

**A MOBILE APP FOR COMMUNITY ASSOCIATION**

**GOH CHONG XIAN**


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

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

**April 2020**

## 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 :   
Name : Goh Chong Xian  
ID No. : 1604378  
Date : 23 April 2020

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled “**A MOBILE APP FOR COMMUNITY ASSOCIATION**” was prepared by **GOH CHONG XIAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Hons) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ms Chean Swee Ling

Date : 23 April 2020

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2020, Goh Chong Xian. All right reserved.



## ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I offer my sincere appreciation to my research supervisor, Ms Chean Swee Ling 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 throughout the process of preparing and completing this report. The completion of this project could not have been accomplished without the support of them.

Lastly, I would like to thank all the authors of the documentations I referred and the library I used. Their professional knowledges and selfless contributions had helped me solve many obstacles I encountered throughout the whole project.

## ABSTRACT

Existing residential community association management is a hassle and manual process as it involves many handiworks that require a large number of time and cost. These behind time operations had caused decreasing in efficiency and effectiveness, which results in bad influences on the residence' image. Therefore, this project aims to investigate and analyse the root causes of the predicament and provide a solution to it. The main causes discovered including heavily rely on paperwork, outdated operational workflow, poor privacy concerns on personal information of residents, etc. A system that involving Ruby on Rails API backend server, Vue.js web admin panel, Flutter mobile application for residents was proposed to tackle the aforementioned problems. Furthermore, Kanban Agile methodology was practiced throughout the implementation phase to manage work tasks and visualize the project progress. The proposed system reduced the manual work of fee payment, increase the announcement delivers effectiveness by instant notification of the mobile application. In addition, the mobile application provides an official channel for residents to submit their feedback and visitor application and track the progress.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF FIGURES</b>	<b>xiii</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>xx</b>
<b>LIST OF APPENDICES</b>	<b>xxi</b>

### CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>22</b>
	1.1 Introduction	22
	1.2 Background of the Project	22
	1.3 Problem Statement	23
	1.4 Project Objectives	24
	1.5 Project Solution	24
	1.6 Project Approach	27
	1.7 Scope of the Project	28
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>31</b>
	2.1 Introduction	31
	2.2 Review on Similar Community Association System	31
	2.2.1 Cloud-based System	31
	2.2.2 MyTaman Community Application	33
	2.2.3 Homeserva Smart Community System	34
	2.3 Demands of System and Solutions	35

2.4	Review on Project Methodology	38
2.4.1	Waterfall	38
2.4.2	Scrum in Agile	39
2.4.3	Kanban in Agile	40
2.4.4	Prototyping	41
2.4.5	Extreme Programming	42
2.4.6	Comparison of Methodology	43
2.5	Review on Backend Server Framework	45
2.5.1	Ruby on Rails	45
2.5.2	Laravel	46
2.5.3	Node.js	46
2.5.4	Comparison of Backend Framework	47
2.6	Review on Front-end Web Application Framework	48
2.6.1	React	48
2.6.2	Vue.js	49
2.6.3	Comparison of Front-end Framework	50
2.7	Review on Cross-platform Mobile Application Framework	51
2.7.1	Flutter	51
2.7.2	React Native	52
2.7.3	Comparison of Hybrid Mobile App Framework	52
2.8	Review on Cloud Computing Services	53
2.8.1	Amazon Web Services	54
2.8.2	Google Cloud Platform	54
2.8.3	Microsoft Azure	55
2.8.4	Comparison of Cloud Computing Services	56
<b>3</b>	<b>METHODOLOGY AND WORK PLAN</b>	<b>58</b>
3.1	Introduction	58
3.2	Development Methodology	58
3.3	Proposed Workplan	61
3.4	Technology and Development Tools Involved	67
3.4.1	Ruby on Rails	67
3.4.2	Vue.js	68

3.4.3	Flutter	68
3.4.4	Amazon Web Services	69
3.4.5	Heroku	69
3.4.6	Firebase Cloud Messaging	69
3.4.7	RQRCode	70
3.4.8	Visual Studio Code	70
3.4.9	Git	70
3.4.10	Trello	71
3.4.11	Axure	71
<b>4</b>	<b>PROJECT SPECIFICATION</b>	<b>72</b>
4.1	Introduction	72
4.2	Fact-finding	72
4.2.1	Interview	72
4.2.2	Observation	73
4.2.3	Questionnaire	74
4.3	Requirement Specification	80
4.3.1	Mobile Application for Community Resident	80
4.3.2	Web-based Application for Management User	82
4.3.3	Mobile Application for Security Personnel	84
4.4	Use Case Modelling	85
4.4.1	Use Case Diagram	85
4.4.2	Use Case Description	91
<b>5</b>	<b>SYSTEM DESIGN</b>	<b>104</b>
5.1	Introduction	104
5.2	System Architecture	104
5.3	Software Design Pattern	106
5.4	Database Design	108
5.4.1	Physical Entity Relationship Diagram	108
5.4.2	Logical Entity Relationship Diagram	109
5.4.3	Data Dictionary	110
5.5	User Interface Design	125

	5.5.1	Web Application for Community Management	125
	5.5.2	Mobile Application for Community Resident	143
	5.5.3	Mobile Application for Security Personnel	156
<b>6</b>		<b>SYSTEM IMPLEMENTATION</b>	<b>159</b>
	6.1	Backend Server	159
	6.1.1	Overview of Backend Server	159
	6.1.2	Controller Layer	161
	6.1.3	Model Layer	170
	6.1.4	Special Integration	173
	6.1.5	Available Endpoints	179
	6.1.6	Deployment	184
	6.2	Web Application for Community Management	185
	6.2.1	Overview of Web Application	185
	6.2.2	Pages Hierarchy	187
	6.2.3	API Client	188
	6.2.4	Deployment	189
	6.3	Mobile Application for Community Residents and Security Personnel	190
	6.3.1	Overview of Mobile Application	190
	6.3.2	Screen Hierarchy	192
	6.3.3	State Management	193
	6.3.4	Deployment	196
<b>7</b>		<b>SYSTEM TESTING</b>	<b>197</b>
	7.1	Introduction	197
	7.2	Unit Testing	197
	7.2.1	Admin Module	197
	7.2.2	Mobile Module	206
	7.3	Integration Testing	210
	7.4	Usability Testing	215
<b>8</b>		<b>CONCLUSION AND RECOMMENDATION</b>	<b>219</b>

8.1	Conclusion	219
8.2	Future Implementation	220
<b>REFERENCES</b>		<b>221</b>
<b>APPENDICES</b>		<b>225</b>

**LIST OF TABLES**

Table 2.1 User reviews of MyTaman application at Google Play Store	34
Table 2.2: Table of comparison between multiple methodologies	43
Table 2.3 Comparison of Ruby on Rails, Laravel and Node.js	47
Table 2.4 Comparison of React and Vue.js	50
Table 2.5 Comparison between Flutter and React Native	52
Table 2.6 Comparison of Amazon Web Services, Google Cloud Platform and Microsoft Azure	56
Table 3.1: Project Schedule Summary	61
Table 6.1 Admin Endpoints Listing	179
Table 6.2 Mobile Endpoints Listing	182
Table 7.1 Test Cases for Admin Module	197
Table 7.2 Test Cases for Admin Module	206
Table 7.3 Test Suites for Integration Testing	210
Table 7.4 Usability Testing Participant Checklist	215
Table 7.5 Post-study Usability Questionnaire	216
Table 7.6 Average Time Usage of Task	217
Table 7.7 Post-study usability questionnaire average score	218



## LIST OF FIGURES

Figure 1.1: Proposed solution technologies involved modules diagram	25
Figure 1.2: Personal Kanban digital board by Trello visualization	27
Figure 2.1: The Six Stages of Waterfall Methodology	38
Figure 2.2: Diagram of Prototyping Methodology	41
Figure 2.3: Diagram of Prototyping Methodology	42
Figure 2.4 Gartner "magic quadrant"	57
Figure 3.1: Iteration of Agile	59
Figure 3.2: Part 1 of project Gantt Chart	63
Figure 3.3: Part 2 of project Gantt Chart	64
Figure 3.4: Work breakdown structure of the project	65
Figure 4.1: Pie chart of the residence type	74
Figure 4.2: Pie chart of percentage of residence having a	75
Figure 4.3: Pie chart of percentage of resident encounter fee payment	75
Figure 4.4: Pie chart of preferable payment method	76
Figure 4.5: Pie chart of preferable way to manage resident's information	76
Figure 4.6: Bar chart of problem encountered in feedback/complaint process	77
Figure 4.7: Pie chart of anonymous feedback	77
Figure 4.8: Pie chart of miss announcement	78
Figure 4.9: Pie chart of online announcement versus normal announcement	78
Figure 4.10: Pie chart of visitor system	79
Figure 4.11 Fee related use case diagram	85

Figure 4.12 Announcement related use case diagram	86
Figure 4.13 Feedback related use case diagram	87
Figure 4.14 Visitor related use case diagram	88
Figure 4.15 Resident information related use case diagram	89
Figure 4.16 Account related use case diagram	90
Figure 5.1 Overview of System Architecture	104
Figure 5.2 MVC Flow Visualization	106
Figure 5.3 Completed physical entity relationship diagram	108
Figure 5.4 Simplified Logical entity relationship diagram	109
Figure 5.5 Login Page of Web Application	125
Figure 5.6 Recover Password Page of Web Application	126
Figure 5.7 Home Analytics Page of Web Application	126
Figure 5.8 Fee list in table	127
Figure 5.9 Create new fee page	127
Figure 5.10 Edit fee page	128
Figure 5.11 Announcement list in table	128
Figure 5.12 Create new announcement page	129
Figure 5.13 Edit announcement page	129
Figure 5.14 Feedback list in table	130
Figure 5.15 Create feedback page	130
Figure 5.16 Update feedback page	131
Figure 5.17 Update feedback dialog	131
Figure 5.18 Complete feedback dialog	132
Figure 5.19 Completed feedback page	132
Figure 5.20 Visitor list in table form	133

Figure 5.21 Create new visitor page	133
Figure 5.22 Update visitor page	134
Figure 5.23 QR code generated for visitor	134
Figure 5.24 Payment histories list in table	135
Figure 5.25 View payment history page	135
Figure 5.26 House list in table	136
Figure 5.27 Create house page	136
Figure 5.28 Select resident dialog	137
Figure 5.29 Delete house dialog	137
Figure 5.30 Resident list in table	138
Figure 5.31 Create resident page	138
Figure 5.32 Select house dialog	139
Figure 5.33 Edit resident page	139
Figure 5.34 Delete resident dialog	140
Figure 5.35 Settings page	140
Figure 5.36 General information of residence page	141
Figure 5.37 Staffs page	141
Figure 5.38 Profile page	142
Figure 5.39 Login screen for mobile application	143
Figure 5.40 Recover password screen	143
Figure 5.41 Fee main screen	144
Figure 5.42 Select fee screen	144
Figure 5.43 Payment screen	145
Figure 5.44 View fee screen	145
Figure 5.45 View all fees screen	146

Figure 5.46 View payment history screen	146
Figure 5.47 View all payment histories screen	147
Figure 5.48 Announcement main screen	147
Figure 5.49 View announcement screen	148
Figure 5.50 View all announcement screen	148
Figure 5.51 Feedback main screen	149
Figure 5.52 Create feedback screen	149
Figure 5.53 View feedback screen	150
Figure 5.54 Action available for feedback	150
Figure 5.55 Update feedback screen	151
Figure 5.56 Complete feedback dialog	151
Figure 5.57 Rate feedback dialog	152
Figure 5.58 View all feedback screen	152
Figure 5.59 Visitor main screen	153
Figure 5.60 Create visitor screen	153
Figure 5.61 View visitor screen	154
Figure 5.62 View all visitor screen	154
Figure 5.63 Profile screen	155
Figure 5.64 Edit profile screen	155
Figure 5.65 Guard home screen	156
Figure 5.66 Guard add visitor screen	156
Figure 5.67 Visitor details screen	157
Figure 5.68 Entered visitor screen	157
Figure 5.69 Exited visitor screen	158
Figure 6.1 Flow of API request from web application	159

Figure 6.2 Flow of API request from mobile	160
Figure 6.3 Hierarchy of Controller Inheritance	161
Figure 6.4 Root Controller Source Code	162
Figure 6.5 Admin and Mobile Base Controller Source Code	163
Figure 6.6 Auth Residence Concern Source Code	163
Figure 6.7 Admin Fees Controller Collapsed Source Code	164
Figure 6.8 List fees function from Admin Fees Controller	165
Figure 6.9 Get fee function from Admin Fees Controller	165
Figure 6.10 Create fee function from Admin Fees Controller	166
Figure 6.11 Update fee function from Admin Fees Controller	167
Figure 6.12 Delete fee function from Admin Fees Controller	168
Figure 6.13 Mobile Fees Controller Collapsed Source Code	169
Figure 6.14 Part of the pay fee function source code from	169
Figure 6.15 Hierarchy of Model Inheritance	170
Figure 6.16 Resident Active Record Class Source Code	171
Figure 6.17 Feedback Active Record Class Source Code	172
Figure 6.18 Feedback Item Active Record Class Source Code	172
Figure 6.19 Fee Payment using Stripe	173
Figure 6.20 FCM Send Notification Function	174
Figure 6.21 Send Notification when Announcement Creation	174
Figure 6.22 Neibor Mailer with Sendinblue	175
Figure 6.23 Send Invitation Email to Resident	175
Figure 6.24 Invitation email	176
Figure 6.25 Password recovery email	176
Figure 6.26 Code Segment of Feedback's Images Upload	177

Figure 6.27 QR Code Generation when Visitor Application Approved	178
Figure 6.28 Deployment of Ruby on Rails API server	184
Figure 6.29 Web application main layout	185
Figure 6.30 Collapsed code of Main.vue	186
Figure 6.31 Views folder for content area	186
Figure 6.32 Web application pages hierarchy	187
Figure 6.33 Code segment of ApiClient.js	188
Figure 6.34 Code segment of ApiClient.js usage	188
Figure 6.35 Deployment of Vue.js Web Application	189
Figure 6.36 Mobile application main layout	190
Figure 6.37 Code segment of AuthWrapper.dart	191
Figure 6.38 Views folder of mobile application	191
Figure 6.39 Screen hierarchy of mobile application	192
Figure 6.40 Overview of Provider State Management	193
Figure 6.41 Code segment of Fee Provider	194
Figure 6.42 Code segment of implementation of ChangeNotifierProvider	195
Figure 6.43 Code segment of FeeScreen	195
Figure 7.1 Code segment to generate mock staff account	205
Figure 7.2 Code segment to assert the response	205
Figure 7.3 Automated unit tests performed	206
Figure 7.4 Code segment to generate mock fee	209
Figure 7.5 Code segment to assert the response	209
Figure 7.6 Automated unit tests performed	210
Figure 7.7 Mock data of visitor	214

Figure 7.8 Code segment of Visitor test suite	214
Figure 7.9 Integration testing performed	215

**LIST OF SYMBOLS / ABBREVIATIONS**

SaaS	Software as a Service
WIP	Work in Progress
IoT	Internet of Things
REAT 2.0	Revised Residential Environment Assessment Tool
XP	Extreme Programming
SDLC	Software Development Lifecycle
FDD	Feature Driven Development
DSDM	Dynamic System Development Method
ASD	Adaptive Software Development
LSD	Lean Software Development
IDE	Integrated Development Environment
VS Code	Visual Studio Code
MBaaS	Mobile Backend as a Service
MVC	Model-View-Controller
DOM	Document Object Model
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
API	Application programming interface



**LIST OF APPENDICES**

APPENDIX A: Interview Question	225
APPENDIX B: Visitor Registration Item at Cypress Condominium	226
APPENDIX C: Questionnaire Form	227
APPENDIX D: Supervisor and moderator comments on project plan	232
APPENDIX E: Kanban board (Trello) progress	235

## CHAPTER 1

### INTRODUCTION

#### 1.1 Introduction

This chapter gives an overview of the project. Firstly, it states the problem discovered in the current community association management system, which is involving many manual paperwork in the problem statement part. Besides, project objective part specifies the expected achievement of the project, which is improve the system by enhance the current workflow with the mobile application. Project solution involves multiple functionalities to achieve the expected project objective. Kanban approach is chosen as the project methodology after evaluating it with other methodologies. Scope of the project determined and documented the expected deliverables and functionality of the project.

#### 1.2 Background of the Project

People always seek for higher standard of comfort, productivity and quality of life. Community association management today are expected to be more advance and integrate with smart technologies in current decade. So, residential management are generally intended to improve the residential management services as people living environment and standard was continuously increasing (Jin, Wu and Huang, 2017). According to Groenendijk, Guild and Barzilay (2000), residential management are found to be fraught with inefficiencies and they are generally backwardness in term of technology. In Malaysia, residential management are facing dilemma as the process involved massive amount of hard-copy paperwork, time and cost is required in the current community association system. Current existing system deserved a better solution to let the residential life reach an optimal level.

### 1.3 Problem Statement

In this century, technology had been everywhere to make things better in many ways such as enhance human living standard or in this case, produce seamless and convenience community association operations. Residences community association management in this decade are expected to be operated in a seamless, efficient method and evolve with the reformation of technology (Nan, Zhou and Li, 2018).

However, there were still many residences, including either low-standard or high-standard residences still had an inefficient and inconvenience operations technique or way to manage community associations in term of time and resources. For examples, community association is managed manually by the residences management officer by a lot of paperwork, they need to input the personal details manually to the system. It was an inefficient process when it came to a large number of information or details registered needed to create or update. Consequently, it causes waste of time and energy to manage the operations for the community activity.

Besides, another example of inefficient manual operations is the traditional method of paying bills or annual fees to residences management, which is manually do a bank transaction to management's bank account. A proof of payment is needed to send to management in order to verify the payment process for her case. This process was tedious and prone to error, as management officer may easily to miss-process about the payment.

When there was a visitor, authentication process of visitor that grant access to the residential area is an inconvenience process also. Existing visitor registration process in residence generally involved registration form which mandatory to fill in and temporary deposit of visitor's identity card may issue many problems such as leakage of personal information.

By wasting human man hours and costing excessive amount of paperwork, the current way to perform the operations within community associations make the residences less competitive, and by contributing to a slow or hassle process to manage community associations, it creates an unfavorable brand image.

#### **1.4 Project Objectives**

This project is aimed to achieve the following objective stated below:

1. To identify the current problem of the existing community association management system.
2. To simplify the existing community association management operations by automating repeated tasks.
3. To develop a system that can used to manage the community association that build community loyalty and enhance the residential image.

The developed system will increase the productivity by allowing the operations to be done quickly and concurrently.

Besides, community association general activity such as, update personal information, receival of important announcement, paying the fees, etc. can be operate without delay and human interaction on developed mobile application anytime and anywhere.

#### **1.5 Project Solution**

The project is aims to solve the problem encounter by current community association management system. An automated system will be developed to act as the solution to the problem. The developed system involved back-end API server, web application served as admin panel and mobile application that used by community resident. By utilizing the developed mobile applications on wireless devices, workflow of the system can be improved as it required less manual paperwork and human interaction as unnecessarily that management spend their hours of the day on the paperwork (Madsen, 2007).

In the mobile applications, the problem can be solved by the ways described below:

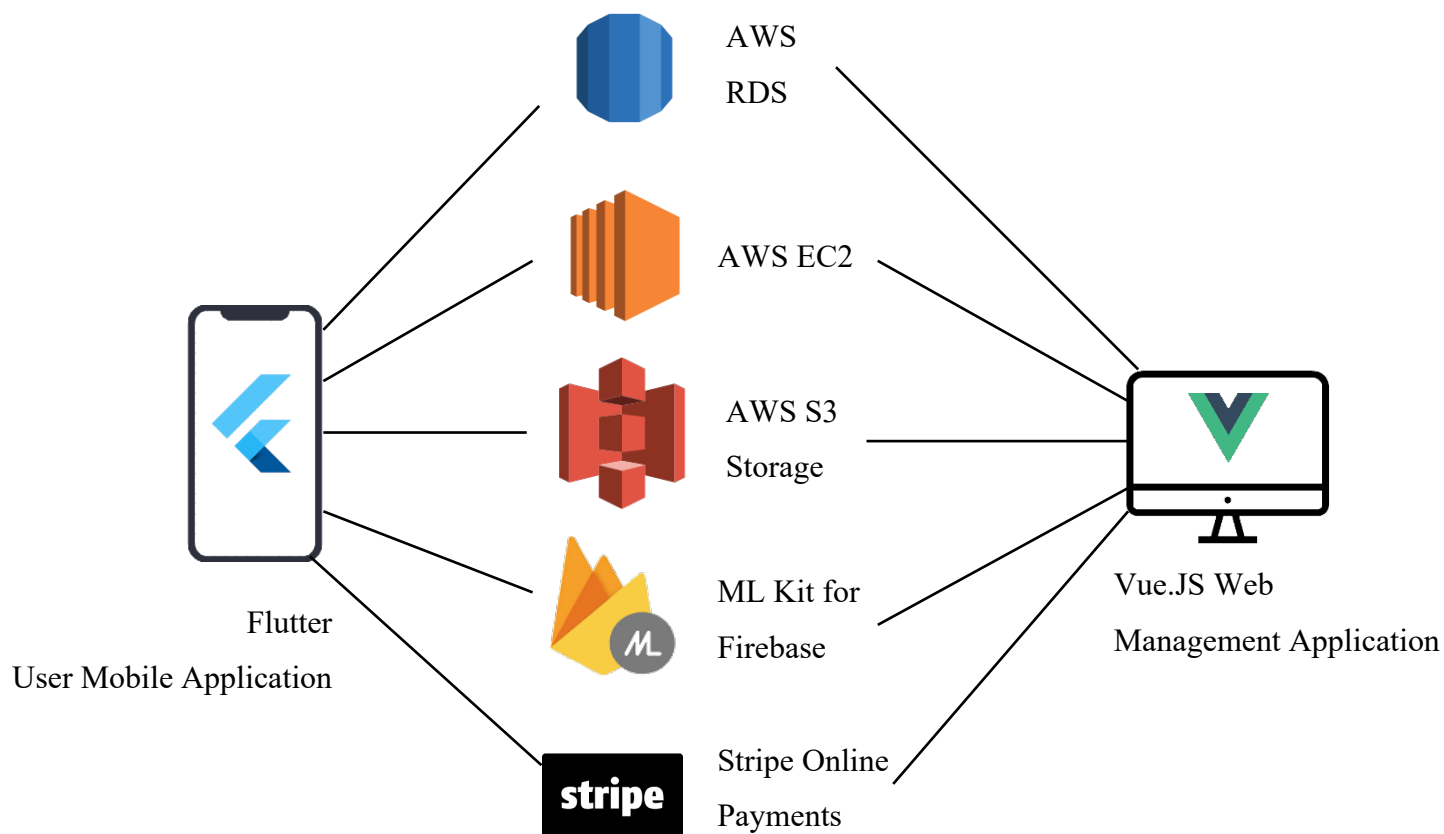


Figure 1.1: Proposed solution technologies involved modules diagram

The community resident related information is stored in the MySQL database instance that provided by Amazon Relational Database Service (RDS), this reduce the amount of manual paperwork to collect the resident's information. Through the developed application, residents can change their information, such as car plate registered easily and without human interaction. Besides, residents' feedback will also store in the MySQL database instance, those data can be access and review by the management officer in a well organize manner. It can increase the productivity as management officer waste no time to collect and organize the hard-copy feedback forms.

A centralized API server to serve and handle the request from two front-end clients will be hosted on Amazon Elastic Compute Cloud (EC2). This centralized server approach provides an extensive and scalable option for the future expansion of

the system. One of the benefits is it leads to better code management and control. Besides, EC2 is a web service providing stable, resizable cloud computing and hosting capabilities. It's planned to make cloud computing on a web scalable simpler for developers. The simple web service interface of Amazon EC2 helps you to get and customize for minimum friction and maximize the capacity. It gives you complete control of your computing resources and allows you to run on the validated computing environment of Amazon.

The community association management system will involve several scenarios that storage of image or custom file of resident. For example, when submitting a feedback, user can upload a photo to make further description on the problem he/she encountered. To fulfill this requirement, Amazon Simple Storage Service (S3) is used. Amazon S3 provides industry-leading scalability, data availability, encryption and performative efficiency to store objects. This helps developer of all sizes and sectors to store and safeguard some volume of data in a variety of applications in different use cases. Furthermore, Amazon S3 offers flexible access, rate, replication and data security management capabilities. Amazon S3 Access Points allow user control to the systems by leveraging a common data collection with different permissions.

By integrating the Stripe online payments, the developed mobile application enables residents to pay their fee online securely and remotely. Stripe is an online payment service provider, it provides credit card payments that can be integrate into Flutter application as Software as a Service (SaaS). There are 3 type of product in Stripe services, which is Payments, Billing and Connect. Payments is suitable in the situation that user pay the payments one time only. Billing is suitable in the situation that user will recurring pay the bill at the subscription basis. Connect is suitable in the situation that user will accept or pay out money to third parties, it suits with the case in e-commerce business, marketplace etc.

Furthermore, hassle visitor registration process can be improved by using QR Code authentication access. Unlike the existing visitor access granting procedure, the process can be held without paperwork by using the application developed. Firstly, resident living in the residential area can register to apply the access granting for their visitor by input the details of visitor in the mobile application. After that, a QR code will be generated which that can be send to the visitor. Visitor can use the QR code received and the guard of the residential area then can scan the QR code to view the details of the visitor and grant access to the visitor.

For the web-based management side application, it will provide a user-friendly system to allow the management team to manage the daily basis task. For examples, make announcement and the system will send notification to the community residents affected. Furthermore, management team can view each resident fee payment status and manage the resident feedback or complaint in the system easily.

## 1.6 Project Approach

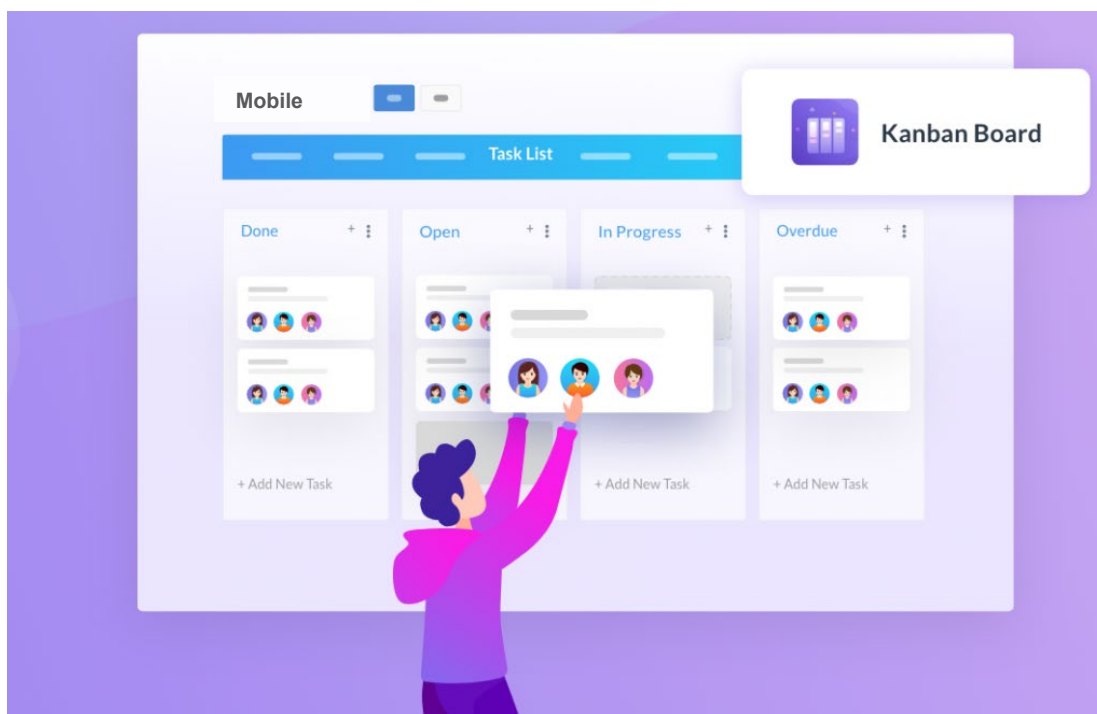


Figure 1.2: Personal Kanban digital board by Trello visualization

This project will apply personal Kanban framework approach in agile principles. There some core element inside Kanban, such as visualize work tasks, limit work-in-progress (WIP), focus on improvement of the project flow and continuous improvement of the product (Wiegand, 2018).

