HOME ENERGY MONITORING SOFTWARE SYSTEM (HEMS) - MOBILE APPLICATION AND WEB-BASED DASHBOARD

WONG KE XIN

UNIVERSITI TUNKU ABDUL RAHMAN

HOME ENERGY MONITORING SOFTWARE SYSTEM (HEMS)- MOBILE APPLICATION AND WEB-BASED DASHBOARD

WONG KE XIN

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

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

May 2024

DECLARATION

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

Signature	:	[illegible]
Name	:	Wong Ke Xin
ID No.	:	2001293
Date	:	16/05/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"HOME ENERGY MONITORING SOFTWARE SYSTEM (HEMS) - MOBILE APPLICATION AND WEB-BASED DASHBOARD"** was prepared by **WONG KE XIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ms Beh Hooi Ching, Michelle
Date	:	16th May 2024

Signature	:	
Co-Supervisor	:	
Date	:	

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Ms Beh Hooi Ching for her invaluable advice, guidance and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement.

Lastly, I would like to thank all MIMOS staff who helped me with this project during my internship period for guidance and support.

ABSTRACT

The increasing energy demand driven by economic expansion, population growth, and urbanization in Malaysia poses environmental challenges due to the heavy reliance on non-renewable fossil fuels to generate electricity, which contributes to greenhouse gas emissions and global warming. The proposed Home Energy Monitoring System (HEMS) leveraging IoT technology has been developed to address these challenges by empowering residents to actively monitor, analyze, and optimize their electricity consumption through mobile applications to foster sustainable energy practices and mitigate carbon emissions in Malaysia. The project aims to determine project requirements, develop a mobile application for monitoring residential energy usage, create a web-based dashboard for visualizing energy consumption data, and evaluate functionalities through various tests. The target user for the mobile application is the residents in Malaysia who have home energy monitoring devices installed in their residences, and the target user for the web-based dashboard is an administrator who manages the accounts for residents. The functionalities of both mobile applications and web-based dashboards, such as monitoring real-time energy consumption, viewing historical energy consumption, setting thresholds to manage energy consumption, managing user profiles and viewing resident dashboards, are successfully developed using iterative and incremental development methodologies to develop different functionalities in phases. Testing such as unit testing, integration testing, system testing, code quality analysis and system usability scale evaluation were conducted throughout development to ensure a smooth user experience. The system has achieved As in maintainability, reliability, security and usability indicating its suitability for routine operations. Through collaboration with MIMOS Berhad, the goal and objectives of the project have been achieved successfully within the timeline, with the mobile application and web-based dashboard prototype as the final deliverables.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
API Gateway	Amazon API Gateway
Amplify	AWS Amplify
AppSync	AWS AppSync
AWS	Amazon Web Services
CloudFront	Amazon CloudFront
CloudWatch	AWS CloudWatch
Cognito	AWS Cognito
DynamoDB	Amazon DynamoDB
HEMS	Home Energy Monitoring System
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
Lambda	AWS Lambda
MQTT	Message Queuing Telemetry Transport
NIALM	Non-intrusive Appliance Load Monitoring
S3	Simple Storage Service
SES	Simple Email Service
SNS	Simple Notification Service
SUS	System Usability Scale
TNB	Tenaga Nasional Berhad

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

The expanding economy, population growth, and rapid urbanization are expected to drive a rise in energy demand in Malaysia by increasing by 2% annually until 2050 (Ministry of Economy, 2023). This increasing energy demand has led to a heavy reliance on non-renewable fossil fuel-based resources, causing greenhouse gas emissions and contributing to global warming (Amin, et al., 2022). To address these environmental issues by lowering the energy consumption of residents in Malaysia, the implementation of a Home Energy Monitoring System (HEMS) is proposed to encourage residents to monitor their electricity consumption via mobile or web applications. With the goal of creating a fully functional prototype for a cutting-edge Home Energy Monitoring System (HEMS) utilising IoT technology, this project represents a key industry link collaboration with MIMOS Berhad. By enabling residents to actively monitor, analyse, and optimise their electricity use, the HEMS system will promote sustainable energy practises and lessen dependency on resources derived from fossil fuels to reduce reliance on fossil fuels and lower carbon emissions.

The scope of this study covers modules including real-time energy monitoring, historical data analysis, notification alerts, and user management to encourage residents to utilise sustainable energy sources. The targeted respondents are residents in Malaysia who own smart meters in their households for energy monitoring.

Moreover, the research approach for this project is quantitative, using a questionnaire to understand residents' electricity consumption patterns, energy-saving behaviours and preferences towards HEMS features. The methodology used in the development process is incremental and iterative, facilitating the prioritisation of important functionality and accommodating requirement modifications in response to user feedback.

Next, the proposed solution for this project is an IoT-based HEMS that includes both a mobile application and a web-based dashboard. Electricity consumption data will be collected by energy monitoring devices and stored in the cloud database so that the mobile application and web application are able to retrieve it in real time. Thanks to the mobile app's on-the-go energy consumption tracking and real-time alerts, users may view their energy consumption statistics whenever and wherever they want.. Malaysian citizens can learn more about their electricity usage, spot inefficiencies, and adopt energy-saving habits to lessen their environmental impact by integrating smart home appliances and energy monitoring systems.

This project is expected to be done by April 2024, with the Home Energy Monitoring Mobile and Web application as the final deliverable.

1.2 Problem Background

Fossil fuel-based energy resources, such as natural gas, crude oil, and coal, dominate Malaysia's energy landscape with proportions of 42.0%, 33.3%, and 21.3% respectively, while renewable energy sources like hydropower and solar make up a significantly smaller portion (SURUHANJAYA TENAGA (ENERGY COMMISSION), 2021). The rapid growth of the population, coupled with an expanding economy and improving standards of living, is driving up a constant increase in the demand for electricity for residences. In 2017, Malaysia's renewable energy capacity under the Feed-in Tariff program reached 528.06 MW, significantly lagging behind Vietnam, Indonesia, and Thailand, which had installed capacities of 2569 MW, 3833 MW, and 6766 MW respectively (Sustainable Energy Development Authority Malaysia [SEDA], 2019). The continuously expanding renewable electricity capacity is unable to meet the high electricity demand driven by a densely populated population and extensive industrial activities. Energy generation using coal stands as the largest contributor to CO2 and N2O emissions, while the natural gas transport and the process of coal mining for energy generation cause leakage of most CH4 emissions (Abdul Latif, et al., 2021). The emissions of greenhouse gases from energy generation contribute to global warming and climate change, causing the greenhouse effect and resulting in rising sea levels due to the rapid melting of ice caps. This poses a significant risk of floods and displacement of low-lying areas. Malaysia, being an island nation facing an immediate threat from rising water levels, is particularly vulnerable to the impacts of increased global warming. Given Malaysia's challenges to quickly satisfy its rising energy needs using only alternative energy sources, it is critical to solve the greenhouse effect's environmental issue. Optimising household energy use that allows user to monitor their own energy consumption can help mitigate greenhouse gas emissions by lowering the demand for energy derived from fossil fuels.

1.3 Problem Statement

1.3.1 Increasing energy demand

The production, use, and emissions of electricity are strongly influenced by several variables, including economic growth, population changes, energy prices, technological innovation, exports, foreign direct investment, and human-related variables like socioeconomic status, psychology, and housing characteristics (Ali, Razman and Awang, 2019). The rapid growth of the population, coupled with an expanding economy and improving standards of living, is driving up a constant increase in residence electricity demand.

Therefore, to meet the energy demand, more expensive fossil fuels, including natural gas and coal, are used due to the restricted capacity of renewable energy sources. Malaysia's electricity demand projected to increase at an annual rate of 3% from 2010 to 2030, the reliance on expensive coal and natural gas, coupled with supply constraints, has emerged as a pressing concern (Tenaga Nasional Berhad, 2011). The price of these fossil fuels is frequently erratic, which raises the price of electricity production and finally causes high cost of electricity. By maximising electricity use, incorporating renewable energy sources, and encouraging energy-efficient practises, the implementation of the HEMS (Home Energy Management System) can assist in addressing the rising demand for energy by monitoring residents' energy consumption. This will reduce reliance on costly fossil fuels and will lessen the impact of price fluctuations on electricity costs.

1.3.2 Inefficient Use of Electronic Gadgets and Limited Information on Energy Consumption

Inefficient use of electronic gadgets by Malaysian residents in their household will cause high energy consumption resulting in high electricity bills. Growing urban electricity consumption creates challenges in performing energy-saving strategies due to limited information on residential energy use (Ali, et al., 2021). However, Nasir, et al. (2020) highlighted Malaysians' complaints about energy services issues such as lacking information about monthly electricity consumed and difficulties in tracking everyday routines corresponding to long-term energy use to perform energy practices. Normative feedback that enhances communication motivates high consuming households to reduce energy consumption, leading to sustained reductions in energy usage (Dominicis, et al., 2019). Therefore, smart energy monitoring systems can provide customized real time energy consumption feedback for more efficient energy use, reducing financial stress and fostering sustainability.

1.4 Aim and Objectives

1.4.1 Aim

The aim of the project is to develop a working prototype which reflects the overall application of Home Energy Monitoring System (HEMS).

1.4.2 Objectives

1. To determine project requirements and investigate existing home energy monitoring systems that align with company needs.
2. To develop a mobile application which monitors the energy usage consumption.
3. To develop a web-based dashboard that visualizes the energy consumption data for the residents.
4. To evaluate mobile application and web-based dashboard functionalities by using unit testing, integration testing, system testing and system usability scale evaluation with less than 10% error result.

1.5 Proposed Solution

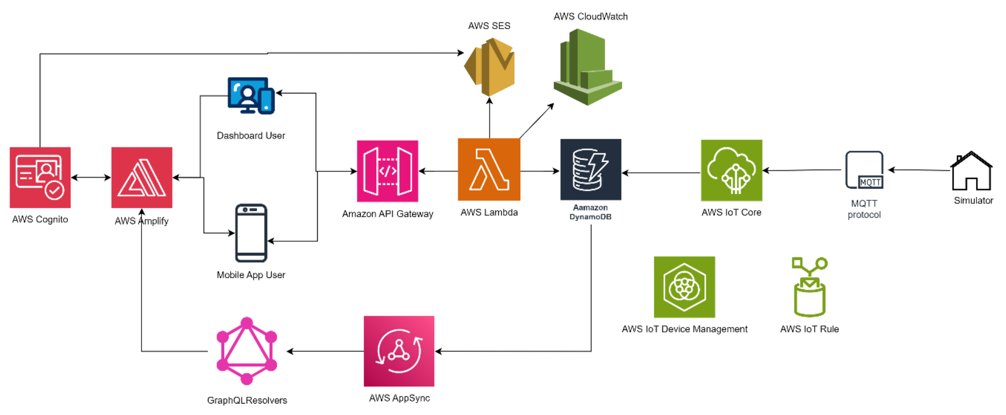


Figure 1.5.1: Overview of the Proposed Solution

The proposed solution is to develop an IoT energy consumption mobile application and web application.

The Home Energy Monitoring System (HEMS) should use both mobile and web applications since this will appeal to a wider user base with a range of device preferences and usage scenarios. Mobile applications offer the convenience of on-the-go monitoring and real-time alerts, while web applications provide flexibility for administrators to monitor different residents' energy consumption by allowing access from any device with a web browser. This strategy improves user happiness with the energy monitoring system in terms of engagement, accessibility, and all-around user pleasure.

React Native and Expo are ultilized to develop the user interface of mobile applications for both Android and iOS. Besides, HTML5, CSS, Laravel and JavaScript frameworks will be selected for developing the front-end view of web-based applications. Next, Amazon DynamoDB is selected to handle real-time data collection retrieved via mobile application and web application from households as its high performance and low latency characteristics are well suited to store real-time energy consumption data, handle frequent data updates and retrieve data quickly to user. The company will provide energy monitoring simulator to provide energy consumption data to the database selected.

Firstly, real-time energy consumption data collected by energy monitoring devices are transmitted to the DynamoDB table via MQTT protocol from the AWS IoT core. A WebSocket is established to send the energy consumption data from DynamoDB to mobile applications in real time. Besides, the historical data stored inside DynamoDB will be retrieved using AWS Lambda functions by performing calculations.

Section 3.5 of Chapter 3 and Chapter 6 provide detailed insights into this proposed solution to offer a comprehensive understanding of this solution to develop a Home Energy Monitoring System.

1.6 Project Approach

1.6.1 Research Approach

The chosen research methodology is the qualitative research approach as the qualitative approach focuses on gathering insight and understanding the home energy monitoring system user's experiences, attitudes, and motivations for energy monitoring. Engineers can select the most appropriate qualitative techniques based on their unique requirements because the use of graph theory, expert systems, and qualitative simulation does not necessarily require a mathematical model or considerable measurement data. Large inputs from residents in Malaysia are required to understand their needs for energy consumption (Cheng, 2022). The qualitative research approach enables a contextually rich understanding of the socio-cultural elements that influence the energy consumption behaviours of people in Malaysia in addition to facilitating a comprehensive investigation of users' perceptions of home energy monitoring devices.

Questionnaires is selected to gain insights about the residents’ electricity consumption awareness, energy usage behaviours, knowledge about energy-saving practices, and their familiarity with Home Energy Management Systems (HEMS) and smart meters. Besides, the residents’ barriers and preferences using HEMS are also being collected to understand and develop HEMS with desired features that fulfil the residents’ needs. Target users’ attitude towards energy consumption is expected to support the problem statements clarified to make sure that the development of HEMS can help to solve the target users’ problems. The result and analysis of the collected questionnaires are presented in Chapter 4, Section 4.2.1 for further discussion.

1.6.2 Development Approach

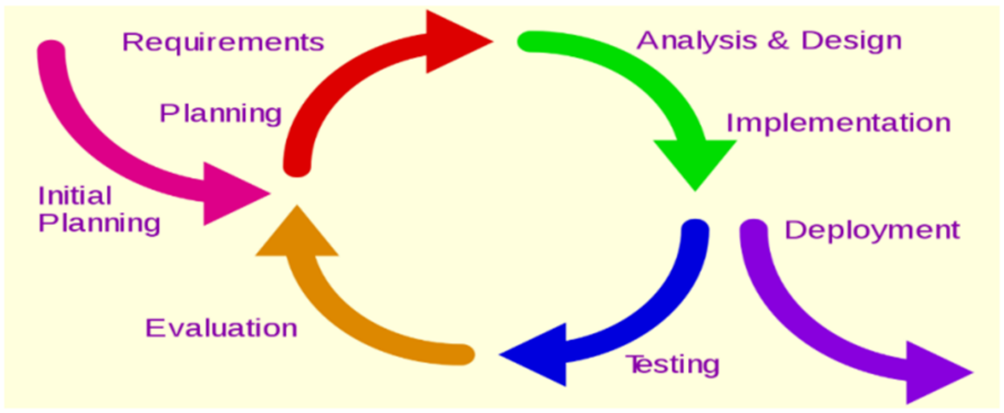


Figure 1.6.1: Iterative and Incremental Development Model (Jevon, 2009)

The proposed methodology for this project is iterative and incremental methodology. The iterative and incremental development (IID) approach entails sequentially identifying requirements, analysing requirement, producing design specification and coding based on the design until final product is delivered (Nonyelum, 2020).

Firstly, the biggest advantage of choosing this methodology is the prioritize development requirements. Since the functionalities of the system had been completely stated by the company, the important requirement can be developed first followed by other minor requirements. The company can get top priority requirement deliverables early that is real-time energy monitoring while the minor functionality such as historical data and analytic and notification and alert will be added in the next iteration.

Besides, this methodology allows easy accommodated requirements changes. Mimos is able to provide feedbacks to each product increment to make sure that the final deliverable fulfils its expectation. The developed system is able to change based on the feedback at the next iteration until end of product development to ensure the system aligns with user needs with continuous improvement.

The in-depth examination of specific tasks performed during each phase are discussed in Chapter 3, Section 3.2 to offer a comprehensive understanding of the development process.

1.7 Scope

This project is aimed to develop IoT energy consumption mobile application and Web-based dashboard called "HEMS” (Home Energy Monitoring System) which integrate with smart household devices. This system is accessible to users through mobile application and internet browser.

1.7.1 Target Users

The target users of this project are the residents in Malaysia who have home energy monitoring devices installed in their residences. They can install the home energy monitoring device and monitor their electricity usage using mobile application and view energy consumption dashboard using a web browser. The target audience is likely made up of eco-aware people who want to keep an eye on and control how much electricity they consume. They might want to cut back on their energy use to save money, lessen their impact on the environment, or both. The intended users are anticipated to be able to access the internet via their cell phones or PCs because the project combines a mobile application and a web dashboard.

1.7.2 Modules Covered

- **Real-time Energy Monitoring Module**
  This module includes visualizing residents’ total energy consumption patterns on a daily, weekly or monthly basis, offering intuitive and informative charts, graphs.
- **Historical Data and Analytics Module**
  This module provides detailed insights and analytics on energy consumption pattern over time based on historical data collected.
- **Notification and Alert Module**
  This module includes real-time notification and alerts to be sent to the residents when the energy consumption exceed predefined level by residents.
- **User Management Module**

This module is for creating resident’s individual accounts, managing profile and performing personalization on energy usage settings including setting preferences, notification, energy consumption level.

1.7.3 Things that not covered in this project

The disaggregation of total energy consumption to specific household energy consumption will not be included in the HEMS's real-time monitoring function.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The global energy landscape stands at a crucial turning point as world energy consumption is unrelenting. The growth in global energy consumption is driven by population growth, urbanization, and industrialization, which brings environmental issues such as fossil fuel depletion and climate change. International agreements such as the Paris Climate Agreement have signified the importance of taking coordinated actions to tackle energy-related challenges in energy management to preserve the environment and climate change (Sachs, 2016). International agreements like the Paris Climate Agreement have recognised the crucial need for international coordination to address energy-related challenges in effective energy management. Residential contributed 26.6 % to electricity consumption in 2019, which is slightly lower than the top electricity contributor in the industry, which was 41.9 % (IEA, 2019). As the global population expands, the demand for electricity in residential areas will also expand.

Energy monitoring systems for residential areas have become the linchpin for worldwide understanding, managing and reducing their energy consumption, providing data for policies and driving conservation efforts in environmental protection. Technological breakthroughs like IoT integration and the adoption of smart grids have transformed energy management, allowing us to collect and analyse data in real-time and distribute energy more efficiently with fewer losses and more reliability. This literature review tends to unravel the relationship between the Home Energy Management System and energy consumption in Malaysia by highlighting the potential of this system to encourage energy efficiency in residential areas. Therefore, the following sections will explore the dimensions of Malaysia's energy consumption, the transformative role of the Internet of Things, the characteristics of real-time systems and the significance of smart meters for energy management.

2.2 Energy Consumption in Malaysia

2.2.1 Total electricity consumption by sector

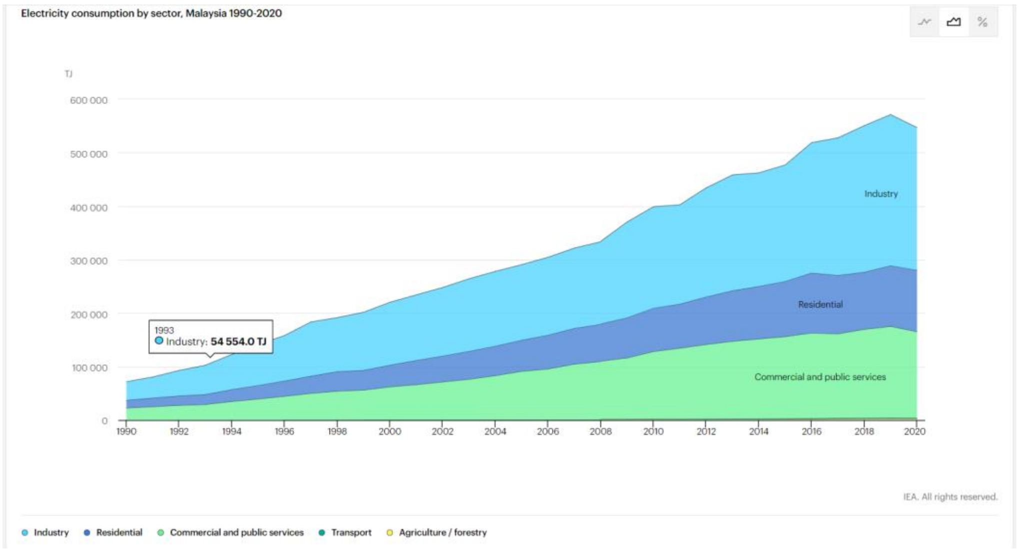


Figure 2.2.1: Total Electricity Consumption by Sector in Malaysia 2023 (International Energy Agency [IEA], 2023)

The industry, residential, and commercial sectors are the top three electricity consumption sectors in Malaysia from 1990 until 2023 (International Energy Agency [IEA], 2023). Since Malaysia is a developing country with industrial sectors such as manufacturing, construction, agriculture, and others that are the largest consumers of electricity, the residential and commercial sectors also play important roles in Malaysia's overall electricity consumption. More than 70% of all households have an air conditioner. Therefore, cooling demand is still rather high even though the contribution of the domestic electricity consumption sector is lower (DOSM, 2020). Energy consumption in residential areas, which is closely related to the usage of household devices, plays a crucial role in controlling electricity demand and provides an opportunity for effective energy management through an energy monitoring system.

In Malaysia, the descending order of electricity-generated resources is coal, oil, natural gas, hydropower, solar energy, biofuels, and waste (IEA, 2022). The consumption of renewable and non-renewable energy sources plays a central role in driving Malaysia's economic growth and supporting a variety of economic sectors, such as services, manufacturing, and agriculture, while also affecting commodity prices and inflation dynamics (Talha, 2021). However, the underutilization of renewable resources is unable to fulfil the growing demand for electricity needs in Malaysia.

Malaysia's government is actively involved in energy monitoring and optimization, especially in sectors such as commercial buildings, utilities, and manufacturing.

2.2.2 The National Energy Efficiency Action Plan

According to the Ministry of Energy, Green Technology and Water (2019), this policy aims to optimise electricity use while understanding its restricted application within the larger energy sector, with the goal of achieving sustainable development, greater welfare, and competitiveness. Malaysia's heightened attention to energy use and consumption is a response to the problems brought on by an increase in energy demand, which is mostly satisfied by non-renewable sources that raise greenhouse gas emissions. This strategy indicates a rising awareness of the significance of sustainable energy practices and the necessity of addressing environmental issues.

2.2.3 Adoption of energy monitoring in Malaysia

In order to track and manage energy use and comply with regulations, Malaysian industries are adopting energy monitoring and optimisation practises in greater numbers through energy management technology and systems. The energy management system is used to track and analyse energy consumption across different production processes in the industry.

2.2.3.1 Manufacturing sector

The integration of smart sensors in Malaysia's IoT ecosystem encourages remote monitoring of factory energy consumption data, makes it easier to deliver transparent supply chain information in real-time and empowers decision-making for energy resource optimisation by IoT-enabled manufacturers, suppliers, and clients (Ling, et al., 2022). These sensors are the foundation of real-time data collecting as they can detect variables including temperature, weight, motion, vibration, acceleration, humidity, and location. As a result, goods are identifiable to track of their movement through the supply chain.

2.2.3.2 Buildings

2.2.3.2.1Educational aspect

The efficient and affordable energy monitoring system developed and deployed across the campus of Universiti Teknikal Malaysia Melaka successfully demonstrates the potential of the Internet of Things (IoT) to provide energy managers with useful insights for efficient energy-saving measures in buildings (Shamshiri, et al., 2019). The installation of an energy monitoring system at UTeM demonstrates that energy monitoring heralding a change towards more proactive and data-driven energy management practices in Malaysia. Besides, this case study demonstrates Malaysia's efforts to adopt cutting-edge technical solutions for resolving energy issues and emphasises the significance of energy monitoring tools in creating a sustainable future.

2.2.3.2.2Energy Monitoring and Management System

According to the Malaysian Green Technology and Climate Chante Centre (2020), a building energy management system (BEMS) is a web-enabled system that integrates the logging of energy data and monitoring of building performance. Next, MyCES EMARS, provided by MyCES SDN BHD, is a sustainable energy monitoring system that enables users to monitor, analyse and report on the building’s energy consumption. As listed on the company’s website, clients from various sectors such as healthcare, manufacturing, administrative, and education, including institutions like University Teknologi Mara, University Malaya Specialist Centre, and Kayaku Safety Systems Malaysia Sdn Bhd has implemented this system to monitor their building's energy consumption (Myces Sdn Bhd, 2023). The effective adoption and deployment of energy monitoring systems in Malaysia demonstrate a dedication to energy conservation and sustainable practices in a variety of industries.

2.2.3.2.3 Centralised Premium AI Smart Home System

With the help of a mobile application, users of the SmartZone Malaysia system may control household appliances, including air conditioners, lights, gates, and other equipment, while keeping an eye on their energy usage. Users can watch energy use, read data on energy generation from their home solar energy system, and view calculations for their electric bills. However, the comparatively expensive implementation costs led to a low adoption rate, which made the system less suitable for Malaysian citizens.

The adoption of advanced energy monitoring systems in industries and commercial buildings gave insight into the ways similar home energy monitoring systems can be adopted in residential areas. It showcased the potential benefits of this system to households.

2.3 Home Energy Monitoring and Management

Monitoring energy simplifies means providing feedback considering the level of feedback and type of feedback provided to the end consumers for visualization purposes only (Zhou, et al., 2014). Therefore, home energy monitoring refers to the practice of tracking and analysing energy consumption within a household to understand energy usage patterns using appropriate charts.

Besides, managing energy consumption can be defined as the involvement of Home Energy Management Systems (HEMS) in the demand response mechanism, which allows end-use customers to modify their electricity usage from their typical consumption behaviour in response to changes in electricity prices that vary over time (Zhou, et al., 2014).

The effectiveness of HEMS is highly dependent on highly motivated consumer's habits around HEMS to encourage rapid and regular use of the HEMS system, so providing timely feedback and ensuring the system's simplicity to motivate engagement of home energy monitoring system should be considered during developments of energy monitoring system (Dam, et al., 2010). Therefore, real-time feedback on energy consumption provided by real-time systems empowers residents to make energy usage decisions and promotes active user engagement to ensure energy consumption practices within residential.

2.4 Real-Time System

A real-time system is a system that must stick to explicit (bounded) response-time requirements and risk negative consequences, including failure to function properly (Laplante, 2004). This means that the system needs to satisfy requirements stated in the system specification by receiving a set of inputs and mapping them into a set of outputs within the required response time. The purpose of the system determines the response time set.

Real-time systems are usually used widely in different sectors such as industries, automotive and transportation, peripheral equipment, medical, customer electronics and the Internet.

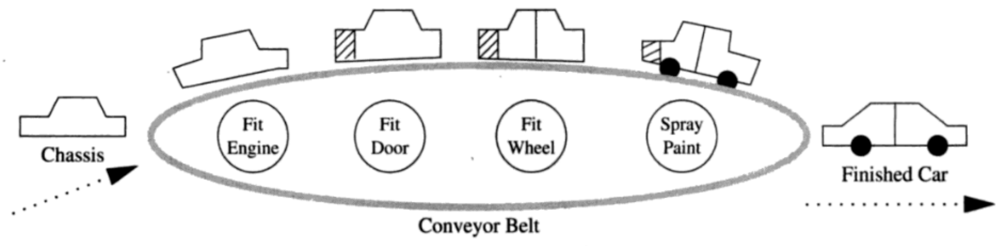


Figure 2.4.1: Schematic Representation of an Automated Car Assembly Plant (Mall, 2009)

Based on Mall (2009), the automated car assembly plant is categorized as real-time system due to its time constraints imposed on each workstation in a few hundred milliseconds.

The figure above shows the production of car assembly; the assembled car will move on a conveyor belt to each workstation for different purposes, such as fitting doors, fitting wheels, and so on. The sensors plugged into each station will sense the arrival of the assembled products, and the workstation will perform its tasks in a strict time-bound. To maintain a smooth and effective production line flow, the car assembly process depends on exact timing. The maintenance of the intended production pace and ensuring that the built automobiles move over the conveyor belt without interruptions depend on the coordination and synchronization of duties at each workstation.

2.4.1 Characteristic of real time system

2.4.1.1 Type of real time system

The real-time concept can be categorized into three parts: soft, firm, and hard real-time systems. A hard real-time system faces catastrophic consequences if a single deadline is missed, a firm real-time system allows for a few missed deadlines without total failure. However, it risks catastrophic consequences if multiple deadlines are not met. A soft real-time system is characterized by performance degradation rather than complete failure when response-time constraints are not met (Laplante, 2004). In this case, a home energy monitoring system is considered a firm real-time system because a few mistakes in real-time energy consumption data will not cause any issues that would affect the resident's ability to visualize the overall energy consumption data.

2.4.1.2 Synchronous and Asynchronous Events

Synchronous events usually happen at predictable times in the flow of control, while asynchronous event occurrences cannot be predicted and are commonly caused by external sources (Laplante, 2004). Home energy monitoring systems (HEMS) tend to be designed to identify synchronous events, which take place at predictable times in the flow of control.

2.5 Internet of Things

Internet of Things (IoT) continues to be eagerly anticipated and actively discussed in the evolving IT world. However, the global user community is currently lacking in agreement on what Internet of Things exactly means due to its consideration of a wide range of interconnected devices and applications. This makes it difficult to establish a universally agreed-upon definition that accommodates the variety of interpretations and applications across different industries and stakeholders. According to Madakam, Ramaswamy and Tripathi (2015), all the proposed definitions of the Internet of Things share the common idea that is the focus of the Internet was on data generated by humans and on data produced by things. In general, the Internet of Things is an open and extensive network of intelligent objects that can auto-organize and share information, resources, and data, responding to and taking action in response to situations and environmental changes. In this research context, IoT can be defined as linking devices such as actuators and sensors, which are embedded in physical home devices linked through a wired and wireless network that can communicate and react to environmental change that monitors energy consumption.

Main Components of IoT

According to Nettikadan and Raj (2018), the main components include sensors that collect data, connectivity for sensors to connect to the Internet for communication, a platform to enable the integration of IoT applications, analytics for data, and a user interface for presenting data to users.

2.5.1 Fundamental Basis for Smart Homes Monitoring

The fundamental basis for smart homes within the IoT framework is the seamless integration of sensors, actuators, and devices through a wireless home automation network, giving consumers the opportunity to remotely monitor and control their networked items and devices (Lee, Teng and Hou, 2016). For example, turning on and off air conditioners, gates, and light switches using voice commands or mobile applications. To relate this concept to home energy monitoring systems, smart irrigation systems with individualized schedules and IoT-enabled thermostats for adaptive temperature control and energy reporting are a few examples of how IoT can improve resource management and energy efficiency in residential settings (Gunge and Yalagi, 2016). This reporting feature concept can be implemented into the Home Energy Monitoring System to monitor and optimize residents' energy consumption while contributing to overall sustainability.

2.5.2 Potential of IoT in Manufacturing for Energy Management and Monitoring

The slow progress of Industry 4.0 adoption in developing nations like Malaysia, as opposed to countries like Germany, is attributed to factors including the predominance of small and medium-sized industries, outdated manual machinery, limited automation, time-consuming data collection, and the economic challenges of upgrading to modern Industry 4.0-compatible equipment (Lee, et al., 2022). However, there are still companies that implement IoT in their production line to manage their working procedures to gain a streamlined-controlled manufacturing process. The integration of cutting-edge technology by Proton in their advanced engine assembly line establishes a precedent for the use of IoT in the automotive industry. It demonstrates their dedication to innovation, quality assurance, and efficiency enhancement in manufacturing (Proton unveils a new hi-tech engine assembly line in Tanjung Malim, 2022). Through this integration, Proton can make use of the real-time data insights, predictive maintenance capability, and quality monitoring offered by IoT. Consequently, operations are streamlined, downtime is decreased, and consistently high-quality products are produced.

2.6 Smart Meters of TNB

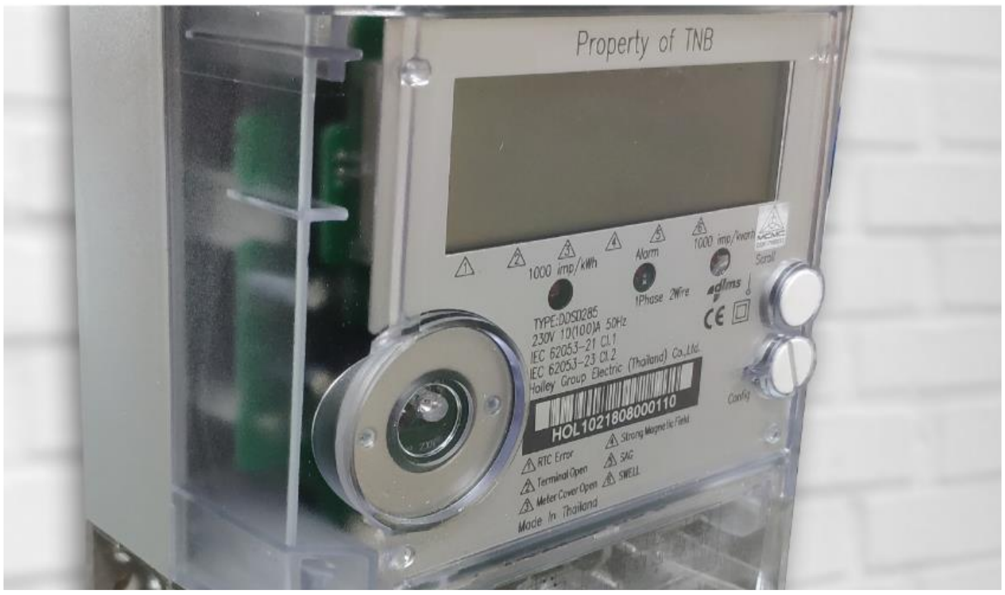


Figure 2.6.1: Smart meters of TNB (Smart meters - Tenaga Nasional Berhad, 2023)

2.6.1 Introduction to Smart Meters

myTNB had implemented a smart meter for Malaysians to get meter readings of their own energy consumption. It is a tool used to measure electrical consumption by enabling two-way communication between the smart meter and TNB via radio-frequency waves. Smart meters will record the reading every 30 minutes and send data automatically to the utility provider, which is TNB daily (Smart meters - Tenaga Nasional Berhad, 2023). Anomalies and disruptions in electrical supply can also be detected by smart meters and reported to TNB. Moreover, these smart meters provide detailed energy consumption data such as electric cost usage in kilowatts per hour. The residents can access this information via myTNB mobile application or myTNB self-service portals. Although these installed smart meters are not capable of analysing the break of the electricity consumption pattern of home appliances, they successfully grabbed Malaysians' attention to energy-saving sectors (Janardhana and Deekshit Shashikala, 2016).

2.6.2 Security and Safety

According to TENAGAofficial (2019), their smart meters have proved to fulfil the standards of the Energy Commission and the Malaysian Communications and Multimedia Commission (MCMC) for wireless communication. In terms of security, TNB protects residents' data and improves network cybersecurity to prevent system compromise by following the Personal Data Protection Act (PDPA). Resident data and account details are synchronized during installation to ensure the accuracy of data sent to specific resident accounts. Furthermore, smart meters are manufactured to meet the standards set by the Energy Commission and are tested by the manufacturer to avoid the risk of fire or explosion.

2.6.3 Benefits of Smart Meters

Smart meters enable automatic billing, which allows residents to monitor their energy consumption data on time using their mobile phones. Residents can plan and control their energy usage by having a direct view of their energy consumption in interactive charts.

Besides, the use of smart meters contributes to environmental benefits by promoting efficient energy consumption, which will indirectly reduce carbon emissions. Smart meters did not cause a decrease in electricity demands, but their real-time feedback display contributed to the behavioural change of residents in performing energy-saving activities (Torriti, 2020). Residents will perform according to the real-time energy feedback to reduce energy consumption and energy costs.

2.6.4 Smart Meter Rollout Phases

TNB planned to replace all existing meters with smart meters in 4 phases starting from Melaka, which had successfully replaced 340,000 residential meters during phase 1 (Subhi, 2020). In phase 2, from the year 2019 to 2021, 1.5 million smart meters were installed in Putrajaya, Melaka, and various locations in Klang Valley. Currently, TNB is conducting phase 3, starting in the year 2022 and beyond, for the deployment of smart meters in Penang, Johor, Langkawi and Ipoh. TNB is planning to install 9.1 million smart meters in

Peninsula Malaysia by the year 2026. The installation of smart meters is free of charge and is performed by certified meter installers who cooperate with TNB. Residential and small businesses will receive smart meters in stage by following replacement standards set by the Energy Commission of Malaysia, which is in line with the Malaysia Electricity Supply Act of 1994.

2.7 AI-Based Residential Energy Monitoring (AIREM)

This project falls under the MIMOS AI-Based Residential Energy Monitoring (AIREM) project, which is part of the Strategic Research Fund under the MOSTI program.

The program's objective is to expedite the advancement of products and technologies in renewable energy and the future electricity grid, with a focus on maximizing local content and contributing to national energy sustainability. The project has a duration of 36 months, spanning from 2022 to 2024.

The primary goal of AIREM is to propose a digitized measurement system powered by artificial intelligence or machine learning. This system aims to capture transient signals from the home distribution board, enabling non-intrusive monitoring (NILM) of residential energy for self-administered power management. The overarching aim is to mitigate energy usage and costs, which directly impact climate change, by empowering residents to calculate their return on investment monthly.

The project's key objectives include developing extraction methodologies for electrical appliances, implementing non-intrusive mechanisms for monitoring home energy usage, and creating a reporting system for assessing the efficiency and health of appliances for maintenance purposes.

The major components of this AI-based residential energy monitoring initiative include:

1. NIALM (Non-intrusive Appliance Load Monitoring) Smart Energy Device:
    - This component serves as the core technology capturing and processing transient signals from the home distribution board.
2. Analytics Operation Center:
    - The analytics operation center is the hub where data collected by the NIALM device is analyzed, interpreted, and transformed into actionable insights.

3. End-User Application:

    - The end-user application is the interface through which residents interact with the monitoring system. It provides information on energy consumption, efficiency reports, and tools for self-administered power management.

Through the integration of these components, the AIREM project aims to revolutionize residential energy monitoring, contributing to both environmental sustainability and cost-effective power management for residents.

2.8 Appliance Load Monitoring

The Appliance Load Monitoring (ALM) system is a smart energy management framework that guarantees the effective stability of energy consumption management (Fagiani, 2019). It consists of two categories: Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring.

Intrusive Load Monitoring records data for each single appliance, while Non-Intrusive Load Monitoring collects total energy consumption and analyzes the energy consumption of each appliance based on the data collected. Although Intrusive Load Monitoring offers higher accuracy in measuring energy consumption of appliances, practical problems such as the high cost of implementation and complex meter configuration caused by Intrusive Load Monitoring make people tend to seek solutions with Non-Intrusive Load Monitoring, which does not require monitoring each appliance's power separately (Ramadan, et al., 2022).

Non-Intrusive Load Monitoring (NILM)

According to Faustine, et al. (2017), non-intrusive load monitoring (NILM), non-intrusive appliance load monitoring (NIALM), or energy disaggregation is a technique that enables the estimation of power consumption attributed to individual appliances from residential aggregate power consumption recorded by smart meters in real-time. It is non-intrusive as the data acquisition is done from the smart meter outside the building without any additional equipment installation inside residential. By providing a breakdown of energy consumption for each household appliance, it helps residents understand their energy consumption patterns. It has been proven that real-time actionable feedback promotes awareness of energy consumption and encourages positive behavioral changes toward more sustainable energy consumption (United Nations Environment Programme, 2009).

Energy Consumption Pattern

Every appliance has its own distinct energy usage pattern that is useful for machine learning algorithms in recognizing the operation of each appliance from the overall energy consumption. According to Hart (1992), residential appliances can be classified according to their operational states in the following manner:

Type-I: The appliances with two states of operation (ON/OFF) such as kettle, lamp and toaster.

Type-II: Multi-state appliances with different operating state (finite State Machines (FSM)) such as washing machine.

Type-III: Continuously Variable Devices (CVD) with no fixed number of states such as power drill. Its power consumption pattern is not repeatable.

Type-IV: Appliances that stay turned on for weeks or days, consistently using energy at a steady rate such as smoke detector (Zeifman and Roth, 2011).

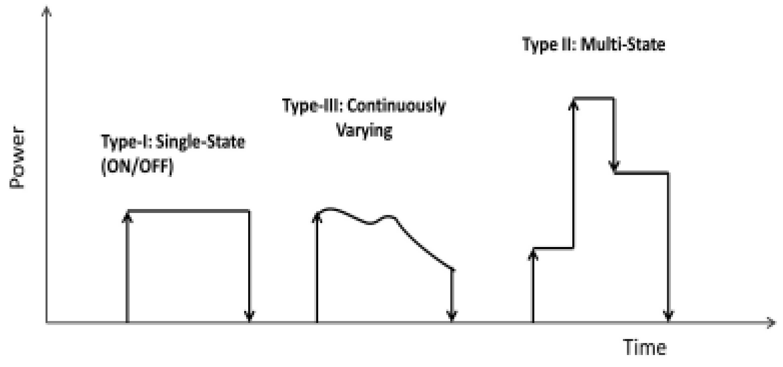


Figure 2.8.1: Different type of energy consumption patterns (Zoha, et al., 2012)

Figure above shows the 3 types of energy consumption patterns listed above that will be transformed to appliance feature for different appliance categories classification.

General Framework of NILM Approach

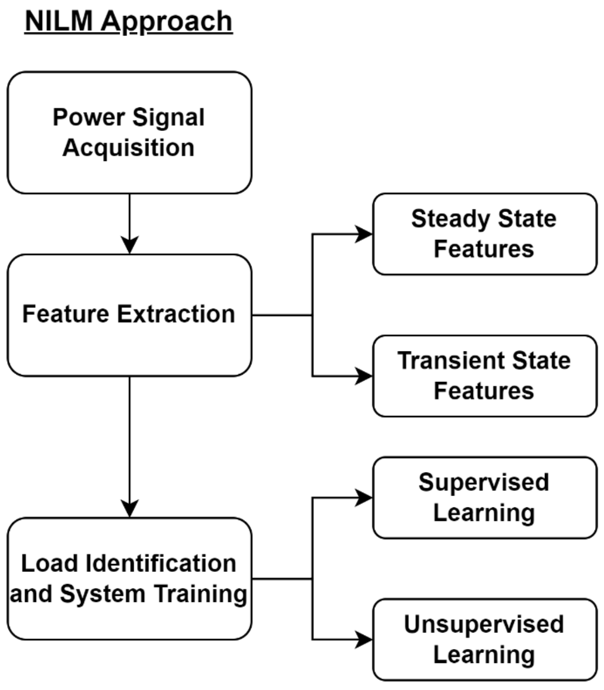


Figure 2.8.1: NILM Approach Architecture

Power Signal Acquisition

Different smart meters are designed to collect aggregated power data at different sampling frequency that is determined by the measurement and electrical characteristics used by NILM algorithm.

Feature Extraction

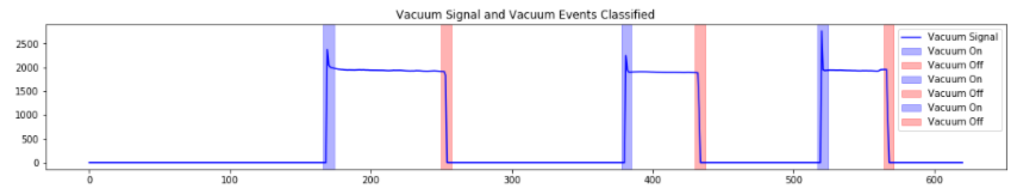


Figure 2.8.3: Result of Steady-State Identification

The raw data is process for power metrics computation such as from power to active and reactive power. Next, event detection is performed to detect appliance state transition (On to Off) by analyzing the changes in power levels using event detection module. Feature extraction methods, including steady-state identification of appliances based on variations in steady-state signatures during on/off transitions, and transient event-based recognition of appliance state transitions using features like duration, size, and shape of transient waveforms (Norford and Leeb, 1996). Figure 2.8.3 shows the event detected using steady-state identification for vacuum by detecting the on and off state of the vacuum.

Load Identification

Load identification algorithms are used to further analyze the extracted appliance features, identifying appliance-specific states through supervised machine learning techniques, which require labeled data to train the model. Pattern recognition approach is a common way for load aggregation by comparing extracted features with load signatures to identify events associated with appliance operations (Liang, et al., 2010).

Simulator for System Training

NILM system requires training or a pre-learning phase before deployment. Supervised disaggregation algorithms depend on labelled data for model training, in context of NILM, obtaining labelled data involves identifying and annotating the ON/OFF states of each appliance within a dataset. However, the manual labelling process for each appliance state can be time-consuming when dealing with a large number of appliances and high-frequency data collection (Hart, 1992). For example, with over 6000 data points collected per second, setting up data collection devices for different appliances will become both costly and time-consuming.

To address this issue, a simulation platform is proposed to simulate the behaviour of various household appliances, allowing the analysis of energy consumption patterns without the costly installation of a smart meter for recording the specific consumption pattern of each appliance (Park, et al., 2010). The simulator provided by MIMOS Berhad tend to mimic the energy consumption data from various household devices and generate synthetic datasets to reduce the manual effort required for training supervised disaggregation algorithms and analyze energy consumption patterns of different home appliances.

2.9 Review of Similar Systems

1) myTNB

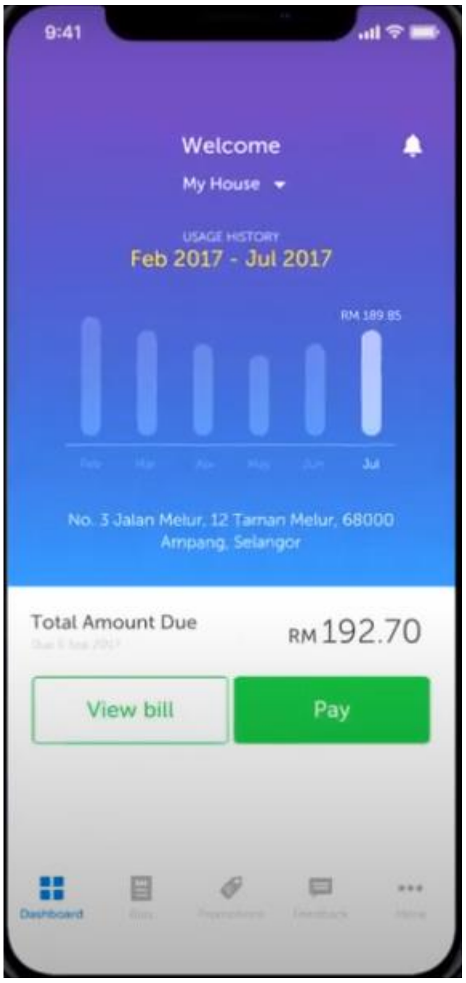


Figure 2.9.1: myTNB Interface for Checking Monthly Electric Bills

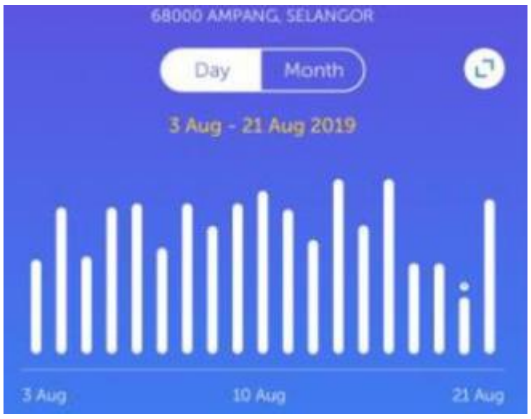
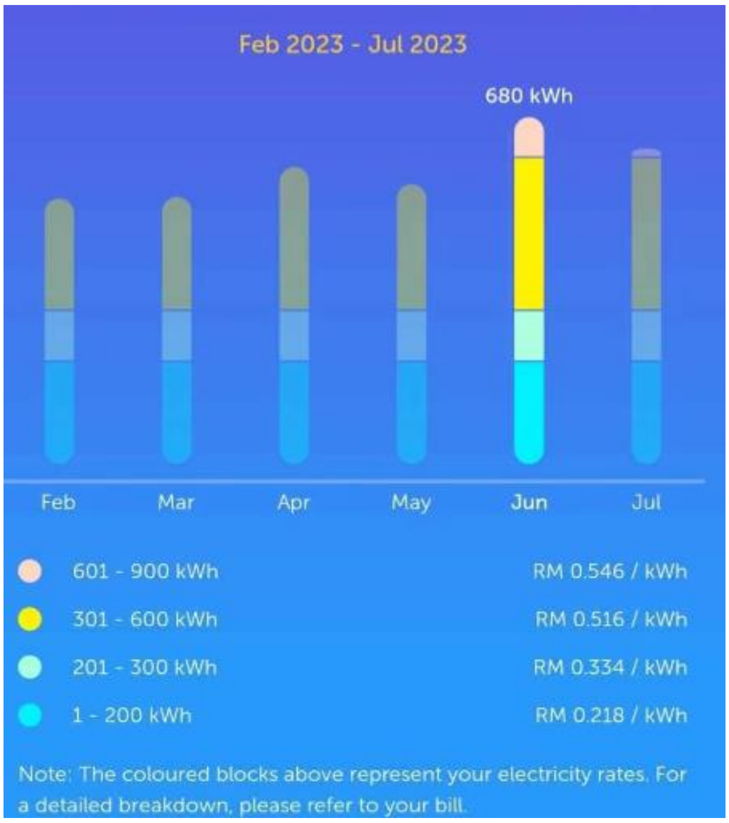


Figure 2.9.2: myTNB Interface for Real-Time Monitoring Feature



●	601 - 900 kWh	RM 0.546 / kWh
●	301 - 600 kWh	RM 0.516 / kWh
●	201 - 300 kWh	RM 0.334 / kWh
●	1 - 200 kWh	RM 0.218 / kWh

Note: The coloured blocks above represent your electricity rates. For a detailed breakdown, please refer to your bill.

Figure 2.9.3: myTNB Interface for Calculating Monthly Electric Bills

Main Features

- **View electricity bills**

  User can view total cost of electricity bills and specific electricity bills in same registered account. View details feature is provided for user by selecting specific bill to view and download specific electricity bills inside their own devices.

- **Monitor electricity consumption**

  Users can monitor electricity consumed and expenditure using interactive usage dashboard which allow users to visualize and track energy consumption or bills in daily or monthly basic.

- **Make payment for electricity bill**

Users can make payments directly through app without getting redirected to another platform. The payment methods provided are online banking payments and credit/debit card.

- **Submit feedback for any bills related matters**

  Users can submit questions via application and get response by Careline agent.

- **Apply Self Meter Reading service**

  Users can make application for self-meter reading service directly and track the application progress via myTNB.

- **Receive personalised notifications**

  Users can receive on time customized updates when their electricity bills are ready, payment reminders and read meter reminders. Besides, users can set their energy budget and receive timely notification once the energy budget is nearly reached.

- **Manage multiple TNB accounts**

  Users can manage and link all TNB accounts on one platform.

Review

myTNB is an energy monitoring application that enables users (residents in Malaysia that own TNB account) to track their energy consumption and conveniently pay their electricity bills. It is available on iOS, Android, and the web. However, the web version primarily serves as an information platform, providing general details about electricity bills and applications for smart meters. The mobile-based application offers additional features and functionality beyond what the web version provides.

Advantages

The "Check Bills" feature enables users to access and view their electricity bills conveniently, anytime and anywhere, without the need for a hardcopy. This feature simplifies the process of checking electricity bills at the telecom company in case the user has misplaced the hardcopy. Moreover, the feature allows the owner to check multiple TNB accounts, ensuring that tenants pay their electricity bills on time.

Electric charges are displayed clearly based on electricity consumption, indicating different grid charging levels. This transparent presentation allows users to understand their billing structure and the corresponding costs associated with various levels of electricity usage.

Drawbacks

Only the account holder, who is the owner, is allowed to view the complete details of the electricity bills, while tenants are unable to directly check their own bills. The electricity bills downloaded by tenants lack personal information such as account number and address, which makes it difficult for companies to record their expenses accurately.

The unstable application also causes delays in receiving electricity bills and notifications of successful payments. Consequently, users are hesitant to pay via the application due to concerns about the payment not being recorded correctly.

2) mySunPower

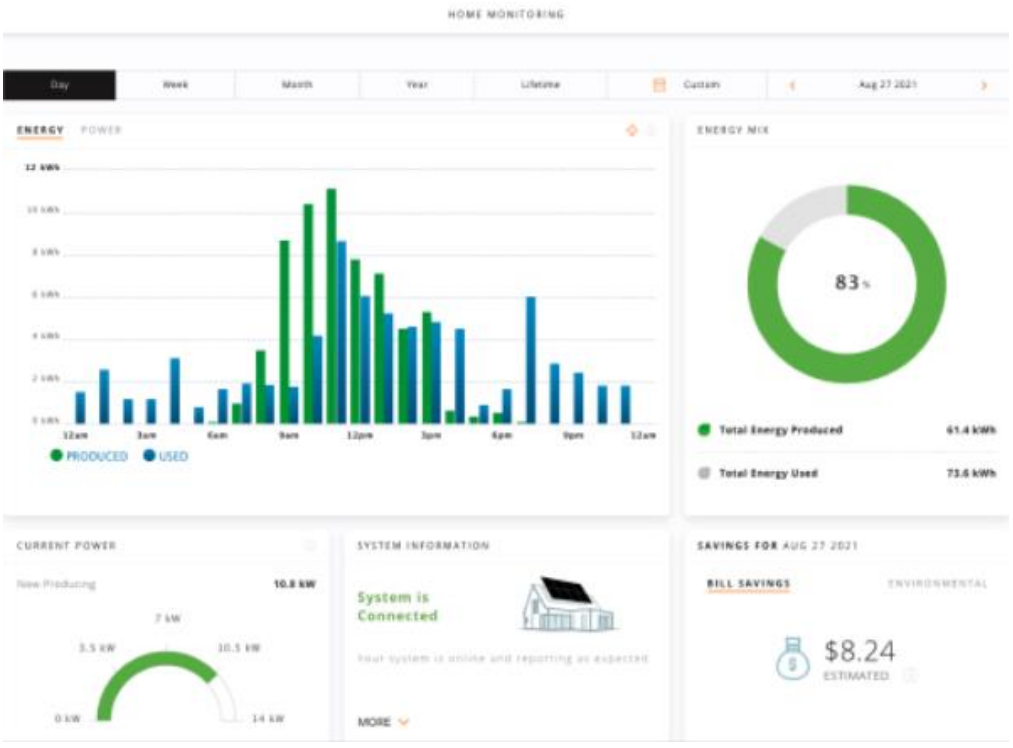
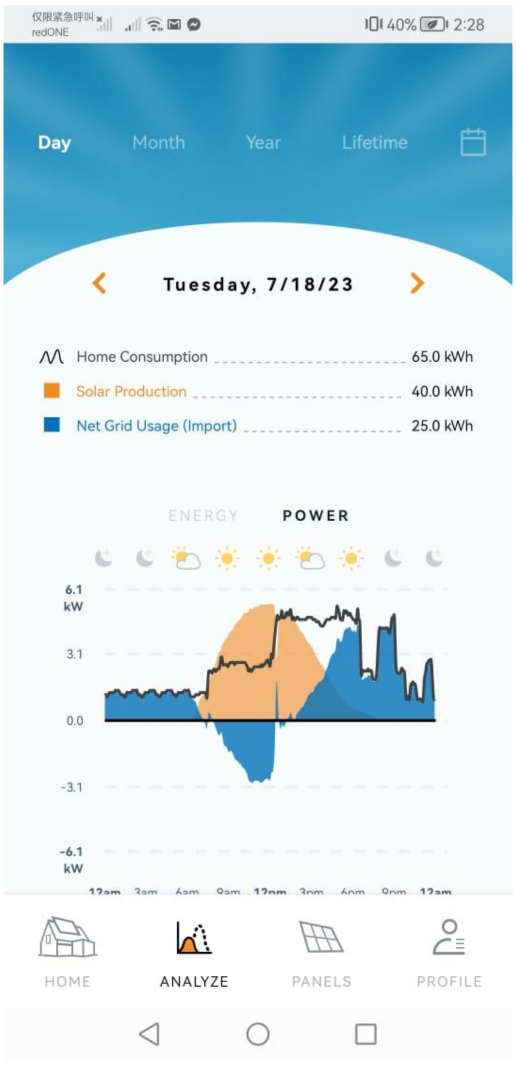
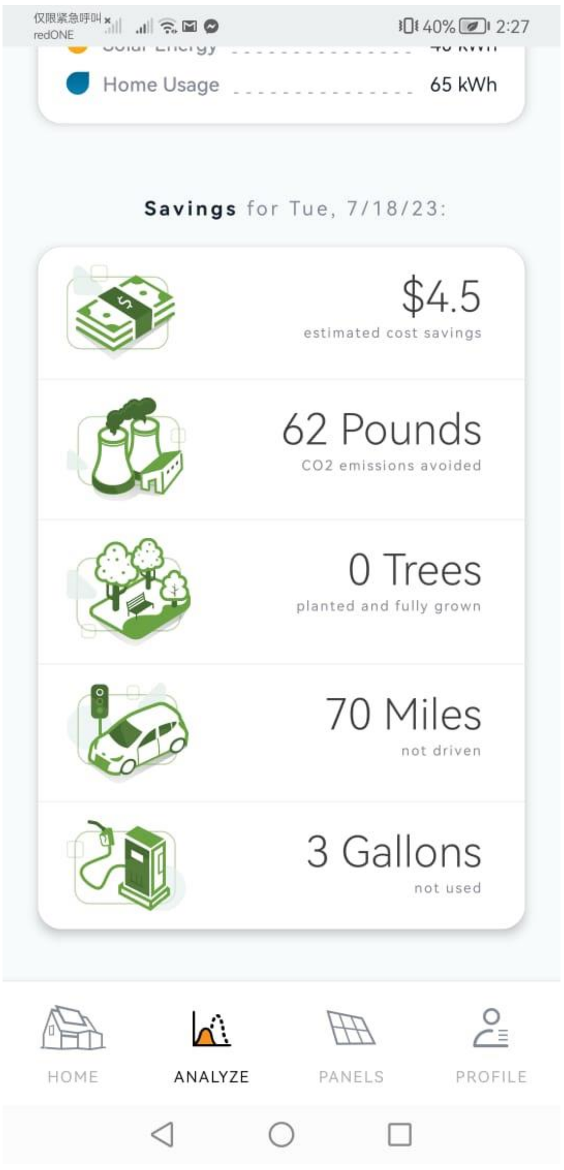


Figure 2.9.4: mySunPower Web-based Interface



Home Consumption		65.0 kWh
🟧	Solar Production	40.0 kWh
🟦	Net Grid Usage (Import)	25.0 kWh

Figure 2.9.5: mySunPower Mobile Application Real-time Monitoring Feature


Home Usage 65 kWh

Savings for Tue, 7/18/23:

$4.5	estimated cost savings
62 Pounds	CO2 emissions avoided
0 Trees	planted and fully grown
70 Miles	not driven
3 Gallons	not used

Figure 2.9.6: mySunPower Data Analysis Feature

Main Features:

- **Monitor solar production, home consumption and battery power flows**

  Visualize energy and power consumption using real-time graphs which allows users to select information in daily, monthly, yearly and lifetime basis.

- **Check historical system performance such as energy consumption data for home with energy consumption meters**

Users can analyze the performance of each solar panel with real-time, life reporting on energy production, updating every 5 minutes. This allows them to explore how different parts of the system are affected by the sun, weather conditions, and seasons.

- **Show live weather data and system alerts for energy decisions**

  Users can analyze the relationship between solar production and the weather by visualizing real-time energy production alongside the current weather conditions. This feature allows them to understand how weather changes impact the energy generated by the solar panels at any given moment. Moreover, users will receive system alerts if the panels are not functioning well. Users can also change the solar panel battery mode based on the system alert.

Review

This application is similar to myTNB as it also provides informative dashboards displaying energy consumption. However, there are notable differences. This application goes beyond and tracks the energy production of solar panels, offering insights into solar power generation.

While myTNB describes electricity costs based on electricity consumption, this application takes a different approach. It primarily focuses on the usage of solar energy, showcasing the total estimated cost savings. This emphasis on solar energy is due to its ability to replace electricity consumption from power stations, resulting in potential savings for users.

Advantages

Interactive infographics show energy production, energy consumption, energy mix, and energy bill saving estimations. They utilize suitable charts, such as a combination of line chart and bar chart, to provide direct insights into the comparison of solar production and home consumption. Additionally, a pie chart is used to display the contribution of solar energy to home usage in the energy mix. These infographics offer an informative and visually engaging way to understand the dynamics of energy usage and savings in a home.

Drawbacks

This application is exclusively designed for SunPower Equinox customers who have purchased their home solar systems. While users can visualize energy consumption, the notifications or alerts provided are limited. For instance, users may receive notifications for switching battery modes or when reaching a certain threshold of energy usage. However, the description below the solar panel feature is not clear enough for users to comprehend or take appropriate actions without contacting a person in charge for assistance.

3) Energy Monitor

- **Monitor real time data about power generation, income and saving**

  The information of current power, daily energy, total energy, daily income and total energy are shown to help user to understand financial benefits of utilizing energy resource.

- **View historical energy consumption data on daily, monthly and yearly basic**

  The users can visualize their energy consumption data using line graph that enable users to gain better understanding on the energy consumption pattern throughout different period.

- **Promote green environment by showing resources saved**

  The total number of trees planted today, the overall count of trees planted, the carbon offset realised today, and the overall carbon offset are all included in the report. Users can clearly see the beneficial effect their eco-friendly behaviours are having on the environment because these metrics are shown in terms of both trees and tonnes.

Review

The interface design and energy data visualisation techniques of the application contribute to its relative lack of attractiveness when compared to earlier applications. The user experience and pleasure could be impacted if the user interface is not as engaging or intuitive.

In addition, several elements of the application provide information that users may find unneeded or meaningless for an accurate analysis of their energy usage. Such information may be confusing or prevent customers from learning important information about their energy usage habits.

However, this application shows its efforts on persuading users to protect environment by lowering energy consumption which is not performed by other application.

Advantages

Through its savings feature, this application motivates users to use energy resources and protect the environment. Users obtain a clear and direct awareness of the significance of their own personal energy consumption habits as well as how their energy consumption behaviour is strongly tied to environmental issues. The programme seeks to empower users to make thoughtful decisions and embrace more sustainable energy consumption practises, contributing to a greener and healthier planet by drawing attention to the possible savings and environmental impact.

Drawbacks

Most real-time information about electricity generation is supplied in numerical format only, without any visual aids. Users find it challenging to understand the trends of energy usage over time and contrast it with their prior data due to the absence of visualisation. Additionally, without context or comparisons, the statistics on daily and overall energy use may not be meaningful enough for people to understand.

Since the system design just provides a straightforward line graph to represent energy consumption, it lacks interactive functionalities such as notification and alert or goal setting for energy consumption.

Table 2.9.1: Comparison of Home Energy Monitoring Application

Aspect	myTNB	mySunPower	Energy Monitor
Platform	Android, iOS	Android, iOS	iOS, web app
Used In	Malaysia	More than 100 countries	United States
Bill Viewing	View and download electricity bills	Not applicable	Not applicable
Consumption Monitoring	Track energy usage and expenditure	Not applicable	Monitor real-time data
Payment Methods	Online banking, credit/debit card	Not applicable	Not applicable
Feedback Submission	Submit questions and get responses	Not applicable	Not applicable
Self-Meter Reading	Apply for self-meter reading service	Yes	Yes
Notifications	Receive personalized updates	Yes	Yes

Multi-Account Management	Manage multiple TNB accounts	Not applicable	Not applicable
Solar Production Tracking	Not applicable	Monitor solar production and battery flows	Not applicable
Historical Performance	Yes	Yes	Yes
Weather Integration	Not applicable	Show live weather data and system alerts	Not applicable
Sharing and Alerts	Yes	Share solar insights, change battery mode	Not applicable
Focus	Energy consumption and payment	Solar energy production and consumption	Overall energy monitoring and eco-friendliness
Advantages	Convenient bill viewing, transparent billing structure, multi-account management	Real-time solar data tracking, insight into energy generation, savings	Encourages eco-friendly behavior, clear savings visualization
Drawbacks	Limited access for tenants,	Exclusive for SunPower	Less engaging UI, unnecessary

	unstable app, payment concerns	customers, limited notifications clarity	information, lack of visualization

2.10 Summary

This literature review focuses on energy consumption and monitoring in Malaysia by highlighting key sectors like industry, residential, and commercial contributing to electricity consumption. Overall, the Malaysian government has actively participated in sustainable energy development through policies such as the National Energy Efficiency Action Plan. The successful integration of energy monitoring systems in manufacturing and commercial buildings has brought confidence in the adoption of Home Energy Monitoring adoption and the potential for residents to manage energy consumption effectively. Next, the role of the Home Energy Monitoring System in empowering residents to perform energy practices through real-time feedback and user engagement is discussed.

Besides, the review covers the importance of real-time systems, particularly in manufacturing and their characteristics, which are in line with the real-time energy monitoring features in Home Energy Monitoring Systems. Furthermore, it also introduces the role of IoT in energy monitoring and management and its adoption in various sectors.

Next, the AI-Based Residential Monitoring project is introduced to understand the connection of this project with the industry-linked company project.

Moreover, similar applications, including myTNB, mySunPower and Energy Monitor, are being compared by showcasing their features and drawbacks.

Finally, smart meters provide the foundation for seamless integration with the home energy monitoring system to collect real-time energy consumption data. Therefore, this literature review shows the overview of TNB’s smart meter implementation in Malaysia by emphasizing benefits, security, and safety, as well as its phased rollout across the country to enhance energy monitoring among Malaysian residents.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter provides exploration of chosen development and research methodology employed within this project. Besides, work break-down structure and Grant Chart will be provided to describe the overall project workflow. Lastly, the technologies and tools involved including hardware, software programming language, application and frameworks selected will be discussed.

3.2 Project Methodology

3.2.1 Development Methodology

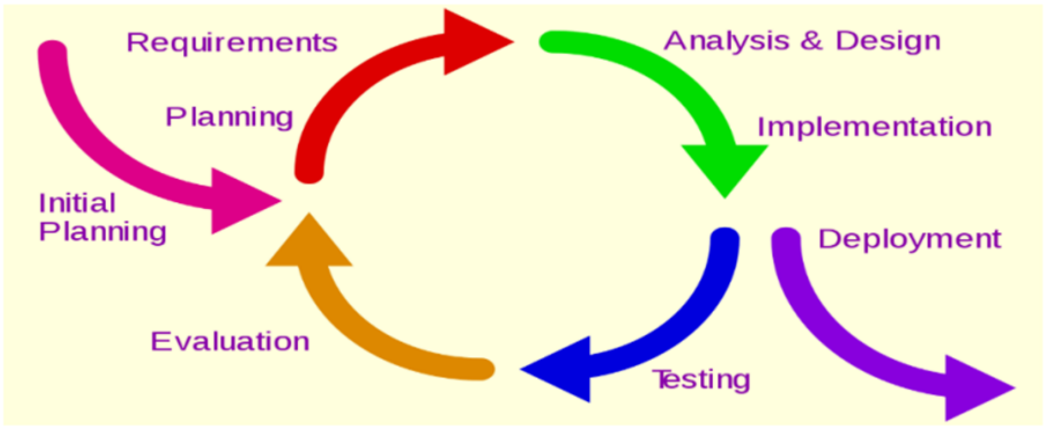


Figure 3.2.1: Iterative and Incremental Development Model (Jevon, 2009)

The development methodology selected for this project is an iterative and incremental development methodology. As shown in Figure 3.2.1.1, the main phases of this methodology including planning, analysing and designing, implementing, testing, evaluating and deploying. After the initial planning stages, planning, analysis and design, implementation, testing and evaluation phases were conducted by the researcher iteratively until all requirements were fulfilled and ready for deployment. This allowed the project to develop iteratively, responding to adjustments and revisions based on the results of each cycle.

