*Figure 5.4.1.2 User Register Screen*

There are two different types of registration pages in the system. If a user wants to register as an admin, they must provide a valid admin registration code during the sign-up process. Regular users can register through the standard user registration page without requiring a code. After completing the registration form, a verification link is sent to the user's email. By clicking the link, the account is verified and the registration process is successfully completed.

Login Page

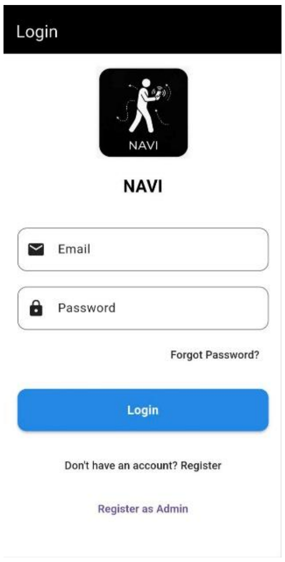
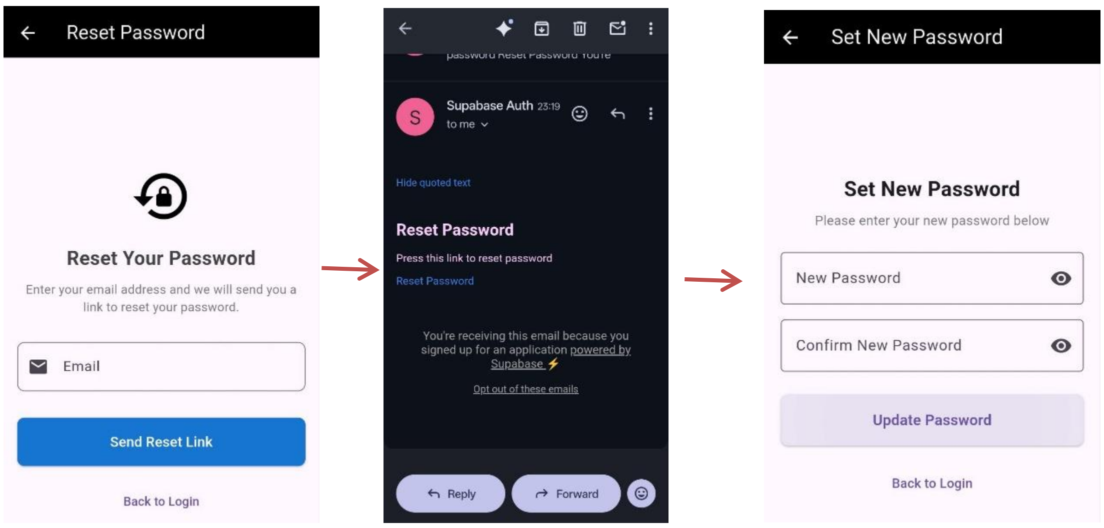


Figure 5.4.1.3 Login Screen

This figure displays the system's login screen. Proper error handling mechanisms are implemented to ensure that invalid inputs, such as incorrect credentials or empty fields, are detected and clearly communicated to the user.

Forgot/Reset Password Page



Enter your email address and we will send you a link to reset your password.

Reset Password

Press this link to reset password

Reset Password

You're receiving this email because you signed up for an application powered by Supabase ⚡

Opt out of these emails

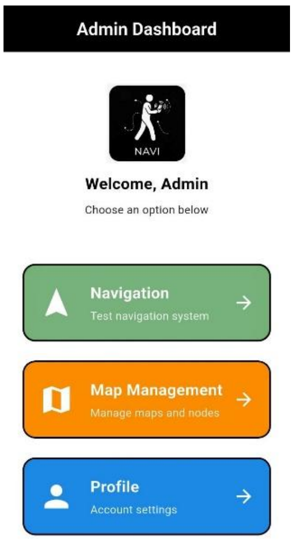
Please enter your new password below

Figure 5.4.1.4 Forgot/Reset Password Screen

To reset a password, the user is required to enter their registered email address to receive a reset link. By clicking on the link sent to their email, the user will be redirected to the reset password page, where a new password can be created securely.

5.4.2 Admin Only Features

Administrators have access to additional functions for managing the system. They can upload a map, add nodes, connect nodes, edit nodes/edges, and delete the map by clicking the specific map.



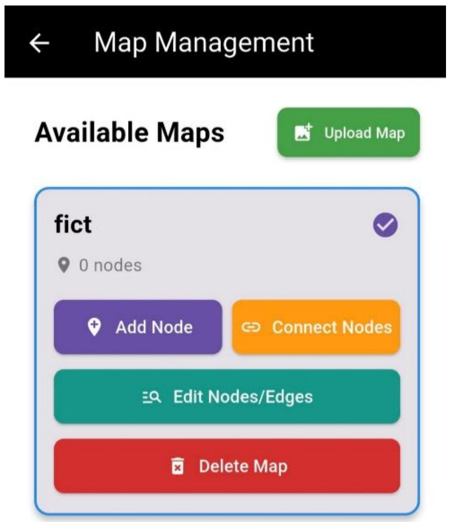


Figure 5.4.2.1 Admin Home Screen *Figure 5.4.2.2 Map Management Screen*

Upload Map Screen

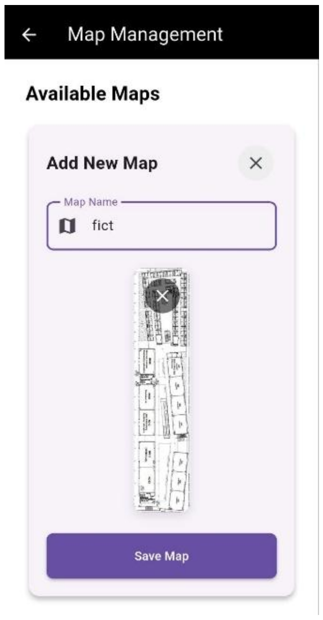
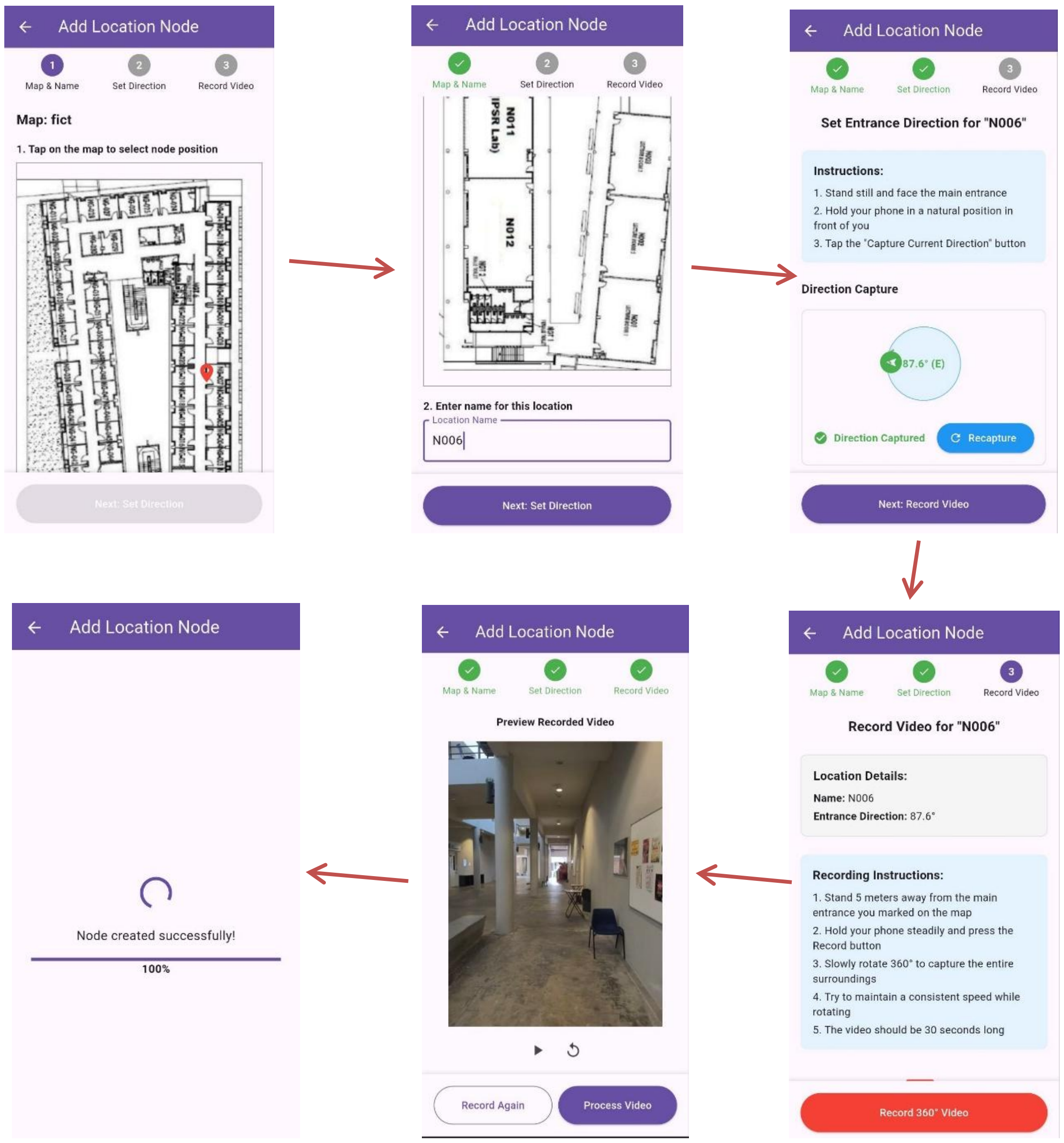