Firstly, work task is visualized by using a Kanban board, which can be either a digital board or reality board, we can list out the work need to be done, the work in progress and the work had done. Through this kind of visualization of work component, it makes the structure of the project more obvious and lead to high productivity. As like Benson (2010) stated on Øredev Conference, personal Kanban reduce the chaos in the project and promote the individual effectiveness to success.

Besides, by limiting the amount of work-in-progress at one time, it can assist the project to be more focus on the specific task and avoid time wasted on task switching or reorganize the priority of the work components. It allows the project to be deliver in a high quality and faster manner.

Moreover, using the WIP limits in the scope of Agile principles, we can continuously improve the project flow and consummate the implementation from the previous iteration. The software tools will be used in this project as Kanban digital board is Trello as it provides a user-friendly and sufficient functionality for this project approach.

### **1.7 Scope of the Project**

The developed mobile application is limited to the use in specific residence's community and residences management officer only. The mobile applications can be accessed by users though their mobile devices on both iOS and Android platform. This project will deliver two mobile applications for community association management, which is the community-side application and residence management-side application and the details are described as follow:

#### **1. Community-side application**

In this application, it will cover the basic community association activity functionality:

- **Log in**

Community resident is able to log in to the system using the ID and Password that given by the management officer, which is pre-register by the management officer.

- **Update personal information**

Community resident can initialize their profile at their first log in at the initialization page after the log in page. At that page, fill in the form with the details such as the title before name, resident name, gender, IC number, email and phone number. After that, if resident wish to change their personal details, resident able to update their personal information at the updating page also.



- **Pay fees**

Community residents can pay their fees through the application online securely and hassle-free by using bank transaction gateway. There is no requirement of email proof of payment send to the management office. In this functionality, residents able to view the fees need to pay, the respective deadline, and also the paid transactions in the payment history.
- **Receive notification of announcement**

Community residents will get notified whenever there is an announcement made through the application without delay. Besides, recent announcement or message will also be recorded in the announcement board, residents can access to it anytime and anywhere through their mobile application.
- **Provide feedback**

If there was any issues or feedback, community residents can make an immediate feedback with attachments if any through the mobile application. Moreover, the submitted feedback status can also be tracked through the application and provide a channel to allow residents to communicate with the management office until the problem resolved. After that, residents can mark the specific feedback as solved.
- **Visitor registration**

Community residents register their visitor(s) in this module to apply the approve for the access of visitor(s) into the residential area. Personal information of visitor(s) is required for the verification of the access granting process and serve as backup to track if any incident happen. After submitting the information, a respective QR code will be auto generated by the system and the specific QR code can be send to the visitor and act as a pass to gain access.

## 2. Management-side web application

In this web application, it will cover the basic residence management activity functionality:

- Make announcement

Residence management officer can send announcement to either, all residents, specific block residents through the application. In the announcement writing page, officer need to fill in subject and content of the announcement, attached file if applicable and finally selected the portions of residents will receive the announcement.

- Monitor fee-paying status

Through the application, residence management officer can view and monitor the fee-paying status of each residents and send private notification messages to the specific residents to remind the deadline and fee-paying matters, such as how to pay the fee.

- Manage feedback

Residence management officer can view the feedback that submitted by the residents and reply to it. The page will show each status of the feedback is either, pending, in-progress, and solved. Through the application, officer can reply to the resident's enquiry and communicate with them.

- Manage visitor

In this part, residence management can view the application of visitor access from the resident community and approve or reject the application based on the information submitted. Besides, history of the visitor access will be record as a backup for reviewed.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

In this chapter, multiple existing community association management system had been discussed and evaluated to gain some inspirations and ideas. Besides, the demands and need of the resident community is investigated and exploratory study is done to identified possible requirement of the system. Furthermore, evaluation and comparison between various software development methodology had been done and Kanban in Agile will be apply in this project. The system architecture combination options would be worth pondering topic in this project including the backend and front-end framework, cloud services provider, etc.

#### 2.2 Review on Similar Community Association System

Community association system is still not a standard equipment for the residential area in Malaysia. However, the trend of it had begun as some residential area had slowly integrated and implement some system to enabling better management.

##### 2.2.1 Cloud-based System

From community association networks and multimedia to multiple community management automation systems, smart resident technologies have evolved quickly. In 2016, Lee et al. had proposed a cloud-based system for the integrating community services that can be access through internet in smart devices is proposed in this study.

The objective of the proposed system is to:

- Manage the device operations involved
- Reduce manual work and labour required in the community management process
- Provide electronic information services
- Support diversified services
- Support location-based services

One of the limitations is that the proposed system involved Internet of Things (IoT) which is hardly implemented to current resident's area as it may not accept by the existing resident. However, the study still gave an overview software application architecture of similar existing system that can be tailor to the project and suggest optimal approach of protocol implementation for specific purpose, such as control of resident management device.

Besides, patent of "Security and Property Management System" describe a well-defined property management system should consist of an interactive and user-friendly system for reporting, tracking and rectifying security and maintenance items and incidences occurred (Alonso, 2006).

The modules to be considered included:

- Database server, for storing data information
- User devices, for accessing, inputting and receiving information from the database server
- Interface system, for allowing input and output from the system remotely
- Real-time access, for providing the immediate receive of information
- Property management customization, for setting up specialized solution for specific type of property

The invention should have many built in and features to increase the overall efficiency of property management. The patent application is based on and claims priority on U.S. Provisional Patent Application No. 60/377,013 having a filling date of 30 Apr. 2003, currently pending. The patent provided a details and rigorous information about the property management system.

Furthermore, another similar community association system is the management system proposed by Li et al. (2010). The proposed community association management system utilized a technology known as WebGIS, to implement functionalities such as real-time video monitoring, sensor monitoring alarming, multimedia network transmission technologies etc. In the system study, requirement analysis was done to analyze and verify the system requirement collected. The integrated functionalities were featured by advancement, practicality, visualization and automation. Overview design of the system:

- GIS module, for real time monitoring and alarming
- Residential module, for management information of community general condition, building and tenement that provide query, browse and satiating function
- Business manage module, for property expenditure, client complaining, repairing report, staff dispatching and arrangement

By analogy approach, the system design and determined system requirement can be adopted into the project. On the other hand, the proposed system had some limitations that lacking of involvement of user perception and experience. In addition, there are no clarification or proof on the impact of the system built.

### **2.2.2 MyTaman Community Application**

MyTaman is an internet of things (IoT) smart community system, with the latest technologies and framework to protect the community. It offers apps that serve everyone, and that will support them in their everyday lives. The apps assist them when they're in danger (SOS button), inform them about the arrival of their guests (Guests warning, keep updated on the current neighborhood events (News events), even provide a private platform for the trustworthy service providers with your own residential area (Recommendations), and more. MyTaman consists of several main functionality such as MyTaman Pay, MyTaman Visitor, Resident Association, Recommendation, etc. It aims to reduce the criminal activities, simplify the complex cable intercom, inefficient visitor management and build a safer environment for resident community.

According to MyTaman official website, they had been used by 200+ communities in Malaysia such as Suasana Sentral Loft Kuala Lumpur, Sunsuria Residence, etc. However, there are some drawbacks of MyTaman community application is that residential management association need to invest a number of moneys to purchase the MyTaman related hardware device to use the system.

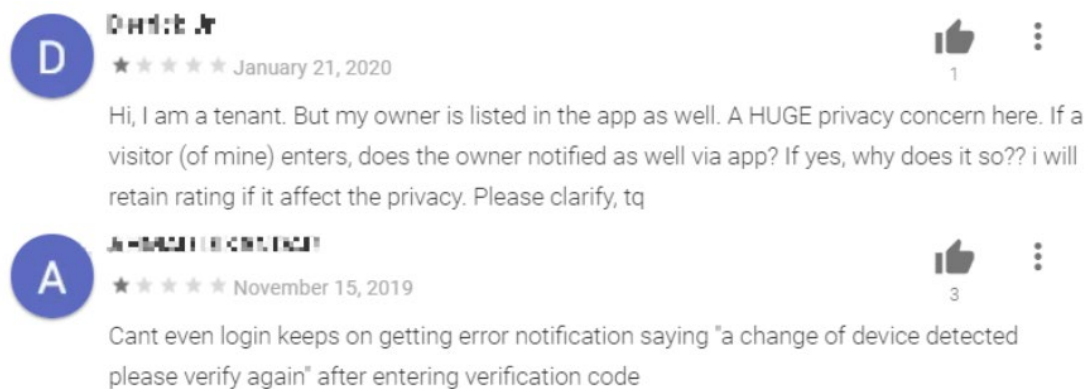


Table 2.1 User reviews of MyTaman application at Google Play Store

In addition, there are some reviews at Google Play Store for MyTaman application that reflect the application may have some privacy issue and logical error like the figure above shown.

The following are some special features of MyTaman community application:

- Smart door access using smart phone
- Complete visitor management system
- Recommendation system
- Guard patrol monitoring

### 2.2.3 Homeserva Smart Community System

Same as MyTaman, Homeserva smart community system introduced by VYROX involved internet of things (IoT) technology and aim to build smart community that based on property management. As VYROX is originally focus in IoT smart home integration and automation, Homeserva had its advantages when integrate with VYROX other IoT hardware such as CCTV, burglar alarm sensors, Amazon Echo, video intercom, etc. Apart from them, Homeserva also support facilities management that help residential community book the facilities conveniently and auto arrangement of the request. However, there are only a small number of communities implemented Homeserva system, which means the market share of Homeserva is still low.

The following are some special features of Homeserva smart community system:

- Focus on IoT integration
- Lift controls

- Community marketplace
- Community on-demand services
- Parking lots management

### **2.3 Demands of System and Solutions**

According to Madsen (2007), There are 4 related impact of technology on property management area identified. Firstly, skillset demand for property management staff had changed as evolution of technology had produced more advanced and user-friendly management software. In the past, the property management staff are mostly technician that have professional knowledge on that software used. But times changed, todays property management required less professional knowledge to operate that management software as the functionality are increasingly automated by technology. Secondly, expectation of tenants/clients getting higher as they may be conscious of the technology used and expect higher about property management firm to provide better services. Thirdly, overall services deliverable is better as many organizations are investing in their property management system to attract their clients. Fourthly, property management staff no longer required to be restrained to desks as wireless devices can facilitate the remote work.

In Madsen's research, careful research and study on the property management case study had been held. Besides, interview session with Glen Fernald, managing senior vice president of management services, Mid-Atlantic region, Transwestern, Bethesda, MD on the property management topic had done to get some insights from high level management.

Residential needs are essential to be known in order to improve residential service provided for association community. Jin, Wu and Huang (2017) had mentioned several residential services demands determined in their research which are:

- Commination skills
- Maintenance efficiency
- Garbage removal
- Greening layout
- Traffic management
- Elderly activities

This provide an overview understanding to the potential residential service demands that can be act as the problem of project need to solve.

Residential and community satisfaction is found to be associated with the service quality level provided by community association management team (Poortinga et al., 2017). This study by Poortinga and his team assesses the residential area by a tool that known as Revised Residential Environment Assessment Tool (REAT 2.0). The factor considered can be categorized into street level and property level. For street level factor examples, it consists of litter in public space, condition of public space, recreational space etc. On the other hand, for property level factor, it consists of property maintenance, property service quality, garden maintenance etc.

The neighborhood quality data were collected from a program in Carmarthenshire, Wales. The neighborhood perception study is conducted at Cardiff in the form of questionnaire that involving 1160 participant. Major strength of this study is the breadth of analysis conducted to identify and reliability of the REAT 2.0 tool and the coefficient between the neighborhood quality with the neighborhood satisfaction. Limitations of this study is the relatively small sample size and narrow neighborhood aspect may not reveal the actual insight on general residential area. Besides, location of study is limited in United Kingdom. However, it still provided a prove that the community association satisfaction is linked to the quality of property management service provided and suggested an approach to evaluate community association satisfaction.

Rahman et al. (2015) examined the relationship between housing expenses, affordability, service quality by management side, hygiene factor and community attachment and residents' satisfactory. There is one factor that closely related to the project, which is service quality. The findings of the study stated that the satisfaction of a customer depends on the service quality level provided. The community association management services quality can be measured by:

- Reliability, to perform promised service dependably and accurately by the management team
- Responsiveness, to help residents on problem and provide prompt services
- Assurance, to convey trust and confidence to resident's community
- Empathy, to provide caring and individualized attention to resident's community



In the study, questionnaire session was carried out to collect data. Besides, some other data gathering method also involved in this study such as descriptive analysis, exploratory factor analysis confirmatory factor analysis and structural equation modelling approach to verify the hypothesis statement. The context of the finding is in South East Asia which may highly applicable to this project. It contributes to this project by suggesting some measurement indicator to measure residential community satisfaction to the residential management services.

Under the concept of “Internet Thinking”, Yan (2018) had conduct a research on various operation status of residential property management and proposed solution on those status. There are several main situations determined:

### **1. Uneven Development Level of Residential Property Management**

There are three types of development level of residential property management, which the first one is no property management system. In this case, the cleaning fees is paid by household and guard salary is paid by the management company. Besides, another type of property management is “four guarantees” service that included cleaning, security, green protection and guarantee repair in residential area. Furthermore, residential management system that automated to improve the efficiency was implement in large-scale and well-known residential community.

### **2. Management system mostly small and incomplete**

The management technology level is backward, and the technology content portion is not high. The existing system operate in an inefficient and non-standard way.

### **3. Lack of professional property personnel and systematic training**

Property management is a staff-intensive service type. The process mostly involves low employee with uneven quality and lack of training. Large mobility of personnel may interrupt the process of experience pass down.

## 2.4 Review on Project Methodology

A right project methodology is critical to the success of the project to achieve the expected deliverable effectively and efficiently. There are a wide range of development methodology that can be investigated and adapted to ensure the success. As like Muslihat (2018) stated, there are no a perfect methodology suit for every project, selecting the appropriate methodology is the key to contribute to the success of project. In this section, four methodologies, which is Waterfall, Scrum in Agile, Prototyping and Extreme Programming model will be discussed in term of their characteristics.

### 2.4.1 Waterfall

Waterfall methodology known as the first Software Process Model, and first introduced by Dr Winston W. Royce in 1970 (Powell-Morse, 2017). As its linear characteristic, it is bringing Waterfall model very simple and intuitive phase of process. The model focused that current phase must be completed before moving into the next phase of development process. However, it also restricts the project to backward or overlapping of phase if there was any problem encounter.

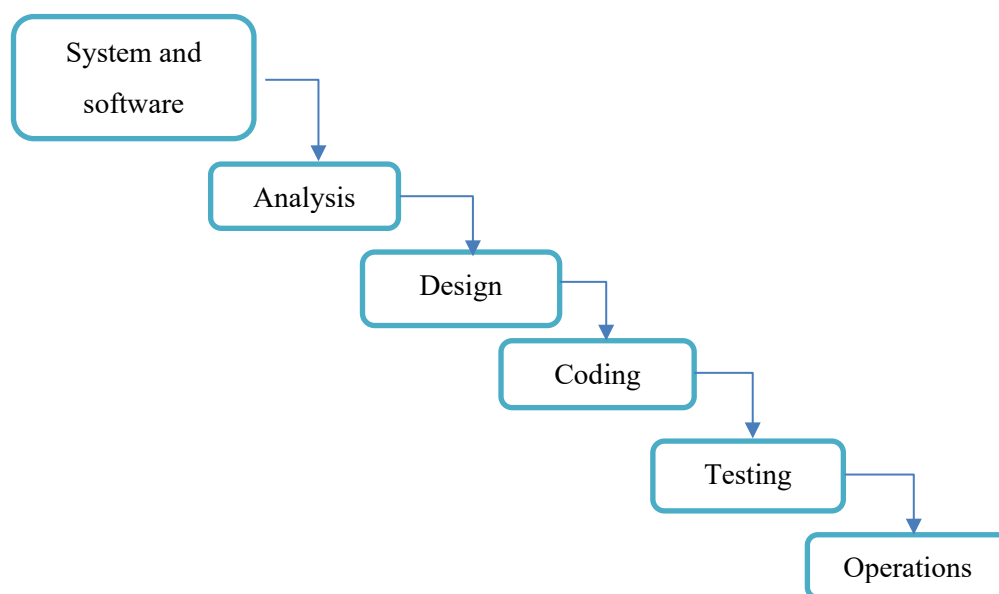


Figure 2.1: The Six Stages of Waterfall Methodology

### 2.4.2 Scrum in Agile

Scrum is one of the methodology frameworks under Agile. Introduced by the empirical inspection, Scrum's early proponents adapted feedback loops in order to deal with complexity and danger. Scrum underlines real-world decision-making rather than speculation. Time is organized into short task cadences, usually one or two weeks long known as sprints. The item is always held in a (correctly incorporated and tested) shippable condition. Stakeholders and teammates will meet at the end of each sprint to see a proven potential increase in products and plan their next steps. Scrum is a straightforward collection of positions, tasks and meetings which never alter. By removing unnecessary unpredictability, we can better deal with the required unpredictability of continuing learning and discovery.

In a Scrum context, it normally consists of 3 roles:

- **Product owner**

A visionary, authoritative, and availability individual. The Product Owner communicates the vision and priorities to the development team continually.

- **Scrum Master**

The Scrum Master serves as the Product Owner and Team's facilitator. The team is not managed by Scrum Master. The Scrum Master operates to remove all obstacles to the sprint objectives of the team. This enables the team stay creative and efficient and ensures that the Product Owner sees its achievements. The Scrum Master also advises the producer how to maximize the team's ROI.

- **Teams**

According to Schwaber and Beedle (2001) who is the founder of Scrum, he stated that, "The team manages totally itself," the team develops itself for completing the work. A development team of Scrum comprises around seven dedicated members (officially 3-9), perfectly in one team room protected against distractions from outside. A typical team of software developers, architects, programmers, analysts, QA professionals, testers and UI developers involves a combination of software projects. Every sprint, the team determines how the work to be finished is done. The team has independence and accountability to achieve the sprint objectives.

### 2.4.3 Kanban in Agile

Agile Kanban is Kanban approached for Agile Software Development. The workflow is visualized in Agile Kanban by the Kanban board. The board of Kanban is usually placed on a project room wall. On the Kanban Board with floating Kanban cards the status and advancement of the growth of the story is monitored visually. The Kanban board is used to represent task flow through the value stream. It offers everyone who participates in the project simple access. In addition, it also facilitates communication where needed and task advancement is exhibited visually. As quickly as they happen, bottlenecks are noticeable.

Kanban cards represent the duties and stories. Task available in different columns of the board, the current status for every assignment is known. Every job passes from to do and then to perform. As the development progress advances with growth, the Kanban Board is updated daily.

In Kanban, there are several important elements and concepts which are:

- **WIP Limit**

The Doing column label also contains a number representing the maximum number of tasks that can be in that column at any point in time, i.e. the number associated with the Doing column is the WIP (Work-In-Progress) limit.

- **Pull Approach**

Pull approach is used as and when the Doing column completes a job. Another card from the To Do column is taken.

- **Self-Directing**

The team is accountable for planning, monitoring, reporting and communication in the project in Agile Development. The team is entitled to make choices and is responsible for completing the creation and the quality of the item. This aligns with the Kanban team's empowerment feature.

- **Continuous Workflow**

There is no door strategy in agile development and the job flows through the various tasks without waiting time. This helps to minimize Kanban's cycle time feature.

#### 2.4.4 Prototyping

In this methodology, a prototype will be developed, tested and altered continually until a required final product is delivered, which constitutes an approximation of the features of the final scheme. With minimal features, the first prototype is created and from moment to moment adds extra features to the final scheme. First of all, it will determine the system requirements and develop a preliminary prototype based on the specifications. This methodology involves users in analysing and evaluating the first prototype. The developers will collect and save all user comments and reviews for next changes. After a change to the first prototype, a second prototype is developed and sent again for assessment with further characteristics. The above steps are repeated until the user satisfies the prototype. Next, the last prototype is the reference for the final scheme (Rouse, 2005). After completion of the design stage, a series of tests such as unit testing, integration testing and acceptance testing will be applying to the deliverable. Usually this methodology is used in a project where some of the project requirements are not well defined. This methodology works well in this type of project because it is iterative and frequently interacts with customers.

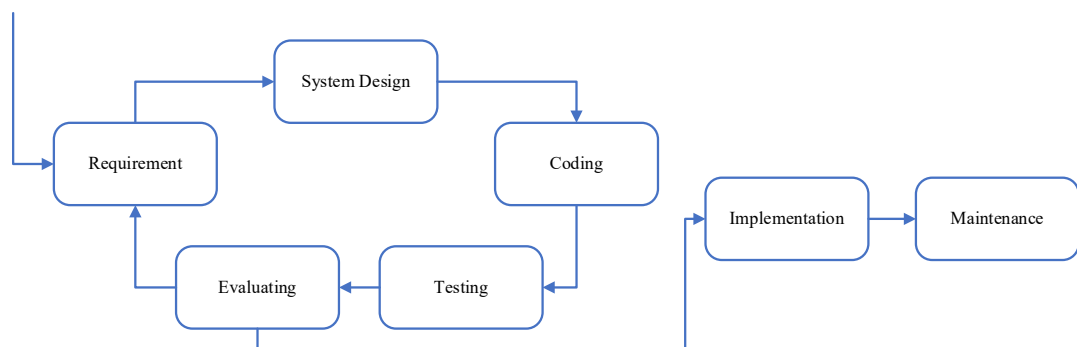


Figure 2.2: Diagram of Prototyping Methodology

### 2.4.5 Extreme Programming

Extreme programming (XP) is a software development methodology that aims to enhance software quality and client response. It promotes frequent releases in brief development cycles as a form of agile software development to increase efficiency and introduce controls that enable fresh demands of customers. XP is a simple, powerful, low-risk, flexible, predictable, scientific and enjoyable way of software development. Extreme programming was conceived and developed in the face of vague and changing requirements to meet the specific needs of software development for small teams. In larger teams with a team size of 12-16 developers, Extreme programming is considered to be efficient. The interest of the clients engaged in the process can be hard to maintain. Members of the team may not be adapted to the intensive participation of agile methods. When various stakeholders exist, prioritizing modifications can be hard.

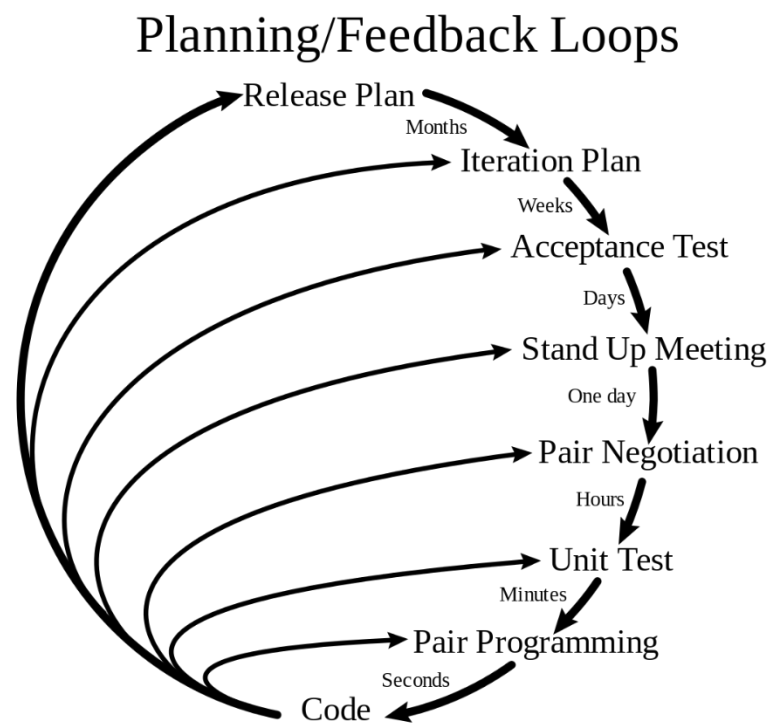


Figure 2.3: Diagram of Prototyping Methodology

### 2.4.6 Comparison of Methodology

The process of software development can be regarded at two separate stages: initial or first-level activities linked to software acquiring data, growth and maintenance, next or second-level activities linked to dentition, functioning, measuring, and upgrading of the software system itself. The study by R.Raval and M. Rathod (2013) presents a comparative analysis of different software development process models based on different parameters; it also lists different factors in the selection of partial software model in the software development globe. The table below showed the comparison of various methodology considered in this project:

Table 2.2: Table of comparison between multiple methodologies

Parameter	Waterfall	Agile	Prototyping	Extreme Programming
Clear Requirement Specification	Initial Level	Change incrementally	At medium level	Initial level
User feedback	No	No	Yes	Yes
Adaptability to change	Low	High	Medium	High
Predictability	Low	High	High	High
Risk Management	At initial level	Yes	No	Yes
Practical Implementation	No	High	Medium	High
Usability	Basic	Most use now a days	High	Medium
Elasticity	No	Very high	Yes	Medium

From the table above, Agile methodology is identified to be more suitable for this project. The reason is the requirement specification prepared may need to change over time. Besides, Agile model have high level of predictability, good risk

management and required many practical implementations. Due to its high level of elasticity, Agile model can be easily tailored into this project.

There are two Agile frameworks to compare with is Scrum and Kanban. According to Alqudah and Razali (2018), they determined various differences between Scrum and Kanban methodology in term of the selection factors and identified the suitable situation to adopt each methodology stated in their study. Based on their study:

Kanban is suitable for:

- Prefer not to follow prescribed method
- Prefer not to follow predefined roles and responsibilities
- Team size is more flexibility, such as less than 5 team members or more than 11 team members
- Batching work and do requirement prioritization daily
- Focus more on cutting lead time
- Improving quality
- Cutting cost

Scrum is suitable for:

- Prefer to follow prescribed method
- Prefer to follow predefined roles and responsibilities
- Team size comprise of 5-11 team members
- Batching works in one-, two- or four-week basis
- Not emphasize cutting lead time
- Focus more on knowledge, experience and decision making based on situation
- Quality is not emphasized at high level
- Not emphasize of cost saving

As the study provide a reference and concept about Scrum and Kanban methodology and suggest a systematic selection method to choose between Scrum and Kanban. After considering the parameters stated, Kanban in Agile is chosen to be the methodology in this project.



## **2.5 Review on Backend Server Framework**

Frameworks have become an important aspect of web development as web application requirements continue to grow, as does the technical sophistication necessary. Without this advanced methodology, it is completely impractical to rebuild the whole framework infrastructure manually again. That's why it is an essential approach to implement a backend framework supported by thousands of developers worldwide to create rich and immersive web applications. However, there are numerous of backend framework outside the world, choosing a right framework that can fulfill the needs of developer is a vital factor that contribute to the success of the project. There is no best framework but the most suitable framework.