Iterative development of the Home Energy Monitoring System meant that high-priority elements were prioritised first, and then lower-priority features were incorporated. This methodology was chosen for HEMS because it can adapt to changing requirements while guaranteeing that higher-priority functionalities are handled first. The researcher was instructed to report progress on development to the cooperating company for further improvement and modification of system functionalities. The iterative technique permitted adaptation as user needs and energy consumption patterns can change. The methodology's feedback-driven design allowed for ongoing improvement, which aided in the development of an improved and optimised system. Additionally, the resulting product was assessed and improved upon with each iteration, aiding in the continuous improvement of the user experience. The incremental technique made it possible to integrate these complicated features gradually, ensuring that each component was fully created, examined, and improved before being combined. This reduced the risks brought on by system complexity and made it possible to thoroughly test the system at every level, so ensuring its dependability.

3.2.2 Initial Planning

3.2.2.1 Requirement Gathering and Elicitation

Home Energy Monitoring system requirements were gathered and elicited using a combination of company-provided specifications, in-depth discussions with the supervisor, site visits, and informal interviews with company representatives. Moreover, questionnaire was distributed starting from 20 August 2023 to 30 September 2023 to get residents' viewpoint on Home Energy Monitoring system. Furthermore, relevant research papers and application comparisons were used to confirm the system's originality. Finally, the requirements were finalized based on the analysis of collected results.

3.2.2.2 Informal Interview and Questionnaire

The site visit and informal interview both offered insightful perspectives into the viewpoint of the organisation. Through these encounters, it was possible to gain a deeper understanding of what they needed from the Home Energy Monitoring System (HEMS) and what they wanted to achieve with it. Discussions revealed that they considered real-time energy monitoring to be essential functionality. It became clear that the company wanted to spread knowledge about energy and environmental practices, placing a big emphasis on user-friendly interfaces. These exchanges further highlight the significance of post-deployment support for system maintenance and upgrades. Through these discussions, the company's goals for the HEMS and its particular requirements were made clear, effectively driving the project's development.

The questionnaire was conducted using Google Forms to collect large numbers of respondents from Malaysia within one month, starting from 13 August 2023 to 13 September 2023. The extensive questionnaire probed into factors like housing type, occupants, awareness of consumption patterns, energy-saving practises, familiarity with Home Energy Management Systems (HEMS), use of smart metres, and preferences for HEMS features. It covered various aspects of electricity consumption and energy management behaviours. Responses from participants revealed information about their knowledge of energy consumption, usage patterns, past exposure to energy management technologies, and openness to implementing HEMS. This questionnaire was distributed to residents of the existing myTNB smart meter and residents who had never used a smart meter before. The respondents of smart meter users need to be improved due to the limited deployment of TNB smart meters in selected locations of Peninsular Malaysia, mainly at Melaka, Putrajaya, and Klang Valley. This extensive analysis revealed potential obstacles, driving forces, and desired HEMS features, providing important information for the creation of a powerful and user-centred home energy monitoring system.

3.2.2.3 Internship with Industry Linked Company

The developer had been given the opportunity to conduct an internship with the project industry-linked company MIMOS Bhd for 3 months and was involved in its project named AI Based Residential Energy Monitoring (AIREM) project. The mobile application and web-based dashboard developed by the researcher were part of this project and aimed to deliver energy consumption data from IoT devices in residential areas. The developer gained an understanding of the entire project, starting from the development of the home energy monitoring device to detect household energy consumption. This includes the processing and transformation of data into readable energy consumption data, the AI algorithms used to disaggregate the energy consumption, and finally, the presentation of energy consumption data, all of which align with the aim of developing this mobile application and web-based dashboard. Additionally, the developer had several discussions with the company team to understand their requirements for user interface and functionalities, ensuring that the proposed software meets their needs.

3.2.2.4 Review on Existing Systems

Three comparable applications were examined and compared to understand the workflow of energy monitoring from energy monitoring devices to the applications. The common energy monitoring devices are smart meters and solar power panels. Moreover, the presentation of real time energy data was also being investigated. Most of the applications from app stores and research papers used bar charts to present these data with specific period.

3.2.2.5 Schedule Project

A comprehensive work breakdown structure was painstakingly created for the Home Energy Monitoring System (HEMS) project to define the numerous activities inside each project phase and milestone. This breakdown ensured a detailed comprehension of all crucial actions taken. A Gantt chart was constructed and synced with the work breakdown structure to visually represent task start dates, end dates, and durations for each phase. The development team was given a visual depiction via this dynamic technology to continuously track and manage work progress, allowing for the prompt completion of project deliverables. Thus, the effective management of the HEMS project towards its successful execution was made possible by the integrated use of the WBS and Gantt chart.

3.2.3 Analysis and Design

Project scope was analysed using several diagrams including use case diagram, entity relationship diagram, class diagram and data flow diagram to understand system’s structure and architecture before conducting implementation started. Use case description for every listed use case were also being prepared to understand the workflow of every functionality provided.

Simple system prototypes for both web based, and mobile based application were generated to visualize the design of the system before actual implementation to have a general view on the final deliverable.

3.2.4 Implementation and testing

3.2.4.1 Identified High-Priority Features

The critical and high-priority features of HEMS that offer immediate value to user were identified. The sequence of functionality implementation was based on the priority of the features.

3.2.4.2 Iteration 1: Initial iteration

The core functionalities of the HEMS were being developed as identified. The basic version of the mobile application and web-based dashboard were crafted. The basic backend architecture of this project was implemented using AWS Amplify, AWS DynamoDB and AWS AppSync. The connection between the energy monitoring device simulator that acts as the IoT device with AWS DynamoDB and the real-time connection of energy consumption table with the mobile application were established using MQTT protocol and WebSocket. Functionalities such as visualization of real-time data and view historical data and analysis were implemented for energy consumption tracking. Unit testing and integration testing were conducted to verify the functionality of the core features.

3.2.4.2.1Gathered User Feedback for Evaluation

After completing the initial iteration, the developer conducted a meeting with the cooperated company representatives to gather feedback for further improvement. The company team had reviewed the project and provided recommendations on further improvements of the presentation of energy consumption data.

3.2.4.3 Iteration 2: Functionalities Enhancement and Development

The developer prioritized feature improvements and adjustments based on gathered feedback from stakeholder. Proposed solutions for improvement were presented to the company for approval, ensuring alignment with project goals and objectives. She modified the presentation of energy consumption data based on recommendation such as refining the selection of attributes to enhance understanding and introducing more complex data visualizations.

3.2.4.4 Iteration 3: Advanced Functionalities

The developer introduced advanced notification and alert feature aimed at notifying users about their energy consumption pattern in real-time. The unit testing was done for this notification and alert feature to make sure that seamless integration of mobile application and AWS services on minimizing latency. The developer implemented modifications to the configuration of the code to maintain the consistency in the backend architecture across both Android and iOS platforms. Next, testing was conducted to verify the functionality of the updated configuration across different operating systems.

3.2.4.5 Iteration 4: Iterative Enhancements

The system's functionalities and user interfaces were further improved by the developer based on the results of user testing. Added features and enhancements were added as a result of user suggestions such as user profile page, setting page and report problem page. The user interface design was improved by collaborating with friends from a design course to make it more user-friendly.

3.2.4.6 Final Iteration and Testing

To verify the system's coherence and seamlessness, the developer thoroughly tested the final version and included any remaining functionality with unit testing, integration testing and system testing.

3.2.5 System Usability Scale Evaluation and Code Quality Analysis

System Usability Scale (SUS) evaluation and code quality analysis were conducted by gathering both expert and novice users to assess the usability of the system.

3.2.6 Deployment

The fully developed and tested HEMS was deployed to production servers and is accessible to users. Next, the documentation of the system was created including the explanation of architecture of the system, the introduction of the functionalities, the screenshot of the system output and the presentation slide.

3.3 Work Breakdown Structure

1 Planning

- 1.1 Analyze Project Title
- 1.2 Conduct Informal Interview with Company Representative
  - 1.2.1 Analyze Company Project Background
- 1.3 Study Background of the Problem
  - 1.3.1 Relate Company Problem to Project Problem
- 1.4 Define Problem Statement
- 1.5 Define Project Objective
- 1.6 Propose Project Solution
  - 1.6.1 Investigate Existing Energy Monitoring Applications
  - 1.6.2 Study Similar HEMS Research Papers
  - 1.6.3 Consider Company Support Tools
  - 1.6.4 Select Appropriate Tools
  - 1.6.5 Conclude Project Solution
- 1.7 Suggest Project Approach
  - 1.7.1 Suggest Research Approach
    - 1.7.1.1 Draft Questionnaire for Residents
  - 1.7.2 Suggest Development Approach
- 1.8 Define Project Scope
  - 1.8.1 Recognize Target Users

1.8.2 Define System Scope

1.8.3 Identify Modules Covered

1.9 Gather and Elicit Requirements

1.9.1 Conduct Questionnaire

1.9.1.1 Distribute Questionnaire

1.9.1.2 Collect Questionnaire

1.9.1.3 Analyze Questionnaire Result

1.9.2 Conduct Informal Interview

1.9.2.1 Prepare Interview Questions

1.9.2.2 Conduct Interview with Company Representative

1.9.2.2.1 Record and Analyze Interview Result

1.9.2.3 Conduct Site Visit with Company

1.9.2.3.1 Understand Energy Monitoring Device

1.9.2.3.2 View Demonstration of Energy Monitoring Device

1.9.3 Review Similar Systems

1.9.3.1 Review Similar Home Energy Monitoring System Research Papers

1.9.3.2 Review Home Energy Monitoring System on Apple Store

1.9.3.3 Review Similar Home Energy Monitoring System on Google Play

1.9.3.4 Review Other Energy Monitoring Systems

1.9.3.5 Identify Common HEMS Features

1.9.3.6 Identify Ways of Real-Time Energy Visualization

1.9.3.7 Identify Type of Energy Monitoring Devices

1.9.3.8 Compare System Developed in Malaysia with Others

1.9.4 Literature Review

1.9.4.1 Understand Energy Consumption in Malaysia

1.9.4.2 Identify Energy Monitoring in Malaysia

1.9.4.3 Identify Real-Time System

1.9.4.3.1 Define Real-Time Concept for HEMS

1.9.4.4 Understand Internet of Things

1.9.4.5 Identify Energy Consumption and Monitoring Globally

1.9.5 Enhance Provided Features

1.9.6 Draft Requirements

1.10 Schedule Project

1.10.1 Establish Work Breakdown Structure

1.10.1.1 Define Main Activities for Different Phases

1.10.1.2 Subdivide Activities within Key Activities

1.10.2 Develop Gantt Chart

1.10.2.1 Determine Duration

1.10.2.2 Determine Task Dependencies

1.10.2.3 Draw Gantt Chart

1.10.2.4 Schedule Timeline

2 Analysis and Design

  2.1 Draw Use Case Diagrams

  2.2 Write Use Case Descriptions

  2.3 Design User Interface Diagrams

  2.4 Design Data Flow Diagrams

  2.5 Create Low-Level Prototype

    2.5.1 Prepare Wireframe for Mobile Application

    2.5.2 Prepare Wireframe for Web Application

  2.6 Create High-Level Prototype

    2.6.1 Prepare Design Prototype for Mobile Application

    2.6.2 Prepare Design Prototype for Web Application

3 Iteration 1 (Core Functionalities Development)

  3.1 Project Initiation and Planning

  3.2 Design and Analyze Core Functionalities Architecture

  3.3 Implement Core Functionalities

    3.3.1 Setup Database

    3.3.2 Implement Register User Account Functionality

    3.3.3 Implement Login User Account Functionalities

    3.3.4 Implement Manage User Account Functionality

    3.3.5 Develop Real-Time Data Collection Mechanism

3.3.6 Implement Visualization of Real-Time Data

3.3.7 Integrate Historical Data Analysis

3.3.8 Craft Mobile Application and Dashboard

3.4 Testing and Quality Assurance

3.4.1 Unit Testing of Core Functionalities

3.4.1.1 Unit Testing for Register User Account Functionality

3.4.1.2 Unit Testing for Login Account Functionality

3.4.1.3 Unit Testing for Manage User Account Functionality

3.4.1.4 Unit Testing for Real-Time Energy Monitoring Function

3.4.1.5 Unit Testing for View Historical Data and Analysis Function

3.4.2 Integration Testing of Core Functionalities

3.4.3 Verify Real-Time Data Collection and Visualization

3.4.4 Ensure Historical Data Analysis Accuracy

3.5 Report and Documentation

3.5.1 Document Core Functionality Specification

3.6 Iteration 1 Review and Feedback

3.6.1 Demonstrate Initial Iteration HEMS Product to Stakeholders

3.6.2 Collect Stakeholder Feedback and Suggestions

3.6.3 Evaluate and Incorporate Feedback for Iteration 2

4 Iteration 2 (Functionalities Enhancement and Development)

4.1 Review and Analysis of Stakeholders' Feedback

  4.1.1 Prioritize Functionalities and Adjustments

4.2 Implement Energy Consumption Optimization Functionality

  4.2.1 Define Criteria for Personalized Advice Generation

  4.2.2 Develop Algorithms for Energy Usage Recommendations

  4.2.3 Implement User-Specific Energy Saving Tips

  4.2.4 Integrate Personalized Advice into Mobile App and Dashboard

4.3 Complex Data Visualizations for Dashboard

  4.3.1 Identify Data Elements for Complex Visualization

  4.3.2 Design Enhanced Data Visualization Components

  4.3.3 Develop Interactive Charts and Graphs

  4.3.4 Integrate Complex Data Visualizations into Dashboard

4.4 Usability Testing and Interface Improvement

  4.4.1 Plan Usability Testing Scenarios

  4.4.2 Conduct Usability Testing with Real Users

  4.4.3 Analyze Usability Testing Results and Feedback

  4.4.4 Identify UI/UX Improvement Areas

  4.4.5 Implement Interface Enhancements based on Testing Results

4.5 Report and Documentation

  4.5.1 Document Enhanced Features and Functionalities

4.6 Iteration Review and Feedback

4.6.1 Demonstrate Functionalities Enhancement to Stakeholders

4.6.2 Collect Stakeholder Feedback and Suggestions

4.6.3 Evaluate and Incorporate Feedback for Subsequent Phases

5 Iteration 3 (Advanced Functionalities)

5.1 Device-Specific Consumption Tracking

5.1.1 Define Data Collection Mechanism for Device Consumption

5.1.2 Develop Device Consumption Visualization Components

5.1.3 Integrate Device-Specific Consumption Tracking

5.2 Notification and Alert for Energy Consumption

5.2.1 Design Energy Consumption Thresholds for Notifications

5.2.2 Develop Real-time Monitoring for Alert Triggering

5.2.3 Implement Notification and Alert Mechanisms

5.2.4 Incorporate Notifications and Alerts into Mobile App

5.3 User Testing and Feedback Collection

5.3.1 Plan Comprehensive User Testing Scenarios

5.3.2 Invite Larger User Base for Testing

5.3.3 Collect Detailed User Feedback and Observations

5.3.4 Analyze Testing Results and Feedback

5.3.5 Address Identified Issues and Concerns

5.3.6 Enhance System Based on User Testing Insights

5.4 Documentation and Reporting

5.4.1 Document Advanced Functionalities and Features

5.5 Iteration Review and Feedback

5.5.1 Demonstrate Advanced Functionalities to Stakeholders

5.5.2 Collect Stakeholder Feedback and Suggestions

5.5.3 Evaluate and Incorporate Feedback for Further Phases

6 Iteration 4 (Enhancements)

6.1 User Testing Feedback Incorporation

6.1.1 Review User Testing Results and Feedback

6.1.2 Identify Key Areas for Enhancements

6.1.3 Incorporate User Feedback into Functionality and UI

6.2 Added Features and User-Suggested Enhancements

6.2.1 Gather and Evaluate User Suggestions and Ideas

6.2.2 Prioritize and Select Functionalities for Improvement

6.3 Usability and Performance Improvements

6.3.1 Identify Usability and Performance Metrics

6.3.2 Analyze and Assess Current Usability and Performance

6.3.3 Plan and Execute Usability and Performance Improvements

6.4 Final Testing

6.4.1 Test Final Version of HEMS

6.4.2 Verify Coherence and Functionality Alignment

7 Closing

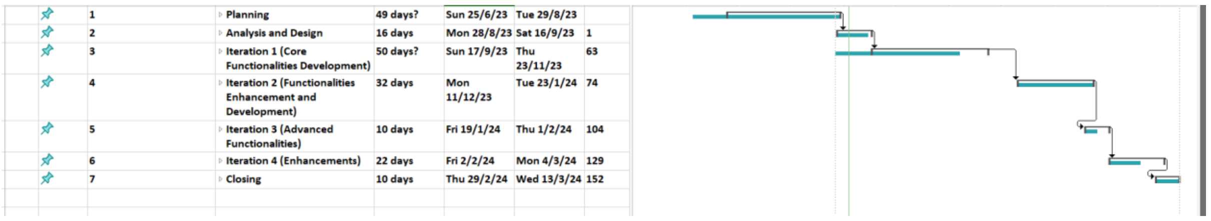
7.1 Conduct System Usability Scale Evaluation

7.2 Finalize System Documentation

7.3 Prepare Presentation Slide

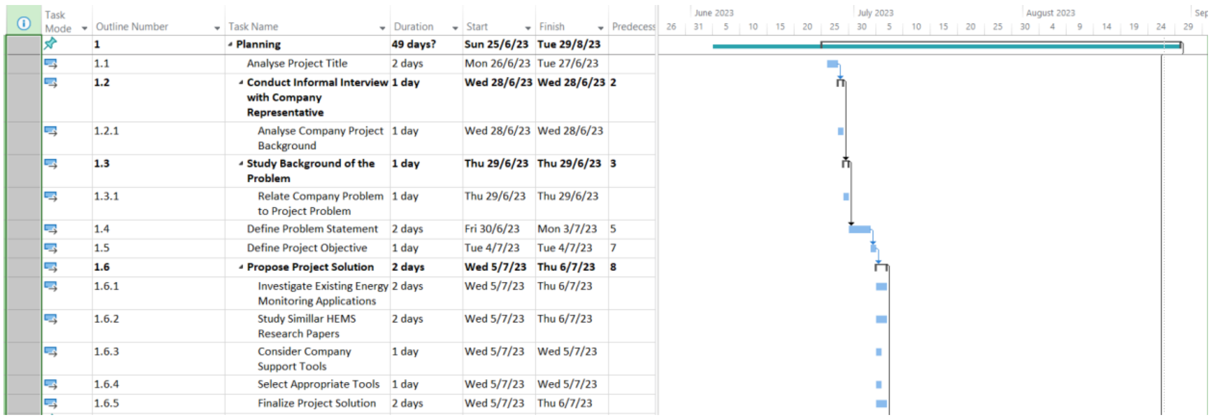
7.4 Create System Poster

3.4 Gantt Chart



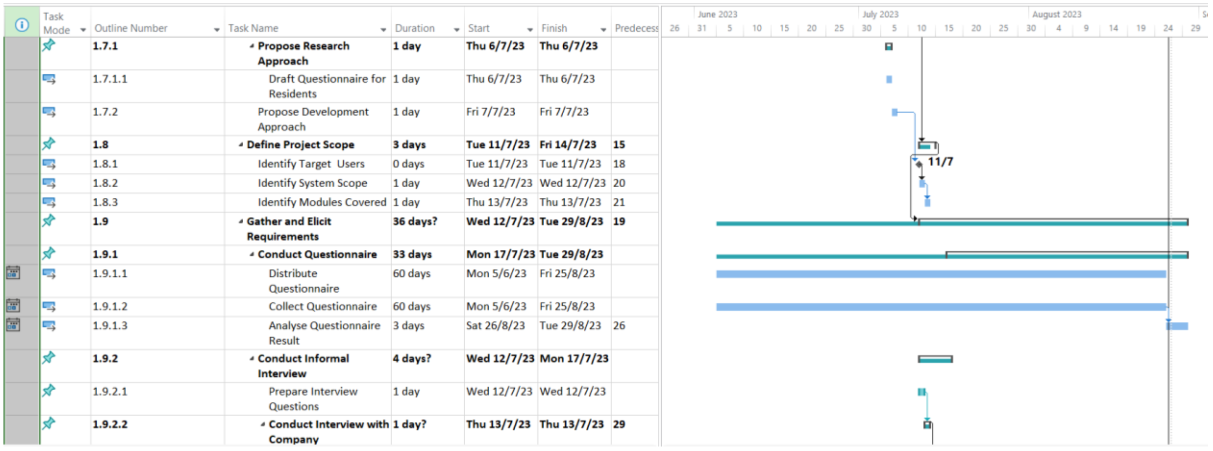
1	**Planning**	49 days?	Sun 25/6/23	Tue 29/8/23	
2	**Analysis and Design**	16 days	Mon 28/8/23	Sat 16/9/23	1
3	**Iteration 1 (Core Functionalities Development)**	50 days?	Sun 17/9/23	Thu 23/11/23	63
4	**Iteration 2 (Functionalities Enhancement and Development)**	32 days	Mon 11/12/23	Tue 23/1/24	74
5	**Iteration 3 (Advanced Functionalities)**	10 days	Fri 19/1/24	Thu 1/2/24	104
6	**Iteration 4 (Enhancements)**	22 days	Fri 2/2/24	Mon 4/3/24	129
7	**Closing**	10 days	Thu 29/2/24	Wed 13/3/24	152

Figure 3.4.1: Overview of Work Breakdown Structure



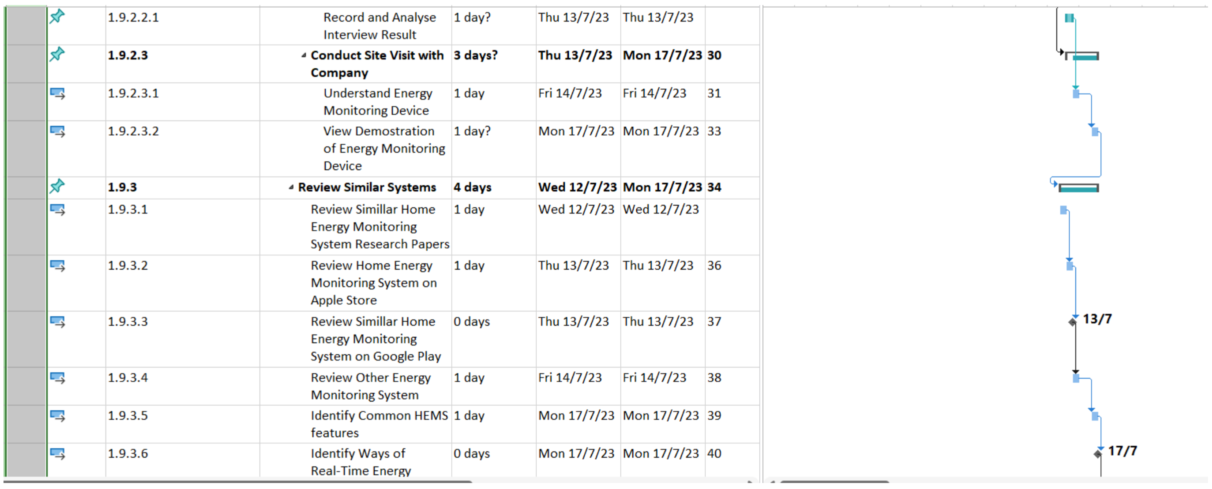
Outline Number	Task Name	Duration	Start	Finish	Predecess
1	**Planning**	**49 days?**	**Sun 25/6/23**	**Tue 29/8/23**	
1.1	Analyse Project Title	2 days	Mon 26/6/23	Tue 27/6/23	
1.2	**Conduct Informal Interview with Company Representative**	**1 day**	**Wed 28/6/23**	**Wed 28/6/23**	**2**
1.2.1	Analyse Company Project Background	1 day	Wed 28/6/23	Wed 28/6/23	
1.3	**Study Background of the Problem**	**1 day**	**Thu 29/6/23**	**Thu 29/6/23**	**3**
1.3.1	Relate Company Problem to Project Problem	1 day	Thu 29/6/23	Thu 29/6/23	
1.4	Define Problem Statement	2 days	Fri 30/6/23	Mon 3/7/23	5
1.5	Define Project Objective	1 day	Tue 4/7/23	Tue 4/7/23	7
1.6	**Propose Project Solution**	**2 days**	**Wed 5/7/23**	**Thu 6/7/23**	**8**
1.6.1	Investigate Existing Energy Monitoring Applications	2 days	Wed 5/7/23	Thu 6/7/23	
1.6.2	Study Simillar HEMS Research Papers	2 days	Wed 5/7/23	Thu 6/7/23	
1.6.3	Consider Company Support Tools	1 day	Wed 5/7/23	Wed 5/7/23	
1.6.4	Select Appropriate Tools	1 day	Wed 5/7/23	Wed 5/7/23	
1.6.5	Finalize Project Solution	2 days	Wed 5/7/23	Thu 6/7/23	

Figure 3.4.2: Planning Phase Timeline



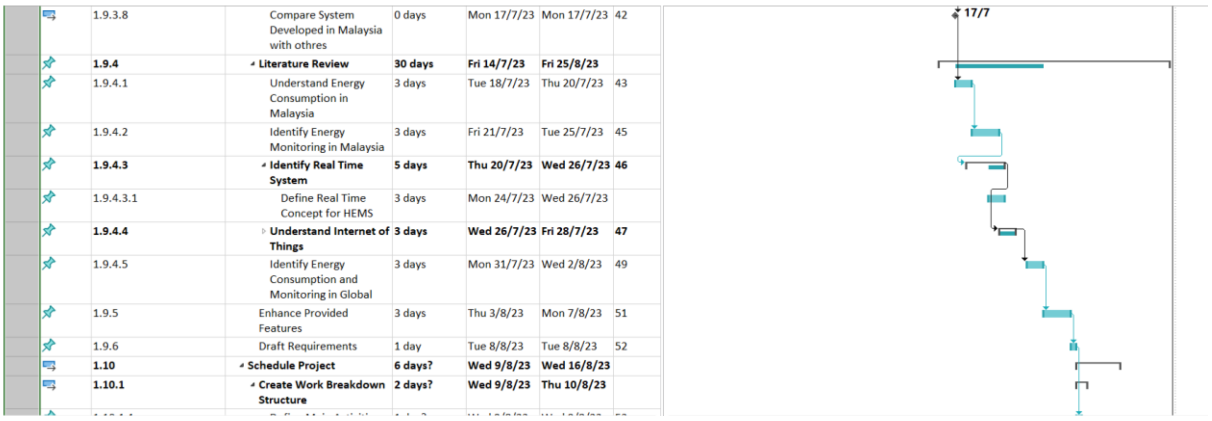
Outline Number	Task Name	Duration	Start	Finish	Predecess
1.7.1	**Propose Research Approach**	**1 day**	**Thu 6/7/23**	**Thu 6/7/23**	
1.7.1.1	Draft Questionnaire for Residents	1 day	Thu 6/7/23	Thu 6/7/23	
1.7.2	Propose Development Approach	1 day	Fri 7/7/23	Fri 7/7/23	
1.8	**Define Project Scope**	**3 days**	**Tue 11/7/23**	**Fri 14/7/23**	**15**
1.8.1	Identify Target Users	0 days	Tue 11/7/23	Tue 11/7/23	18
1.8.2	Identify System Scope	1 day	Wed 12/7/23	Wed 12/7/23	20
1.8.3	Identify Modules Covered	1 day	Thu 13/7/23	Thu 13/7/23	21
1.9	**Gather and Elicit Requirements**	**36 days?**	**Wed 12/7/23**	**Tue 29/8/23**	**19**
1.9.1	**Conduct Questionnaire**	**33 days**	**Mon 17/7/23**	**Tue 29/8/23**	
1.9.1.1	Distribute Questionnaire	60 days	Mon 5/6/23	Fri 25/8/23	
1.9.1.2	Collect Questionnaire	60 days	Mon 5/6/23	Fri 25/8/23	
1.9.1.3	Analyse Questionnaire Result	3 days	Sat 26/8/23	Tue 29/8/23	26
1.9.2	**Conduct Informal Interview**	**4 days?**	**Wed 12/7/23**	**Mon 17/7/23**	
1.9.2.1	Prepare Interview Questions	1 day	Wed 12/7/23	Wed 12/7/23	
1.9.2.2	**Conduct Interview with Company**	**1 day?**	**Thu 13/7/23**	**Thu 13/7/23**	**29**

Figure 3.4.3: Planning Phase Timeline (Continued)



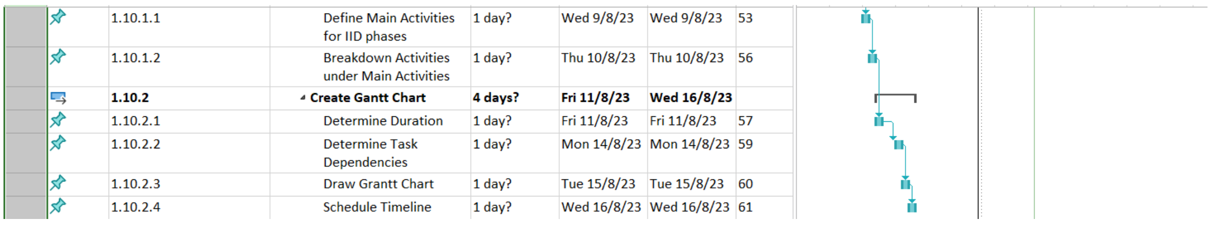
1.9.2.2.1	Record and Analyse Interview Result	1 day?	Thu 13/7/23	Thu 13/7/23	
1.9.2.3	**Conduct Site Visit with Company**	**3 days?**	**Thu 13/7/23**	**Mon 17/7/23**	**30**
1.9.2.3.1	Understand Energy Monitoring Device	1 day	Fri 14/7/23	Fri 14/7/23	31
1.9.2.3.2	View Demostration of Energy Monitoring Device	1 day?	Mon 17/7/23	Mon 17/7/23	33
1.9.3	**Review Similar Systems**	**4 days**	**Wed 12/7/23**	**Mon 17/7/23**	**34**
1.9.3.1	Review Simillar Home Energy Monitoring System Research Papers	1 day	Wed 12/7/23	Wed 12/7/23	
1.9.3.2	Review Home Energy Monitoring System on Apple Store	1 day	Thu 13/7/23	Thu 13/7/23	36
1.9.3.3	Review Simillar Home Energy Monitoring System on Google Play	0 days	Thu 13/7/23	Thu 13/7/23	37
1.9.3.4	Review Other Energy Monitoring System	1 day	Fri 14/7/23	Fri 14/7/23	38
1.9.3.5	Identify Common HEMS features	1 day	Mon 17/7/23	Mon 17/7/23	39
1.9.3.6	Identify Ways of Real-Time Energy	0 days	Mon 17/7/23	Mon 17/7/23	40

Figure 3.4.4: Planning Phase Timeline (Continued)



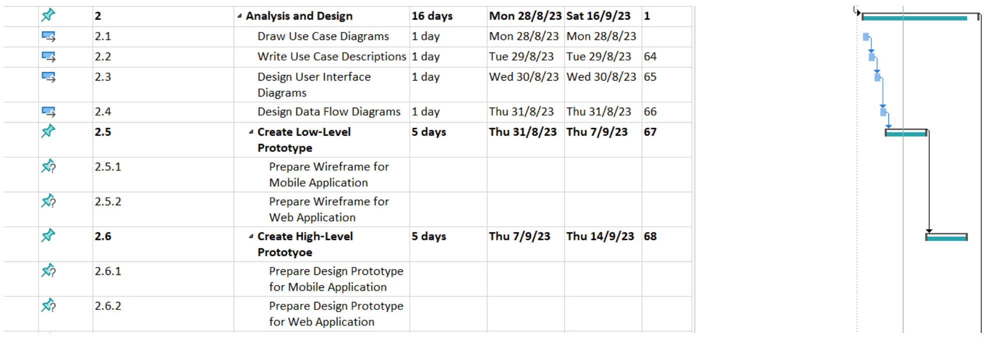
1.9.3.8	Compare System Developed in Malaysia with othres	0 days	Mon 17/7/23	Mon 17/7/23	42
1.9.4	**Literature Review**	**30 days**	**Fri 14/7/23**	**Fri 25/8/23**	
1.9.4.1	Understand Energy Consumption in Malaysia	3 days	Tue 18/7/23	Thu 20/7/23	43
1.9.4.2	Identify Energy Monitoring in Malaysia	3 days	Fri 21/7/23	Tue 25/7/23	45
1.9.4.3	**Identify Real Time System**	**5 days**	**Thu 20/7/23**	**Wed 26/7/23**	**46**
1.9.4.3.1	Define Real Time Concept for HEMS	3 days	Mon 24/7/23	Wed 26/7/23	
1.9.4.4	**Understand Internet of Things**	**3 days**	**Wed 26/7/23**	**Fri 28/7/23**	**47**
1.9.4.5	Identify Energy Consumption and Monitoring in Global	3 days	Mon 31/7/23	Wed 2/8/23	49
1.9.5	Enhance Provided Features	3 days	Thu 3/8/23	Mon 7/8/23	51
1.9.6	Draft Requirements	1 day	Tue 8/8/23	Tue 8/8/23	52
1.10	**Schedule Project**	**6 days?**	**Wed 9/8/23**	**Wed 16/8/23**	
1.10.1	**Create Work Breakdown Structure**	**2 days?**	**Wed 9/8/23**	**Thu 10/8/23**	

Figure 3.4.5: Planning Phase Timeline (Continued)



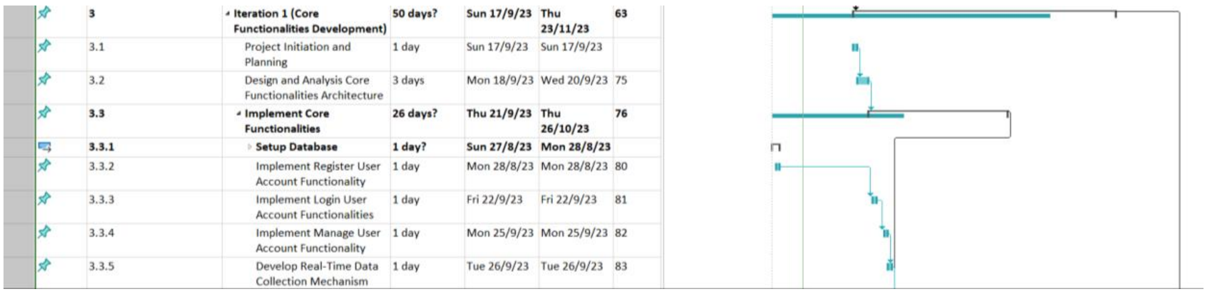
1.10.1.1	Define Main Activities for IID phases	1 day?	Wed 9/8/23	Wed 9/8/23	53
1.10.1.2	Breakdown Activities under Main Activities	1 day?	Thu 10/8/23	Thu 10/8/23	56
1.10.2	**Create Gantt Chart**	**4 days?**	**Fri 11/8/23**	**Wed 16/8/23**	
1.10.2.1	Determine Duration	1 day?	Fri 11/8/23	Fri 11/8/23	57
1.10.2.2	Determine Task Dependencies	1 day?	Mon 14/8/23	Mon 14/8/23	59
1.10.2.3	Draw Grantt Chart	1 day?	Tue 15/8/23	Tue 15/8/23	60
1.10.2.4	Schedule Timeline	1 day?	Wed 16/8/23	Wed 16/8/23	61

Figure 3.4.6: Planning Phase Timeline (Continued)



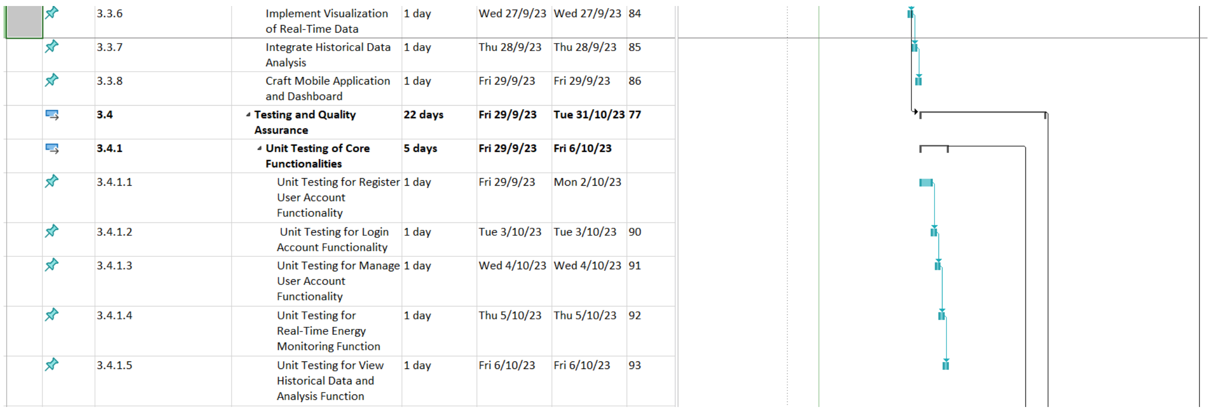
2	**Analysis and Design**	**16 days**	**Mon 28/8/23**	**Sat 16/9/23**	**1**
2.1	Draw Use Case Diagrams	1 day	Mon 28/8/23	Mon 28/8/23	
2.2	Write Use Case Descriptions	1 day	Tue 29/8/23	Tue 29/8/23	64
2.3	Design User Interface Diagrams	1 day	Wed 30/8/23	Wed 30/8/23	65
2.4	Design Data Flow Diagrams	1 day	Thu 31/8/23	Thu 31/8/23	66
2.5	**Create Low-Level Prototype**	**5 days**	**Thu 31/8/23**	**Thu 7/9/23**	**67**
2.5.1	Prepare Wireframe for Mobile Application				
2.5.2	Prepare Wireframe for Web Application				
2.6	**Create High-Level Prototye**	**5 days**	**Thu 7/9/23**	**Thu 14/9/23**	**68**
2.6.1	Prepare Design Prototype for Mobile Application				
2.6.2	Prepare Design Prototype for Web Application				

Figure 3.4.7: Analysis and Design Phase Timeline



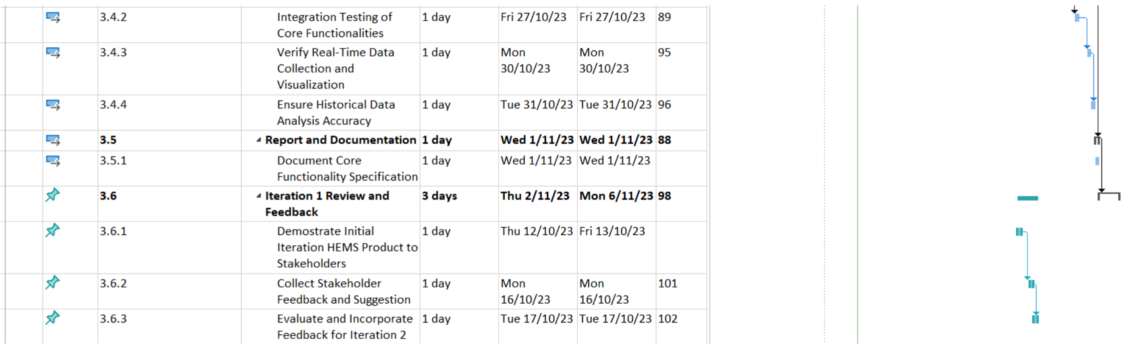
3	**Iteration 1 (Core Functionalities Development)**	**50 days?**	**Sun 17/9/23**	**Thu 23/11/23**	**63**
3.1	Project Initiation and Planning	1 day	Sun 17/9/23	Sun 17/9/23	
3.2	Design and Analysis Core Functionalities Architecture	3 days	Mon 18/9/23	Wed 20/9/23	75
3.3	**Implement Core Functionalities**	**26 days?**	**Thu 21/9/23**	**Thu 26/10/23**	**76**
3.3.1	**Setup Database**	**1 day?**	**Sun 27/8/23**	**Mon 28/8/23**	
3.3.2	Implement Register User Account Functionality	1 day	Mon 28/8/23	Mon 28/8/23	80
3.3.3	Implement Login User Account Functionalities	1 day	Fri 22/9/23	Fri 22/9/23	81
3.3.4	Implement Manage User Account Functionality	1 day	Mon 25/9/23	Mon 25/9/23	82
3.3.5	Develop Real-Time Data Collection Mechanism	1 day	Tue 26/9/23	Tue 26/9/23	83

Figure 3.4.8: Development Iteration 1 Timeline



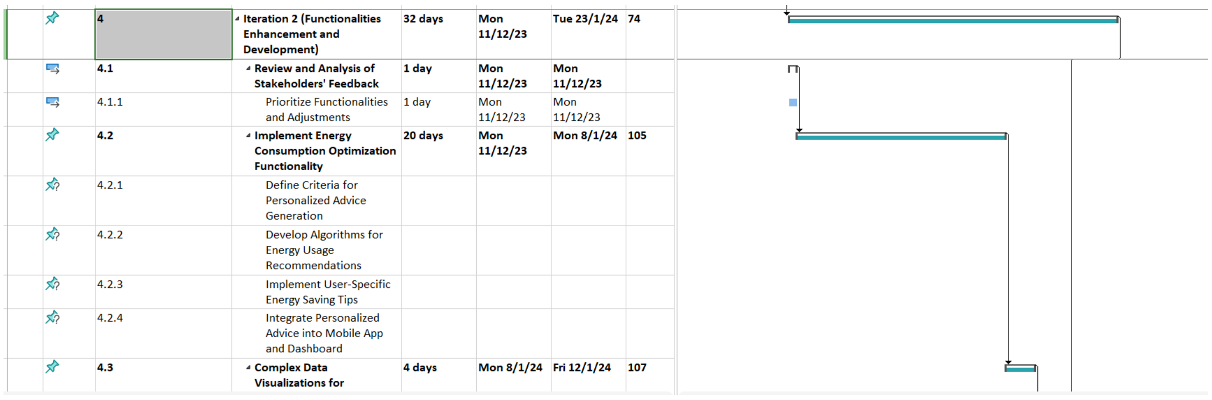
3.3.6	Implement Visualization of Real-Time Data	1 day	Wed 27/9/23	Wed 27/9/23	84
3.3.7	Integrate Historical Data Analysis	1 day	Thu 28/9/23	Thu 28/9/23	85
3.3.8	Craft Mobile Application and Dashboard	1 day	Fri 29/9/23	Fri 29/9/23	86
3.4	**Testing and Quality Assurance**	**22 days**	**Fri 29/9/23**	**Tue 31/10/23**	**77**
3.4.1	**Unit Testing of Core Functionalities**	**5 days**	**Fri 29/9/23**	**Fri 6/10/23**	
3.4.1.1	Unit Testing for Register User Account Functionality	1 day	Fri 29/9/23	Mon 2/10/23	
3.4.1.2	Unit Testing for Login Account Functionality	1 day	Tue 3/10/23	Tue 3/10/23	90
3.4.1.3	Unit Testing for Manage User Account Functionality	1 day	Wed 4/10/23	Wed 4/10/23	91
3.4.1.4	Unit Testing for Real-Time Energy Monitoring Function	1 day	Thu 5/10/23	Thu 5/10/23	92
3.4.1.5	Unit Testing for View Historical Data and Analysis Function	1 day	Fri 6/10/23	Fri 6/10/23	93

Figure 3.4.9: Development Iteration 1 Timeline (Continued)



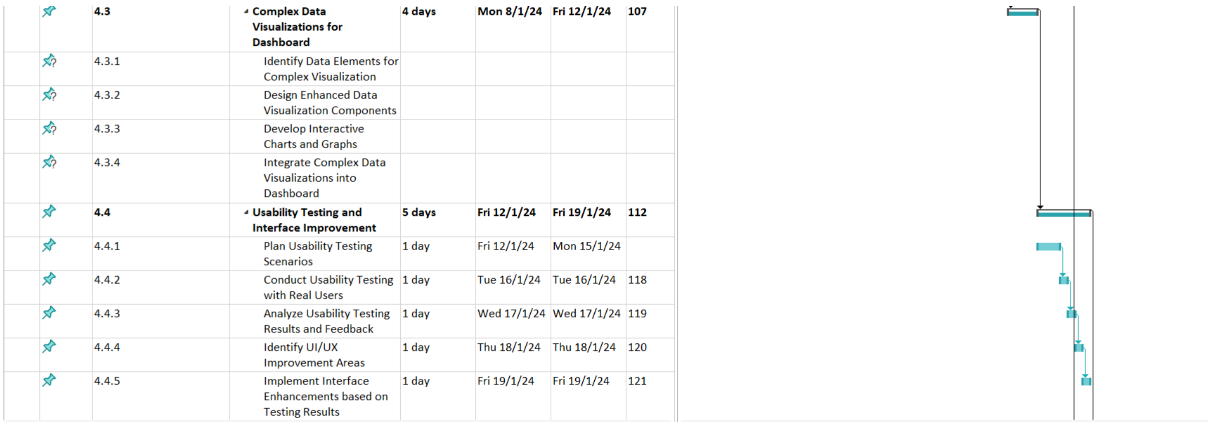
3.4.2	Integration Testing of Core Functionalities	1 day	Fri 27/10/23	Fri 27/10/23	89
3.4.3	Verify Real-Time Data Collection and Visualization	1 day	Mon 30/10/23	Mon 30/10/23	95
3.4.4	Ensure Historical Data Analysis Accuracy	1 day	Tue 31/10/23	Tue 31/10/23	96
3.5	**Report and Documentation**	**1 day**	**Wed 1/11/23**	**Wed 1/11/23**	**88**
3.5.1	Document Core Functionality Specification	1 day	Wed 1/11/23	Wed 1/11/23	
3.6	**Iteration 1 Review and Feedback**	**3 days**	**Thu 2/11/23**	**Mon 6/11/23**	**98**
3.6.1	Demostrate Initial Iteration HEMS Product to Stakeholders	1 day	Thu 12/10/23	Fri 13/10/23	
3.6.2	Collect Stakeholder Feedback and Suggestion	1 day	Mon 16/10/23	Mon 16/10/23	101
3.6.3	Evaluate and Incorporate Feedback for Iteration 2	1 day	Tue 17/10/23	Tue 17/10/23	102

Figure 3.4.10: Development Iteration 1 Timeline (Continued)



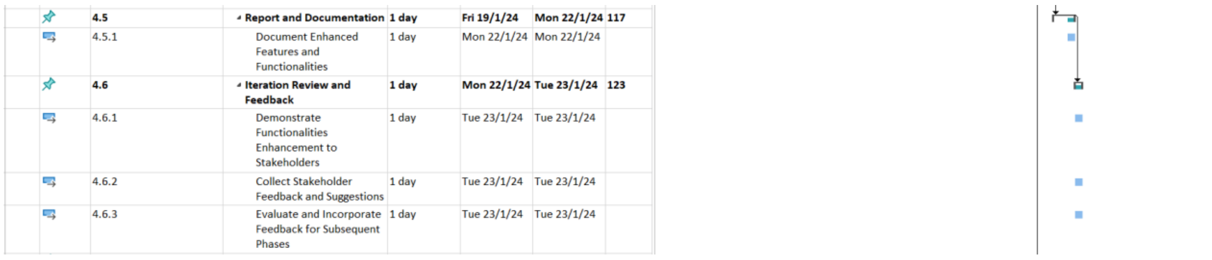
4	Iteration 2 (Functionalities Enhancement and Development)	32 days	Mon 11/12/23	Tue 23/1/24	74
4.1	Review and Analysis of Stakeholders' Feedback	1 day	Mon 11/12/23	Mon 11/12/23	
4.1.1	Prioritize Functionalities and Adjustments	1 day	Mon 11/12/23	Mon 11/12/23	
4.2	Implement Energy Consumption Optimization Functionality	20 days	Mon 11/12/23	Mon 8/1/24	105
4.2.1	Define Criteria for Personalized Advice Generation				
4.2.2	Develop Algorithms for Energy Usage Recommendations				
4.2.3	Implement User-Specific Energy Saving Tips				
4.2.4	Integrate Personalized Advice into Mobile App and Dashboard				
4.3	Complex Data Visualizations for	4 days	Mon 8/1/24	Fri 12/1/24	107

Figure 3.4.11: Development Iteration 2 Timeline



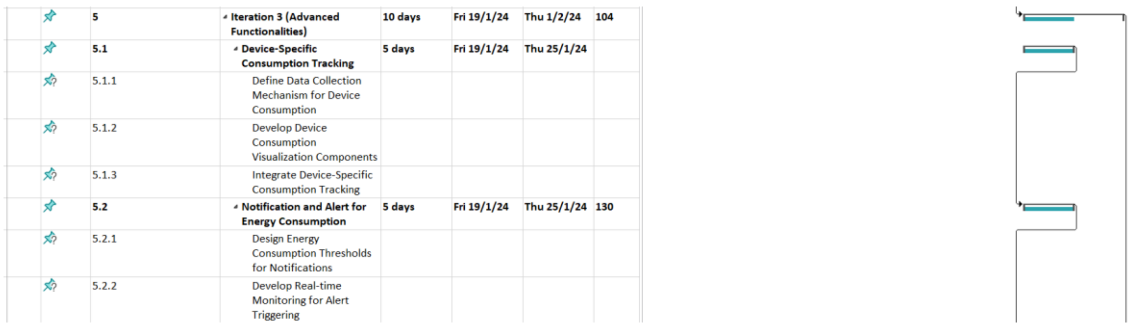
4.3	Complex Data Visualizations for Dashboard	4 days	Mon 8/1/24	Fri 12/1/24	107
4.3.1	Identify Data Elements for Complex Visualization				
4.3.2	Design Enhanced Data Visualization Components				
4.3.3	Develop Interactive Charts and Graphs				
4.3.4	Integrate Complex Data Visualizations into Dashboard				
4.4	Usability Testing and Interface Improvement	5 days	Fri 12/1/24	Fri 19/1/24	112
4.4.1	Plan Usability Testing Scenarios	1 day	Fri 12/1/24	Mon 15/1/24	
4.4.2	Conduct Usability Testing with Real Users	1 day	Tue 16/1/24	Tue 16/1/24	118
4.4.3	Analyze Usability Testing Results and Feedback	1 day	Wed 17/1/24	Wed 17/1/24	119
4.4.4	Identify UI/UX Improvement Areas	1 day	Thu 18/1/24	Thu 18/1/24	120
4.4.5	Implement Interface Enhancements based on Testing Results	1 day	Fri 19/1/24	Fri 19/1/24	121

Figure 3.4.12: Development Iteration 2 Timeline (Continued)



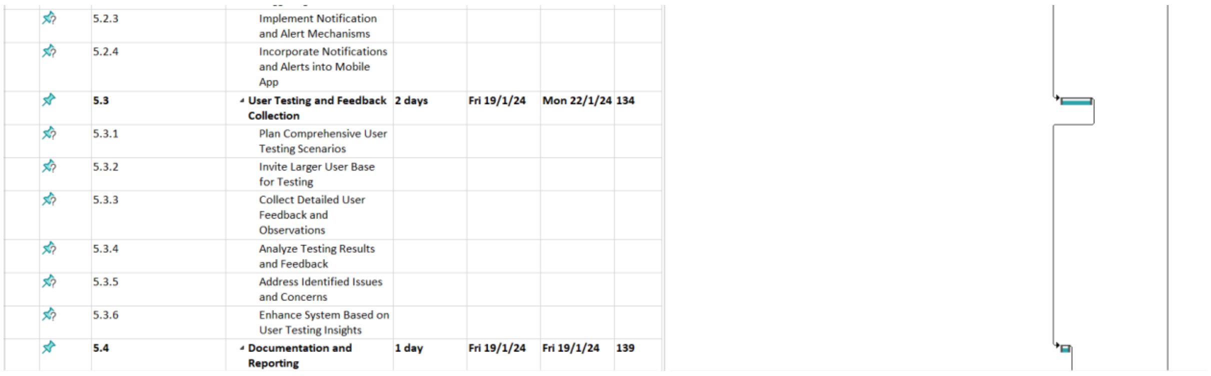
4.5	Report and Documentation	1 day	Fri 19/1/24	Mon 22/1/24	117
4.5.1	Document Enhanced Features and Functionalities	1 day	Mon 22/1/24	Mon 22/1/24	
4.6	Iteration Review and Feedback	1 day	Mon 22/1/24	Tue 23/1/24	123
4.6.1	Demonstrate Functionalities Enhancement to Stakeholders	1 day	Tue 23/1/24	Tue 23/1/24	
4.6.2	Collect Stakeholder Feedback and Suggestions	1 day	Tue 23/1/24	Tue 23/1/24	
4.6.3	Evaluate and Incorporate Feedback for Subsequent Phases	1 day	Tue 23/1/24	Tue 23/1/24	

Figure 3.4.13: Development Iteration 2 Timeline (Continued)



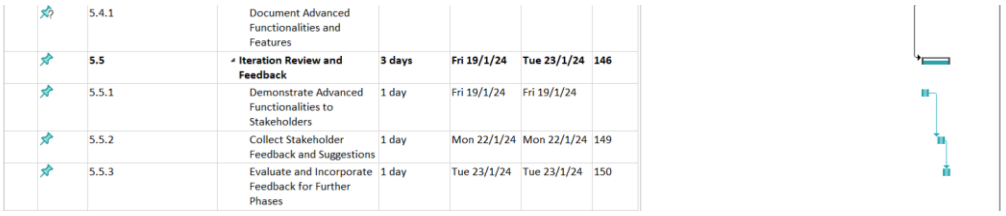
5	Iteration 3 (Advanced Functionalities)	10 days	Fri 19/1/24	Thu 1/2/24	104
5.1	Device-Specific Consumption Tracking	5 days	Fri 19/1/24	Thu 25/1/24	
5.1.1	Define Data Collection Mechanism for Device Consumption				
5.1.2	Develop Device Consumption Visualization Components				
5.1.3	Integrate Device-Specific Consumption Tracking				
5.2	Notification and Alert for Energy Consumption	5 days	Fri 19/1/24	Thu 25/1/24	130
5.2.1	Design Energy Consumption Thresholds for Notifications				
5.2.2	Develop Real-time Monitoring for Alert Triggering				

Figure 3.4.14: Development Iteration 3 Timeline



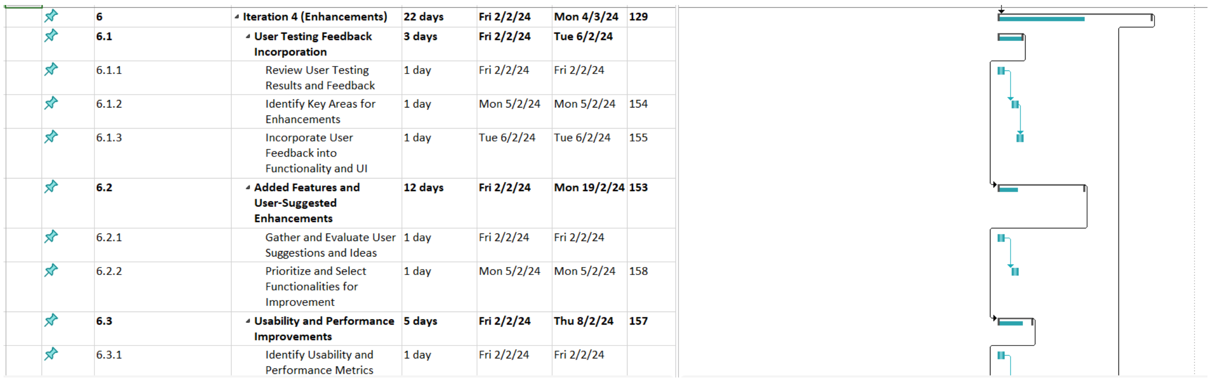
5.2.3	Implement Notification and Alert Mechanisms				
5.2.4	Incorporate Notifications and Alerts into Mobile App				
5.3	**User Testing and Feedback Collection**	**2 days**	**Fri 19/1/24**	**Mon 22/1/24**	**134**
5.3.1	Plan Comprehensive User Testing Scenarios				
5.3.2	Invite Larger User Base for Testing				
5.3.3	Collect Detailed User Feedback and Observations				
5.3.4	Analyze Testing Results and Feedback				
5.3.5	Address Identified Issues and Concerns				
5.3.6	Enhance System Based on User Testing Insights				
5.4	**Documentation and Reporting**	**1 day**	**Fri 19/1/24**	**Fri 19/1/24**	**139**

Figure 3.4.15: Development Iteration 3 Timeline (Continued)



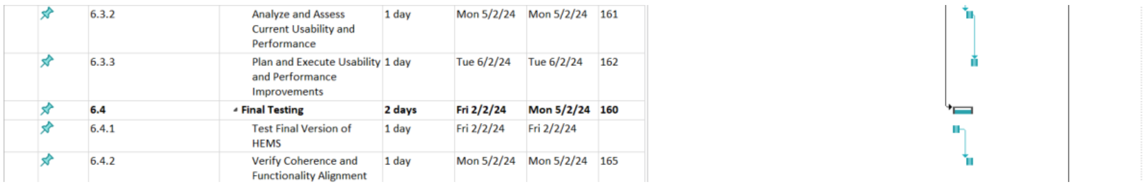
5.4.1	Document Advanced Functionalities and Features				
5.5	**Iteration Review and Feedback**	**3 days**	**Fri 19/1/24**	**Tue 23/1/24**	**146**
5.5.1	Demonstrate Advanced Functionalities to Stakeholders	1 day	Fri 19/1/24	Fri 19/1/24	
5.5.2	Collect Stakeholder Feedback and Suggestions	1 day	Mon 22/1/24	Mon 22/1/24	149
5.5.3	Evaluate and Incorporate Feedback for Further Phases	1 day	Tue 23/1/24	Tue 23/1/24	150

Figure 3.4.16: Development Iteration 3 Timeline (Continued)



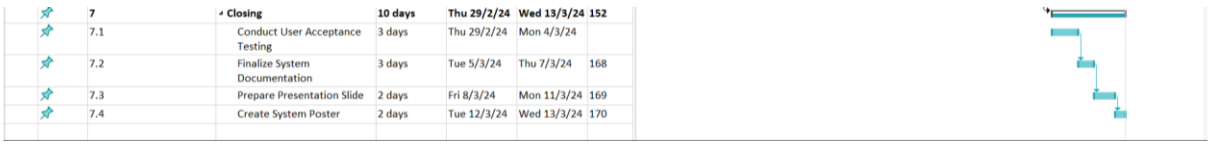
6	**Iteration 4 (Enhancements)**	**22 days**	**Fri 2/2/24**	**Mon 4/3/24**	**129**
6.1	**User Testing Feedback Incorporation**	**3 days**	**Fri 2/2/24**	**Tue 6/2/24**	
6.1.1	Review User Testing Results and Feedback	1 day	Fri 2/2/24	Fri 2/2/24	
6.1.2	Identify Key Areas for Enhancements	1 day	Mon 5/2/24	Mon 5/2/24	154
6.1.3	Incorporate User Feedback into Functionality and UI	1 day	Tue 6/2/24	Tue 6/2/24	155
6.2	**Added Features and User-Suggested Enhancements**	**12 days**	**Fri 2/2/24**	**Mon 19/2/24**	**153**
6.2.1	Gather and Evaluate User Suggestions and Ideas	1 day	Fri 2/2/24	Fri 2/2/24	
6.2.2	Prioritize and Select Functionalities for Improvement	1 day	Mon 5/2/24	Mon 5/2/24	158
6.3	**Usability and Performance Improvements**	**5 days**	**Fri 2/2/24**	**Thu 8/2/24**	**157**
6.3.1	Identify Usability and Performance Metrics	1 day	Fri 2/2/24	Fri 2/2/24	

Figure 3.4.17: Development Iteration 4 Timeline



6.3.2	Analyze and Assess Current Usability and Performance	1 day	Mon 5/2/24	Mon 5/2/24	161
6.3.3	Plan and Execute Usability and Performance Improvements	1 day	Tue 6/2/24	Tue 6/2/24	162
6.4	**Final Testing**	**2 days**	**Fri 2/2/24**	**Mon 5/2/24**	**160**
6.4.1	Test Final Version of HEMS	1 day	Fri 2/2/24	Fri 2/2/24	
6.4.2	Verify Coherence and Functionality Alignment	1 day	Mon 5/2/24	Mon 5/2/24	165

Figure 3.4.18: Development Iteration 4 Timeline (Continued)



7	**Closing**	**10 days**	**Thu 29/2/24**	**Wed 13/3/24**	**152**
7.1	Conduct User Acceptance Testing	3 days	Thu 29/2/24	Mon 4/3/24	
7.2	Finalize System Documentation	3 days	Tue 5/3/24	Thu 7/3/24	168
7.3	Prepare Presentation Slide	2 days	Fri 8/3/24	Mon 11/3/24	169
7.4	Create System Poster	2 days	Tue 12/3/24	Wed 13/3/24	170

Figure 3.4.19: Closing Phase Timeline

3.5 Development tools

3.5.1 Software

3.5.1.1 Visual Studio

Visual Studio is used in integrated development environment by supporting various types of development project such as mobile application and web application with different languages and framework. It is used to develop web application of Home Energy Monitoring System and APIs.

3.5.1.2 Android Studio

Android Studio provides integrated development environment for building Android application by providing tools to design, code, test and debug application. Android Studio will be used to emulate mobile application using Android Virtual Device Manager function provided to show the interface of the project in Android application.

3.5.1.3 Microsoft Project

Microsoft Project acts as a project management software for managing project flow. It is used to create Work Breakdown Structure by creating project schedule, defining milestones for each phase, tracking progress and managing timeline of the project.

3.5.1.4 Axure

Axure is a prototyping tool that help to create prototypes for software application. It is used to create prototypes for web dashboard and mobile application to visualize and validate the design with the supervisor and industry before actual development to ensure the outcome is satisfied.

3.5.1.5 Enterprise Architect

Enterprise Architect is used to create models for software system, business process, dataflow and others. It is used to create use cases, data flow diagrams, sequence diagrams and others that help in visualizing the Home Energy

Monitoring System's structure to enhance communication among stakeholders in this project.

3.5.1.6 Expo

Expo is a development tool and platform for building universal React applications that target both iOS and Android platforms. Its development environment simplifies the setup and configuration of React Native project to build applications quickly. In this project, Expo is used to build cross-platform mobile applications by scanning the QR code provided inside the terminal using iOS and Android devices after launching the project repository for checking different configurations of code on different platforms.

3.5.2 Programming Language and Markup Languages

3.5.2.1 HTML

HTML is a markup language for structuring web content using various elements like heading, lists, paragraphs, tables and more. It is used in this project to define the elements on web dashboard such as charts, graphs and form for user to interact and view their energy consumption data.

3.5.2.2 CSS

Cascading Style Sheets is a stylesheet language to design visual presentation of HTML elements by defining appearance of web pages. It is used to style the web-based dashboard and mobile application to ensure an interactive design for application.

3.5.2.3 JavaScript

JavaScript is a powerful programming language that may be used to add dynamic components, interactivity, and behaviour to web sites. Web applications become more responsive and engaging because it is carried out on the client side. It is selected to manage user inputs, run calculations, update real-time data, and produce animations.

3.5.2.4 Python

High-level programming language Python is renowned for its readability and adaptability. It is used for many kinds of development, including machine learning, data analysis, and web applications. Python can be utilised for a variety of backend project activities, including data processing, database communication, and RESTful API implementation. It is suitable for a variety of server-side activities due to its simplicity and vast libraries.

3.5.3 Framework

3.5.3.1 Angular

Angular is a popular open-source front-end framework that is used for building dynamic web application. It is used to create user interfaces for both mobile application and web-based dashboard by using UI components to display real-time energy consumption data and suer setting in user-friendly mode.

3.5.3.2 GitHub

GitHub is used to host and manage source code repositories, track changes over time and collaboration with other developers. In this case, a Home Energy Monitoring System repository will be created in GitHub to store its source code and documentation. Since this project is an industry link project, having a GitHub repository will enable others to keep track on the current work progress and provide guidance directly.

3.5.4 Service

3.5.4.1 Amazon DynamoDB

Amazon DynamoDB is a NoSQL database service offered by Amazon Web Services (AWS) to handle low-latency and high velocity workloads. It is used to handle real-time energy consumption data collected from energy monitoring devices and allow quick retrieval of information to mobile application and web application for real time energy monitoring.

3.5.4.2 AWS IoT Core

AWS IoT Core is a cloud services that connect IoT devices to other devices and AWS cloud services. The energy monitoring simulator will be connected to AWS IoT so that AWS IoT can connect this simulator to the other cloud services such as AWS DynamoDB and AWS Cognito.

3.5.4.3 AWS Lambda

AWS Lambda is a compute services that runs code in response to events such as DynamoDB stream events and automatically manages the compute resources. It is used to perform calculations on DynamoDB table data and integrate with different service such as AWS API Gateway and AWS AppSync in this project.

3.5.4.4 AWS API Gateway

AWS API Gateway is an AWS service for creating, publishing, maintaining and monitoring HTTP, REST and WebSocket APIs. This project will use this service to create HTTP API and WebSocket API for retrieving and transmitting data from DynamoDB to both mobile and web application.

3.5.4.5 AWS AppSync

AWS AppSync creates serverless GraphQL for mobile application development through a single point to perform retrieve, update, delete and create data on different storage such as AWS DynamoDB and AWS S3. It is used to perform certain part of DynamoDB table management in mobile application in this project for residents to perform CRUD functions directly using their mobile devices.

3.5.4.6 AWS Amplify

AWS Amplify is used to build cloud-enabled application which integrates with different AWS resources. It is installed inside this project mobile application to quickly set up backend services such as APIs, authentication, storage, and databases using AWS resources.

3.5.4.7 AWS Cognito

AWS Cognito provides user sign-up and authentication to mobile and web applications in simple way. It also enables the system to authenticate users through external identity such as email by sending verification code. It is used to integrate with AWS Amplify to streamlines the development process of mobile applications with the implementation of authentication workflows by providing tools and libraries to interact with AWS services in this project.

3.5.4.8 AWS SES (Simple Mail Service)

AWS SES is an email service developed by Amazon.com to send transactional emails, marketing messages and other customize email to users. It is used in this project to send notifications to residents when their energy consumption exceeds thresholds set.

3.5.4.9 AWS CloudWatch

AWS CloudWatch is a performance monitoring tools to monitor the execution of different AWS services. It is used to check the execution results of different AWS services resources and trigger lambda function to detect energy consumption threshold in specific period.

3.5.5 Design architecture style

3.5.5.1 HTTP API

HTTP APIs are used to send requests to AWS Lambda functions so that the AWS Lambda function can perform calculations based on the data retrieved from the DynamoDB table and return the function response to the mobile and web applications in this project.

3.5.5.2 WebSocket

WebSocket is a computer communication protocol used to broadcast the same data to multiple recipients simultaneously. It is selected for real-time data transmission from home energy monitoring devices to mobile applications, web applications and databases using WebSocket. WebSocket is useful for continuous connection between applications and the server to provide users with immediate and continuous data updates without frequent API requests.

3.6 System Architecture Workflow

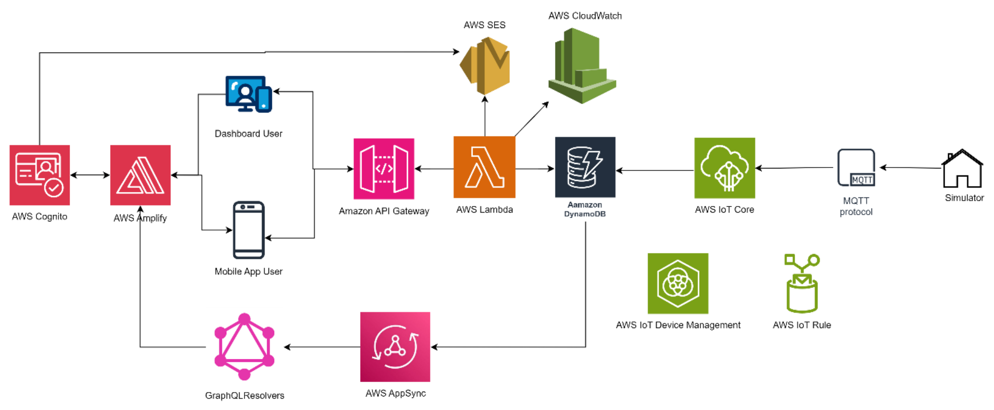


Figure 3.2: Overview of the Proposed Solution

The process of Home Energy Monitoring System is broken down into 5 steps. The implementation of this proposed solution is presented in Chapter 6 in detail with code and figure.