Figure 5.4.2.3 Upload Map Screen

Admin can add a new map and save the map name. The map will be stored in the Supabase storage.

Add Node Screen


Add Location Node
1 Map & Name 2 Set Direction 3 Record Video
Map: fict
1. Tap on the map to select node position
Next: Set Direction

Add Location Node
Map & Name 2 Set Direction 3 Record Video
2. Enter name for this location
Location Name
N006
Next: Set Direction

Add Location Node
Map & Name Set Direction 3 Record Video
Set Entrance Direction for "N006"
Instructions:
1. Stand still and face the main entrance
2. Hold your phone in a natural position in front of you
3. Tap the "Capture Current Direction" button
Direction Capture
87.6° (E)
Direction Captured
Recapture
Next: Record Video

Add Location Node
Map & Name Set Direction 3 Record Video
Record Video for "N006"
Location Details:
Name: N006
Entrance Direction: 87.6°
Recording Instructions:
1. Stand 5 meters away from the main entrance you marked on the map
2. Hold your phone steadily and press the Record button
3. Slowly rotate 360° to capture the entire surroundings
4. Try to maintain a consistent speed while rotating
5. The video should be 30 seconds long
Record 360° Video

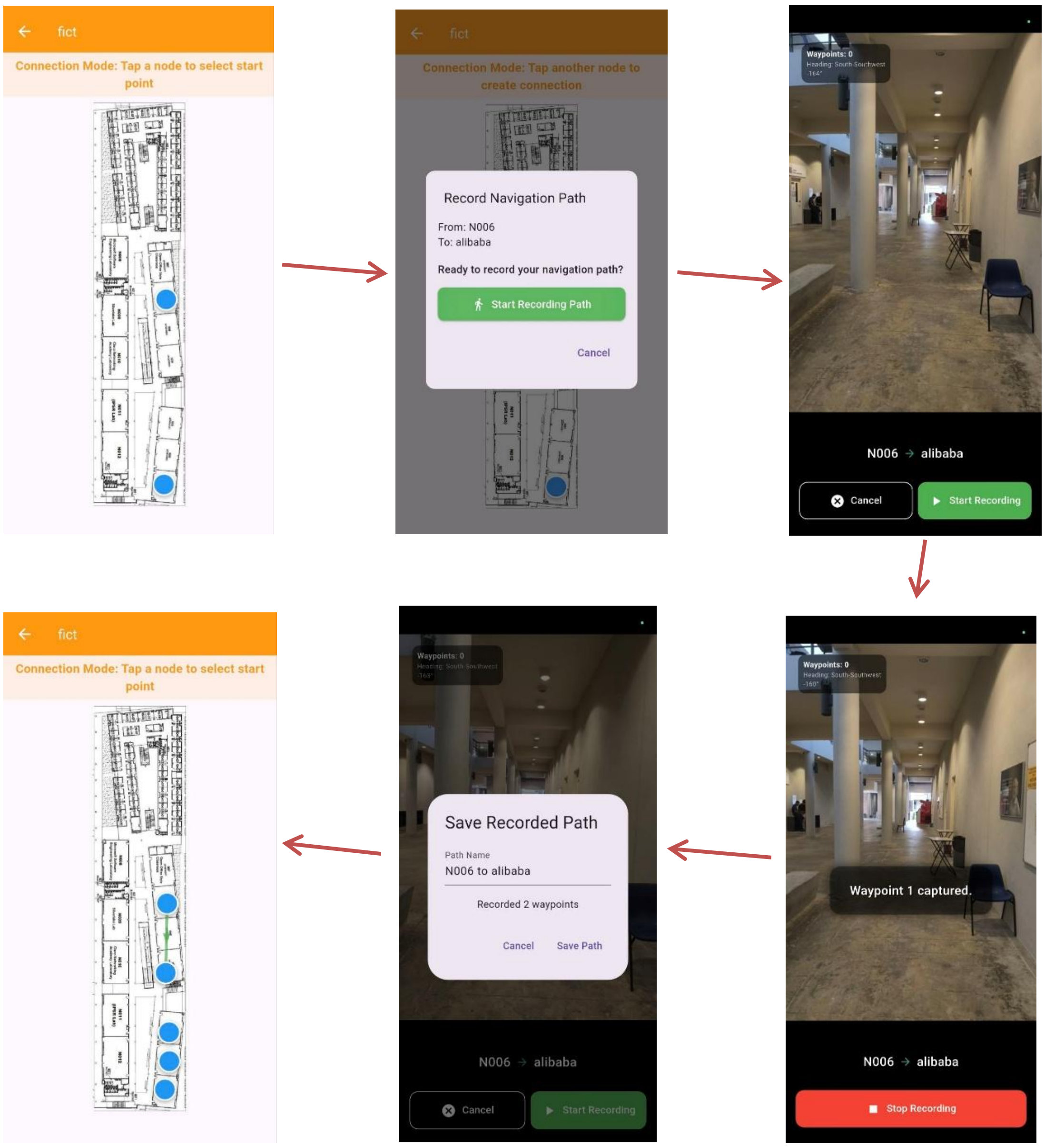
Add Location Node
Map & Name Set Direction Record Video
Preview Recorded Video
Record Again
Process Video

Add Location Node
Node created successfully!
100%

Figure 5.4.2.4 Add Node Procedure

To complete the process of adding a node, the admin should first tap on the map to select the node position. Based on Figure 5.4.2.4, a location icon is added on the N006. If the map is too big, the admin needs to scroll down to insert the location name. Then, the admin can only proceed to set the direction for the location. The admin is asked to stand still and face the main door of the location. Once the direction is captured, the user's current heading or orientation will be recorded and saved into the database as the reference direction. Next, the location details and recording instructions are displayed to ensure a better quality for the frame extraction and embedding process. The admin can then start recording a 360° video of the location. After the recording is completed, the admin can choose to record again or directly process the video. A preview of the recorded video is also provided to see the result. Upon processing, the video will be extracted frame by frame every 0.5 seconds and each frame will be processed by the embedding model DINOv2 to generate corresponding embeddings for storage and later recognition. Each frame will also be stored in the bucket for image comparison by VLM verification.

Edge Connection Screen



Connection Mode: Tap a node to select start point

Connection Mode: Tap another node to create connection

Record Navigation Path

From: N006
To: alibaba

Ready to record your navigation path?

Waypoints: 0
Heading: South Southwest
164°

N006 → alibaba

Waypoint 1 captured.

Save Recorded Path

Path Name
N006 to alibaba

Recorded 2 waypoints

Figure 5.4.2.5 Edge Connection Procedure

The user first selects two nodes and then clicks the Start Recording Path button. The system then proceeds to the camera screen, where the user can press the Start Recording button to begin capturing. As waypoints are recorded, audio feedback is provided (e.g., "Waypoint 1 captured"). Once the destination is reached, the user clicks Stop Recording, and the total number of waypoints will be displayed. Finally, by selecting Save Path, the edge is successfully connected.