Backend framework can be judged by several aspect such as programming tools, language it used, interface or feature they offer. Moreover, pre-configured tools and templates that help developer's productivity could be one of the considerations included also. Section below will evaluate characteristics of each backend framework.

### **2.5.1 Ruby on Rails**

Rails is a free, open-source server-side web development framework that fully written in Ruby. Ruby has always been considered as one of the most developer-friendly languages and Rails incorporates Ruby's capabilities and advantages feature. Rails also provides developers innumerable predefined solutions for repetitive tasks in a very high readability and clean manner. Furthermore, it facilitates agile development and provide almost every component that a backend server needed (Tachibana, Kon and Yamaguchi, 2018). There are several popular applications that are developed in Ruby on Rails, such as Shopify, GitHub, Airbnb, etc.

The following are some unique feature of Ruby on Rails backend framework:

- Allow customized extension on almost every primitive type or class, such as String class, True class, etc.

- Active Record, the built-in object-relational mapping (ORM) of Ruby on Rails provide simple, clean and yet powerful operations to the database.
- Focus on MVC framework
- Massive and supportive community
- Various of plugged-in library that ready to use

### 2.5.2 Laravel

Laravel is a modern architecture that simplifies the development process and eliminates a great deal of pressure from the web app project. Strength of Laravel is its simplification on the complex tasks such as authentication, containerization, queues and routing (Laaziri et al., 2019). Laravel has its own code base for the migration system. It is the easiest way to build a web application with complex backend requirements. Besides, Laravel has a large community. For examples, Laracasts is a platform for screencasts of over one thousand videos on the Laravel environment of PHP, Laravel, and front-end technology that could be considered a blue sky for beginners.

The following are some unique feature of Laravel backend framework:

- Minimize bootstrapping and offers maximum versatility
- Easy switching of current framework to Laravel
- Library that support fast route to handle request with appropriate response
- Built-in caching, error, log-handling and authentication

### 2.5.3 Node.js

Node.js is a runtime framework that written in JavaScript and built on Chrome's V8 JavaScript Engine. It promotes event-driven programming, non-blocking I/O model and this led to rapid speed and lightweight in computing (Rose and Survesh, 2017). It eliminates the waiting time for code statement to execute but it run parallely continue

with the next request, as it run single-threaded and asynchronously programming nature.

The following are some unique feature of Node.js backend framework:

- NPM, stands for Node package manager, which is a large ecosystem that contains a variety of library that can installed by Node.js framework
- Lightweight technology and memory efficient
- Can combine with React front-end framework to build full stack JavaScript environment
- High performance as process several requests concurrently

#### 2.5.4 Comparison of Backend Framework

Every backend framework has its own set of advantages and drawbacks, there are a few considerations can be compared.

Table 2.3 Comparison of Ruby on Rails, Laravel and Node.js

Characteristics	Ruby on Rails	Laravel	Node.js
ORM	Active Record	Eloquent	Sequelize
Speed	Average	Average	Fast
Learning Curve	Average	Shallow	Shallow
Community support	Large	Average	Average
Special features	Focus on MVC framework, extensive infrastructure	Built-in authentication, caching	Non-blocking, concurrently processing

As the table above shown, each framework had their own characteristics. Active Record feature of Ruby on Rails is the preferable ORM in this project as clean code, high readability and maintainability is emphasized in this project. Although Ruby on Rails has several weaknesses such as slower speed compared Node.js. However, Ruby on Rails had better advantages in computing intensive task compared to Node.js as Node.js single-threaded nature while Ruby on Rails support multi-threaded. Furthermore, Ruby on Rails facilitation on MVC design pattern suited with the requirement of the project as MVC pattern will be implemented in this project. After considering the parameters stated, Ruby on Rails is chosen to be the backend framework in this project.

## **2.6 Review on Front-end Web Application Framework**

Front-end user interface is used as the View in the overall context of Model View Controller design pattern. For contemporary web application projects today's, majority of the projects had front-end development involved. Front-end framework served as a scaffold for the construction of the project's front-end. It always brings some good approach to structure the project file, make request, associate data with DOM elements, etc. Additionally, front-end framework also brings some benefit to the project such as separation of concerns, speed in development, well-structured prebuilt patterns and better maintainability.

Most of the front-end framework in the industry is designed for single page application. The system architecture for a single page application revolves around having one page from the server. The page should include an HTML elements frame, as well as references to different frameworks and the application on the client side. All other resources (like the data to display) will be fetched on demand by the request using library. In the domain, several popular front-end frameworks are ruling the preferable of developer such as React, Vue.js, Angular, Ember.js, etc. Comparison between React and Vue.js will be discussed in the section below on the aspect of their characteristics.

### **2.6.1 React**

React is a front-end development framework developed by Facebook. Rather define React as a front-end framework, its official definition is more like a UI library.

Philosophy of React is emphasized on the interaction, stateful and reusable UI components creation. It enables complex web application to change its data for render without refreshing the subsequent page. React reorganize the Document Object Model (DOM) in a more abstract form to provide simpler and more robust web application development experience.

The following are some unique feature of React front-end framework:

- Better performance by more efficient and lightweight DOM
- Memory efficient by using virtual DOM
- Large community and maintain by Facebook
- Unidirectional data flow

### 2.6.2 Vue.js

Vue is a front-end framework developed by Evan You, an ex-engineer of Google. One of the similarities between React and Vue.js is that both of them implement virtual DOM that claim better performance. Vue is well-known by its low complexity, small build size and perfect integration of JavaScript nature. It built in with high popularity web technologies feature that most commonly used by development and build on top of the layer to deliver the convenience to developer. Even though Vue.js having a minimal size of core which make it very lightweight, developer is still able to integrate additional library to scale up the project whenever needed (Comparison with Other Frameworks — Vue.js, 2020). Famous use case of Vue.js are Gitlab, Alibaba, etc.

The following are some unique feature of Vue.js front-end framework:

- High flexibility as many official package or extension ready to implement
- Shallow learning curve as well-written documentation which is very intuitive and complete
- Simple implementation, suitable for wide-range of project
- Small framework size that optimize the performance

### 2.6.3 Comparison of Front-end Framework

The evaluation of the characteristics of React and Vue.js can be discussed in various aspects such as performance, learning curve, documentation, etc.

Table 2.4 Comparison of React and Vue.js

Characteristics	React	Vue.js
Flexibility	High	Relatively High to React
Performance	High	High
Learning Curve	Shallow	Relatively Shallow to React
Community support	Large	Average
Documentation	Well written	Well written

As the table shown, React and Vue.js are very similar in characteristic and feature. According to the third-party benchmarking result, React and Vue.js had similar average performance on web application. The selection within these two front-end frameworks would subjective to developer. The weakness of Vue.js is it has smaller community worldwide, which can minorly affect its reliability. However, Vue.js still gained a large number of popularity and well-known by its ease to learn and integrate and its great official documentation. In the nutshell, Vue.js will be chosen as the front-end development framework for the web application development in this project.

## **2.7 Review on Cross-platform Mobile Application Framework**

Hybrid app framework gained its popularity today as it removes the burden to develop separate native mobile applications. One single general solution needs to be developed, maintained and controlled for various mobile platforms such as iOS and Android by using a hybrid app framework.

Flutter and React Native are the two main cross-platform mobile application frameworks in the industry which have gained popularity in the community. In 2015, Facebook released the first version of React Native at the React JavaScript Configuration Conference. On the other hand, Flutter was introduced by Google to enter the mobile application world much later compared to React Native. Both Flutter and React Native share a number of similarities such as creating cross-platform apps using one codebase, hot reloading feature, excellent native UI, native functions etc.

### **2.7.1 Flutter**

The objective of Flutter is to eliminate the performance issue of hybrid mobile applications. It aims to produce high-performance mobile applications. Besides the aim for high-performance, Flutter also aims for high productivity for developer concerns. It supports stateful hot-reload in the development process, which is considered as a vital factor to increase the speed of the development cycle.

Widgets are one of the special features in Flutter, they act like the components in React or React Native. Widgets control how the view is rendered and behaves, additionally they also handle and respond to the events from the application. In Flutter, the application is built with elegant Material Design Widgets and Cupertino Design Widgets (specialized for iOS), which make developers can easily build a high standard and beautiful application.

The following are some unique features of Flutter hybrid mobile application framework:

- High native performance
- Modern technology that gained popularity
- Expressive and high flexibility user interface
- Fast development cycle

### 2.7.2 React Native

One of the special characteristics to be observed in React Native is about the usage of JavaScript XML (JSX). JSX is a special syntax extension to JavaScript languages, which served as a foundation to describe what user interface to be show and how they to be render. React Native will compile the JSX file into a normal JavaScript object during the mobile application compilation process (Wu, 2018).

Same as React, React Native also implemented virtual DOM technology to act as the bridge between the native app and actual DOM object. By using virtual DOM, it can transform the native app event and configuration to more efficient approach before updating the actual DOM object.

The following are some unique feature of React Native hybrid mobile application framework:

- Using JSX to describe user interface object
- Can corporate with React and Node.js to build full stack JavaScript system
- Concept of FLUX that are unidirectional data flow
- Large community

### 2.7.3 Comparison of Hybrid Mobile App Framework

Both of this hybrid mobile application development framework had different fundamental concepts and characteristic, comparison between them will discussed n term of degree of developer friendly, performance, community support etc.

Table 2.5 Comparison between Flutter and React Native

Characteristics	Flutter	React Native
Developer friendly	Relatively Higher to React Native	Hight
Performance	High	Average
Learning Curve	Shallow	Average



Community support	Average, started to gain popularity	Large
Flexibility	High	Average

Based on the table above, Flutter is more developer friendly compared to React Native. The reason is Flutter support hot reloading that boost the development process. Developer doesn't need to wait the whole application recompile and render again. Apart from that, Flutter had relatively higher performance compared to React Native (Sharma and Gupta, 2020). Flutter has its strength of using Dart languages of its own high-performance rendering engine that doesn't need a JavaScript bridge to build the interaction with the device native components like React Native. As like previously stated, Flutter had its own rendering engine which mean it move the renderer from system level into the application level, which enabling high flexibility and more customizable user interface can be build. Considering several characteristics of both hybrid mobile application framework, Flutter is chosen to be used to develop hybrid mobile app in this project.

## 2.8 Review on Cloud Computing Services

Cloud computing services had become trend and mainstream rather than private on-premise hosting. There are several reasons for the scenarios, such as reliability, scalability, stability or availability of services. Through the implementation of cloud computing services, the service delegate many of works like configure complex server cluster computing system from developer to manage the server, storage and security. One of the benefits of cloud computing services compared to traditional hosting is the charging type, cloud computing services are pay-as-you-go basis, which mean customers only need to pay based on the resources they used.

There are several cloud services providers in the industry like Amazon Web Services, Google Cloud Platform and Microsoft Azure. Section below will evaluate the similarities and difference among the services provider stated above.

### **2.8.1 Amazon Web Services**

Amazon Web Services (AWS) that launched in year 2006 is quite a dominant in the cloud computing sector as it provides very complete and full-featured cloud platform for IT infrastructure like server hosting, storage, database instance hosting, messaging, load balancer etc. One of the services that AWS provide is Elastic Beanstalk which is a web hosting tools that support multiple popular language and framework in the industry such as Java, PHP, Node.js, Ruby on Rails, etc. Through Elastic Beanstalk, developer only need to upload the source code to the console and Elastic Beanstalk will automatically deploy and continuously monitor the availability (Amazon Inc., 2018).

The strength of AWS is all the service you need can be found in AWS bundle, like simple storage service (S3), relational database service (RDS), auto scaling, any kind of integration. The following are some unique feature of Amazon Web Services:

- Complete and well-equipped infrastructure and services
- Charge based on rounded up hour
- Built-in auto scaling group and load balancer
- Built-in security group to handle inbound and outbound traffic
- Easy deployment and provisioning
- Global reach

### **2.8.2 Google Cloud Platform**

Google Cloud Platform is introduced by Google in year 2008 and it provides various type of services such as PaaS, IaaS and serverless (Google Inc., 2020). As a later comer in the cloud services industry, Google Cloud Platform still aggressively grow and share a number of market segment because its strength and experienced in artificial intelligence computing. By its excellent performance in deep learning algorithm, machine learning and data analytics, developer can utilize this significant advantage to produce powerful artificial intelligence related system.

However, at the downside, Google Cloud Platforms doesn't offer as many various types of different services and features as AWS and Azure. On the other hands, there might be some latency if developer country is far from the data centers as Google

doesn't have as many global data centers as AWS and Azure, although it expands quickly. The following are some unique feature of Google Cloud Platform:

- Outstanding performance in deep learning, machine learning and data analytics
- Charge based on rounded up minutes used (minimum 10 minutes)
- Flexible discounts and contracts
- Offer hybrid cloud

### **2.8.3 Microsoft Azure**

Microsoft Azure is a cloud services that introduced by Microsoft in year 2010 and it come with five main components which is Compute, Storage, Content Delivery Network, SQL Azure and Azure Fabric Controller. Microsoft Azure is favorable by many enterprises as it well-integrate with other Microsoft application and services. This build a good ecosystem and customer loyalty for existing Microsoft customers. In additions, a considerable discount on services charging will be offers to the existing customers. However, one of the drawbacks of Microsoft Azure is it supported limited languages such as .NET, Java, C#, JavaScript, etc.

The following are some unique feature of Microsoft Azure:

- Advantages to existing customer
- Good integration with other Microsoft services
- Offering virtual machines (VMS) oriented services
- Auto Scaling
- Global reach

### 2.8.4 Comparison of Cloud Computing Services

Cloud computing has fundamentally improved the way of developments and operate software applications. Cloud computing reduces the expense and difficulty of reviewing, buying, configuring and maintaining all the equipment and software needed for enterprise applications at its heart. Each of those cloud computing services has its pros and cons. Comparison between among them will be discussed below.

Table 2.6 Comparison of Amazon Web Services, Google Cloud Platform and Microsoft Azure

Characteristics	Amazon Web Services	Google Cloud Platform	Microsoft Azure
Pricing	Based on rounded up hours	Based on rounded up minutes	Based on rounded up commitments and minutes
Strength	More experience in general solution and complete services	Specialized in artificial intelligence	Good ecosystem in Microsoft services
Flexibility	More open source tools integration	Average	Average
Community support	Large community	Average community	Average community
Documentation	Well-written	Well-written	Well-written

Based on the table above, Amazon Web Services is more experienced in general solution and had more complete services compared to Google Cloud Platform and Microsoft Azure (Dutta and Dutta, 2019). Besides, there are also many open source tools that can be integrate into Amazon Web Services which enabling it had high flexibility and scalability.



Figure 2.4 Gartner "magic quadrant"

The figure above shows the relative position of AWS, Google Cloud Platform, Microsoft Azure, Alibaba Cloud, Oracle Cloud and IBM Cloud in the evaluation of ability to execute versus completeness of vision as of July 2019 (Huang, 2019). After comprehensive consideration of the parameters stated above, we can draw a conclusion that Amazon Web Services would be a good choice for a wide range of project.

There is no universal best solution for every project when it comes to cloud computing services. After comprehensive consideration of the parameters stated above, Amazon Web Service will be chosen as the cloud computing services in this project.

## CHAPTER 3

### METHODOLOGY AND WORK PLAN

#### 3.1 Introduction

In this chapter, the methodology of the project and work plan throughout the project timeline will be discussed. Every software project can be described by the software development lifecycle (SDLC), it breaks down those project different phases. In more specific viewpoint, there are plenty of software development methodology framework to implement different flow software development lifecycle. The software development implements in this project will be Kanban Agile.

Furthermore, 13-weeks Gantt chart and work breakdown structure chart are constructed to outline and manage the project by break down the project into smaller components that is more manageable. Development tools involved in this project mentioned and briefly described in this chapter as well.

#### 3.2 Development Methodology

The applications development methodology that is being chosen for this project is Agile Software Development Model with Kanban approach.

Agile software development lifecycle achieved high effectiveness and efficiency by focus on multiple iterative and incremental process models on process development and requirement alignment by rapid delivery of product.

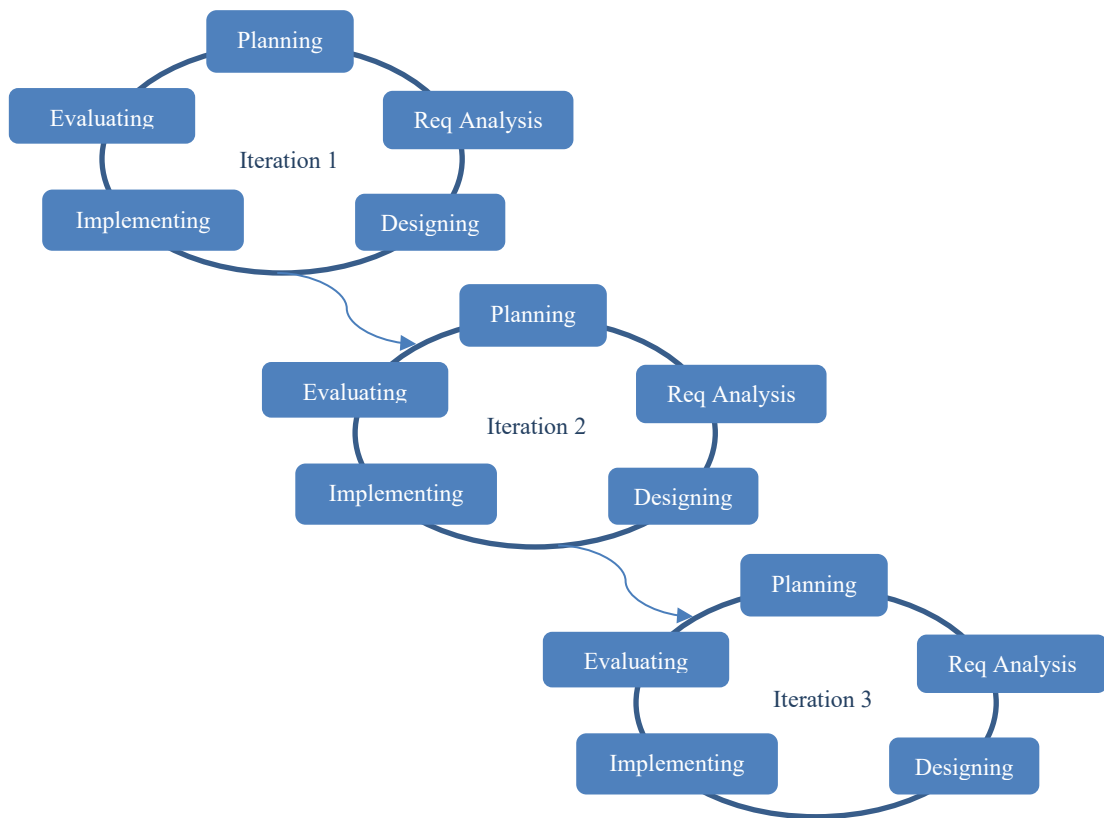


Figure 3.1: Iteration of Agile

Agile model focuses on the incremental process of developing software by recycle the software development lifecycle iteratively. In each iteration, it consists of planning, requirement analysis, designing, implementing and evaluating phases. Under Agile model perspective, a final product that fulfil at most requirement will be delivered after multiple iteration of development that successively build from the before iteration. There multiple framework under Agile principles such as Scrum, Kanban, Extreme Programming (XP), Feature Driven Development (FDD), Dynamic System Development Method (DSDM), Adaptive Software Development (ASD), Crystal and Lean Software Development (LSD) (Stackify, 2017).

By apply Agile model, project can gain strength and benefit as it allow for change at any time. Throughout the whole process that involve multiple iteration, there will always has opportunities to constantly refine, improve and reprioritize the work item and those changes should contribute to better fit to the requirement.

Kanban is a term in Japanese that literally means “visual card” (Ahmad, Markkula and Oivo, 2013). Kanban card idea is origin by Taiichi Ohno in Toyota

manufacture industry to limit the amount of inventory stuck up in “work-in-progress” on the manufacturing floor (Fries, 2019). When Toyota applied this Kanban system to its manufacturing floor, the result was aligning better their massive inventory with the actual consumption of materials.

As Kanban had evolved and utilized by many Agile software development teams popularly in this decade, it gives those teams more flexible planning space, more productive output, clearer focus to achievement and enough transparency throughout the software development lifecycle. Kanban approach is being built on the foundation of three main foundation, which is visualization of work items, limit the work-in-progress (WIP) task and focus on flow enhancement. Visualization of work items in Kanban present all the work items in a context that can be very intuitive manner by splitting the complicating structure of work into well-defined section or states. By limiting work-in-progress (WIP) tasks at one time, time and energy can be more focus or concentrate on certain high priority tasks and reduce time waste on switching from one task to another. Flow enhancement explained that when some task had been completed, the next high priority task from the backlog is pulled into develop. By implementing the core concept of Kanban, it promotes continuous delivery and encourage active, ongoing learning and improving the developed product by defining the best possible development workflow cycle through multiple iteration. As in Agile development model, there is no boundary approach and the workflows will across multiply functions without wait-time.

In the implementation of Kanban, a digital or physical board will be used to visualize each segment of work items throughout the project. A Kanban board provide the high transparency to the stakeholder of the project and emphasize communication as and when it necessary. Progress of the project also visualize by the Kanban board and it ease the process to track the process as it is very intuitive. By utilizing Kanban board, work items completion is emphasized, and bottlenecks will be obvious whenever it occurred.



### 3.3 Proposed Workplan

In this part, a proposed workplan for the project had been described by project schedule table, Gantt chart and Work Breakdown Structure. It provides an overview to the task item in the project and act as the guidance throughout the project.

Table 3.1: Project Schedule Summary

Work Task	Duration	Start	Finish
<b>Preliminary Planning Phase</b>	<b>21d</b>	<b>9 Jun 2019</b>	<b>29 Jun 2019</b>
- Background research	3d	9 Jun 2019	11 Jun 2019
- Problem statement	4d	11 Jun 2019	14 Jun 2019
- Determine project objectives	3d	13 Jun 2019	15 Jun 2019
- Determine possible project solution	6d	15 Jun 2019	20 Jun 2019
- Determine project approach	4d	20 Jun 2019	23 Jun 2019
- Determine scope of project	6d	24 Jun 2019	29 Jun 2019
<b>Requirement and Analysis Phase</b>	<b>25d</b>	<b>30 Jun 2019</b>	<b>25 Jul 2019</b>
- Review on existing similar community association system	2d	30 Jun 2019	01 Jul 2019
- Review on project methodology	2d	02 Jul 2019	03 Jul 2019
- Review on backend server framework	2d	04 Jul 2019	05 Jul 2019
- Review on front-end web application framework	2d	05 Jul 2019	06 Jul 2019
- Review on cross-platform mobile application framework	2d	06 Jul 2019	07 Jul 2019
- Review on cloud computing service	2d	07 Jul 2019	08 Jul 2019
- Interview for fact-findings	1d	09 Jul 2019	09 Jul 2019
- Observation for fact-findings	1d	10 Jul 2019	10 Jul 2019
- Questionnaire for fact-findings	3d	11 Jul 2019	13 Jul 2019
- Requirement analysis for resident related functionalities	3d	13 Jul 2019	15 Jul 2019
- Requirement analysis for guard related functionalities	2d	15 Jul 2019	16 Jul 2019
- Requirement analysis for admin related functionalities	3d	16 Jul 2019	18 Jul 2019

- Requirement specification	4d	19 Jul 2019	22 Jul 2019
- Determine project methodology	1d	23 Jul 2019	23 Jul 2019
- Determine appropriate development tools	3d	23 Jul 2019	25 Jul 2019
<b>System Design Phase</b>	<b>30d</b>	<b>26 Jul 2019</b>	<b>24 Aug 2019</b>
- System architecture design	5d	26 Jul 2019	30 Jul 2019
- Software design pattern	5d	31 Jul 2019	04 Aug 2019
- Database design	5d	05 Aug 2019	10 Aug 2019
- User interface design	5d	11 Aug 2019	16 Aug 2019
- Screen prototyping	10d	15 Aug 2019	24 Aug 2019
<b>Implementation Phase</b>	<b>64d</b>	<b>13 Jan 2020</b>	<b>16 March 2020</b>
- Develop resident side modules 1. Fee related function 2. Announcement related function 3. Feedback related function 4. Visitor related function 5. Others function	39d	13 Jan 2020	20 Feb 2020
- Develop security personnel side modules 1. Visitor related function	23d	10 Feb 2020	03 Mar 2020
- Develop management side modules 1. Fee related function 2. Announcement related function 3. Feedback related function 4. Visitor related function 5. Others function	43d	03 Feb 2020	16 Mar 2020
<b>Testing Phase</b>	<b>41d</b>	<b>18 Feb 2020</b>	<b>28 Mar 2020</b>
- Unit testing	14d	18 Feb 2020	02 Mar 2020
- Integration testing	11d	02 Mar 2020	12 Mar 2020
- System testing	11d	13 Mar 2020	23 Mar 2020
- User acceptance testing	6d	24 Mar 2020	29 Mar 2020
<b>Deployment Phase</b>	<b>3d</b>	<b>30 Mar 2020</b>	<b>31 Mar 2020</b>
- Deploy Android version	1d	30 Mar 2020	30 Mar 2020
- Deploy iOS version	2d	30 Mar 2020	31 Mar 2020