1. **Real-Time Energy Consumption Data Collection**

The home energy monitoring simulator collect real-time energy consumption data such as apparent power and device ID. This simulator will mimic real-world energy consumption patterns and behaviours accurately.

2. **Energy Consumption Data Transmission**

Home energy consumption data generated by the simulator is inserted into the AWS DynamoDB table via MQTT protocol and integrated with AWS IoT Core.

3. **Energy Consumption Data Retrieval**

At the same time, when the home energy consumption data are inserted into the AWS DynamoDB table, this data will be retrieved from the DynamoDB table via DynamoDB stream to WebSocket for real-time data streaming. The WebSocket will receive the data from the DynamoDB stream and send it to the mobile application.

4. **Historical Energy Consumption Data Calculation**

The historical energy consumption data on a daily, weekly and monthly basis are being retrieved from the DynamoDB table using AWS Lambda. The average energy consumption data is calculated and sent to the residents' mobile application using HTTP API.

5. Energy Consumption Data Threshold Detection

The energy consumption data in the DynamoDB table will be checked every 20 minutes based on the threshold set by the residents in their mobile application. The residents will be notified via email if their energy consumption data exceeds specified thresholds.

3.7 Summary

The selected system development methodology is iterative and incremental development methodology. All of the processes involved in each main stage were identified and explained through the utilization of a Gantt chart and a work breakdown structure. Moreover, the development tools ultilized in this project were being evaluated and the workflow of system architecture was under investigation.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter covers the analysis of online questionnaire conducted by Google Form. Moreover, requirement specification, use case diagrams, use case descriptions and interface flow diagrams are presented to enable an in-depth understanding of the functionality and user interactions of the project for both stakeholder communication and development purposes.

4.2 Findings

4.2.1 Questionnaire

A questionnaire was conducted with the residents in Malaysia, both those with and without home energy monitoring devices in their residential from 13 August 2023 until 13 September 2023. The questionnaire was distributed across peninsular Malaysia and the gathered responses were analysed and condensed. As of 29 August 2023, the report received 33 responses and the questionnaire will be conducted continuously until 13 September 2023. The complete questionnaire responses and result discussion are included in Appendix B.

4.2.1.1 Demographics

Most respondents fall within the age groups of 21-40 and 41-60 and are mainly those responsible for their economically stable electricity bills. The gender distribution among respondents is nearly equal, as this questionnaire was gender-neutral. Besides, most of the respondents are from Johor, Selangor, and Melaka, as this questionnaire targeted the respondents who live in the state with my TNB’s smart meter implementation. Next, most respondents have a monthly household income exceeding RM10000.

4.2.1.2 Energy Consumption Behavior

The respondents commonly live in townhouses, apartments, or single-family houses, which aligns with my TNB’s focus on these houses for smart meter installation. Moreover, their monthly electricity bills typically range from RM 101 to RM 250, with 97% of the respondents noticing a significant increase in electrical bills over the past year. Although many respondents are unaware of the reasons for high electricity bills, they thought that the factors contributing to high energy consumption include excessive appliance usage, standby mode usage, and peak-hour energy consumption.

4.2.1.3 Attitudes Toward HEMS

Many respondents have not previously used energy management or monitoring tools due to illation costs for smart meters and have a limited understanding of energy consumption patterns. Furthermore, the respondents who are willing to use HEMS are more focused on real-time energy monitoring and potential cost savings by using this system. They are concerned about the initial cost of implementing HEMS and fear increased electricity bills due to monitoring. Most prefer smartphone applications for remote monitoring and real-timeenergy monitoring for HEMS.

4.2.1.4 Conclusion of Questionnaire

Overall, the respondents favor HEMS, which offers real-time monitoring, cost-saving opportunities, and a user-friendly interface. They are concerned about the costs of installing the smart meter for electricity monitoring but are willing to try the system after knowing the benefits of HEMS. Focusing on the costs of installing smart meters and educating users about HEMS functionalities should be highlighted for widespread adoption of HEMS in the future.

4.2.2 Summary of Informal Interview and Site Visit

An informal interview session was held with Mr. Yong to gain insights into the project background and the expected functionalities of the Home Energy Monitoring System. Mr Yong serves as the Principal Researcher within the Advanced Intelligence Lab of MIMOS Berhad, and he represents the company as the project's designated representative. During our informative site visit to MIMOS, we had the opportunity to capture a memorable moment by taking a group picture to commemorate our visit. The figure is listed in Figure A-1, Appendix A.

The research, according to Mr. Yong, is a component of their bigger Home Energy Monitoring initiative. He clarified that their vast project has been broken up into several sections, each of which deals with a different element. Their ambitious project's main goal is to reduce greenhouse gas emissions, demonstrating their commitment to environmental sustainability and a more environmentally friendly future.

Additionally, Mr. Yong provided in-depth justifications for each of the features being supplied, which helped him clarify his expectations for the project and the resources that will be offered, as well as the anticipated time frame for project completion. The figure of attending the informal meeting is listed in Figure A-2, Appendix A.

Next, Mr. Yong proposed that the creation of a real-time energy monitoring capability should be the main goal of this project. The purpose of this function is to give detailed insights into energy consumption trends while highlighting how dynamic power usage patterns are. Users will have access to real-time data on their energy consumption, allowing them to make informed choices about their consumption patterns and spot possible areas for improvement.

A demonstration within the lab was planned during a site visit to display the energy monitoring device in action. The pictures of visiting the lab and demonstration are included in Figure A-3, Appendix A and Figure A-4, Appendix A. The device was used to offer real-time energy consumption information for several items, including a refrigerator and a kettle. The line graph used to visually portray this data allowed audience members to see how the amount of energy used changed over time.

The demonstration's emphasis on low-energy-consumption gadgets deserves special note. The graph's "other devices" category was used to aggregate represent all the devices with low energy consumption. This method streamlined the visualisation by emphasising the major energy consumers while still recognising the presence of smaller energy consumers.

4.3 Requirements Specification

4.3.1 Mobile Application

4.3.1.1 Functional Requirements

FR	Functional Requirements
FR001	The system shall allow user to register user account.
FR002	The system shall allow user to login user account.
FR003	The system shall allow user to monitor real-time energy by viewing his current energy consumption in kilowatt-hours (kWh) through chart visualizations.
FR004	The system shall allow user to view historical data and analysis of his own energy consumption for different time intervals such as daily, weekly, monthly, and yearly through graphs, charts and report.
FR005	The system shall allow user to receive real-time notification and alert when his energy consumption exceeds predefined thresholds.
FR006	The system shall provide users with comprehensive functionality to manage thresholds, including the ability to create, delete, and edit threshold settings for energy consumption according to their specific requirements and preferences.
FR007	The system shall allow user to manage own user profile to edit personal data.

4.3.1.2 Non-functional requirements

NFR	Non-Functional Requirements
NFR001	Performance requirements 1.1. The system shall respond to user input within 1 second. 1.2. The system shall display real-time energy consumption data within 1 second. 1.3. The system shall handle 98% of the exception without causing the project to crash.
NFR002	Compatibility requirements 2.1. The system shall run smoothly with Android 8.0 and above. 2.2. The system shall be able to run smoothly using phone and tablet devices with different display.
NFR003	Security requirements 3.1. The system shall authenticate user with username and password.
NFR004	Usability requirement 4.1. The system shall have a user-friendly interface.
NFR005	Availability requirement 5.1. The system shall be available 99% of time a week when the device is connected to Internet.

4.3.1.3 Use Case Diagram and Use Case Description for Mobile Application

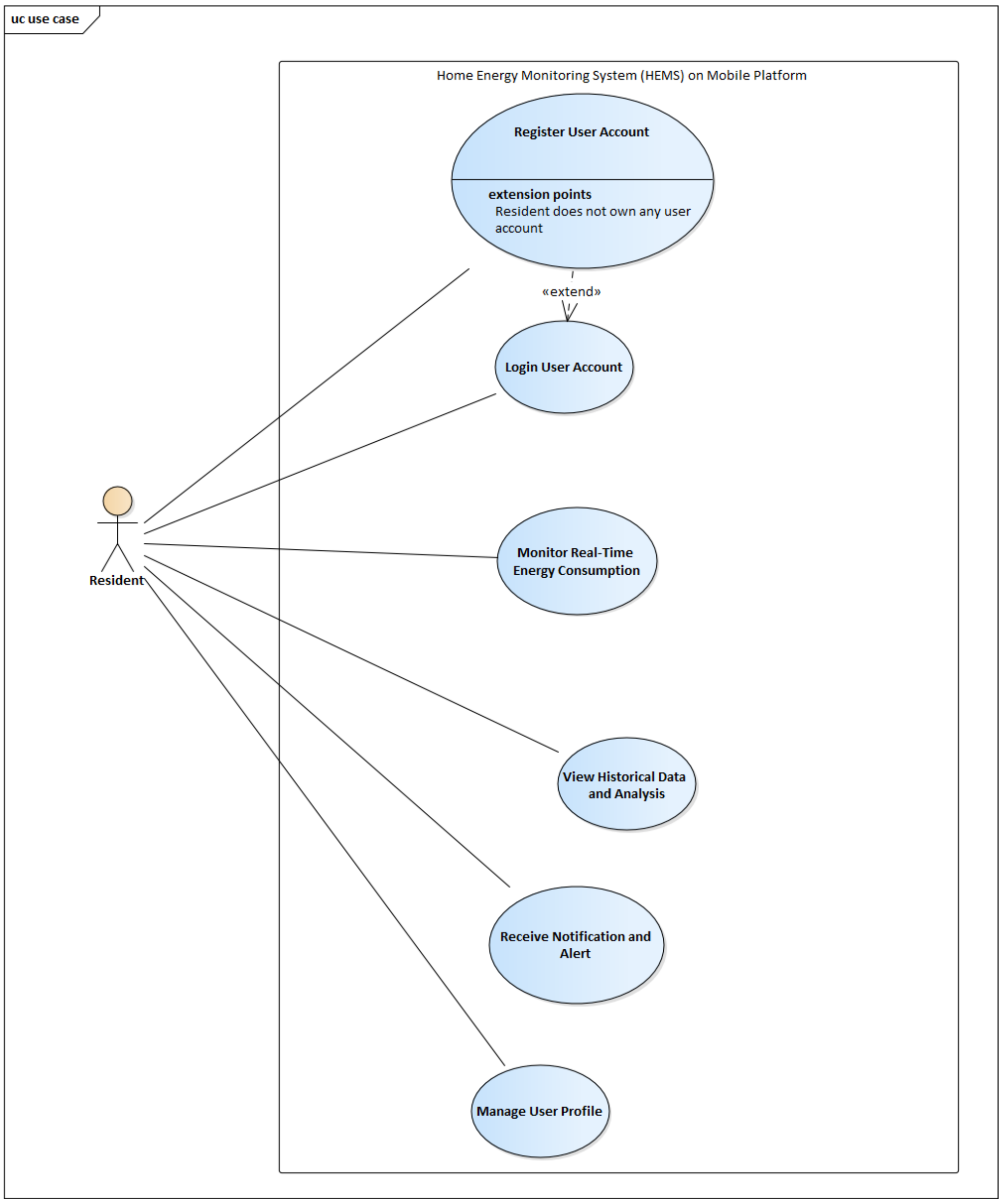

uc use case
Home Energy Monitoring System (HEMS) on Mobile Platform
Register User Account
extension points
Resident does not own any user account
«extend»
Login User Account
Monitor Real-Time Energy Consumption
View Historical Data and Analysis
Receive Notification and Alert
Manage User Profile
Resident

Figure 4.3.1: Mobile Application Use Case Diagram

Table 4.3.1: Register User Account

Name: Register User Account	ID: FR001	Priority: High
Actor: Resident	Type: Detail, Real	
Stakeholder's Information: Resident: User who wants to register user account through mobile application.		
Summary of Use Case: This use case describes the process of registering user account using mobile application by resident.		
Triggering Situation: The resident wants to register user account using mobile application.		
Relationship: - Association: Resident - Include: - - Extend: -		
Normal Event Flow: 1. The resident taps the register new user account button. 2. The resident fills in all required field in the registration form. 3. The resident registers new user account. 4. The system directs the residents to confirmation page. 5. The resident fill in the confirmation code to verify his email address. 6. The resident confirms his email account.		

7. The system redirects the resident back to the login page. 8. If resident want to login to his own account, perform sub-flow 5.1.
Sub Event Flow: 5.1 Resident enters both user email and password to login his account. 5.2 If the user email and password are correct, the system navigates to the real time monitoring home page.

Table 4.2: Monitor Real-Time Energy Consumption

Name: Monitor Real-Time Energy Consumption	ID: FR003	Priority: High
Actor: Resident	Type: Detail, Real	
Stakeholder's Information: Resident: User who wants to monitor energy consumption through mobile application.		
Summary of Use Case: This use case describes the process of monitoring real-time energy consumption using mobile application by resident.		
Triggering Situation: The resident wants to monitor energy consumption using mobile application.		
Relationship: - Association: Resident - Include: - - Extend: View Historical Data and Analysis		
Normal Event Flow: 1. The resident taps the monitor energy consumption navigation tab icon. 2. The system navigates resident to monitor real-time energy consumption page. 3. The system displays the default display of real-time energy consumption over 60 minutes time range.		

4. If the resident wants to choose a specific time range to view, perform View Historical Data and Analysis function (FR002). 5. The system redirects the resident back to the home page with default display of real-time energy consumption.
Sub Event Flow:

Table 4.3: View Historical Data and Analysis

Name: View Historical Data and Analysis	ID: FR004	Priority: High
Actor: Resident	Type: Detail, Real	
Stakeholder's Information: Resident: User who wants to view historical energy consumption data and analysis through mobile application.		
Summary of Use Case: This use case describes the process of viewing historical energy consumption data and analysis using mobile application by resident.		
Triggering Situation: The resident wants to view historical energy consumption data and analysis using mobile application.		
Relationship: - Association: Resident - Include: - - Extend: -		
Normal Event Flow: 1. The resident taps the view historical energy consumption data navigation tab icon. 2. The system displays the weekly energy consumption data. 3. If the resident wants to select monthly energy consumption data, perform sub-flow 3.1.		

4. The system redirects the resident back to the home page.
Sub Event Flow: 3.1 The resident clicks View Monthly button. 3.2 The system updates the energy consumption data display to show information relevant to the monthly energy consumption data and analysis.

Table 4.4: Receive Notification and Alert

Name: Receive Notification and Alert	ID: FR005	Priority: High
Actor: Resident	Type: Detail, Real	
Stakeholder's Information: Resident: User who wants to receive notification and alert about energy consumption through mobile application.		
Summary of Use Case: This use case describes the process of receive notification and alert using mobile application by resident.		
Triggering Situation: The resident wants to receive notification and alert regarding to his energy consumption using mobile application.		
Relationship: - Association: Resident - Include: - - Extend: -		
Normal Event Flow: 1. The resident taps the receive notification and alert navigation tab icon. 2. The system navigates resident to receive notification and alert page for setting up notifications and alerts. 3. The system displays all thresholds set by residents before. 5. If the resident wants to create new threshold, perform sub-flow 3.1.		

6. If the resident wants to delete threshold, perform sub-flow 3.3. 7. If the resident wants to update threshold, perform sub-flow 3.4.
Sub Event Flow: 3.1 The resident enters desired threshold value in kW and time interval for threshold detection. 3.2 The resident clicks Add Threshold button to create new threshold. 3.3 The resident clicks Delete button associated to specific threshold data to delete threshold. 3.4 The resident clicks Update button associated to specific threshold data to delete threshold. 3.5 The resident enters updated threshold value in kW and time interval for threshold detection. 3.6 The resident clicks Save button to update threshold.

4.3.2 Web Based Dashboard

4.3.2.1 Functional Requirements

FR	Functional Requirements
FR008	The system shall allow administrator to view different residents' dashboard with comprehensive energy analysis by providing detailed insights and visualizations encompassing consumption trends and peak usage times.
FR009	The system shall allow administrator to manage user accounts.
FR010	The system shall allow administrator to login administrator account.
FR011	The system shall allow administrator to register administrator account.

4.3.2.2 Non-functional requirements

NFR	Non-Functional Requirements
NFR005	Performance requirements 5.1. The system shall respond to user input within 0.1 second. 5.2. The system shall display real-time energy consumption data within 1 second. 5.3. The system shall handle 98% of the exception without project crashing.
NFR006	Compatibility requirements 6.1. The system shall run smoothly with Google Chrome.

NFR007	Security requirements 7.1. The system shall authenticate user with username and password.
NFR008	Usability requirement The system shall have a user-friendly interface.
NFR009	Availability requirement The system shall be available 99% of time throughout the week when the device is connected to the Internet.

4.3.2.3 Use Case Diagram and Use Case Description for Web Based Dashboard

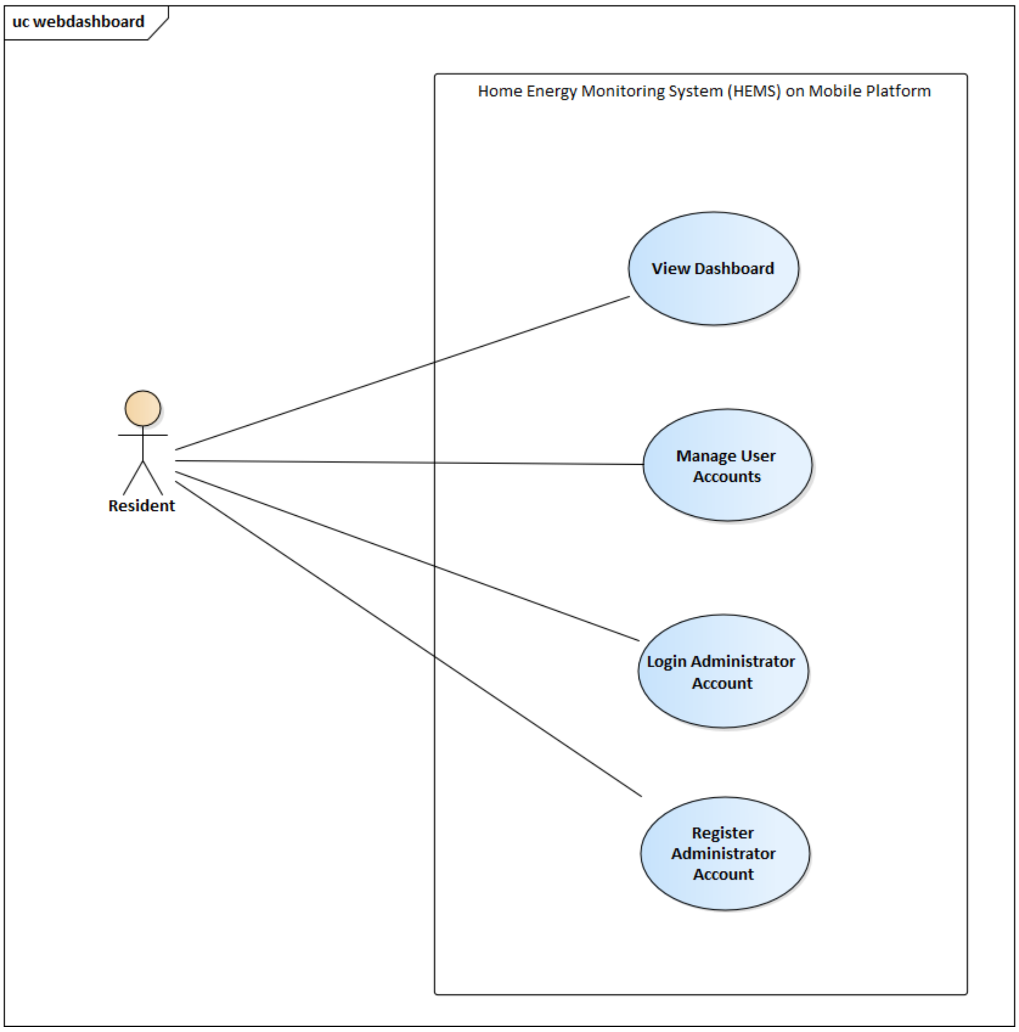

uc webdashboard
Home Energy Monitoring System (HEMS) on Mobile Platform
Resident
View Dashboard
Manage User Accounts
Login Administrator Account
Register Administrator Account

Figure 4.3.2: Web Dashboard Use Case Diagram

Table 4.3.2: View Dashboard

Name: View Dashboard	ID: FR008	Priority: High
Actor: Resident	Type: Detail, Real	
Stakeholder's Information: Resident: Administrator who wants to view specific resident's dashboard about energy consumption through web browser.		
Summary of Use Case: This use case describes the process of viewing dashboard using web browser by administrator.		
Triggering Situation: The administrator wants to view dashboard regarding to specific resident's energy consumption using web browser.		
Relationship: - Association: Administrator - Include: - - Extend: -		
Normal Event Flow: 1. The administrator logins his user account using user account number and password. 2. The system displays an overall summary of different residents' account information for administrator to perform view dashboard, update and delete operation. 3. The administrator selects specific resident dashboard to view.		

4. The system displays an overall summary of energy consumption for default selected time interval including key metrics such as total energy consumed and average energy consumption.
Sub Event Flow: -

4.3.3 Interface Flow Diagram

4.3.3.1 Mobile Application

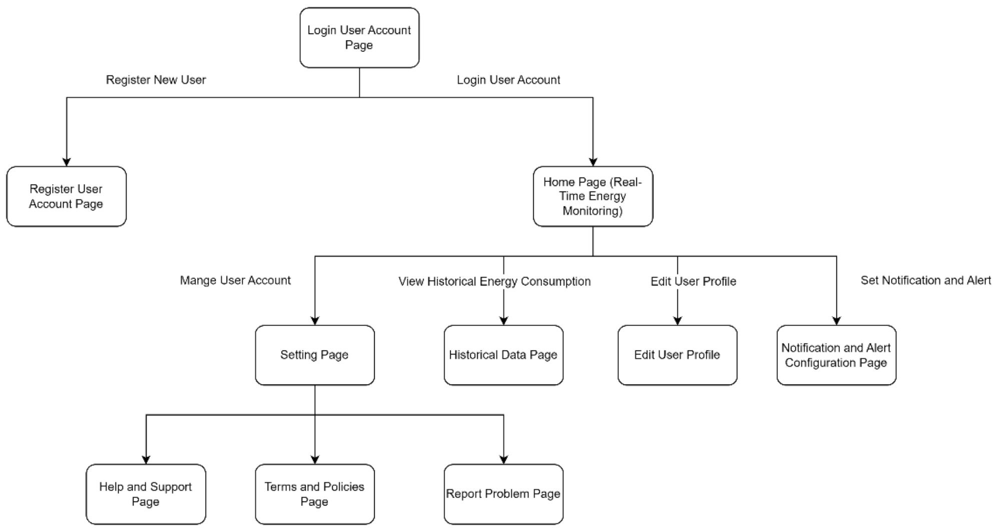


Figure 4.3.3: Mobile Application Interface Flow Diagram

4.3.3.2 Web-based Dashboard

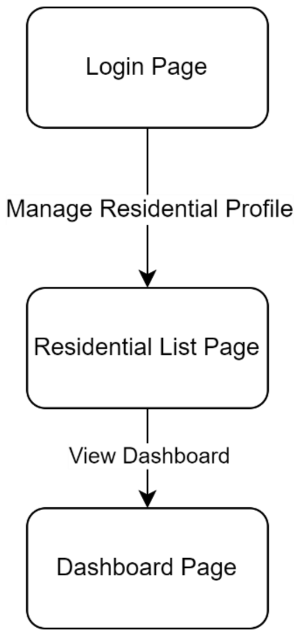


Figure 4.3.4: Web Dashboard Interface Flow Diagram

4.4 Initial Low Fidelity Prototypes

Both mobile application and web-based dashboard low fidelity prototypes are being provided to gain better understanding of the workflow of the system.

4.4.1 Mobile Application

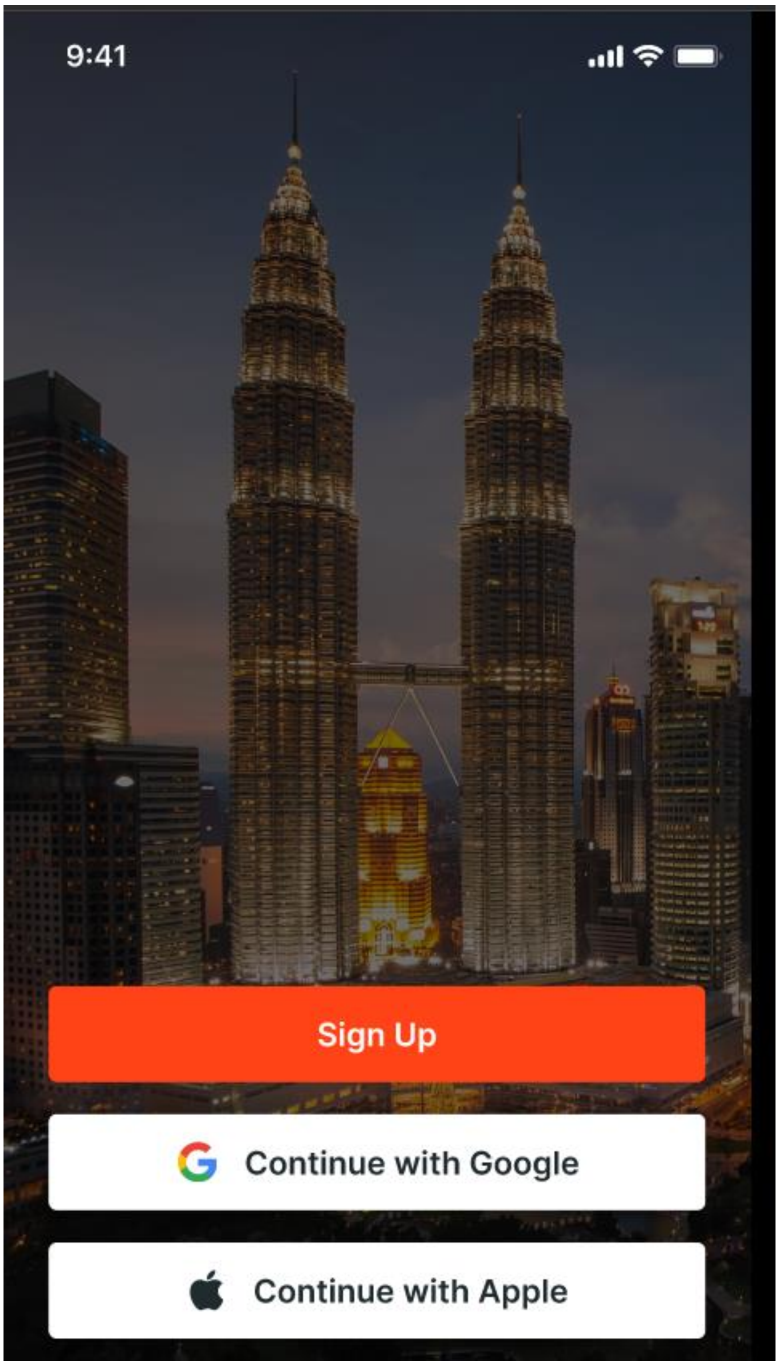


Figure 4.4.1: Low Fidelity Prototype- Register Page

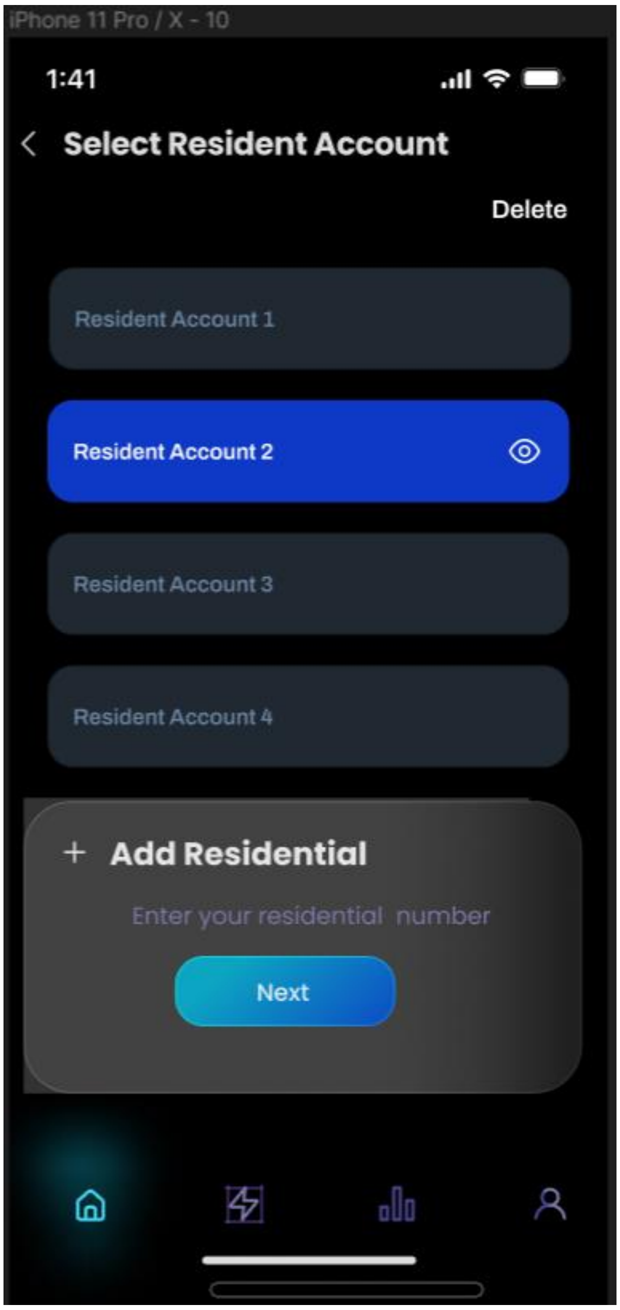


Figure 4.4.2: Low Fidelity Prototype- Manage User Account Page

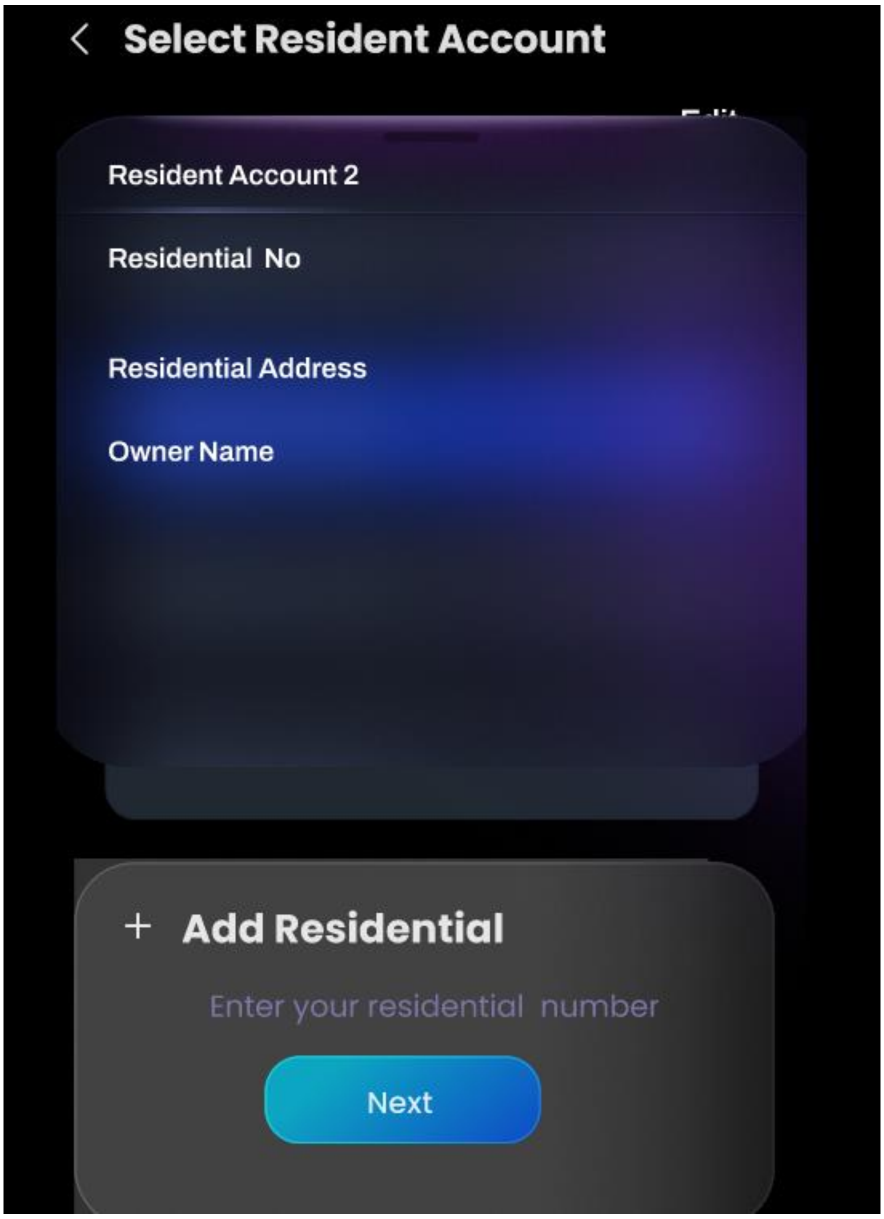


Figure 4.4.3: Low Fidelity Prototype- Manage User Account Page (Continued)

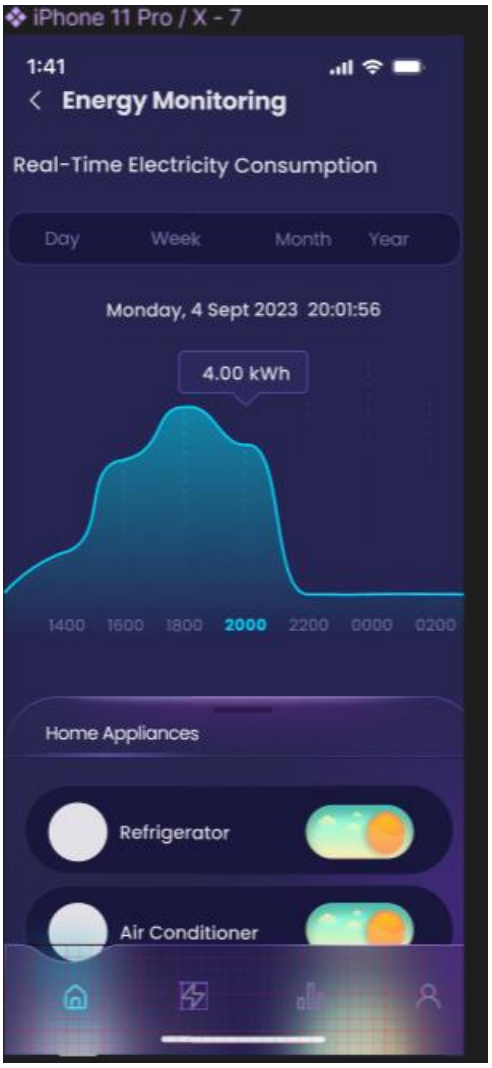


Figure 4.4.4: Low Fidelity Prototype- Energy Monitoring Page

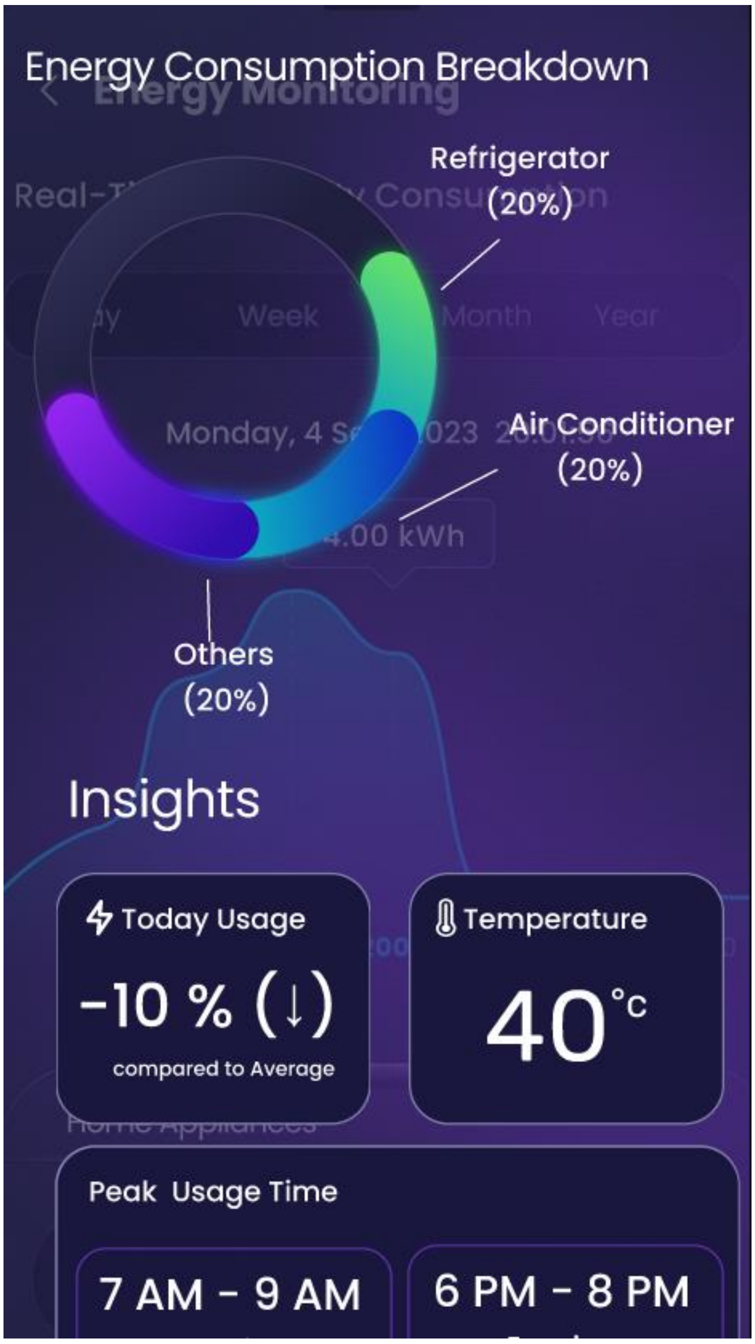


Figure 4.4.5: Low Fidelity Prototype- Energy Monitoring Page (Continued)

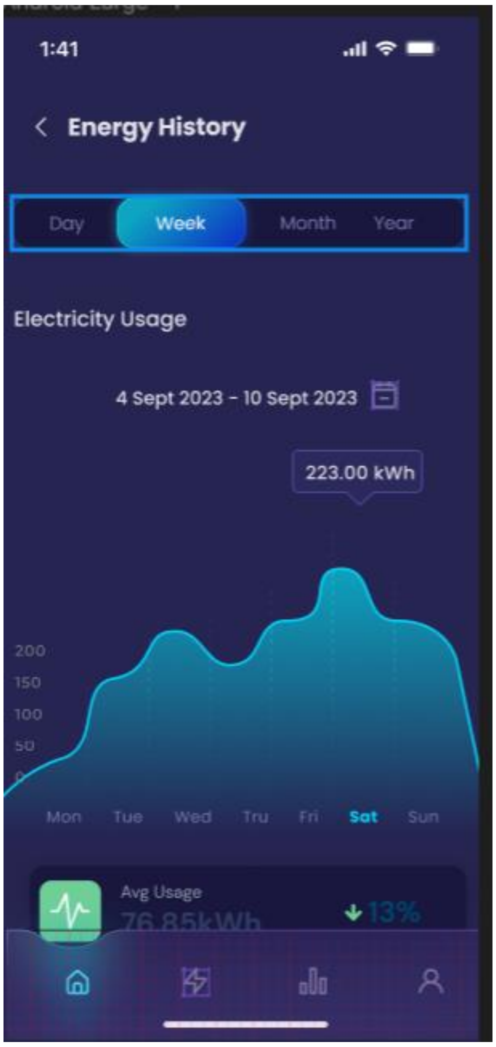


Figure 4.4.6: Low Fidelity Prototype- Energy History Page

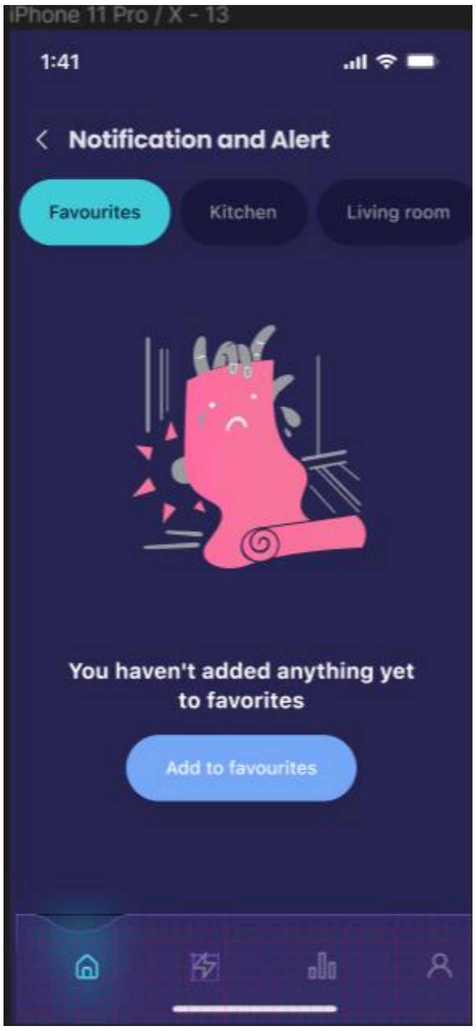


Figure 4.4.7: Low Fidelity Prototype- Notification and Alert Page

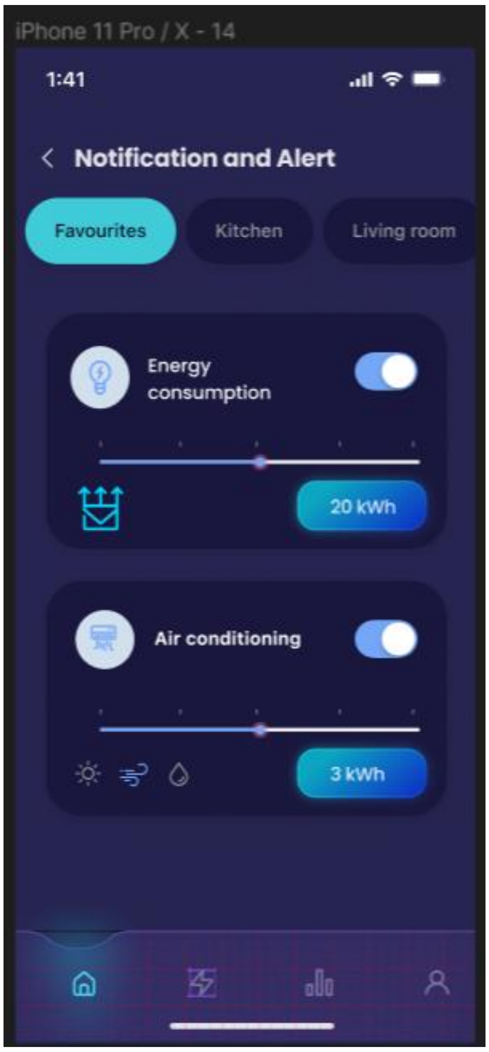


Figure 4.4.8: Low Fidelity Prototype- Notification and Alert Page (Continued)

4.4.2 Web Based Dashboard

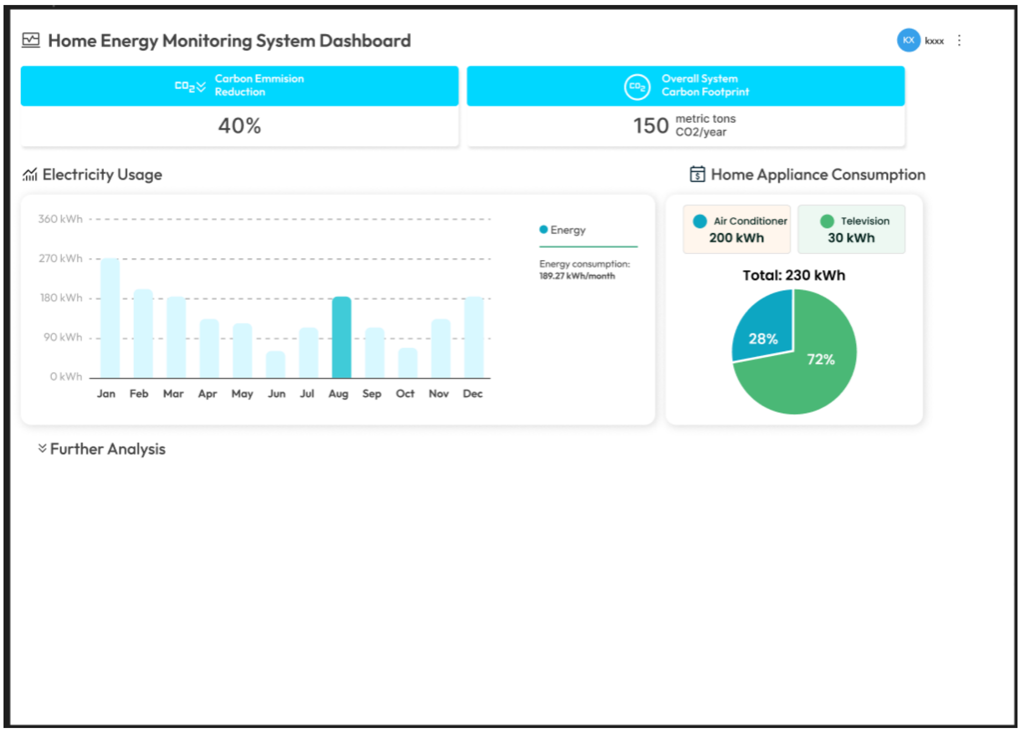

Home Energy Monitoring System Dashboard
KX kxxx
Carbon Emmision Reduction
40%
Overall System Carbon Footprint
150 metric tons CO2/year
Electricity Usage
360 kWh
270 kWh
180 kWh
90 kWh
0 kWh
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Energy
Energy consumption: 189.27 kWh/month
Home Appliance Consumption
Air Conditioner 200 kWh
Television 30 kWh
Total: 230 kWh
28%
72%
Further Analysis

Figure 4.4.9: Low Fidelity Prototype- Web Based Dashboard

4.5 Summary

In short, this chapter analysed the questionnaires results and the findings gained from informal interview. The data collected are used to support the problem statement of the project and the features implementation of the system. Next, the functional requirements and non-functional requirement for both mobile application and web-based dashboard are listed and evaluated through collected result. Finally, use case diagrams, use case descriptions, interface flow diagrams and initial low fidelity prototypes are included to describe the functionalities in the system.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter will discuss about the system architecture design, class diagram, entity relationship diagram, data flow design and user interface design.

5.2 System Architecture Design

<u>Introduction of Components Used</u>

Table 5.2.1: List of Components Used in System

	Component	General Description
1	Energy Monitoring Simulator	A simulator which simulates residential energy consumption data from different household appliances.
2	MQTT Protocol	A lightweight messaging protocol which connects IoT devices (Energy Monitoring Simulator) with AWS backend server.
3	AWS IoT Core	AWS IoT Core provides secure communication between energy monitoring simulator and AWS services via MQTT protocol. AWS IoT Device Management, AWS IoT Rule are being set to establish connection with energy monitoring simulator and insert data to AWS DynamoDB.
4	AWS DynamoDB	A fully managed NoSQL database service which used to store energy consumption data received

		from energy monitoring simulator, connection details with WebSocket and user, and threshold data set by residents.
5	AWS Lambda	A serverless computing service is being implemented to process messages from energy monitoring devices, perform data transformation, trigger notifications and alerts, and handle other backend tasks in response to events. This service will be used to integrate with various AWS backend services, including retrieving data from AWS DynamoDB for use in WebSocket API, performing calculations on different energy consumption data, setting thresholds, sending notifications, and performing other functions as required.
6	AWS API Gateway	A service for creating, publishing, maintaining and securing APIs. WebSocket API and HTTP API are being implemented to send and retrieve energy consumption data between AWS DynamoDB, web dashboard and mobile application.
7	AWS AppSync	A managed GraphQL service to create scalable GraphQL API to interact with backed data sources such as AWS DynamoDB Table that enables real-time data synchronization for mobile and web application. GraphQL resolvers that are functions to resolve the fields in a GraphQL query is used with AppSync to fetch data from AWS DynamoDB table and perform CRUD functions from mobile application.

8	AWS Amplify	A set of tools and services to build cloud-power application. It is being implemented inside mobile application for this project to create threshold DynamoDB table, perform CRUD operation, and manage API resources.
9	AWS Cognito	A fully managed identity service being used to handle user authentication, authorization and user management for mobile application.
10	AWS SES	Simple Email Service that sends email notification is being used to send notification to specific user according to different thresholds set and user sign up experiences which required user to verify email address after register account.
11	AWS CloudWatch	A service to watch the result of each service(log) and trigger Lambda function execution at certain period such as setting checking threshold function being executed each 20 minutes.

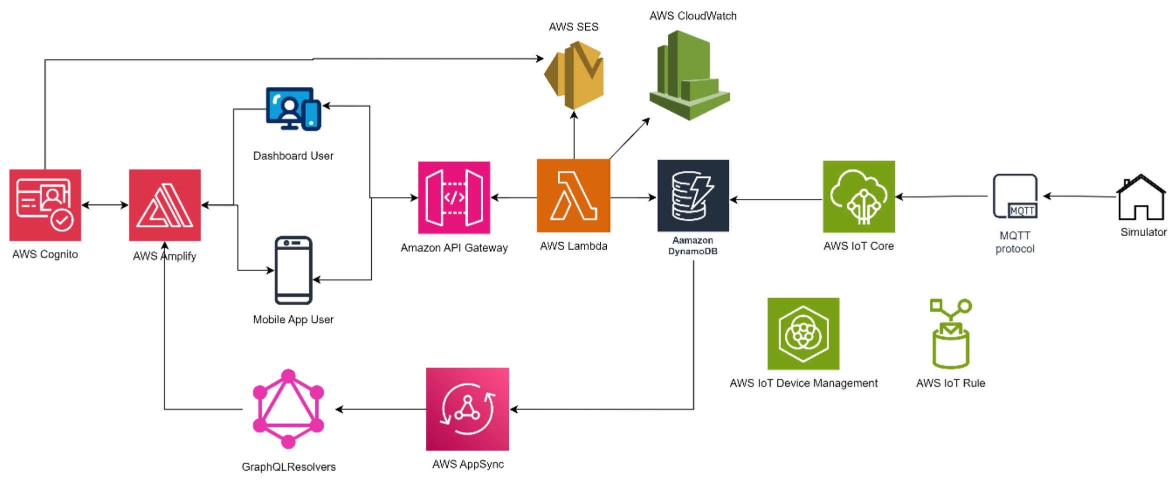


Figure 5.2.1: System Architecture Overview

The figure above illustrates the high-level architecture of the web-based dashboard and mobile application, delineating two primary layers: the frontend layer and the backend layer. The backend layer orchestrates the integration between the mobile application and the web-based dashboard backend, interfacing with AWS-backed services. As the database is configured within AWS infrastructure, the backend layer primarily concentrates on the processing of business logic, facilitating data transactions, and managing the storage and retrieval of data from the database. It serves as the intermediary for data exchange between the frontend and the AWS-backed services.

The assumption being made is that the energy monitoring simulator will be invoked continuously. As depicted in the figure and table, the energy monitoring simulator will invoke the passing of data via the MQTT protocol to AWS IoT Core. AWS IoT Core will receive this data and insert it into DynamoDB. A DynamoDB stream is configured to detect new entries (energy consumption data) inserted into the DynamoDB table. Lambda functions, which interact with the DynamoDB stream, will execute functions that send the incoming data in real time to the specified WebSocket API connected with mobile applications. Subsequently, residents using the mobile application will receive the energy consumption data and display it on their mobile screens. Mobile application residents will register their own accounts via AWS Cognito from AWS Amplify and perform a sign-in operation to log in to their accounts to view specific energy consumption data. AWS SES is configured to send email verification to users once they register a new account. Additionally, residents can set their own thresholds, which will detect specific energy consumption data at certain intervals. A Lambda function is set to continuously check the energy consumption data using AWS CloudWatch, triggering a function within a certain period and sending emails to specific users when thresholds are exceeded. For the web-based dashboards, HTTP APIs will be utilized to request the necessary data from DynamoDB tables through Lambda functions. Additionally, data retrieval from AWS Cognito and execution of CRUD (Create, Read, Update, Delete) operations will be performed. Further details regarding this backend implementation will be elaborated upon in Chapter 6.

The front end of this home energy monitoring application is developed using Expo, React Native, JavaScript, and AWS frontend components for the mobile application. Expo facilitates the creation of cross-platform applications for iOS and Android platforms using a unified codebase, while React Native leverages JavaScript and React to build native mobile apps. For the web application, the front end utilizes React, Laravel, HTML, CSS, and JavaScript, alongside Python. React enables the creation of interactive user interfaces, while Laravel, following the MVC architecture pattern, manages various databases based on specifications. AWS frontend components play a crucial role in managing API resources, user authentication, and authorization, integrating seamlessly with other AWS services to ensure secure and efficient communication with the backend.

5.3 Analysis Class Diagram

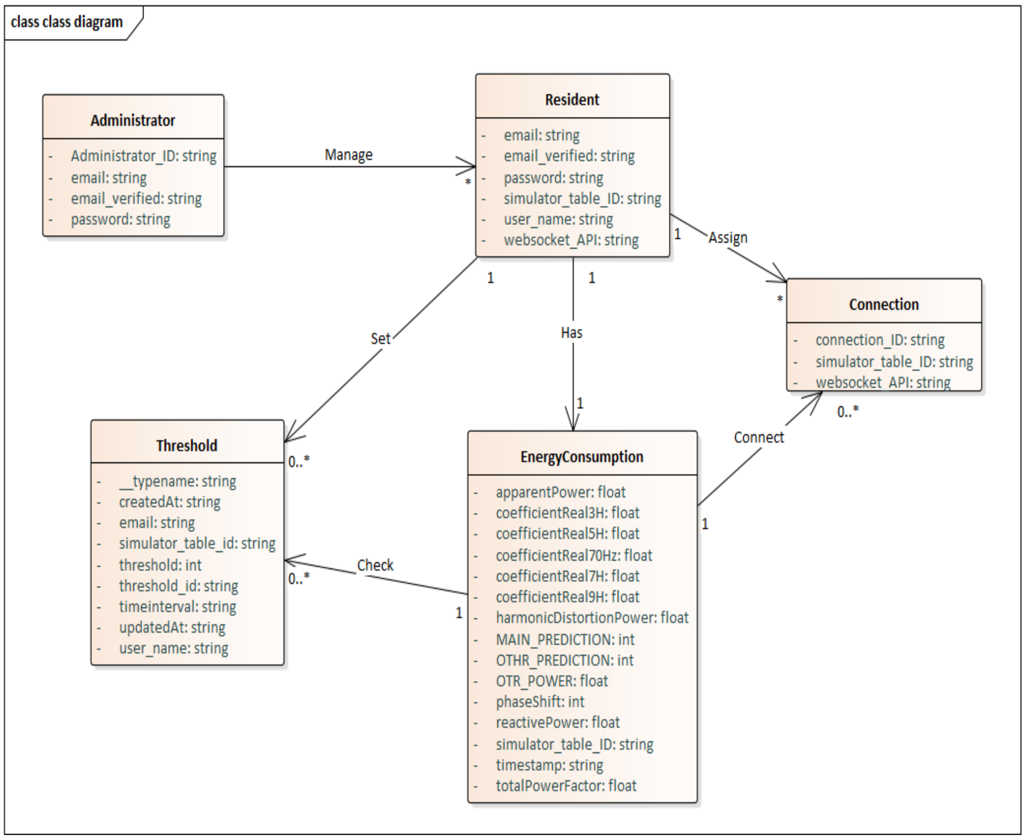


Figure 5.3.1: Analysis Class Diagram

5.4 Entity Relationship Diagram

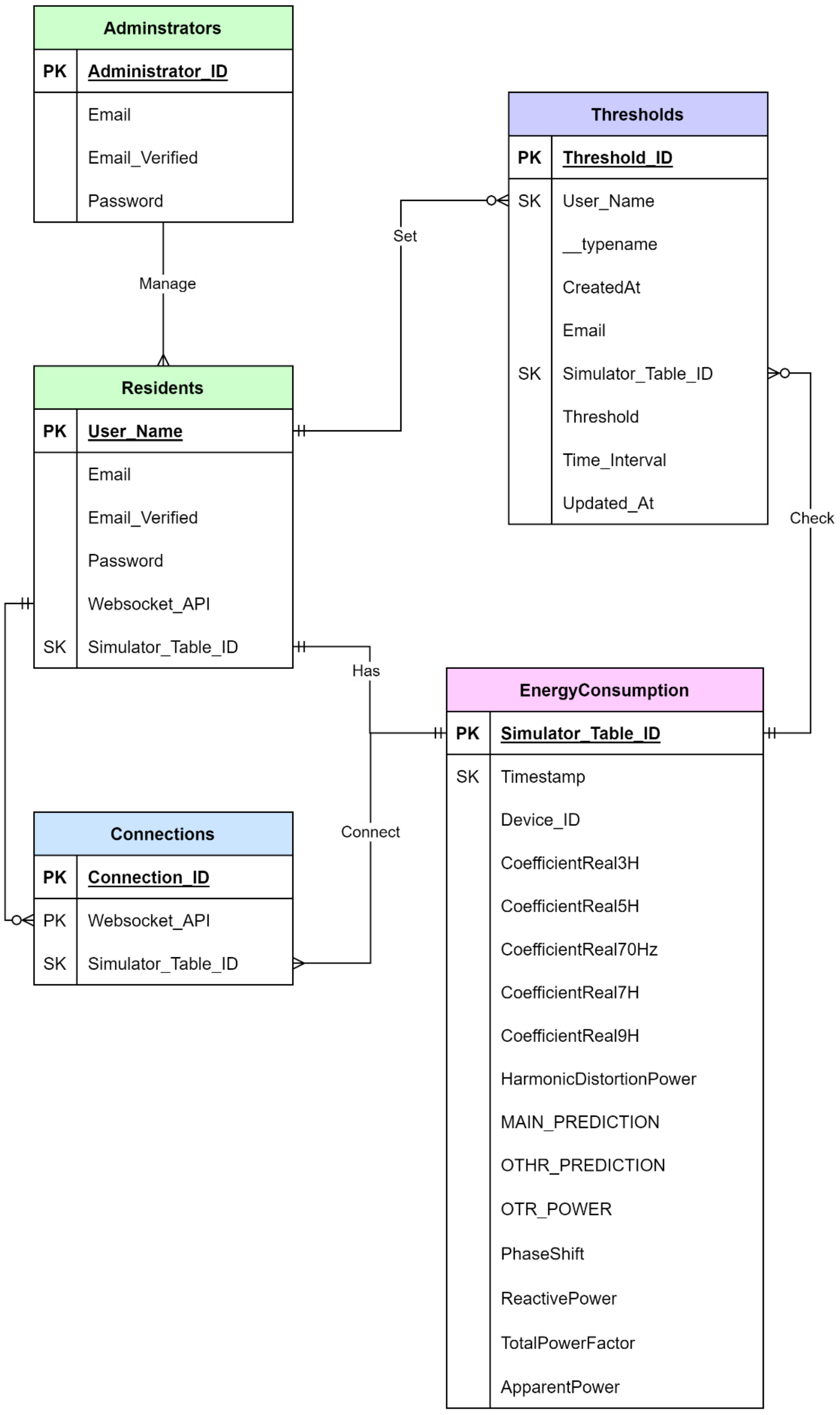


Figure 5.4.1: Entity Relationship Diagram

5.5 Table Description

Table 5.5.1: Description of Entity Relationship Table

Table Name	Description
Residents	Contains resident profile details
EnergyConsumption	Contains energy consumption details for specific resident. Each resident will have own energyConsumption table
Connections	Contains connection details for residents' mobile application and WebSocket API to send energy consumption data to correct resident
Thresholds	Contains all thresholds set by different residents
Administrators	Contains administrator account details for web application

5.6 Data Flow Diagram

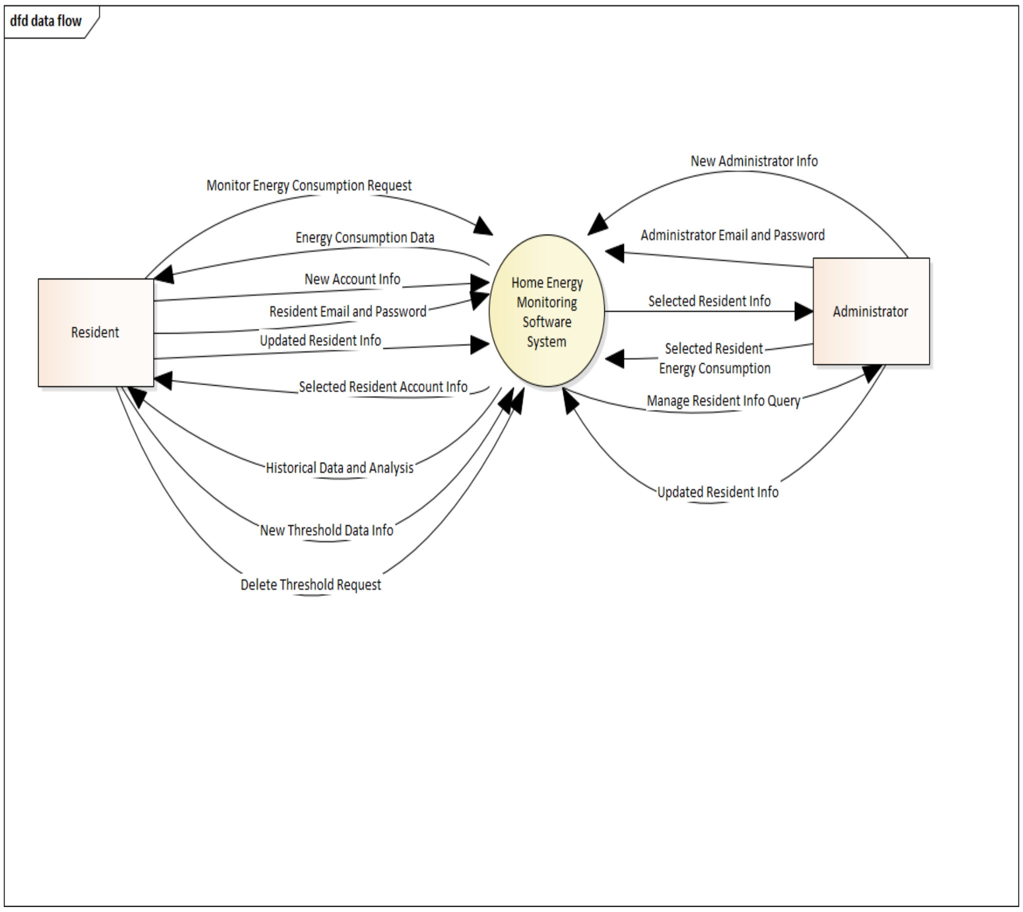

dfd data flow
Resident
Home Energy Monitoring Software System
Administrator
Monitor Energy Consumption Request
Energy Consumption Data
New Account Info
Resident Email and Password
Updated Resident Info
Selected Resident Account Info
Historical Data and Analysis
New Threshold Data Info
Delete Threshold Request
New Administrator Info
Administrator Email and Password
Selected Resident Info
Selected Resident Energy Consumption
Manage Resident Info Query
Updated Resident Info

Figure 5.6.1: Data Flow Diagram Context Diagram

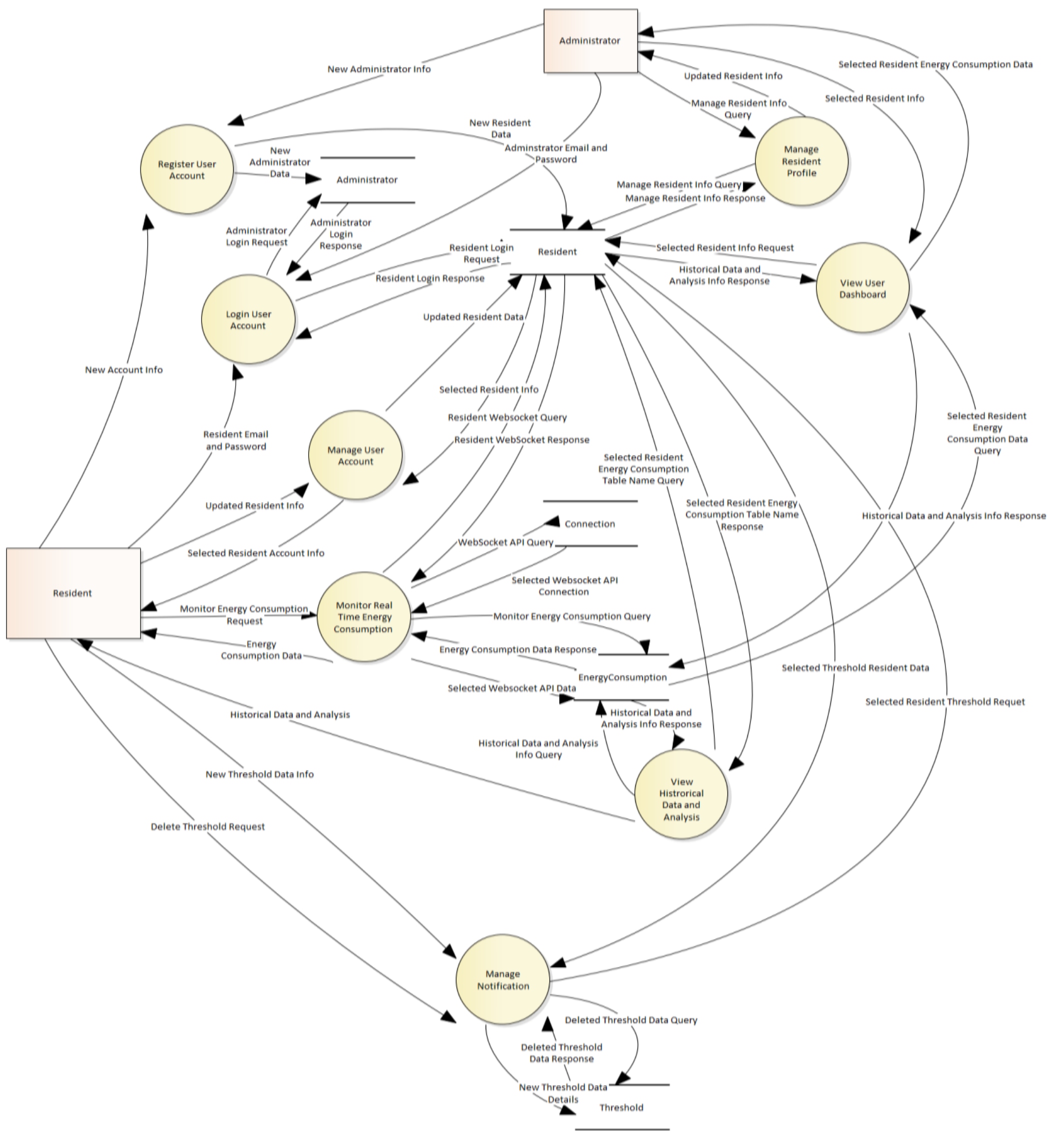

Administrator
New Administrator Info
Updated Resident Info
Selected Resident Energy Consumption Data
Manage Resident Info Query
Selected Resident Info
Register User Account
New Resident Data
New Administrator Data
Adminstrator Email and Password
Administrator
Manage Resident Profile
Manage Resident Info Query
Manage Resident Info Response
Administrator Login Request
Administrator Login Response
Resident Login Request
Resident
Selected Resident Info Request
Historical Data and Analysis Info Response
View User Dashboard
Resident Login Response
Login User Account
Updated Resident Data
New Account Info
Selected Resident Info
Resident Websocket Query
Resident WebSocket Response
Resident Email and Password
Manage User Account
Selected Resident Energy Consumption Table Name Query
Selected Resident Energy Consumption Data Query
Updated Resident Info
Selected Resident Energy Consumption Table Name Response
Historical Data and Analysis Info Response
Connection
WebSocket API Query
Selected Resident Account Info
Resident
Selected Websocket API Connection
Monitor Energy Consumption Request
Monitor Real Time Energy Consumption
Monitor Energy Consumption Query
Energy Consumption Data
Energy Consumption Data Response
EnergyConsumption
Selected Threshold Resident Data
Selected Websocket API Data
Selected Resident Threshold Requet
Historical Data and Analysis
Historical Data and Analysis Info Response
Historical Data and Analysis Info Query
View Historical Data and Analysis
New Threshold Data Info
Delete Threshold Request
Manage Notification
Deleted Threshold Data Query
Deleted Threshold Data Response
New Threshold Data Details
Threshold

Figure 5.6.2: Data Flow Diagram Level 0

5.7 User Interface Design

This section reported the actual user interface design for the web dashboard and mobile applications in this project.