Edit Nodes/Edges Screen

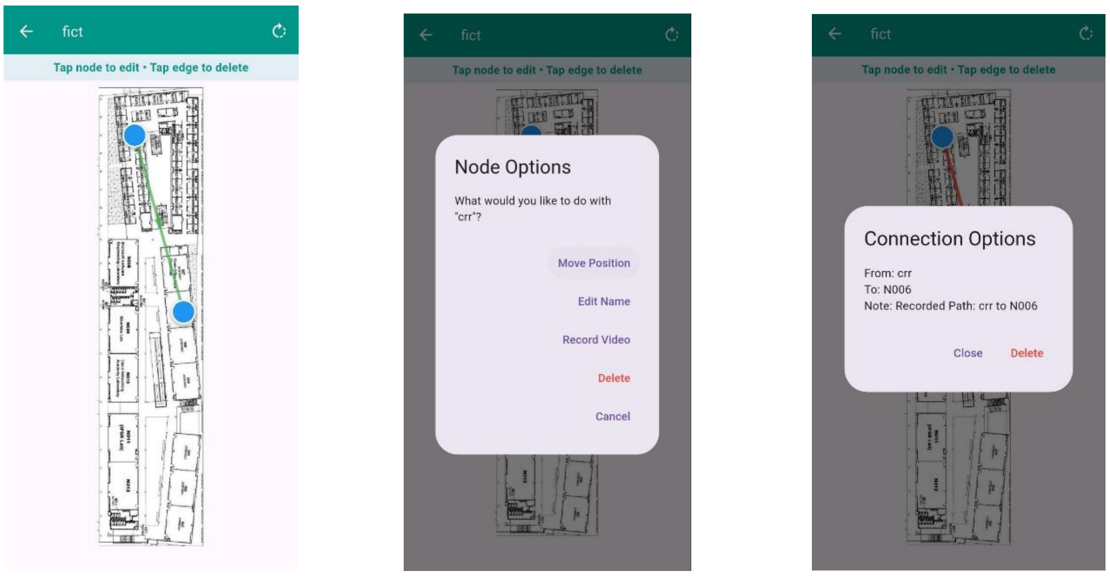
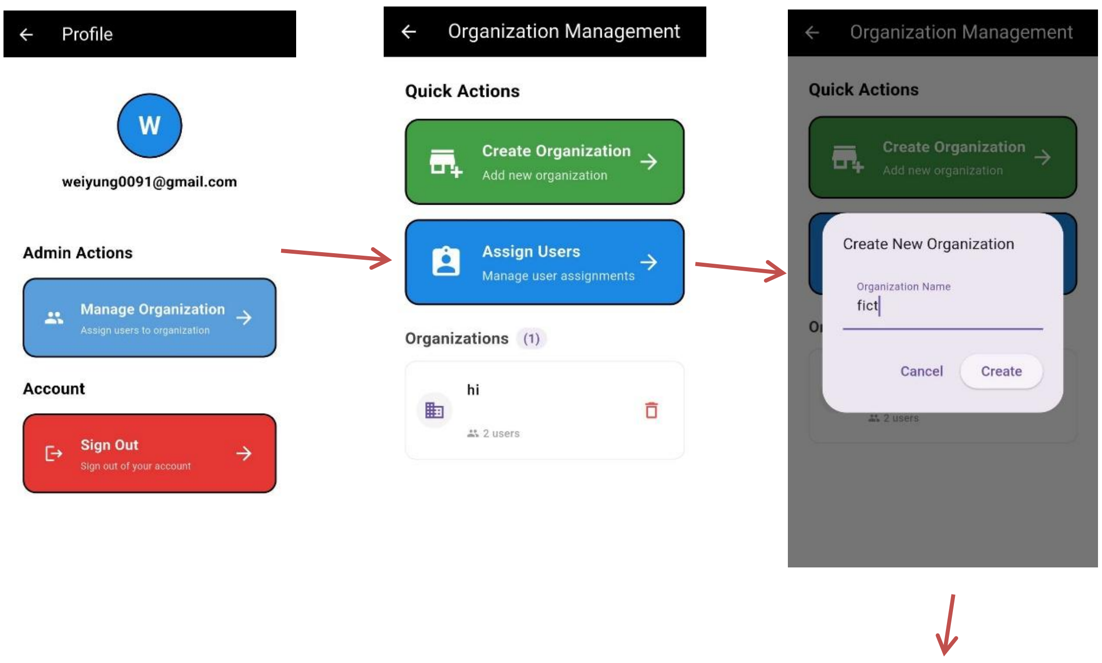
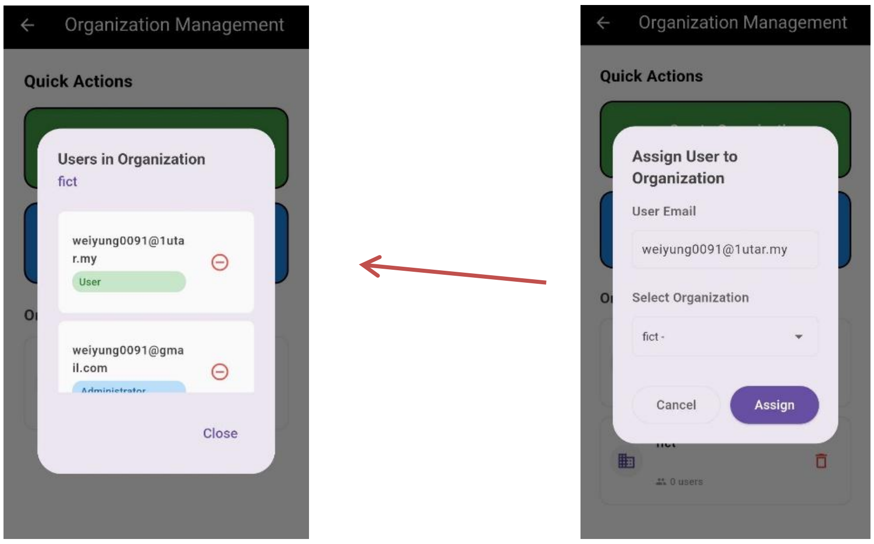


Figure 5.4.2.6 Edit Nodes & Delete Edges Screen

If the user wants to edit a node, they can simply tap on it. A list of options will appear, allowing the user to move the node's position, edit the node's name, record a video again, or delete the node. Similarly, if the user wants to delete an edge, they can click on the line connecting the nodes and select the delete option.

Organization Management





weiyung0091@1utar.my
weiyung0091@gmail.com

User Email
weiyung0091@1utar.my
Select Organization
fict -

Figure 5.4.2.7 Organization Management

To create an organization, the user navigates to the Profile screen and clicks the Manage Organization button. From there, the user can select Create Organization to set up a new organization. By clicking on Assign Users, the admin can invite users to join the organization by entering their email addresses. This allows the invited users to access the admin's data for localization and navigation. Additionally, the admin can delete the organization or remove users from it when necessary.

5.4.3 Common Users Features

These features are accessible to all users, including VIP users and administrators.