**A MOBILE APP FOR COMMUNITY ASSOCIATION**  
Read-only view, generated on 19 Mar 2020

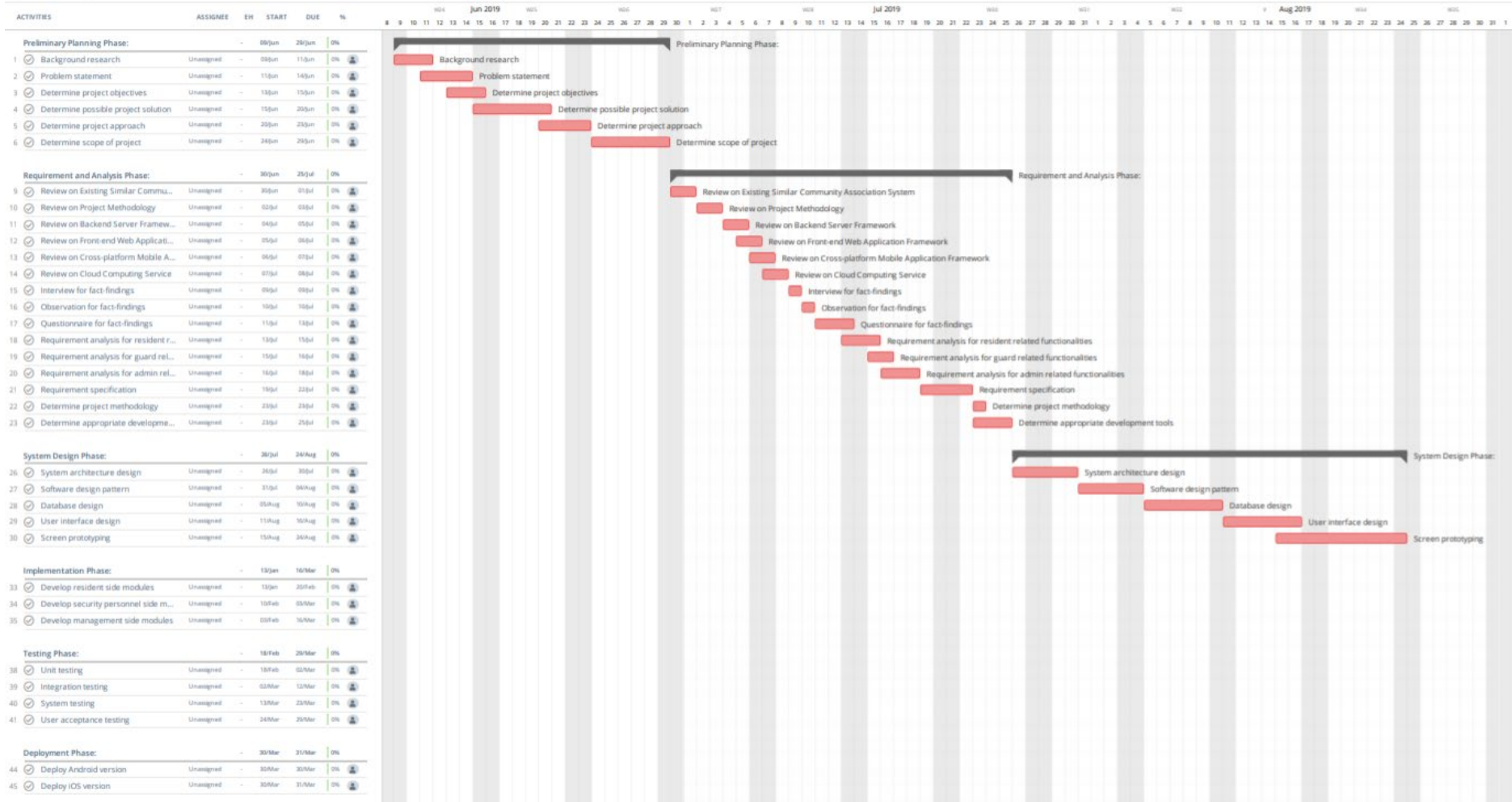


Figure 3.2: Part 1 of project Gantt Chart

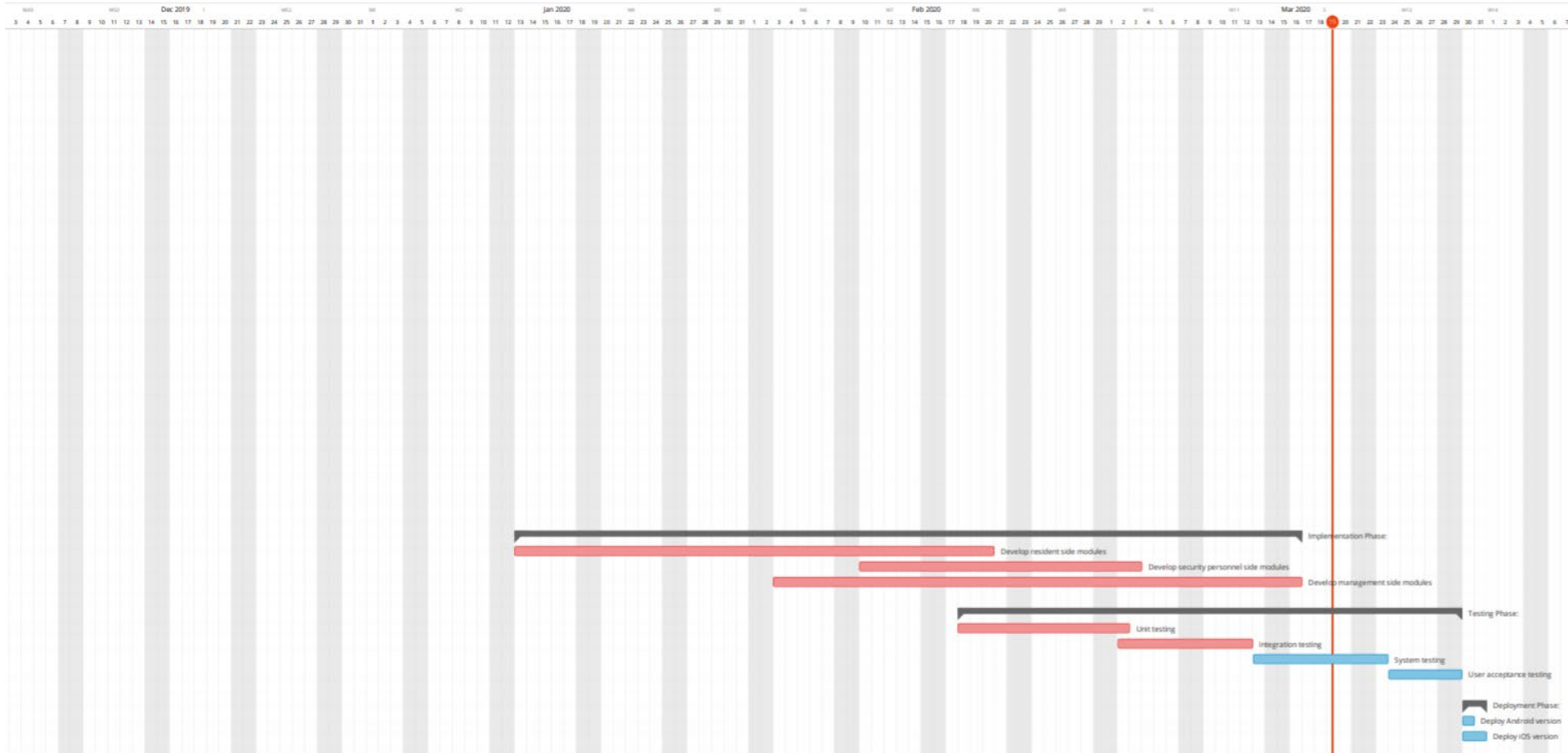


Figure 3.3: Part 2 of project Gantt Chart

A work breakdown structure is shown to visualize the work items that need to be carry out in each phase.

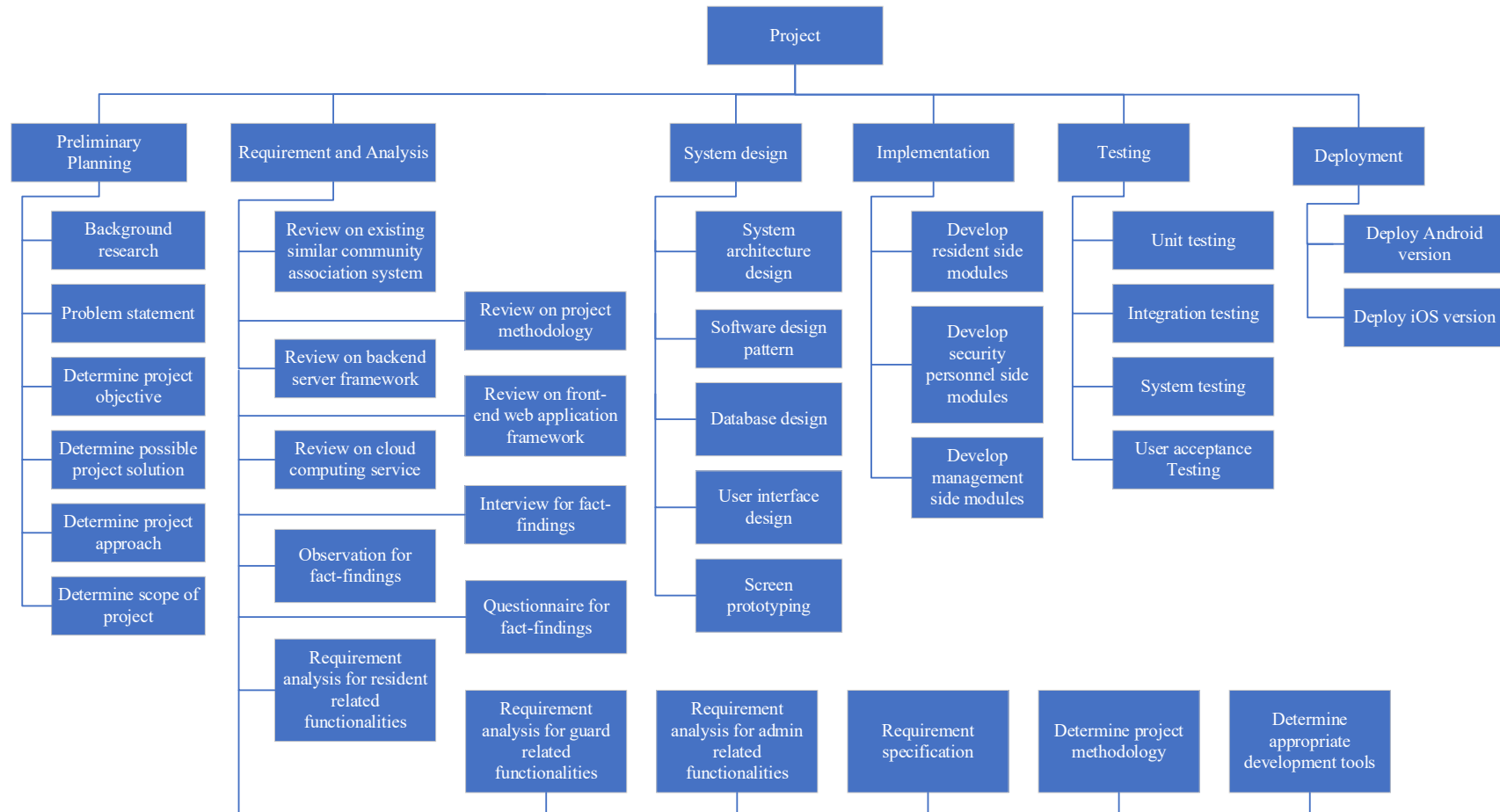


Figure 3.4: Work breakdown structure of the project

As Kanban Agile is the development methodology of this project, it consists of generally 5 phases for each iteration, while there may have multiple iterations before the final product. The work items in each phase will be explained below:

1. Planning

In this phase, a well-scheduled plan should be constructed by a series of investigation that gain more understanding on the relevant information to the project. That information will be used to plan the project approach and conduct study in the economic, operational and technical aspect. Planning is the foundation of the project; it is essential to ensure the successful of the project. Gantt chart had been created to illustrate how each work tasks of this project carried out along the timeline to ease the management and more trackable. Besides, work breakdown structure constructed to separate each work tasks into collection and this reduced the complexity of the project. Problem formulation had been done in this phase to identify the problem exist in the current system. The study of the background also been conducted through literature review. After that, project objective proposed and finalize to give the project a specific, measurable and achievable target.

2. Requirement Gathering and Analysis

In the second stage of the software development lifecycle, requirement of the system should be gathered and analyse those requirements for the software. There are few requirement gathering methods such as questionnaires, interviews, inspection and literature reviews etc. After the requirement analysis is done, requirement specification should be carried out to refine and document the product functional and non-functional requirement. Requirement should be verifying and analyse based on their validity and incorporation to the project. One of the outcomes of this phase is a requirement specification document which guide the next phase of the project.

3. Designing

In designing phase, the information of requirement gathered should be used to construct the system architecture design. Multiple software design diagram should be drawn based on the requirement specification document. By using Enterprise Architecture, UML diagrams such as use case diagram, sequence diagram, class diagram and entity diagrams is produced to illustrate the

architecture of the system and how the system works. On the other hand, screen prototyping carried out to design the user-friendly interface by using the development tool, which known as Axure. The implementing of the software should base on the user interface prototype produced in this phase.

#### 4. Implementing

Once the system design is done and finalized, the implementation of the software begins. The whole implementation of software will break into several modules or components to make the work tasks more manageable. All the work tasks should clip on the digital Kanban board that powered by Trello to visualize and track the progress of development. At the end of this phase, consistency and quality of the product iteration produced will be evaluate in the next phase.

#### 5. Evaluating

After each iteration of product had been developed, the implementation should always evaluate to align with the requirement specification to ensure the final product will in the scope of the system requirement. Testing will be conducted to find any potential or existing bugs or measure the product against the requirement specified. Testing process that included functional testing, non-functional testing, integration testing, system testing, and acceptance testing should be completed in this phase.

### **3.4 Technology and Development Tools Involved**

Software development is a complex process that involving multiple problem from different aspect. A suitable software development tools and right technologies implemented will definitely ease the process and increase the productivity of the development.

#### **3.4.1 Ruby on Rails**

For the backend API server that serve the mobile application and web application clients, Ruby on Rails, a web application development framework written in the Ruby programming language is used. One of the strength of Rails is that it is a Model-View-Controller (MVC) friendly framework which is suite with this project software design pattern, it supports database, web server and web pages with default structures. It

promotes and enables the use of web standards for data storage such as JSON. In additions, Active Record in Rails is a great advantage in implementing MVC software pattern, as it serves as the Model in MVC. It provides multiple benefit such as easy development of reading, writing data and defining relationship between each model. The Rails backend server maintain high availability and reliability to handle and respond the request from resident mobile application or admin web application.

### **3.4.2 Vue.js**

Vue.js is used in this project as a front-end web application framework that written in JavaScript. Philosophy of Vue.js is about progressive development to build user interface, which let it easy to learn and develop. Additionally, it has well-constructed high-level architecture for consolidating the state, Vue instance method, custom methods, so developers can easily understand the code. Vue.js is very flexible and less fragmented as it focusses on view layer only. It gains high popularity as it facilitates flexibility which make it can be easily integrate to existing applications by adding Vue.js CDN. Most of the third-party libraries and components are also available and supported by Vue.js CDN, npm or node is no more a necessarily to install libraries or package. By using Vue.js, web application that consists of beautiful and elegant user interface can be produced efficiently in this project.

### **3.4.3 Flutter**

In the world of mobile application development, development cycles, deployment time and quality always been the concern of developer. As well-known, current mobile application market had dominated by two main operating system which are Android and iOS. To optimizing the process of development, saving time and cost, hybrid app frameworks was introduced to solve the obstacle. Instead of developing two projects for Android device and iOS device, hybrid app frameworks allow the same project shared by both platforms that mentioned. Flutter is one of those hybrid app frameworks that can natively build a cross-platform applications for Android and iOS in JavaScript framework. Flutter is chosen to use in this project to develop the mobile application as it covers both Android and iOS platforms, natively and efficiently support the rendering of hybrid mobile application. Flutter enable quick development of mobile



application and single code base of this project can be deploy to iOS and Android platform by minimum configuration efforts.

#### **3.4.4 Amazon Web Services**

Amazon Web Services (AWS) is used in this project for several purpose such as backend server hosting, computing power and relational database instance hosting. AWS Elastic Beanstalk enable the quick deployment and easy management of application in cloud services without configuring complex servicer architecture. It delegates the tasks such as load balancing, auto scaling and provisioning of the web application from developer, which free the time of developer so developer can focus of the software development itself. The backend server of this project is deployed to Elastic Beanstalk environment on the platform on Puma with Ruby 2.6 that running on 64bit Amazon Linux.

Apart from that, MySQL database instance of this project that used to store data is created using relational database service (RDS) of Amazon Web Services. Amazon RDS provides reliable infrastructure and offering flexible, scalable and secured database solution with low-cost, developer can avoid the complexities of creating a new production database from scratch. As the Amazon Web Service official website stated, there are several well-known featured companies that using Amazon RDS such as, Expedia, Netflix, Airbnb, etc.

#### **3.4.5 Heroku**

Similar to Amazon Elastic Beanstalk, Heroku is a cloud services provider that offering Platform as a Service (PaaS) in container-based basis to customers. It is used in this project to deploy the web application for community association management usage. Heroku support the auto deployment whenever there is new commit at the master branch it observed, which enabling the continuous deployment of project.

#### **3.4.6 Firebase Cloud Messaging**

Firebase Cloud Messaging (FCM) is a cross-platform messaging service that provides various type of message sending approach. By using FCM in this project, the mobile application of resident will be notified instantly whenever there are new announcement or update on feedback case. For the real-time notification, a message can transfer with

a payload of maximum 4KB of size to the client application. To programmatically send notification message to resident, Ruby FCM SDK is used in the backend Ruby on Rails server for communicating with the FCM protocol.

### **3.4.7 RQRCode**

RQRCode is a Ruby library that provide QR codes creation and rendering function. It is implemented to generate QR codes when the visitor request is accepted and render it into the format extension defined such as SVG or PNG.

### **3.4.8 Visual Studio Code**

An integrated development environment (IDE) is essential for software development. In this project, Visual Studio Code (VS Code) will be comfortable choice. VS Code is a very lightweight but powerful source code editor which can run on Windows, macOS and Linux operating system. It is very extensive, customizable and included supports for multiple programming language by extension as well as Flutter. Besides, VS code also embedded Git control to enhance the develop experience. In order to create a development environment for Flutter projects, an extension in the VS Code extension library, which called “Flutter Tools” can be install. The extension customizes the VS Code to better support to the Flutter project. Besides of extensive support of Flutter, VS Code also well-support Ruby on Rails and Vue.js with interpreter installed.

### **3.4.9 Git**

Version control system are one of the spirits of the software development. A well-define version control system will always keep track of every modification, commit and branch, provide fault-tolerance capability to the project. Version control tools always been contributing to the project by reducing the cost to manage multiple version of product and protect the source code in the repository. Git is a free and open-source distributed version control system and will be used to handle every change in this project natively. Apart from that, Git is also corporate use with Elastic Beanstalk Command Line Interface which make the deployment process extremely easy. By using the Elastic Beanstalk Command Line Interface, the Elastic Beanstalk environment will automatically retrieve the latest commit and use it as latest deployment source code.

### **3.4.10 Trello**

Trello is used in this project to act as a digital Kanban board. It gives a visualization of overview to the project work tasks. Trello consist of a numerous board, lists and cards. In this project, several lists are created such as “General backlog”, “Mobile app backlog”, “Work-In-Progress”, “Done”, etc. By utilizing Trello as Kanban boards, it helps the project doesn’t lost direction and prevent extra switching time from one task to another takes.

### **3.4.11 Axure**

Screen prototyping of this project will be developed using Axure. Axure is a powerful tool to design, wire-framing and rapid user interface prototyping that suit with the need of this project. The user interface design will give a clear and straight forward visual guidance to the project and keep the progress align with the requirement and user interface design principles. Before proceeding to the development of user interface, Axure will be used to produce the screen prototype that can be act as the blueprint of mobile application UI and web application UI.

## CHAPTER 4

### PROJECT SPECIFICATION

#### 4.1 Introduction

This chapter will discuss about the fact-finding process (interview, observation and questionnaires) and information gathered and analysed. Besides, the information gathered and analysed at the previous phase will be used to produce the software requirement specification of the system. Furthermore, screen prototype included in this chapter.

#### 4.2 Fact-finding

There are several data gathering method involved in this project which is interview, observation and questionnaires.

##### 4.2.1 Interview

Firstly, an interview session was held with Mr. Loh Ka Keng on 10 July 2019. The interview session mainly is a discussion about his opinion about residential community association management and problem encountered in residential association. He stated that one of his problem encountered in residential community association management is need to personally go to the management office and take the hard copy receipt for the fee payment. As the office only open from 10.00 a.m. to 5.00 p.m. on weekdays, it become an inconvenience process. Mr. Loh, as a university student normally have class from 9.00 a.m. to 4.00 p.m., the office working hours always overlap with the Mr. Loh's class hours. Mr. Loh hard to find an available timeslot to visit the management office and get the hard-copy receipt.

The interview questions had attached in appendix section as "Appendix A".

#### 4.2.2 Observation

Observation on the visitor registration process had been held at Cypress Condominium, Bandar Sungai Long in 15 July 2019. The observation session found that the current visitor registration process basic steps are:

1. Visitor walk in to guard house personally.
2. Fill in the registration form.
3. Pass identity card or license card to guard as deposit.
4. Granted access.
5. Identity card or license card give back to visitor when visitor exit the residential area.

Some information required in the visitor registration:

- Name
- IC number
- Duration of the visitation
- Unit number that visit
- Car plate number (optional)

The visitor registration items of Cypress Condominium and related photo are attached in appendix part as “Appendix B”.

### 4.2.3 Questionnaire

Questionnaire is a useful technique in data gathering. In this specific questionnaire to get insight from real life, there are consist of 8 sections which is basic information, security/maintenance fee payment, resident's information update, feedback, announcement, visitor system and others section. Throughout the 8 sections, there are in total 18 questions. The questionnaire form had attached in appendix section as "Appendix C". Below shown the first question analysed chart:

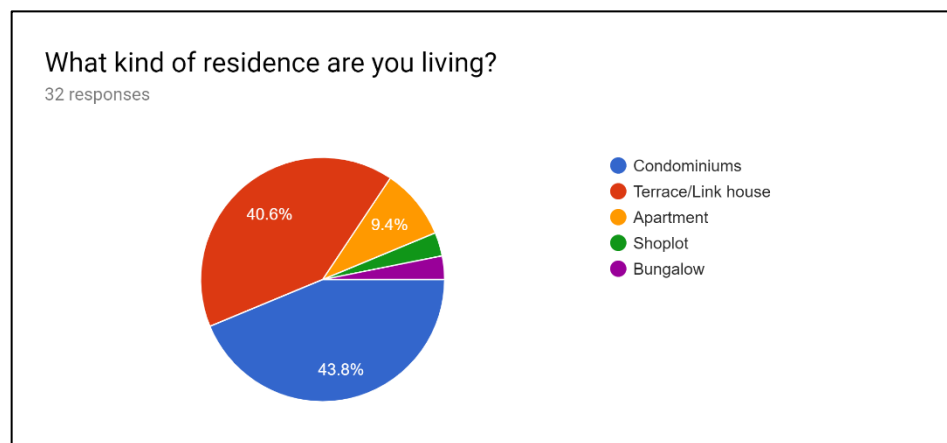


Figure 4.1: Pie chart of the residence type

The chart above visualized the percentage of residence type from the 32 responses. Majority of them are live in Condominiums or Terrace/Link house. There are 43.8% of respondents live in condominiums and 40.6% of respondents live in terrace/link house.

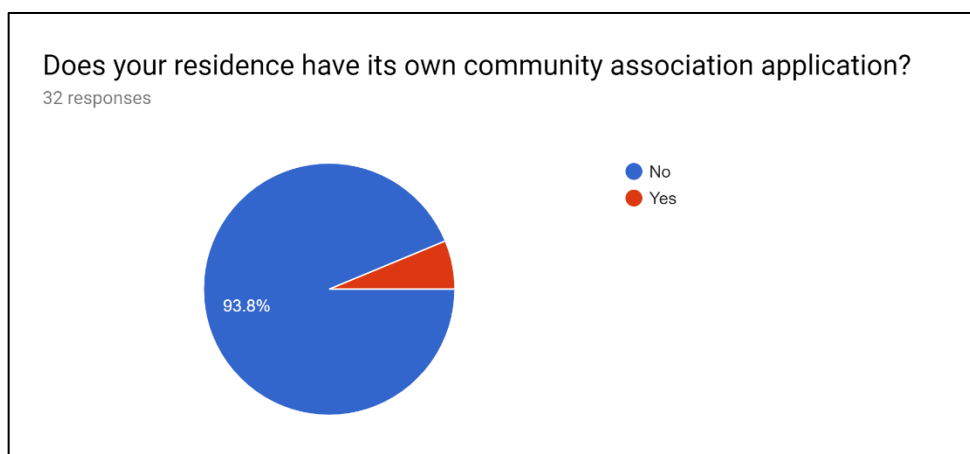


Figure 4.2: Pie chart of percentage of residence having a community association application

The chart above gave an information that most of the residential area in Malaysia doesn't equip with any community association application. Only 2 out of 32 respondent (6.3%) residential area have their own community association application. One of the community association application name submitted from respondent is MyTaman. This data show that community association management in Malaysia is in a very traditional and backward in term of technology.

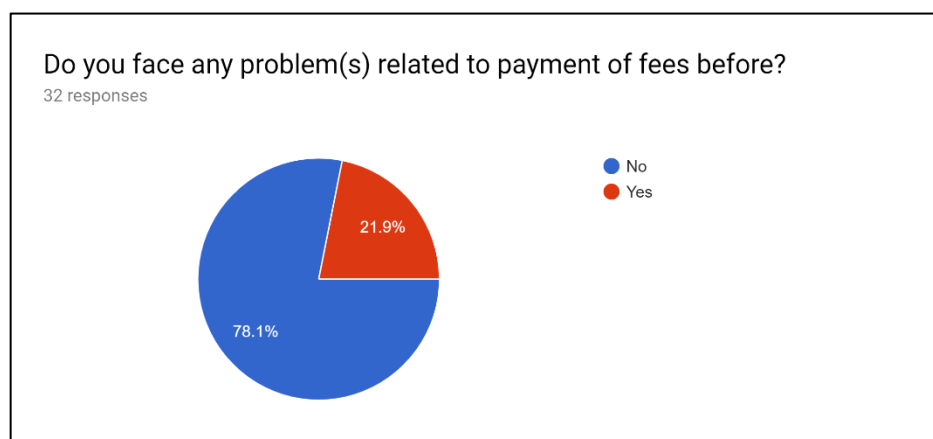


Figure 4.3: Pie chart of percentage of resident encounter fee payment related problem

Based on the Pie chart, a portion of respondent, which is 21.9% encounter one or more problem in fee payment related matters. From the problem the respondents stated, one of the problems is the individual need to personally go to guard house and take the hardcopy receipt of the fee payment that completed through online and it was an inconvenience process.

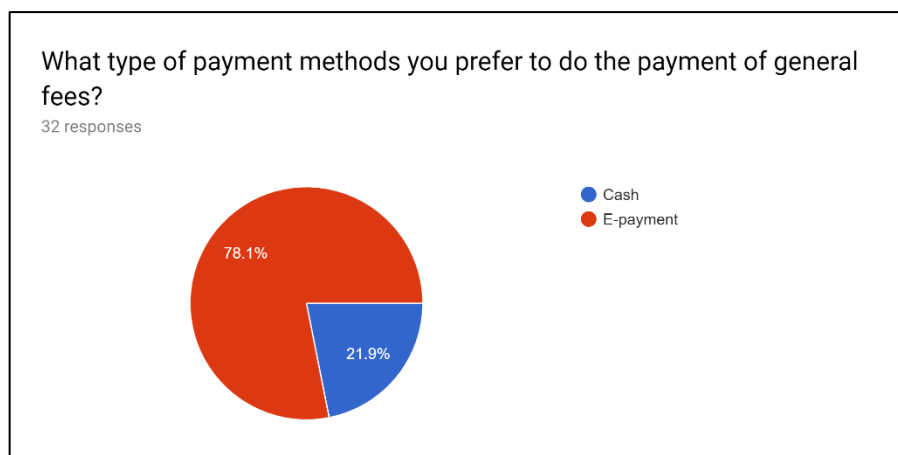


Figure 4.4: Pie chart of preferable payment method

There are almost 80% of respondents prefer the E-payment method as their fee payment method. One possible reason of this phenomena is E-payment is more convenience and residents can pay the payment whenever they want without considering the office hours and doesn't required personally visit the office. Besides, this 80% of respondents choose the electronic side as they prefer to receive electronic receipt instead of hard copy receipt.