5.7.1 Web Application

The web dashboard is designed for the administrator to manage the residents' account information and view specific resident dashboards. This is because the administrator is responsible for assigning different WebSocket API and energy consumption DynamoDB table setups for different residents, which residents should not perform.

5.7.1.1 Administrator Register Page

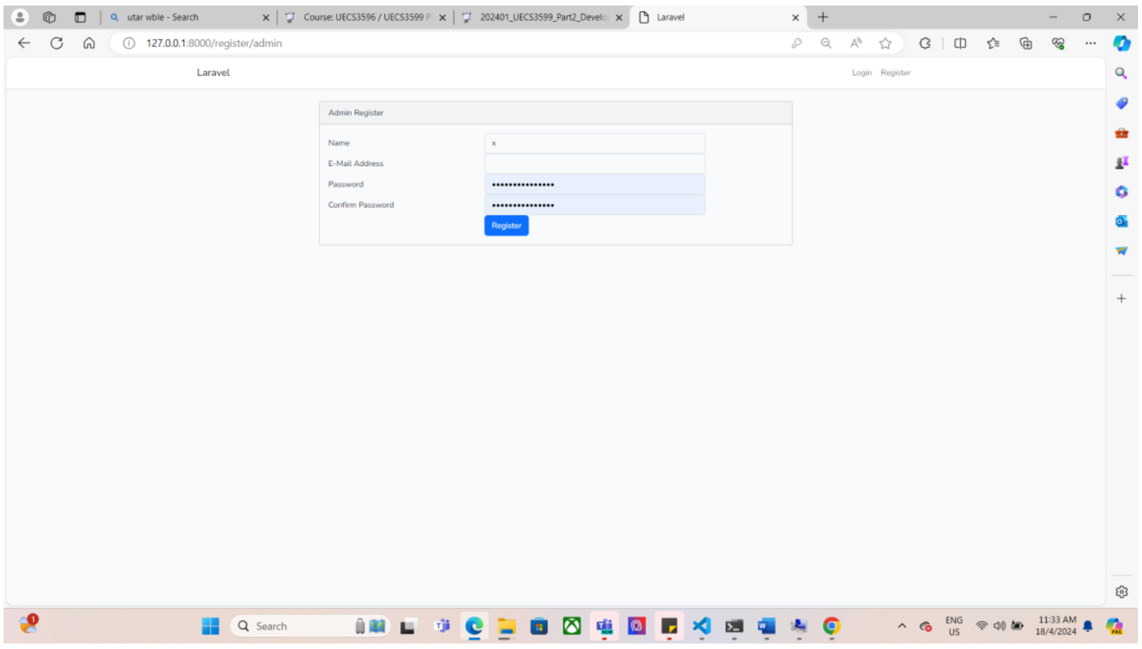
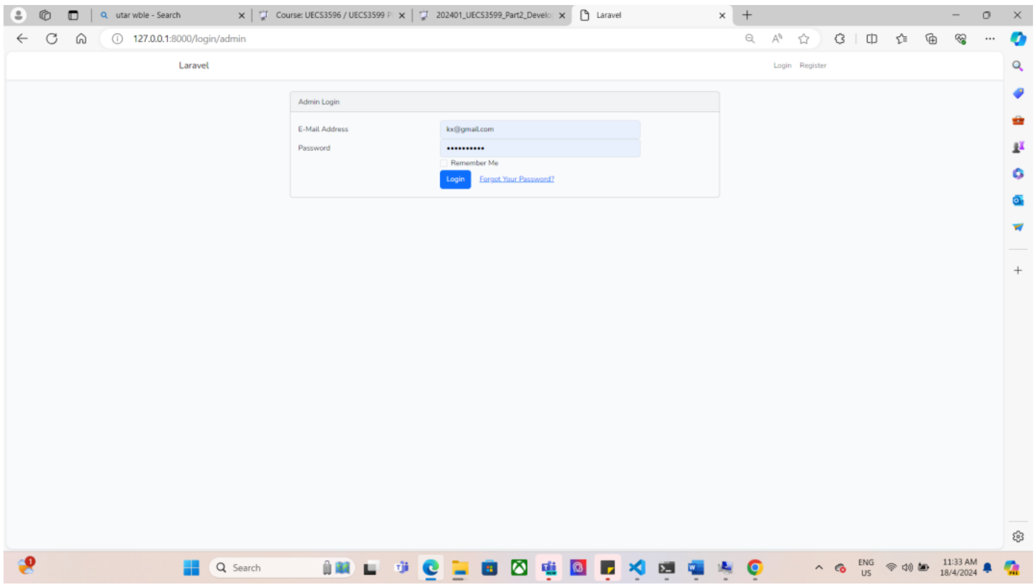


Figure 5.7.1: Administrator Register Screen

5.7.1.2 Administrator Login Page


Admin Login

E-Mail Address: kx@gmail.com

Password: ••••••••••

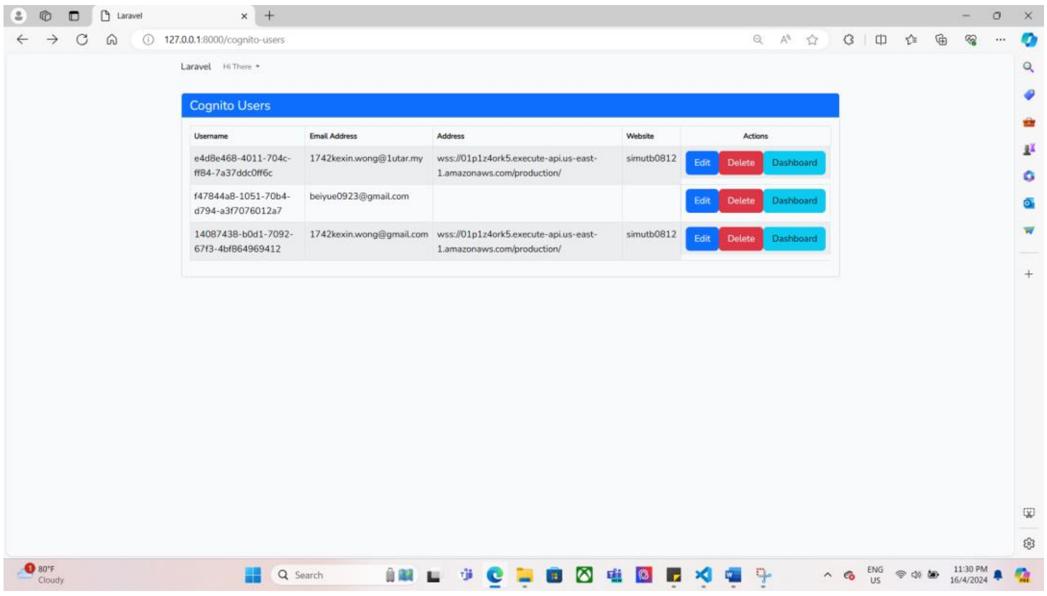
Remember Me

Login

Forgot Your Password?

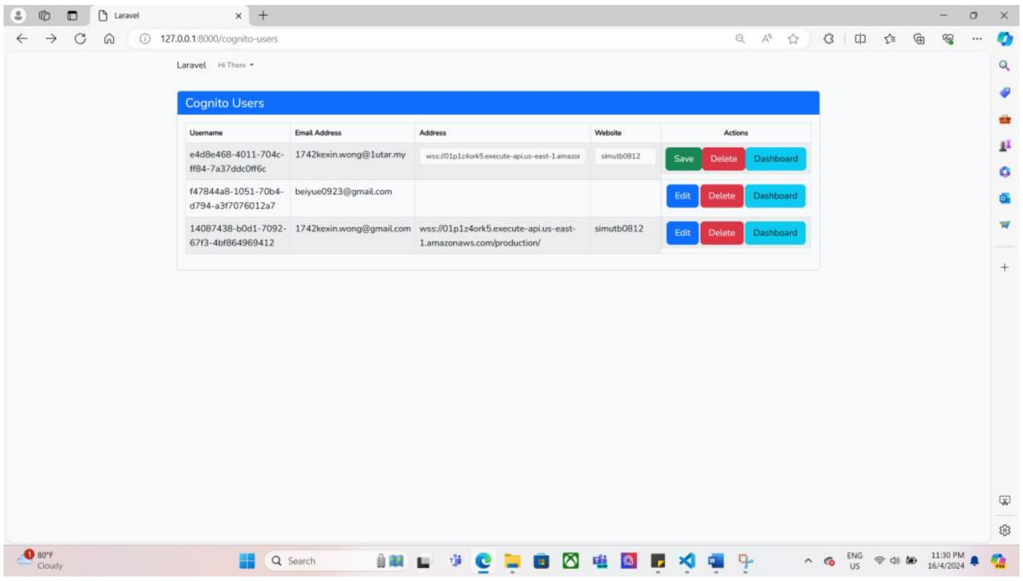
Figure 5.7.2: Administrator Login Screen

5.7.1.3 Manage User Page


Cognito Users

Username	Email Address	Address	Website	Actions
e4d8e468-4011-704c-ff84-7a37ddc0ff6c	1742kexin.wong@1utar.my	wss://01p1z4ork5.execute-api.us-east-1.amazonaws.com/production/	simutb0812	Edit Delete Dashboard
f47844a8-1051-70b4-d794-a3f7076012a7	beiyue0923@gmail.com			Edit Delete Dashboard
14087438-b0d1-7092-67f3-4bf864969412	1742kexin.wong@gmail.com	wss://01p1z4ork5.execute-api.us-east-1.amazonaws.com/production/	simutb0812	Edit Delete Dashboard

Figure 5.7.3: Manage User Screen


127.0.0.1:8000/cognito-users

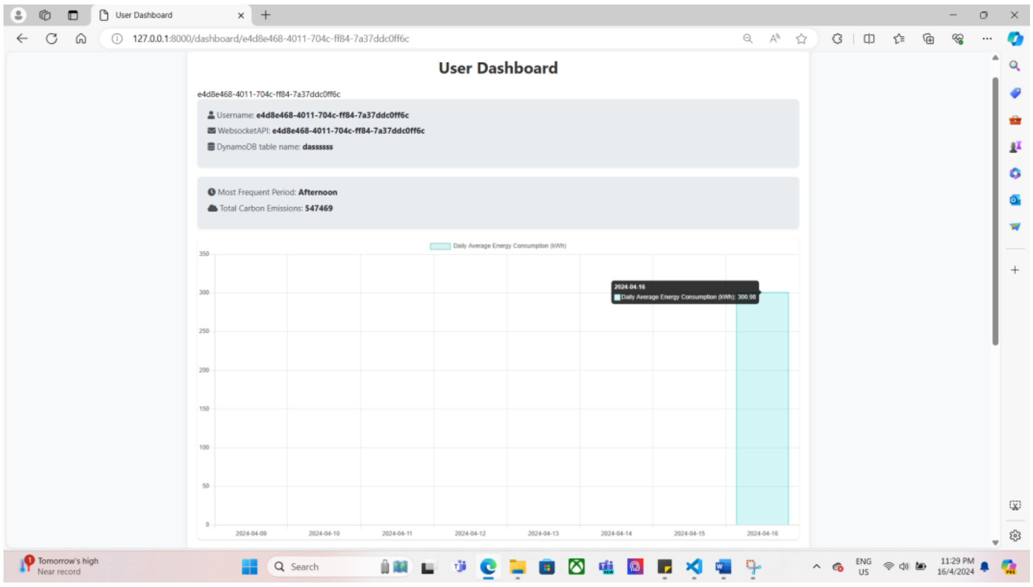
Laravel   Hi There

Cognito Users

Username	Email Address	Address	Website	Actions
e4d8e468-4011-704c-ff84-7a37ddc0ff6c	1742kexin.wong@1utar.my	wss://01p1z4ork5.execute-api.us-east-1.amazor	simutb0812	Save Delete Dashboard
f47844a8-1051-70b4-d794-a3f7076012a7	beiyue0923@gmail.com			Edit Delete Dashboard
14087438-b0d1-7092-67f3-4bf864969412	1742kexin.wong@gmail.com	wss://01p1z4ork5.execute-api.us-east-1.amazonaws.com/production/	simutb0812	Edit Delete Dashboard

Figure 5.7.4: Edit User Screen

5.7.1.4 User Dashboard


127.0.0.1:8000/dashboard/e4d8e468-4011-704c-ff84-7a37ddc0ff6c

User Dashboard

e4d8e468-4011-704c-ff84-7a37ddc0ff6c

Username: **e4d8e468-4011-704c-ff84-7a37ddc0ff6c**
WebsocketAPI: **e4d8e468-4011-704c-ff84-7a37ddc0ff6c**
DynamoDB table name: **dassssss**

Most Frequent Period: **Afternoon**
Total Carbon Emissions: **547469**

Figure 5.7.5: User Dashboard 1

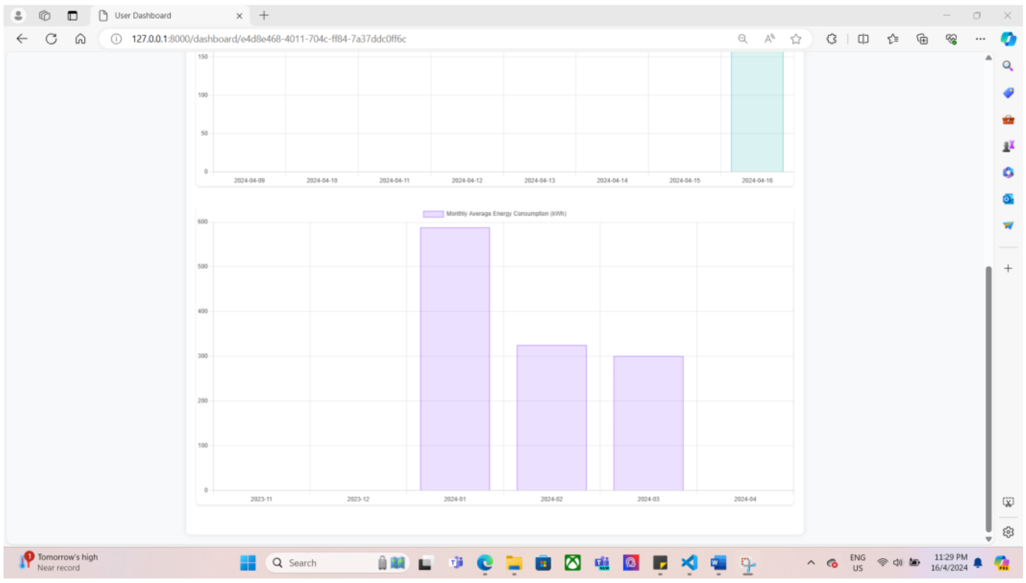


Figure 5.7.6: User Dashboard 2

5.7.2 Mobile Application

This mobile application designed for residents to monitor their energy consumption includes welcome page, login page, create account page, home page, historical energy consumption page, user profile page, energy monitoring page, setting page, term and policy page, help and support page and report problem page.

5.7.2.1 Welcome Page

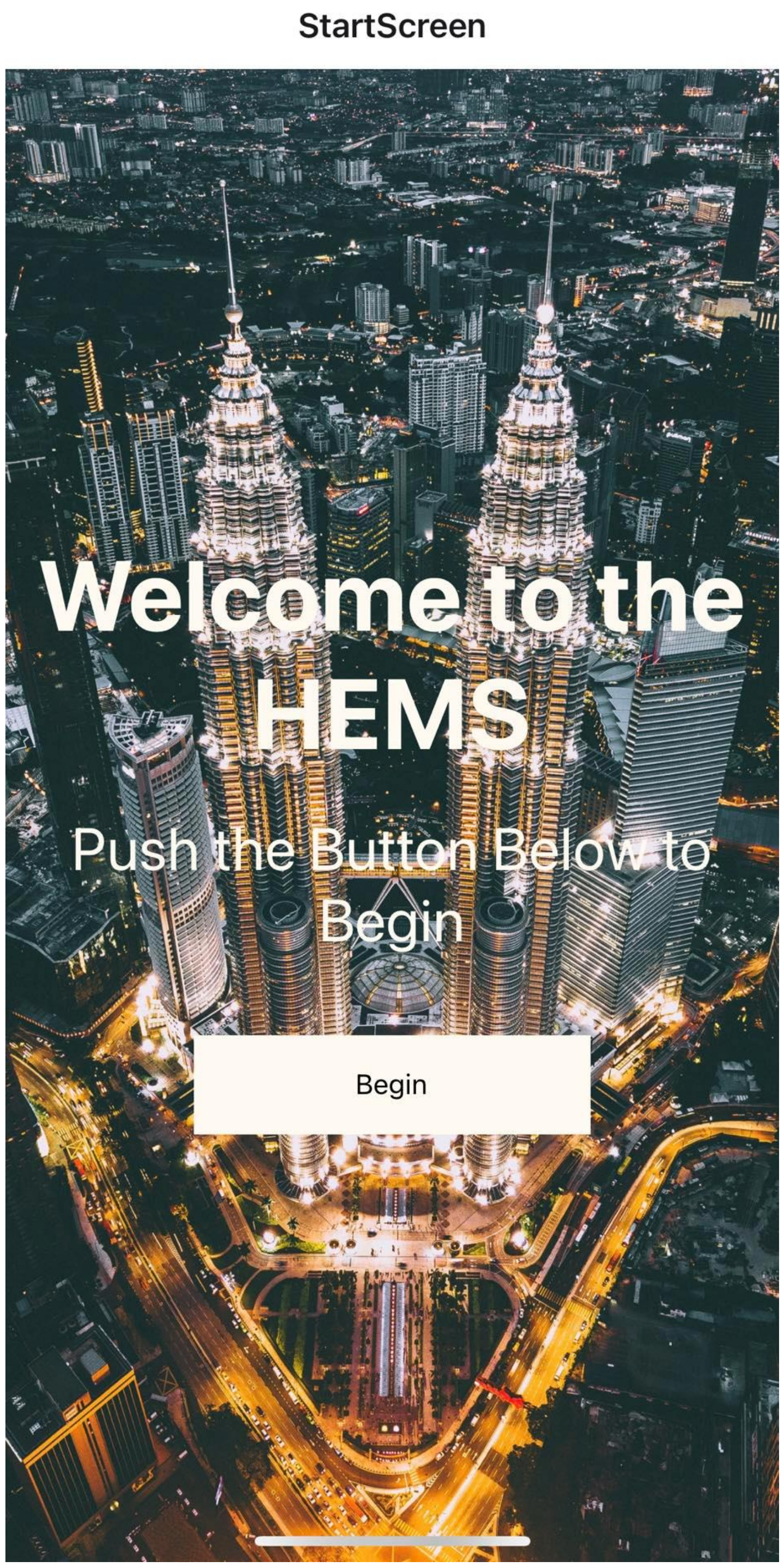


Figure 5.7.7: Start Page

5.7.2.2 Login Page

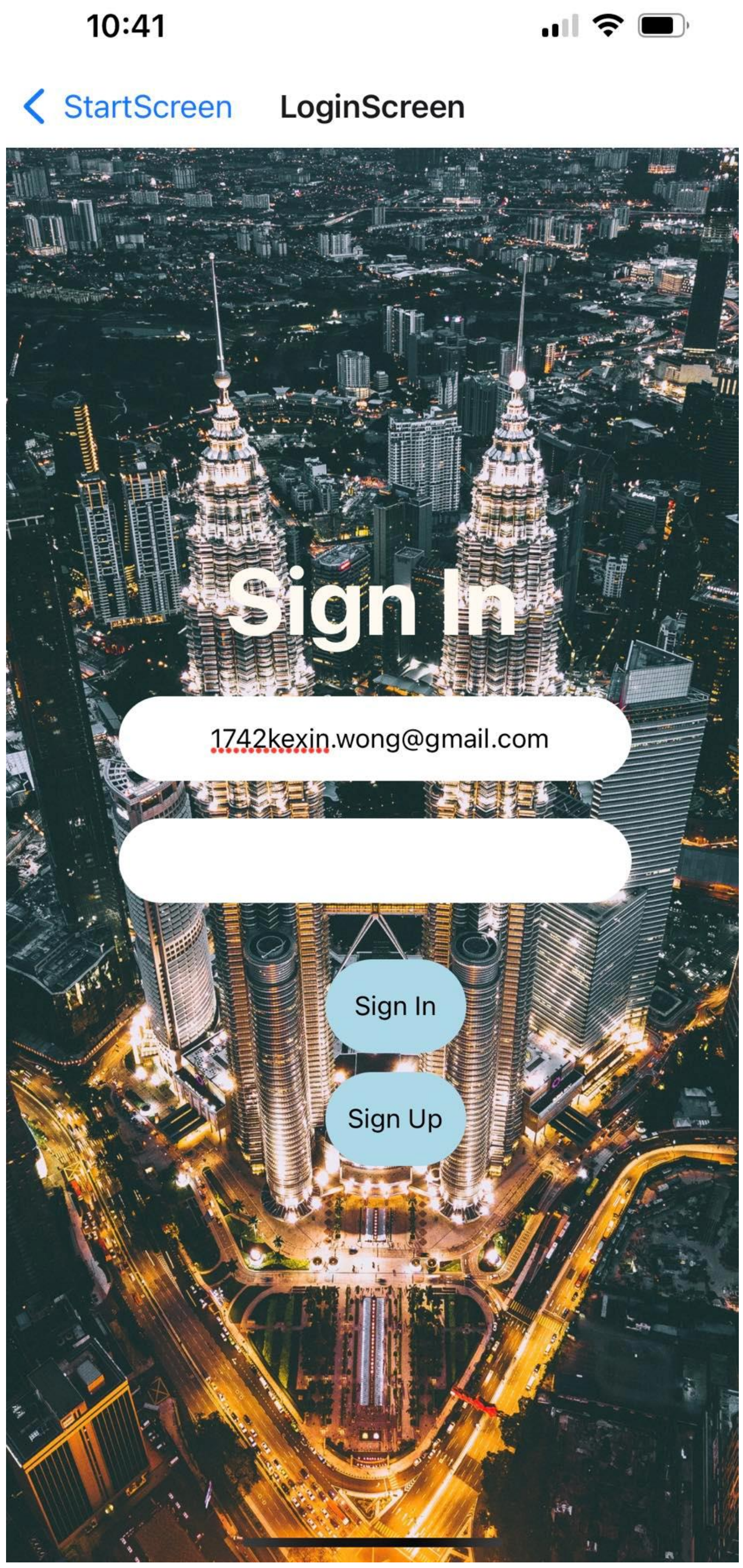


Figure 5.7.8: Login Page

5.7.2.3 Create Account Page

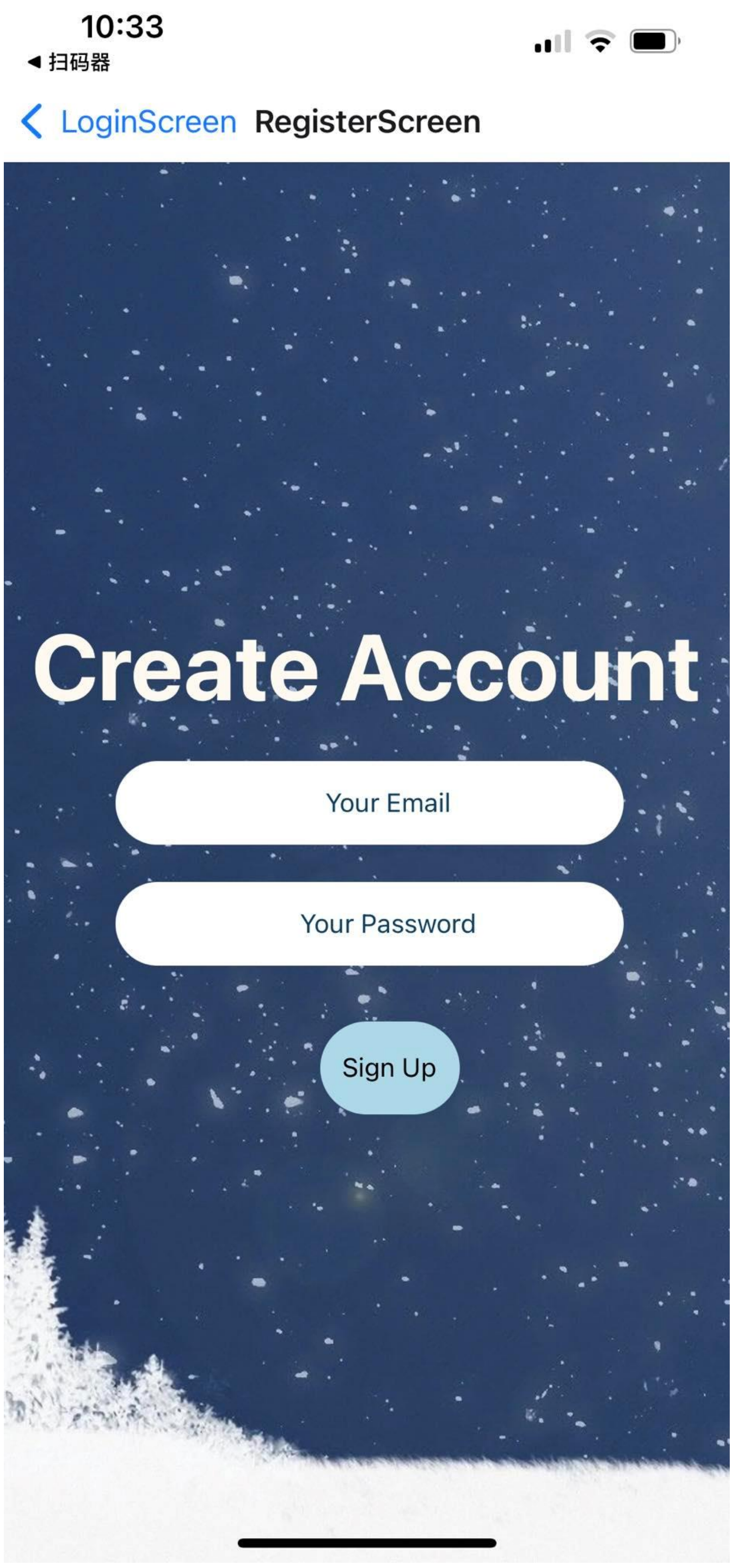


Figure 5.7.9: Create Account Page

5.7.2.4 Confirmation Page

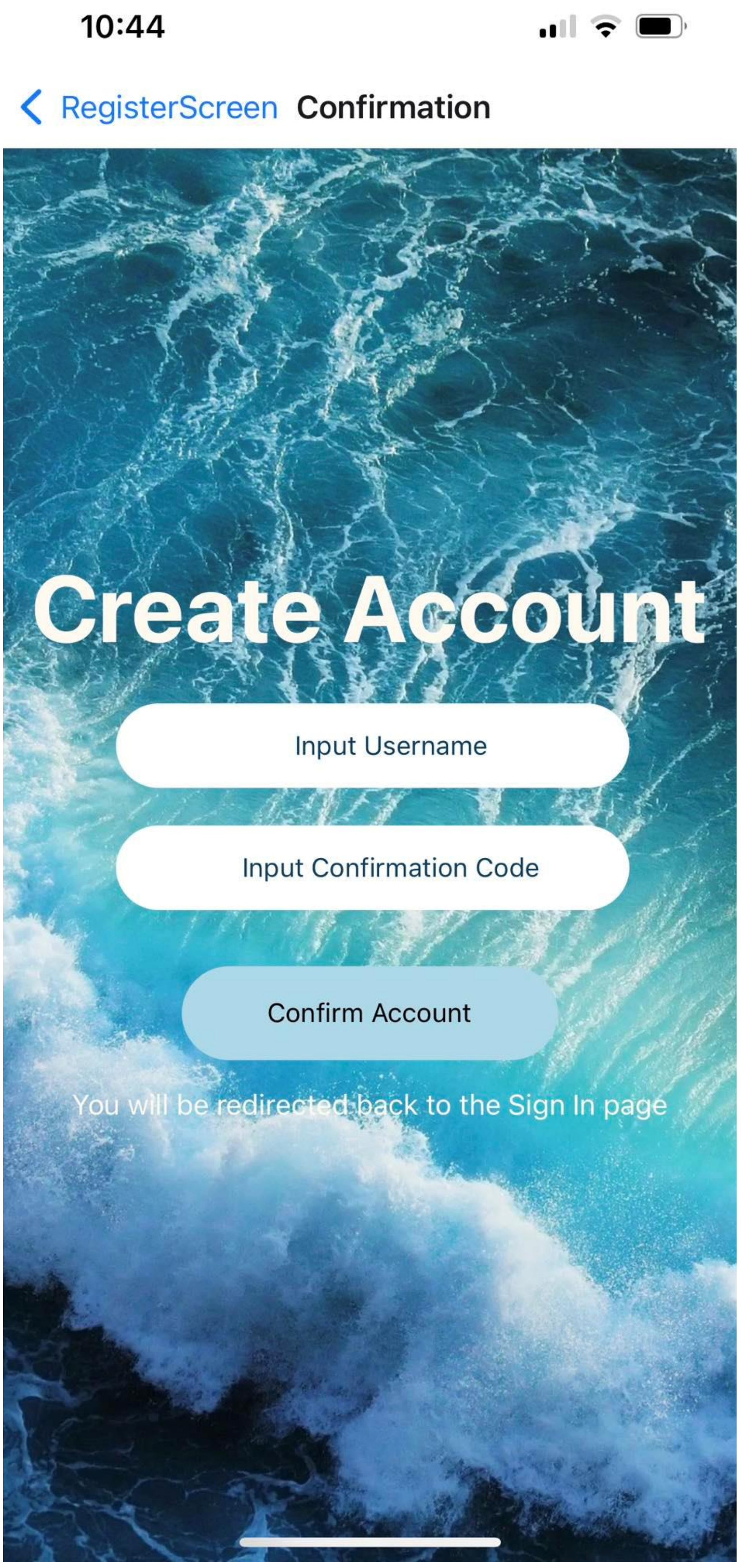


Figure 5.7.10: Confirmation Page

5.7.2.5 Home Page

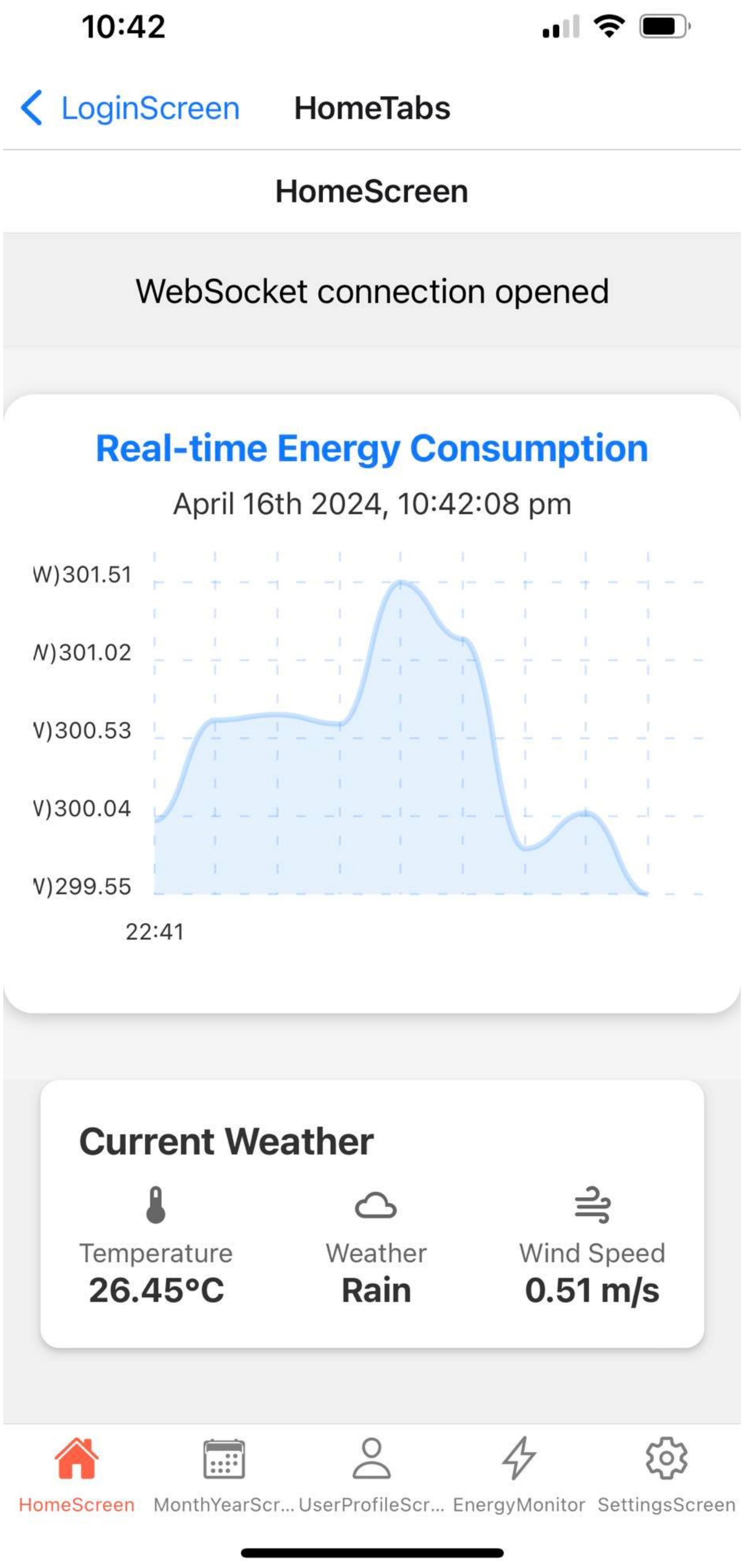


Figure 5.7.11: Home Page

5.7.2.6 Historical Energy Consumption Page (Weekly)

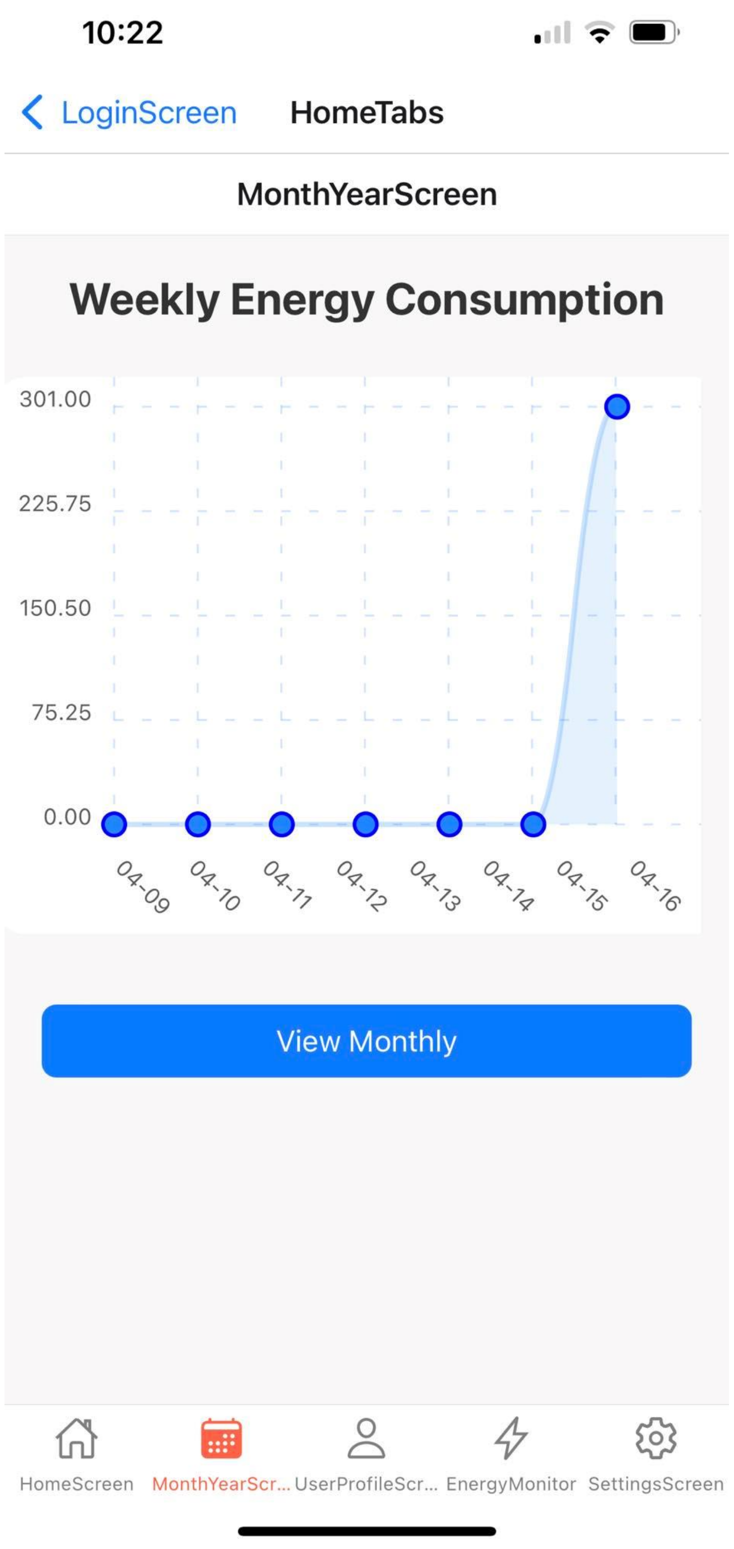
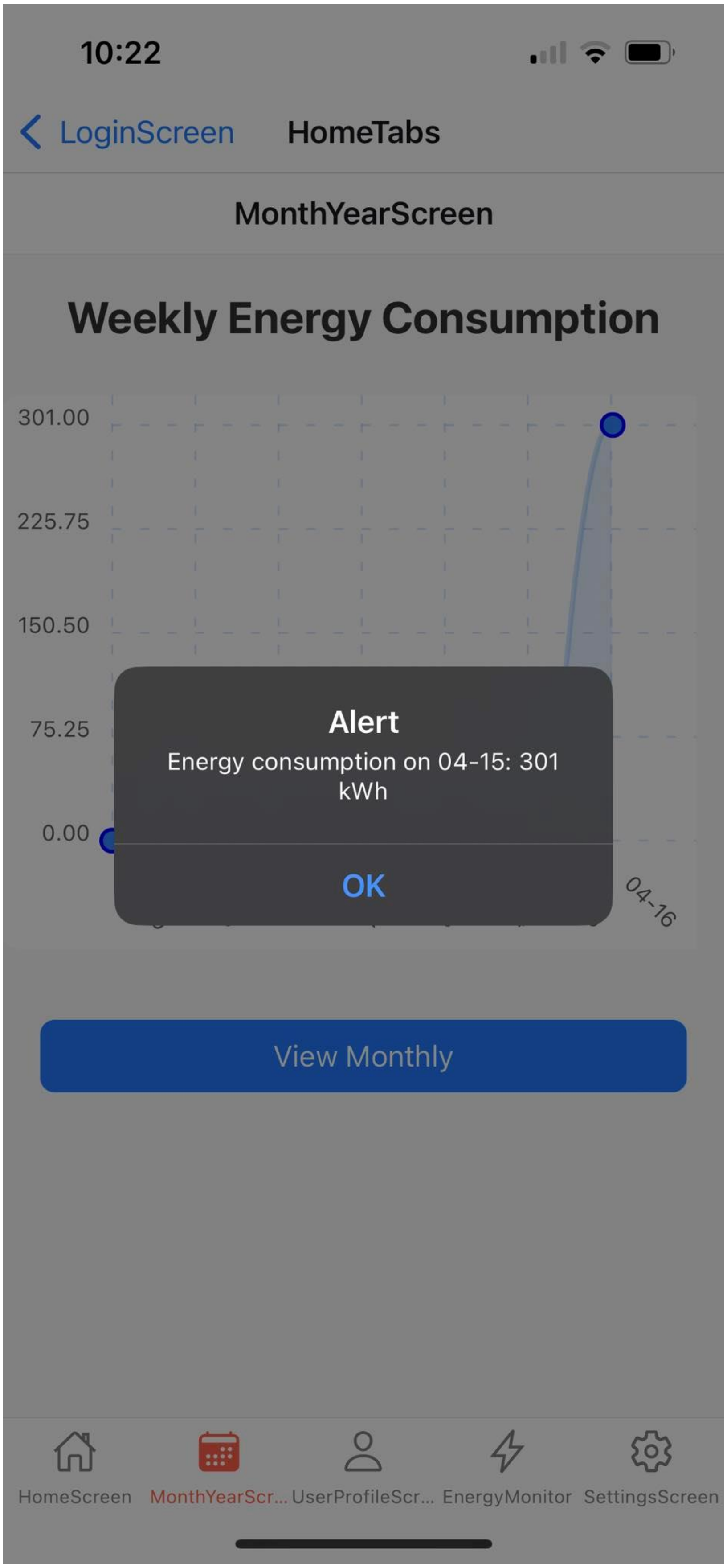


Figure 5.7.12: Historical Energy Consumption Page (Weekly)



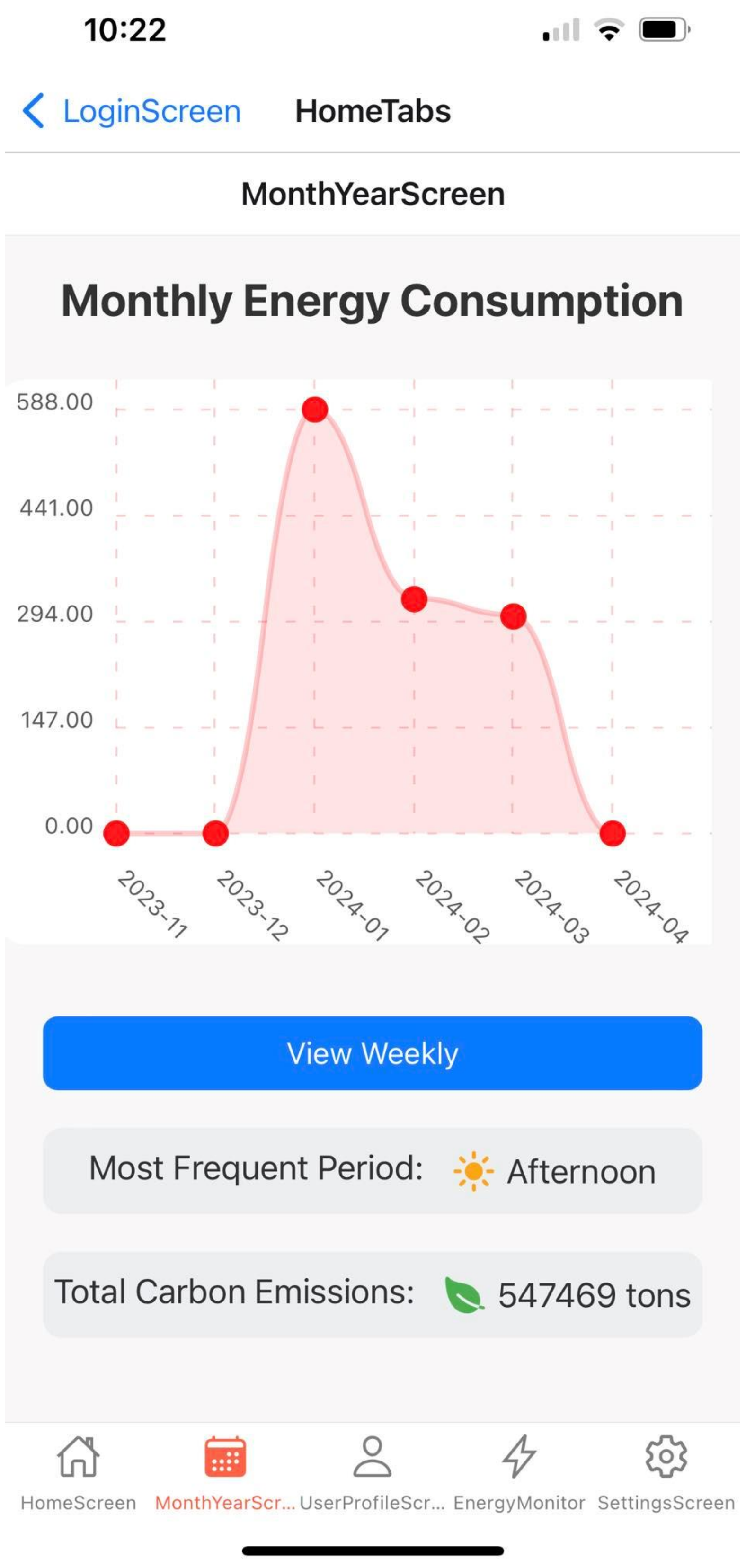
Weekly Energy Consumption

Alert

Energy consumption on 04-15: 301 kWh

Figure 5.7.13: Historical Energy Consumption Page Zoom In (Weekly)

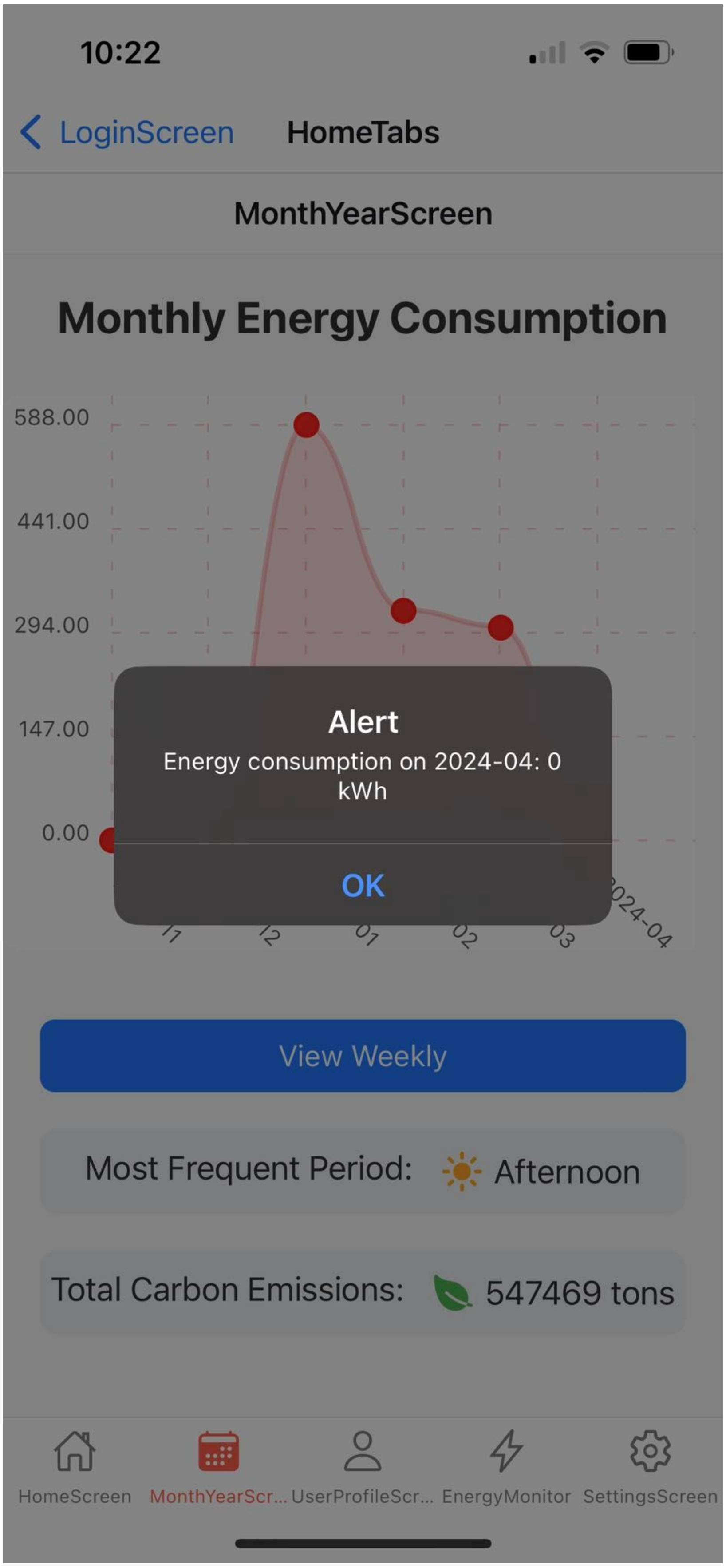
5.7.2.7 Historical Energy Consumption Page (Monthly)



Most Frequent Period: Afternoon

Total Carbon Emissions: 547469 tons

Figure 5.7.14: Historical Energy Consumption Page (Monthly)



Alert

Energy consumption on 2024-04: 0 kWh

OK

Most Frequent Period: Afternoon

Total Carbon Emissions: 547469 tons

Figure 5.7.15: Historical Energy Consumption Page Zoom In (Monthly)

5.7.2.8 User Profile Page

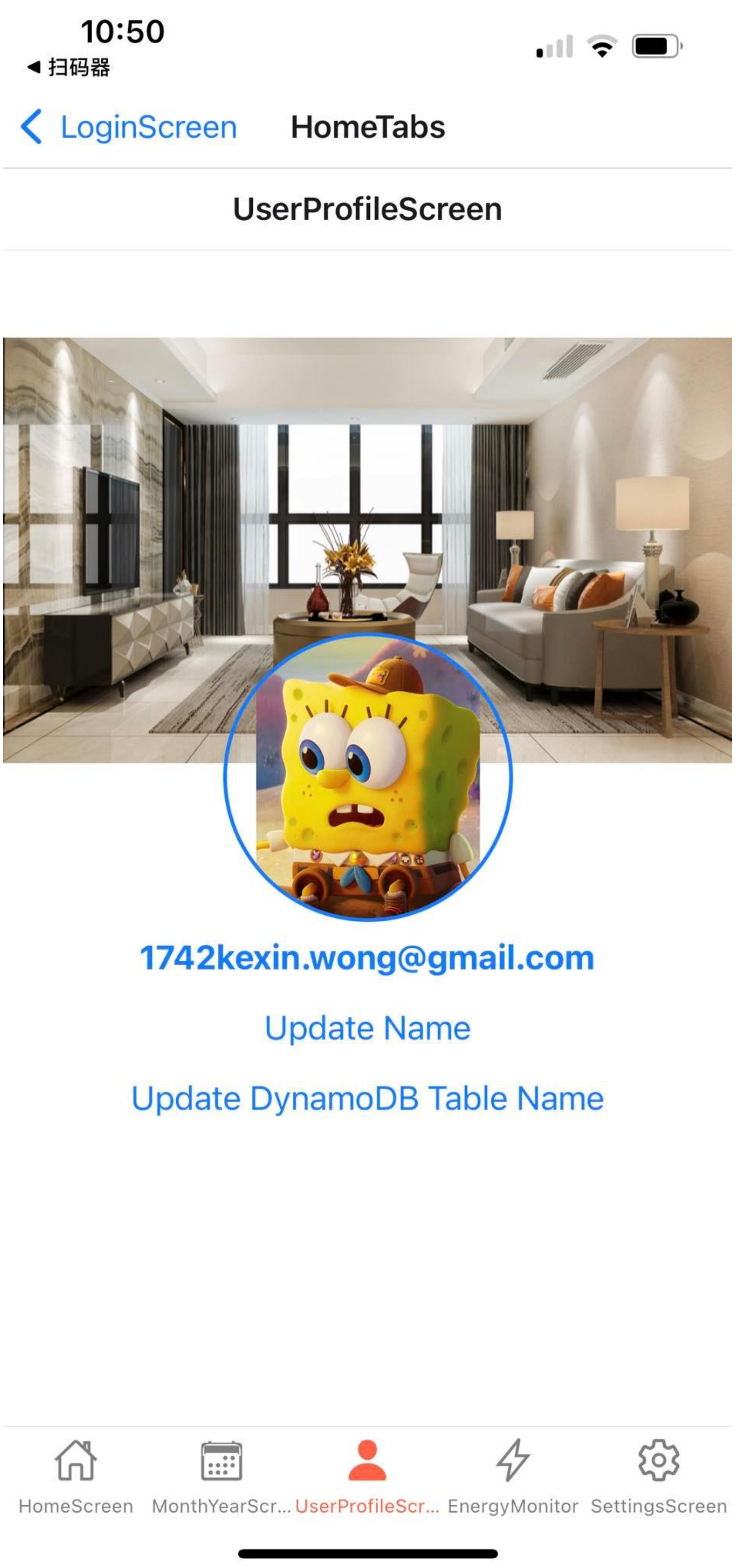
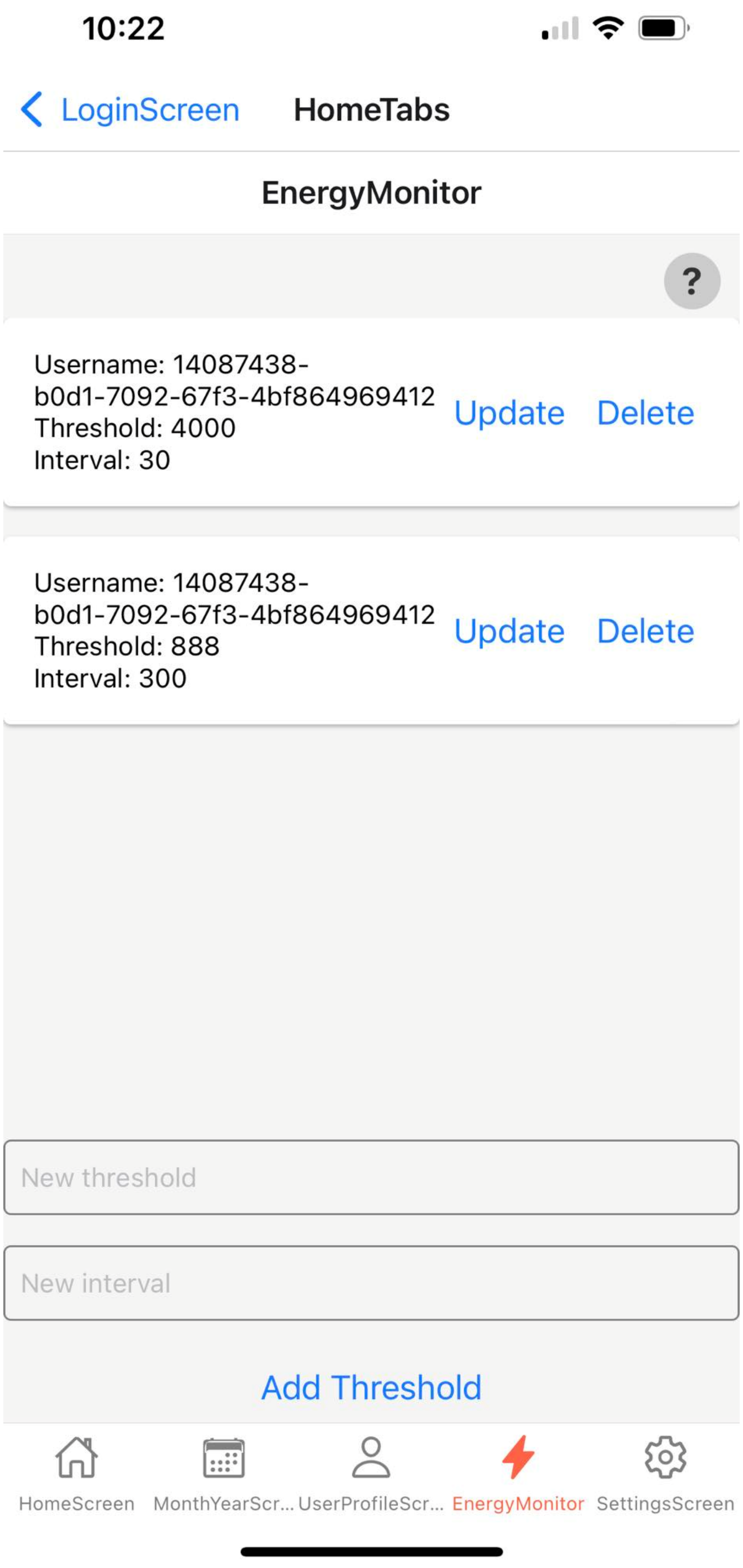


Figure 5.7.16: User Profile Page

5.7.2.9 Energy Monitoring Page


10:22

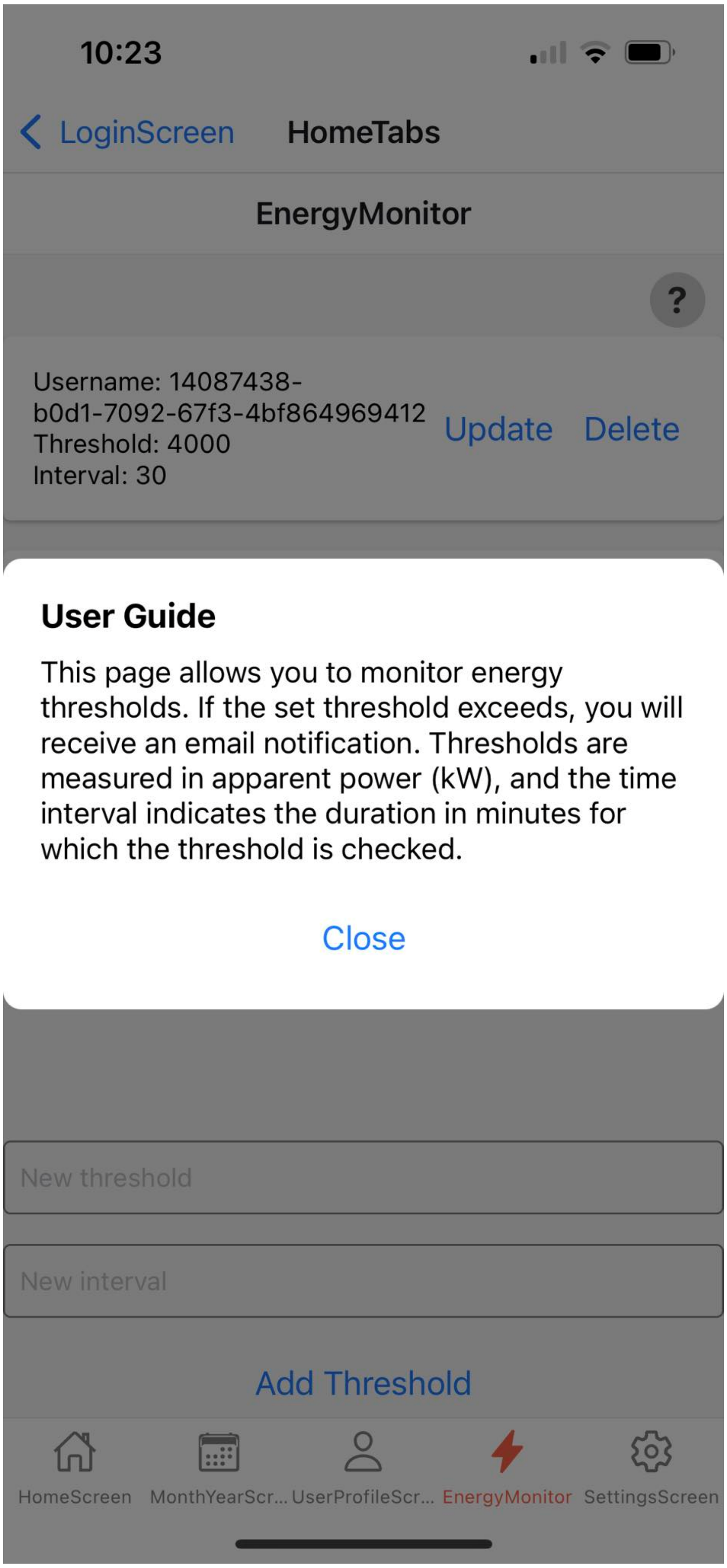
LoginScreen HomeTabs

EnergyMonitor

Username: 14087438-b0d1-7092-67f3-4bf864969412
Threshold: 4000
Interval: 30

Username: 14087438-b0d1-7092-67f3-4bf864969412
Threshold: 888
Interval: 300

Figure 5.7.17: Energy Monitoring Page

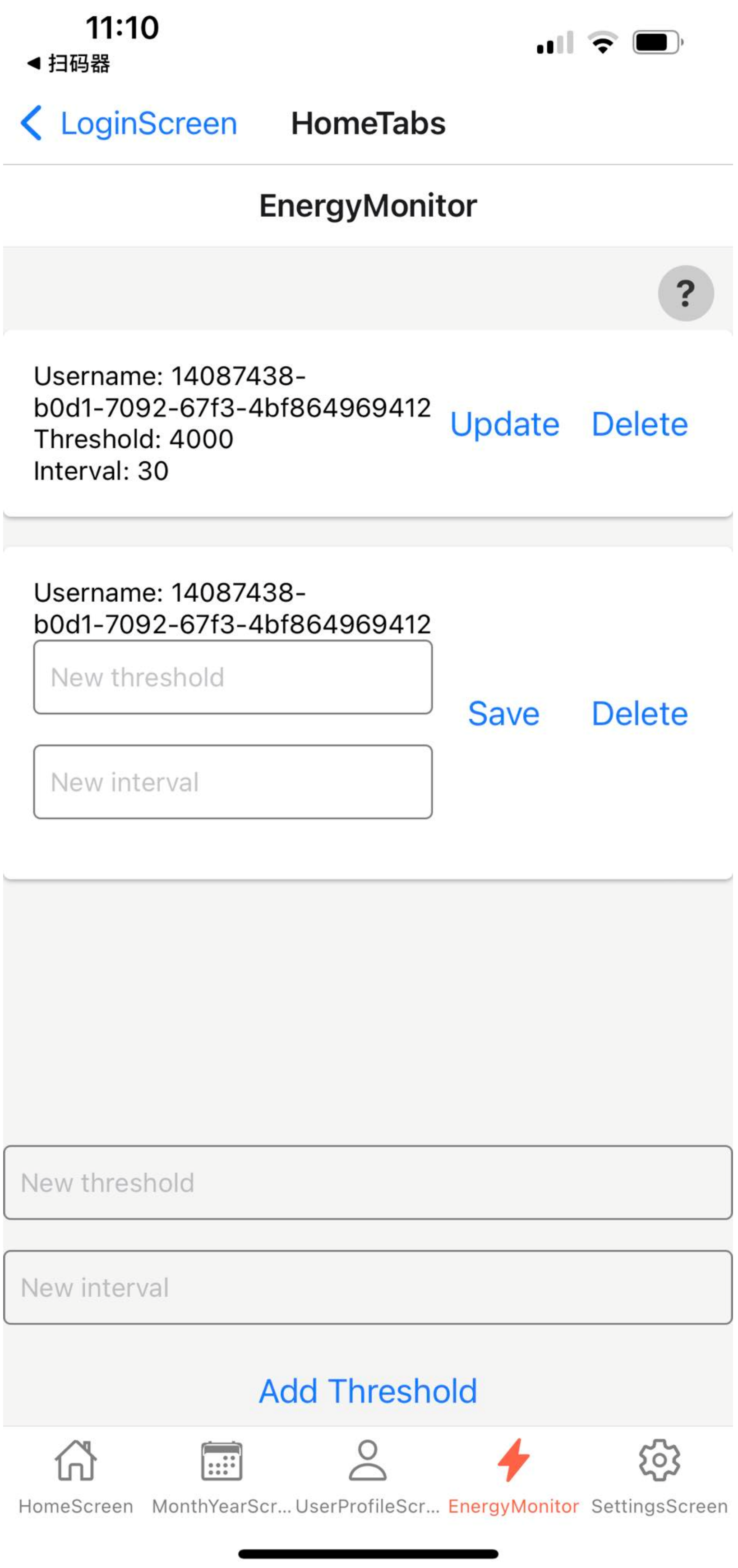


Username: 14087438-b0d1-7092-67f3-4bf864969412
Threshold: 4000
Interval: 30

User Guide

This page allows you to monitor energy thresholds. If the set threshold exceeds, you will receive an email notification. Thresholds are measured in apparent power (kW), and the time interval indicates the duration in minutes for which the threshold is checked.

Figure 5.7.18: Energy Monitoring Page Introduction


11:10

◀ 扫码器

LoginScreen HomeTabs

EnergyMonitor

?