Localisation Screen

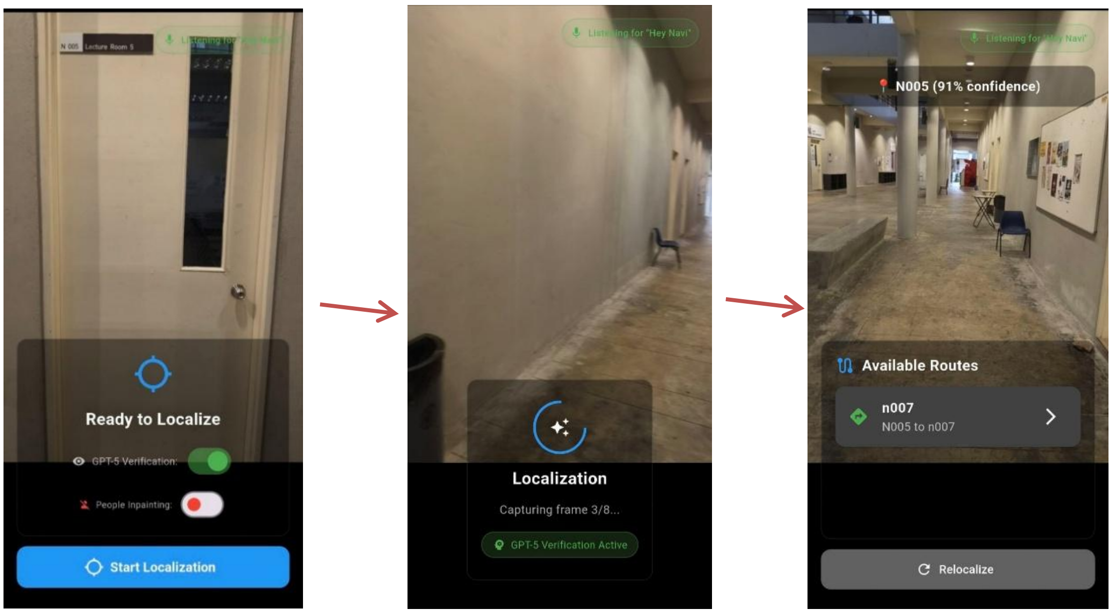
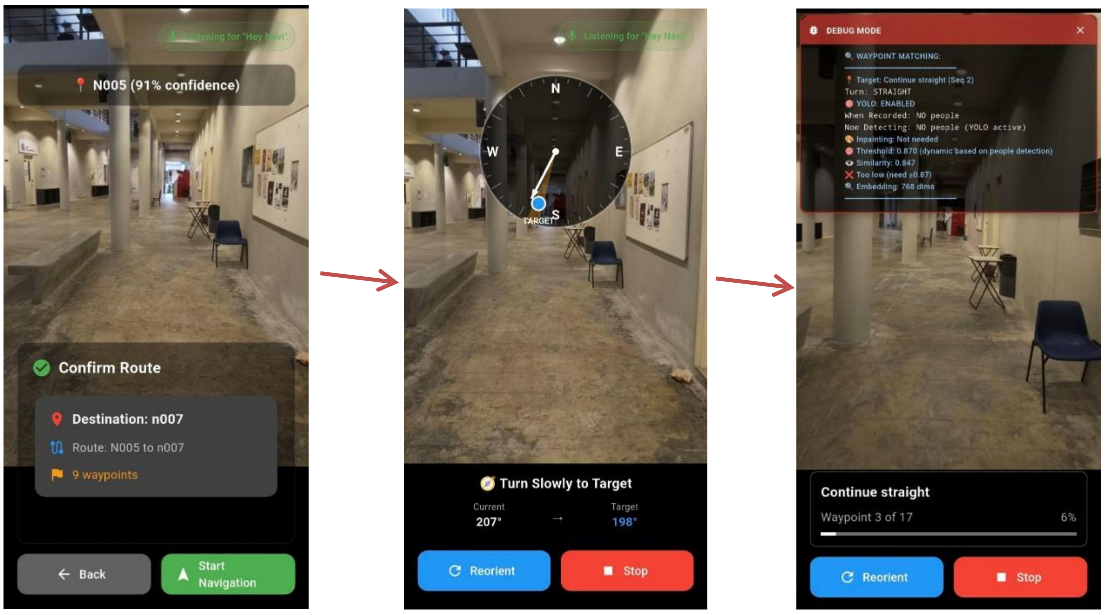


Figure 5.4.3.1 Localisation Screen

On the Localization Screen, users have several options before starting the localization process. They may choose to disable GPT-4 verification, allowing the system to rely solely on embedding comparison to identify the location without sending images for image-pair verification. Users can also enable people inpainting, which removes people from the captured images during localization and navigation. Localization can then be initiated either by clicking the Start Localization button or by using voice commands.

Once started, the user scans the surroundings to capture frames. Each frame is converted into embeddings and compared against stored embeddings to find the highest similarity match. The corresponding stored image is then paired with the captured frame and if GPT-4 verification is enabled, it will be sent for confirmation. If the similarity score exceeds the threshold value, the detected location is displayed along with the available routes. The user also has the option to perform relocalization if needed.

Navigation Screen



```
WAYPOINT MATCHING:
Target: Continue straight (Seq 2)
Turn: STRAIGHT
YOLO: ENABLED
When Recorded: NO people
Now Detecting: NO people (YOLO active)
Inpainting: Not needed
Threshold: 0.870 (dynamic based on people detection)
Similarity: 0.847
Too low (need ≥0.87)
Embedding: 768 dims
```

Figure 5.4.3.2 Navigation Screen

To navigate to a destination, a path must first be recorded by the admin so that the available routes can be displayed to the user. After selecting and confirming a route, the user can start navigation. The app begins with an orientation phase to ensure that the user is facing the correct initial direction before departure. Once orientation is complete, the system provides real-time audio guidance, matching captured frames with stored embeddings to direct the user step by step toward the destination. Once the user reaches the destination, the system will produce the audio “You have reached the destination. Alibaba is on your left” by calculating the angular difference between the current degrees and the location degrees. For example, the user is facing north (0°) right now, but Alibaba’s entrance direction was captured as northwest (-45°) during node creation, the system would announce "Alibaba is on your left" because the angular difference is -45°.

The third image shows the debug mode screen in the admin navigation interface. It won't appear in the VIP navigation interface. So, it provides detailed information such as the current waypoint, turn instructions, YOLO detection status, inpainting activity, similarity scores, dynamic thresholds, and embedding information. This debug view helps the admin monitor the navigation process, verify the accuracy of frame matching, and ensure that the system functions correctly during testing and real-time navigation. Also, it can be hidden if it blocks the view.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

This chapter discusses the testing setup and results, system testing and performance metrics, project challenges, and objectives evaluation.

6.1 Testing Setup and Result

6.1.1 Localisation

Localisation using Embeddings Only

Test Case	Location	Expected Result	Actual Result	Pass/Fail
TC1	Alibaba Lab	Alibaba	Alibaba	Pass
TC2	Cisco Lab	Cisco	Cisco	Pass
TC3	Classroom Room001	Room001	Room001	Pass
TC4	Classroom Room003	Room003	Room001	Fail
TC5	Classroom Room005	Room005	Room003	Fail

Table 6.1.1.1 Localisation Result without Using VLM

Localisation using Embeddings and VLM

Test Case	Location	Expected Result	Actual Result	Pass/Fail
TC1	Alibaba Lab	Alibaba	Alibaba	Pass
TC2	Cisco Lab	Cisco	Cisco	Pass
TC3	Classroom Room001	Room001	Room001	Pass
TC4	Classroom Room003	Room003	Room003	Pass
TC5	Classroom Room005	Room005	Room005	Pass

Table 6.1.1.2 Localisation Result using Embeddings and VLM

The localisation module was evaluated using two approaches which are embeddings only and embeddings with Vision-Language Model (VLM) verification. In the first approach, embeddings generated by the DINOv2 model were compared against stored embeddings in Supabase using cosine similarity. This method achieved an accuracy of 60%, correctly identifying 3 out of 5 test locations. It was efficient, with an average response time of about 350 ms, making it highly responsive for real-time use. However, the system occasionally misclassified visually similar areas, such as confusing room001 with room003 and room005, which highlights the limitations of relying solely on visual embeddings in environments with repetitive structures.

When VLM verification was integrated into the pipeline, accuracy improved to 100%, correctly identifying all 5 test locations. The VLM not only validated the visual similarity between candidate locations but also leveraged additional capabilities such as OCR for reading signs, text labels, and room numbers. This allowed the system to differentiate between otherwise similar-looking spaces by interpreting contextual cues like "Room001" versus "Room003". The model also provided better semantic understanding of the scene, improving robustness in cases where the visual appearance alone was insufficient for precise localisation. The trade-off was a higher latency, averaging 1.5–2 seconds per localisation attempt, due to the extra verification step.