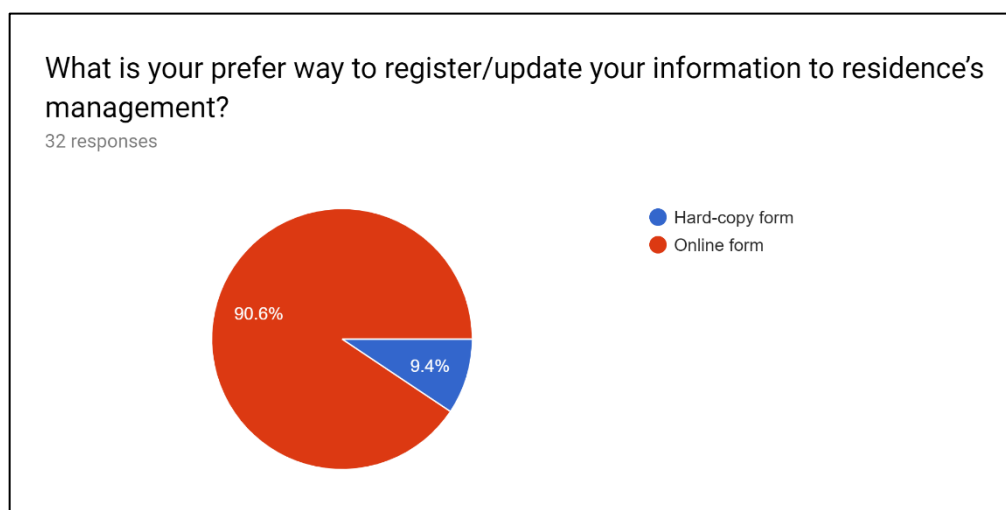


Figure 4.5: Pie chart of preferable way to manage resident's information

Majority of the respondents (90.6%) prefer to use online form to register/update their residential information. This can improve the level of convenience as resident can manage their residential information more easily.



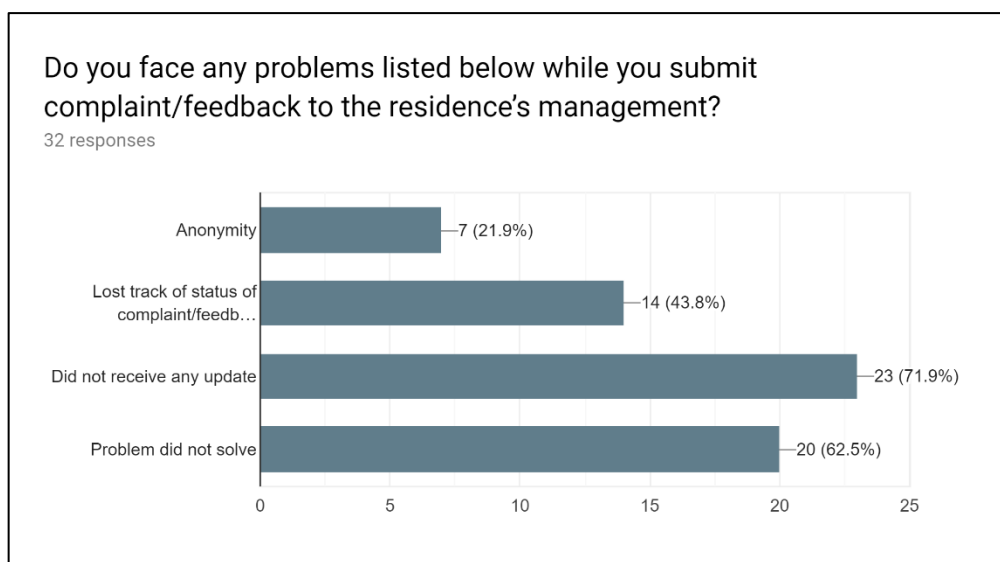


Figure 4.6: Bar chart of problem encountered in feedback/complaint process

The bar chart above shown that most of the respondents faced more than one problem. The most frequent problem to faced is didn't receive any update related to the feedback/complaint process from management team after submit the feedback/complaint. Large number of respondents are concerned about the problem didn't solved or ameliorate. Feedback/complaint that resident most probably to submit can be categorized into facilities problem, maintenance problem, parking problem, security problem, electricity problem, Wi-fi problem etc.

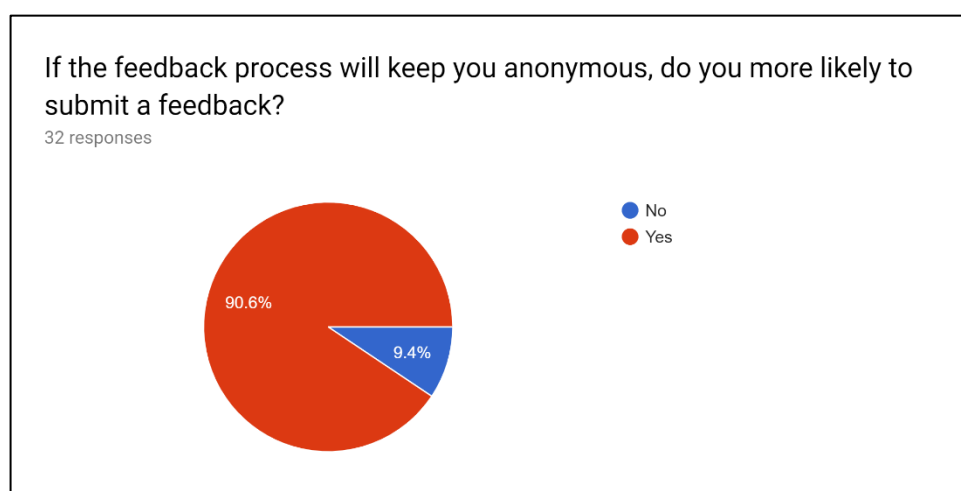


Figure 4.7: Pie chart of anonymous feedback

The pie chart shown that anonymous does promote the willingness to submit a feedback/complaint.

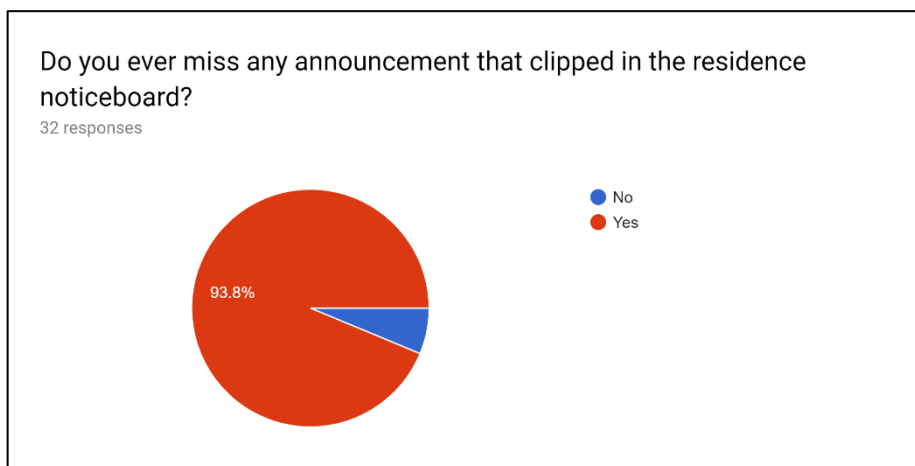


Figure 4.8: Pie chart of miss announcement

The pie chart shown that most of the residents had ever missed announcement that clipped in the residence noticeboard.

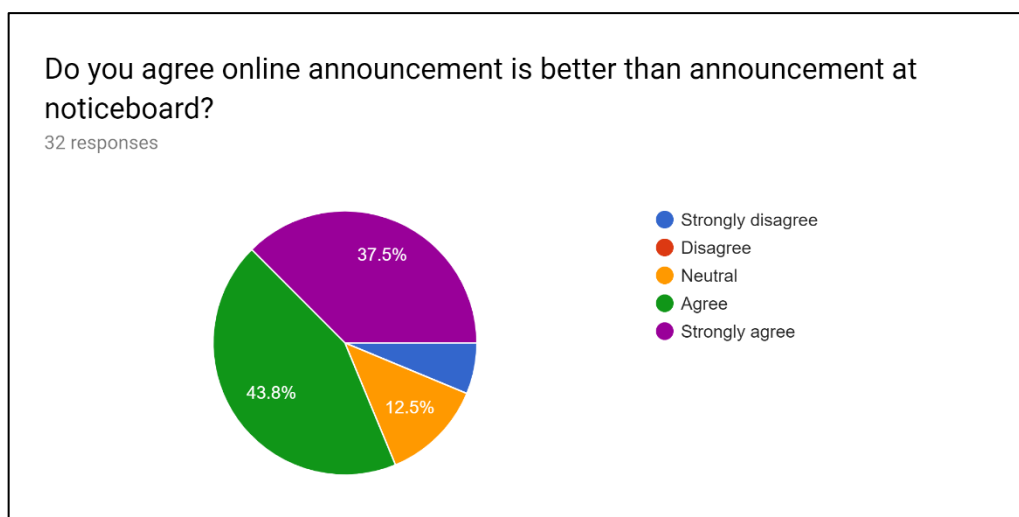


Figure 4.9: Pie chart of online announcement versus normal announcement

Majority of the respondent agree or strongly agree that online announcement will perform better than normal announcement that clipped at noticeboard. Notification can be sent to respondent to notify them about new announcement made.

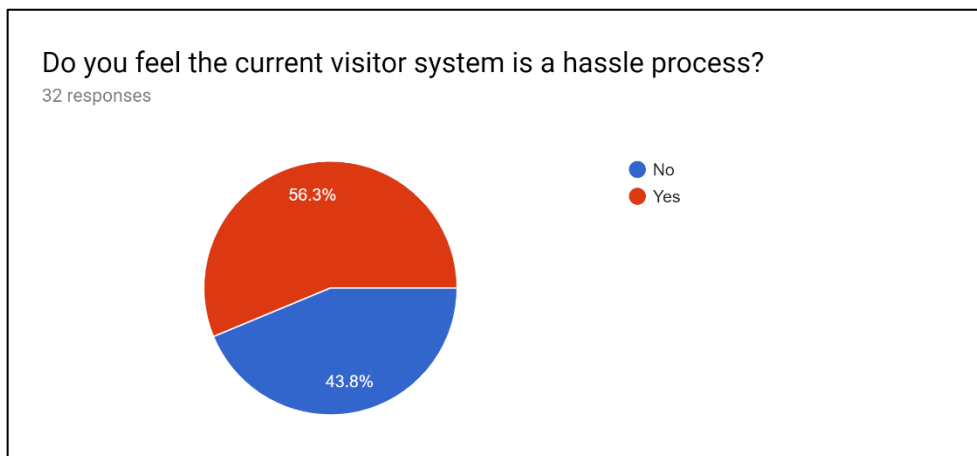


Figure 4.10: Pie chart of visitor system

56.3% percent of respondents have the opinion that current visitor system is a hassle process as need to fill in hard-copy registration form. In some case, the residents need to personally go to guard house and prove to guard that the visitor is valid. On the other hands, generally most of the respondent have 0-2 visitor in one week.

In the others session, an open-ended question had been asked to collect the opinion of respondents about the weakness current existing system. Inefficient is the main weakness of the current existing community association management system as it involved large amount of labour work and hard-copy form. Besides, communication skill and attitude of guard is also one of the considerations of this issues. Based on the response, some respondent stated that slacking of guard and inefficient community with guard are the weakness of current system.

### **4.3 Requirement Specification**

The aim of this software requirement specification (SRS) document is to provide a comprehensive summary, parameters and objectives of our software product. This document defines the target audience of the project and its specifications for user interface and software. It describes how the product and its functionality are viewed by our customer, team and audience. Nonetheless, it allows any software distribution lifecycle (SDLC) processes to proceed smoothly and guide designer and developer.

#### **4.3.1 Mobile Application for Community Resident**

This section will describe the requirements for each functionality of mobile application for community association resident.

##### **4.3.1.1 Fee/Bill Payment**

1. Resident user shall be able to view the user's outstanding bill balance required to pay.
2. Resident user shall be able to pay the user's bill through Stripe online payment gateway integrated into the application using credit card or debit card.
3. Resident user shall be able to view the details of the fee listed including fee title, fee description, fee creation time and fee due date.
4. Resident user shall be able to view the user's bill payment history.
5. Resident user shall be able to receive notification when there was a new bill payment required to pay.

##### **4.3.1.2 Announcement**

1. Resident user shall be able to view the announcements in list form.
2. Resident user shall be able to view the details for a specific announcement in details including announcement title, announcement content and announce time.
3. Resident user shall be able to bookmark or un-bookmark specific announcement.
4. Resident user shall be able to receive notification whenever there was a new announcement made.

#### **4.3.1.3 Feedback/Complaint**

1. Resident user shall be able to submit the user's feedback/complaint.
2. Resident user shall be able to track the user's feedback/complaint progress and status.
3. Resident user shall be able to update the user's feedback/complaint progress.
4. Resident user shall be able to mark the specific feedback/complaint as completed.
5. Resident user shall be able to rate the specific feedback/complaint process after he/she marks that specific feedback/complaint as completed.

#### **4.3.1.4 Visitor**

1. Resident user shall be able to submit visitor application for the visitor access by input the visitor information including, visitor name, visitor car plate number, visitor phone number and visit reason (optional).
2. Resident user shall be able to receive a QR code for visitor access after the visitor application approved.
3. Resident user shall be able to view the rejected reason if the visitor application was rejected.
4. Resident user shall be able to view the user's unit visitor history.

#### **4.3.1.5 Residential Information**

1. Resident user shall be able to view the user's residential information, such as name, unit address, phone number etc.
2. Resident user shall be able to edit the user's residential information, such as name, phone number, car plate number etc.

#### **4.3.1.6 Account**

1. Resident user shall be able to log in to the application by using the given pre-created account from the management.
2. Resident user shall be able to log out from the system.
3. Resident user shall be able to change their password.
4. Resident user shall be able to recover their password if they forgot it.

#### **4.3.2 Web-based Application for Management User**

This section will describe the requirements for each functionality of web-based application for community management user.

##### **4.3.2.1 Fee/Bill Payment**

1. Management user shall be able to view all community payment status in the system.
2. Notification shall be able to send to community residents' device when a new fee is created.
3. Management user shall be able to create new fees by specify the title, amount, due date and affected community resident of the fee.

##### **4.3.2.2 Announcement**

1. Management user shall be able to view the announcements list.
2. Management user shall be able to create an announcement by inputting the announcement title, content and publish status.
3. Management user shall be able to edit and update an existing announcement by inputting the announcement title, content and publish status.
4. Management user shall be able to delete an existing announcement.

#### **4.3.2.3 Feedback/Complaint**

1. Management user shall be able to view the feedback/complaint list.
2. Management user shall be able to update the feedback/complaint process.
3. Management user shall be able to view rating of specific feedback/complaint process that given by the community resident.

#### **4.3.2.4 Visitor**

1. Management user shall be able to view the visitor application list.
2. Management user shall be able to view a specific visitor application in details form.
3. Management user shall be able to approve or reject the specific visitor access registration.

#### **4.3.2.5 Residential Information**

1. Management user shall be able to view each community residential information.

#### **4.3.2.6 Account**

1. Management user shall be able to create a new account for new resident user.
2. Management user shall be able to create a new account for new security personnel user.
3. Management user shall be able to suspend specific account.

### **4.3.3 Mobile Application for Security Personnel**

This section will describe the requirements for each functionality of mobile application for community security personnel user.

#### **4.3.3.1 Visitor**

1. Security personnel user shall be able to scan the QR code present by visitor approach and view the approved visitor application details, including name of visitor, car plat number of visitors, phone number of visitors.

#### **4.3.3.2 Account**

1. Security personnel user shall be able to log in to the application by using the given pre-created account from the management.
2. Security personnel user shall be able to log out from the system.



## 4.4 Use Case Modelling

This section will visualize and modelling the flow of the use cases involved in this system.

### 4.4.1 Use Case Diagram

Use case diagram is used to visualize the users (community resident, community management and security personnel) interactions with the community association system. Several use case diagrams are created to describe relationship between users and use case.