Username: 14087438-b0d1-7092-67f3-4bf864969412
Threshold: 4000
Interval: 30

Update Delete

Username: 14087438-b0d1-7092-67f3-4bf864969412

New threshold

New interval

Save Delete

New threshold

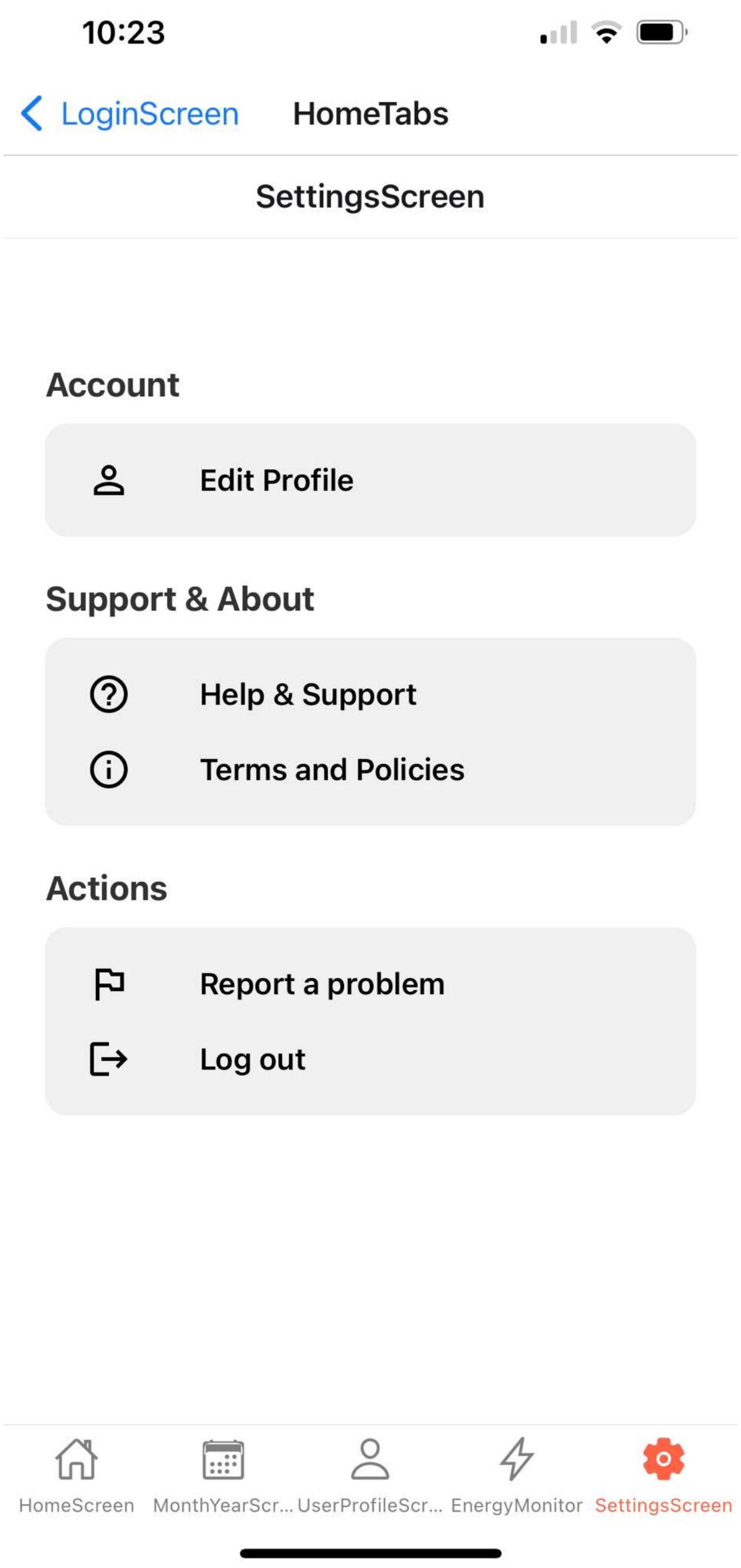
New interval

Add Threshold

HomeScreen MonthYearScr... UserProfileScr... EnergyMonitor SettingsScreen

Figure 5.7.19: Energy Monitoring Page Edit Mode

5.7.2.10 Setting Page


10:23

LoginScreen HomeTabs

SettingsScreen

Account

Edit Profile

Support & About

Help & Support

Terms and Policies

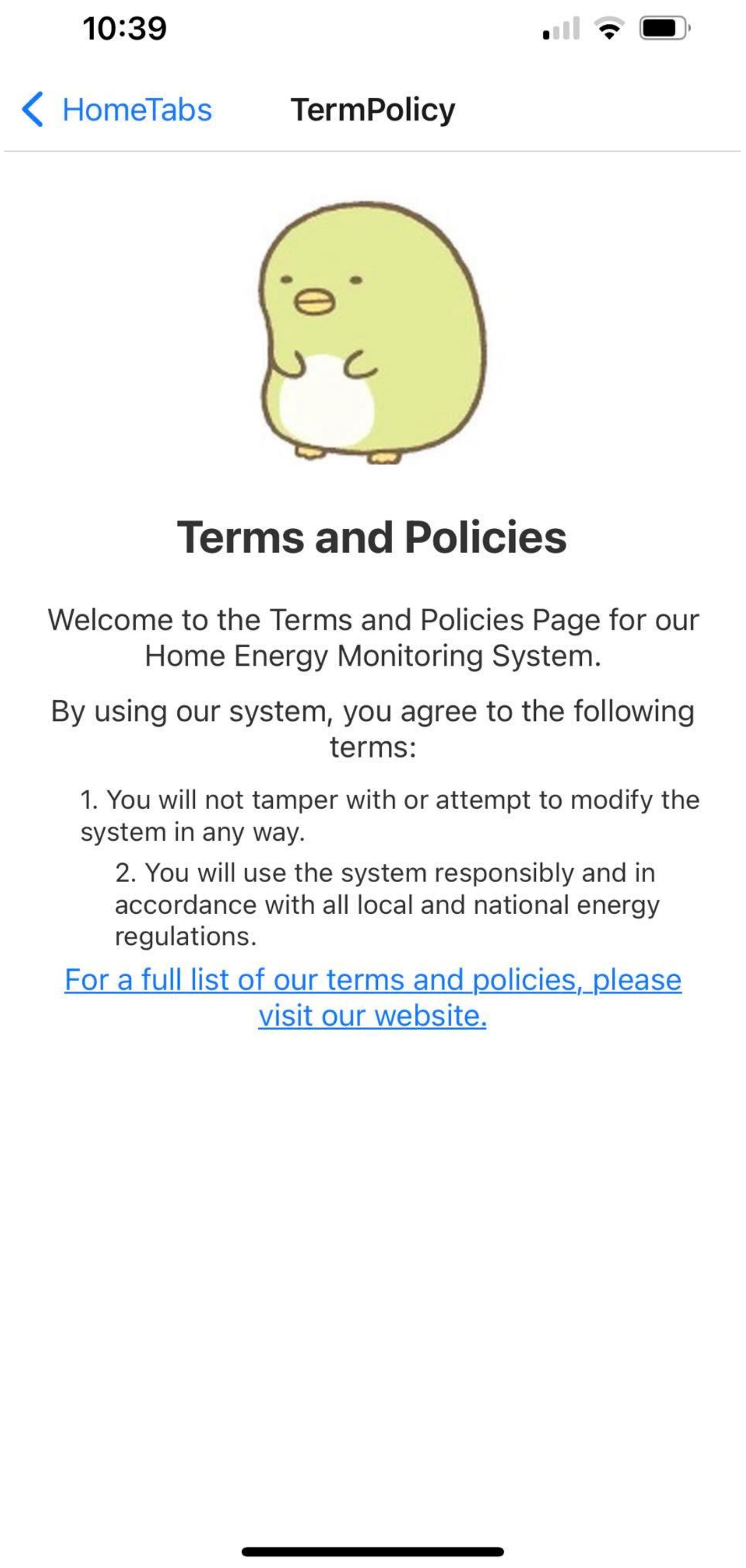
Actions

Report a problem

Log out

Figure 5.7.20: Setting Page

5.7.2.11 Term and Policy Page



Terms and Policies

Welcome to the Terms and Policies Page for our Home Energy Monitoring System.

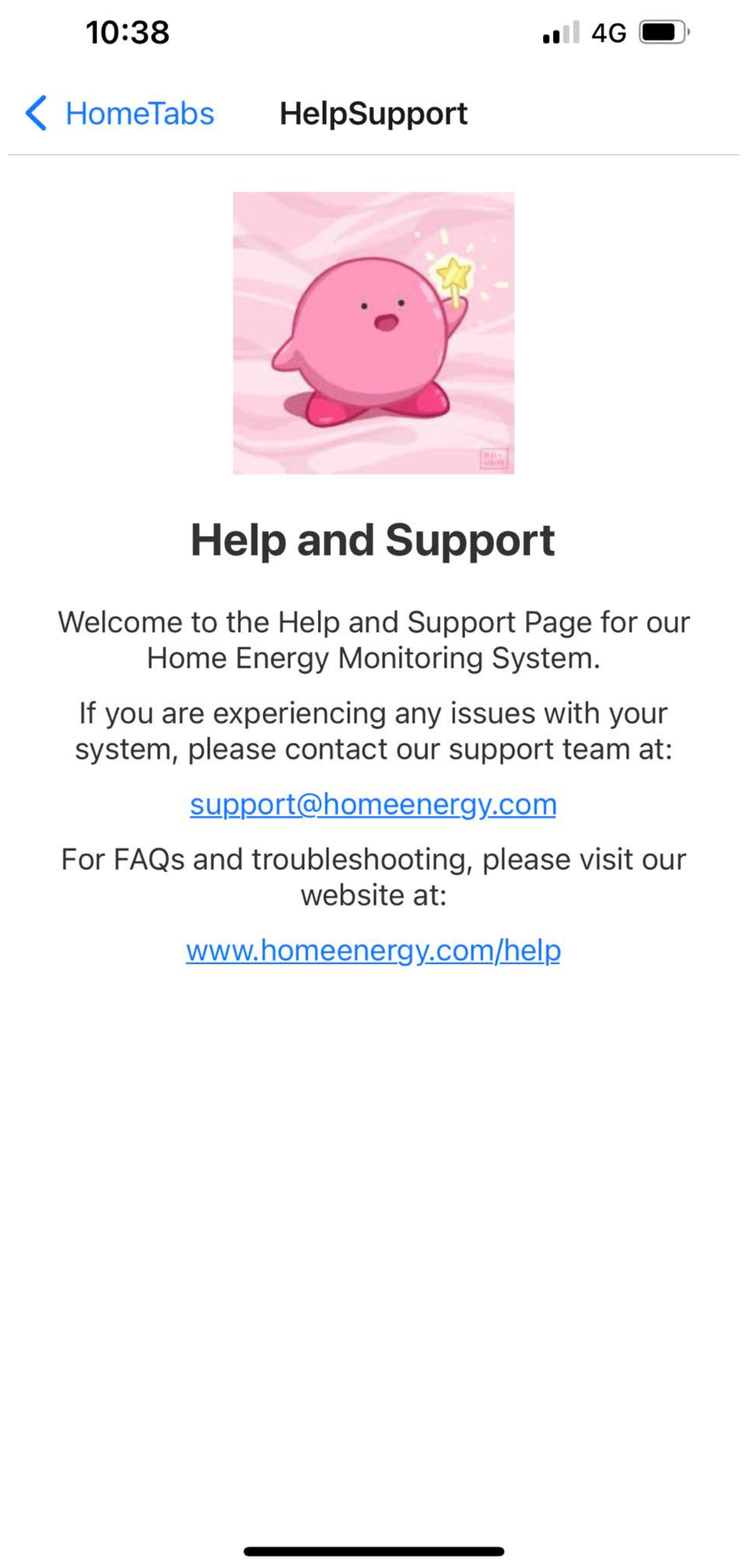
By using our system, you agree to the following terms:

1. You will not tamper with or attempt to modify the system in any way.
2. You will use the system responsibly and in accordance with all local and national energy regulations.

For a full list of our terms and policies, please visit our website.

Figure 5.7.21: Term and Policy Page

5.7.2.12 Help and Support Page



Help and Support

Welcome to the Help and Support Page for our Home Energy Monitoring System.

If you are experiencing any issues with your system, please contact our support team at:

support@homeenergy.com

For FAQs and troubleshooting, please visit our website at:

www.homeenergy.com/help

Figure 5.7.22: Help and Support Page

5.7.2.13 Report Problem Page

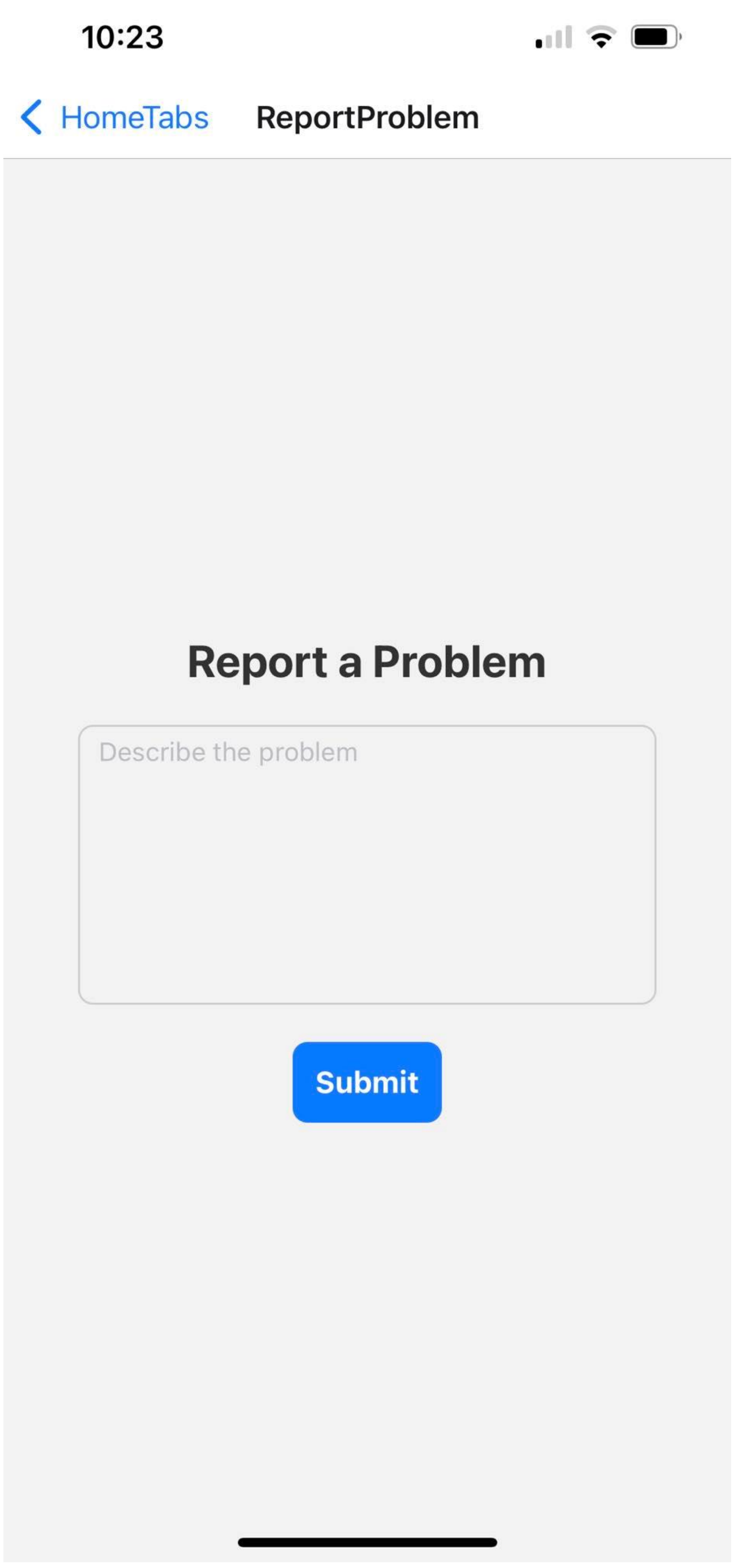
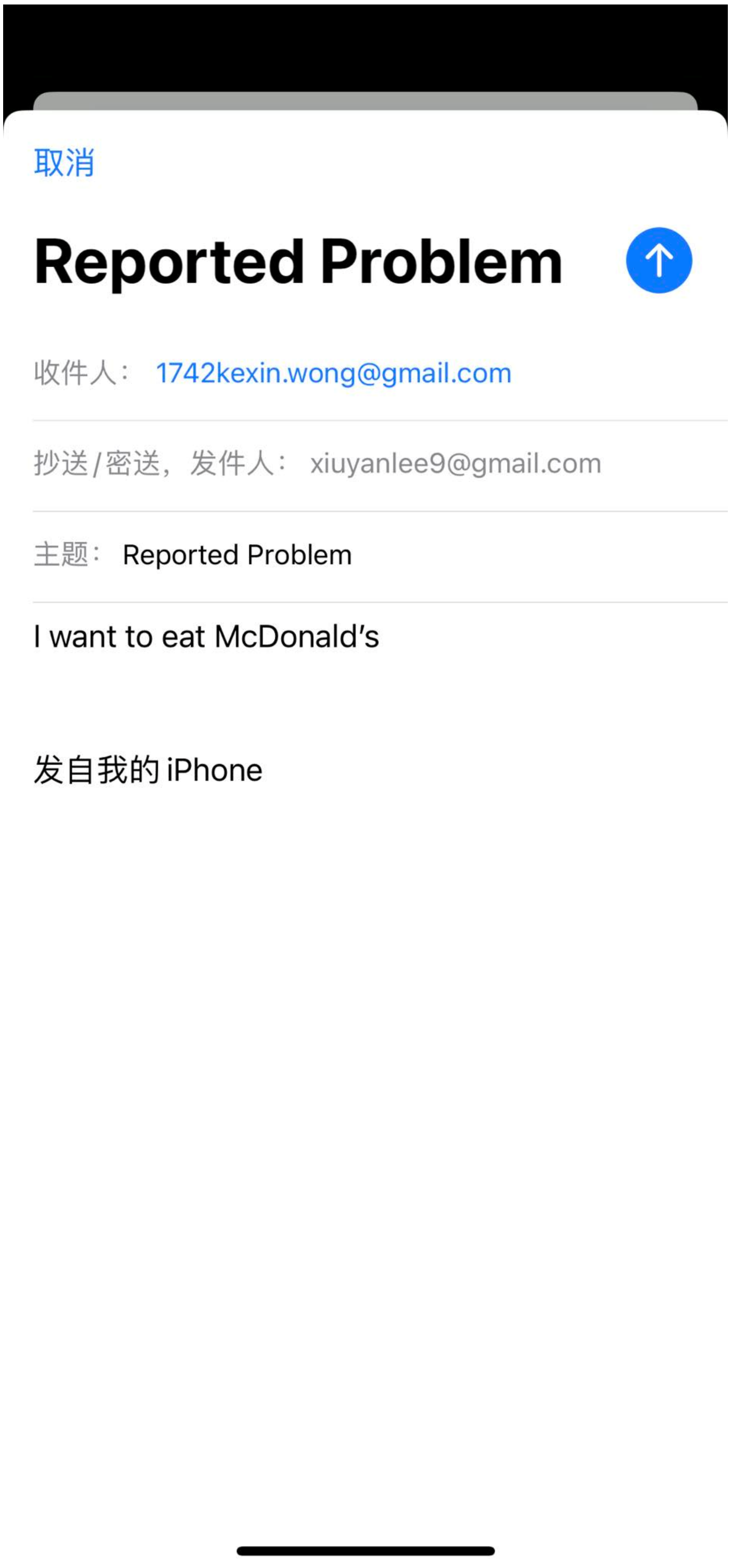


Figure 5.7.23: Report Problem Page



Reported Problem

收件人： 1742kexin.wong@gmail.com

抄送/密送，发件人： xiuyanlee9@gmail.com

主题： Reported Problem

I want to eat McDonald’s

发自我的 iPhone

Figure 5.7.24: Report Problem Page -Send Email

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Introduction

This chapter describes the workflow of the backend architecture of the whole project using AWS services. It covers the introduction of energy consumption simulator, the integration of different AWS services listed at the last chapter with mobile application and web application.

6.2 Energy Consumption Simulator Provided by MIMOS Bhd

6.2.1 Description of the Energy Consumption Simulator

The energy consumption simulator is provided by the industry linked company, MIMOS Bhd. The company laboratory is equipped with different home appliances as listed in the table below.

Table 6.2.1: Home Appliances in Laboratory

No	Appliance
1	Water Heater
2	Samsung Washing Machine
3	Washing Machine
4	Electrolux Oven (build in)
5	LG Clothes Dryer
6	Samsung Refrigerator (Freezer)
7	Refrigerator (Freezer)
8	Steam Iron
9	Khind Kettle
10	Vacuum Cleaner
11	York Air Conditioner
12	Panasonic Air Conditioner
13	Daikin Air Conditioner

Each of them is plugged into a different socket for data collection in order to collect data of different appliances in different conditions and modes.

Collected data for single appliances and the combination of different appliances data will be shown in raw data format and be converted to txt file. The company team performed feature extraction and data transformation so that the readable summarized data is stored in different csv file. According to these data collected, the company produced an energy consumption simulator which simulates the total energy consumption from on off status of different appliances. This simulator will be used to mimic the actual result of energy monitoring device in this project.

The energy consumption data are recorded using 13 attributes listed below:

1. Timestamp

   The specific time when the energy consumption data was recorded.
2. RMS Current

   RMS=1nii=1,2,3xi2

   RMS=Root Mean Square

   n=Number of Samples

   xi=Each Value

   It measures the average current flow through the circuit by considering both magnitudes and the variation of the signal over time.
3. Average Current

   The average amount of current flowing through the circuit during a specific time interval.
4. Peak Current

   The maximum current intensity reached by an appliance during its operation or the highest load the appliance draws.
5. Apparent (VA)

   The total power in an electrical circuit which combine both real power and reactive power.

   It is amount of power used to operate appliances during a certain period.
6. Active (W)

   The real power used by the appliance to do work.
7. Reactive (VAR)

The wasted power that produced during operation without performing work.

8. Power Factor

   The ratio between active power and apparent power.

   It indicates the effectiveness of the appliances to convert electricity to useful work.

9. Fundamental Frequency (50Hz)

   Primary frequency of electrical supply

10. Third Harmonics (150Hz)

    It is a frequency that is three times the fundamental frequency

    It affects the efficiency of devices and the quality of the electrical supply.

11. Fifth (250Hz)

    It represents a frequency that is five times over the fundamental frequency.

    It is used to diagnosis the power quality issues.

12. Seventh (350Hz)

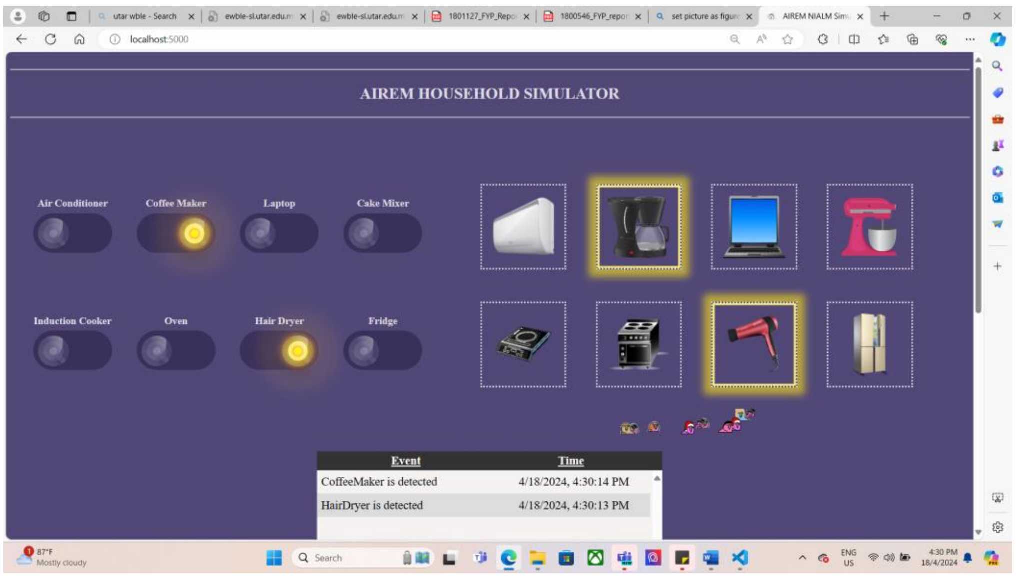
    It shows a frequency that is seven times over the fundamental frequency.

    It can used to detect the overheating of appliances.

13. Ninth (450Hz)

    It represents a frequency that is nine times over the fundamental frequency.

    It helps in maintaining the efficiency and longevity of appliances.



Event	Time
CoffeeMaker is detected	4/18/2024, 4:30:14 PM
HairDryer is detected	4/18/2024, 4:30:13 PM

Figure 6.2.1: Web Interface of Energy Monitoring Simulator

6.3 Sending Data from Simulator to DynamoDB database

The details implementation of AWS backend services from energy monitoring device will be explained with some code segments.

6.3.1 Send data from energy monitoring simulator to DynamoDB by setting up AWS IoT Core and configuring the IoT device to communicate with AWS services

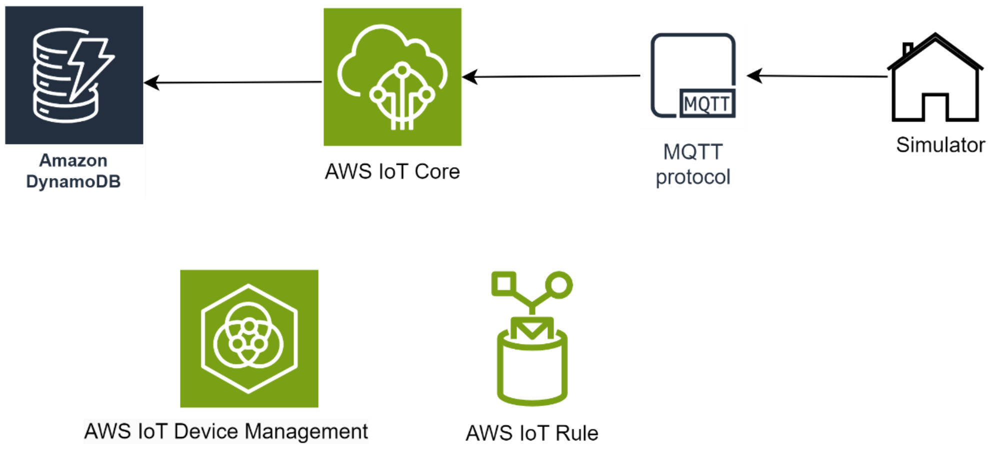


Figure 6.3.1: Send Data to DynamoDB Workflow

6.3.1.1 Creating IAM User and Downloading Certificate

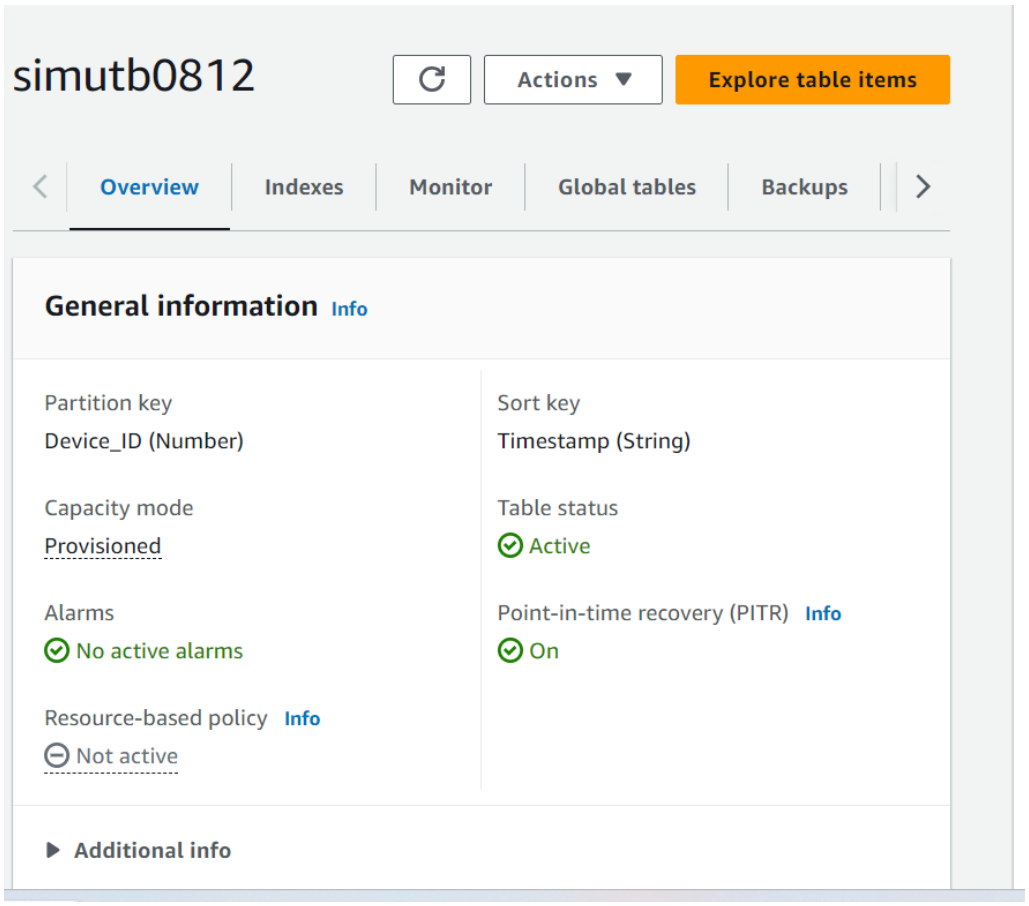
Firstly, a new IAM user is created with policies related to AWS IoT Core, DynamoDB and DataAccess. AmazonRootCA1.pem, certificate.pem.crt and private.pem.key must be downloaded from this user to place inside IoT device that is energy monitoring simulator.

6.3.1.2 Create Energy Consumption DynamoDB Table

An Energy Consumption DynamoDB table has been designed within Amazon DynamoDB to cater for the storage and management of energy consumption data from the energy monitoring simulator. This DynamoDB table serves as a robust and scalable repository to efficiently handle large volumes of energy consumption data and integrate it with different AWS services.

The Device_ID attribute, which is the Number type, is designated as the partition key in this DynamoDB table. This key is instrumental in distributing data across DynamoDB partitions to enable high-performance and parallel processing of queries. Furthermore, the Timestamp attribute, defined as a String type, acts as the sort key within the table to sort energy consumption data, facilitating temporal analysis and trend identification in AWS Lambda later.

In addition to the key attributes, the DynamoDB table encompasses a comprehensive set of attributes capturing various aspects of energy consumption such as ApparentPower, CoefficientReal3H, CoefficientReal5H, CoefficientReal70Hz, CoefficientReal7H, CoefficientReal9H, HarmonicDistortionPower, MAIN_PREDICTION, OTHR_PREDICTION, OTR_POWER, PhaseShift, ReactivePower, and TotalPowerFactor.



General information

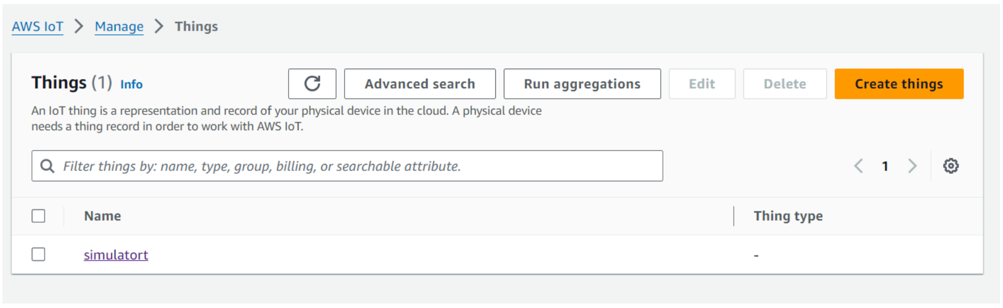
Partition key Device_ID (Number)	Sort key Timestamp (String)
Capacity mode Provisioned	Table status Active
Alarms No active alarms	Point-in-time recovery (PITR) On
Resource-based policy Not active	

Figure 6.3.2: Creation of Energy Consumption DynamoDB Table

This figure presents the configuration of DynamoDB table for energy consumption. The energy consumption DynamoDB table guarantees effective data storage, retrieval, and analysis by utilizing DynamoDB's adaptable and scalable architecture. Monitor features included in DynamoDB can be used to investigate and monitor the CRUD function performed inside DynamoDB table overtime.

6.3.1.3 Creating a 'Thing' in AWS IoT Core

Moreover, a ‘thing’ is created inside the AWS IoT core to serve as a digital representation of a physical device. This energy monitoring simulator pretends to mimic the real performance of the energy monitoring device and sends detected energy consumption data. This ‘thing’ will act as the endpoint for communication between the energy monitoring simulator and AWS services. This simulator can communicate with the AWS IoT core using the MQTT protocol by sending data and interacting with other AWS services.


AWS IoT > Manage > Things

Things (1) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Advanced search
Run aggregations
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
1

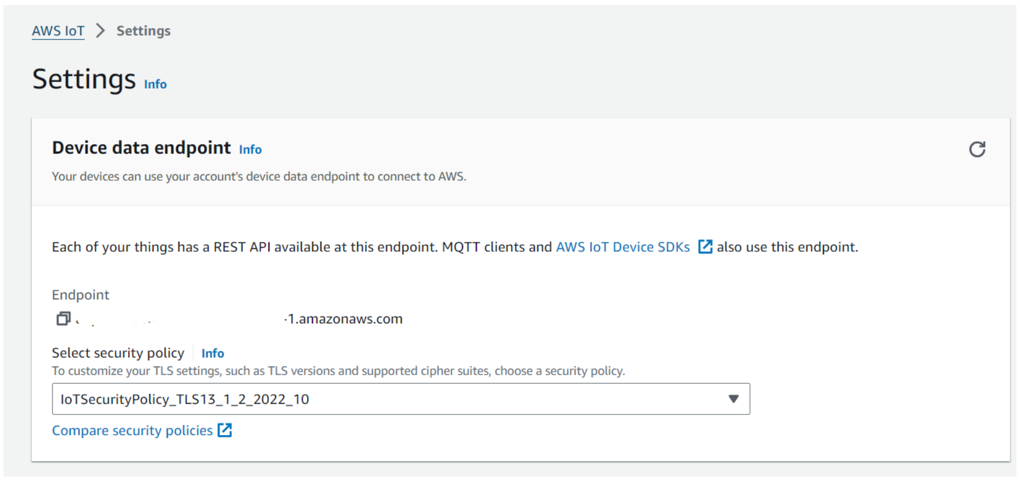
Name	Thing type
simulatort	-

Figure 6.3.3: Created Things in AWS IoT Core

The figure illustrates the creation of a Thing within AWS IoT Core, which is a pivotal component utilized in this project. It establishes the foundation for effective device management, data exchange, and IoT workflow orchestration to leverage real-time insights and actionable intelligence from the energy monitoring simulator.

6.3.1.4 Retrieving Device Data Endpoint

Next, the device data endpoint is retrieved via the setting of AWS IoT Core so that this endpoint can be inserted into the simulator to connect to AWS. The URL or endpoint that a device can transmit its data to for processing, storing, or analysis is known as the device data endpoint. The device data endpoint in an Internet of Things (IoT) project usually refers to a web service or API endpoint that is offered by a cloud service such as Google Cloud IoT Core, Azure IoT Hub, or Amazon IoT Core. A device can use different communication protocols, such as HTTP, MQTT, or WebSocket, to send the data it creates to the device data endpoint, depending on the features and specifications of the IoT platform being used.



AWS IoT > Settings

Settings Info

Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint

-1.amazonaws.com

Select security policy Info

To customize your TLS settings, such as TLS versions and supported cipher suites, choose a security policy.

IoTSecurityPolicy_TLS13_1_2_2022_10

Compare security policies

Figure 6.3.4: Settings in AWS IoT Core

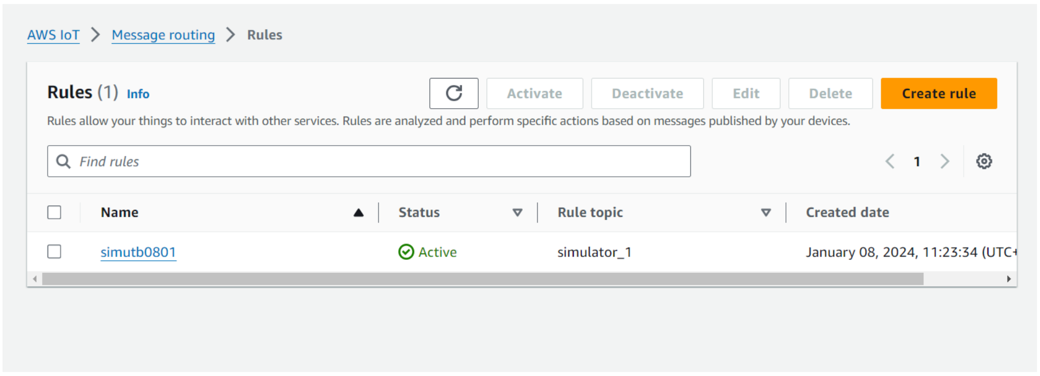
The figure shows the location to retrieve Device data endpoint in AWS IoT Core.

6.3.1.5 Setting Up a Rule

Furthermore, a ‘rule’ is created to specify actions that should be taken when certain conditions are met and a rule is used to process messages that are published to MQTT topics by an energy monitoring simulator within the IoT environment. This means that when new energy consumption data is published to the defined topic, AWS IoT Core will examine the data against the created ‘rule’ and execute the associated action when the specified conditions have been fulfilled. In this case, a simple SQL statement is set below to retrieve all data from the MQTT topic.

```
SELECT * FROM "simulator_1"

//Tale note that simulator_1 is the topic name for this rule
which will be used later
```

The action of this ‘rule’ will be set to DynamoDBv2, which will split energy consumption data into multiple columns of the DynamoDB table, which is the energyConsumption table for the specific residents. This action will be executed if the SQL statement set has been executed correctly.



Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

Name	Status	Rule topic	Created date
simutb0801	Active	simulator_1	January 08, 2024, 11:23:34 (UTC

Figure 6.3.5: Creation of Rules in AWS IoT Core

This figure demonstrates the creation of Rules with Rules name and Rule topic that will be used inside energy monitoring simulator.

6.3.1.6 Set Up AWS Credential and AWS Host

The client and thing name are the same, which can be retrieved from ‘thing’, the device data endpoint, and the topic name will be placed inside the script of the energy monitoring simulator to connect to AWS IoT core using MQTT over TLS, which publishes messages to the specified topic and insert data into DynamoDB table.

6.4 Retrieve Realtime Energy Consumption Data from DynamoDB Table to Mobile Application via WebSocket API

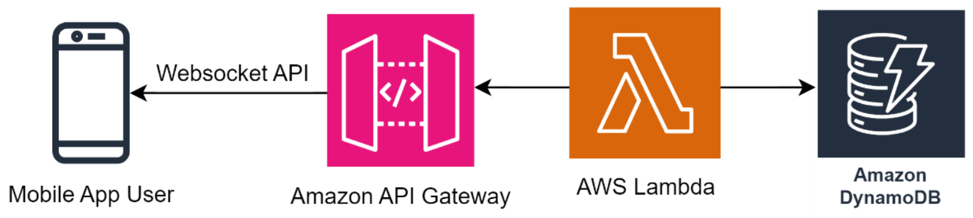


Figure 6.4.1: Retrieve Realtime Energy Consumption Data from DynamoDB Table to Mobile Application Workflow

The energy consumption simulator continues to send energy consumption data to the Energy Consumption DynamoDB table. A WebSocket API is being implemented to transmit energy consumption data to the mobile app in real time.

6.4.1 Set DynamoDB stream

A DynamoDB stream is a feature that detects changes to the items in a DynamoDB table, such as inserting and deleting items. It is enabled in the energy consumption table to allow real-time processing of these inserted energy consumption data by triggering events upon modification. AWS Lambda will consume these event functions to integrate with other AWS services and mobile applications.

6.4.2 Create connection DynamoDB table

This table with connectionID that will be created for specific user once they connect to WebSocket API and WebSocket API for each resident.

6.4.3 WebSocket API Management

6.4.3.1 Create WebSocket API using AWS API Gateway

A WebSocket API will be created to send real time energy consumption data from DynamoDB table to mobile application.

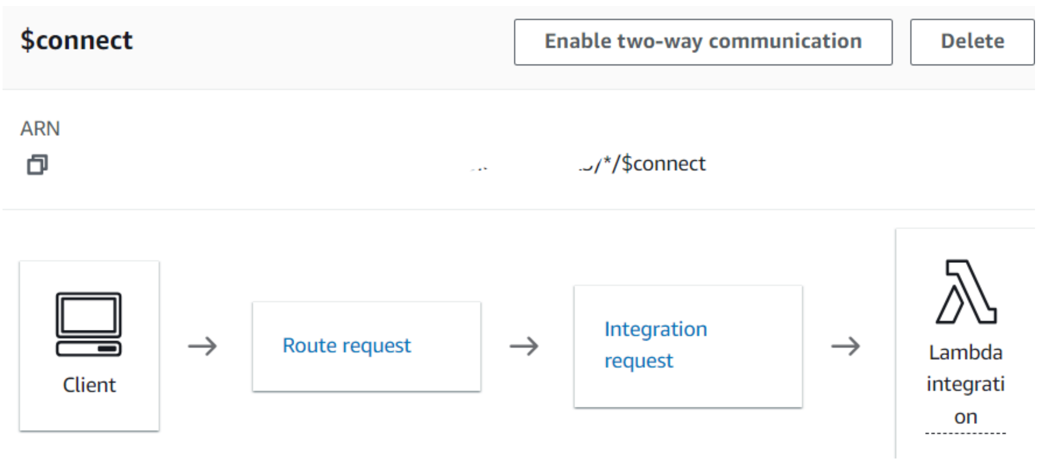


Figure 6.4.2: $connect route in WebSocket API

This figure presents the $connect route defined inside WebSocket API. The $connect route is integrated with an AWS Lambda function that is responsible for inserting the connectionID into the connection DynamoDB table. When a resident opens the mobile application, it establishes a connection to the respective WebSocket API. As part of this connection process, a connectionID is generated within the response of the WebSocket API invocation. This connectionID is crucial for the Lambda function that integrates with the DynamoDB stream event to facilitate the subsequent sending of energy consumption data.

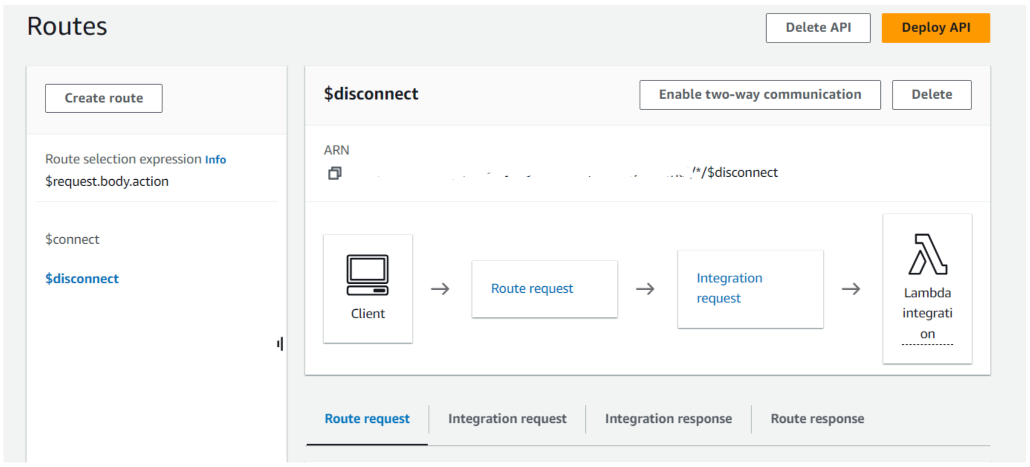
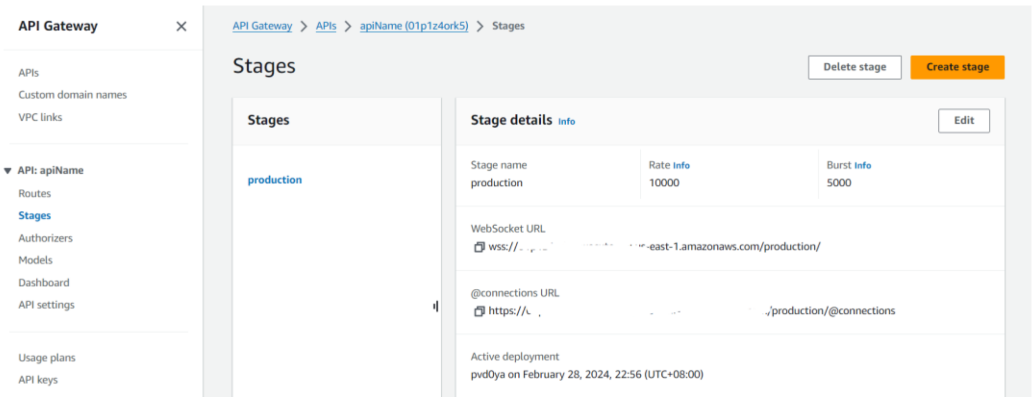


Figure 6.4.3: $disconnect route in WebSocket API

This figure presents the $disconnect route defined inside WebSocket API. The $disconnect route is integrated with an AWS Lambda function that deletes the connectionID from the connection DynamoDB table. When the mobile application disconnects from the WebSocket API, the generated connectionID will be removed. Each connection with the WebSocketAPI generates a new connectionID, rendering the old connectionID obsolete.



Stage name: production
Rate Info: 10000
Burst Info: 5000
WebSocket URL: wss://...-east-1.amazonaws.com/production/
@connections URL: https://.../production/@connections
Active deployment: pvd0ya on February 28, 2024, 22:56 (UTC+08:00)

Figure 6.4.4: Stage of WebSocket API

The WebSocket URL is obtained from the 'Stage' section of the AWS API Gateway console, allowing the mobile application to establish a connection. This URL serves as the endpoint through which the application connects to the WebSocketAPI.

6.4.4 Integration of AWS Lambda functions, Amazon DynamoDB, and Amazon API Gateway to handle WebSocket communication

The energy monitoring simulator will send data to DynamoDB every second, but the real-time data will only reach residents once they connect to the WebSocket by opening the mobile application. The energy monitoring simulator continuously sends data to DynamoDB at regular intervals, ensuring that the latest information is consistently updated in the database. However, the delivery of this data to residents occurs only when they establish a connection to the WebSocket via the mobile application.

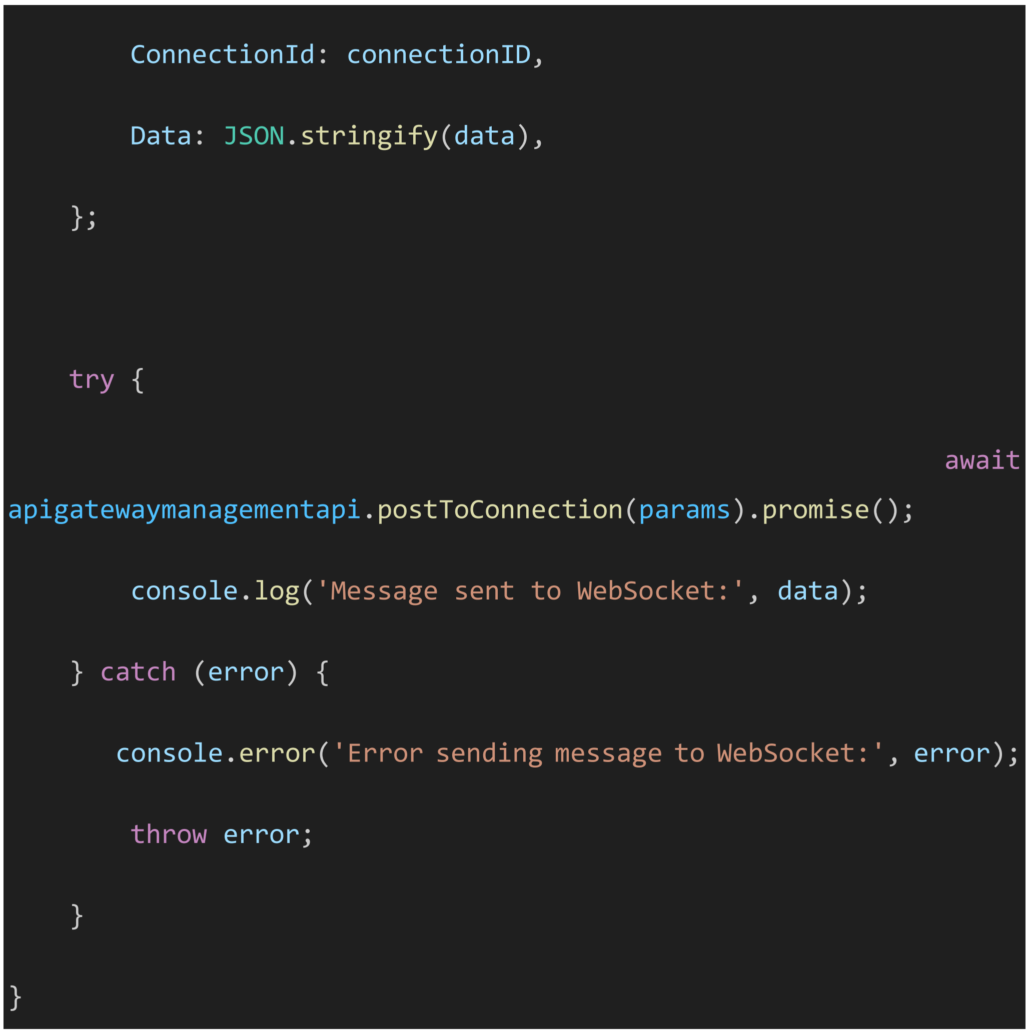
This approach ensures efficient resource utilization and guarantees that the latest energy consumption data is always available in the DynamoDB table. This is achieved by decoupling data generation (simulator sending data to DynamoDB) from data consumption (residents retrieving data via WebSocket). Real-time interaction is facilitated by enabling residents to receive updates only when they connect to the WebSocket. This strategy reduces overhead and enhances the overall user experience by minimizing unnecessary network traffic.

```
const AWS = require('aws-sdk');

const apigatewaymanagementapi = new AWS.ApiGatewayManagementApi({

    apiVersion: '2018-11-29',

        endpoint:    'https://ownWebsocketAPI.execute-api.us-east-
1.amazonaws.com/production/'

});



AWS.config.update({ region: 'us-east-1' });



const ddb = new AWS.DynamoDB.DocumentClient();
```

```
async function fetchConnectionId() {

    try {

        const params = {

            TableName: 'connectionTb2',

           ScanIndexForward: false, // Sort the items in descending
order

            Limit: 1 // Limit the result to only 1 item

        };


        const data = await ddb.scan(params).promise();


        if (data.Items.length === 0) {

            throw new Error('No connection ID found in the table');

        }


        return data.Items[0].connectionID;

    } catch (error) {

        console.error('Error fetching connection ID:', error);

        throw error;

    }

}
```

```
exports.handler = async (event) => {

    console.log('Received DynamoDB event:', JSON.stringify(event,
null, 2));


    try {

        const connectionID = await fetchConnectionId();


        for (const record of event.Records) {

            if (record.eventName === 'INSERT') {

                                                            const     newItem     =
AWS.DynamoDB.Converter.unmarshall(record.dynamodb.NewImage);

                console.log('New Item Inserted:', newItem);


                // Send the newItem to the WebSocket

                await sendToWebSocket(newItem, connectionID);

            } else if (record.eventName === 'MODIFY') {

                                                        const     modifiedItem     =
AWS.DynamoDB.Converter.unmarshall(record.dynamodb.NewImage);

                console.log('Item Modified:', modifiedItem);


                // Send the modifiedItem to the WebSocket

                await sendToWebSocket(modifiedItem, connectionID);

            } else if (record.eventName === 'REMOVE') {

                                                          const     removedItem     =
AWS.DynamoDB.Converter.unmarshall(record.dynamodb.OldImage);
```

```
                console.log('Item Removed:', removedItem);


                // Send the removedItem to the WebSocket

                await sendToWebSocket(removedItem, connectionID);

            }

        }


        return {

            statusCode: 200,

            body: JSON.stringify('Event processed successfully'),

        };

    } catch (error) {

        console.error('Error processing DynamoDB event:', error);


        return {

            statusCode: 500,

            body: JSON.stringify('Error processing event'),

        };

    }

};


async function sendToWebSocket(data, connectionID) {

    const params = {
```



```
        ConnectionId: connectionID,

        Data: JSON.stringify(data),

    };


    try {

                                                                              await
apigatewaymanagementapi.postToConnection(params).promise();

        console.log('Message sent to WebSocket:', data);

    } catch (error) {

       console.error('Error sending message to WebSocket:', error);

        throw error;

    }

}
```

Figure 6.4.5: Lambda Function to Fetch Connection and Send Data to Mobile Application

The diagram illustrates an AWS Lambda function tasked with processing events from the energy consumption DynamoDB table. It is triggered specifically by inserting events occurring in the DynamoDB table. This Lambda function logs incoming DynamoDB events and processes each record within the event.

The connection DynamoDB table stores information regarding WebSocket connections from mobile applications, utilizing connection IDs for actively connected clients. This Lambda function interacts with the connection

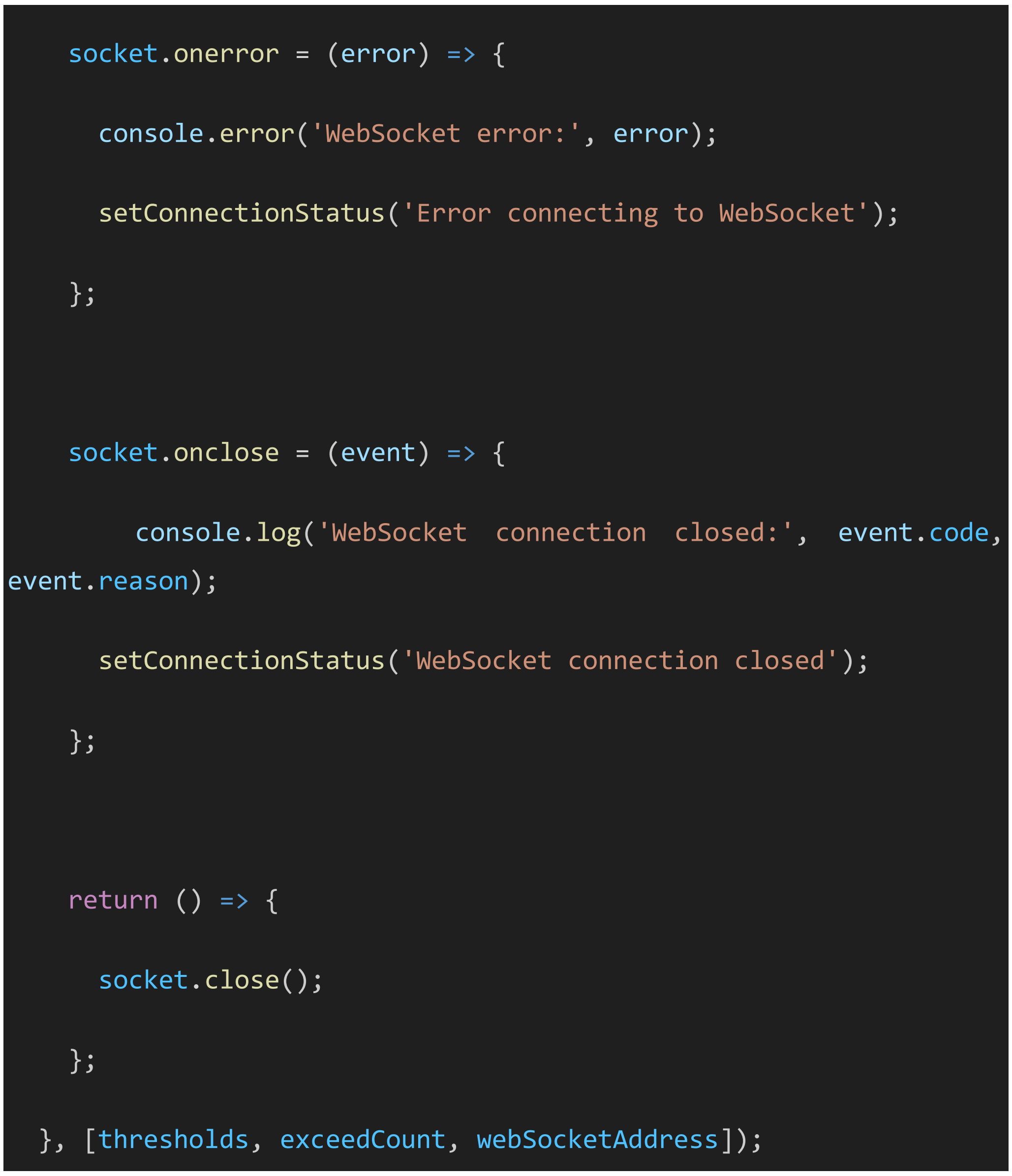
DynamoDB table through the AWS.DynamoDB.DocumentClient enables it to scan the table and retrieve connection IDs.

Amazon API Gateway's WebSocket feature facilitates bidirectional communication between this backend Lambda function and connected clients from mobile applications. The WebSocket URL is acquired from the API Gateway's 'stage', allowing the Lambda function to communicate with the WebSocket endpoint effectively.

Within the Lambda function, the exports.handler method is utilized to process DynamoDB events received as input from the energy consumption DynamoDB table. It iterates through each energy consumption record in the event, discerning the type of operation such as insert, modify, or remove. Subsequently, it extracts pertinent data from the DynamoDB event and dispatches it to connected WebSocket clients utilizing the sendToWebSocket function. Additionally, error-handling mechanisms are implemented to manage any errors encountered during processing or communication with WebSocket clients.

The sendToWebSocket function is responsible for crafting a message with data received from the energy consumption DynamoDB and transmitting it to the designated WebSocket connection ID. This is achieved using the postToConnection method of apigatewaymanagementapi, ensuring that clients subscribed to the WebSocket receive real-time updates based on changes in the DynamoDB table.

```
const HomeScreen = ({ navigation }) => {
      const      [connectionStatus,      setConnectionStatus]      =
useState('Connecting...');

  const [receivedData, setReceivedData] = useState([]);

  const [thresholds, setThresholds] = useState([]);

  const [exceedCount, setExceedCount] = useState(0);

  const [webSocketAddress, setWebSocketAddress] = useState(null);
// New state variable for WebSocket address



  useEffect(() => {

    const loadWebSocketAddress = async () => {

      try {

                              const       userData       =       await
Auth.currentAuthenticatedUser({ bypassCache: true });

        console.log('User attributes:', userData.attributes);

        const newTableName = userData.attributes['address']

        setWebSocketAddress(newTableName);

        console.log('WebSocket address:', newTableName);

      } catch (error) {

        console.log('Error getting user data', error);

      }

    };



    loadWebSocketAddress();

    const loadThresholds = async () => {
      try {
                         const     savedThresholds     =     await
AsyncStorage.getItem('thresholds');
        if (savedThresholds !== null) {
          setThresholds(JSON.parse(savedThresholds));
        }
      } catch (error) {
        console.error('Error loading thresholds: ', error);
      }
    };

    loadThresholds();
  }, []);

  useEffect(() => {
    if (!webSocketAddress) return; // If WebSocket address is not
loaded yet, do nothing

    const checkEnergyConsumption = (energyConsumption) => {
      thresholds.forEach((item) => {
        if (item.status && energyConsumption > item.threshold) {
          setExceedCount(prevCount => prevCount + 1);
          if (exceedCount >= 6000) {
            sendNotification();
          }

        }

      });

    };


     const  socket  =  new  WebSocket(webSocketAddress);  //  Use  the
WebSocket address from state


    socket.onopen = () => {

      console.log('WebSocket connection opened');

      setConnectionStatus('WebSocket connection opened');

    };


    socket.onmessage = (event) => {

      try {

        const data = JSON.parse(event.data);

        setReceivedData(prevData => [...prevData, data]);

        checkEnergyConsumption(data.energyConsumption);

      } catch (error) {

        console.error('Error parsing JSON data:', error);

      }

    };
```



```
    socket.onerror = (error) => {
      console.error('WebSocket error:', error);
      setConnectionStatus('Error connecting to WebSocket');
    };

    socket.onclose = (event) => {
      console.log('WebSocket connection closed:', event.code, event.reason);
      setConnectionStatus('WebSocket connection closed');
    };

    return () => {
      socket.close();
    };
  }, [thresholds, exceedCount, webSocketAddress]);
```

Figure 6.4.6: Code to Connect WebSocket API

This figure illustrates the method for handling WebSocket communication in a React Native environment. The WebSocket address is asynchronously fetched from AWS Cognito, which associates specific residents with their respective WebSocket endpoints. Upon retrieval of the WebSocket address, the code proceeds to establish a WebSocket connection.

Event handlers such as onopen, onmessage, onerror, and onclose are utilized to manage WebSocket communications effectively. When the

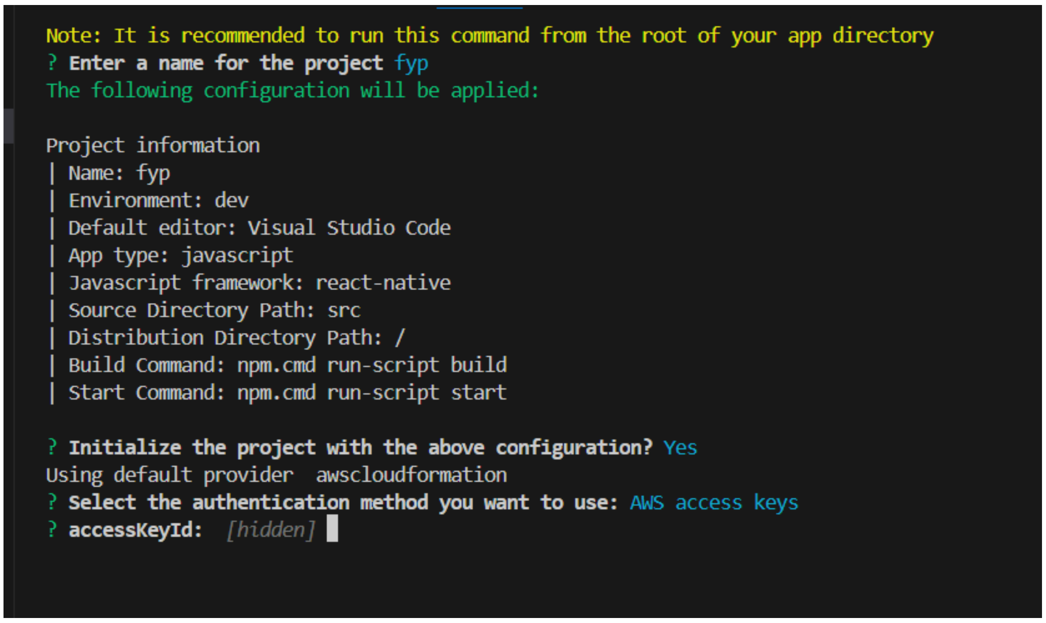
WebSocket connection is successfully opened (onopen), the application updates its status accordingly. Messages received from the WebSocket (onmessage) are parsed as JSON data and displayed within the mobile application interface.

In case of any errors (onerror) during WebSocket communication, appropriate error handling mechanisms are implemented to provide feedback to the user or log the error for debugging purposes. Similarly, when the WebSocket connection is closed (onclose), the application updates its status to reflect the closure of the connection.

This approach ensures seamless and real-time communication between the mobile application and the backend server via WebSocket, providing residents with up-to-date information and enabling interactive experiences within the application.

6.5 Perform authentication using AWS Amplify, AWS Cognito

6.5.1 Setup AWS Account and Install AWS Amplify CLI

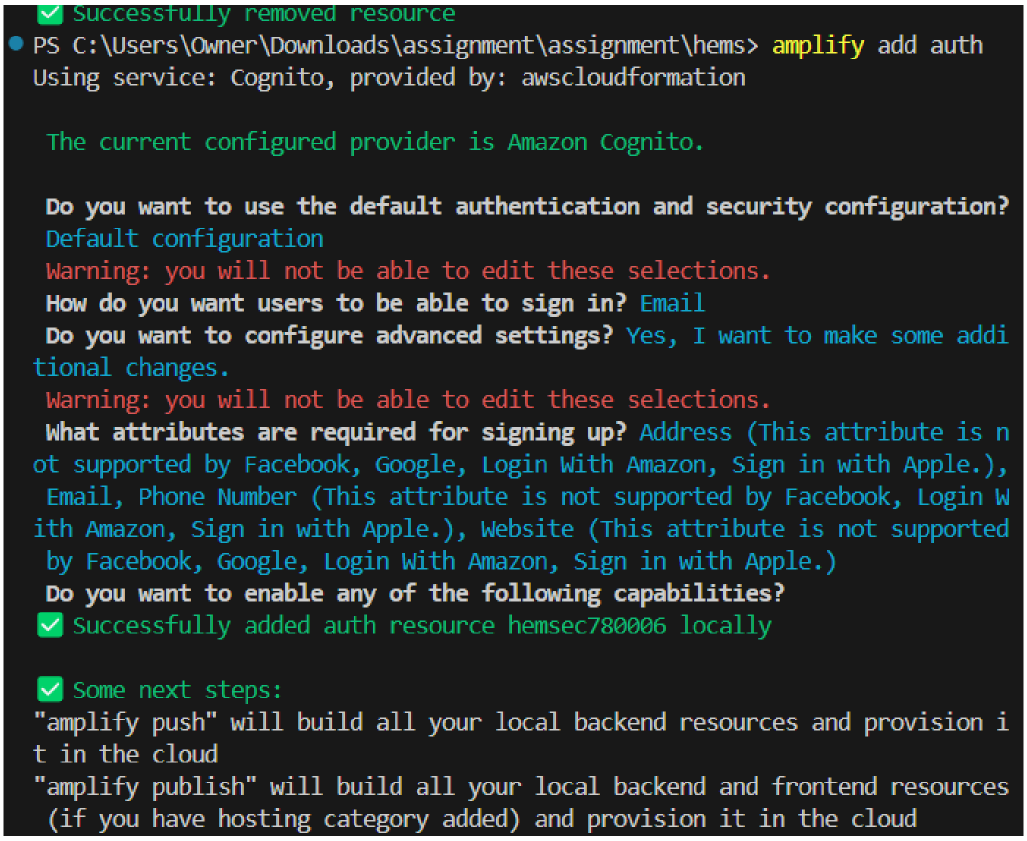


```
Note: It is recommended to run this command from the root of your app directory
? Enter a name for the project fyp
The following configuration will be applied:

Project information
| Name: fyp
| Environment: dev
| Default editor: Visual Studio Code
| App type: javascript
| Javascript framework: react-native
| Source Directory Path: src
| Distribution Directory Path: /
| Build Command: npm.cmd run-script build
| Start Command: npm.cmd run-script start

? Initialize the project with the above configuration? Yes
Using default provider  awscloudformation
? Select the authentication method you want to use: AWS access keys
? accessKeyId:  [hidden]
```

Figure 6.5.1: AWS Amplify Setup in React Native

As shown in the figure, an AWS account is first created to enable access to AWS services so that the AWS Amplify CLI is integrated into the React Native project to facilitate seamless integration with AWS services. Initialization of Amplify within the project is then carried out using the 'amplify init' command to establish the project's configuration. Additionally, an IAM user role is established, equipped with access keys (access key ID and secret access key) to enable secure programmatic access to various AWS services such as AWS Lambda, AWS Cognito and AWS Amplify. This setup ensures that the React Native project can interact with AWS resources effectively and securely. The access key ID and secret access key are inserted into amplify for this project using the command line when amplify is initiated.

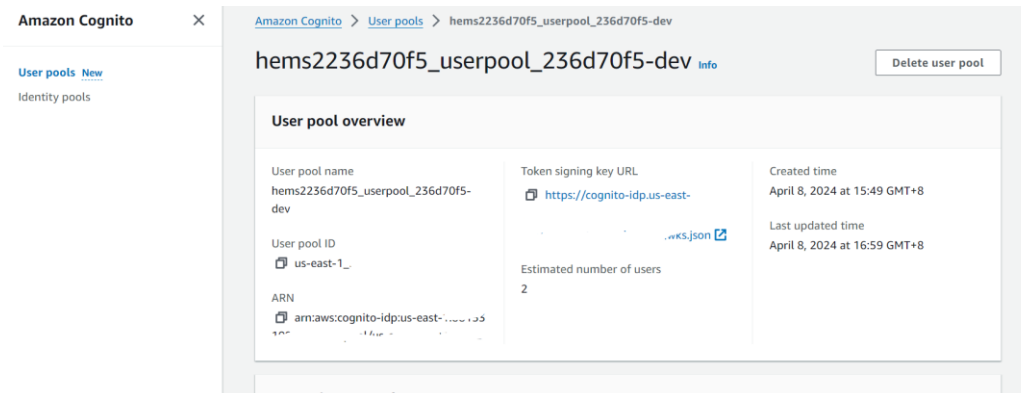
6.5.2 Create AWS Cognito User Pool and Identity Pool



```
 ✅ Successfully removed resource
● PS C:\Users\Owner\Downloads\assignment\assignment\hems> amplify add auth
  Using service: Cognito, provided by: awscloudformation

   The current configured provider is Amazon Cognito.

   Do you want to use the default authentication and security configuration?
   Default configuration
   Warning: you will not be able to edit these selections.
   How do you want users to be able to sign in? Email
   Do you want to configure advanced settings? Yes, I want to make some addi
  tional changes.
   Warning: you will not be able to edit these selections.
   What attributes are required for signing up? Address (This attribute is n
  ot supported by Facebook, Google, Login With Amazon, Sign in with Apple.),
   Email, Phone Number (This attribute is not supported by Facebook, Login W
  ith Amazon, Sign in with Apple.), Website (This attribute is not supported
   by Facebook, Google, Login With Amazon, Sign in with Apple.)
   Do you want to enable any of the following capabilities?
 ✅ Successfully added auth resource hemsec780006 locally

 ✅ Some next steps:
  "amplify push" will build all your local backend resources and provision i
  t in the cloud
  "amplify publish" will build all your local backend and frontend resources
   (if you have hosting category added) and provision it in the cloud
```

Figure 6.5.2: Add Authentication in AWS Amplify

In this figure, the AWS Amplify CLI was used to initialize the project named 'fyp', with the environment set to 'dev' and the default editor configured as Visual Studio Code. The project, developed in JavaScript using the React Native framework, specified the source directory path as 'src'. During initialization, the application was configured to use AWS Cognito as the authentication service. The default authentication method selected was email-based sign-in, with advanced settings configured to include required attributes for signing up, such as Address, Email, Phone Number, and Website. Subsequently, the authentication resource was successfully added locally to the project, paving the way for further development and deployment, such as the sign-in function and sign-up functions.