6.1.2 Navigation

Navigation Without People

Test Case	**Start → Destination**	**Expected Output**	**Actual Output**	**Pass/Fail**
TC1	Room001 → Cisco Lab	Straight → Left turn	Straight → Left turn	Pass
TC2	Room002 → Silver Lab	Straight → Right turn	Straight → Right turn	Pass
TC3	Room003 → Apple Lab	Continue Straight	Continue Straight	Pass
TC4	Room004 → Room005	Turn right → Turn left	Turn right→ Turn left	Pass
TC5	Room006 → Alibaba Lab	Off-route detection enabled	Off-route detection worked	Pass

Table 6.1.2.1 Navigation Result Without People

Navigation With People

Test Case	Condition	Start → Destination	Expected Output	Actual Output	Pass/Fail
TC3	Few people (3 persons)	Room003 → Apple Lab	Straight → Left turn	Straight → Left turn	Pass
TC4	Few people (3 persons)	Room004 → Room005	Continue Straight	Continue Straight	Pass
TC5	Moderate crowd	Room006 → Silver Lab	Straight → Right turn	Straight → Right turn	Pass
TC6	Moderate crowd	Apple Lab → Room007	Straight → Left → Straight	Frame Not Match, Miss Waypoint	Fail
TC7	Crowded area	Room002 → Cisco Lab	Straight → Left turn	People Block View, Frame Not Match	Fail
TC8	Crowded area	Room003 → Alibaba Lab	Straight → Left turn	Frame Not Match, Miss Waypoint	Fail

Table 6.1.2.2 Navigation Result with People

The navigation tests were carried out under two conditions: without people and with people in the environment. In the absence of people, all test cases produced accurate results, with the system successfully recognising frames and delivering correct turn-by-turn guidance. This shows that under static conditions, where the camera has a clear and unobstructed view of the environment, the navigation process is highly reliable. In contrast, navigation with people presented more difficulties. When only a few individuals were present, the system still managed to provide correct directions, though occasional delays occurred as the camera view was partially obstructed. However, as the density of people increased to moderate or crowded levels, navigation accuracy declined noticeably. Moving individuals often blocked key environmental features, causing the system to capture incomplete or mismatched frames, which in turn led to missed waypoints or incorrect turns.

To reduce this issue, a people inpainting model was integrated to remove humans from the frames and reconstruct the background. This method worked reasonably well when the occluded areas were small, as the inpainting process restored the background effectively and maintained embedding quality. Nonetheless, when large portions of the scene were blocked, the inpainting algorithm often generated unrealistic or mismatched textures, introducing noise into the embeddings. Consequently, while inpainting mitigates some of the challenges caused by people in the environment, it cannot fully guarantee stable and accurate navigation in highly dynamic or crowded spaces.

6.1.3 Voice Output Clarity

Test Case	Instruction	Clarity	Timing	Remarks
TC1	“Continue straight”	Clear	On-time	Works well in hallway
TC2	“Turn left”	Clear	On-time	No issue
TC3	“Destination reached”	Clear	On-time	No issue
TC4	“You are off route”	Clear	On-time	Helps recovery effectively
TC5	“Turn right”	Clear	On-time	No issue

Table 6.1.3.1 Voice Output Response Time

6.2 System Testing and Performance Metrics

6.2.1 Embedding Model

Model	Embedding Dim.	Findings/Remarks	Inference Time (ms/frame)
MobileNetV2	1280	Very fast but less accurate	**45 ms**
CLIP (ViT-L/14)	768	Matches even when frames are not every similar	380 ms
DINOv2-Base	768	More precise on visual similarity,	120 ms

		best overall balance	

Table 6.2.1.1 Embeddings Model Performance Metrics

The evaluation results highlight a trade-off between model efficiency and recognition reliability. **MobileNetV2**, a convolutional neural network (CNN), achieves the fastest inference time at 45 ms per frame with an embedding dimension of 1,280, making it highly suitable for mobile deployment. CNNs like MobileNetV2 are computationally efficient due to their convolutional layers, which excel at capturing local features such as edges, textures, and simple patterns. However, their reliance on local receptive fields limits their ability to capture long-range dependencies and global contextual information. This limitation can reduce robustness in indoor navigation, where visually similar areas or subtle spatial cues (e.g., identical corridors, partially occluded signs) are common. As a result, while MobileNetV2 is extremely fast, it tends to mislocalise in challenging indoor environments.

CLIP (ViT-L/14), based on a visual transformer (ViT) architecture, benefits from global self-attention mechanisms that model relationships across the entire image. This enables stronger semantic understanding of scenes, allowing the system to match locations even when frames are not visually identical. Such semantic robustness makes CLIP effective in scenarios with visual ambiguity. However, its inference time of 380 ms per frame introduces noticeable latency, which may hinder responsiveness during real-time navigation.

DINOv2-Base, also transformer-based, strikes the optimal balance between precision and efficiency. With an inference speed of 120 ms per frame, it remains feasible for mobile deployment while offering highly reliable embeddings for distinguishing visually similar places. Unlike CNNs, DINOv2-Base captures both fine-grained local details and broader spatial context, making it more resilient to variations in lighting, occlusion, and viewpoint. This balance of speed and robustness makes DINOv2-Base the most practical choice for real-time indoor navigation, particularly for visually impaired users, where recognition errors could compromise safety and usability.

6.2.2 Inpainting Model

Model	Avg. Time (s)	Removal Quality	Artifacts	Result
LaMa	0.8	Medium	Artifacts, mosaic pattern	Bad
SD 1.5	1.5	Moderate	Clean, a few artifacts	Moderate
SD 2.0	1.8	Good	Clean, realistic	Good
SD 3.5	12.4	Excellent	Perfect	Best

Table 6.2.2.2 Inpainting Models Evaluation Result

Compared to alternative methods such as LaMa, which is faster at 0.8 seconds but produces blurrier results, or Stable Diffusion 1.5, which takes 1.5 seconds with good quality, or Stable Diffusion 3.5, which takes 12.8 seconds with good quality, but the SD 2.0 + ROI approach offers the optimal balance for people inpainting in indoor navigation.

Instead of processing the entire 224×224 camera frame which would typically require 4–6 seconds for Stable Diffusion, the system crops only the minimal region containing detected individuals, often reducing the processing area to just 150–300 pixels. This ROI extraction, combined with adaptive resolution scaling that adjusts parameters based on region size, dramatically reduces computational requirements while maintaining high-quality results. For example, small ROIs (≤256 px) are processed at 256×256 resolution with four inference steps and a guidance scale of 2.5, while larger regions receive appropriately scaled parameters.

Performance is further optimized through simplified prompts such as "empty space, background," with negative prompts excluding "people" and "objects," alongside real-time mode settings that prioritize speed over absolute quality. This approach achieves an average processing time of 1.8 seconds for Stable Diffusion 2.0, producing high-quality inpainting with clean, realistic backgrounds that integrate seamlessly with the original scene through advanced compositing techniques.

6.2.3 Latency Measurement of Navigation

Process	**Average Time (ms)**
Frame Capture → Embedding	120 ms
Embedding → Supabase Query	80 ms
Supabase Result → App Response	50 ms
App Response → TTS Output	100 ms
Total End-to-End Delay	**350 ms**

Figure 6.2.3.1 Latency Measurement

The latency of the indoor navigation system was measured throughout each stage of the real-time pipeline to gain an indication of responsiveness to guidance for visually impaired users. Extracting a camera frame and creating its embedding with the DINOv2-Base model takes approximately 120 ms, accounting for both acquisition overheads and computation for a visual transformer, even though it is heavier than a CNN, to achieve high accuracy with robust localization. Querying the Supabase vector database with an embedding adds a further 80 ms to account for network comms, results retrieval and similarity calculations, but again optimised to be near-realtime. Analysing database results into actionable app responses takes some 50 ms to execute, but translating these instructions into audible guidance with Text-to-Speech (TTS) adds a further 100 ms. As a total, end-to-end delay from frame acquisition to TTS output is some 350 ms, comfortably low where users perceive lag to occur. To ensure smooth and timely feedback, computational intensiveness is balanced against responsiveness to allow the system to offer accurate and real-time navigation guidance even within dynamic indoor environments.

6.3 Project Challenges

Indoor localization accuracy is one of the most critical and challenging aspects of the navigation app. Unlike outdoor environments, indoor spaces often lack distinctive and consistent visual cues, making it difficult for the system to reliably identify and differentiate between locations. Variations in lighting conditions, such as bright sunlight through windows or dimly lit corridors can significantly affect the quality of the camera feed and the feature embeddings extracted by the model. Additionally, indoor environments frequently contain repetitive patterns such as similar-looking doors, hallways, or staircases which can confuse the recognition model. Changes in the environment over time, like furniture rearrangement or temporary obstacles, further complicate localization. These factors can lead to misrecognition or false positives, which is particularly critical for visually impaired users who rely solely on the app's guidance. To maintain high localisation accuracy, the system must not only utilize robust deep learning models capable of extracting discriminative embeddings but also implement strategies such as multi-frame verification, embedding averaging, or leveraging auxiliary data like compass orientation to reduce errors. Despite these measures, achieving consistently high accuracy in diverse indoor settings remains a significant technical challenge.