#### 4.4.1.1 Fee/Bill Related

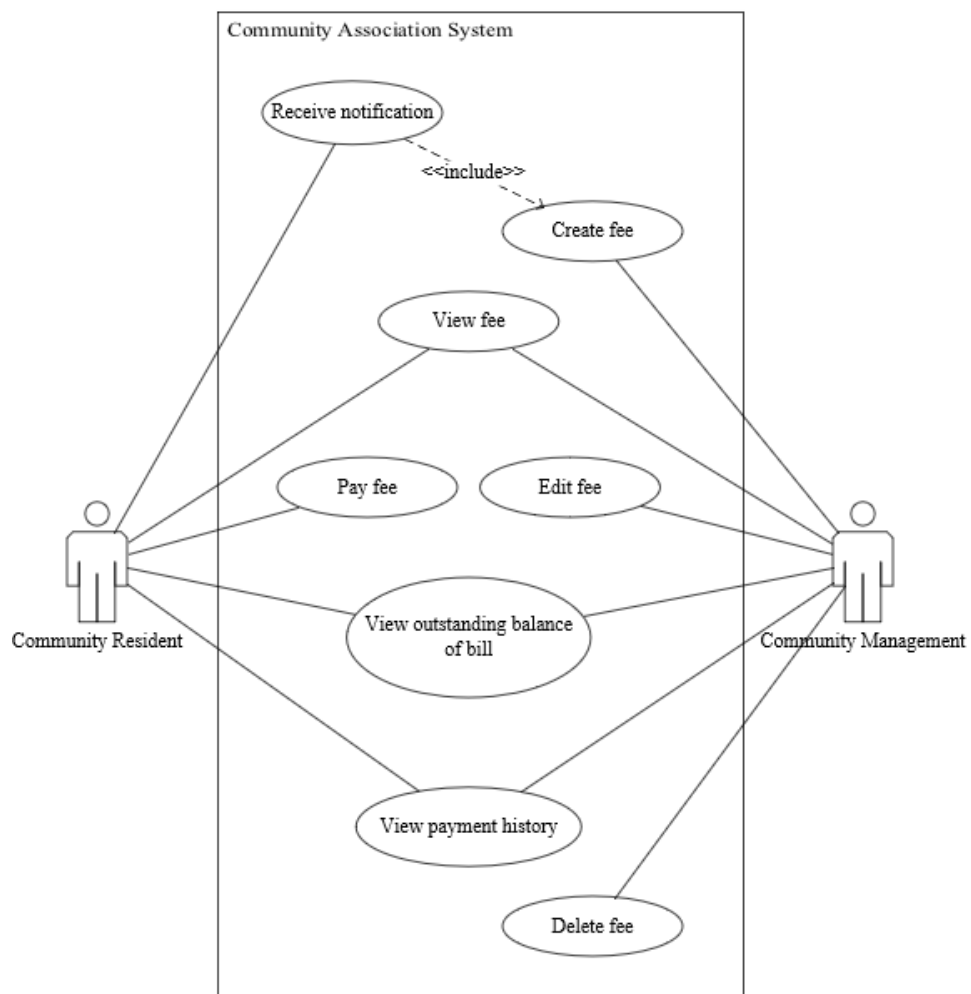


Figure 4.11 Fee related use case diagram

#### 4.4.1.2 Announcement Related

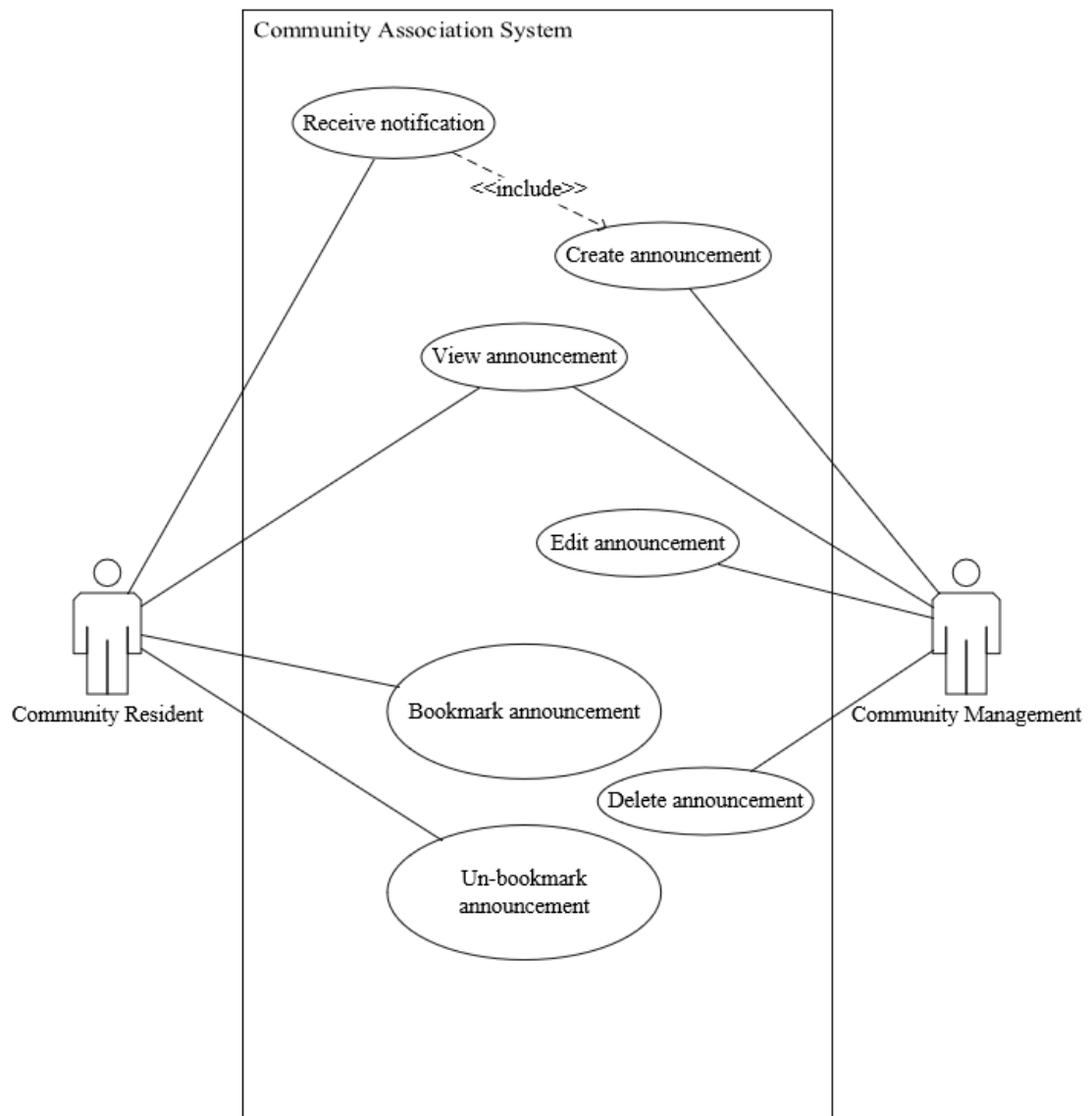


Figure 4.12 Announcement related use case diagram

### 4.4.1.3 Feedback Related

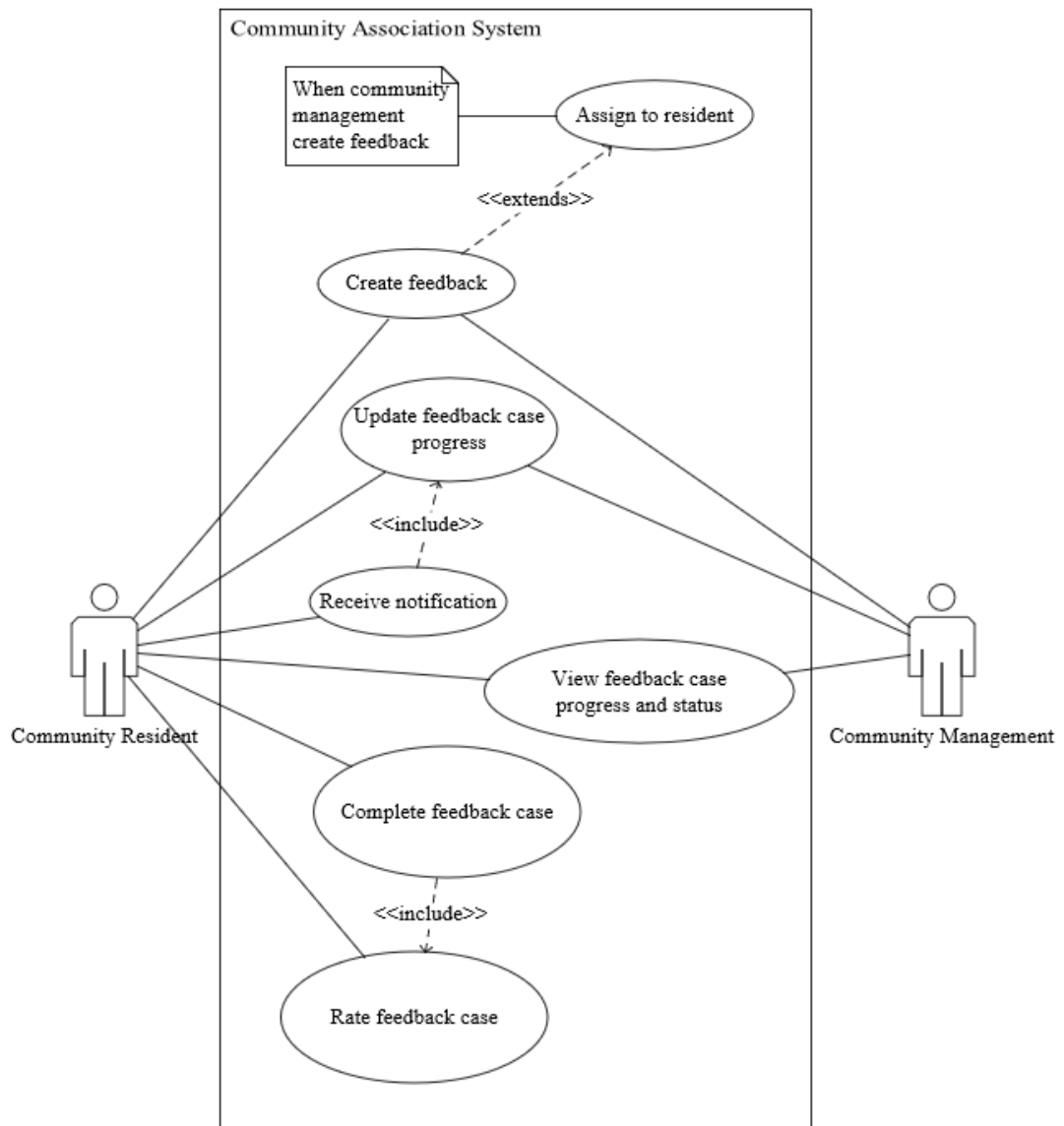


Figure 4.13 Feedback related use case diagram

#### 4.4.1.4 Visitor Related

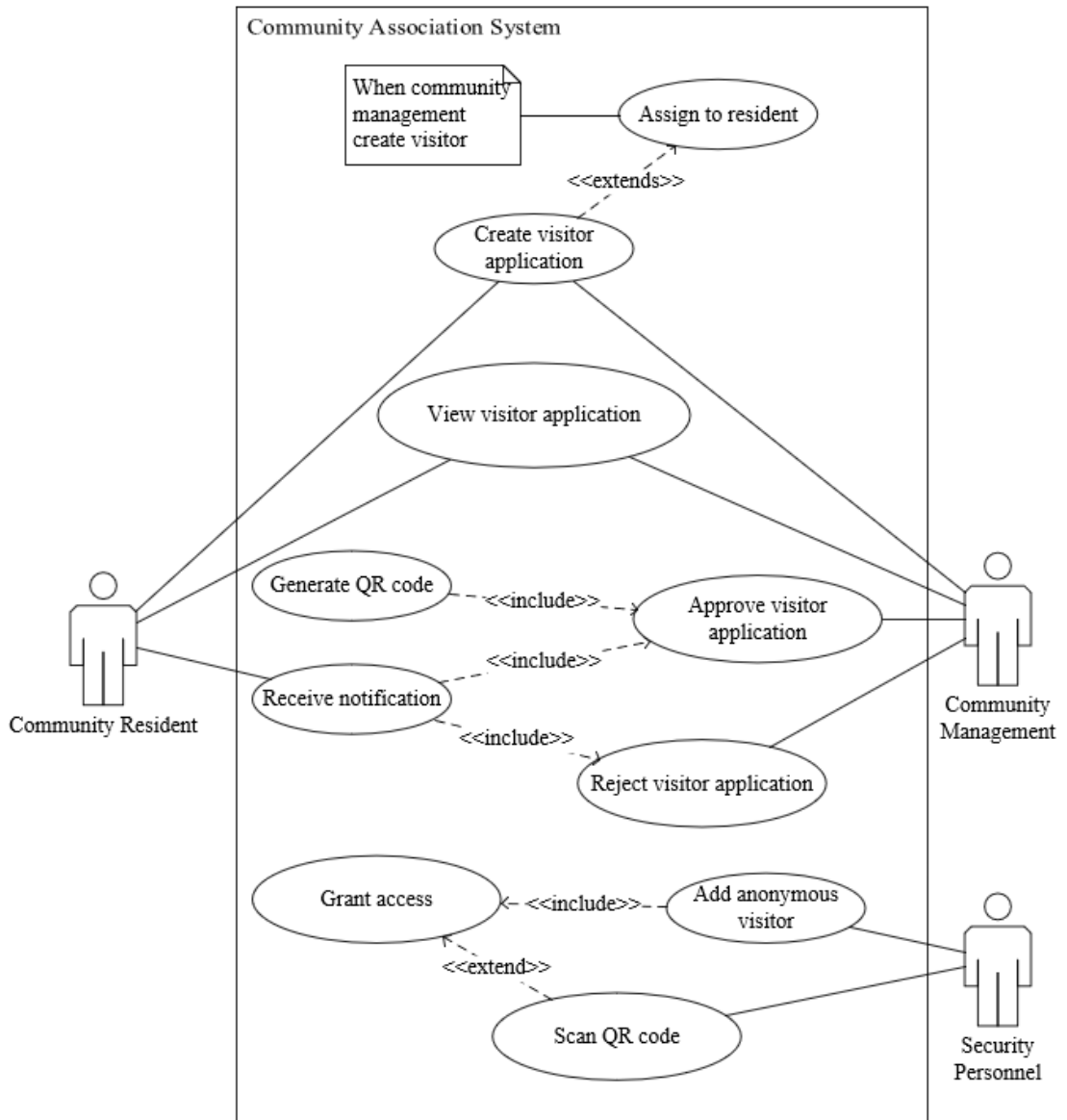


Figure 4.14 Visitor related use case diagram

#### 4.4.1.5 Resident Information Related

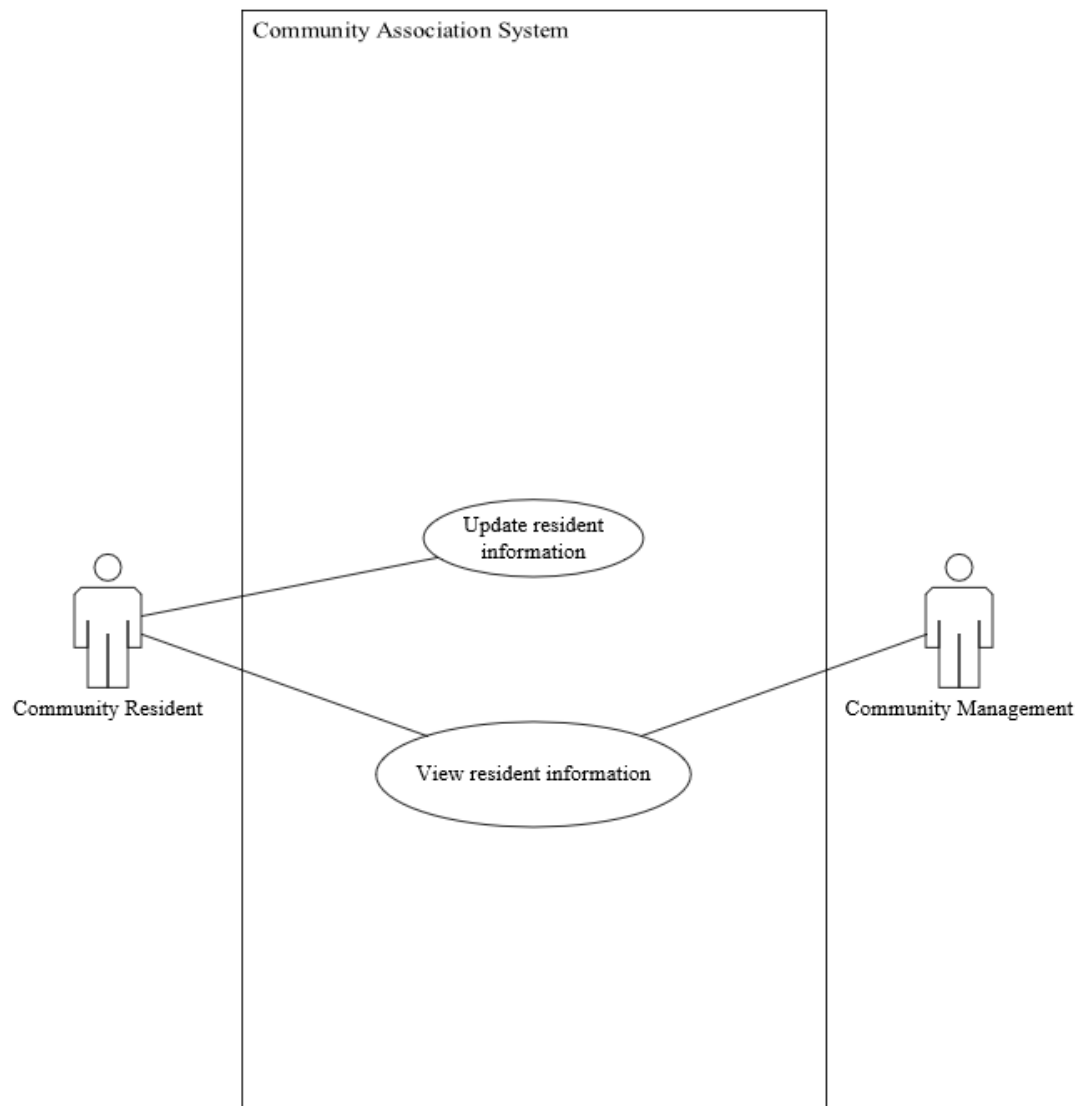


Figure 4.15 Resident information related use case diagram

#### 4.4.1.6 Account Related

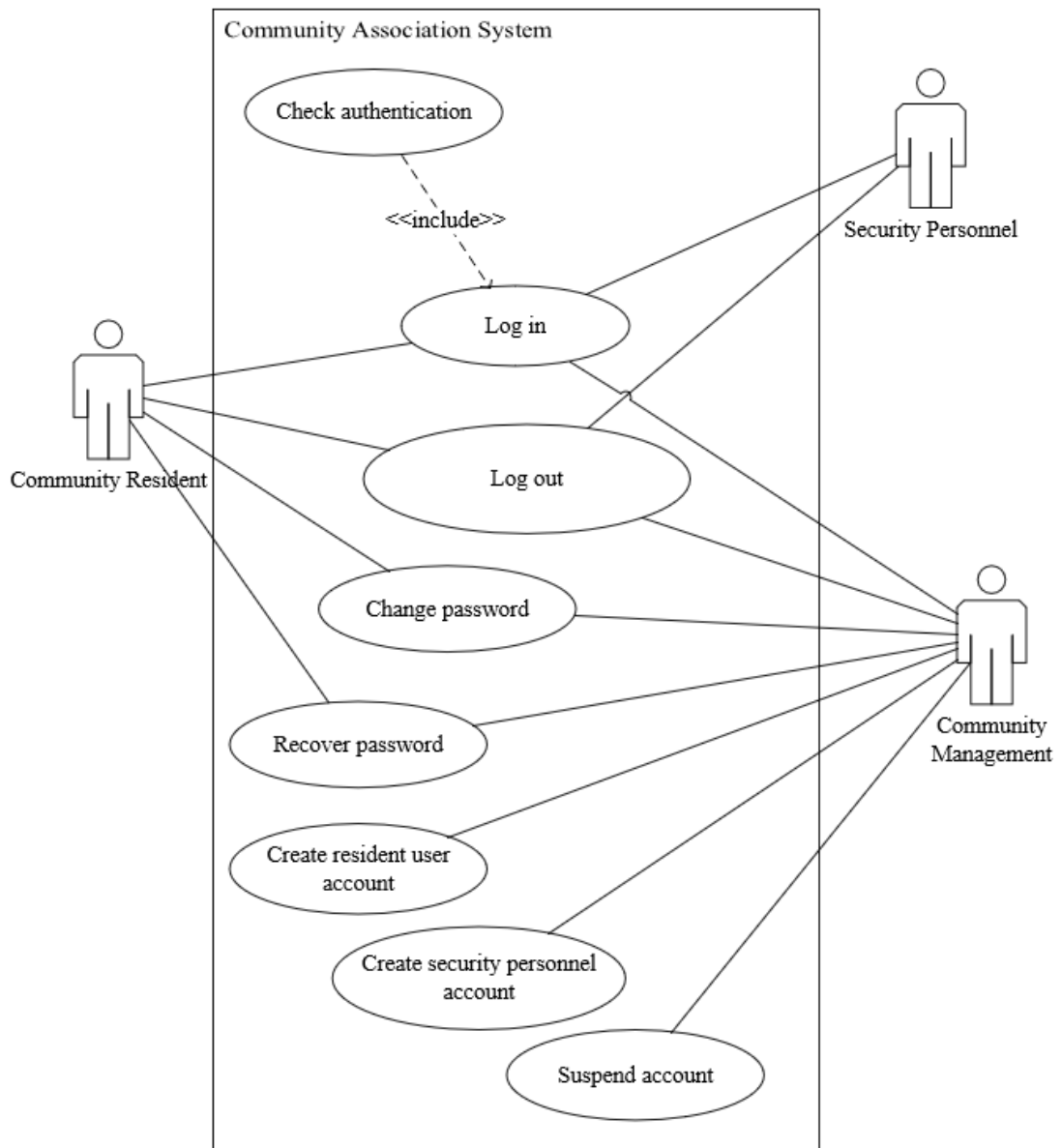


Figure 4.16 Account related use case diagram

#### 4.4.2 Use Case Description

This section describes each use cases of how users will perform functionality of the system.

##### 4.4.2.1 Mobile Application for Community Resident

Use case name	Pay fee
Actor	Community resident
Description	Community resident pay fee using the payment gateway integrated into the mobile application.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident select a fee to pay.</li> <li>2. Community resident filled in credit/debit card details and billing details.</li> <li>3. System sends request to server and communicate with Stripe API to verify the payment.</li> <li>4. System display payment is succeeded.</li> </ol>	
<p>Alternative Flow of Events:</p> <ol style="list-style-type: none"> <li>3.1 Payment failed             <ol style="list-style-type: none"> <li>3.1.1 Use case terminates</li> </ol> </li> </ol>	

Use case name	View fee
Actor	Community resident
Description	Community resident view specific fee details.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident select a fee to view.</li> <li>2. System retrieve the fee details.</li> <li>3. System display the fee details.</li> </ol>	

Use case name	View payment history
Actor	Community resident
Description	Community resident view specific payment history details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select a payment history to view.</li> <li>2. System retrieve the payment history details.</li> <li>3. System display the payment history details.</li> </ol>	

Use case name	View announcement
Actor	Community resident
Description	Community resident view specific announcement details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select an announcement to view.</li> <li>2. System retrieve the announcement details.</li> <li>3. System display the announcement details.</li> </ol>	

Use case name	Bookmark announcement
Actor	Community resident
Description	Community resident bookmark specific announcement details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select an announcement to view.</li> <li>2. System retrieve the announcement details.</li> <li>3. System display the announcement details.</li> <li>4. Community resident bookmark an announcement.</li> </ol>	



Use case name	Un-bookmark announcement
Actor	Community resident
Description	Community resident un-bookmark specific announcement details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select an announcement to view.</li> <li>2. System retrieve the announcement details.</li> <li>3. System display the announcement details.</li> <li>4. Community resident un-bookmark an announcement.</li> </ol>	

Use case name	Submit feedback/complaint
Actor	Community resident
Description	Community resident submit feedback/complaint.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident press the submit feedback/complaint button.</li> <li>2. Community resident fill in the information required such as type of feedback/complaint, details information, subject etc.</li> <li>3. System display feedback/complaint received message.</li> </ol>	

Use case name	View feedback/complaint progress and status
Actor	Community resident
Description	Community resident view a specific feedback/complaint progress and status.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select a feedback/complaint process to view progress and status of it.</li> <li>2. System retrieve the feedback/complaint process progress and status.</li> <li>3. System display the feedback/complaint process progress and status.</li> </ol>	

Use case name	Update feedback/complaint progress
Actor	Community resident
Description	Community resident update a specific feedback/complaint progress.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident select a feedback/complaint process to update progress and status of it.</li> <li>2. Community resident filled in the title and description of the update.</li> <li>3. Community resident submit the update.</li> <li>4. System update the feedback/complaint progress.</li> </ol>	

Use case name	Mark feedback/complaint process as completed
Actor	Community resident
Description	Community resident marks feedback/complaint process as completed.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident select a feedback/complaint process to update status of it.</li> <li>2. Community resident complete the feedback/complaint process.</li> <li>3. System display the process as completed.</li> </ol>	

Use case name	Rate feedback/complaint process
Actor	Community resident
Description	Community resident rate the feedback/complaint process after completed.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. After community resident complete a feedback/complaint process, system prompt user rating on that process.</li> <li>2. Community resident input the rate for the feedback/complaint process.</li> <li>3. System display rating received message.</li> </ol>	

Use case name	Create visitor application
Actor	Community resident
Description	Community resident register visitor application for visitor access.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident press the registration of visitor access button.</li> <li>2. Community resident fill in the personal information for the visitor(s) such as name, phone number etc.</li> <li>3. System display visitor application received message.</li> </ol>	

Use case name	View visitor application
Actor	Community resident
Description	Community resident view specific visitor application details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident select a visitor application to view.</li> <li>2. System retrieve the visitor application details.</li> <li>3. System display the visitor application details.</li> </ol>	

Use case name	View resident information
Actor	Community resident
Description	Community resident view their personal information.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident press profile button.</li> <li>2. System retrieved the personal information.</li> <li>3. System display the personal information.</li> </ol>	

Use case name	Update resident information
Actor	Community resident
Description	Community resident update their personal information.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident press profile button.</li> <li>2. System retrieved the personal information.</li> <li>3. System display the personal information.</li> <li>4. Community resident press edit button.</li> <li>5. Community resident edit the personal information.</li> <li>6. Community resident press save button.</li> <li>7. System update the personal information</li> </ol>	

Use case name	Log in
Actor	Community resident
Description	Community resident log in to the system.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident filled in email/username and password to log in to the system.</li> <li>2. System verify the resident authentication.</li> <li>3. Community resident log in to the system successfully.</li> </ol>	
Alternative Flow of Events:	
<ol style="list-style-type: none"> <li>1.1 Wrong email/username or password. <ol style="list-style-type: none"> <li>1.1.1 Community resident authentication failed</li> <li>1.1.2 Use case terminated.</li> </ol> </li> </ol>	

Use case name	Log out
Actor	Community resident
Description	Community resident log out from the system.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community resident press the log out button.</li> <li>2. Community resident log out from the system successfully.</li> </ol>	

Use case name	Change password
Actor	Community resident
Description	Community resident change password their account password.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident press the profile button.</li> <li>2. Community resident press the change password button.</li> <li>3. Community resident enter the old password.</li> <li>4. Community resident enter the new password.</li> <li>5. Community resident re-enter the new password.</li> <li>6. Community resident press change button.</li> </ol>	

Use case name	Recover password
Actor	Community resident
Description	Community resident recover their account forgotten password.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community resident press the forgot password button at login screen.</li> <li>2. Community resident enter the email of their account.</li> <li>3. System will send a recovery password email to the email account.</li> <li>4. Community resident press the recovery link in the recovery password email.</li> <li>5. Community resident will redirect to a web page to reset their password.</li> </ol>	

#### 4.4.2.2 Web-based application for Management Users

Use case name	Manage fee
Actor	Community management
Description	Community management create new fee, edit fee details and delete specific fee.
<p>Flow of Events</p> <ul style="list-style-type: none"> <li>• Create new fee <ol style="list-style-type: none"> <li>1. Community management press the new fee button.</li> <li>2. Community management fill in the fee details such as, fee title, fee amount, fee deadlines etc.</li> <li>3. System create a new fee.</li> <li>4. System send notification to residents' devices.</li> </ol> </li> <li>• View fee details <ol style="list-style-type: none"> <li>1. Community management select the fee to view details.</li> <li>2. System retrieved the specific fee record.</li> <li>3. System display the specific fee record.</li> </ol> </li> <li>• Edit fee details <ol style="list-style-type: none"> <li>1. Community management select the fee to edit.</li> <li>2. Community management fill in the fee details such as, fee title, fee amount, fee deadlines etc.</li> <li>3. System will update the specific fee.</li> </ol> </li> <li>• Delete specific fee <ol style="list-style-type: none"> <li>1. Community management select the fee to delete.</li> <li>2. System double confirm the deletion with community management.</li> <li>3. System will delete the specific fee.</li> </ol> </li> </ul>	

Use case name	Manage Announcement
Actor	Community management
Description	Community management create new announcement to broadcast important message, edit the announcement and delete the announcement.
<p>Flow of Events</p> <ul style="list-style-type: none"> <li>• Create new announcement <ol style="list-style-type: none"> <li>1. Community management press the new announcement button.</li> <li>2. Community management fill in the announcement subject, announcement content etc.</li> </ol> </li> </ul>	

<ol style="list-style-type: none"> <li>3. System will add the announcement into announcement list.</li> <li>4. System send notification to residents' devices.</li> </ol> <ul style="list-style-type: none"> <li>• View announcement details <ol style="list-style-type: none"> <li>1. Community management select the announcement to view details.</li> <li>2. System retrieved the specific announcement record.</li> <li>3. System display the specific announcement record.</li> </ol> </li> <li>• Edit announcement details <ol style="list-style-type: none"> <li>1. Community management select the announcement to edit.</li> <li>2. Community management fill in the announcement subject, announcement content etc. to edit.</li> <li>3. System will update the specific announcement.</li> </ol> </li> <li>• Delete announcement <ol style="list-style-type: none"> <li>1. Community management select the announcement to delete.</li> <li>2. System will double confirm the deletion with community management.</li> <li>3. System will delete the specific fee.</li> </ol> </li> </ul>
---

Use case name	View feedback/complaint progress and status
Actor	Community management
Description	Community management view a specific feedback/complaint progress and status.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management select a feedback/complaint process to view progress and status of it.</li> <li>2. System retrieve the feedback/complaint process progress and status.</li> <li>3. System display the feedback/complaint process progress and status.</li> </ol>	

Use case name	Update feedback/complaint status
Actor	Community management
Description	Community management update a specific feedback/complaint progress and status.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management select the feedback/complaint process status wish to update.</li> </ol>	

2. Community management fill in the required information such as comment, solution etc.
3. System will update the feedback/complaint progress and status.

Use case name	View visitor application
Actor	Community management
Description	Community management view specific visitor application details.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management select a visitor application to view.</li> <li>2. System retrieve the visitor application details.</li> <li>3. System display the visitor application details.</li> </ol>	

Use case name	Approve/Reject visitor application
Actor	Community management
Description	Community management approve/reject the visitor application.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management select the specific visitor registration and view its information.</li> <li>2. Community management approve/reject the visitor registration.</li> </ol>	

Use case name	Log in
Actor	Community management
Description	Community management log in to the system.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management filled in email/username and password to log in to the system.</li> <li>2. System verify the user authentication.</li> <li>3. Community management log in to the system successfully.</li> </ol>	
Alternative Flow of Events:	
<ol style="list-style-type: none"> <li>1.1 Wrong email/username or password. <ol style="list-style-type: none"> <li>1.1.1 Community management authentication failed</li> <li>1.1.2 Use case terminated.</li> </ol> </li> </ol>	



Use case name	Log out
Actor	Community management
Description	Community management log out from the system.
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management press the log out button.</li> <li>2. Community management log out from the system successfully.</li> </ol>	

Use case name	Create community resident account
Actor	Community management
Description	Community management create new community resident account
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management press create new community resident account button.</li> <li>2. An account based on the unit number and a random password will be generated.</li> </ol>	

Use case name	Create security personnel account
Actor	Community management
Description	Community management create new security personnel account
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management press create new security personnel account button.</li> <li>2. A security personnel account will be generated.</li> </ol>	

Use case name	Suspend account
Actor	Community management
Description	Community management suspend account
Flow of Events	
<ol style="list-style-type: none"> <li>1. Community management select specific account.</li> <li>2. Community management press to suspend the account.</li> <li>3. System will double confirm with the community management about the suspension of account.</li> </ol>	

4. System will suspend the specific account.

Use case name	Change password
Actor	Community management
Description	Community management change password their account password.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community management press the profile button.</li> <li>2. Community management press the change password button.</li> <li>3. Community management enter the old password.</li> <li>4. Community management enter the new password.</li> <li>5. Community management re-enter the new password.</li> <li>6. Community management press change button.</li> </ol>	

Use case name	Recover password
Actor	Community management
Description	Community management recover their account forgotten password.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Community management press the forgot password button at login screen.</li> <li>2. Community management enter the email of their account.</li> <li>3. System will management a recovery password email to the email account.</li> <li>4. Community management press the recovery link in the recovery password email.</li> <li>5. Community management will redirect to a web page to reset their password.</li> </ol>	

#### 4.4.2.3 Mobile Application for Security Personnel

Use case name	Scan QR code
Actor	Security Personnel
Description	Security personnel scan QR code to grant access to visitor.
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Security personnel shall be able to scan QR code that present by visitor.</li> <li>2. System retrieve the specific visitor application that the QR code belongs.</li> <li>3. System display the visitor application details.</li> <li>4. Security personnel grant access to visitor.</li> <li>5. System will update the specific visitor application that the QR code belongs for the status and entered time of visitor.</li> </ol>	
<p>Alternative Flow of Events:</p> <ol style="list-style-type: none"> <li>2.1 Visitor application not found.             <ol style="list-style-type: none"> <li>2.1.1 Use case terminates</li> </ol> </li> </ol>	

Use case name	Add anonymous visitor
Actor	Security Personnel
Description	Security personnel add anonymous visitor with no QR code such as delivery personnel
<p>Flow of Events</p> <ol style="list-style-type: none"> <li>1. Security personnel press the add new visitor button.</li> <li>2. Security personnel fill in the visitor information.</li> <li>3. Security personnel press the add button and grant access to the visitor.</li> </ol>	

## CHAPTER 5

### SYSTEM DESIGN

#### 5.1 Introduction

This chapter will discuss about the system architecture, software design pattern applied, database design. Besides, user interface design of both clients which are web application and mobile application will be visualized.

#### 5.2 System Architecture

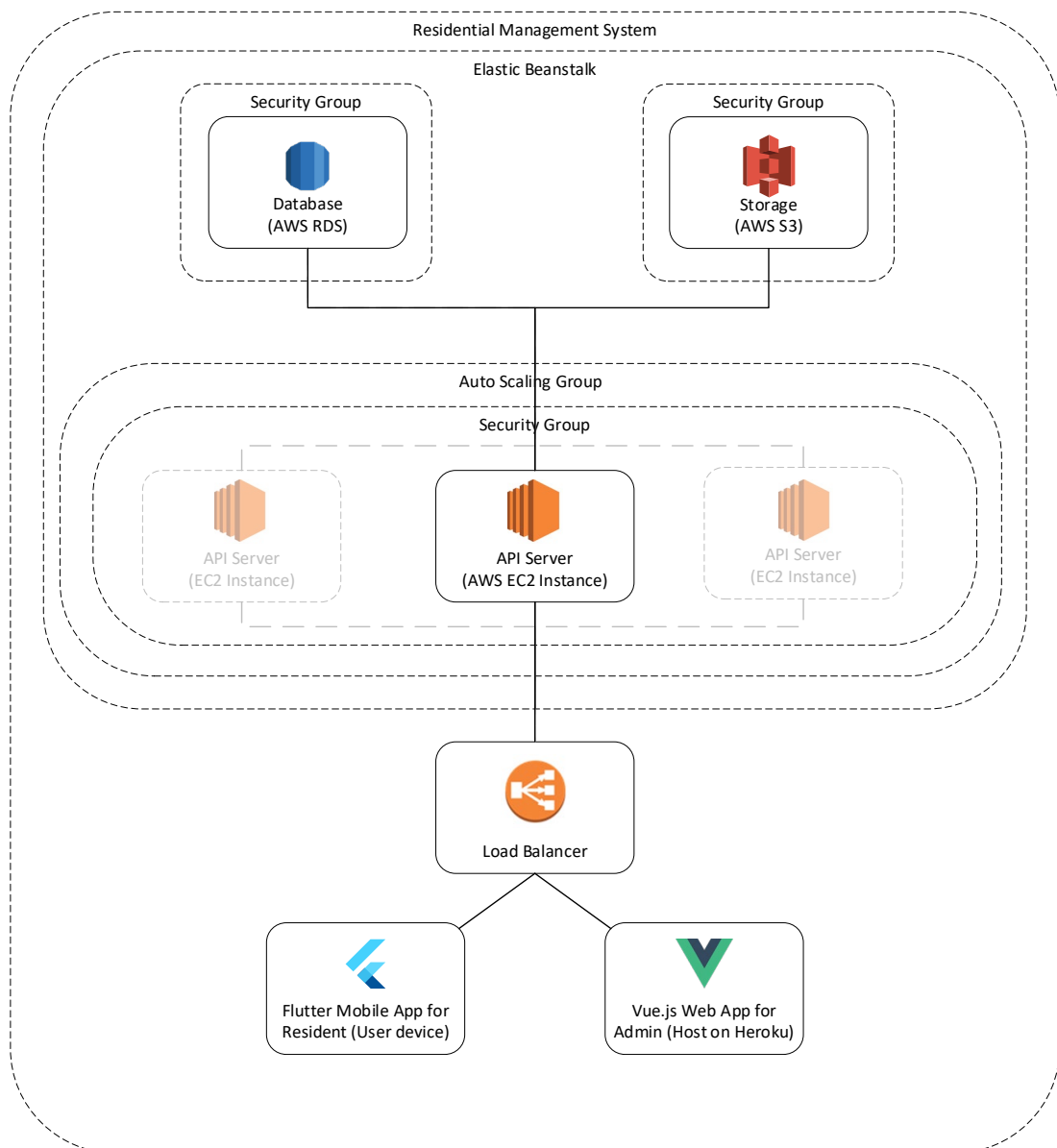


Figure 5.1 Overview of System Architecture

Residential management system in this project involved multiple module and layers of service. The figure above shows the overview of the system architecture that including one back-end server and two front-end clients.

AWS Elastic Beanstalk is an Amazon Web Services orchestration framework for deploying applications that orchestrates various AWS services including EC2, S3, Simple Notification Framework (SNS), CloudWatch, Autoscaling and Elastic Load Balancers. As AWS Elastic Beanstalk provide a centralized or full-stacked panel to hosting an application, it is chosen to serve as a web hosting tools in this project. Elastic Beanstalk provides an additional abstraction layer over the bare server and OS by using the pre-built combination of OS platform. It helps developer to delegate the effort to deploy and increase the productivity by enabling developer to focus of the product development.