Amazon Cognito ✕

User pools New

Identity pools

Amazon Cognito > User pools > hems2236d70f5_userpool_236d70f5-dev

hems2236d70f5_userpool_236d70f5-dev Info

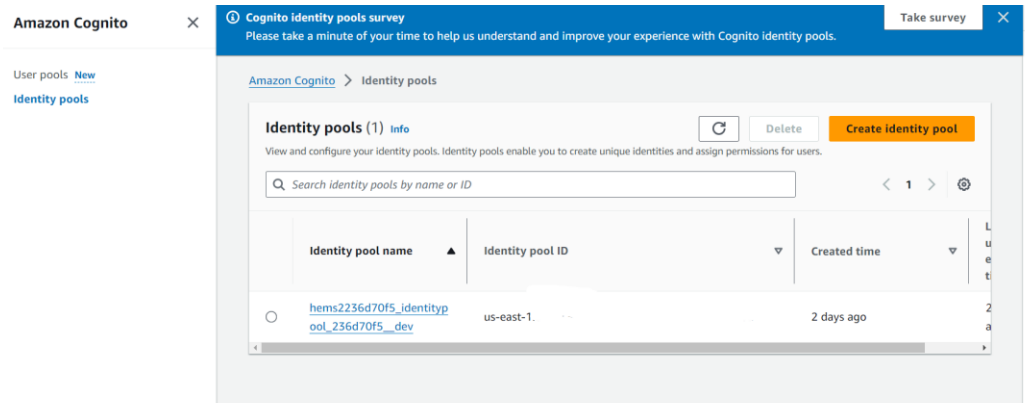
Delete user pool

User pool overview

User pool name	Token signing key URL	Created time
hems2236d70f5_userpool_236d70f5-dev	https://cognito-idp.us-east- ...wks.json	April 8, 2024 at 15:49 GMT+8
User pool ID		Last updated time
us-east-1_		April 8, 2024 at 16:59 GMT+8
ARN	Estimated number of users	
arn:aws:cognito-idp:us-east-	2	

Figure 6.5.3: AWS Cognito User Pool

The figure above illustrates the user pool created using the ‘amplify add auth’ command inside the project. A user pool in AWS Cognito represents a user directory that stores user attributes and credentials, allowing for user authentication and management. Different authentication settings such as sign-in methods., multi-factor authentication and password policy can be defined within the created user pool. This user pool acts as the central repository for user authentication within this project to manage user identities.


Amazon Cognito ✕

User pools New

Identity pools

Cognito identity pools survey

Please take a minute of your time to help us understand and improve your experience with Cognito identity pools.

Take survey ✕

Amazon Cognito > Identity pools

Identity pools (1) Info

Delete | Create identity pool

View and configure your identity pools. Identity pools enable you to create unique identities and assign permissions for users.

Search identity pools by name or ID

	Identity pool name	Identity pool ID	Created time
○	hems2236d70f5_identitypool_236d70f5__dev	us-east-1.	2 days ago

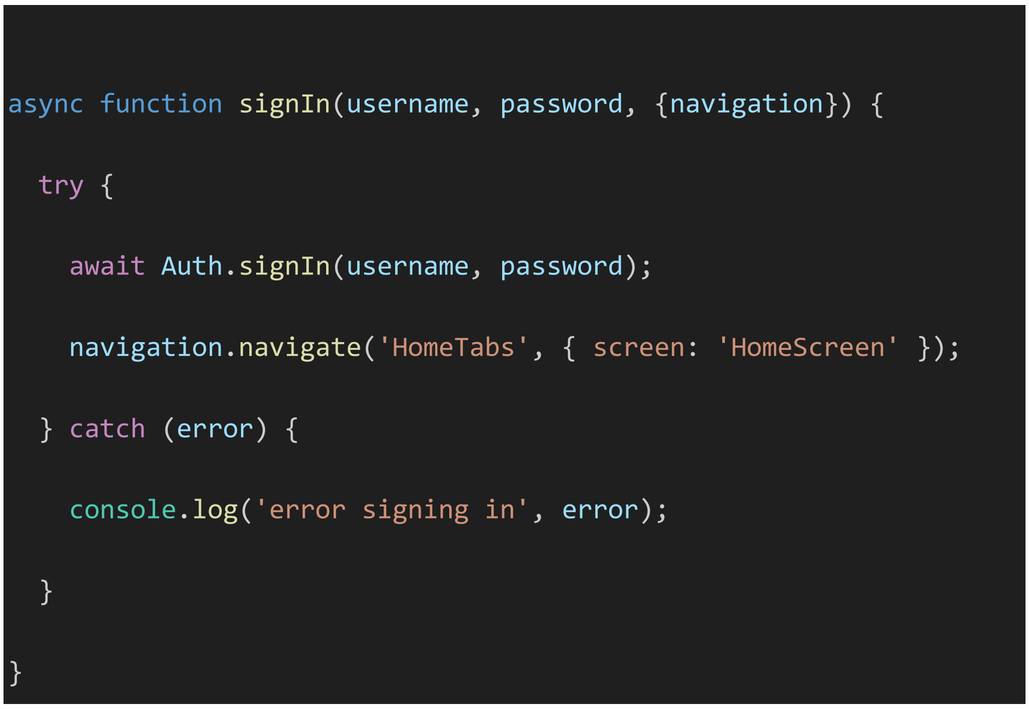
Figure 6.5.4: AWS Cognito Identity Pool

This figure shows the identify pool created associated with the user pool. An identity pool in AWS Cognito enables the application to grant temporary AWS credentials to users, allowing them to access AWS services securely. The integration between user authentication and access to AWS resources, such as DynamoDB tables and S3 buckets, can be established by associating these identity pools.

The authentication screens, such as sign-up and sign-in screens, are created using React Native components and Amplify's Auth module to handle authentication operations.

```
async function signUpAWS(email, username, password,name) {
  try {
    const { user } = await Auth.signUp({
      username,
      password,
      attributes: {
        email,


      },
      autoSignIn: {
        enabled: false,
      }
    });
    console.log(user);
  } catch (error) {
    console.log('error signing up:', error);
  }
}
```

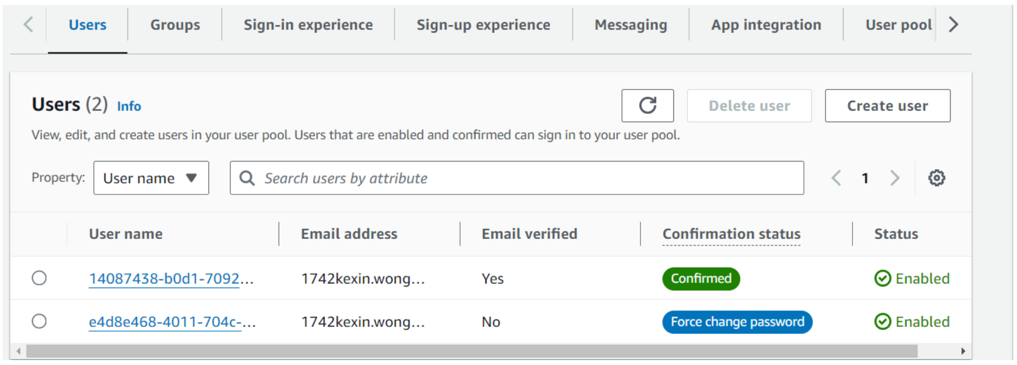
Figure 6.5.5: Code to Implement SignUp Function in React Native



```
async function signIn(username, password, {navigation}) {

  try {

    await Auth.signIn(username, password);

    navigation.navigate('HomeTabs', { screen: 'HomeScreen' });

  } catch (error) {

    console.log('error signing in', error);

  }

}
```

Figure 6.5.6: Code to Implement Sign In Function in React Native

The 'Auth.signIn' and 'Auth.signUp' functions from the AWS Amplify authentication module are vital components in managing user authentication and registration processes securely and efficiently within the application. 'Auth.signUp' serves the purpose of registering a new user within the system using parameters such as email and password for user registration. Amplify then communicates with the configured authentication service, AWS Cognito, to create a new user account. Additionally, users are prompted to verify their email through a link sent by AWS Simple Email Service (SES) for added security. The user needs to enter the verification code inside mobile application Confirmation screen to verify his email before login. On the other hand, 'Auth.signIn' facilitates the initiation of the sign-in process for users by necessitating the provision of email and password as initial parameters for user input. Subsequently, AWSAmplify communicates with AWS Cognito to authenticate the user's credentials. Upon successful authentication, the user is granted access to the application and is redirected to the home screen. In case of authentication failure, an error message is generated to notify user the error. This comprehensive approach ensures a seamless and secure user authentication and registration experience using AWS Cognito and AWS Amplify and adhering to best practices in mobile application development.



Users (2)

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name

Search users by attribute

User name	Email address	Email verified	Confirmation status	Status
14087438-b0d1-7092...	1742kexin.wong...	Yes	Confirmed	Enabled
e4d8e468-4011-704c-...	1742kexin.wong...	No	Force change password	Enabled

Figure 6.5.7: Created Users in AWS Cognito User Pool

The figure illustrates the users within a user pool in AWS Cognito, a service for managing user identities and authentication in the cloud. In AWS Cognito, user accounts can have different confirmation statuses, which provide insights into the authentication status of each user. These confirmation statuses typically include "Not Verified," "Confirmed," and "Force change password". When a user signs up for an account in AWS Cognito, their confirmation status initially appears as "Not Verified". Once a user verifies their email address or phone number by entering a confirmation code, their confirmation status changes to "Confirmed". The "Force change password" status indicates that the user is required to change their password before they can fully access the application or system.

6.6 Retrieve Historical Energy Consumption Data to Web-based Dashboard and Mobile Application

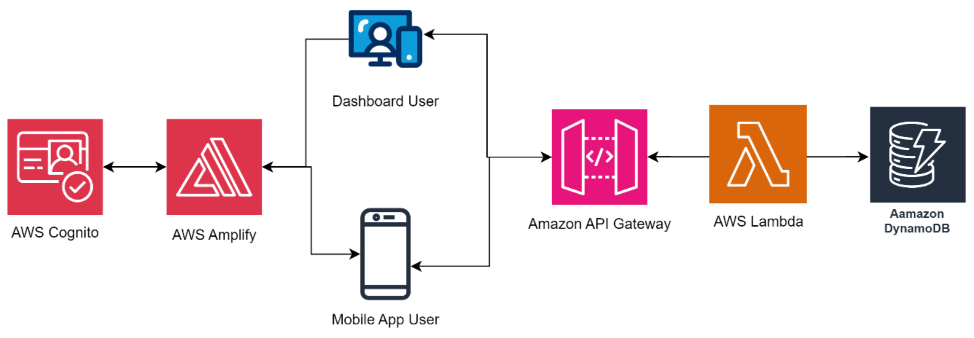


Figure 6.6.1: Retrieve Historical Energy Consumption Data to Web-based Dashboard and Mobile Application Workflow

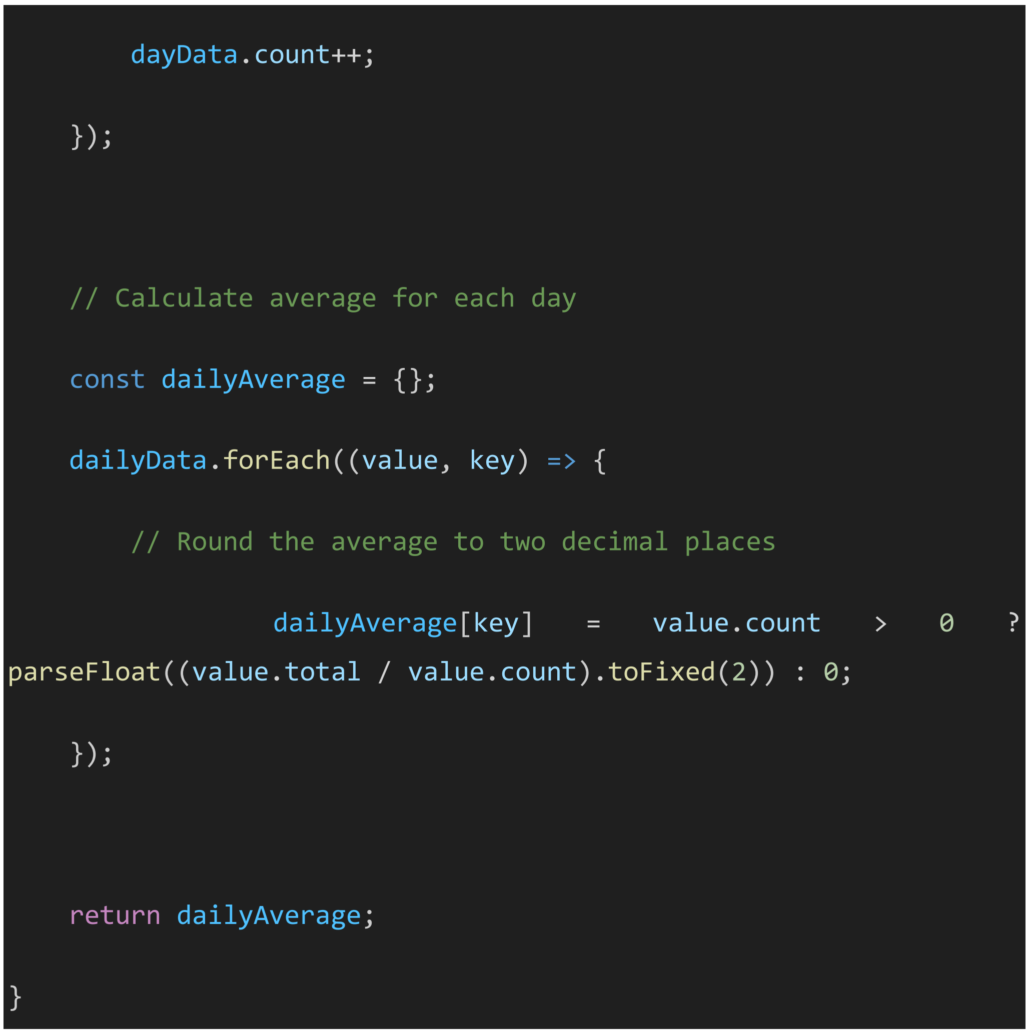
This architecture is designed to retrieve historical data from an Amazon DynamoDB table, such as daily, weekly, and monthly energy consumption data. An HTTP API is created using Amazon API Gateway to facilitate the passing of processed data. AWS Lambda functions are utilized to retrieve specific period data based on user requests via mobile and web applications, subsequently sending the requested data to users. This setup ensures efficient retrieval and delivery of data to users based on their requirements. The methods used for both web dashboard and mobile applications are almost the same, with minor modifications due to different configurations of different platforms.

6.6.1 Retrieve Past Week Energy Consumption Data

The provided code is an AWS Lambda function written in Node.js designed to retrieve energy consumption data from an energy consumption DynamoDB table for the past week, calculate the average apparent power for each day, and return this processed data in response to an HTTP request. This AWS Lambda function will integrate with the route of an HTTP API using a POST operation to receive resident data that specifies the name of the energy consumption table from which to retrieve data.

```
const AWS = require('aws-sdk');


// Create DynamoDB Document Client
const docClient = new AWS.DynamoDB.DocumentClient();


exports.handler = async (event) => {
    try {
        // Parse the event body to extract the tableName
        const requestBody = JSON.parse(event.body);
        const tableName = requestBody.tableName;


        // Check if the tableName is provided
        if (!tableName) {
           throw new Error('Missing required parameter: TableName');
        }


        // Calculate the time threshold in milliseconds
        const oneWeekAgo = new Date(Date.now() - 7 * 24 * 60 * 60
* 1000).toISOString(); // One week ago
```

```
        // Define the params for the DynamoDB query to retrieve the
data for the past week
        const params = {
             TableName: tableName, // Use the table name from the
event payload
                 KeyConditionExpression:  "Device_ID  =  :d_id  and
#ts > :oneWeekAgo",
             ProjectionExpression: '#ts, ApparentPower', // Use an
alias for Timestamp using ExpressionAttributeNames
            ExpressionAttributeNames: {
                "#ts": "Timestamp" // Alias for Timestamp
            },
            ExpressionAttributeValues: {
                ":oneWeekAgo": oneWeekAgo,
                  ":d_id": 1, // Assuming Device_ID is a numeric
attribute, adjust if it's not
            }
        };

        // Query DynamoDB to retrieve the data for the past week
        const data = await docClient.query(params).promise();

        console.log('Retrieved data:', data);
        // Process the data and calculate the average apparent
power for each day

        const dailyAverage = calculateDailyAverage(data.Items);


        return {

            statusCode: 200,

            body: JSON.stringify({ dailyAverage })

        };

    } catch (error) {

        console.error('Error:', error);

        return {

            statusCode: 500,

            body: JSON.stringify({ message: error.message ||
'Internal server error' }) // Return error message if available

        };

    }

};


// Function to calculate the average apparent power for each day

function calculateDailyAverage(items) {

    // Create a map to store apparent power values for each day

    const dailyData = new Map();


    // Initialize the date for one week ago
    let currentDate = new Date();

    currentDate.setDate(currentDate.getDate() - 7);



    // Iterate through the past week

    for (let i = 0; i < 7; i++) {

        const dateString = currentDate.toISOString().substr(0, 10);

        dailyData.set(dateString, { total: 0, count: 0 });

        currentDate.setDate(currentDate.getDate() + 1);

    }



    // Aggregate apparent power values for each day from retrieved
items

    items.forEach(item => {

        const timestamp = item.Timestamp.substr(0, 10); // Extract
YYYY-MM-DD part of timestamp

        const apparentPower = item.ApparentPower || 0; // If
apparent power is undefined, default to 0



        if (!dailyData.has(timestamp)) {

            dailyData.set(timestamp, { total: 0, count: 0 });

        }



        const dayData = dailyData.get(timestamp);

        dayData.total += apparentPower;
```



```
        dayData.count++;
    });


    // Calculate average for each day
    const dailyAverage = {};
    dailyData.forEach((value, key) => {
        // Round the average to two decimal places
        dailyAverage[key] = value.count > 0 ? parseFloat((value.total / value.count).toFixed(2)) : 0;
    });


    return dailyAverage;
}
```

Figure 6.6.2: Lambda Function to Retrieve Historical Data

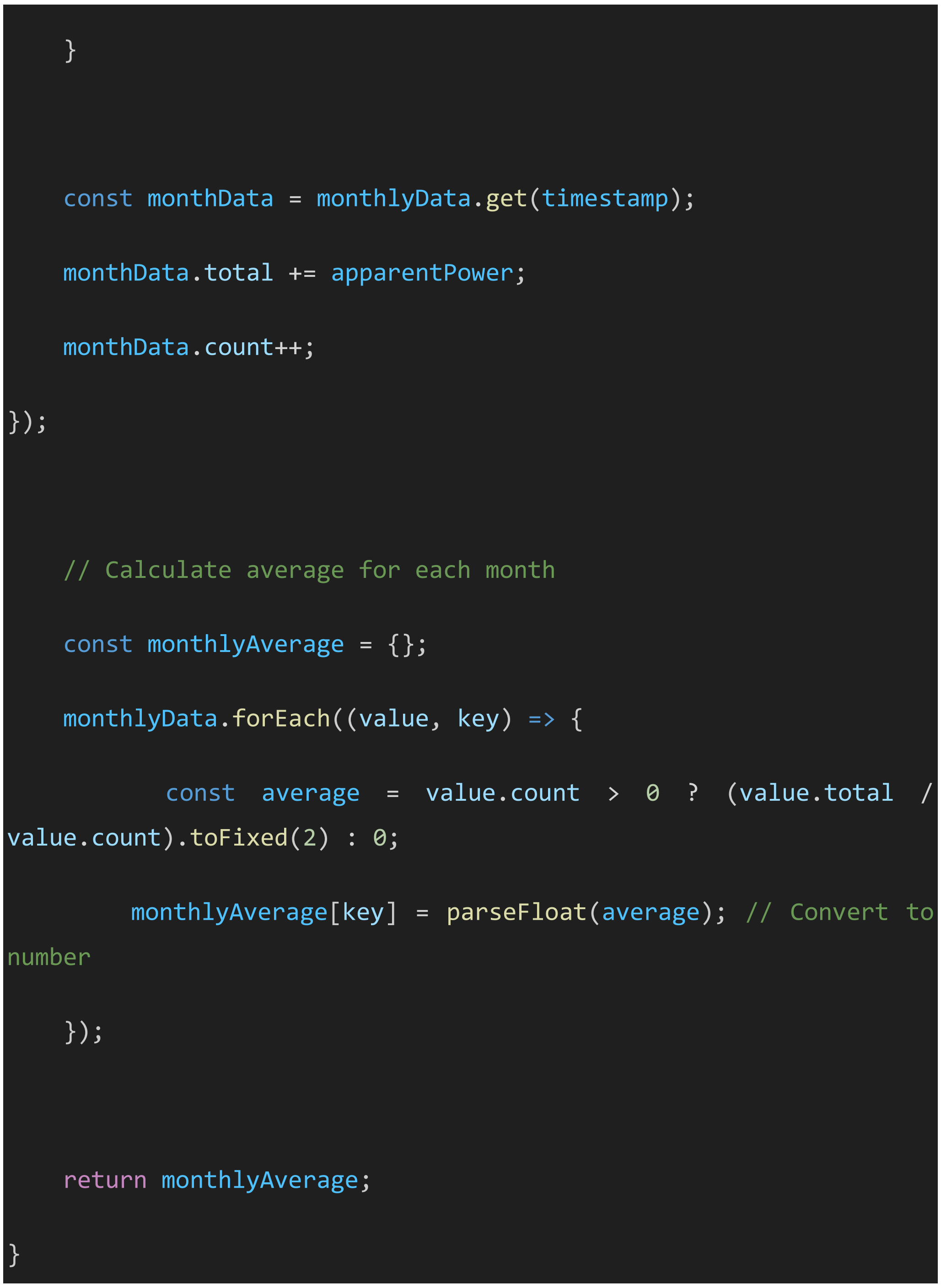
The purpose of this AWS Lambda function is to compute the daily average apparent power by retrieving energy consumption information from an Amazon DynamoDB table for the previous week. It builds DynamoDB query parameters upon receiving an HTTP request, parses the request body to identify the table name, and then runs the query to get the specific data. It then uses the calculateDailyAverage() function to evaluate the collected data and determine the daily average apparent power for each day during the previous week. Error messages and the proper status codes are returned in response to errors. Lastly, if the processing of the data is successful, a 200 status code is returned in the response body; if an error occurs, a 500 status code is returned.

6.6.2 Retrieve Past 6 Months Energy Consumption Data

The code provided constitutes an AWS Lambda function crafted in Node.js. Its primary function is to extract energy consumption data from a DynamoDB table for the preceding six months, compute the average apparent power for each day, and furnish this processed information in response to an HTTP request. This Lambda function is engineered to seamlessly integrate with an HTTP API route, employing a POST operation to acquire resident data. This data includes specifics regarding the designated energy consumption table.

```
const AWS = require('aws-sdk');


// Create DynamoDB Document Client

const docClient = new AWS.DynamoDB.DocumentClient();


exports.handler = async (event) => {

    try {

        const requestBody = JSON.parse(event.body);

        const tableName = requestBody.tableName;


        // Check if the tableName is provided

        if (!tableName) {

           throw new Error('Missing required parameter: TableName');

        }


// Get the current date

const currentDate = new Date();
```

```
// Set the date to the first day of the current month
const firstDayOfMonth = new Date(currentDate.getFullYear(),
currentDate.getMonth(), 1);


// Set the date to six months ago from the first day of the current
month
const sixMonthsAgo = new Date(firstDayOfMonth);
sixMonthsAgo.setMonth(sixMonthsAgo.getMonth() - 6);


// Set hours, minutes, seconds, and milliseconds to the minimum
sixMonthsAgo.setHours(0, 0, 0, 0);


// Set the end date to the last millisecond of the current month
const lastDayOfMonth = new Date(currentDate.getFullYear(),
currentDate.getMonth() + 1, 0);
lastDayOfMonth.setHours(23, 59, 59, 999);


// Convert the dates to ISO string
const localISOTimeStart = sixMonthsAgo.toISOString();
const localISOTimeEnd = lastDayOfMonth.toISOString(); // Use the
last millisecond of the current month as the end date


const params = {
    TableName: tableName,
       KeyConditionExpression:   "Device_ID   =   :d_id   and   #ts
BETWEEN :sixMonthsAgo AND :currentDate",

    ProjectionExpression: '#ts, ApparentPower',

    ExpressionAttributeNames: {

        "#ts": "Timestamp"

    },

    ExpressionAttributeValues: {

        ":sixMonthsAgo":  localISOTimeStart,

        ":currentDate": localISOTimeEnd,

        ":d_id": 1

    }

};



         // Query DynamoDB to retrieve the data for the past six
months

        const data = await docClient.query(params).promise();


        console.log('Retrieved data:', data);


         // Process the data and calculate the average apparent
power for each month

       const monthlyAverage = calculateMonthlyAverage(data.Items);


        return {
            statusCode: 200,

            body: JSON.stringify({ monthlyAverage })

        };

    } catch (error) {

        console.error('Error:', error);

        return {

            statusCode: 500,

                body:  JSON.stringify({  message:  'Internal  server
error' })

        };

    }

};



// Function to calculate the average apparent power for each month

function calculateMonthlyAverage(items) {

    // Create a map to store apparent power values for each month

    const monthlyData = new Map();



    // Initialize the date for six months ago

    let currentDate = new Date();

     currentDate.setMonth(currentDate.getMonth() - 5); // Subtract
5 months to get 6 months ago



    // Iterate through the latest six months
    for (let i = 0; i < 6; i++) {
        const year = currentDate.getFullYear();
        const month = currentDate.getMonth() + 1; // Month index starts from 0, so add 1
        const monthKey = `${year}-${month < 10 ? '0' + month : month}`; // Format month key as YYYY-MM
        monthlyData.set(monthKey, { total: 0, count: 0 });
        currentDate.setMonth(currentDate.getMonth() + 1); // Move to the next month
    }

   items.forEach(item => {
    // Extract the year and month from the Timestamp
    const date = new Date(item.Timestamp);
    const year = date.getFullYear();
    const month = date.getMonth() + 1; // Month index starts from 0, so add 1
    const timestamp = `${year}-${month < 10 ? '0' + month : month}`; // Format as YYYY-MM

    const apparentPower = item.ApparentPower || 0; // If apparent power is undefined, default to 0

    if (!monthlyData.has(timestamp)) {
        monthlyData.set(timestamp, { total: 0, count: 0 });
```



```
    }


    const monthData = monthlyData.get(timestamp);

    monthData.total += apparentPower;

    monthData.count++;

});



    // Calculate average for each month

    const monthlyAverage = {};

    monthlyData.forEach((value, key) => {

            const  average  =  value.count  >  0  ?  (value.total  /
value.count).toFixed(2) : 0;

         monthlyAverage[key] = parseFloat(average); // Convert to
number

    });


    return monthlyAverage;

}
```

Figure 6.6.3: Lambda Function to Retrieve Past Six Month Data

The Node.js-written AWS Lambda function is used to get energy usage information from a DynamoDB table for the previous six months. It uses the processed data to reply to HTTP requests and calculates the average apparent power for each month. It computes the start and end dates for the six months based on the current date after scanning the request body to retrieve the table name. It builds DynamoDB query parameters using these dates and then runs the query to retrieve the pertinent data. After the data is retrieved, the function uses a map to aggregate the results for each month in order to calculate the monthly average apparent power. Error management techniques are integrated to provide relevant error messages and status codes. Once data processing is successful, the response body's computed monthly average apparent power is returned with a 200 status code; if an error occurs, on the other hand, a 500 status code is returned.

6.6.3 Calculate Monthly Peak Usage Hours and Total Carbon Emissions

The code provided constitutes an AWS Lambda function crafted in Node.js. Its primary function is to extract energy consumption data from a DynamoDB table for the preceding six months, compute the average apparent power for each day, furnish this processed information in response to an HTTP request and calculate monthly peak usage hours and total carbon emissions. This Lambda function is engineered to seamlessly integrate with an HTTP API route, employing a POST operation to acquire resident data. This data includes specifics regarding the designated energy consumption table.

```
const AWS = require('aws-sdk');


// Create DynamoDB Document Client

const docClient = new AWS.DynamoDB.DocumentClient();


exports.handler = async (event) => {

    try {

       // Parse the event body to extract the tableName
```

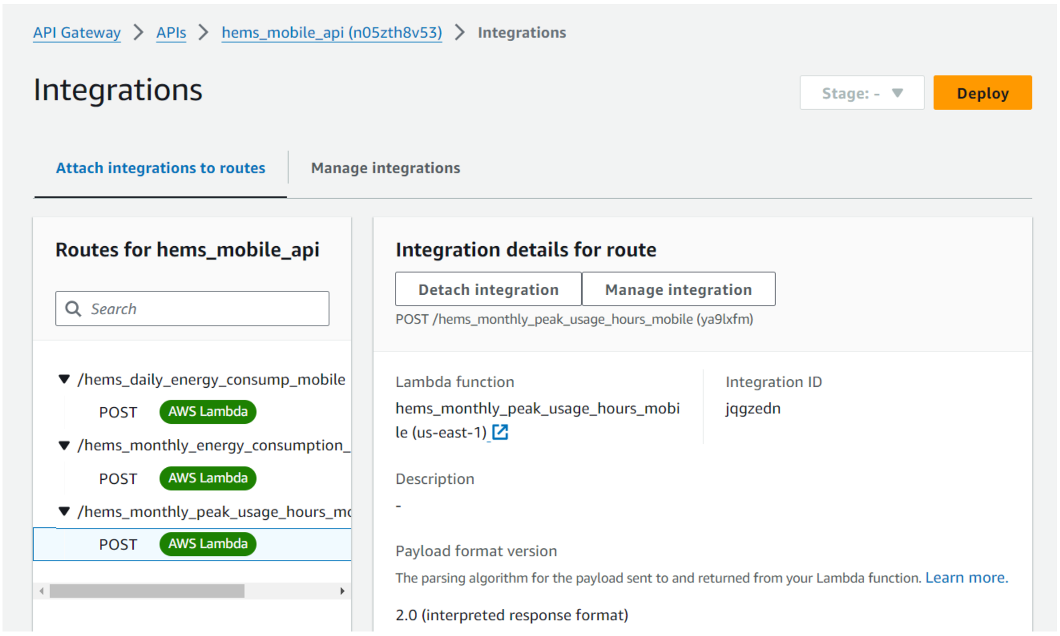
```
        const requestBody = JSON.parse(event.body);

        const tableName = requestBody.tableName;



        // Check if the tableName is provided

        if (!tableName) {

           throw new Error('Missing required parameter: TableName');

        }



        // Calculate the time threshold for six months ago

        const sixMonthsAgo = new Date();

          sixMonthsAgo.setMonth(sixMonthsAgo.getMonth()  -  5);  //
Subtract 5 months to get 6 months ago

          const  sixMonthsAgoISO  =  sixMonthsAgo.toISOString();  //
Convert to ISO format



        // Define the params for the DynamoDB query to retrieve the
data for the past six months

        const params = {

            TableName: tableName,

                 KeyConditionExpression:  "Device_ID  =  :d_id  and
#ts > :sixMonthsAgo",

            ProjectionExpression: '#ts, ApparentPower', // Use an
alias for Timestamp using ExpressionAttributeNames

            ExpressionAttributeNames: {

                "#ts": "Timestamp" // Alias for Timestamp
            },

            ExpressionAttributeValues: {

                ":sixMonthsAgo": sixMonthsAgoISO,

                  ":d_id": 1, // Assuming Device_ID is a numeric
attribute, adjust if it's not

            }

        };



        // Query DynamoDB to retrieve the data for the past six
months

        const data = await docClient.query(params).promise();



        console.log('Retrieved data:', data);



        // Process the data and calculate the peak usage hours
period for each month

                                const       peakUsageHours       =
calculatePeakUsagePeriodPerDay(data.Items);



        // Calculate the period with the most votes among the peak
usage periods

                             const      mostFrequentPeriod      =
calculateMostFrequentPeakUsagePeriod(peakUsageHours);



        // Calculate total carbon emissions
```

```
                            const      totalCarbonEmissions      =
calculateTotalCarbonEmissions(data.Items);


        return {
            statusCode: 200,
                     body:  JSON.stringify({  mostFrequentPeriod,
totalCarbonEmissions })
        };
    } catch (error) {
        console.error('Error:', error);
        return {
            statusCode: 500,
                body:  JSON.stringify({  message:  'Internal  server
error' })
        };
    }
};


// Function to calculate the period of peak energy consumption for
each day
function calculatePeakUsagePeriodPerDay(items) {
    const peakUsagePerDay = {};


    // Iterate through the items
    items.forEach(item => {
        const timestamp = new Date(item.Timestamp);

        const dateKey = timestamp.toISOString().substr(0, 10); //
Extract YYYY-MM-DD part of timestamp



        // Initialize peak usage data for the day if not already
present

        if (!peakUsagePerDay[dateKey]) {

            peakUsagePerDay[dateKey] = {

                peakPeriod: null,

                peakEnergy: -Infinity

            };

        }



        const energyConsumption = item.ApparentPower || 0; // If
apparent power is undefined, default to 0



        // Update peak usage data if current period has higher
energy consumption

        if (energyConsumption > peakUsagePerDay[dateKey].peakEnergy)
{

            peakUsagePerDay[dateKey] = {

                peakPeriod: getPeriodOfDay(timestamp.getHours()),
// Determine the period of the day

                peakEnergy: energyConsumption

            };
        }

    });


    return peakUsagePerDay;

}


// Function to determine the period of the day based on the hour

function getPeriodOfDay(hour) {

    if (hour >= 5 && hour < 12) {

        return 'Morning';

    } else if (hour >= 12 && hour < 18) {

        return 'Afternoon';

    } else {

        return 'Evening';

    }

}


// Function to calculate the period with the most votes among the
peak usage periods for each day

function calculateMostFrequentPeakUsagePeriod(peakUsagePerDay) {

    const periodCounts = {

        Morning: 0,

        Afternoon: 0,
        Evening: 0

    };



    // Count occurrences of each period

    Object.values(peakUsagePerDay).forEach(({ peakPeriod }) => {

        periodCounts[peakPeriod]++;

    });



    // Find the period with the maximum count

    let maxCount = -Infinity;

    let mostFrequentPeriod = null;

    Object.entries(periodCounts).forEach(([period, count]) => {

        if (count > maxCount) {

            maxCount = count;

            mostFrequentPeriod = period;

        }

    });



    return mostFrequentPeriod;

}



function calculateTotalCarbonEmissions(items) {

    let totalCarbonEmissions = 0;
    const carbonEmissionsPerKWh = 0.5; // Hypothetical value for
carbon emissions per kWh


    items.forEach(item => {

        const energyConsumption = item.ApparentPower || 0; // If
apparent power is undefined, default to 0

        const carbonEmissions = energyConsumption *
carbonEmissionsPerKWh; // Calculate carbon emissions for this item

        totalCarbonEmissions += carbonEmissions; // Add to the
total carbon emissions

    });


    // Round up total carbon emissions to nearest whole number

    totalCarbonEmissions = Math.round(totalCarbonEmissions);


    return totalCarbonEmissions;

}
```

Figure 6.6.4: Calculate Peak Month Usage

This Node.js-based AWS Lambda code retrieves energy consumption information over a six-month period from a DynamoDB table. It computes the monthly average apparent power and returns the processed data in response to HTTP queries. Next, it parses the request body to retrieve the table name, and then it uses the current date to calculate the start and end dates for the six-month period. It builds DynamoDB query parameters using these dates and then runs the query to obtain pertinent data. After retrieval, the function uses a map to aggregate values and calculates each day's peak energy use period. After that, it determines which of the high usage periods has received the most votes. Additionally, it calculates the total carbon emissions based on the energy consumption data. Error handling mechanisms are incorporated to furnish appropriate status codes and error messages. In case of successful data processing, a 200 status code accompanies the calculated most frequent peak usage period and total carbon emissions in the response body; conversely, a 500 status code is returned in the event of an error.

6.6.4 Detach Integration with HTTP Routes


API Gateway > APIs > hems_mobile_api (n05zth8v53) > Integrations

Integrations

Stage: - Deploy

Attach integrations to routes | Manage integrations

Routes for hems_mobile_api

Search

▼ /hems_daily_energy_consump_mobile
POST AWS Lambda
▼ /hems_monthly_energy_consumption_
POST AWS Lambda
▼ /hems_monthly_peak_usage_hours_mo
POST AWS Lambda

Integration details for route

Detach integration | Manage integration

POST /hems_monthly_peak_usage_hours_mobile (ya9lxfm)

Lambda function: hems_monthly_peak_usage_hours_mobile (us-east-1)

Integration ID: jqgzedn

Description: -

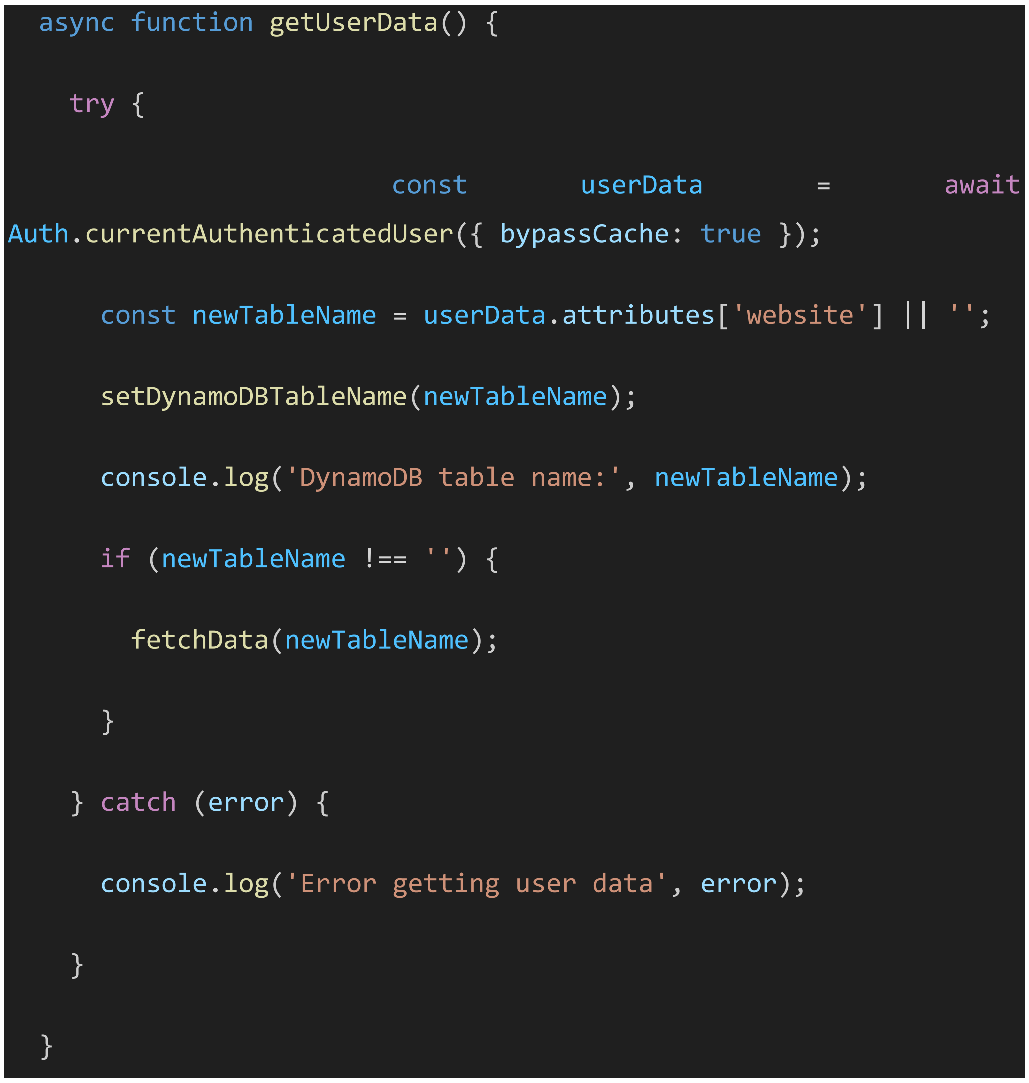
Payload format version: The parsing algorithm for the payload sent to and returned from your Lambda function. Learn more.

2.0 (interpreted response format)

Figure 6.6.5: HTTP Route with Lambda Function

This figure shows a system architecture in which POST operations on HTTP routes are not influenced by AWS Lambda functions. In this arrangement, AWS Lambda functions are separated from particular HTTP endpoints. Instead, HTTP routes manage incoming requests and responses, and they carry out background operations, data processing, or other functions. In order to create real-time bidirectional communication with the backend, the mobile application also establishes connections with WebSocket API routes. With the help of this WebSocket integration, the mobile application can quickly receive notifications or data updates, improving user experience with dynamic and responsive information.

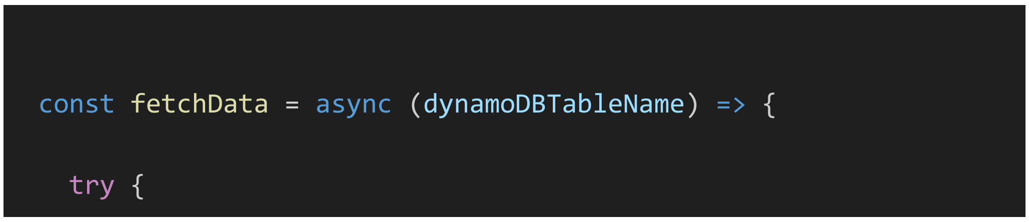
6.6.5 Connect HTTP API to Retrieve Energy Consumption Data



```
async function getUserData() {

  try {

                    const      userData       =       await
Auth.currentAuthenticatedUser({ bypassCache: true });

    const newTableName = userData.attributes['website'] || '';

    setDynamoDBTableName(newTableName);

    console.log('DynamoDB table name:', newTableName);

    if (newTableName !== '') {

      fetchData(newTableName);

    }

  } catch (error) {

    console.log('Error getting user data', error);

  }

}
```

Figure 6.6.6: Retrieve User Data from React Native

This code retrieved specific residents' data, such as the energy consumption table name, to be sent via HTTP API to the AWS Lambda function to retrieve energy consumption data for specific residents.After setting up HTTP API to retrieve energy consumption data and perform calculations, the React Native mobile application screen is configured to fetch the data required to build React Native components.



```
const fetchData = async (dynamoDBTableName) => {

  try {
    // Fetch weekly energy consumption

    try {

                      const      weeklyResponse      =      await
fetch('https://n05zth8v53.execute-api.us-east-
1.amazonaws.com/hems_daily_energy_consump_mobile', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json'

        },

        body: JSON.stringify({ tableName: dynamoDBTableName })

      });

      if (!weeklyResponse.ok) {

        throw new Error('Network response for weekly data was not
ok');

      }

      const weeklyJsonData = await weeklyResponse.json();

      if (weeklyJsonData && weeklyJsonData.dailyAverage) {

        setWeeklyData(weeklyJsonData.dailyAverage);

      } else {

            throw  new  Error('Weekly  data  is  null  or  missing
dailyAverage');

      }

    } catch (error) {

      console.error('Error fetching weekly data: ', error);

    }
```

```
   // Fetch monthly energy consumption

  const monthlyResponse = await fetch('https://n05zth8v53.execute-api.us-east-1.amazonaws.com/hems_monthly_energy_consumption_mobile', {

    method: 'POST',

    headers: {

      'Content-Type': 'application/json'

    },

    body: JSON.stringify({ tableName: dynamoDBTableName })

  });

  if (!monthlyResponse.ok) {

    throw new Error('Network response for monthly data was not ok');

  }

  const monthlyJsonData = await monthlyResponse.json();

  setMonthlyData(monthlyJsonData.monthlyAverage);

    // Fetch total carbon emissions

    const carbonResponse = await fetch('https://n05zth8v53.execute-api.us-east-1.amazonaws.com/hems_monthly_peak_usage_hours_mobile', {

      method: 'POST',

      headers: {

        'Content-Type': 'application/json'

      },

      body: JSON.stringify({ tableName: dynamoDBTableName })
    });
    if (!carbonResponse.ok) {
      throw new Error('Network response for total carbon emissions
was not ok');
    }
    const carbonJsonData = await carbonResponse.json();
    setMostFrequentPeriod(carbonJsonData.mostFrequentPeriod);
    setTotalCarbonEmissions(carbonJsonData.totalCarbonEmissions);
    setIsLoading(false);
    } catch (error) {
    console.error('There was a problem with the fetch operation:',
error);
    setIsLoading(false);
    }
  };

  const fetchMostFrequentPeriod = async () => {
    try {
      // Fetch most frequent period data
        const response = await fetch('https://n05zth8v53.execute-
api.us-east-1.amazonaws.com/hems_monthly_peak_usage_hours_mobile',
{
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({ tableName: dynamoDBTableName })
      });
      if (!response.ok) {
        throw new Error('Network response for most frequent period
data was not ok');
      }
      const jsonData = await response.json();
      setMostFrequentPeriod(jsonData.mostFrequentPeriod);
    } catch (error) {
     console.error('There was a problem with the fetch operation:',
error);
    }
  }
```

Figure 6.6.7: Fetch HTTP API Response

The goal of this React Native implementation is to leverage HTTP queries to retrieve statistics and data about energy use from AWS Lambda functions. The asynchronous fetchData method accepts an argument called dynamoDBTableName retrieved from AWS Cognito. It starts by attempting to submit a POST request to the designated AWS Lambda endpoint in order to retrieve weekly energy consumption data. The function retrieves the JSON data from the response and uses the setWeeklyData method to update the application state with the weekly energy consumption statistics if the response is successful, as indicated by weeklyResponse.ok. The relevant error messages are logged to the console in the event that any errors occur throughout the procedure, such as a network problem or missing data.

In a similar manner, the function uses different POST requests to the appropriate AWS Lambda APIs to retrieve statistics on monthly energy use and overall carbon emissions. Sending the request, verifying the answer, parsing the JSON data, and changing the application state are the steps that every retrieve operation consists of.

Additionally, the fetchMostFrequentPeriod function is responsible for fetching the most frequent period of peak energy usage. It uses the same protocol as fetchData to handle the result after executing a POST request to the specified AWS Lambda endpoint. The application state variable mostFrequentPeriod is updated following the successful retrieval of the most frequent period data.

6.7 Send Email Notification if Resident's Energy Consumption Data Exceed Threshold Defined

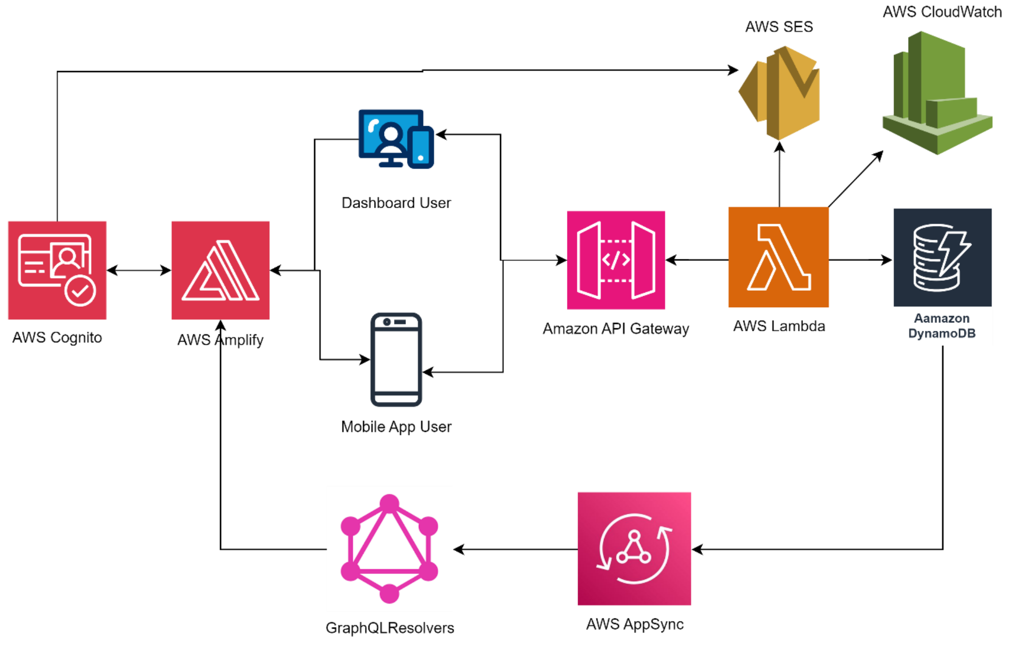


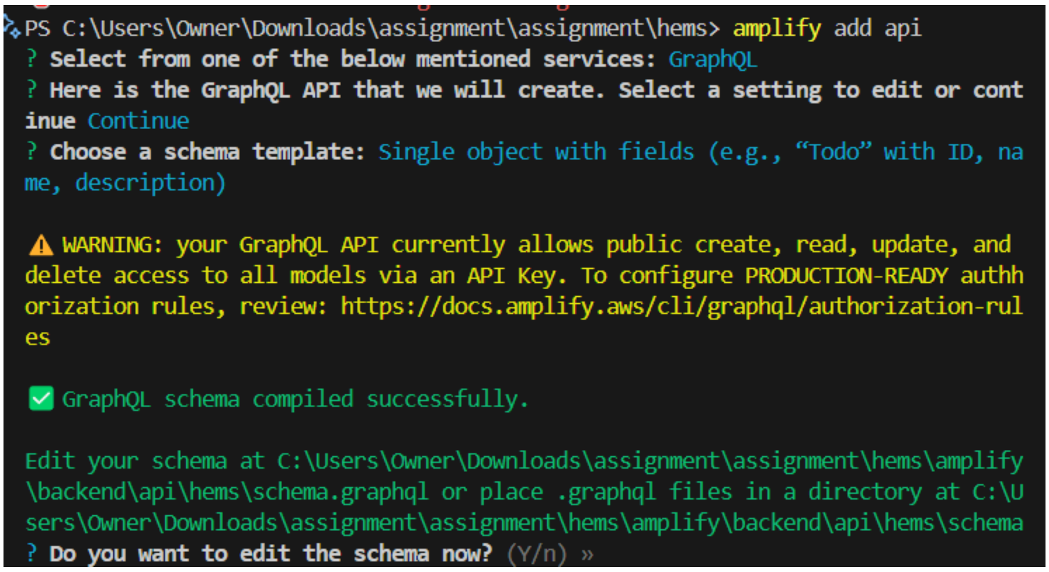
Figure 6.7.1: Send Email Notification if Resident's Energy Consumption Data Exceed Threshold Defined Workflow

As energy consumption data continuously inserts into the energy consumption DynamoDB table and considering that the mobile application may not always be connected to the WebSocket API, a solution utilizing AWS Lambda with DynamoDB is implemented to periodically detect energy consumption data. This Lambda function is tasked with monitoring the DynamoDB table for any threshold breaches and triggering email notifications to residents accordingly. AWS SES (Simple Email Service) is integrated into this solution to facilitate the sending of email notifications to residents. This approach circumvents the need for the mobile application to constantly connect to the WebSocket API and ensures efficient and reliable threshold detection without relying on the mobile application's connectivity. Additionally, leveraging AWS Lambda and DynamoDB for this task offers scalability and flexibility, while integrating AWS SES streamlines the process of sending email notifications. This approach is particularly advantageous given that the mobile application is developed using Expo for both IOS and Android platforms, where configuring push notifications can be complex and platform-specific. By centralizing the threshold detection and notification process within AWS Lambda and DynamoDB, the solution achieves simplicity, reliability, and scalability, enhancing the overall effectiveness of energy consumption monitoring and notification for residents.

6.7.1 Create Threshold DynamoDB Table

A threshold DynamoDB table has been established to manage various thresholds designated by individual residents, encompassing attributes such as "id" (String), "__typename", "createdAt", "email" (for sending notifications to the corresponding email addresses), "table" (representing the energy consumption table name retrieved from AWS Cognito), "threshold", "timeInterval" (determining which historical data period to examine), "updatedAt", and "username" (derived from AWS Cognito user data). This table serves as a centralized repository for storing and organizing resident-specific threshold settings, facilitating streamlined management and configuration of energy consumption thresholds. By incorporating properties like "createdAt" and "updatedAt," the table facilitates the recording of threshold entry creation and modification timestamps, guaranteeing precise audit trails and historical record-keeping. Through the integration of email notifications to the assigned email addresses, residents are able to monitor and efficiently manage their energy use since they receive timely alerts regarding threshold breaches. A smooth connection with the associated data source is also made possible by the association with the name of the energy consumption table that was obtained from AWS Cognito. This allows for effective monitoring and analysis of energy usage patterns based on resident-defined thresholds. This all-encompassing method optimizes the monitoring and alerting of energy consumption threshold management for residents by improving its granularity, flexibility, and functionality.

6.7.2 Use AWS AppSync GraphQL Schema to Create Threshold DynamoDB table



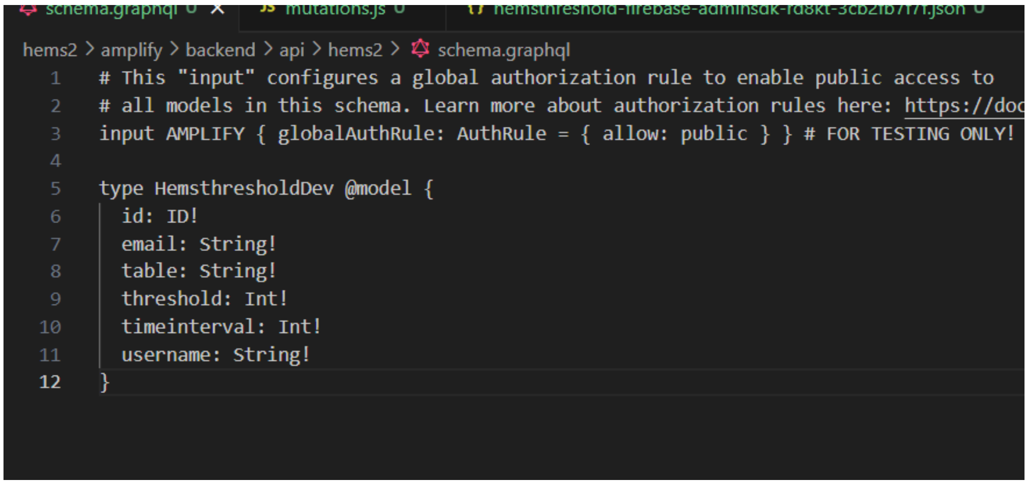
```
PS C:\Users\Owner\Downloads\assignment\assignment\hems> amplify add api
? Select from one of the below mentioned services: GraphQL
? Here is the GraphQL API that we will create. Select a setting to edit or cont
inue Continue
? Choose a schema template: Single object with fields (e.g., “Todo” with ID, na
me, description)

⚠ WARNING: your GraphQL API currently allows public create, read, update, and
delete access to all models via an API Key. To configure PRODUCTION-READY authh
orization rules, review: https://docs.amplify.aws/cli/graphql/authorization-rul
es

✅ GraphQL schema compiled successfully.

Edit your schema at C:\Users\Owner\Downloads\assignment\assignment\hems\amplify
\backend\api\hems\schema.graphql or place .graphql files in a directory at C:\U
sers\Owner\Downloads\assignment\assignment\hems\amplify\backend\api\hems\schema
? Do you want to edit the schema now? (Y/n) »
```

Figure 6.7.2: Create DynamoDB Table in React Native

This figure illustrates the step to create a DynamoDB table inside the React Native project. ‘amplify add API’ is executed to create GraphQL API to allow mobile application users to create, read, update and delete all models or tables created via an API key. This command facilitates the addition of the DynamoDB table, enabling its integration and association with the existing Amplify project. By incorporating this table into the project, it becomes seamlessly linked to the AppSync service, allowing GraphQLResolvers to execute Create, Read, Update, and Delete (CRUD) operations on the DynamoDB table via the mobile application utilized by residents. Through GraphQLResolvers, AppSync acts as a middleware, handling data interactions between the mobile application and the DynamoDB table. By abstracting away the complexity of direct database interfaces, this method provides developers with a streamlined and uniform interface for managing data activities. When Amplify and AppSync collaborate, developers can effectively incorporate CRUD features into the mobile application, improving user experience and giving residents convenient access to data management tools.

6.7.3 Create GraphQL schema

GraphQL schema defines the structure of the threshold data that residents can query or mutate through the GraphQL API created to create and delete threshold.



schema.graphql U ×
mutations.js U
hemsthreshold-firebase-adminsdk-fd8kt-3cb2fb7f71.json U
hems2 > amplify > backend > api > hems2 > schema.graphql

```
# This "input" configures a global authorization rule to enable public access to
# all models in this schema. Learn more about authorization rules here: https://doc
input AMPLIFY { globalAuthRule: AuthRule = { allow: public } } # FOR TESTING ONLY!

type HemsthresholdDev @model {
  id: ID!
  email: String!
  table: String!
  threshold: Int!
  timeinterval: Int!
  username: String!
}
```

Figure 6.7.3: GraphQL Schema

The figure presents that a type named HemsthresholdDev is defined in the supplied GraphQL schema. Type is a fundamental building block used to define the shape of data that can be queried or mutated. Its fields correspond to characteristics of threshold data, including email, table name, threshold value, time interval, and username. The @model directive instructs AWS AppSync and AWS Amplify to handle this type as a model, allowing for the automatic creation of GraphQL CRUD operations and the provisioning of DynamoDB tables. Although it is stated that this is only for testing, the comments also propose creating a global permission rule allowing public access to all models in the schema. More particular authorization rules are advised for real-world circumstances. The overall goals of this schema are to define the threshold data's structure and allow for easy connection with backend services and GraphQL operations via Amazon Amplify.

6.7.4 Create Mutation and Query

A mutation in GraphQL is used to modify threshold data on the server-side that is the threshold energy consumption table which allows residents to perform creating and deleting operation to data managed by the GraphQL server. Mutations are analogous to POST, PUT, PATCH, and DELETE requests in RESTful APIs.

```
export const createHemsthresholdDev = /* GraphQL */ `

              mutation             CreateHemsthresholdDev($input:
CreateHemsthresholdDevInput!) {

    createHemsthresholdDev(input: $input) {

      id

      email

      table

      threshold

      timeinterval

      username

    }

  }

`;



export const deleteHemsthresholdDev = /* GraphQL */ `
```

```
              mutation              DeleteHemsthresholdDev($input:
DeleteHemsthresholdDevInput!) {

    deleteHemsthresholdDev(input: $input) {

      id

    }

  }

`;

export const updateHemsthresholdDev = /* GraphQL */ `

  mutation UpdateHemsthresholdDev(

    $input: UpdateHemsthresholdDevInput!

    $condition: ModelHemsthresholdDevConditionInput

  ) {

    updateHemsthresholdDev(input: $input, condition: $condition) {

      id

      email

      table

      threshold

      timeinterval

      username

      createdAt

      updatedAt

      __typename

    }
  }

`;
```

Figure 6.7.4: GraphQL Mutation

The figure shows the GraphQL mutations are responsible for creating, updating and deleting entries in the HemsthresholdDev GraphQL type. The createHemsthresholdDev mutation takes an input of type CreateHemsthresholdDevInput, which contains the necessary fields for creating a new entry. Upon execution, it returns the created entry with fields including id, email, table, threshold, timeinterval, and username.

The deleteHemsthresholdDev mutation takes an input of type DeleteHemsthresholdDevInput, specifying the ID of the entry to be deleted. When executed, it removes the entry from the HemsthresholdDev type based on the provided ID, and it returns the deleted entry's ID as confirmation.

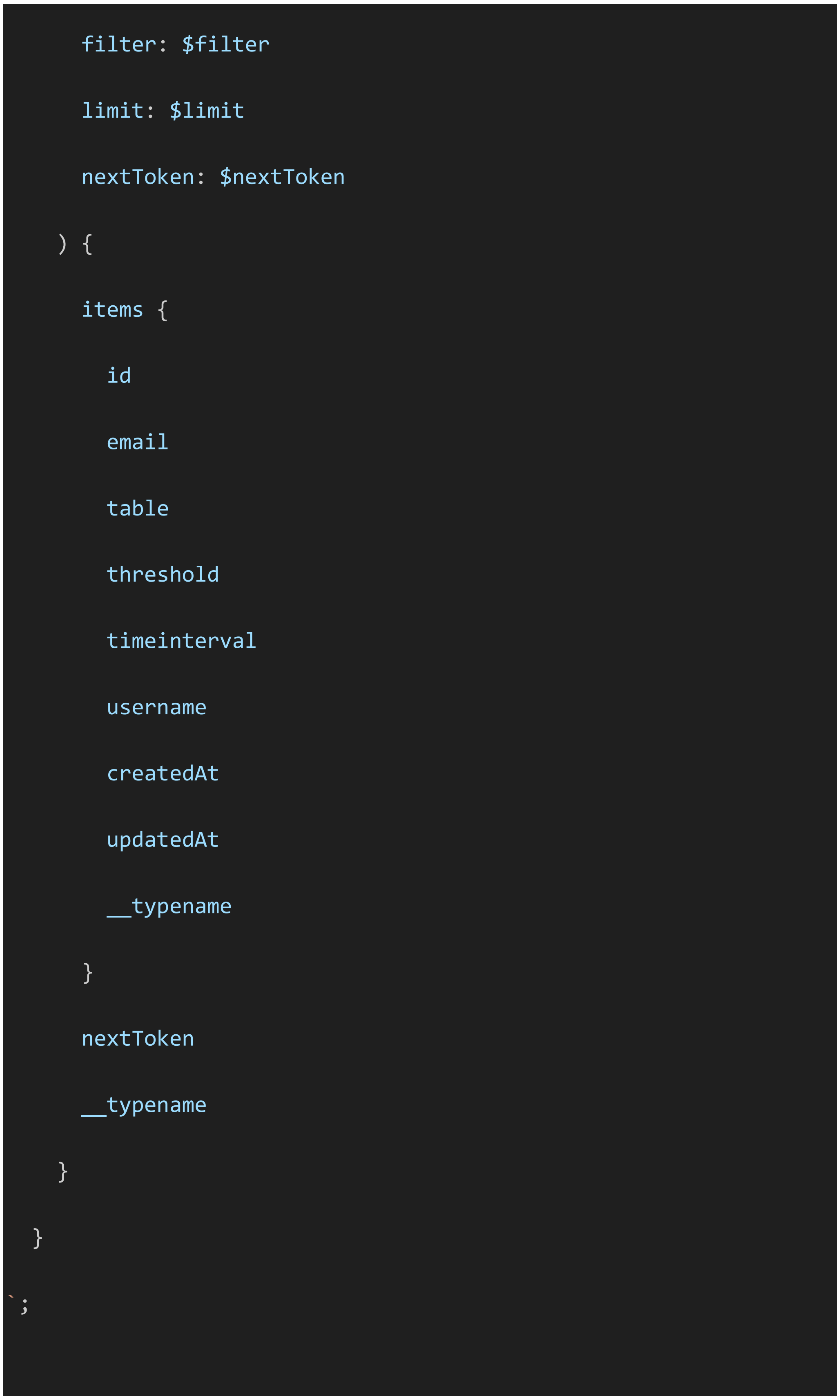
The updateHemsthresholdDev mutation takes an input of type UpdateHemsthresholdDevInput, specifying the ID of the entry to be updated and the new threshold and time interval data. When executed, it returns the created entry with fields including id, email, table, threshold, timeinterval, and username.

These mutations provide the necessary functionality to interact with the HemsthresholdDev type, allowing clients to create and delete entries in the associated DynamoDB table through the GraphQL API.

A query in GraphQL is to retrieve data from the GraphQL server which is similar like RESTful APIs' GET requests so that residents can use queries to define what data they require and only receive the required data in response.

```
/* eslint-disable */

// this is an auto generated file. This will be overwritten
```

```
export const getHemsthresholdDev = /* GraphQL */ `

  query GetHemsthresholdDev($id: ID!) {

    getHemsthresholdDev(id: $id) {

      id

      email

      table

      threshold

      timeinterval

      username

      createdAt

      updatedAt

      __typename

    }

  }

`;

export const listHemsthresholdDevs = /* GraphQL */ `

  query ListHemsthresholdDevs(

    $filter: ModelHemsthresholdDevFilterInput

    $limit: Int

    $nextToken: String

  ) {

    listHemsthresholdDevs(
```



```
      filter: $filter
      limit: $limit
      nextToken: $nextToken
    ) {
      items {
        id
        email
        table
        threshold
        timeinterval
        username
        createdAt
        updatedAt
        __typename
      }
      nextToken
      __typename
    }
  }
`;
```

Figure 6.7.5: GraphQL Queries

This figure presents the GraphQL queries that are auto-generated and provided by AWS AppSync for interacting with the HemsthresholdDev GraphQL type.

The getHemsthresholdDev query retrieves a single entry from the HemsthresholdDev type based on the provided ID. It returns fields including id, email, table, threshold, timeinterval, username, createdAt, updatedAt, and __typename.

The listHemsthresholdDevs query retrieves a list of entries from the HemsthresholdDev type, optionally filtered and paginated. It returns a list of items with fields including id, email, table, threshold, timeinterval, username, createdAt, updatedAt, __typename, as well as pagination information such as nextToken.

These queries enable clients to fetch data from the HemsthresholdDev type via the GraphQL API, providing both single-item retrieval and listing capabilities.