Furthermore, the complexity of system integration poses a significant challenge for the indoor navigation app, as it requires coordinating multiple components that communicate in real-time. In this setup, the DINOv2 model is hosted behind a FastAPI server, which generates embeddings from the camera frames. At the same time, the Vision-Language Model (VLM) is accessed via an external API for additional processing or verification. The Flutter app must capture live camera frames, send them to the FastAPI server for embedding generation, query Supabase for nearest-neighbor searches, process compass data, and provide timely audio guidance. Each of these components uses different data formats, operates on separate networks, and introduces potential latency. Ensuring smooth integration involves managing asynchronous communication, handling network delays or API failures, and maintaining proper synchronization between embedding generation, database queries, and user feedback. Any misalignment such as slow API responses, incorrect embedding dimensions, or sensor inaccuracies can propagate through the system, causing recognition errors or delayed guidance. This complex coordination highlights the challenge of integrating machine learning models, cloud services, and mobile app functionality into a seamless real-time navigation experience.

6.4 Objectives Evaluation

This section evaluates the project's success by assessing the extent to which each of its stated objectives was achieved. The evaluation is based on the system's implemented features, design architecture, and performance results as detailed in the preceding chapters.

- **To design a visual localization system that accurately determines the**

user's position using real-time camera input and stored image embeddings.

This objective was successfully achieved. The system features a sophisticated two-stage localization module designed for high accuracy. It uses real-time frames from the smartphone camera, which are processed by the DINOv2 model to generate robust visual embeddings. These embeddings are then compared against a database of pre-recorded locations stored in Supabase to find the most likely match. To further enhance accuracy in visually ambiguous environments, the system integrates an optional Vision-Language Model (VLM) for final verification. The evaluation in Section 6.1.1 demonstrated the effectiveness of this design, showing that while using embeddings alone yielded 60% accuracy, the addition of VLM verification increased the accuracy to a perfect 100% across all test cases.

- **To build a route-based navigation system that guides users step-by-step using pre-recorded visual routes, compass data, and real-time scene matching.**

  This objective was successfully achieved. The system provides a complete workflow for route-based navigation. Administrators can create navigation paths (edges) by walking a route and recording a sequence of waypoints, where each waypoint captures both a visual frame and compass data. For the end-user, the navigation module provides step-by-step guidance by continuously matching the live camera feed against the pre-recorded waypoints in the selected route. The system also implements a crucial initial orientation phase, using the smartphone's compass to ensure the user is facing the correct direction before beginning navigation. Testing results in Section 6.1.2 confirmed that the system delivered accurate and correct turn-by-turn instructions in environments without obstructions.

- **To implement a user-friendly mobile interface that delivers clear and accessible navigation feedback through voice prompts, compass orientation, and vibration cues.**

  This objective was successfully achieved. The mobile interface was designed with accessibility as a primary focus. The system delivers clear, real-time navigation instructions through audio feedback using a text-to-speech (TTS) engine. The effectiveness of this feedback was validated in Section 6.1.3, where all voice outputs were found to be clear and delivered on time. The user experience is further enhanced by a low end-to-end latency of just 350 ms from frame capture to audio output, ensuring guidance feels instantaneous. For hands-free operation, a voice assistant was implemented that can be activated with the custom wake word "Hey Navi," allowing users to initiate localization and navigation with simple voice commands.

- **To ensure the system functions effectively without relying on additional sensors or external infrastructure, making it portable and scalable for real-world indoor environments.**
  This objective was successfully achieved. The core design of the system is fundamentally infrastructure-free, relying only on the user's smartphone camera and its built-in compass sensor. This approach makes the solution highly portable and easy to deploy in any new indoor setting without the cost and maintenance associated with physical hardware like Bluetooth beacons or NFC tags. The system's scalability is addressed through its architecture, which offloads intensive computations to a FastAPI backend and uses a scalable cloud database (Supabase) capable of handling large embedding datasets and concurrent users, as outlined in the non-functional requirements.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

This project successfully developed and evaluated a comprehensive, smartphone-based indoor navigation system designed to enhance the independence and safety of visually impaired individuals. Moving beyond traditional systems that depend on costly physical infrastructure like beacons or markers, this solution presents a cost-effective and highly portable alternative. The system's core innovation lies in its advanced visual place recognition engine, which leverages a **DINOv2 vision transformer** to generate robust semantic embeddings from real-time camera input. These embeddings are managed and queried against a **Supabase vector database**, enabling rapid and accurate localization. To achieve state-of-the-art accuracy, the system implements a two-stage verification process where initial embedding-based matches are confirmed by a **Vision-Language Model (VLM)**, a technique that proved highly effective in testing, achieving 100% localization accuracy. The system also addresses the challenge of dynamic environments by integrating a **YOLOv8-seg model** and **Stable Diffusion 2.0** to detect and inpaint people from scenes, thereby improving navigation reliability in moderately busy areas. Key implemented features include a full route management module for administrators and an intuitive navigation interface for users, providing step-by-step guidance through clear, responsive audio prompts. Accessibility is further enhanced by a hands-free **voice assistant**, activated by the wake word "Hey Navi," allowing users to control the app's core functions with natural speech. Ultimately, this project delivers a complete and functional proof-of-concept that is accurate, scalable, and user-centric, directly addressing the critical need for better assistive navigation technology.

7.2 Future Consideration and Recommendation

Based on the project's implementation and evaluation findings, several key recommendations are proposed to transition the system from a development prototype to a robust and production-ready application. Firstly, the backend services, including the DINOv2 embedding model currently hosted on a local laptop should be migrated to a dedicated cloud platform with GPU support, such as AWS or Google Cloud. This would ensure 24/7 availability, reliability, and scalability for concurrent users without dependency on a local machine. Secondly, to enhance navigation reliability, the system should move towards leveraging a Vision-Language Model (VLM) for step-by-step guidance. However, a significant challenge is that the current cloud-based VLM processes discrete images and does not support real-time video streaming, with a latency of 1.5-2 seconds per frame that is unsuitable for continuous navigation. Therefore, a critical future step is to research and integrate a next-generation, on-device multi-modal model. Such models are designed to run efficiently on mobile hardware, which would drastically reduce latency by eliminating the network round-trip, allow for real-time visual processing, and improve privacy by keeping camera data on the user's device. To further enhance robustness, especially in visually obstructed environments, the system should incorporate Visual-Inertial Odometry (VIO) by fusing camera and IMU sensor data to maintain continuous position tracking. Finally, to improve scalability and functionality, an offline mode and a crowdsourcing feature could be introduced to ensure functionality without internet and to accelerate the mapping of new environments.

REFERENCES

[1] World Health Organization, "Blindness and vision impairment," *World Health Organization*, Aug. 10, 2023. https://www.who.int/news-room/fact-sheets/detail/blindness- and-visual-impairment

[2] Bourne, R.R.A.; Flaxman, S.R.; Braithwaite, T.; Cicinelli, M.V.; Das, A.; Jonas, J.B.; Keeffe, J.; Kempen, J.H.; Leasher, J.; Limburg, H.; et al. Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: A systematic review and meta-analysis. *Lancet Glob. Health* 2017, *5*, e888–e897.

[3] Maghdid HS, Lami IA, Ghafoor KZ, Lloret J. Seamless outdoors-indoors localization solutions on smartphones: implementation and challenges. ACM Comput Surv (CSUR). 2016;48(4):53.

[4] J. Andrew, César Vargas Rosales, David Muñoz Rodríguez, and F. Brena, "Evolution of Indoor Positioning Technologies: A Survey," *Repositorio.tec.mx*, 2017, doi: https://doi.org/1687725X.

[5] Rabbi, I. and Ullah, S., 2013. A survey on augmented reality challenges and tracking. *Acta graphica: znanstveni časopis za tiskarstvo i grafičke komunikacije*, *24*(1-2), pp.29-46.

[6] Fiala, M., 2005, June. ARTag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 2, pp. 590-596). IEEE.

[7] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J. and Medina-Carnicer, R., 2016. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern recognition*, *51*, pp.481-491.

[8]"Infrared technology and it's usability for IPS.," *Mapsted Technology*, 2014. https://mapsted.com/indoor-location-technologies/infrared-positioning (accessed Sep. 03, 2024).

[9] P. Tan, Tinaye Hamufari Tsinakwadi, Z. Xu, and H. Xu, "Sing-Ant: RFID Indoor Positioning System Using Single Antenna with Multiple Beams Based on LANDMARC Algorithm," *Applied Sciences*, vol. 12, no. 13, pp. 6751–6751, Jul.