Auto Scaling group is included in the AWS Elastic Beanstalk environment to handle the auto configuration of capacity based on the triggering of metric. The Auto Scaling group will ensure that there is always one instance running to achieve high availability. In corporate with load balancer, the Auto Scaling group will add or deletes instance depends on the load of instances and the configuration of maximum number of instances defined. In another words, the Auto Scaling group will ensure your server is available anytime using the minimum resource effectively and dynamically.

Security is another important part of the project as it involved many sensitive personal information such as resident address, name, phone number etc. Preventive action is needed to assure the safety of the data from leakage. In AWS Elastic Beanstalk environment, the Security Group is included to acts as a virtual firewall for the API server to control all the inbound and outbound traffic and request. Rules and regulation can apply to those security group and configure the security layer. The security group will protect the MySQL database instance, AWS S3 bucket and API server from the unauthorized access and service attacks.

The aim of this system architecture approach is ensuring high availability, high security, better system management and consolidation of each modules.

### 5.3 Software Design Pattern

The Model-View-Controller (MVC) design pattern is applied in this project. MVC design pattern conceptually divides an application into three main logical interconnected components, which is model, view and controller. Each of these components is designed to deal with specific aspects of an application's development. MVC is one of the industry-standard Web development system most commonly used to build scalable and configurable projects. Besides the beneficial in software architecture, MVC also contribute to better organizing and understanding of the business logic of the software. It helps to transform the business logic into code segments in the software.

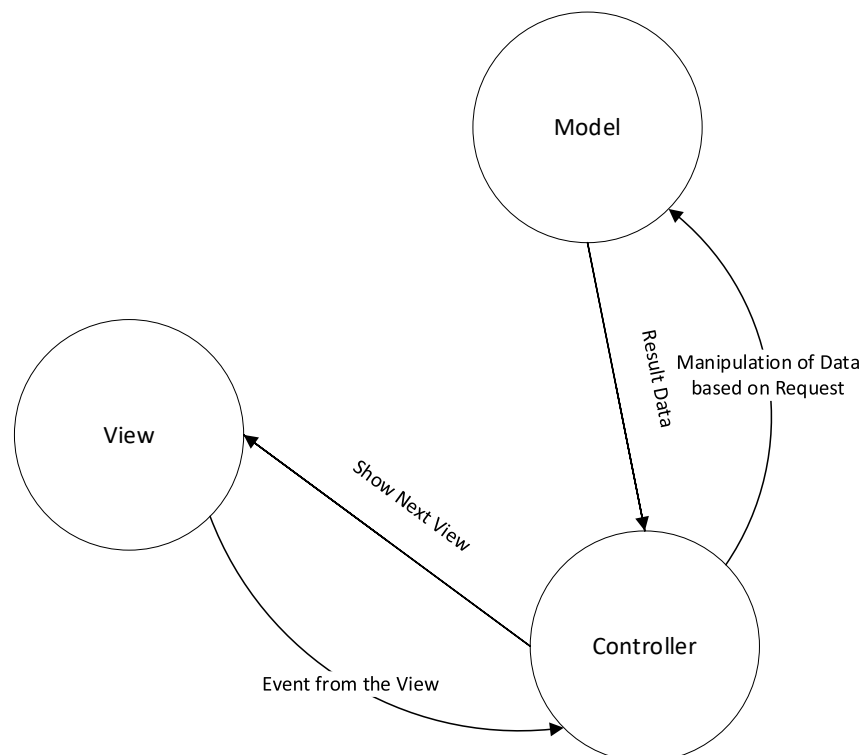


Figure 5.2 MVC Flow Visualization

The figure above briefly visualizes the flow of the MVC pattern. The Model act as a central component to store the application data and data-oriented logic in which the user operates. The View is used to present the formatted model's data in user interface view that user could understand. The Controller exists between the Model and the View to serve as an agent to manage the communication between the two parties. Controller will process the input event from the View and manipulate the Model's data based on the request.

Apart from that, MVC plays a great role in the development of object-oriented software as it facilitates rapid and parallel development and allows creation and maintenance simpler. Application of MVC in this project will bring additional advantages along with the implementation of object-oriented approach. Additionally, this helps in creating business logic for the development of specific object-oriented applications. In addition, developers may rely on design patterns for MVC that are commonly accepted as solutions to recurring problems and are used to build scalable, reusable and modular applications (Thakur and Pandey, 2019). It provides multiple views without any problems, due to its faster development process. Because MVC has many advantages in the development of object-oriented applications, it is also used in the development of interactive web and iOS.

For example, in this project, announcement related data will be store in announcement model. An announcement controller will be created to handle the request from the two clients' view, mobile application and web application and perform retrieve or update operation on the announcement model. This approach helps to consolidate each function from each other and lead to better readability and control in function development.

## 5.4 Database Design

This section will describe the organization of database model and the relationship between each other's. Several diagrams are used to visualize the relationship between entities and how the data be classified such as physical entity relationship diagram and logical entity relationship diagram. In additions, data dictionary is created to explain each table and the attributes.

### 5.4.1 Physical Entity Relationship Diagram

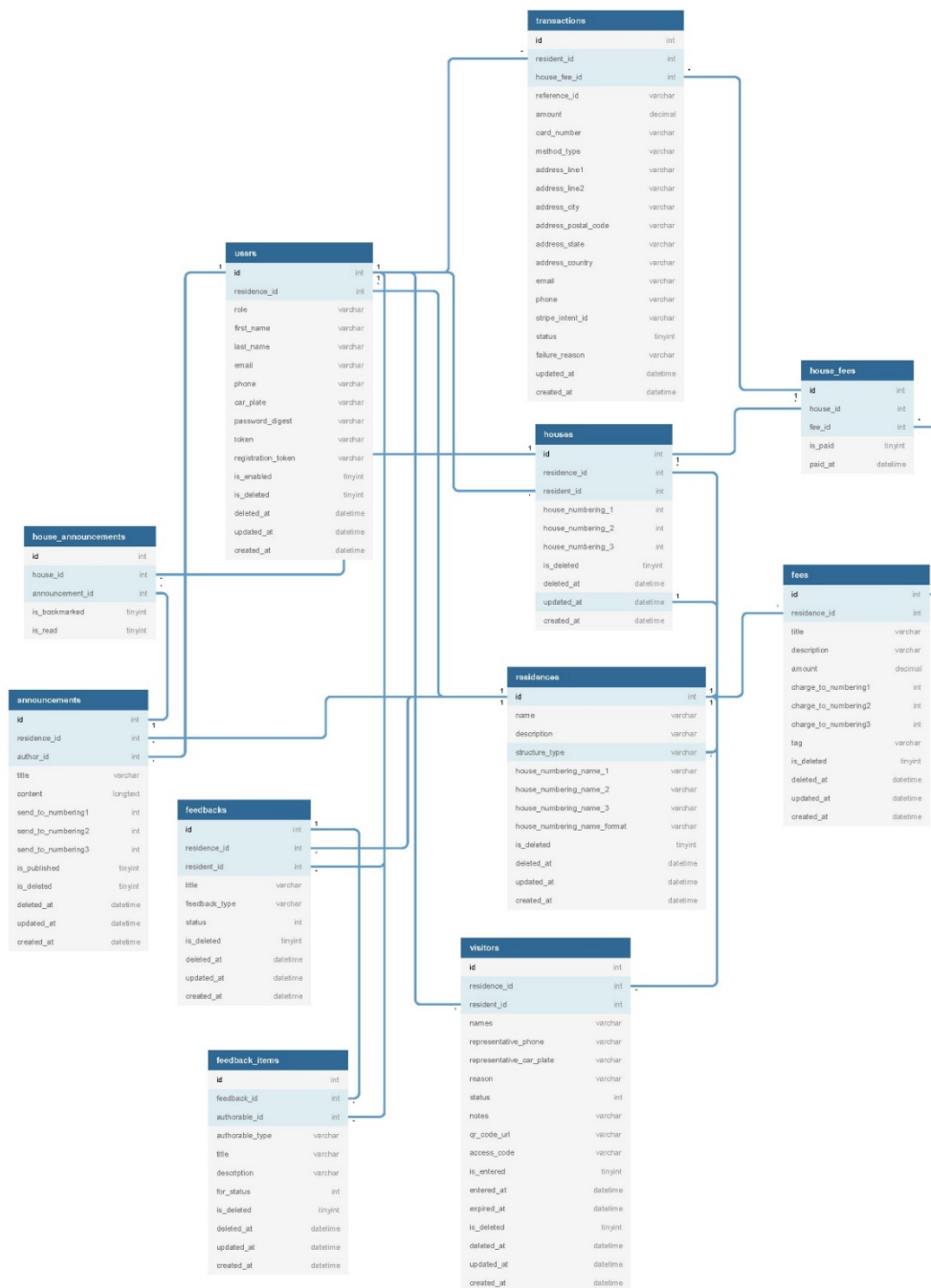


Figure 5.3 Completed physical entity relationship diagram



## 5.4.2 Logical Entity Relationship Diagram

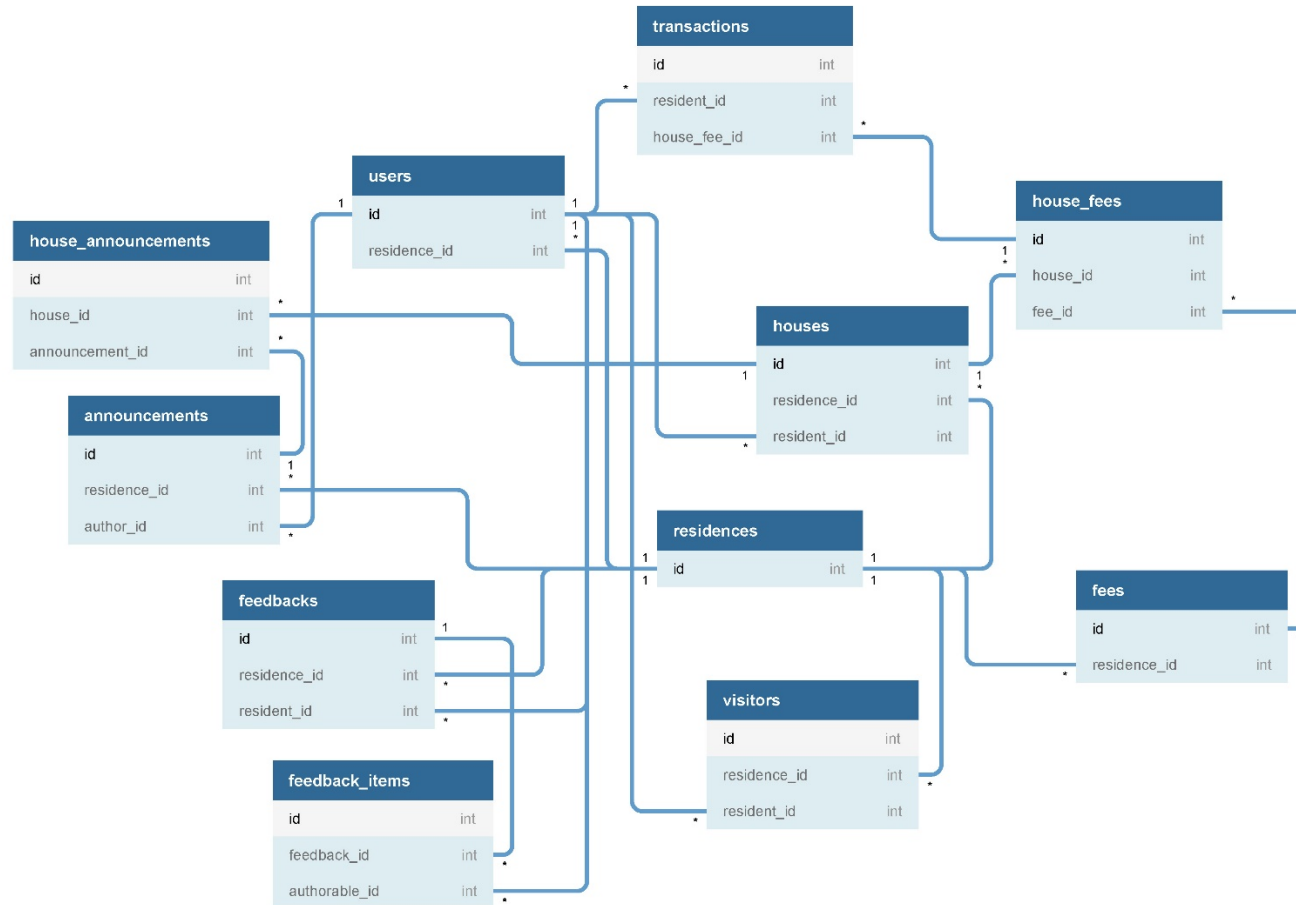


Figure 5.4 Simplified Logical entity relationship diagram

### 5.4.3 Data Dictionary

Data dictionary act as the centralized documents that describe the collections of database and attributes inside each table. There are 11 tables in the database instance created.

#### 5.4.3.1 Residences

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for residence	INT	1
<b>name</b>	Display name of residence	VARCHAR	Cypress
<b>description</b>	Brief description of residence	VARCHAR	Welcome to Cypress
<b>structure_type</b>	Residential building type	VARCHAR	terrace
<b>house_numbering_name_1</b>	House numbering name level 1	VARCHAR	Unit
<b>house_numbering_name_2</b>	House numbering name level 2	VARCHAR	Floor
<b>house_numbering_name_3</b>	House numbering name level 3	VARCHAR	Block
<b>house_numbering_name_format</b>	House numbering name format	VARCHAR	B{{numbering_1}}- F{{numbering_2}}- {{numbering_3}}

<b>is_deleted</b>	Is the residence deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the residence had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the residence last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the residence is created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.2 Houses

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for residence	INT	1
<b>[FK] residence_id</b>	ID of residence that the houses belongs to	INT	1
<b>[FK] resident_id</b>	ID of resident that the houses belongs to	INT	1
<b>house_numbering_1</b>	Number of house numbering level 1	VARCHAR	1
<b>house_numbering_2</b>	Number of house numbering level 2	VARCHAR	1
<b>house_numbering_3</b>	Number of house numbering level 3	VARCHAR	1

<b>is_deleted</b>	Is the house deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the house had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the house last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the house created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.3 Users

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for user	INT	1
<b>[FK] residence_id</b>	ID of residence that the user belongs to	INT	1
<b>role</b>	The role of user	VARCHAR	resident
<b>first_name</b>	First name of user	VARCHAR	Ming
<b>last_name</b>	Last name of user	VARCHAR	Tan
<b>email</b>	Email address of user, can be used for login	VARCHAR	user@neibor.com

<b>phone</b>	Contact number of users	VARCHAR	0187363552
<b>car_plate</b>	Car plate number of users	VARCHAR	NJH 0377
<b>password_digest</b>	The encrypted password of user	VARCHAR	\$2a\$05\$18vboB/F9ML...
<b>token</b>	The Neibor Access Token of user	VARCHAR	eyJhbGciOiJIUzI1NiJ9....
<b>registration_token</b>	The registration token for the user devices, stored for the purpose of notification	VARCHAR	eMYgQobrRBKpiAlUxmo...
<b>is_enabled</b>	Is the user enabled?	TINYINT	1
<b>is_deleted</b>	Is the user deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the user had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the user last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the user created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.4 Fees

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for fee	INT	1
<b>[FK] residence_id</b>	ID of residence that the fee belongs to	INT	1
<b>title</b>	The title of fee, should be intuitive title	VARCHAR	Annual
<b>description</b>	The description of fee, can be used to describe the purpose of fee	VARCHAR	Use to maintain the community operation and maintenance of facilities
<b>amount</b>	The charging amount of fee	DECIMAL	100.00
<b>charge_to_numbering_1</b>	The level 1 target area of houses to be charged	INT	1
<b>charge_to_numbering_2</b>	The level 1 target area of houses to be charged	INT	1
<b>charge_to_numbering_3</b>	The level 1 target area of houses to be charged	INT	1
<b>tag</b>	Tags that the fee can be classified	VARCHAR	water
<b>due_at</b>	The date and time when the fee is expired	DATETIME	2020-02-15 10:11:13
<b>is_deleted</b>	Is the fee deleted?	TINYINT	0

<b>deleted_at</b>	The date and time when the fee had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the fee last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the fee created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.5 House Fee Pivot

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for house fee	INT	1
<b>[FK] house_id</b>	ID of house that has the fee	INT	1
<b>[FK] fee_id</b>	ID of fee that has charge to the house	INT	1
<b>is_paid</b>	Is the fee paid?	TINYINT	0
<b>paid_at</b>	The date and time when the fee had been paid	DATETIME	2020-02-15 10:11:13

### 5.4.3.6 Transaction

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for transaction	INT	1
<b>[FK] residence_id</b>	ID of residence that the transaction belongs to	INT	1
<b>house_fee_ids</b>	The list of house fee ID	VARCHAR	1,2
<b>reference_id</b>	Unique reference ID for Neibor internal reference	VARCHAR	#111-00001
<b>amount</b>	The total charging amount of all the fees	DECIMAL	100.00
<b>card_number</b>	Last four digits of the whole card number	VARCHAR	0002
<b>method_type</b>	Payment method of the transaction	VARCHAR	card
<b>address_line1</b>	Address line 1 of billing address	VARCHAR	C1-F1-1, Cypress
<b>address_line2</b>	Address line 2 of billing address	VARCHAR	Bandar Sungai Long
<b>address_city</b>	City of billing address	VARCHAR	Kajang
<b>address_postal_code</b>	Postal code of billing address	VARCHAR	43000
<b>address_state</b>	State of billing address	VARCHAR	Selangor



<b>address_country</b>	Country of billing address	VARCHAR	MY
<b>email</b>	Email of billing details	VARCHAR	user@neibor.com
<b>phone</b>	Phone of billing details	VARCHAR	0188262672
<b>stripe_intent_id</b>	ID of Stripe intent created	VARCHAR	pi_1GMnrtJagR7ayQ...
<b>status</b>	Indicated the transaction is successful or not	TINYINT	1
<b>failure_reason</b>	State the reason of transaction failure	VARCHAR	Invalid card number
<b>created_at</b>	The date and time when the transaction created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.7 Announcements

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for announcement	INT	1
<b>[FK] residence_id</b>	ID of residence that the announcement belongs to	INT	1
<b>[FK] author_id</b>	ID of user that create the announcement	INT	1
<b>title</b>	Title of the announcement	VARCHAR	TEMPORARY WATER OUTAGE

<b>content</b>	HTML code that use to render the content of announcement	LONGTEXT	<p>Good Afternoon Westwood Suites ...
<b>send_to_numbering_1</b>	The level 1 target area of houses will receive this announcement	INT	1
<b>send_to_numbering_2</b>	The level 2 target area of houses will receive this announcement	INT	1
<b>send_to_numbering_3</b>	The level 3 target area of houses will receive this announcement	INT	1
<b>is_published</b>	Is the announcement publish?	TINYINT	1
<b>is_deleted</b>	Is the announcement deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the announcement had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the announcement last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the announcement created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.8 House Announcement Pivot

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for house announcement	INT	1
<b>[FK] house_id</b>	ID of house that will receive the announcement	INT	1
<b>[FK] announcement_id</b>	ID of announcement that send to the house	INT	1
<b>is_bookmarked</b>	Is the announcement bookmarked by the resident?	TINYINT	0
<b>is_read</b>	Is the announcement read by the resident?	TINYINT	0

#### 5.4.3.9 Feedbacks

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for feedback	INT	1
<b>[FK] residence_id</b>	ID of residence that the feedback belongs to	INT	1
<b>[FK] resident_id</b>	ID of resident that create the feedback	INT	1
<b>title</b>	The title of the feedback	VARCHAR	Lift out of order

<b>feedback_type</b>	The type of the feedback	VARCHAR	facilities
<b>status</b>	The status of the feedback, can be received, in progress and completed	INT	1
<b>rating</b>	The rating of the feedback rate by resident when a feedback is completed, 1 to 5	INT	5
<b>review</b>	The review of the feedback given by resident when a feedback is completed	LONGTEXT	Good job
<b>is_deleted</b>	Is the feedback deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the feedback had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the feedback last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the feedback created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.10 Feedback Items

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for feedback item	INT	1
<b>[FK] feedback_id</b>	ID of feedback that the feedback item belongs to	INT	1

<b>[FK] authorable_id</b>	ID of user that create the feedback item	INT	1
<b>authorable_type</b>	The user type of the author creates the feedback item	VARCHAR	resident
<b>title</b>	The title of the feedback item	VARCHAR	Feedback received
<b>description</b>	The status of the feedback, can be received, in progress and completed	VARCHAR	Action taken
<b>for_status</b>	Indicate the feedback item is for which status	INT	1
<b>is_deleted</b>	Is the feedback item deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the feedback item had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the feedback item last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the feedback item created	DATETIME	2020-02-15 10:11:13

#### 5.4.3.11 Visitors

Attribute	Description	Data type	Example value
<b>[PK] id</b>	Unique identifier for visitor application	INT	1
<b>[FK] residence_id</b>	ID of residence that the visitor application belongs to	INT	1
<b>[FK] resident_id</b>	ID of resident that create the visitor application	INT	1
<b>names</b>	List of names of visitors	VARCHAR	Ming Tan, Ali Mhd.
<b>representative_phone</b>	The phone number of the representative person	VARCHAR	0187363552
<b>representative_car_plate</b>	The car plate number of the representative person	VARCHAR	NJH 0377
<b>visitation_purpose</b>	The visit purpose of the visitor application	VARCHAR	Assignment

<b>status</b>	Indicate the visitor application is pending, approved or rejected	INT	1
<b>notes</b>	The notes to record the reject reason from community management	VARCHAR	Please provide phone number
<b>qr_code_url</b>	The url that link to the QR code image (QR code generated when visitor application approved)	VARCHAR	<a href="https://bt-neibor.s3.ap-southeast-1.amazonaws.com/visitor-qr/3">https://bt-neibor.s3.ap-southeast-1.amazonaws.com/visitor-qr/3</a>
<b>access_code</b>	The special access code to verify the visitor QR code	VARCHAR	857b7ecb3a6ca7783724
<b>is_entered</b>	Indicate the visitors had entered into the residential area	TINYINT	1
<b>entered_at</b>	The date and time when the visitors entered into the residential area	DATETIME	2020-02-15 10:11:13
<b>is_exited</b>	Indicate the visitors had exited into the residential area	TINYINT	0
<b>exited_at</b>	The date and time when the visitors exited into the residential area	DATETIME	2020-02-15 10:11:13

<b>expired_at</b>	The date and time when the QR code generated expired	DATETIME	2020-02-15 10:11:13
<b>is_deleted</b>	Is the visitor application deleted?	TINYINT	0
<b>deleted_at</b>	The date and time when the visitor application had been deleted	DATETIME	2020-02-15 10:11:13
<b>updated_at</b>	The date and time when the visitor application last updated	DATETIME	2020-02-15 10:11:13
<b>created_at</b>	The date and time when the visitor application created	DATETIME	2020-02-15 10:11:13



## 5.5 User Interface Design

User interface design is done by screen prototyping to produce a blueprint for the development phase. The user interface is designed and aim to achieve the objectives of this project such as complete a payment process by using shorter time compare to manual payment. There are several user interface design principles followed in this project:

1. Consistent user interface
2. Interfaces exists according to its purpose
3. User in control
4. No more than one primary action per section
5. Consider user expectation
6. Recognition rather than recall
7. Tolerance to user
8. Error prevention

The design activities of user interfaces are occurred in 3 modules which is the web application for community management, mobile application for community resident, and mobile application for security personnel.

### 5.5.1 Web Application for Community Management

The end user of this web application is community management, which consist of role admin and staff. The figure below is the login page of the web application for community management.

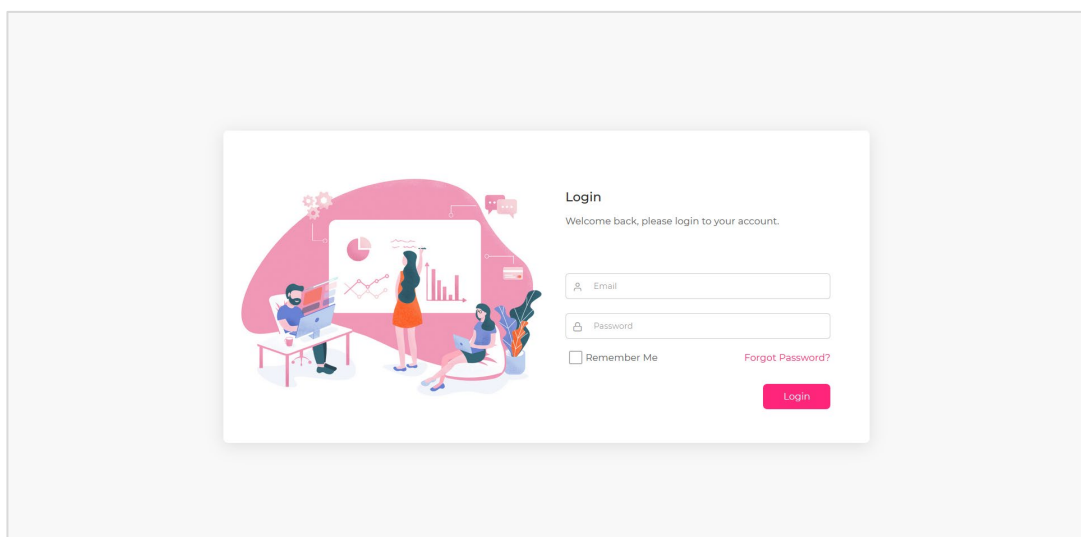


Figure 5.5 Login Page of Web Application

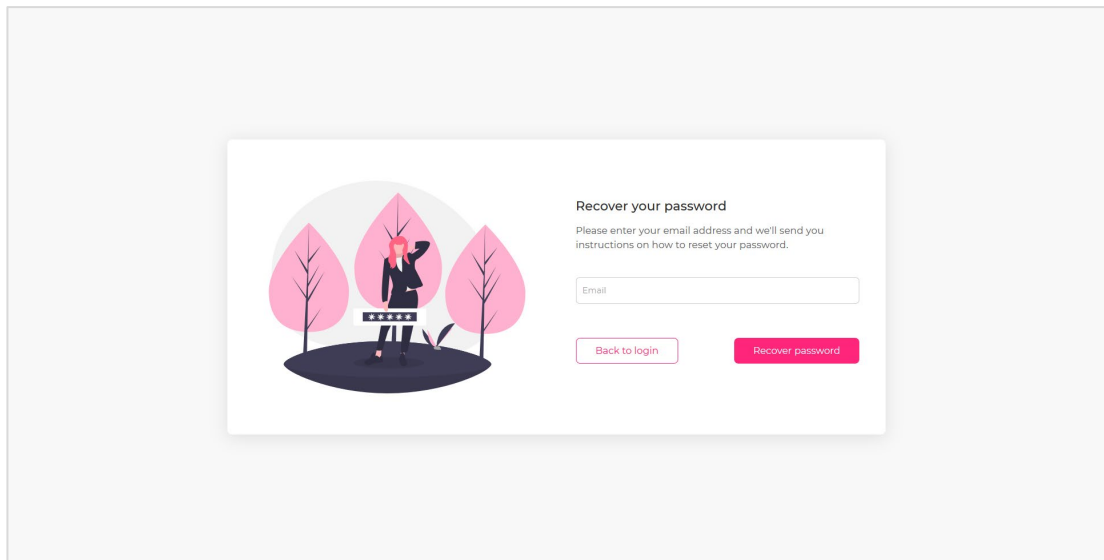


Figure 5.6 Recover Password Page of Web Application

The figure above shows the recover password screen of web application. User can enter the account's email to receive password recovery email and proceed to reset their forgotten password.