6.7.5 Create Retrieve, Create, Update and Delete functions in Mobile Application

```
import { listHemsthresholdDevs } from './src/graphql/queries';

import { createHemsthresholdDev, deleteHemsthresholdDev } from
'./src/graphql/mutations';




  const loadThresholds = async () => {

    try {

      setLoading(true);
      const currentUser = await Auth.currentAuthenticatedUser();

      const username = currentUser.username;

                      const      thresholdData      =      await
API.graphql(graphqlOperation(listHemsthresholdDevs, {

        filter: {

          username: {

            eq: username,

          },

        },

      }));

                             const        thresholdList         =
thresholdData.data.listHemsthresholdDevs.items;

      setThresholds(thresholdList);

    } catch (error) {

      console.log('Error fetching thresholds', error);

      Alert.alert('Error', 'Failed to fetch thresholds. Please try
again later.');

    } finally {

      setLoading(false);

    }

  };


  const handleCreateThreshold = async () => {

    try {
      setLoading(true);

      const currentUser = await Auth.currentAuthenticatedUser();

      const username = currentUser.username;

      await API.graphql(graphqlOperation(createHemsthresholdDev, {

        input: {

          username,

          threshold: newThreshold,

          timeinterval: newInterval,

          email: currentUser.attributes['email'],

          table: currentUser.attributes['website']

        }

      }));

      loadThresholds();

      setNewThreshold('');

      setNewInterval('');

    } catch (error) {

      console.log('Error creating new threshold', error);

      Alert.alert('Error', 'Failed to create threshold. Please try
again later.');

    } finally {

      setLoading(false);

    }

  };
  const handleRemoveThreshold = async (id) => {

    try {

      setLoading(true);

        await  API.graphql(graphqlOperation(deleteHemsthresholdDev,
{ input: { id } }));

      loadThresholds();

    } catch (error) {

      console.log('Error removing threshold', error);

      Alert.alert('Error', 'Failed to remove threshold. Please try
again later.');

    } finally {

      setLoading(false);

    }

  };

const handleUpdateThreshold = async (id, newThreshold, newInterval)
=> {

    try {

      const currentUser = await Auth.currentAuthenticatedUser();

      const username = currentUser.username;



      await API.graphql(graphqlOperation(updateHemsthresholdDev, {

        input: {

          id,
          username,

          threshold: newThreshold,

          timeinterval: newInterval,

        }

      }));


      loadThresholds();

    } catch (error) {

      console.log('Error updating threshold', error);

      Alert.alert('Error', 'Failed to update threshold. Please try
again later.');

    }
```

Figure 6.7.6: CRUD Code for Threshold in DynamoDB

The following example of code offers several functions for handling thresholds in a HemsthresholdDev DynamoDB table that are linked to a particular user. The thresholds linked to the currently authorized user are retrieved by the loadThresholds function. It first changes the loading state to true and then uses the Amplify Auth module to retrieve the currently authenticated user. Finally, it uses GraphQL to query the DynamoDB table and retrieve the thresholds that the username of the current user has filtered. After the data is fetched, the loading state is changed back to false and the retrieved thresholds are set in the component's state.

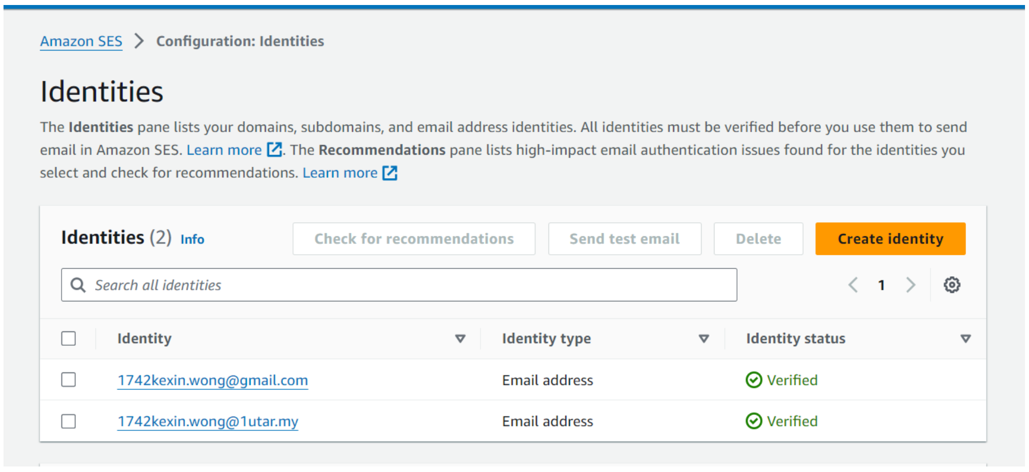
The function handleCreateThreshold is in charge of establishing a new threshold. It obtains the currently authenticated user and sets the loading state to true, just like loadThresholds. Then, in order to add a new threshold item to the DynamoDB table, it calls the createHemsthresholdDev GraphQL mutation operation. The username of the user, the new threshold value (newThreshold), the time interval, the email address of the user obtained from the current user's attributes, and the table name obtained from the user's website attribute are the input parameters for this mutation. It resets the new threshold and interval state variables and uses the loadThresholds function to reload the thresholds after they have been created successfully.

The function handleRemoveThreshold eliminates threshold connected to a certain ID. Like the previous functions, it initializes the loading state to true and uses the supplied ID to trigger a GraphQL mutation operation (deleteHemsthresholdDev) that removes the threshold from the DynamoDB table. It uses the loadThresholds function to reload the thresholds following the threshold's successful removal.

The function handleUpdateThreshold is responsible for updating a threshold associated with a specific ID. It follows a similar pattern to previous functions, starting by fetching the currently authenticated user and extracting their username. Then, it performs a GraphQL mutation operation (updateHemsthresholdDev) to update the threshold in the DynamoDB table. The mutation operation includes the ID of the threshold, the username of the current user, the new threshold value (newThreshold), and the new time interval (newInterval).

After successfully updating the threshold, it calls the loadThresholds function to reload the thresholds to ensure that the UI reflects the updated state. However, if an error occurs during the process, it catches the error and displays an alert message informing the user that the update failed, prompting them to try again later.

6.7.6 Create Identity via AWS SES



Amazon SES > Configuration: Identities

Identities

The **Identities** pane lists your domains, subdomains, and email address identities. All identities must be verified before you use them to send email in Amazon SES. Learn more. The **Recommendations** pane lists high-impact email authentication issues found for the identities you select and check for recommendations. Learn more

Identities (2) Info

	Identity	Identity type	Identity status
☐	1742kexin.wong@gmail.com	Email address	✓ Verified
☐	1742kexin.wong@1utar.my	Email address	✓ Verified

Figure 6.7.7: Identities Created in AWS SES

This figure illustrates the identities created for sending threshold notifications via AWS SES email. These identities are essential for establishing sender reputations and ensuring successful email delivery. By configuring these identities within AWS SES, such as verifying email addresses or domains, users can authorize SES to send emails on their behalf. This process enhances email deliverability and reduces the likelihood of emails being marked as spam by recipient servers. After setting up an identity with a specific email, this email can be used to send email notifications to other emails about the threshold information later.

6.7.7 Create AWS Lambda Function to Detect Threshold and Send Email Notification

```
const AWS = require('aws-sdk');


AWS.config.update({ region: 'us-east-1' });

const ddb = new AWS.DynamoDB.DocumentClient();

const ses = new AWS.SES({ apiVersion: '2010-12-01' });
```

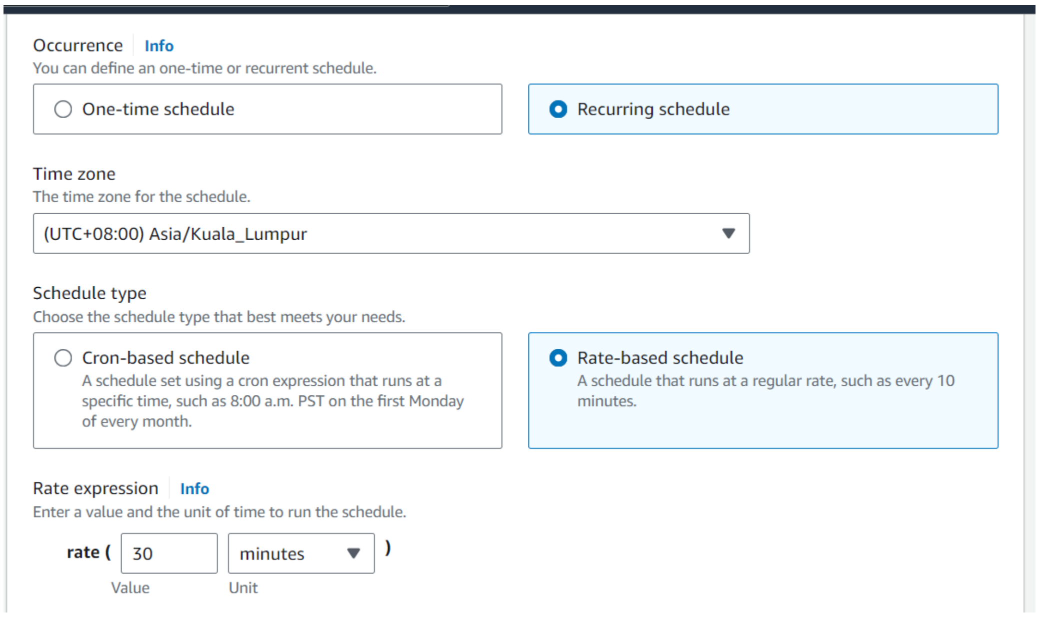
```
exports.handler = async (event) => {

    try {

        const thresholdData = await getAllThresholdData();



        for (const rowData of thresholdData) {

          const { table, email, threshold, timeinterval, username }
= rowData;



              // Calculate the time threshold for the previous time
interval

              const previousTime = Math.floor(Date.now() / 1000) -
timeinterval;

             console.log(rowData);

                         console.log( new Date(previousTime *
1000).toISOString());

            const previousTimeDate = new Date(previousTime * 1000);

                               const previousTimeFormatted =
`${previousTimeDate.getFullYear()}-${("0" +
(previousTimeDate.getMonth() + 1)).slice(-2)}-${("0" +
previousTimeDate.getDate()).slice(-2)} ${("0" +
previousTimeDate.getHours()).slice(-2)}:${("0" +
previousTimeDate.getMinutes()).slice(-2)}:${("0" +
previousTimeDate.getSeconds()).slice(-2)}`;



             // Define the params for the DynamoDB query to retrieve
the data for the previous time interval

             const params = {
                TableName: table,

                  KeyConditionExpression: "Device_ID = :d_id and
#ts >= :previousTime",

                ProjectionExpression: '#ts, ApparentPower', // Use
an alias for Timestamp using ExpressionAttributeNames

                ExpressionAttributeNames: {

                    "#ts": "Timestamp" // Alias for Timestamp

                },

                ExpressionAttributeValues: {

                    ':previousTime': previousTimeFormatted ,

                    ":d_id": 1, // Assuming Device_ID is a numeric
attribute, adjust if it's not

                }

            };



            // Query DynamoDB to retrieve the data for the previous
time interval

            const data = await ddb.query(params).promise();

            const records = data.Items;

            console.log(data);

            // Process the data and calculate the average apparent
power for the previous time interval

            if (records.length > 0) {

                let totalApparentPower = 0;

                for (const record of records) {
                    totalApparentPower += record.ApparentPower ||
0; // If apparent power is undefined, default to 0
                }
                const averageApparentPower = totalApparentPower /
records.length;
                console.log(averageApparentPower);
                if (averageApparentPower > threshold) {
                                                            await
sendEmailNotification({ averageApparentPower, username, email });
                }
            }
        }

        return {
            statusCode: 200,
            body: JSON.stringify('Event processed successfully'),
        };
    } catch (error) {
        console.error('Error processing event:', error);
        return {
            statusCode: 500,
            body: JSON.stringify('Error processing event'),
        };
    }
```

```
};


async function getAllThresholdData() {
    const params = {
         TableName:  'HemsthresholdDev-arhor534effbdfyibnvcawiq6u-
dev'
    };


    try {
        const data = await ddb.scan(params).promise();
        return data.Items;
    } catch (error) {
        console.error('Error retrieving threshold data:', error);
        return [];
    }
}


async function sendEmailNotification(data) {
     const thresholdExceededMessage = `The average apparent power
over the previous time interval has exceeded the threshold for user
${data.username}. Threshold: ${data.threshold}. Average Apparent
Power: ${data.averageApparentPower}`;


    const emailParams = {
        Destination: {
            ToAddresses: [data.email]
        },
        Message: {
            Body: {
                Text: {
                    Data: thresholdExceededMessage
                }
            },
            Subject: {
                Data: 'Threshold Exceeded Notification'
            }
        },
        Source: '1742kexin.wong@1utar.my'
    };

    try {
        await ses.sendEmail(emailParams).promise();
        console.log('Email sent successfully');
    } catch (error) {
        console.error('Error sending email:', error);
        throw error;
    }
}
```

Figure 6.7.8: Lambda Function to Detect Threshold

This figure shows AWS Lambda function to monitor energy consumption data stored in a DynamoDB table and send email notifications when the average apparent power exceeds predefined thresholds for individual users. Upon invocation, the function retrieves threshold data for all users and iterates through each record, querying DynamoDB to fetch energy consumption data for a specified time interval. If the average apparent power exceeds the threshold, an email notification is composed and sent using the Simple Email Service (SES). Error handling is included to manage any exceptions during processing, ensuring robust operation. Overall, this function provides a proactive approach to notify users of high energy consumption levels, facilitating timely awareness and potential mitigation actions.

6.7.8 Set CloudWatch Rule to trigger AWS Lambda Function Periodically



Occurrence Info
You can define an one-time or recurrent schedule.

One-time schedule [✓ Recurring schedule]

Time zone
The time zone for the schedule.

(UTC+08:00) Asia/Kuala_Lumpur

Schedule type
Choose the schedule type that best meets your needs.

Cron-based schedule
A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

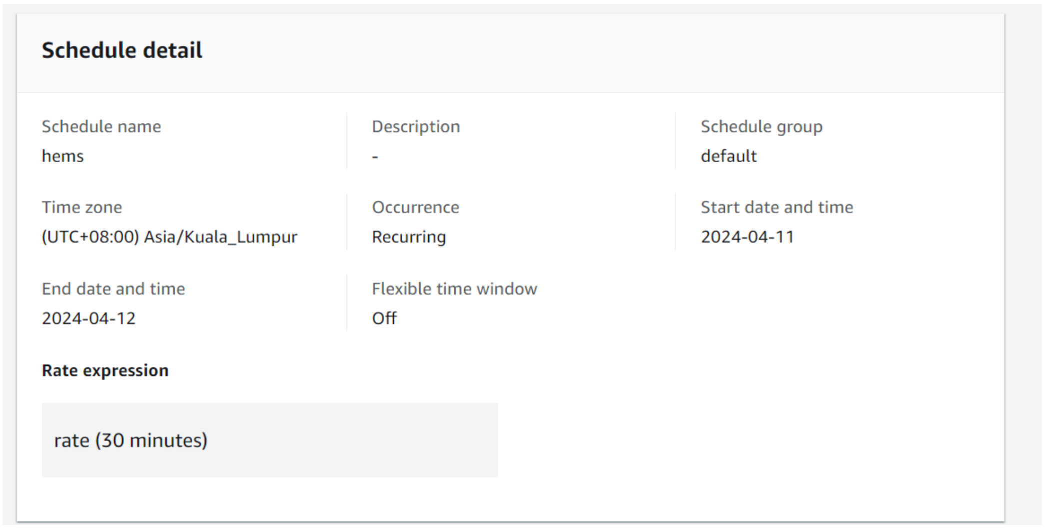
[✓ Rate-based schedule]
A schedule that runs at a regular rate, such as every 10 minutes.

Rate expression Info
Enter a value and the unit of time to run the schedule.

rate (30 minutes)
Value Unit

Figure 6.7.9: CloudWatch Rule Setting

Based on the figure, the configuration interface for AWS EventBridge Scheduler, a schedule named "hems" is being established to trigger a Lambda function for detecting threshold periodically. The schedule, set within the 'Default' group, is defined as a rate-based schedule with a frequency of every 30 minutes. This schedule is designed to invoke a Lambda function that detects threshold breaches and sends email notifications accordingly. The time zone is specified as "(UTC+08:00) Asia/Kuala_Lumpur" to align with the desired time zone. Additionally, there is an option to configure a flexible time window, although it's currently set to "Off" in this setup. This schedule is set to start on April 11, 2024, and continue indefinitely, as no end date is specified.



Schedule detail

Schedule name hems	Description -	Schedule group default
Time zone (UTC+08:00) Asia/Kuala_Lumpur	Occurrence Recurring	Start date and time 2024-04-11
End date and time 2024-04-12	Flexible time window Off	

Rate expression

```
rate (30 minutes)
```

Figure 6.7.10: CloudWatch Rule Schedule Details

This figure shows the created rule in AWS CloudWatch which will be invoked each 30 minutes to check the thresholds of different residents and send email notifications to them if the energy consumption data exceed the thresholds.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

This chapter describes various testing methodologies carried out within project including Unit Testing, Integration Testing, System Testing, Code Quality Analysis and System Usability Scale Evaluation.

7.2 Unit Testing

The Jest testing framework are used to test the React Native mobile application in this project to write unit tests and snapshot tests to ensure the correctness and reliability of the project code. The unit testing conducted covers a range of scenarios, such as rendering components correctly, navigating between screens, handling user input, calling external functions, and logging errors, to provide comprehensive coverage of the project functionalities and prove its reliability and robustness. Manual testing is also used to test both mobile application and web-based dashboard to maximize code testing coverage.

Mobile Application

Table 7.2.1: Unit Test Case for Register Screen 1

Test Case ID	UC001
Related FR ID	FR001
Test Case	Navigate to RegisterScreen when SignUp button is pressed
Test Case Description	This test case verifies the navigate function is called with the parameter 'RegisterScreen' when the Sign Up button is pressed.

Test Steps	• Render the Login component. • Press the Sign Up button.
Expected Result	The navigate function should be called with the parameter 'RegisterScreen'.
Status	Pass

Table 7.2.2: Unit Test Case for Register Screen 2

Test Case ID	UC002
Related FR ID	FR001
Test Case	SignUpPage - Call signUpAWS Functionality
Test Case Description	This test case verifies that the SignUpPage component correctly calls the signUpAWS function from the aws-amplify library with the provided email and password when the Sign Up button is pressed.
Test Steps	• Mock the aws-amplify library using Jest to provide custom implementations for the signIn and signUp functions. • Render the SignUpPage component using the render function from @testing-library/react-native. • Simulate user input by changing the text in the email and password input fields to 'test@example.com' and 'password123' respectively. • Simulate a button press on the Sign Up button. • Verify that the Auth.signUp function is called with the following parameters: username: 'test@example.com'

	password: 'password123' attributes: { email: 'test@example.com' } autoSignIn: { enabled: false }
Expected Result	The SignUpPage component should call the Auth.signUp function with the provided email and password when the Sign Up button is pressed, ensuring that the signUp functionality functions correctly.
Status	Pass

Table 7.2.3 : Unit Test Case for Login Screen 1

Test Case ID	UC003
Related FR ID	FR002
Test Case	Navigate to LoginScreen
Test Case Description	This test case verifies that when the button on the StartScreen is pressed, it navigates to the LoginScreen component as expected.
Test Steps	• Render the StartScreen component. • Locate the start image by testID. Verify that the start image is rendered. • Locate the title element containing the text "Welcome to the HEMS". • Locate the subtitle element containing the text "Push the Button Below to Begin". • Verify that both the title and subtitle are rendered. • Simulate a press event on the button element labeled "Begin".

Expected Result	Pressing the "Begin" button should trigger navigation to the LoginScreen component.
Status	Pass

Table 7.2.4: Unit Test Case for Login Screen 2

Test Case ID	UC004
Related FR ID	FR002
Test Case	Renders Login Screen Components
Test Case Description	This test case verify that the Login component renders the textInput fields of username and password and Sign In and Sign Up buttons.
Test Steps	• Render the Login component. • Locate the username TextInput field. • Locate the password TextInput field. • Locate the Sign In button. • Locate the Sign Up button.
Expected Result	The Login component should render the username and password TextInput fields, along with the Sign In and Sign Up buttons.
Status	Pass

Table 7.2.5: Unit Test Case for Login Screen 3

Test Case ID	UC005
Related FR ID	FR002

Test Case	Calls signIn function with username and password
Test Case Description	This test case verifies that the signIn function is called with the provided username and password when the Sign In button is pressed.
Test Steps	• Render the Login component. • Enter a username into the username TextInput field. • Enter a password into the password TextInput field. • Press the Sign In button.
Expected Result	The signIn function should be called with the entered username and password.
Status	Pass

Table 7.2.6: Unit Test Case for Login Screen 4

Test Case ID	UC006
Related FR ID	FR002
Test Case	Does not call signIn function when SignIn button is pressed with empty fields
Test Case Description	This test case verifies that the signIn function is not called when the Sign In button is pressed with empty username and password fields.
Test Steps	• Render the Login component. • Press the Sign In button without entering any username or password.
Expected Result	The signIn function should not be called.

Status	Pass

Table 7.2.7: Unit Test Case for Login Screen 5

Test Case ID	UC007
Related FR ID	FR002
Test Case	Logs an error when signIn throws an error
Test Case Description	This test case verifies that an error is logged when the signIn function throws an error.
Test Steps	• Render the Login component. • Enter a username and password into the respective TextInput fields. • Press the Sign In button, causing the signIn function to throw an error.
Expected Result	An error should be logged indicating the failure to sign in.
Status	Pass

Table 7.2.8: Unit Test Case for Real Time Graph

Test Case ID	UC008
Related FR ID	FR003
Test Case	RealTimeGraph - Render Correctly with Non-empty receivedData
Test Case Description	This test case verifies that the RealTimeGraph component renders correctly when the receivedData prop is not empty.

Test Steps	• Mock the Date object to return a fixed time ('2024-04-13T17:32:27Z') using jest.spyOn and Date.now. • Define the receivedData array containing mock data points with timestamps and values. • Render the RealTimeGraph component with the mock receivedData using the render function from '@testing-library/react-native'. • Locate the text element displaying the timestamp in the rendered component. • Use the expected date string 'April 13th 2024, 11:02:27 PM' in the 'Asia/Kolkata' timezone.
Expected Result	The RealTimeGraph component should render correctly with the provided receivedData, and the timestamp should be displayed in the expected format ('April 13th 2024, 11:02:27 PM').
Status	Pass

Table 7.2.9: Unit Test Case for Home Screen

Test Case ID	UC009
Related FR ID	FR003
Test Case	Renders HomeScreen without crashing
Test Case Description	This test case verifies that the HomeScreen component renders without crashing.
Test Steps	• Create a snapshot of the HomeScreen component using react-test-renderer. • Convert the snapshot to JSON format.

	• Verify that the JSON representation of the HomeScreen component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the HomeScreen component should be truthy.
Status	Pass

Table 7.2.10: Unit Test Case for Historical Energy Consumption Screen

Test Case ID	UC0010
Related FR ID	FR004
Test Case	Renders Energy Consumption Screen without crashing
Test Case Description	This test case verifies that the Energy Consumption Screen component renders without crashing.
Test Steps	• Create a snapshot of the Energy Consumption Screen using react-test-renderer. • Convert the snapshot to JSON format. • Verify that the JSON representation of the Energy Consumption Screen component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the Energy Consumption Screen component should be truthy.
Status	Pass

Table 7.2.11: Unit Test Case for Energy Monitoring Screen

Test Case ID	UC011
Related FR ID	FR005, FR006
Test Case	Renders Energy Monitoring Screen without crashing
Test Case Description	This test case verifies that the Energy Monitoring Screen component renders without crashing.
Test Steps	• Create a snapshot of the Energy Monitoring Screen using react-test-renderer. • Convert the snapshot to JSON format. • Verify that the JSON representation of the Energy Monitoring Screen component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the Energy Monitoring Screen component should be truthy.
Status	Pass

Table 7.2.12: Unit Test Case for Start Screen 1

Test Case ID	UC012
Related FR ID	-
Test Case	Render Start Screen Components
Test Case Description	This test case verifies that the StartScreen Component renders the start image, title, and subtitle correctly.
Test Steps	• Render the StartScreen component.

	• Locate the start image by testID. Verify that the start image is rendered. • Locate the title element containing the text "Welcome to the HEMS". • Locate the subtitle element containing the text "Push the Button Below to Begin". • Verify that both the title and subtitle are rendered. • Simulate a press event on the button element labeled "Begin".
Expected Result	The start image, title, and subtitle should be rendered successfully.
Status	Pass

Table 7.2.13: Unit Test Case for Start Screen 2

Test Case ID	UC013
Related FR ID	-
Test Case	Render Start Screen Components
Test Case Description	This test case verifies that the StartScreen Component renders the start image, title, and subtitle correctly.
Test Steps	• Render the StartScreen component. • Locate the start image by testID. Verify that the start image is rendered. • Locate the title element containing the text "Welcome to the HEMS". • Locate the subtitle element containing the text "Push the Button Below to Begin".

	• Verify that both the title and subtitle are rendered. • Simulate a press event on the button element labeled "Begin".
Expected Result	The start image, title, and subtitle should be rendered successfully.
Status	Pass

Table 7.2.14: Unit Test Case for Setting Screen

Test Case ID	UC014
Related FR ID	-
Test Case	Navigation Button Press
Test Case Description	This test case verifies that the Settings component correctly navigates to the specified screens when navigation buttons are pressed.
Test Steps	• Render the Settings component. • Mock navigation functions for Notifications and Privacy screens. • Simulate a button press on the Notifications button. • Verify that the navigateToNotifications function is called. • Simulate a button press on the Privacy button. • Verify that the navigateToPrivacy function is called.
Expected Result	When the Notifications button is pressed, the Settings component should navigate to the Notifications screen by calling the navigateToNotifications function.

	When the Privacy button is pressed, the Settings component should navigate to the Privacy screen by calling the navigateToPrivacy function.
Status	Pass

Table 7.2.15: Unit Test Case for User Profile Screen

Test Case ID	UC015
Related FR ID	FR007
Test Case	Renders User Profile Screen without crashing
Test Case Description	This test case verifies that the User Profile Screen component renders without crashing.
Test Steps	• Create a snapshot of the User Profile Screen using react-test-renderer. • Convert the snapshot to JSON format. • Verify that the JSON representation of the User Profile Screen component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the User Profile Screen component should be truthy.
Status	Pass

Table 7.2.16: Unit Test Case for Help and Support Screen

Test Case ID	UC016
Related FR ID	-
Test Case	HelpAndSupport - Render Content

Test Case Description	This test case verifies that the HelpAndSupport component renders its content correctly.
Test Steps	• Render the HelpAndSupport component. • Locate and verify the presence of the welcome message: "Welcome to the Help and Support Page for our Home Energy Monitoring System." • Locate and verify the presence of the support contact information: "If you are experiencing any issues with your system, please contact our support team at support@homeenergy.com." • Locate and verify the presence of the website link for FAQs and troubleshooting: "For FAQs and troubleshooting, please visit our website at www.homeenergy.com/help."
Expected Result	The HelpAndSupport component should render the welcome message, support contact information, and website link for FAQs and troubleshooting.
Status	Pass

Table 7.2.17: Unit Test Case for Term and Policy Screen

Test Case ID	UC017
Related FR ID	-
Test Case	Renders Term and Policy Screen without crashing
Test Case Description	This test case verifies that the Term and Policy Screen component renders without crashing.
Test Steps	• Create a snapshot of the Term and Policy Screen using react-test-renderer. • Convert the snapshot to JSON format.

	• Verify that the JSON representation of the Term and Policy component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the Term and Policy Screen component should be truthy.
Status	Pass

Table 7.2.18: Unit Test Case for Help and Support Screen

Test Case ID	UC018
Related FR ID	-
Test Case	Renders Help and Support Screen without crashing
Test Case Description	This test case verifies that the Help and Support Screen component renders without crashing.
Test Steps	• Create a snapshot of the Help and Support Screen using react-test-renderer. • Convert the snapshot to JSON format. • Verify that the JSON representation of the Help and Support component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the Help and Support Screen component should be truthy.
Status	Pass

Table 7.2.19: Unit Test Case for Report Problem Screen

Test Case ID	UC019
Related FR ID	-
Test Case	Renders Report Problem Screen without crashing
Test Case Description	This test case verifies that the Report Problem Screen component renders without crashing.
Test Steps	• Create a snapshot of the Report Problem Screen using react-test-renderer. • Convert the snapshot to JSON format. • Verify that the JSON representation of the Report Problem component is truthy, indicating that the component rendered successfully.
Expected Result	The JSON representation of the Report Problem Screen component should be truthy.
Status	Pass

Web Based Dashboard

Table 7.2.20: Unit Test Case for Web-based Dashboard Page

Test Case ID	UC020
Related FR ID	FR008
Test Case	Renders User Dashboard Page without crashing
Test Case Description	This test case verifies that the User Dashboard Page component renders without crashing.
Test Steps	• Open the web application. • Navigate to the User Dashboard Page.

	• Observe if the page loads without any errors or crashes.
Expected Result	The User Dashboard Page should render without any errors or crashes, displaying all necessary components and functionality.
Status	Pass

Table 7.2.21: Unit Test Case for Web-based Manage User Page

Test Case ID	UC021
Related FR ID	FR009
Test Case	Renders Manage User Page without crashing
Test Case Description	This test case verifies that the Manage User Page component renders without crashing.
Test Steps	• Open the web application. • Navigate to the Manage User Page. • Observe if the page loads without any errors or crashes.
Expected Result	The Manage User Page should render without any errors or crashes, displaying all necessary components and functionality.
Status	Pass

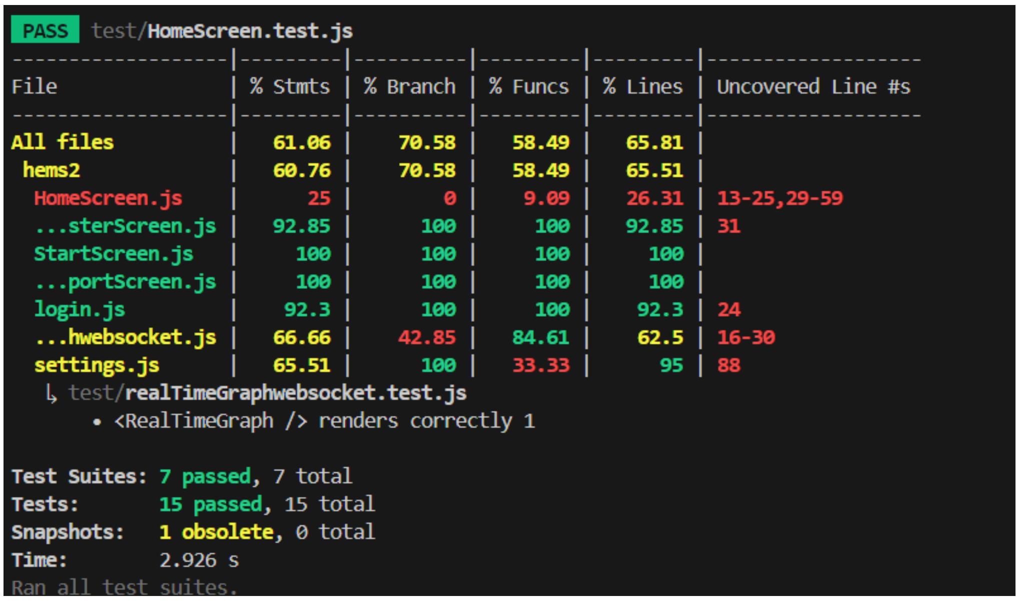
Table 7.2.22: Unit Test Case for Web-based Administrator Login Page

Test Case ID	UC022
Related FR ID	FR010

Test Case	Renders Administrator Login Page without crashing
Test Case Description	This test case verifies that the Administrator Login Page component renders without crashing.
Test Steps	• Open the web application. • Navigate to the Administrator Login Page. • Observe if the page loads without any errors or crashes.
Expected Result	The Administrator Login Page should render without any errors or crashes, displaying all necessary components and functionality.
Status	Pass

Table 7.2.23: Unit Test Case for Web-based Administrator Register Page

Test Case ID	UC023
Related FR ID	FR011
Test Case	Renders Administrator Register Page without crashing
Test Case Description	This test case verifies that the Administrator Register Page component renders without crashing.
Test Steps	• Open the web application. • Navigate to the Administrator Register Page. • Observe if the page loads without any errors or crashes.
Expected Result	The Administrator Register Page should render without any errors or crashes, displaying all necessary components and functionality.
Status	Pass



```
PASS  test/HomeScreen.test.js
-------------------|---------|----------|---------|---------|-------------------
File               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------|---------|----------|---------|---------|-------------------
All files          |   61.06 |    70.58 |   58.49 |   65.81 |
 hems2             |   60.76 |    70.58 |   58.49 |   65.51 |
  HomeScreen.js    |      25 |        0 |    9.09 |   26.31 | 13-25,29-59
  ...sterScreen.js |   92.85 |      100 |     100 |   92.85 | 31
  StartScreen.js   |     100 |      100 |     100 |     100 |
  ...portScreen.js |     100 |      100 |     100 |     100 |
  login.js         |    92.3 |      100 |     100 |    92.3 | 24
  ...hwebsocket.js |   66.66 |    42.85 |   84.61 |    62.5 | 16-30
  settings.js      |   65.51 |      100 |   33.33 |      95 | 88
   ↳ test/realTimeGraphwebsocket.test.js
       • <RealTimeGraph /> renders correctly 1

Test Suites: 7 passed, 7 total
Tests:       15 passed, 15 total
Snapshots:   1 obsolete, 0 total
Time:        2.926 s
Ran all test suites.
```

Figure 7.2.1: Result of Unit Testing

7.3 Integration Testing

Integration testing is done using Jest for automation testing and supplemented with manual testing to cover integrations that are difficult to test automatically including user interface interaction.

Mobile Application

Table 7.3.1: Integration Test Case for Login Screen

Test Case ID	TC001
Test Case	Submit Form and Call Auth.signIn
Test Case Description	This test case verifies that the LoginScreen component submits the form correctly and calls the Auth.signIn function with the provided username and password.
Test Steps	1. Render the LoginScreen component using the render function from @testing-library/react-native. 2. Simulate user input by changing the text in the username and password input fields. 3. Simulate form submission by pressing the Sign In button. Wait for a short delay to allow for any asynchronous operations to complete. Wait for the promise to resolve using the waitFor function from @testing-library/react-native and verify that the Auth.signIn function is called with the expected username and password.
Test Data	Email:"test@example.com" Password: "password123"

Expected Result	The LoginScreen component should submit the form correctly and call the Auth.signIn function with the provided username and password.
Status	Pass
Pass criteria	1. The form submission should be simulated correctly. 2. The Auth.signIn function should be called with the expected username and password. 3. Any asynchronous operations should be allowed to complete before verifying the function call. 4. The test should pass without any errors or exceptions.

Table 7.3.2: Integration Test Case for Login Screen 2

Test Case ID	TC002
Test Case	Navigate to LoginScreen on ButtonPress
Test Case Description	This test case verifies that pressing the "Begin" button on the StartScreen component navigates the user to the LoginScreen.
Test Steps	Preconditions: 1. The application is running on a test environment. 2. The StartScreen component is rendered with the mocked navigation object. Test Steps: 3. Simulate a press event on the "Begin" button. 4. Capture the navigation action triggered by the button press.

Test Data	-
Expected Result	The application should navigate the user to the LoginScreen upon pressing the "Begin" button.
Status	Pass
Pass criteria	The navigate function is called with the correct screen name.

Table 7.3.3: Integration Test Case for Register Screen

Test Case ID	TC003
Test Case	User Sign Up
Test Case Description	This test case verifies that the SignUpPage component successfully signs up a user when the "Sign Up" button is pressed.
Test Steps	Preconditions: 1. The application is running and the SignUpPage component is loaded. 2. The aws-amplify library is properly mocked to simulate the sign-up process. 3. The user has entered valid email and password information into the input fields. Test Steps: User Interaction: 1. Enter a valid email into the email input field. 2. Enter a valid password into the password input field. 3. Press the "Sign Up" button. Verification:

	4. Wait for the asynchronous operations to complete. 5. Verify that the Auth.signUp function from the aws-amplify library is called with the correct parameters: I. Username should be set to the entered email. II. Password should match the entered password. III. Attributes should include the entered email. IV. Auto sign-in should be disabled. 6. Verify that the navigation function (navigation.navigate) is called with the correct screen name ('Confirmation') upon successful sign-up.
Test Data	Email: "test@example.com" Password: "password123"
Expected Result	The SignUpPage component should successfully trigger the sign-up process with the provided user credentials. Upon successful sign-up, the user should be navigated to the 'Confirmation' screen.
Status	Pass
Pass criteria	The test passes if the navigate function is called with the correct screen name.

Table 7.3.4: Integration Test Case for Home Screen

Test Case ID	TC004
Test Case	Renders HomeScreen correctly and opens WebSocket connection

Test Case Description	This test case verifies that the HomeScreen component renders correctly and establishes a WebSocket connection upon loading.
Test Steps	Preconditions: I. Mock the Auth module and the global WebSocket object. Test Steps: 1. Mock the Auth.currentAuthenticatedUser function to return a user with a WebSocket address. 2. Render the HomeScreen component. 3. Wait for the WebSocket connection to open. 4. Verify that the WebSocket connection is opened with the correct address. 5. Verify that the text "WebSocket connection opened" appears on the screen.
Test Data	-
Expected Result	The HomeScreen component renders correctly, establishes a WebSocket connection, and displays the appropriate text indicating that the connection has been opened.
Status	Pass
Pass criteria	The test passes if the WebSocket connection is opened with the correct address and the text "WebSocket connection opened" is displayed on the screen.

Table 7.3.5 : Integration Test Case for Energy Monitoring Screen 1

Test Case ID	TC005
Test Case	Add Threshold Button Click
Test Case Description	This test case verifies that a new threshold is added when the "Add Threshold" button is clicked.
Test Steps	1. Render the EnergyMonitorPage component. 2. Simulate user input for new threshold and interval. 3. Click the "Add Threshold" button. 4. Wait for component to finish rendering and async operations to complete.
Test Data	New threshold value: '50' New interval value: '10'
Expected Result	The threshold with the value of '50' should be added to the list.
Status	Pass
Pass criteria	The test case passes if the threshold with the value of '50' is successfully added to the list after the component finishes rendering.

Table 7.3.6: Integration Test Case for Energy Monitoring Screen 2

Test Case ID	TC006
Test Case	Component Render and loadThresholds Call
Test Case Description	This test case ensures that the component renders without crashing and calls the loadThresholds function on mount.

Test Steps	1. Render the EnergyMonitorPage component. 2. Wait for component to finish rendering and async operations to complete.
Test Data	-
Expected Result	The component should render successfully without any errors, and the loadThresholds function should be called once during mount.
Status	Pass
Pass criteria	The test case passes if the component renders without errors and the loadThresholds function is called once during mount.

Table 7.3.7: Integration Test Case for User Profile Screen

Test Case ID	TC007
Test Case	Navigation to Edit Profile
Test Case Description	This test case verifies if clicking "Edit Profile" navigates to profile screen
Test Steps	1. Render the Settings component.
Test Data	-
Expected Result	The Settings component should be rendered. "Edit Profile" button should be present.
Status	Pass
Pass criteria	Navigation to profile screen is successful.

Table 7.3.8: Integration Test Case for Logout

Test Case ID	TC008
Test Case	Logout
Test Case Description	This test case verifies if clicking "Log out" logs the user out.
Test Steps	1. Render the Settings component. 2. Press Logout button. 3. Navigate to Login page.
Test Data	-
Expected Result	The Settings component should be rendered. "Log out" button should be present. The Login component should be rendered.
Status	Pass
Pass criteria	User is logged out successfully.

Table 7.3.9: Integration Test Case for Historical Energy Consumption Screen

Test Case ID	TC009
Test Case	Display Month Year Energy Consumption Data
Test Case Description	This test case verifies that the MonthYear component displays energy consumption data for the selected month and year.
Test Steps	1. Navigate to the manageEnergyConsumption page. 2. Ensure that energy consumption data for the current month and year is displayed.

	3. Toggle between viewing weekly and monthly energy consumption data. 4. Verify that the chart updates accordingly to display the selected time frame. 5. Click on a data point on the chart. 6. Verify that a pop-up displays the energy consumption for the selected date. 7. Optionally, navigate to the settings to check if the most frequent period and total carbon emissions are displayed correctly.
Test Data	-
Expected Result	The manageEnergyConsumption component should render without errors. Energy consumption data for the current month and year should be displayed. The chart should update dynamically when toggling between weekly and monthly views. Clicking on a data point on the chart should display a pop-up showing the energy consumption for the selected date. The most frequent period and total carbon emissions should be displayed correctly in the settings section.
Status	Pass
Pass criteria	The manageEnergyConsumption component displays energy consumption data accurately and functions as expected.

Web Based Dashboard

Table 7.3.10: Integration Test Case for displaying Cognito Users Data

Test Case ID	TC010
Test Case	Display Cognito Users Data
Test Case Description	This test case verifies that the Cognito users page displays user data correctly.
Test Steps	1. Navigate to the Cognito users page. 2. Check if the user data is displayed in the table.
Test Data	Sample Cognito user data: - User 1: - Username: user1 - Email: user1@example.com - Address: 123 Main St - Website: http://example.com - User 2: - Username: user2 - Email: user2@example.com - Address: 456 Elm St - Website: http://example.org
Expected Result	The Cognito users page should render without errors and display the user data.
Status	Pass

Pass criteria	The Cognito users page displays user data accurately and functions as expected.

Table 7.3.11: Integration Test Case for Dashboard Component

Test Case ID	TC011
Test Case	Verify Integration of User Dashboard Components
Test Case Description	This test case verifies that the integration of user data, chart visualization, and dynamic content in the User Dashboard.
Test Steps	1. Open a web browser. 2. Navigate to the URL of the User Dashboard page. 3. Observe the layout and elements of the User Dashboard. 4. Verify the presence and accuracy of user attributes. 5. Validate the correctness of additional information. 6. Confirm the presence and functionality of the daily energy consumption chart. 7. Confirm the presence and functionality of the monthly energy consumption chart.
Test Data	-
Expected Result	User attributes, including Username, WebsocketAPI, and DynamoDB table name, should be displayed accurately. Most Frequent Period and Total Carbon Emissions should be displayed correctly.

	The daily chart should display accurate energy consumption data for each day. The monthly chart should display accurate energy consumption data for each month.
Status	Pass
Pass criteria	The User Dashboard page should display user attributes, additional information, and energy consumption charts accurately.

Table 7.3.12: Integration Test Case for Register Administrator Account

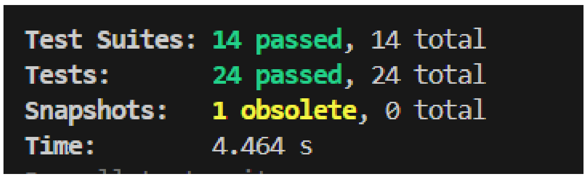
Test Case ID	TC012
Test Case	Register Form Integration Test
Test Case Description	This test case verifies that the integration of the registration form with the application backend.
Test Steps	1. Open a web browser. 2. Navigate to the URL of the registration page. 3. Fill in the registration form with valid data. 4. Submit the registration form. 5. Verify the registration process by checking the database. 6. Attempt to register with invalid data (e.g., missing required fields, invalid email format, weak password).
Test Data	- Name: "John Doe" - Email: "johndoe@example.com" - Password: "Password123"

Expected Result	The form should be submitted successfully without errors, and the user should be redirected to the Cognito user page.
Status	Pass
Pass criteria	Valid user data submitted through the registration form should result in the creation of a new user account. Invalid or incomplete data submission should trigger appropriate error messages.

Table 7.3.13: Integration Test Case for Login Administrator

Test Case ID	TC013
Test Case	Login Functionality Test
Test Case Description	This test case verifies that the login functionality works as expected by allowing users to log in with valid credentials.
Test Steps	1. Visit the login page. 2. Enter valid email address and password. 3. Click on the "Login" button. 4. Verify the redirection to the Cognito user page.
Test Data	Valid email address: test@example.com Valid password: password123
Expected Result	The form should be submitted successfully without errors, and the user should be redirected to the Cognito user page.

Status	Pass
Pass criteria	The user successfully logs in and is redirected to the Cognito user page without any errors.



```
Test Suites: 14 passed, 14 total
Tests:       24 passed, 24 total
Snapshots:   1 obsolete, 0 total
Time:        4.464 s
```

Figure 7.3.1: Result of Integration Testing

7.4 System Testing

Table 7.4.1: System Test Case for Create User Account

Test Case ID:	1	
Test Case:	Create user account	
Test Steps:	1. Press Sign Up button in Login Page. 2. Go to Register Page. 3. Fill in valid Email and Password. 4. Press Sign Up button. 5. Go to Confirmation Page. 6. Get Confirmation Code from email. 7. Fill in Confirmation Code. 8. Press Confirm Account button.	
Test Result	Expected Result	Actual Result
	1. Account has been created and can be used to sign in account. 2. User returned to Sign In page automatically after clicking Confirm Account button.	1. Account has been created and can be used to sign in account. 2. User returned to Sign In page automatically after clicking Confirm Account button.

Table 7.4.2: System Test Case for Login User Account

Test Case ID:	2	
Test Case:	Login user account	
Test Steps:	1. Press Begin button in Start Page. 2. Go to Sign In Page. 3. Fill in valid Email and Password. 4. Press Sign In button. 5. Go to Home Page.	
Test Result	Expected Result	Actual Result
	Navigate to Home Page.	Navigate to Home Page.

Table 7.4.3: System Test Case for Displaying Real Time Energy Consumption

Test Case ID:	3	
Test Case:	Display Real Time Energy Consumption Home Page	
Test Steps:	1. Go to Home Page. 2. Wait for Connecting… text changed to WebSocket connection opened.	
Test Result	Expected Result	Actual Result
	Real time energy consumption graph is shown in Home Page.	Real time energy consumption graph is shown in Home Page.

Table 7.4.4: System Test Case for Viewing Historical Energy Consumption

Test Case ID:	4

Test Case:	Historical Energy Consumption Data Fetching	
Test Steps:	1. Verify that the component fetches weekly, monthly energy consumption data, and total carbon emissions data from the appropriate APIs. 2. Check that the fetched data is stored correctly in the component state.	
Test Result	Expected Result	Actual Result
	The component fetches data from APIs and stores it correctly in the component state.	The component fetches data from APIs and stores it correctly in the component state.

Table 7.4.5: System Test Case for Showing Weekly Energy Consumption

Test Case ID:	5	
Test Case:	Show Weekly Energy Consumption Chart	
Test Steps:	1. Press tab navigation historical energy consumption icon. 2. Go to Historical Energy Consumption Page. 3. Verify that the LineChart component displays the correct data points based on the fetched weekly data.	
Test Result	Expected Result	Actual Result
	The LineChart component displays the correct data points and updates the chart title appropriately.	The LineChart component displays the correct data points and updates the chart title appropriately.

Table 7.4.6: System Test Case for Showing Monthly Energy Consumption

Test Case ID:	6	
Test Case:	Show Monthly Energy Consumption Chart	
Test Steps:	1. Press tab navigation historical energy consumption icon. 2. Go to Historical Energy Consumption Page. 3. Press View Monthly button to switch to display monthly energy consumption data. 4. Verify that the LineChart component displays the correct data points based on the fetched monthly data.	
Test Result	Expected Result	Actual Result
	The LineChart component displays the correct data points and updates the chart title appropriately.	The LineChart component displays the correct data points and updates the chart title appropriately.

Table 7.4.7: System Test Case for Historical Energy Consumption Data Point

Test Case ID:	7	
Test Case:	Historical Energy Consumption Data Point Click	
Test Steps:	1. Press tab navigation historical energy consumption icon 2. Go to Historical Energy Consumption Page 3. Click on a data point on the Weekly Energy Consumption Chart and verify that an alert displays the correct energy consumption value for that date	
Test Result	Expected Result	Actual Result

	Clicking on a data point on the Weekly Energy Consumption Chart displays an alert with the correct energy consumption value.	Clicking on a data point on the Weekly Energy Consumption Chart displays an alert with the correct energy consumption value.

Table 7.4.8: System Test Case for Most Frequent Period

Test Case ID:	8	
Test Case:	Most Frequent Period Display	
Test Steps:	1. Press tab navigation historical energy consumption icon 2. Go to Historical Energy Consumption Page 3. Verify that the most frequent period (e.g., morning, afternoon) is displayed correctly when viewing monthly data 4. Check that the icon displayed next to the most frequent period corresponds to the expected period (e.g., sun icon for morning, moon icon for night)	
Test Result	Expected Result	Actual Result
	The most frequent period is displayed correctly with the appropriate icon	The most frequent period is displayed correctly with the appropriate icon

Table 7.4.9: System Test Case for Total Carbon Emission Fetching

Test Case ID:	9	
Test Case:	Total Carbon Emissions Display	
Test Steps:	1. Press tab navigation historical energy consumption icon. 2. Go to Historical Energy Consumption Page 3. Press View Monthly button. 4. Verify that the total carbon emissions value is displayed correctly when viewing monthly data. 5. Check that the unit of measurement (tons) is displayed along with the total carbon emissions value.	
Test Result	Expected Result	Actual Result
	The total carbon emissions value is displayed correctly with the unit of measurement.	The total carbon emissions value is displayed correctly with the unit of measurement.

Table 7.4.10: System Test Case for Fetching User Data

Test Case ID:	10	
Test Case:	User Data Fetching	
Test Steps:	1. Press tab navigation user profile icon. 2. Go to User Profile Page. 3. Check if user data (username, name, DynamoDB table name) is fetched and displayed correctly.	
Test Result	Expected Result	Actual Result

	1. User data (username, name, DynamoDB table name) is fetched and displayed correctly.	1. User data (username, name, DynamoDB table name) is fetched and displayed correctly.

Table 7.4.11: System Test Case for Updating User Data

Test Case ID:	11	
Test Case:	Update Name	
Test Steps:	1. Press tab navigation user profile icon. 2. Go to User Profile Page. 3. Click on the Update Name button. 4. Enter a new name in the input field. 5. Click on the Confirm button. 6. Check if the name is updated successfully.	
Test Result	Expected Result	Actual Result
	1. Username is updated successfully.	1. Username is updated successfully.

Table 7.4.12: System Test Case for Updating User DynamoDB Table

Test Case ID:	12
Test Case:	Update DynamoDB Table Name
Test Steps:	1. Press tab navigation user profile icon. 2. Go to User Profile Page. 3. Click on the Update DynamoDB Table Name button.

	4. Enter a new DynamoDB table name in the input field. 5. Click on the Confirm button. 6. Check if the DynamoDB table name is updated successfully.	
Test Result	Expected Result	Actual Result
	The DynamoDB table name is updated successfully.	The DynamoDB table name is updated successfully.

Table 7.4.13: System Test Case for Loading Threshold

Test Case ID:	13	
Test Case:	Load Thresholds	
Test Steps:	1. Press tab navigation energy monitor icon. 2. Go to Energy Monitor Page. 3. Check if thresholds are fetched and displayed correctly in the list.	
Test Result	Expected Result	Actual Result
	The thresholds are fetched and displayed correctly in the list.	The thresholds are fetched and displayed correctly in the list.

Table 7.4.14: System Test Case for Create Threshold

Test Case ID:	14
Test Case:	Create Threshold Test
Test Steps:	1. Press tab navigation energy monitor icon 2. Go to Energy Monitor Page 3. Enter a new threshold value and interval in the input fields

	4. Click on the Add Threshold button 5. Check if the new threshold is added to the list and displayed correctly	
Test Result	Expected Result	Actual Result
	The new threshold is added to the list and displayed correctly.	The new threshold is added to the list and displayed correctly.

Table 7.4.15: System Test Case for Remove Threshold

Test Case ID:	15	
Test Case:	Remove Threshold Test	
Test Steps:	1. Press tab navigation energy monitor icon 2. Go to Energy Monitor Page 3. Enter a new threshold value and interval in the input fields 4. Click on the Delete button next to a threshold in the list 5. Check if the selected threshold is removed from the list	
Test Result	Expected Result	Actual Result
	The selected threshold is removed from the list.	The selected threshold is removed from the list.

Table 7.4.16: System Test Case for Edit Threshold

Test Case ID:	16
Test Case:	Edit Threshold Test
Test Steps:	1. Press tab navigation energy monitor icon. 2. Go to Energy Monitor Page.

	3. Press Update button next to a threshold in the list. 4. Enter updated threshold value and time interval value. 5. Check if the selected threshold is updated from the list.	
Test Result	Expected Result	Actual Result
	The selected threshold is updated from the list.	The selected threshold is updated from the list.

Web Based Dashboard

Table 7.4.17: System Test Case for View User Cognito Data

Test Case ID:	17	
Test Case:	Verify Display of Cognito Users Data	
Test Steps:	1. Open a web browser. 2. Navigate to the URL of the Cognito Users page. 3. Observe the page layout and elements. 4. Confirm the presence and accuracy of user data within the table. 5. Validate the functionality of the "Edit" button for each user. 6. Verify the functionality of the "Delete" button for each user. 7. Evaluate the functionality of the "Dashboard" button for each user.	
Test Result	Expected Result	Actual Result
	1. The table correctly showcases user data, including usernames, email addresses,	1. The table correctly showcases user data, including usernames, email addresses,

	addresses, and website URLs. 2. Clicking the "Edit" button enables inline editing of the user's address and website fields. 3. Selecting the "Delete" button prompts a confirmation modal, seeking user confirmation for deleting the corresponding user. The user should be successfully deleted. 4. Clicking the "Dashboard" button redirects the user to their respective dashboard page.	addresses, and website URLs. 2. Clicking the "Edit" button enables inline editing of the user's address and website fields. 3. Selecting the "Delete" button prompts a confirmation modal, seeking user confirmation for deleting the corresponding user. The user should be successfully deleted. 4. Clicking the "Dashboard" button redirects the user to their respective dashboard page.

Table 7.4.18: System Test Case for User Dashboard Functionalities

Test Case ID:	18
Test Case:	User Dashboard Functionality Verification

Test Steps:	1. Open a web browser. 2. Click the specific user dashboard in Cognito user page. 3. Navigate to the URL of the User Dashboard page. 4. Observe the layout and elements of the User Dashboard. 5. Verify the functionality of the daily and monthly energy consumption charts.	
Test Result	Expected Result	Actual Result
	1. User attributes, including Username, WebsocketAPI, and DynamoDB table name, should be displayed accurately. 2. Most Frequent Period and Total Carbon Emissions should be displayed correctly as per the provided data. 3. The daily and weekly charts should visualize energy consumption data effectively, and clicking on data points should	1. User attributes, including Username, WebsocketAPI, and DynamoDB table name, should be displayed accurately. 2. Most Frequent Period and Total Carbon Emissions should be displayed correctly as per the provided data. 3. The daily and weekly charts should visualize energy consumption data effectively, and clicking on data points should

	provide relevant details.	provide relevant details.

Table 7.4.19: System Test Case for Login

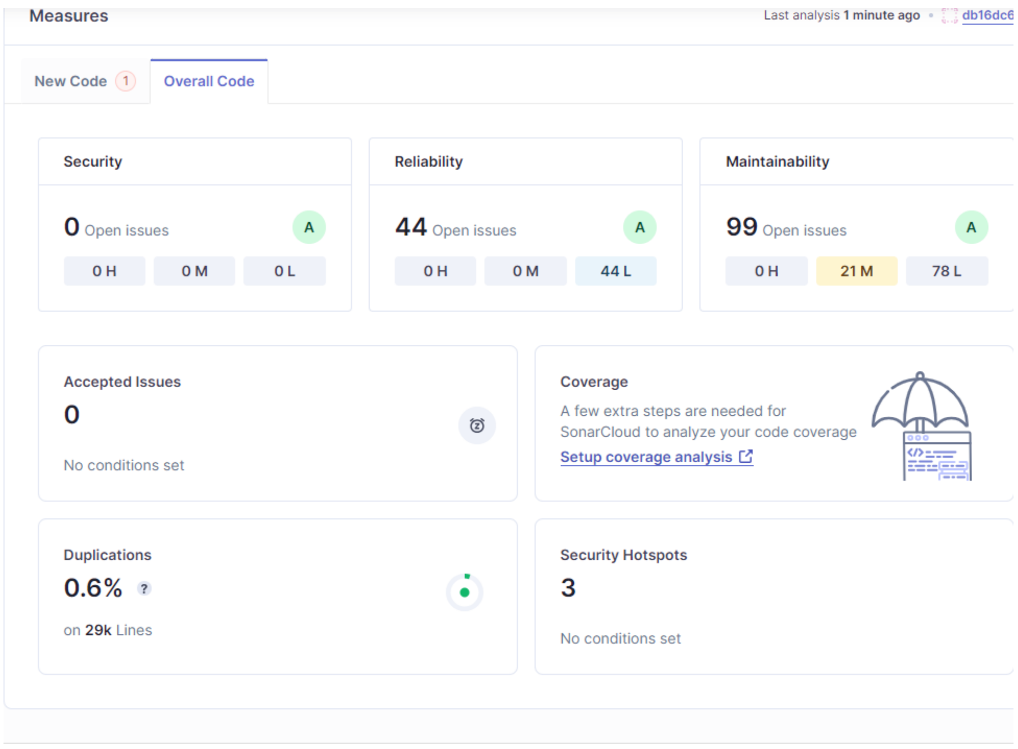
Test Case ID:	19	
Test Case:	Login Test	
Test Steps:	1. Open a web browser. 2. Navigate to the login page of the application. 3. Check if the login page is displayed correctly with the appropriate form fields for email address, password, and "Remember Me" option. 4. Attempt to log in with valid user credentials. 5. Verify that the login process is successful, and the user is redirected to the Cognito user page. 6. Check if the "Forgot Your Password?" link is visible on the login page. 7. Attempt to log in with invalid credentials. 8. Verify that appropriate error messages are displayed for invalid login attempts. 9. Check if the "Remember Me" functionality persists the user's session across browser sessions. 10. Log out from the application. 11. Verify that the user is redirected to the login page after logging out.	
Test Result	Expected Result	Actual Result
	The system should function correctly which allows users to log in with valid credentials and displaying appropriate	The system functioned correctly which allows users to log in with valid credentials and displaying appropriate error

	error messages for invalid login attempts.	messages for invalid login attempts.

7.5 Code Quality Analysis

Code quality analysis examines various aspects of the project code, including security, reliability, maintainability, and duplications. SonarCloud is a cloud-based code analysis service provided by SonarSource that allows developers to continuously inspect the quality of their code by identifying bugs, vulnerabilities, code smells, and other issues that may affect the security, reliability, and maintainability of the software. It is implemented in this project to conduct continuous code quality reviews by analyzing the code quality continuously inside GitHub. SonarCloud will automatically re-examine the code when changes are detected in GitHub when it detects the latest changes.

The figure displays the code quality analysis score, which evaluates various aspects such as security, reliability, and maintainability. In this system, full A grades were achieved in maintainability, security, and reliability, indicating high performance in these areas. Additionally, the system demonstrates a low duplication rate of code, at only 0.6%. This suggests that the system performs exceptionally well in upholding requirements for code quality, guaranteeing strong security procedures, and offering dependable software. A low incidence of duplication indicates that the code is well-organized and free of superfluous repetition, which improves readability and maintainability.



Measures

Last analysis 1 minute ago • db16dc6

New Code 1 | Overall Code

Security	Reliability	Maintainability
0 Open issues A	44 Open issues A	99 Open issues A
0 H · 0 M · 0 L	0 H · 0 M · 44 L	0 H · 21 M · 78 L

Accepted Issues
0
No conditions set

Coverage
A few extra steps are needed for SonarCloud to analyze your code coverage
Setup coverage analysis

Duplications
0.6%
on 29k Lines

Security Hotspots
3
No conditions set

Figure 7.5.1: Result of Code Quality Analysis

7.6 System Usability Scale Evaluation

System Usability Scale evaluation is performed by administering a questionnaire to users after they interact with this project system. This systematic and reliable evaluation can be utilized to examine system usability which aims to surpass the 68-mark industry standard and achieve high usability readiness by evaluating the system usability score based on data collected from novice and expert users (Thomas, 2020). Novice users found for this project evaluation are individuals who are new to home energy monitoring systems and have never used the myTNB app provided by TNB before. They have basic knowledge of energy monitoring but lack proficiency in using such a system. Clear guidance and instructions are given before using the developed system due to their limited technical expertise in related systems. Besides, the expert users found for evaluation are those who have extensive experience with home energy monitoring systems and possess a deep understanding of energy monitoring concepts. They are more interested and focus on the backend configuration and the IoT devices used in this project instead of the user interface. The SUS questionnaire prepared consists of 10 statements for users to rate their agreement level on a scale from 1 to 5, with options ranging from "Strongly Disagree" to Strongly Agree". The questionnaire is listed below:

Table 7.6.1: System Usability Scale Evaluation Questions

	Question	Strongly Disagree 1	2	3	4	Strongly Agree 5
1	I think that I would like to use this system frequently.					
2	I found the system unnecessarily complex.					

3	I thought the system was easy to use.					
4	I think that I would need the support of a technical person to be able to use this system.					
5	I found the various functions in this system were well integrated.					
6	I thought there was too much inconsistency in this system.					
7	I would imagine that most people would learn to use this system very quickly.					
8	I found the system very cumbersome to use.					
9	I felt very confident using the system.					
10	I needed to learn a lot of things before I could get going with this system.					

Next, these scores are transformed and aggregated to measure the usability of this project by formula below:

1. Adjust the score for odd-numbered questions by subtracting 1 from the user's score and for even-numbered questions by subtracting the user's score from 5.
2. Calculate the total score.
3. Multiply the total score by 2.5 to scale it to a range of 0 to 100.

System Usability Scale Evaluation Result

Table 7.6.2: System Usability Scale Evaluation Result

Questions	**Evaluators**						**Average**
	Novice 1	**Novice 2**	**Novice 3**	**Expert 1**	**Expert 2**	**Expert 3**	
1.	3	4	3	3	4	3	3.3/4.0
2.	3	3	2	3	4	3	3.0/4.0
3.	4	3	4	4	4	3	3.7/4.0
4.	4	4	4	2	3	4	3.5/4.0
5.	3	4	2	4	3	3	3.2/4.0
6.	2	4	3	4	2	3	3.0/4.0
7.	4	2	4	3	4	3	3.3/4.0
8.	4	3	4	3	4	4	3.7/4.0
9.	4	4	3	3	2	4	3.3/4.0
10.	3	2	2	4	3	4	3.0/4.0
Total	34	33	31	33	33	34	33.0/ 40.0
SUS Score	85.0	82.5	75.0	82.5	82.5	85.0	82.5

The average system usability scale score is 82.50 which is higher than the industry standard of 68 indicating that the system is good enough for normal user and expert user to use without any issue. The result of system usability evaluation is attached as Appendix C.

All of these tests, including unit testing, integration testing, and system testing for the mobile application and web-based dashboard, passed completely with no errors detected. Additionally, the system usability scale score of 82.50 fulfils the objective of the project.

CHAPTER 8

CONCLUSION

8.1 Conclusions

The project objectives defined in this project were successfully achieved on time with the completion of both the mobile application and a web-based dashboard.

The first objective is to determine project requirements and analyse existing home energy monitoring systems that fulfil company needs. This objective is achieved through the internship program by being involved in the industry-linked company project related to home energy monitoring devices, understanding the company requirements through discussion with their employees, and learning the background of the project. Besides, existing home energy monitoring systems were investigated to know the backend algorithms and user interface for energy consumption visualization.

The second objective is to develop a mobile application that monitors energy usage and consumption. It was achieved by developing a mobile application for residents. This mobile application supports features such as real-time energy consumption monitoring, viewing historical energy consumption data and setting threshold notifications for residents to monitor their energy consumption data.

The third objective is to develop a web-based dashboard that visualizes the energy consumption data of the residents. This objective was fulfilled by developing a web-based dashboard for administrators to manage residents' data and visualize energy consumption data for different residents.

The last objective is to assess the functionalities of mobile application and web-based dashboard through unit testing, integration testing, system testing and system usability scale evaluation while aiming for less than 10% error. This objective was achieved by incorporating users' ratings and feedback from SUS evaluation and collective findings from unit testing, integration testing, and system testing. This contributed to achieving less than 10% error in the evaluation process.

8.2 Recommendations

Despite this project fulfils specifications listed, there are still some limitations in the project.

Firstly, the residents receive notifications via email that is not convenience enough due to this project was developed using Expo for both iOS and Android platforms. However, the push notification function can be implemented in Android platforms but not iOS platform due to the expensive enrolment in Apple Developer Program Membership for iOS signing certificates to implement push notifications functionality. The push notification in Android was removed to maintain consistency in AWS backend architecture across different platforms. It is recommended to purchase Apple Developer Program Membership in future to enable push notifications functionality as this enhancement would significantly improve the suer experience by allowing residents to receive notification directly on their mobile devices.

Next, the AI algorithm designed to predict the residents' energy consumption was removed from this project due to the idealistic data collected from the energy consumption simulator is impractical for commercial use. It is advisable to consider developing AI algorithms which predict residents' energy consumption data collected in DynamoDB table by developing different AI models for specific users with different AWS resources such as AWS Artificial Intelligence service and AWS Lambda.

Furthermore, the details analysis on residents' energy consumption data were not completed as this details analysis was developed inside AWS Quick Sight. This service integrates with AWS S3 and AWS DynamoDB to retrieve latest data automatically and display in both web and mobile component. User can work around on own energy consumption data and select to visualize the data in different format like Excel. However, it is removed from this project due to budget limitation as AWS Quick Sight required around RM 114.96 ringgit monthly to export the chart component out for web dashboard and mobile application. It is recommended to consider integrating this service with AWS backend architecture to offer this function to user in the future.

Lastly, the total energy consumption data can be disaggregated using AI algorithm such is Non-Intrusive Load Monitoring. Different home appliances energy consumption data can be collected with the total energy consumption data to train the AI model to predict the break down energy consumption of each home appliances for further visualization of energy consumption data.

REFERENCES

Abdul Latif, S.N., Chiong, M.S., Rajoo, S., Takada, A., Chun, Y.Y., Tahara, K. and Ikegami, Y., 2021. The trend and status of energy resources and greenhouse gas emissions in the Malaysia Power Generation Mix. *Energies,* [e-journal] 14(8), p. 2200. https://doi.org/10.3390/en14082200.

Ali, S. S. S., 2020. The nexus of population, GDP growth, electricity generation, electricity consumption and carbon emissions output in Malaysia. *International Journal of Energy Economics and Policy.*

Ali, S.S.S., Razman, M.R., Awang, A., Asyraf, M.R.M., Ishak, M.R., Ilyas, R.A. and Lawrence, R.J., 2021. Critical determinants of household electricity consumption in a rapidly growing city. *Sustainability*, 13(8), p.4441.

Amin, M., Shah, H. H., Fareed, A. G., Khan, W. U., Chung, E., Zia, A. and Lee, C., 2022. Hydrogen production through renewable and non-renewable energy processes and their impact on climate change. *International Journal of Hydrogen Energy*, 47(77), 33112-33134.

Apuke, O. D., 2017. Quantitative research methods: A synopsis approach. *Kuwait Chapter of Arabian Journal of Business and Management Review*, 33(5471), 1-8.

Cheng, C., Wang, J., Zhou, Z., Teng, W., Sun, Z., and Zhang, B., 2022. A BRB-Based Effective Fault Diagnosis Model for High-Speed Trains Running Gear Systems. *IEEE Transactions on Intelligent Transportation Systems*, 23(1), pp. 110-121.

Dam, S. S. van, C. A. Bakker, and J. D. M. van Hal., 2010. Home Energy Monitors: Impact over the Medium-Term. *Building Research & Information,* [e-journal] 38(5), 458–469. https://doi.org/10.1080/09613218.2010.494832.

Dominicis, S., Sokoloski, R., Jaeger, C. M. and Schultz, P. W., 2019. Making the smart meter social promotes long-term energy conservation. *Palgrave Communications*, 5(1).

DOSM, 2020. *Household Income and Basic Amenities Survey Report 2019*. [online] Malaysia: Department of Statistics Malaysia. Available at: <www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=120&bul_id=TU00TmRhQ1N5TUxHVWN0T2

VjbXJYZz09&menu_id=amVoWU54UTl0a21NWmdhMjFMMWcyZz09> [Accessed 22 July 2023].

Fagiani, M., Bonfigli, R., Principi, E., Squartini, S. and Mandolini, L., 2019. A non-intrusive load monitoring algorithm based on non-uniform sampling of power data and deep neural networks. *Energies*, 12(7), p.1371.

Faustine, A., Mvungi, N.H., Kaijage, S. and Michael, K., 2017. A survey on non-intrusive load monitoring methodies and techniques for energy disaggregation problem. *arXiv preprint arXiv:1703.00785*.

Gunge, V. S., Yalagi, P. S., 2016. Smart home automation: a literature review. *International Journal of Computer Applications*, 975(8887-8891).

Hart, G.W., 1992. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, *80*(12), pp.1870-1891.

IEA., 2022. Final consumption – Key World Energy Statistics 2021 – Analysis. [online] Paris: International Energy Agency. Available at: <https://www.iea.org/reports/key-world-energy-statistics-2021/final-consumption> [Accessed 10 August 2023].