2022, doi: https://doi.org/10.3390/app12136751.

[10] T. Ji, W. Li X. Zhu and M. Liu, "Survey on indoor fingerprint localization for BLE," 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2022, pp. 129-134, doi: 10.1109/ITOEC53115.2022.9734528.

[11] Blattner, A., Vasilev, Y. and Harriehausen-Mühlbauer, B., 2015. Mobile indoor navigation assistance for mobility impaired people. *Procedia Manufacturing*, *3*, pp.51-58.

[12] Kasprzak, S.; Komninos, A.; Barrie, P. Feature-Based Indoor Navigation Using Augmented Reality. In Proceedings of the 2013 9th International Conference on Intelligent Environments, Athens, Greece, 18–19 July 2013; pp. 100–107.

[13] Xie, T.; Jiang, H.; Zhao, X.; Zhang, C. A Wi-Fi-Based Wireless Indoor Position Sensing System with Multipath Interference Mitigation. Sensors 2019, 19, 3983

[14] Ozdenizci, B.; Ok, K.; Coskun, V.; Aydin, M.N. Development of an Indoor Navigation System Using NFC Technology. In Proceedings of the 2011 Fourth International Conference on Information and Computing, Washington, DC, USA, 28–29 March 2011; pp. 11–14.

[15] Mehta, P.; Kant, P.; Shah, P.; Roy, A.K. VI-Navi: A Novel Indoor Navigation System for Visually Impaired People. In Proceedings of the 12th International Conference on Computer Systems and Technologies, CompSysTech '11, Vienna, Austria, 16–17 June 2011; pp. 365–
371.

[16] Kjærgaard, M.B.; Blunck, H.; Godsk, T.; Toftkjær, T.; Christensen, D.L.; Grønbæk, K. Indoor Positioning Using GPS Revisited. In Pervasive Computing; Floréen, P., Krüger, A., Spasojevic, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 38–56.

[17] D. López-de-Ipiña, T. Lorido, and U. López, "Indoor Navigation and Product Recognition for Blind People Assisted Shopping." Ambient Assisted Living, pp. 33-40, 2011, doi: 10.1007/978-3-642-21303-8_5.

[18] Mulloni, A.; Seichter, H.; Schmalstieg, D. Handheld Augmented Reality Indoor Navigation with Activity-based Instructions. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, Stockholm,
Sweden, 30 August–2 September 2011; pp. 211–220

[19] B. Shin *et al.*, "Motion Recognition-Based 3D Pedestrian Navigation System Using Smartphone," in *IEEE Sensors Journal*, vol. 16, no. 18, pp. 6977-6989, Sept.15, 2016, doi: 10.1109/JSEN.2016.2585655.

[20] A. Mulloni, D. Wagner, I. Barakonyi and D. Schmalstieg, "Indoor Positioning and Navigation with Camera Phones," in IEEE Pervasive Computing, vol. 8, no. 2, pp. 22-31, April-June 2009, doi: 10.1109/MPRV.2009.30.

[21] C. La, V. Catania, S. Monteleone, J. F. De, and J. Bajo, "Computer Vision Based Indoor Navigation: A Visual Markers Evaluation," pp. 165–173, Jan. 2015, doi: https://doi.org/10.1007/978-3-319-19695-4_17.

[22] D. Khan, S. Ullah, and S. Nabi, "A Generic Approach toward Indoor Navigation and Pathfinding with Robust Marker Tracking," Remote Sensing, vol. 11, no. 24, p. 3052, Dec. 2019, doi: https://doi.org/10.3390/rs11243052.

[23] L. C. Huey, P. Sebastian and M. Drieberg, "Augmented reality based indoor positioning navigation tool," 2011 IEEE Conference on Open Systems, Langkawi, Malaysia, 2011, pp. 256-260, doi: 10.1109/ICOS.2011.6079276.

[24] J. Kim and H. Jun, "Vision-based location positioning using augmented reality for indoor navigation," in *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 954-962, August 2008, doi: 10.1109/TCE.2008.4637573.

[25] M. Kalkusch, T. Lidy, N. Knapp, G. Reitmayr, H. Kaufmann and D. Schmalstieg, "Structured visual markers for indoor pathfinding," The First IEEE International Workshop Agumented Reality Toolkit,, Darmstadt, Germany, 2002, pp. 8 pp.-, doi: 10.1109/ART.2002.1107018.

[26] Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real- time 6-dof camera relocalization. In Proceedings of the IEEE international conference on computer vision (pp. 2938-2946).

[27] Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., & Cremers, D. (2017). Image-based localization using LSTMs for structured feature correlation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 627-637).

[28] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations.

[29] Chen, J., Qian, S., Gao, Y., & Chen, C. (2021). INDOOR: Towards Learning-based Indoor Scene Understanding using Visual Transformers. arXiv preprint arXiv:2110.05722.

[30] "ViNT: A Foundation Model for Visual Navigation," *Github.io*, 2023. https://general- navigation-models.github.io/vint/index.html [Accessed: Sep. 03, 2024].

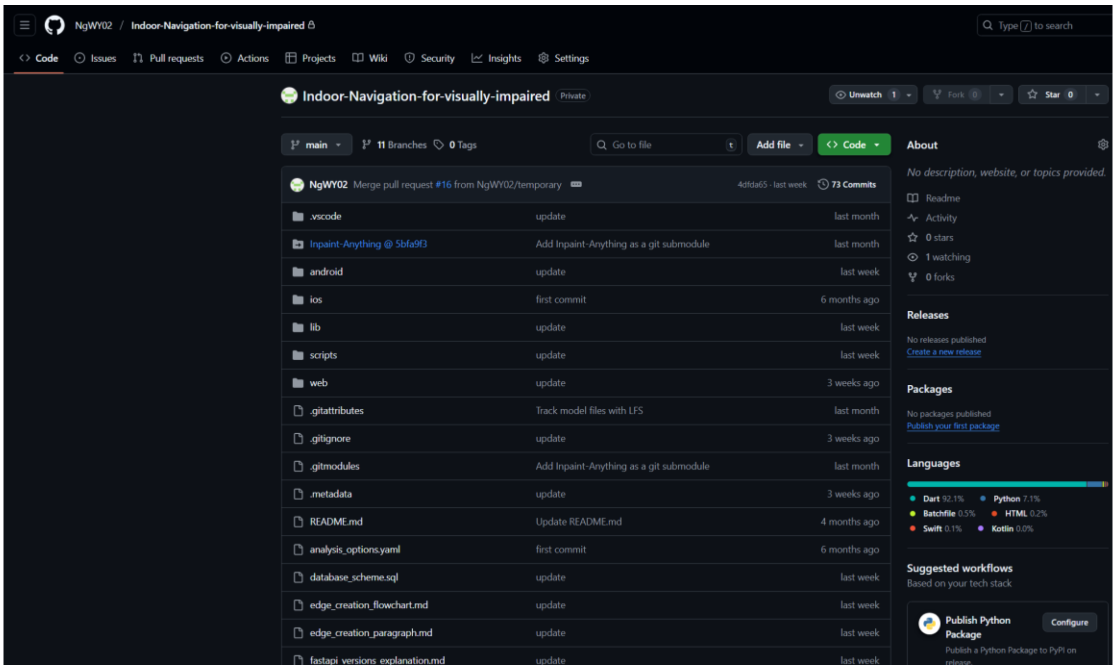
[31] B. Ozdenizci, V. Coskun, and K. Ok, "NFC Internal: An Indoor Navigation System," *Sensors*, vol. 15, no. 4, pp. 7571–7595, Mar. 2015, doi: https://doi.org/10.3390/s150407571.

[32] nisargnp, "GitHub - nisargnp/DeadReckoning: Real-time localization on Android phones using inertial sensors (accelerometer, compass, gyro)," GitHub, 2014. https://github.com/nisargnp/DeadReckoning?tab=readme-ov-file [Accessed: Aug. 31, 2024].

[33] T. Z. Quan, "A Smart Marker-based Indoor Navigation System," B.Sc. thesis, Dept. of Computer Science, Universiti Tunku Abdul Rahman, Kampar, Malaysia, 2024.