IEA., 2022. *Electricity Information - Data product*, *IEA*. [online] Paris: International Energy Agency. Available at: <https://www.iea.org/data-and-statistics/data-product/electricity-information> [Accessed: 10 August 2023].

IEA, 2023. *International Energy Agency*. [online] Paris: International Energy Agency. Available at: <https://www.iea.org/countries> [Accessed 22 July 2023].

Janardhana, S. and Deekshit Shashikala, M. S., 2016. Challenges of smart meter systems. *International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pp. 78-82.

Laplante, P A., 2004. *Real-time systems design and analysis[M]*. New York: Wiley.

Ling, Y. M., Hamid, N. A. A., and Te Chuan, L., 2020. Is Malaysia ready for Industry 4.0? Issues and challenges in manufacturing industry. *International Journal of Integrated Engineering*, 12(7), 134-150.

Lee, K., Teng, W. and Hou, T., 2016. Point-n-Press: An Intelligent Universal Remote Control System for Home Appliances. *IEEE Transactions on Automation Science and Engineering*, 13(3), pp. 1308-1317. http://dx.doi.org/10.1109/TASE.2016.2539381.

Lee, K., Romzi, P., Hanaysha, J., Alzoubi, H., and Alshurideh, M., 2022. Investigating the impact of benefits and challenges of IOT adoption on supply chain performance and organizational performance: An empirical study in Malaysia. Uncertain Supply Chain Management, 10(2), 537-550.

Liang, J., Ng, S.K., Kendall, G. and Cheng, J.W., 2009. Load signature study—Part I: Basic concept, structure, and methodology. *IEEE transactions on power Delivery*, *25*(2), pp.551-560.

Madakam, S., Ramaswamy, R., Tripathi, S., 2015. Internet of Things (IoT): A literature review. *Journal of Computer and Communications*, 3(05), 164.

Malaysian Green Technology and Climate Change Centre, 2020. *Energising Sustainability Annual Report 2020*. [pdf] Selangor: Malaysian Green Technology and Climate Change Centre. Available at: < https://www.mgtc.gov.my/wp-content/uploads/2022/06/AnnualReport2020.pdf> [Accessed 14 July 2023].

Mall, R., 2009. *Real-time systems: theory and practice*. Pearson Education India.

Ministry of Economy, 2023. *National Energy Transition Roadmap*. [online] Malaysia: Putrajaya Ministry of Economy. Available at: <https://www.ekonomi.gov.my/sites/default/files/2023-09/National%20Energy%20Transition%20Roadmap_0.pdf > [Accessed 22 April 2024].

Ministry of Energy, Green Technology and Water., 2019. *National Energy Efficiency Action Plan.* [pdf] Putrajaya: Prime Minister's Office of Malaysia. Available at: < https://www.pmo.gov.my/wp-content/uploads/2019/07/National-Energy-Efficiency-Action-Plan.pdf> [Accessed 14 July 2023].

Myces Sdn Bhd, 2023. *Monitoring System (MyCES EMARS)*. [online] Available at: <https://www.mycesgroup.com/service/monitoring-system/> [Accessed 3 September 2023].

Nasir, S. R. M., Ibrahim, A., Hassan, R., Haron, H., Hassan, S. H., Garieb, S. L. S., and Busrah, A., 2020. Awareness and acceptance in using smart meter by energy customers in Malaysia. *Environment-Behaviour Proceedings Journal*, 5(SI2), 35-41.

Nettikadan, D. and Raj, S., 2018. Smart community monitoring system using Thingspeak IoT platform. *International Journal of Applied Engineering Research*, 13(17), pp.13402-13408.

Nonyelum, O. F., 2020. Iterative and incremental development analysis study of vocational career information systems. *International Journal of Software Engineering & Applications (IJSEA)*, 11(5).

Norford, L.K. and Leeb, S.B., 1996. Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms. *Energy and Buildings*, *24*(1), pp.51-64.

Park, S., Kim, H., Moon, H., Heo, J. and Yoon, S., 2010. Concurrent simulation platform for energy-aware smart metering systems. *IEEE transactions on Consumer Electronics*, *56*(3), pp.1918-1926.

Proton unveils new hi-tech engine assembly line in Tanjung Malim., 2022. *PROTON*. Available at: <https://www.proton.com/en/press-release/2022/july/new-hi-tech-engine-assembly-line-in-tanjung-malim> [Accessed 14 July 2023].

Ramadan, R., Huang, Q., Bamisile, O. and Zalhaf, A.S., 2022. Intelligent home energy management using Internet of Things platform based on NILM technique. *Sustainable Energy, Grids and Networks*, *31*, p.100785.

Sachs, J. D., 2016. Implementing the Paris Climate Agreement. Horizons: *Journal of International Relations and Sustainable Development*, (6), 34-47.

Shamshiri, M., Gan, C. K., Baharin, K. A., and Azman, M. A. M., 2019. IoT-based electricity energy monitoring system at Universiti Teknikal Malaysia Melaka. *Bulletin of Electrical Engineering and Informatics*, *8*(2), 683-689.

Smart meters - tenaga nasional berhad., 2023. *Welcome to myTNB Portal*. [online] Available at: <https://www.tnb.com.my/residential/smartmeters> [Accessed 14 July 2023].

SEDA, 2019. *Sustainable Energy Development Authority Malaysia*. [online] Putrajaya: SEDA. Available at: <http://www.seda.gov.my/ > [Accessed 14 July 2023].

SmartZone Smart Home Malaysia | Building Automation System | Fully customise with in-house programmer., 2023. *SmartZone Smart Home Malaysia | Building Automation System | Fully customise with in-house programmer*. [online] Available at: <https://www.smartzone.info/> [Accessed 3 Sep. 2023].

Subhi, M. H. F. M., 2020. Communication Technology Options for Better Customer Experience–The Case of Advanced Metering Infrastructure (AMI) at TNB. *iLEARNed*, 1(1), 16-24.

SURUHANJAYA TENAGA (ENERGY COMMISSION), 2021. *Malaysia Energy Statistics Handbook 2021- Malaysia Energy Information Hub*. [online] Malaysia: SURUHANJAYA TENAGA (ENERGY COMMISSION). Available at: <https://meih.st.gov.my/documents/10620/0112cea0-c76e-4ed0-a1f0-8d920c59c50f> [Accessed 14 July 2023].

Talha, M., Sohail, M., Tariq, R., and Ahmad, M. T., 2021. Impact of oil prices, energy consumption and economic growth on the inflation rate in Malaysia. *Cuadernos de Economía*, 44(124), 26-32.

Tenaga Nasional Berhad, 2011. *Tenaga Link, the newsletter*. Malaysia: Tenaga Nasional Berhad.

TENAGAofficial., 2019. What Is Smart Meter [Video]. YouTube. Available at: <https://www.youtube.com/watch?v=Dq_fGDd_nsc>

TENAGAofficial., 2019. How Does Smart Meter Works [Video]. YouTube. Available at: <https://www.youtube.com/watch?v=Dq_fGDd_nsc>

Thomas, N., 2020. How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website. [online] Available at: <https://usabilitygeek.com/how-to use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/>.

Torriti, J., 2020. *Appraising the economics of smart meters: Costs and benefits*. Routledge.

United Nations Environment Programme, 2009. *Buildings and Climate Change: Summary for Decision Makers*. Available at: <https://wedocs.unep.org/20.500.11822/32152> [Accessed: 23 April 2024].

Yusoff, N. S., Kaman, Z. K., Zahari, A. R., Norafi, W. H. W. M., and Abdullah, A. B., 2021. Examining Smart Meter Users' Experience on Continuance Intention in Adopting Smart Meter in Malaysia-Result from a Pilot Study. *Asia Proceedings of Social Sciences*, 7(2), 110-113.

Zeifman, M. and Roth, K., 2011. Nonintrusive appliance load monitoring: Review and outlook. *IEEE transactions on Consumer Electronics*, *57*(1), pp.76-84.

Zoha, A., Gluhak, A., Imran, M.A. and Rajasegarar, S., 2012. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, *12*(12), pp.16838-16866.

APPENDICES

Appendix A: Site Visit with MIMOS Berhad

mimos.my
You are now at the hottest point of the innovation ecosystem.
National Applied R&D Centre
MIMOS
MIMOS
K-W
Frontier Tec

Figure A-1: Group Photo with MIMOS Representatives, Supervisors, and Students

Figure A-2: Informal Interview with MIMOS Representative

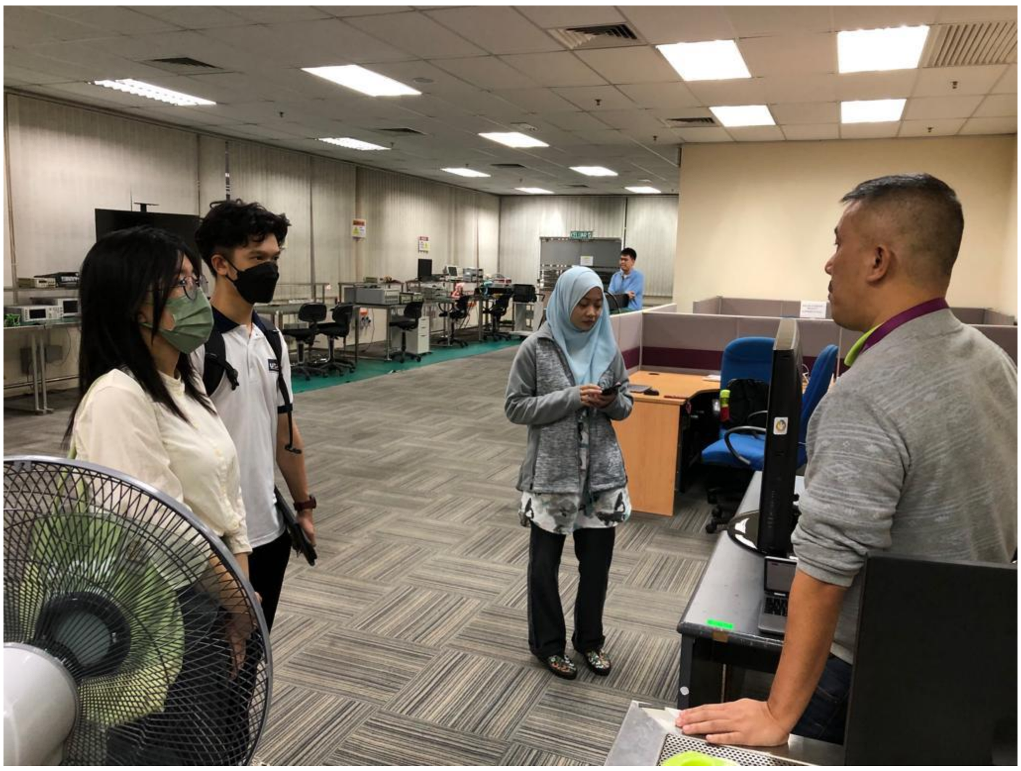


Figure A-3: Demonstration of Home Energy Monitoring Devices

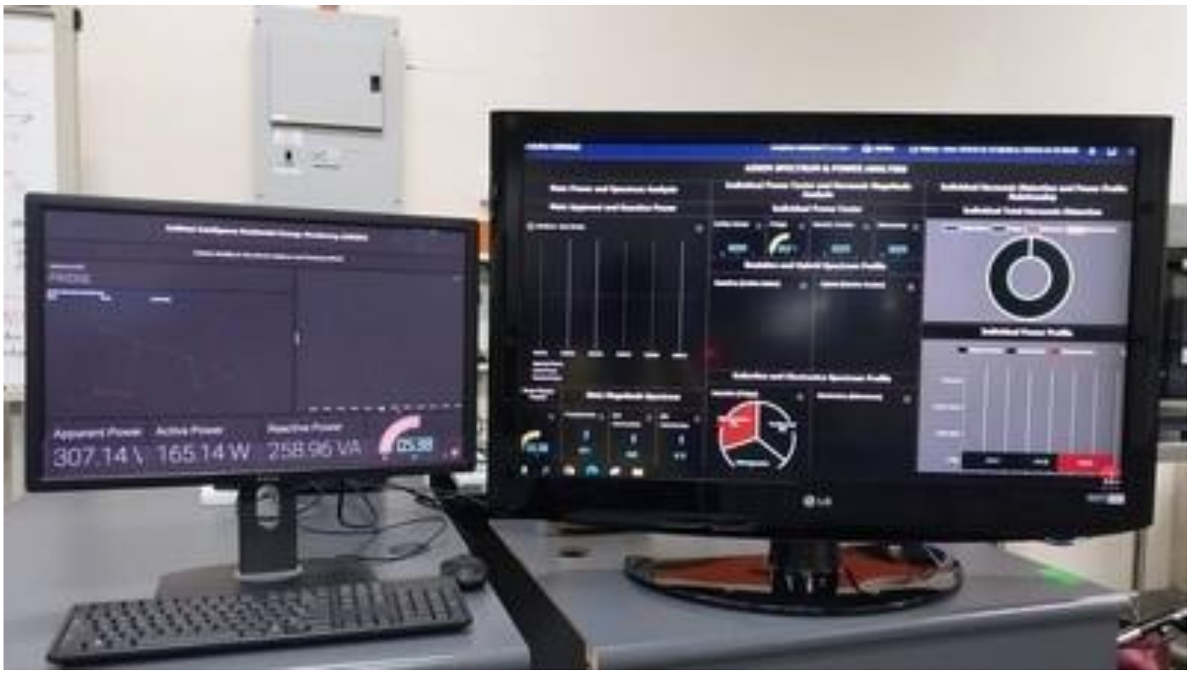


Figure A-4: Demonstration of Home Energy Monitoring Devices Interface

Appendix B: Results and Discussion of Questionnaire

Figure B-1 shows the distribution of age group of respondents from questionnaire. Majority of respondents are from 21 to 40 or 41 to 60 age groups followed by minority age group range from 1 to 20 and 61 to 100. This is because the questionnaire is purposely distributed among residents with own residential or is responsible for paying electricity bills. Most of the respondents in these 2-age range are in economic stable condition with houses and are paying electricity bills themselves.

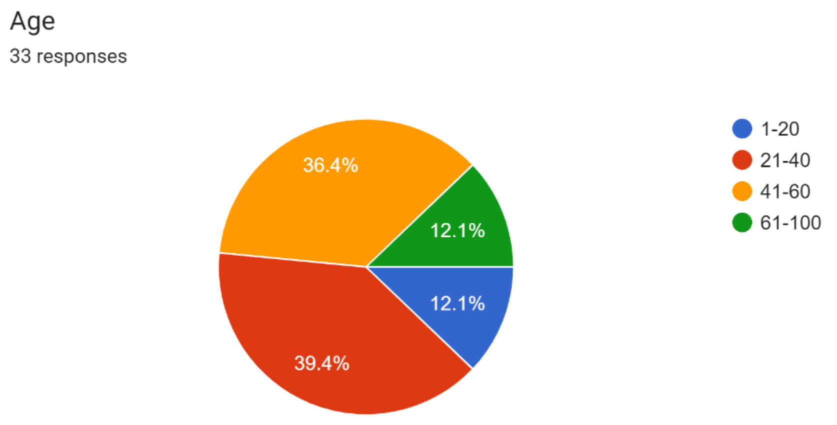


Figure B-1: Different Age Group of Respondents

Next, Figure B-2 describes the gender distribution of the respondents. The number of male and female respondents are close as the questionnaire is sent without concerning the gender of the respondents.

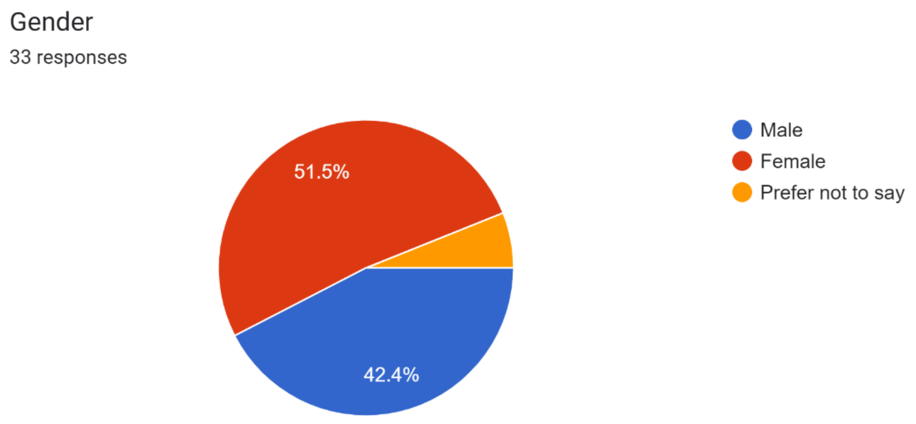


Figure B-2: Different Age Group of Respondents

Gender of Respondents

Moreover, Figure B-8.2.1 presents the states of the respondents collected. Majority of the respondents are from Johor followed by Selangor and Melaka that are 60.6%, 21.2% and 12.1%. The questionnaire tends to distribute among these 3 states as myTNB started to implement its smart meter device to the residents in these 3 states selectively. The residents in these states are more likely to know Home Energy Monitoring System and have a higher potential to use this kind of energy monitoring device before.

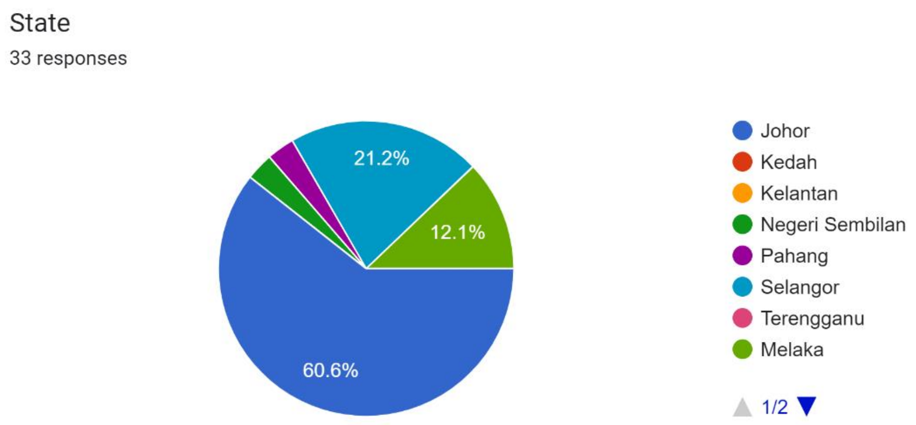


Figure B-8.2.2: Respondents' State

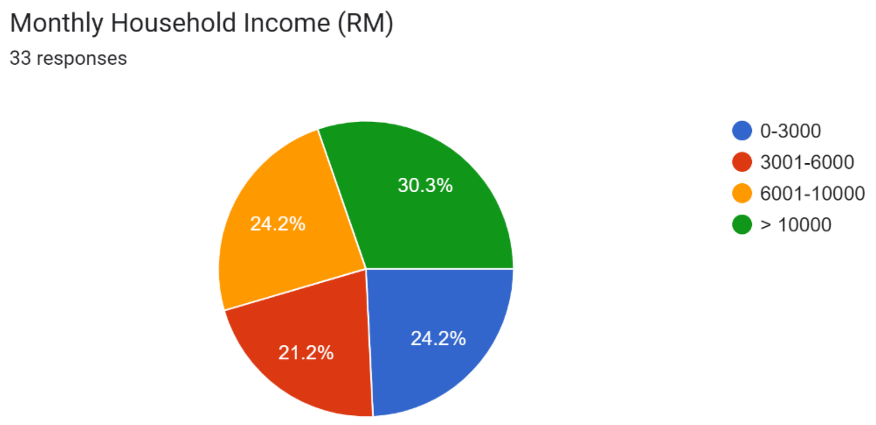


Figure B-4: Monthly household income of respondents

Furthermore, Figure B-4 shows that the monthly household income that is higher than RM10000 stands as majority among all respondents while other respondents are equally distributed among RM 0 to RM 3000, RM 3001 to RM 6000, RM 6001 to RM10000 income groups.

It is known that household income will impact energy consumption pattern as high-income residents might contribute to high energy consumption while low-income households might prioritize energy consumption due to budget constraints. Therefore, it is significant to investigate the relationship between household income, energy consumption, and energy monitoring behaviour.

Next, Figure B-5 presents the number of occupants in respondents' household. 33.3 % of respondents reported that there are 5 people in their household, while 27.3% of respondents are having 3 people living together. As the majority of respondents fall in 21 to 60 age groups, most of them are residing with their family members.

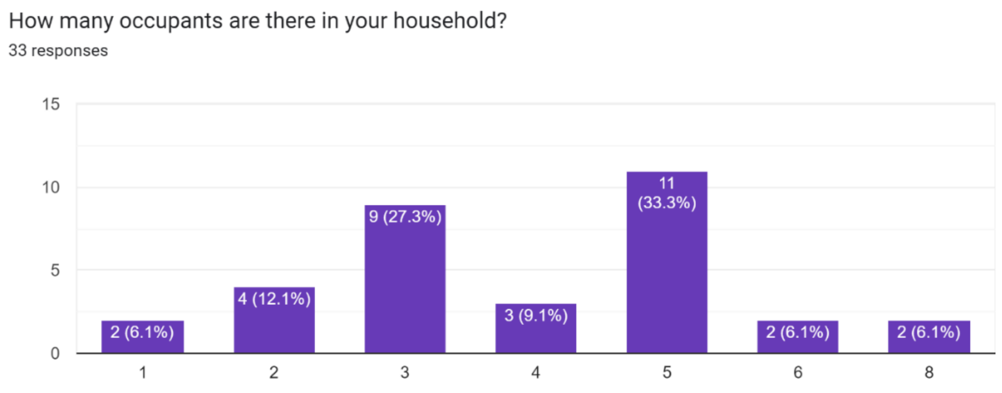


Figure B-5: Occupants in household

Electricity Consumption Awareness and Energy Usage Behaviors

In Figure B-6, majority 36.4% of the respondents are living in town house followed by 30.3% for apartment or condominiums and 27.3% for single-family house. The questionnaire focuses on the respondents who are eligible for implementing Home Energy Monitoring System as these 3 types of housing are having high potential being the target of myTNB to install smart meters in their household.

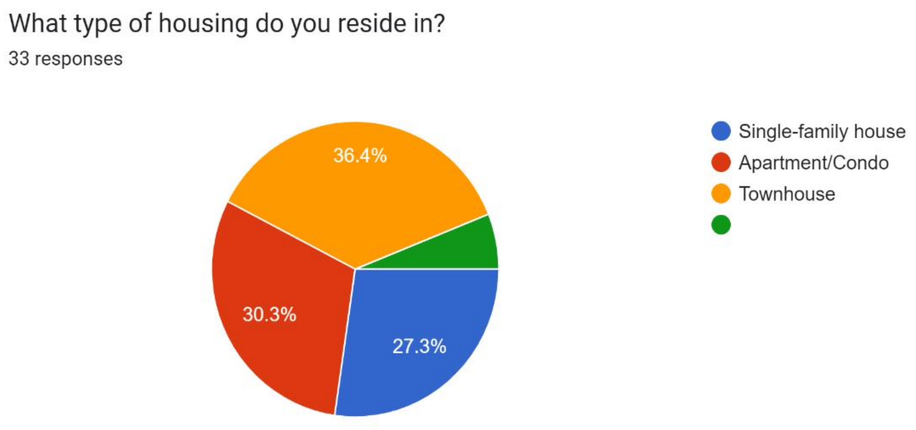


Figure B-6: Type of Respondents' Housing

Figure B-7 indicates that the majority monthly electricity bills amount falls on RM 101 to RM 250 while Figure B-8 shows that almost 97% of the respondents felt that their electricity bills had increased significantly in past year. However, 60.6% of respondents are not aware of the reason of getting high electricity bills while 24.2% of them are having some unconfirmed conjectures about the reasons for the high electricity bills.

Figure B-10 describes that 39.4 % of the respondents will monitor their electricity consumption monthly while 27.3% and 18.2% of respondents rarely and never monitor their electricity consumption. There is a little number of respondents monitor their electricity consumption daily and weekly as the number of respondents who currently using Home Energy Monitoring System is lesser than those without it. The large size of respondents who monitor their energy consumption monthly are viewing their energy usage via monthly electricity bills sent by TNB. From these figures, the developer found that although the respondents feel that their energy consumption is too high, but they rarely seek to find the reasons that contribute to this issue and are less likely to monitor their energy consumption without systematic ways.

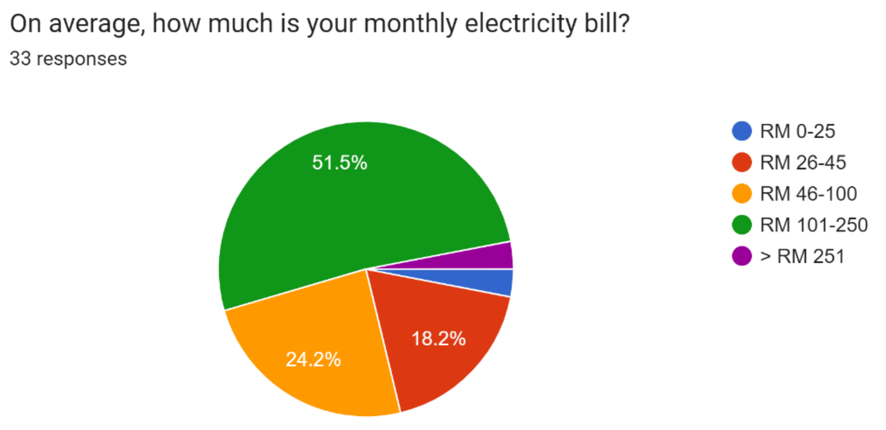


Figure B-7: Monthly Electricity Bills of Respondents

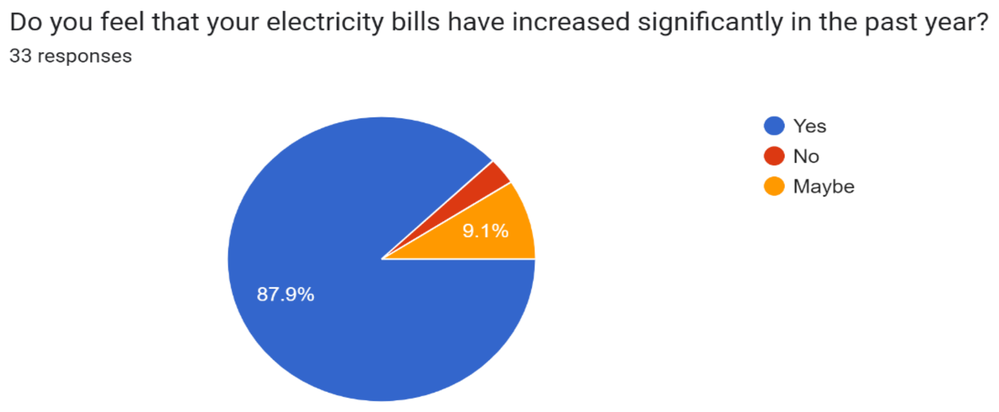


Figure B-8: Awareness of Increasing Electricity Bills

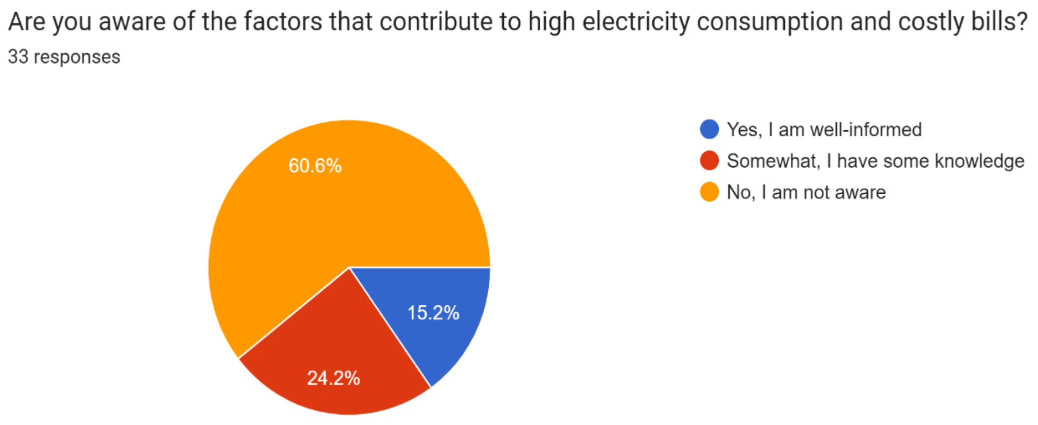


Figure B-9: Awareness of Factors of High Electricity Consumption

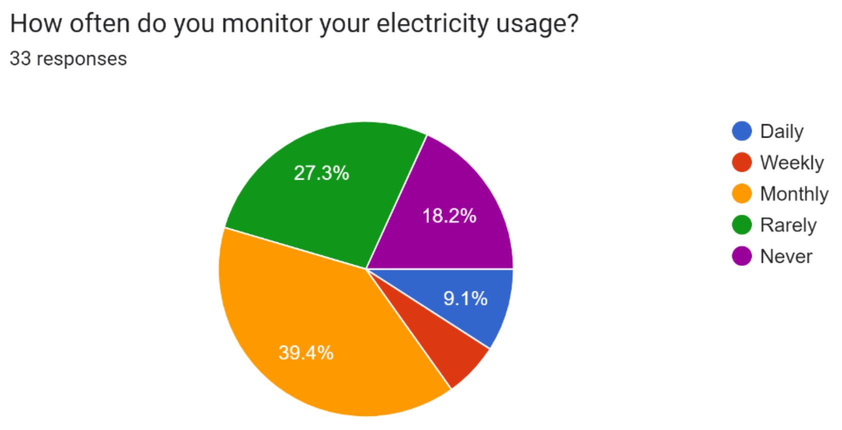


Figure B-10: Period of Monitoring Electricity Usage

Figure B-11 presents the factors that contribute to high electricity consumption. The top 3 factors that cause high electricity usage are excessive usage of energy intensive appliances, leaving electronic appliances in standby mode and high energy usage during peak demand hours. This is because Malaysia is an equatorial country and the weather is generally hot, so the high energy consumption devices such as air conditioners had become an essential household device in Malaysian residential. Besides, significant number of respondents tend to consume more energy during peak hours. Figure 4.12 shows that the majority of the respondents aware of the high electricity consumption device although some of them did not really sure about that.

Figure B-13 indicates that 60.6% of respondents conduct energy practice behaviour sometimes, 21.2 % of them never perform energy practice while 18.2 % of them rarely do this action. This had proved the result above as most of them did not perform energy practices as they leave the electronic devices in standby mode without realizing it can lead to a substantial cumulative energy drain over time. Although they know the reason of high electricity consumption, but the respondents act negatively toward energy-saving practices due to unable to keep track their energy saving behaviour which incline with their energy consumption in standard and easy visualize format.

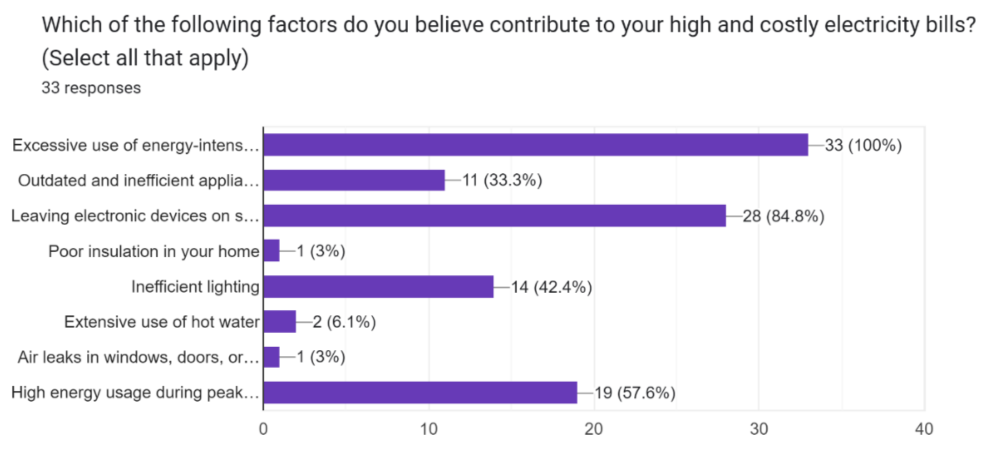


Figure B-11: Factors that contribute to high electricity consumption

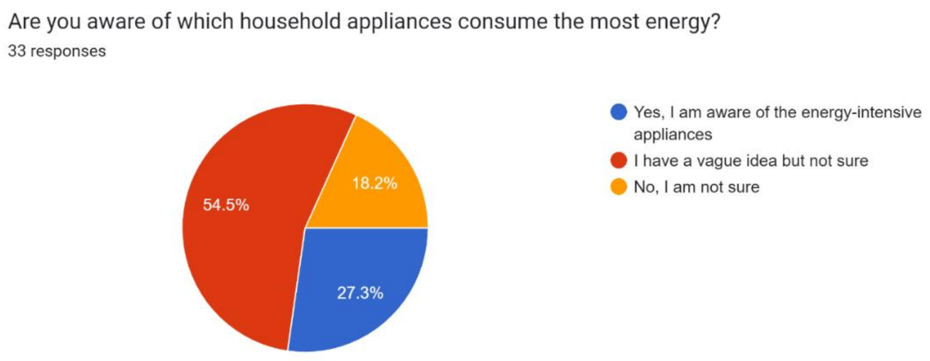


Figure B-12: Awareness of High Electricity Consumption Household Appliances

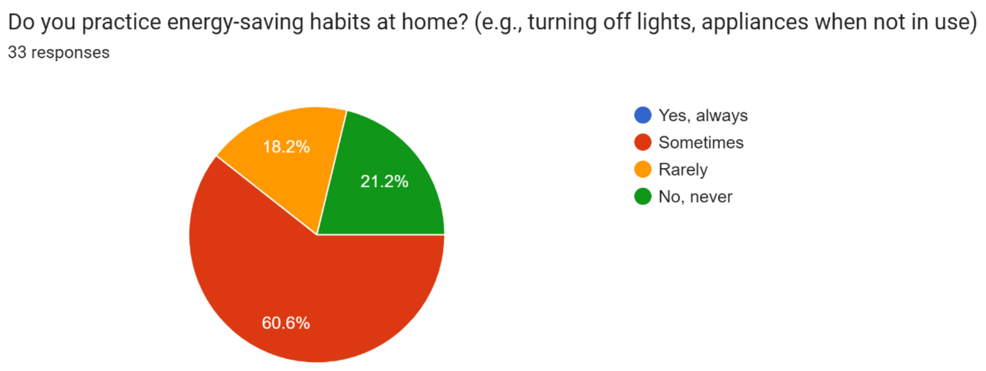


Figure B-13: Practice of Energy-Saving Habits at Home

Home Energy Monitoring System

Figure B-14 indicates that 78.8% of respondents did not use energy management or monitoring tools before while 21.2 % of respondents had experiences in using energy management tools. This is because these kinds of tools are not popular in Malaysia due to high initial implementation cost and neglection of high energy consumption behaviours among residents. Half of the respondents claimed that these kinds of tools did not help them to understand their electricity consumption patterns better while 28.6% of them claimed that they understand their electricity usage patterns better and 21.4% of them thought that it did help slightly in understanding energy usage patterns. This is because the energy management tools provided in market rarely perform analysis on energy usage patterns or it only show a direct energy consumption value to the residents which is hard to investigate and understand without proper knowledge about electricity. Figure B-16 shows that 50% of the respondents thought that the implementation of Home Energy Monitoring System improved their overall energy efficiency while 7.1% of respondents claimed that it helps in identifying high-consumption devices. Home Energy Monitoring System can help residents to visualize their energy consumption data which indirectly motivates them to perform energy saving practices by providing real-time energy consumption feedback.

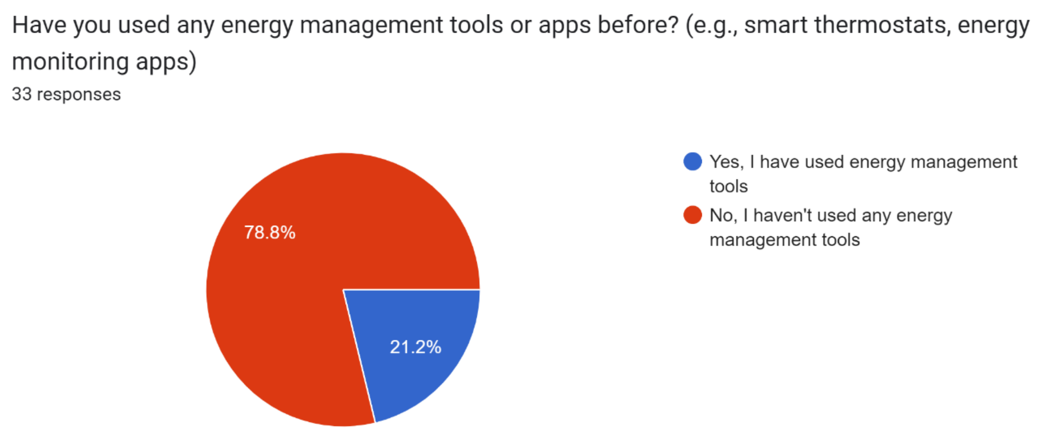

Have you used any energy management tools or apps before? (e.g., smart thermostats, energy monitoring apps)
33 responses

Figure B-14: Use of Energy Monitoring or Management Tools

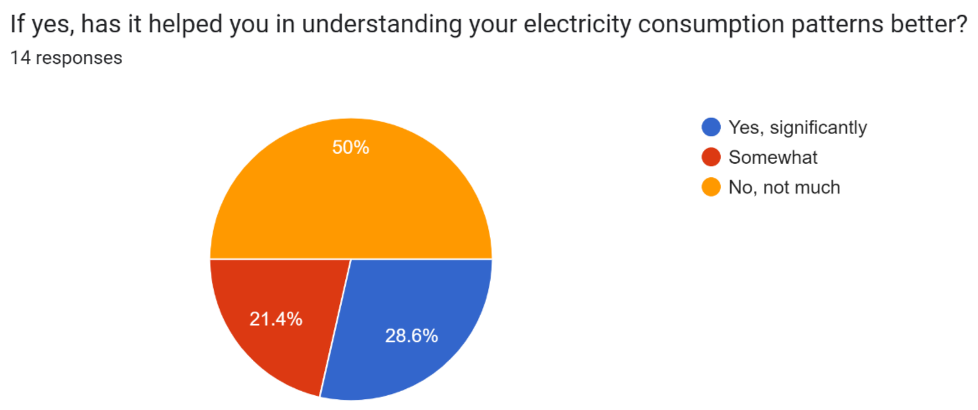


Figure B-15: View on Energy Monitoring Tools

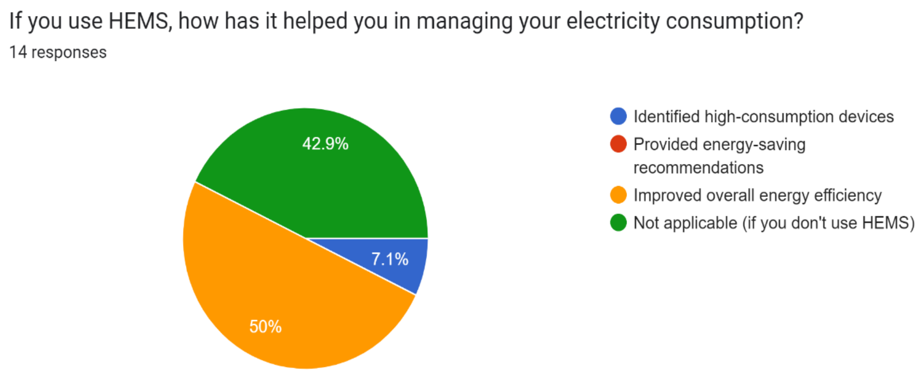


Figure B-16: Effectiveness of Energy Management Tools

Behaviours toward Home Energy Monitoring System

According to Figure B-17, factors encouraged respondents to use Home Energy Monitoring System are presented. 90.9% of the respondents claimed that Real-time monitoring of energy consumption may attract them to use this system most. This indicates that respondents are interested in having immediate insights into their energy consumption. Implementing real-time energy consumption monitoring features can greatly improve the usage of Home Energy Monitoring System. Besides, 72.7% of them believed that potential cost savings on electricity bills will encourage them to use Home Energy Monitoring System. Residents are motivated by potential financial savings to understand own energy usage habits that impact their electricity bills and suggestions to reduce electricity cost. Next, 60.6% respondents thought that energy-savings recommendations provided will be an essential benefit that should be provided by

Home Energy Monitoring System. Respondents tend to seek guidance on the ways to reduce their energy consumption so feature that analyse residents' energy usage pattern and provides tailored recommendation for energy-saving habits will be crucial in this project. These 3 factors will be incorporated in this project with features that can fulfil this factor will be implemented in this system.

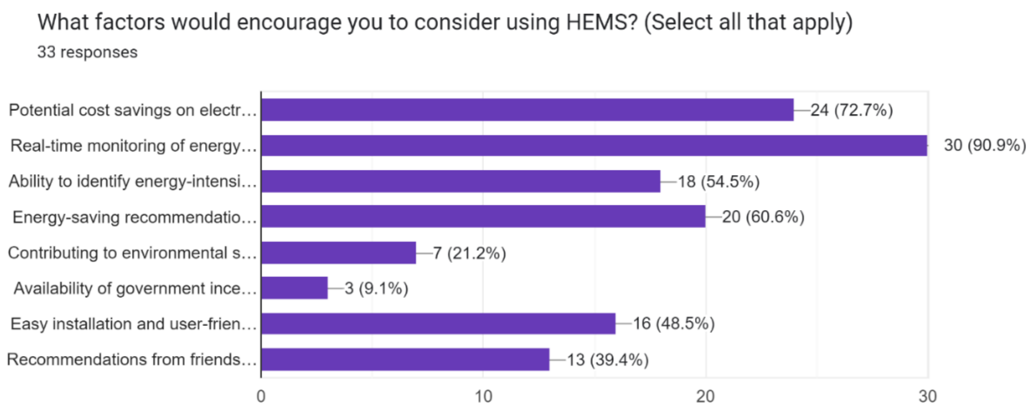


Figure B-17: Factors that Encourage Respondents to Use Home Energy Monitoring System

Figure B-18 shows the respondents' concern on using Home Energy Monitoring System. 78.8% of them considered the initial cost of implementing HEMS. This concern is understandable as Figure B-19 and Figure B-20 proved that majority of respondents are not familiar with Home Energy Monitoring System and having no or limited knowledge about it. Next, 48.5 % of the respondents are afraid that implementing Home Energy Monitoring System might potentially increase their electricity bills due to monitoring as they did not know the workflow of this system. If the respondents are not well-informed about the benefits and workflow of this system, respondents are not willing to spend cost on this system due to uncertainty of returns. It is crucial to address the misconception of Home Energy Monitoring System by emphasizing the importance of addressing respondents' fears of increased electricity bills due to monitoring. As reported in previous session, most of the respondents gained energy saving information online or are not interested in it at all, due to lack of information about Home Energy Monitoring System in Internet and uninterested behaviour towards this issue, the lack of information lead to apprehension and hesitation among potential users. A simple user guide and user-friendly designs might be implemented to ensure that the residents acknowledge about the use and benefit of Home Energy Monitoring System.

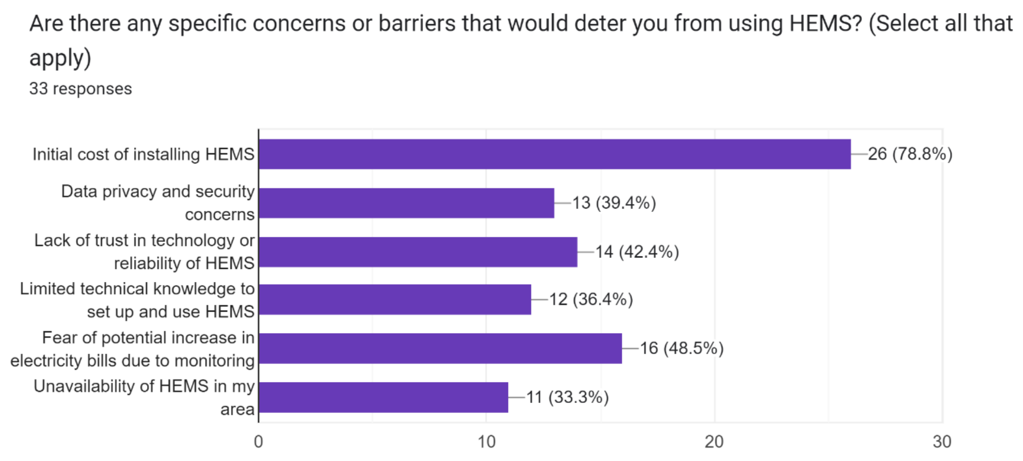


Figure B-18: Respondents' Concern on using Home Energy Monitoring System

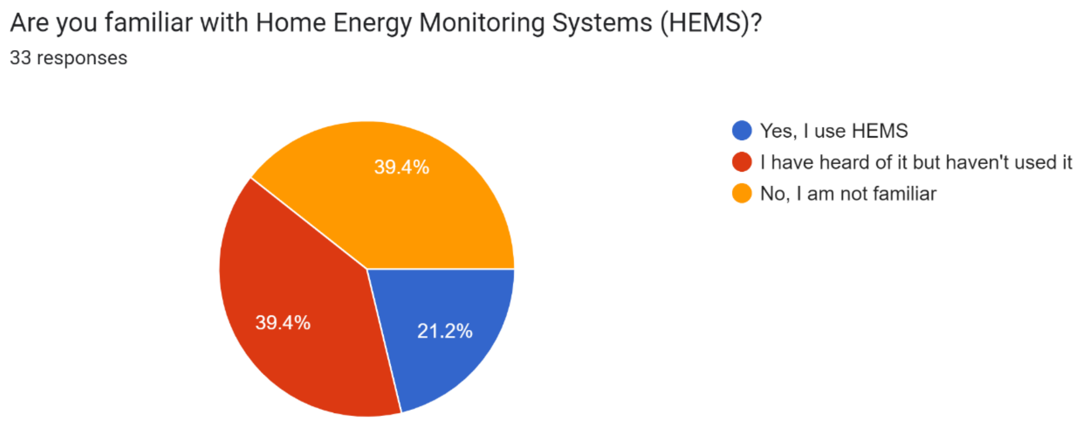


Figure B-19: Familiarity of Home Energy Monitoring System

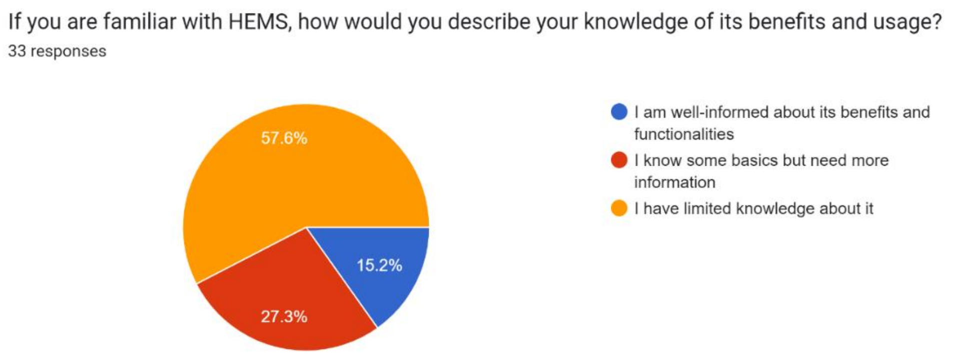


Figure B-20: Knowledges about Home Energy Monitoring System

Figure B-21 reported that 93.21% respondents indicated that smartphone app for remote monitoring is most important aspect for a Home Energy Monitoring System. Next, 84.8% of them highlighted that the most important feature of this system is real-time energy monitoring. A real-time energy monitoring smartphone app will be selected to implemented in this project.

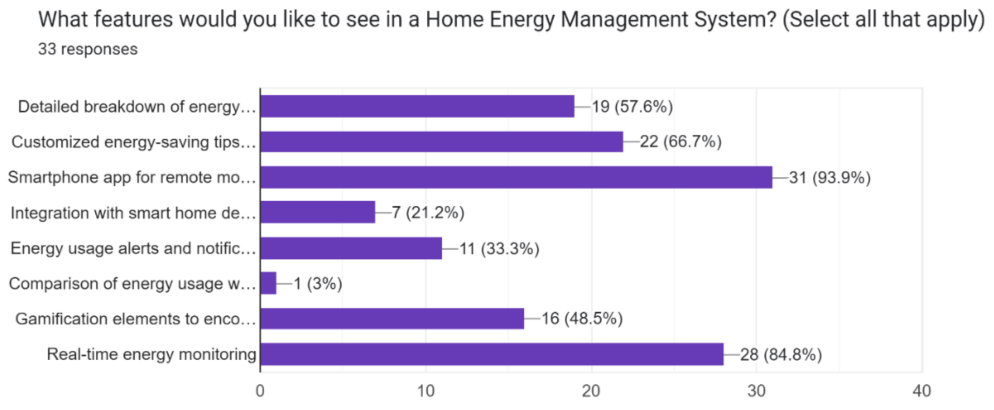


Figure B-21: Important Indicators for Home Energy Monitoring System Feature

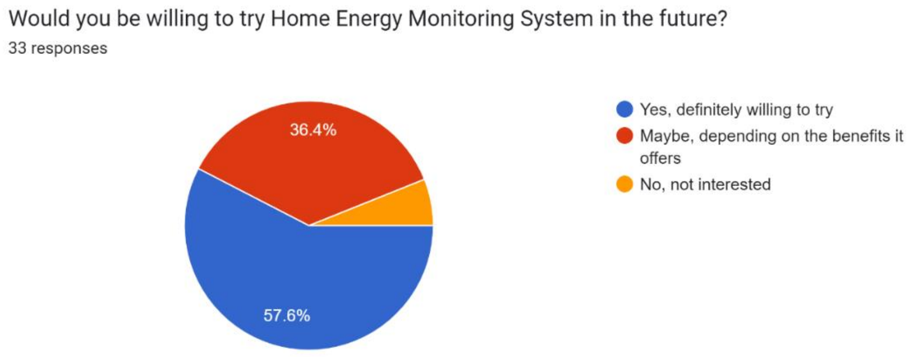


Figure B-22: Willingness of Using Home Energy Monitoring System

In conclusion, majority of the respondents reacted positively toward the implementation of Home Energy Monitoring System if it really helps them in energy efficiency.

Appendix C: System Usability Scale Evaluation Result

I think that I would like to use this system frequently.
（7 条回复）

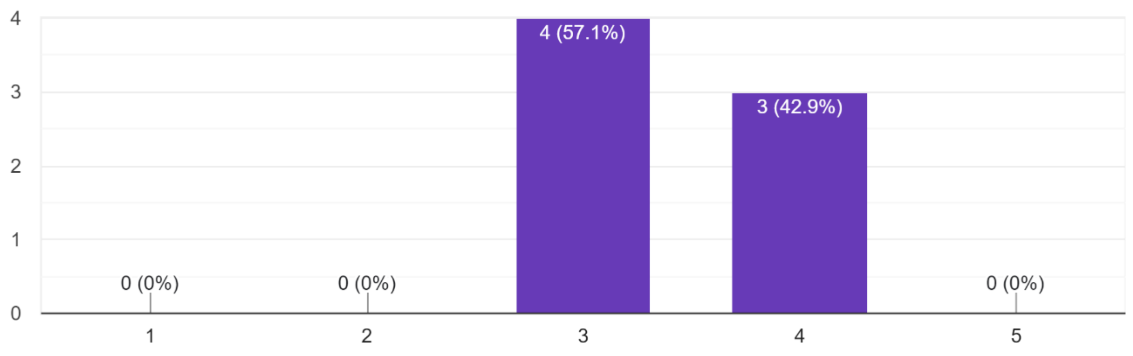


Figure C-1: System Usability Scale Q1

I found the system unnecessarily complex.
（7 条回复）

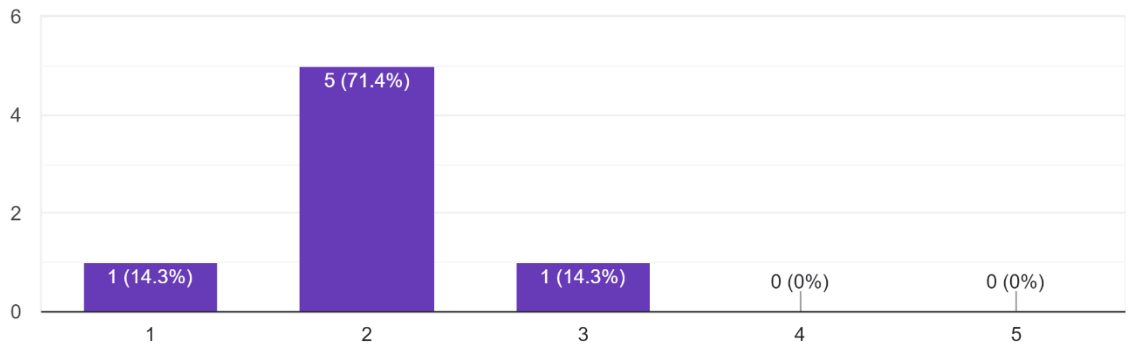


Figure C-2: System Usability Scale Q2

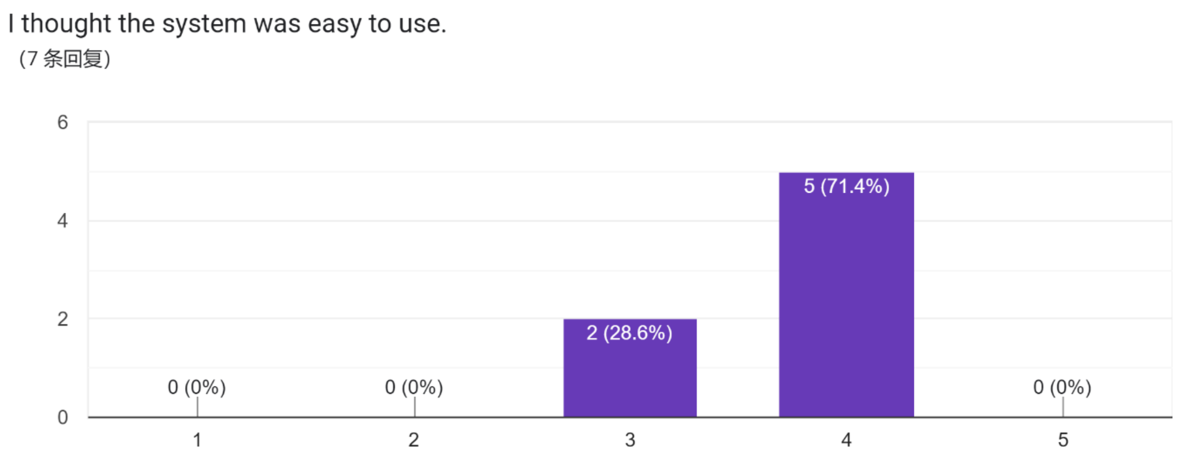


Figure C-3: System Usability Scale Q3

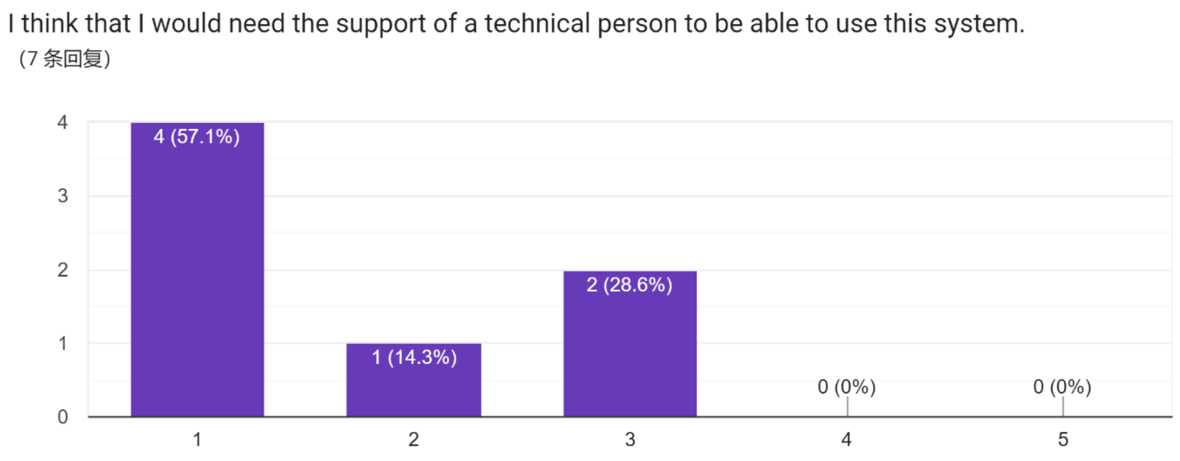


Figure C-4: System Usability Scale Q4

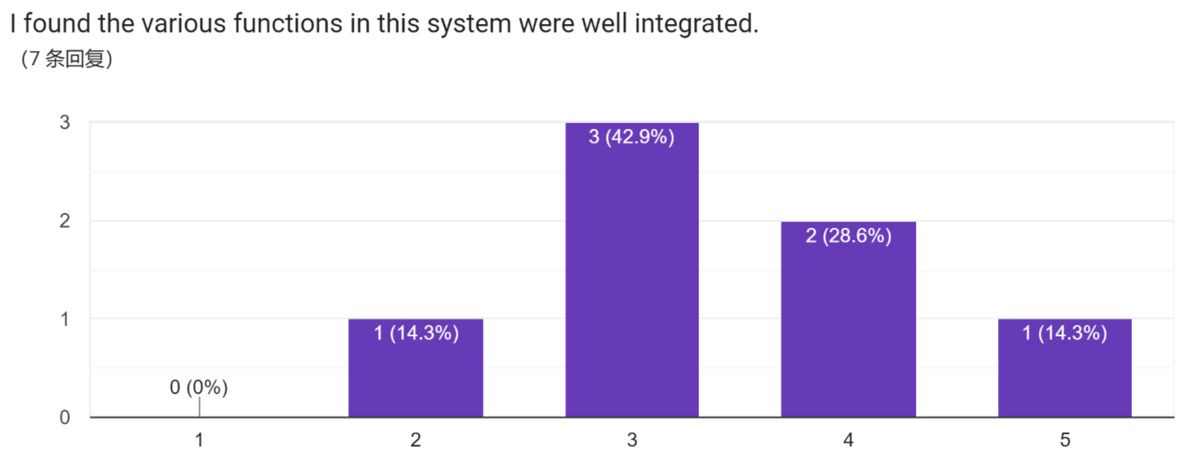


Figure C-5: System Usability Scale Q5

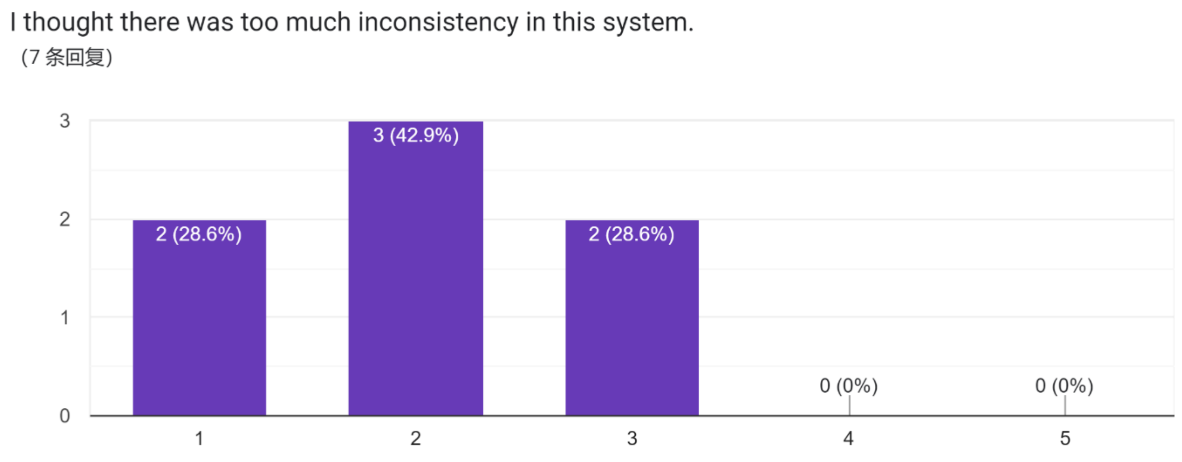


Figure C-6: System Usability Scale Q6

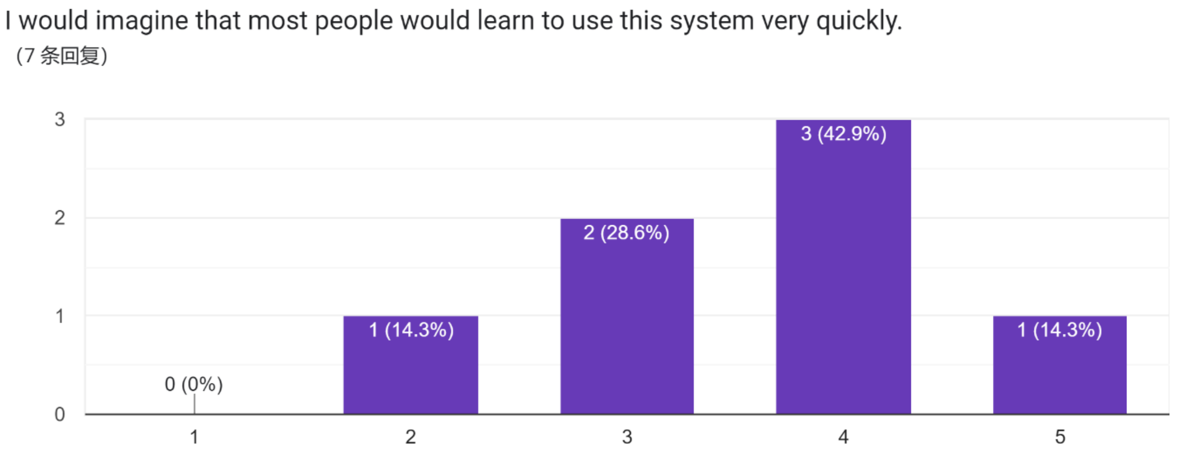


Figure C-7: System Usability Scale Q7

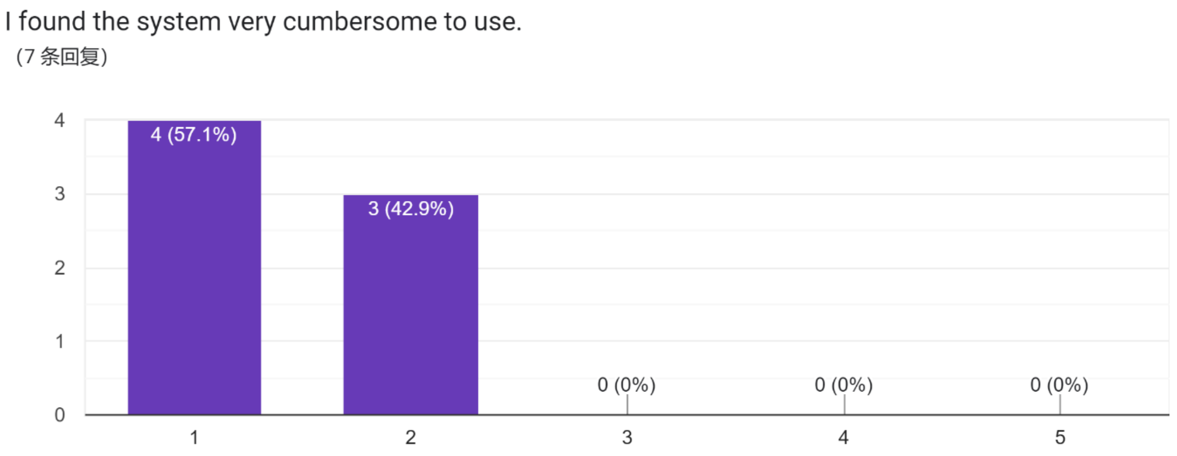


Figure C-8: System Usability Scale Q8

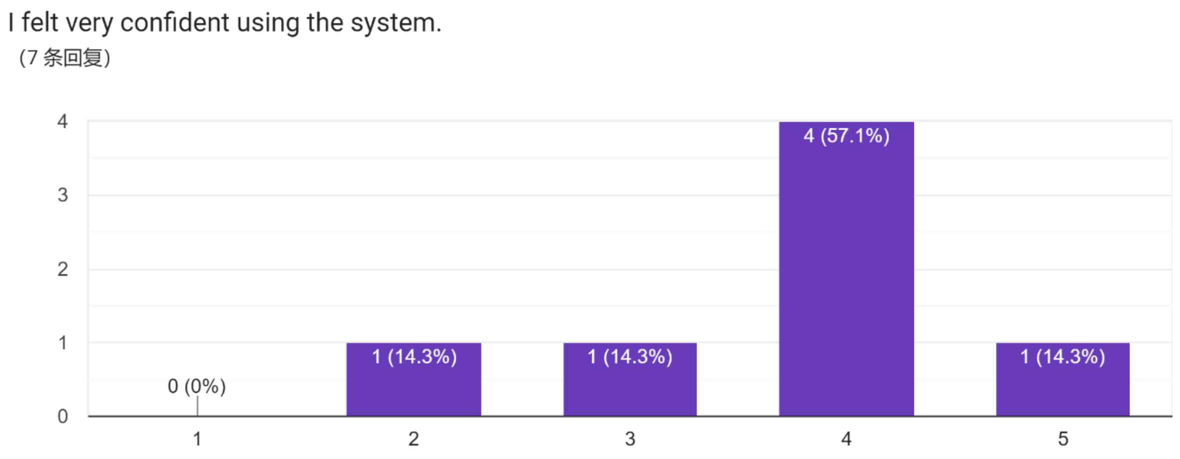

I felt very confident using the system.
(7 条回复)

Figure C-9: System Usability Scale Q9

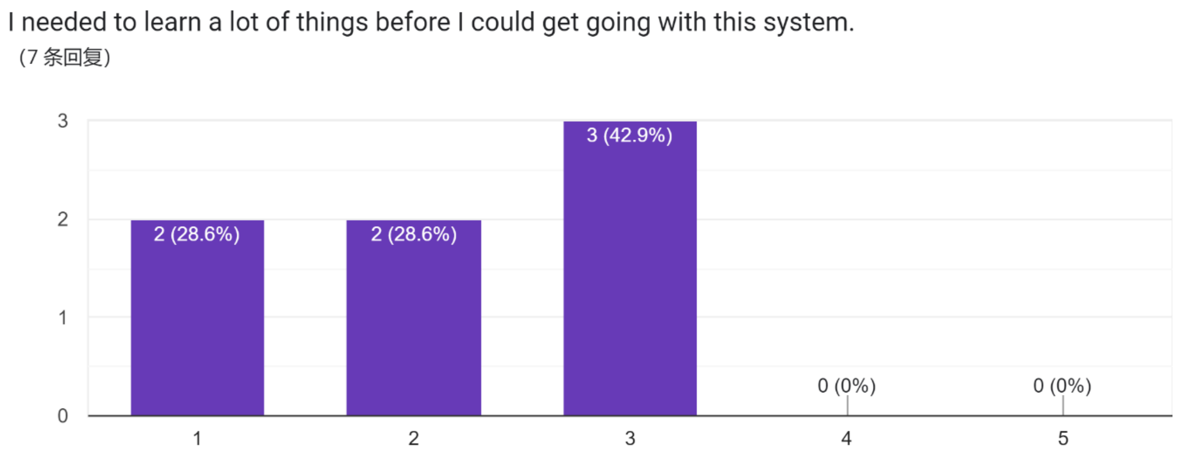

I needed to learn a lot of things before I could get going with this system.
(7 条回复)

Figure C-10: System Usability Scale Q10