APPENDIX

GitHub Repository



Indoor-Navigation-for-visually-impaired

NgWY02 Merge pull request #16 from NgWY02/temporary — 4dfda65 · last week — 73 Commits

.vscode	update	last month
Inpaint-Anything @ 5bfa9f3	Add Inpaint-Anything as a git submodule	last month
android	update	last week
ios	first commit	6 months ago
lib	update	last week
scripts	update	last week
web	update	3 weeks ago
.gitattributes	Track model files with LFS	last month
.gitignore	update	3 weeks ago
.gitmodules	Add Inpaint-Anything as a git submodule	last month
.metadata	update	3 weeks ago
README.md	Update README.md	4 months ago
analysis_options.yaml	first commit	6 months ago
database_scheme.sql	update	last week
edge_creation_flowchart.md	update	last week
edge_creation_paragraph.md	update	last week
fastapi_versions_explanation.md	update	last week

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

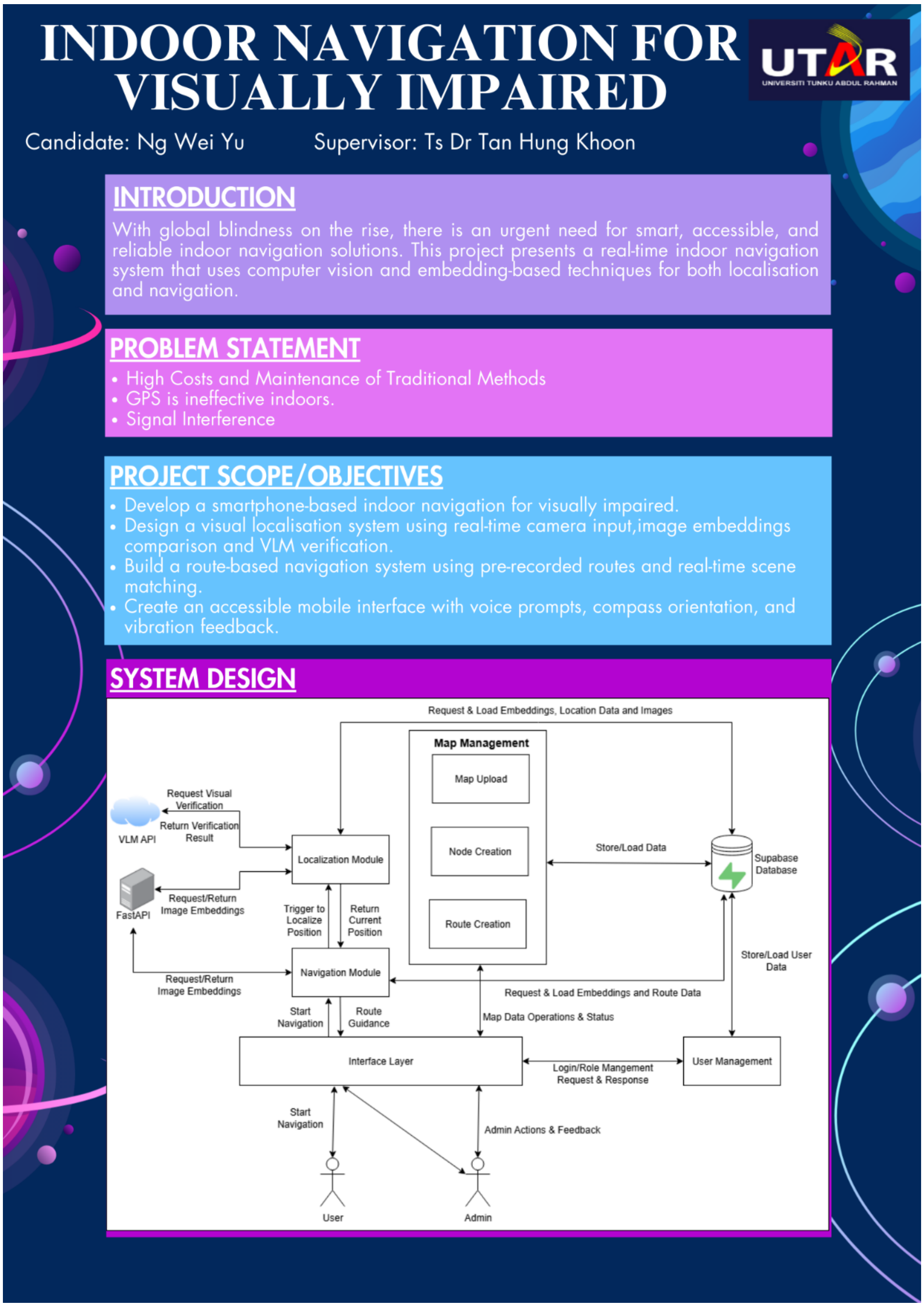
● Dart 92.1% ● Python 7.1%
● Batchfile 0.5% ● HTML 0.2%
● Swift 0.1% ● Kotlin 0.0%

Suggested workflows
Based on your tech stack

Publish Python Package — Configure
Publish a Python Package to PyPI on release.

https://github.com/NgWY02/Indoor-Navigation-for-visually-impaired

Poster



INDOOR NAVIGATION FOR VISUALLY IMPAIRED

Candidate: Ng Wei Yu Supervisor: Ts Dr Tan Hung Khoon

INTRODUCTION

With global blindness on the rise, there is an urgent need for smart, accessible, and reliable indoor navigation solutions. This project presents a real-time indoor navigation system that uses computer vision and embedding-based techniques for both localisation and navigation.

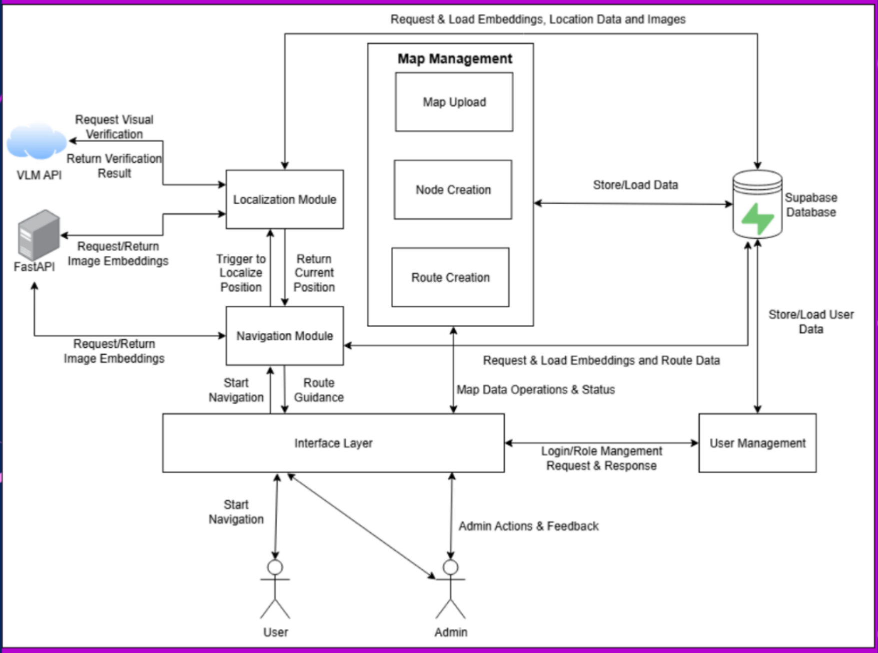
PROBLEM STATEMENT

- High Costs and Maintenance of Traditional Methods
- GPS is ineffective indoors.
- Signal Interference

PROJECT SCOPE/OBJECTIVES

- Develop a smartphone-based indoor navigation for visually impaired.
- Design a visual localisation system using real-time camera input,image embeddings comparison and VLM verification.
- Build a route-based navigation system using pre-recorded routes and real-time scene matching.
- Create an accessible mobile interface with voice prompts, compass orientation, and vibration feedback.

SYSTEM DESIGN


Request & Load Embeddings, Location Data and Images
Map Management
Map Upload
Node Creation
Route Creation
Request Visual Verification
Return Verification Result
VLM API
Localization Module
Request/Return Image Embeddings
FastAPI
Trigger to Localize Position
Return Current Position
Store/Load Data
Supabase Database
Navigation Module
Request/Return Image Embeddings
Store/Load User Data
Request & Load Embeddings and Route Data
Map Data Operations & Status
Start Navigation
Route Guidance
Interface Layer
Login/Role Mangement Request & Response
User Management
Start Navigation
Admin Actions & Feedback
User
Admin