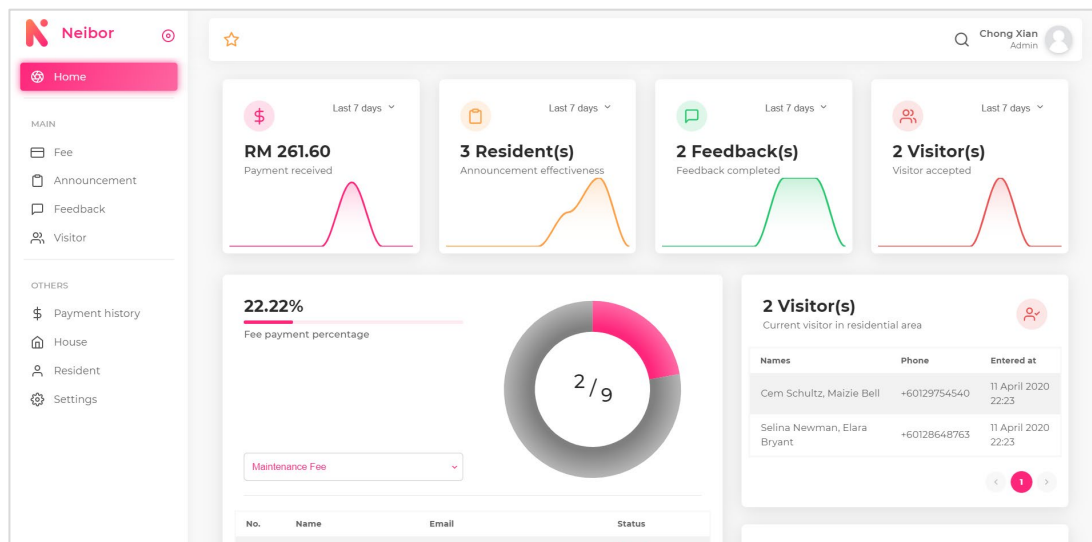
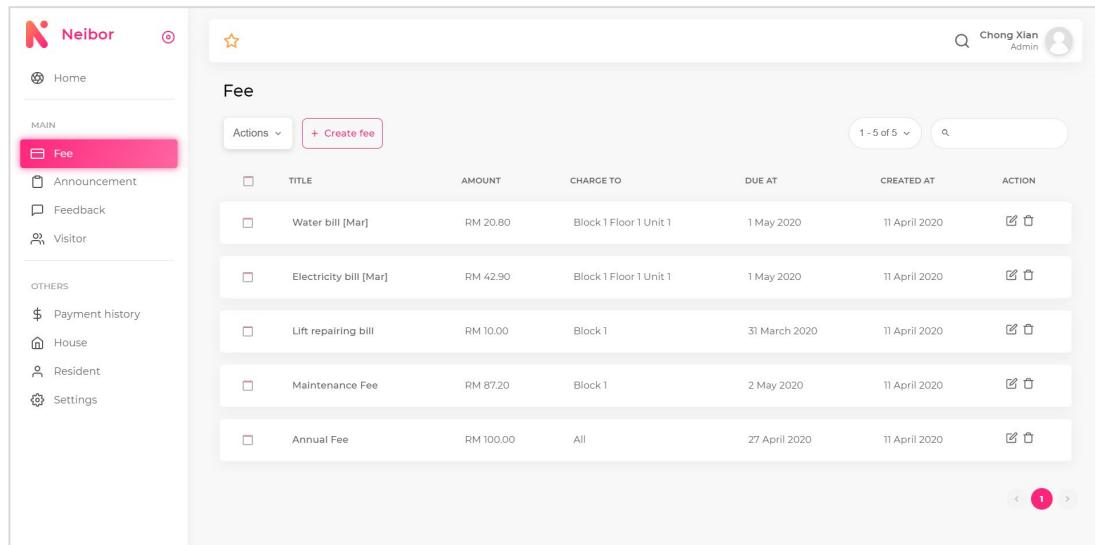


Figure 5.7 Home Analytics Page of Web Application

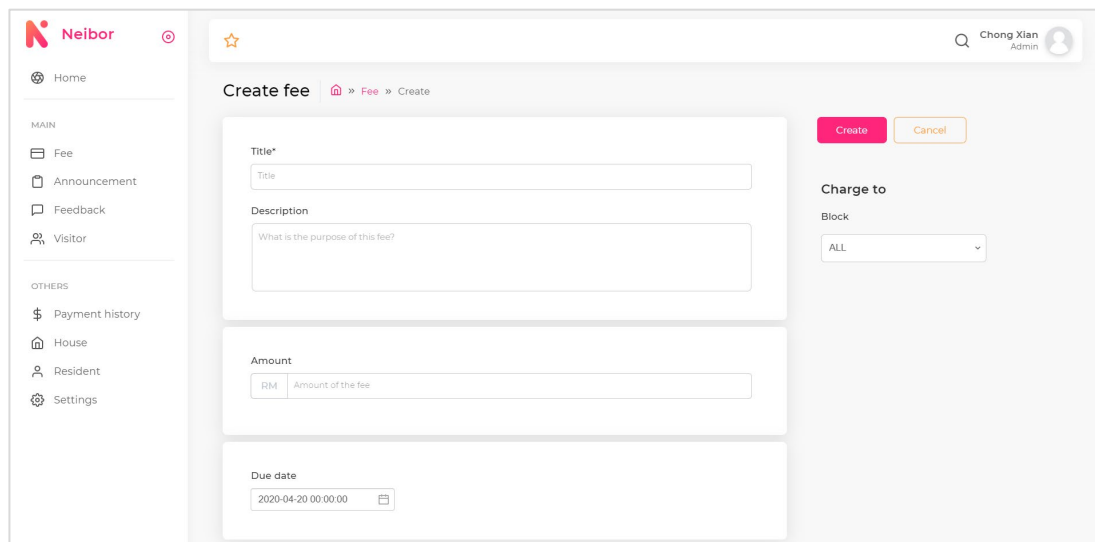
The figure above shown several analytics for the residences such as total payment amount received within a certain period, announcement effectiveness, feedback completion rate, number of visitors accepted, etc. Besides, percentage of fee payment completed by residents for a specific fee had been analysed and displayed. In addition, there are one card showing the total amount of visitor still remained inside the residential area.



TITLE	AMOUNT	CHARGE TO	DUE AT	CREATED AT	ACTION
Water bill [Mar]	RM 20.80	Block 1 Floor 1 Unit 1	1 May 2020	11 April 2020	
Electricity bill [Mar]	RM 42.90	Block 1 Floor 1 Unit 1	1 May 2020	11 April 2020	
Lift repairing bill	RM 10.00	Block 1	31 March 2020	11 April 2020	
Maintenance Fee	RM 87.20	Block 1	2 May 2020	11 April 2020	
Annual Fee	RM 100.00	All	27 April 2020	11 April 2020	

Figure 5.8 Fee list in table

The figure above shows the fees in table form. Community management user can create new fee, select existing fee to edit and delete specific fee in this page.



**Create fee** [Home](#) » [Fee](#) » Create

**Title\***

**Description**  
 What is the purpose of this fee?

**Amount**  
 RM  Amount of the fee

**Due date**

**Charge to**  
 Block:

Figure 5.9 Create new fee page

Community management user can fill in the fee details including title, description, amount, due date and target houses in this page to create new fee.

Figure 5.10 Edit fee page

Community management user can edit in the specific fee details including title, description, amount, due date and target houses in this page to update the fee.

TITLE	SEND TO	VISIBILITY	CREATED BY	CREATED AT	ACTION
Planned Power Outage Notice	All	Published	Chong Xian Goh	11 April 2020	✎ 🗑
Temporary Water Supply Disruption	All	Published	Chong Xian Goh	11 April 2020	✎ 🗑
Annual Fee Draft	All	Unpublished	Chong Xian Goh	11 April 2020	✎ 🗑

Figure 5.11 Announcement list in table

The figure above shows the announcements created in table form. Community management user can create new announcement, select existing announcement to edit and delete specific announcement in this page.

The screenshot shows the 'Create announcement' page. On the left is a sidebar with navigation options: Home, Fee, Announcement, Feedback, Visitor, Payment history, House, Resident, and Settings. The main content area has a breadcrumb 'Announcement » Create'. The form includes a 'Title\*' field, a 'Content\*' field with a rich text editor, and a 'Visibility' section with radio buttons for 'Publish' and 'Unpublish'. Below that is a 'Send to' dropdown menu set to 'ALL' and a 'Tags' section with '0/3' tags and an 'Add' button. At the bottom, the author is identified as 'Chong Xian Goh'. Buttons for 'Create' and 'Cancel' are located at the top right of the form area.

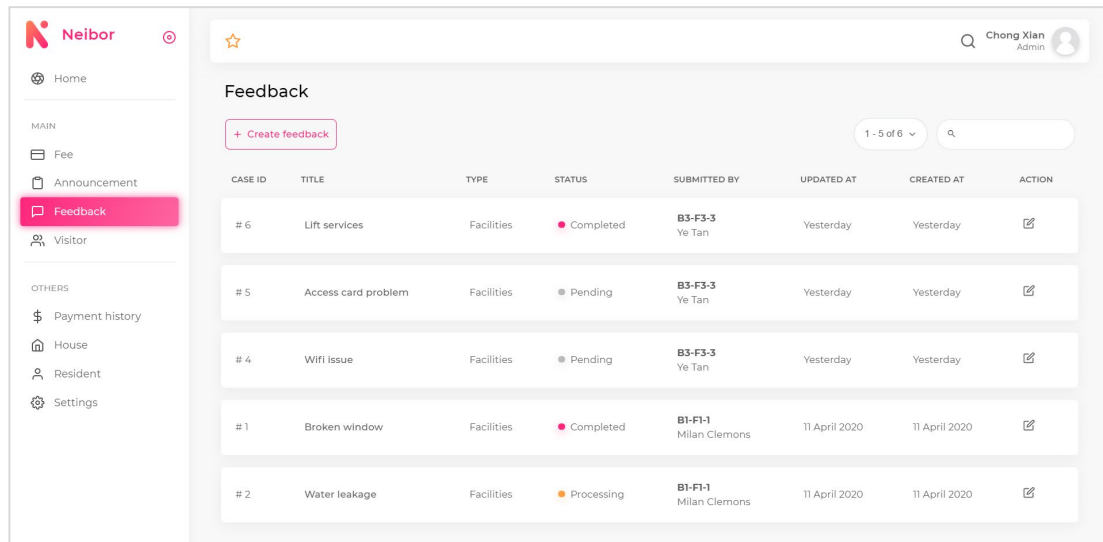
Figure 5.12 Create new announcement page

In this page, community management user can create new announcement by entering the announcement title, its content using the text editor, flag its visibility, its target residents and the tags of announcements.

The screenshot shows the 'Edit announcement' page. The breadcrumb is 'Announcement » Edit'. The form fields are populated: 'Title\*' is 'Planned Power Outage Notice'. The 'Content\*' field contains a detailed notice about a power outage, including dates (Monday April 27th, 2020 and Tuesday April 28th, 2020), reasons for the outage, and instructions for residents. The 'Visibility' section has 'Publish' selected. The 'Send to' dropdown is 'ALL'. The 'Tags' section shows 'Electricity' and 'Power' as existing tags. The author is 'Chong Xian Goh'. Buttons for 'Save' and 'Cancel' are at the top right.

Figure 5.13 Edit announcement page

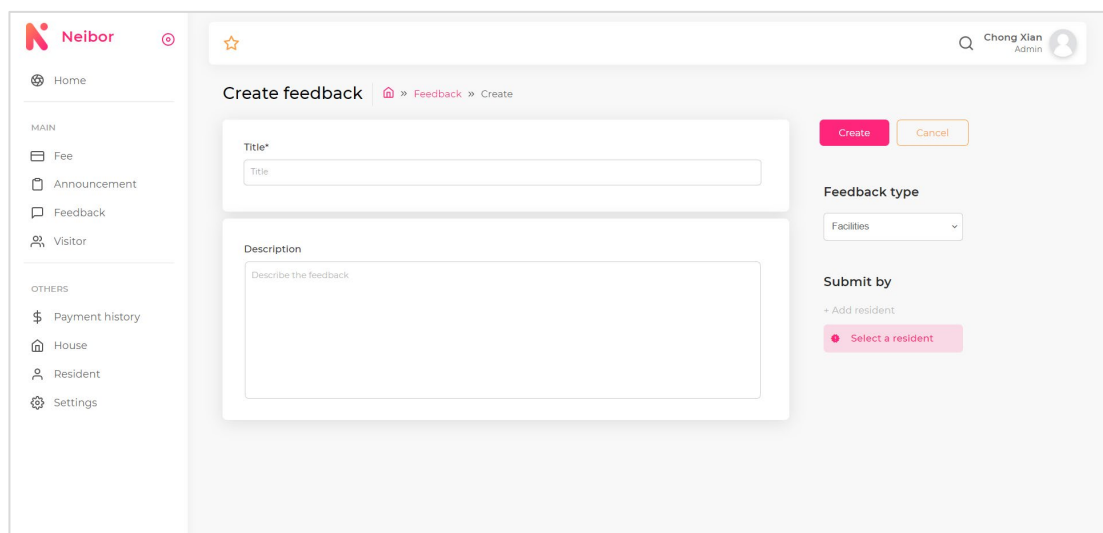
The figure above shown the edit announcement page, community management user can edit specific announcement details in this page. For examples, unpublish a specific announcement or add new tags to the announcement.



CASE ID	TITLE	TYPE	STATUS	SUBMITTED BY	UPDATED AT	CREATED AT	ACTION
# 6	Lift services	Facilities	Completed	B3-F3-3 Ye Tan	Yesterday	Yesterday	
# 5	Access card problem	Facilities	Pending	B3-F3-3 Ye Tan	Yesterday	Yesterday	
# 4	Wifi issue	Facilities	Pending	B3-F3-3 Ye Tan	Yesterday	Yesterday	
# 1	Broken window	Facilities	Completed	B1-F1-1 Milan Clemons	11 April 2020	11 April 2020	
# 2	Water leakage	Facilities	Processing	B1-F1-1 Milan Clemons	11 April 2020	11 April 2020	

Figure 5.14 Feedback list in table

The figure above shows the feedback received in table form. Community management user can create new feedback and select existing feedback to update in this page.



**Create feedback** [Feedback](#) » [Create](#)

**Title\***  
Title

**Description**  
Describe the feedback

**Feedback type**  
Facilities

**Submit by**  
+ Add resident  
[Select a resident](#)

[Create](#) [Cancel](#)

Figure 5.15 Create feedback page

Community management user can help resident to create feedback case and assign the submit by field to the specific resident.

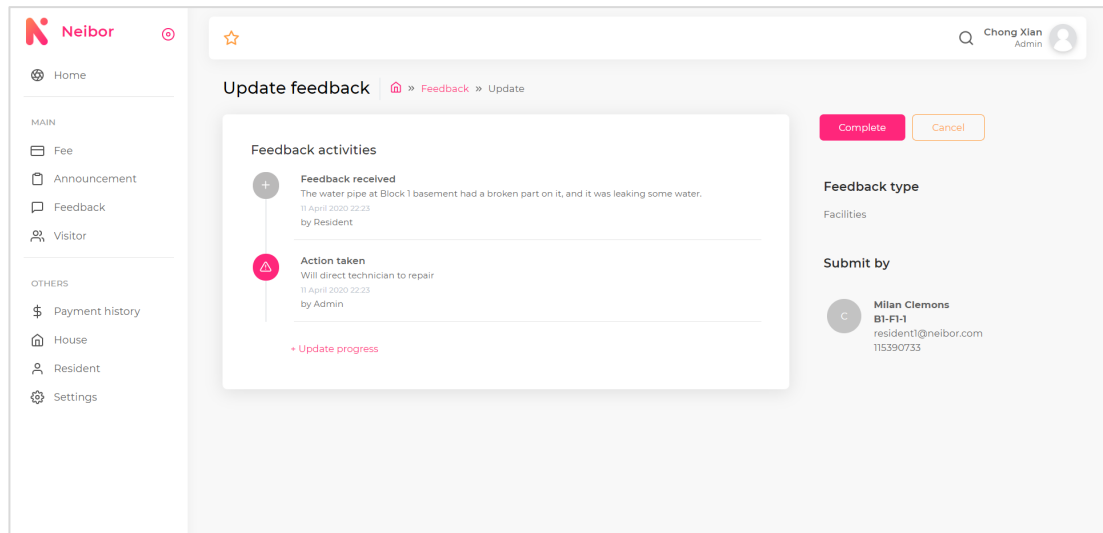


Figure 5.16 Update feedback page

The screen above shown the update feedback page of the system. In this page, residential management user can update the feedback progress to resident. In some circumstances, residential management user can help resident to complete the specific feedback case.

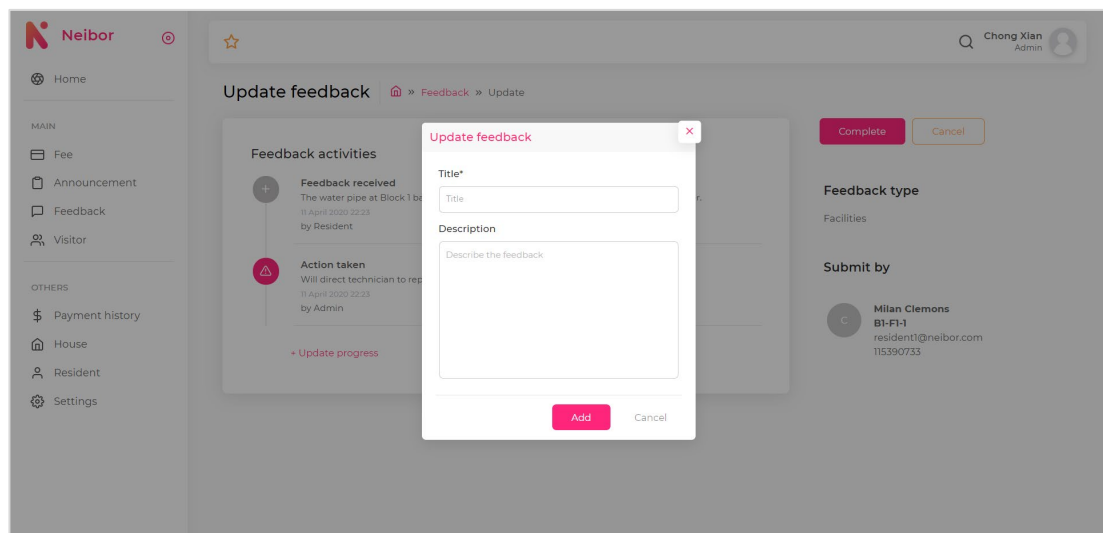


Figure 5.17 Update feedback dialog

The update feedback dialog will pop up when the residential management user clicks the update progress button. Details of the feedback update can be filled in here to update the progress.

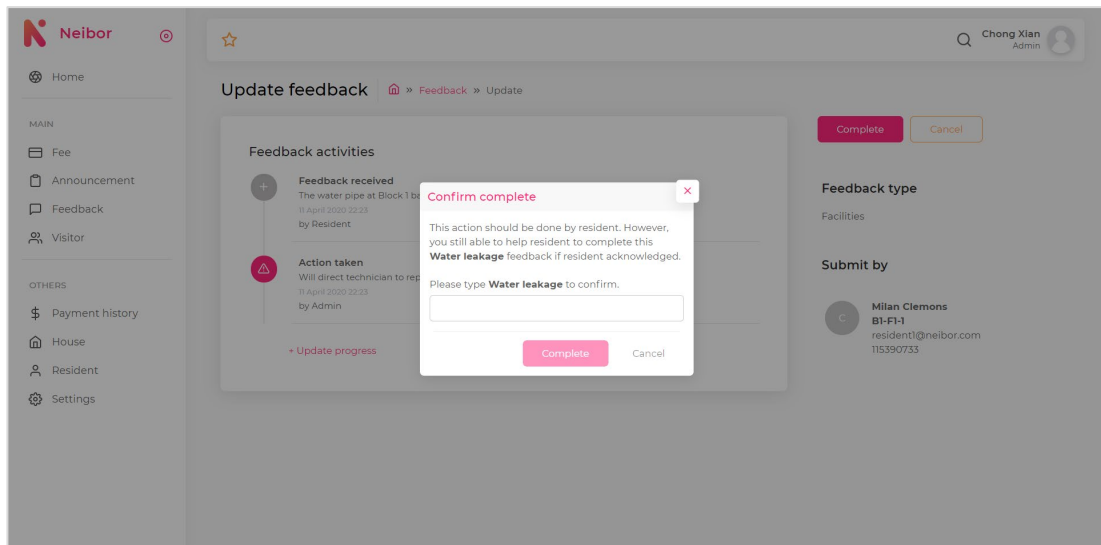


Figure 5.18 Complete feedback dialog

The complete feedback dialog will prompt the residential management user when he/she want to complete the specific feedback. The dialog will display some information to indicate that the feedback case should be completed by resident when the feedback case problem is solved and it needs double confirmation to continue to complete the feedback with role of residential management user.

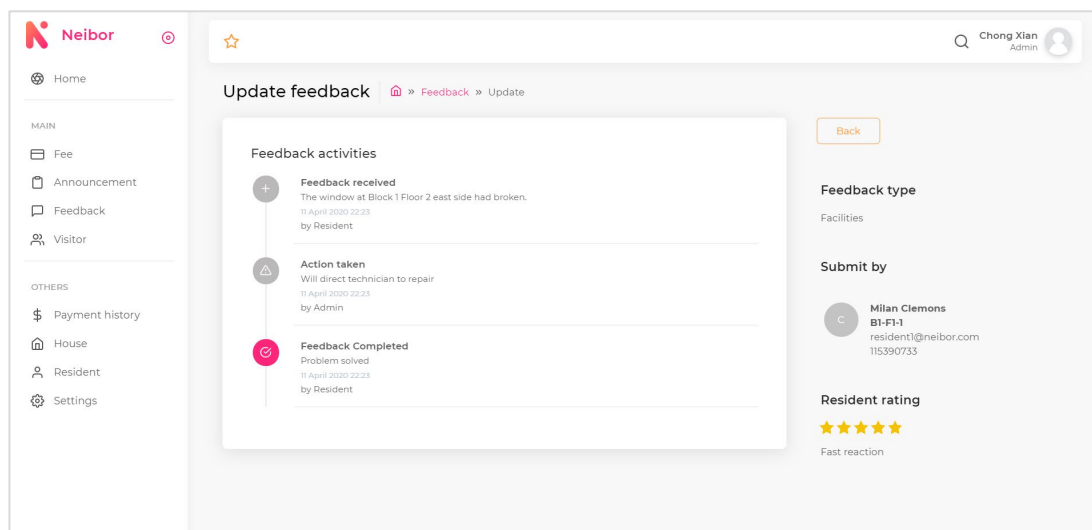
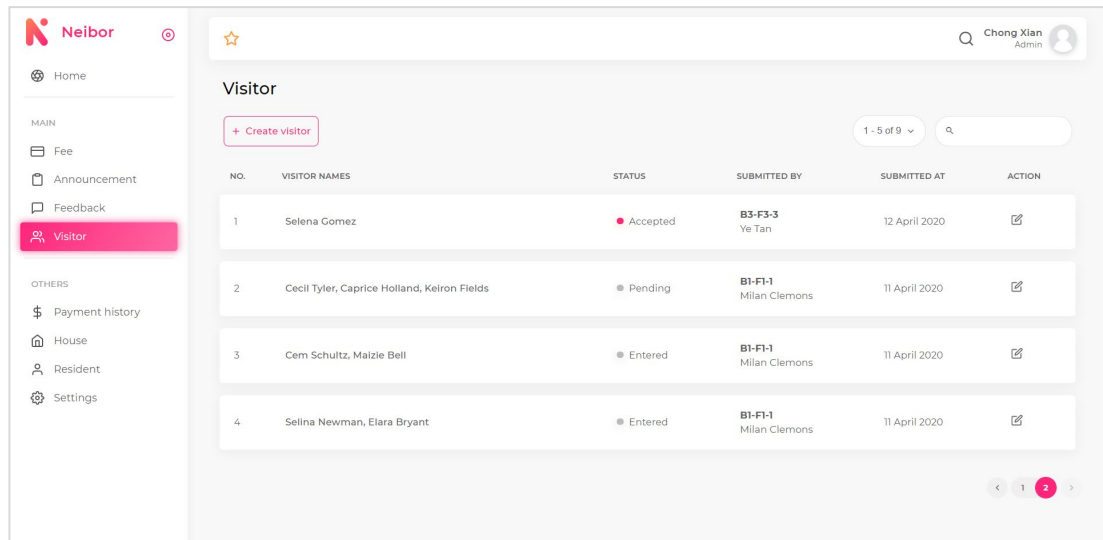


Figure 5.19 Completed feedback page

The page above shown a completed feedback with resident rating. As the figure shown, no other action such as update feedback progress or complete feedback can be done in this page as the feedback case is completed.

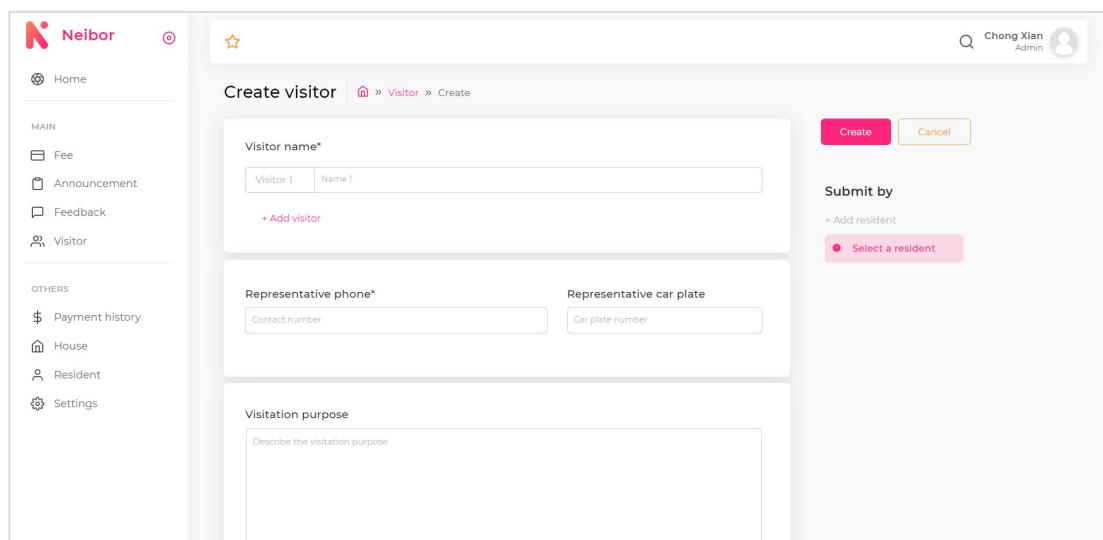




NO.	VISITOR NAMES	STATUS	SUBMITTED BY	SUBMITTED AT	ACTION
1	Selena Gomez	Accepted	B3-F3-3 Ye Tan	12 April 2020	<a href="#">✎</a>
2	Cecil Tyler, Caprice Holland, Keiron Fields	Pending	B1-F1-1 Milan Clemmons	11 April 2020	<a href="#">✎</a>
3	Cem Schultz, Maizie Bell	Entered	B1-F1-1 Milan Clemmons	11 April 2020	<a href="#">✎</a>
4	Selina Newman, Elara Bryant	Entered	B1-F1-1 Milan Clemmons	11 April 2020	<a href="#">✎</a>

Figure 5.20 Visitor list in table form

The figure above shows the visitor application received in table form. Community management user can create new visitor and select existing visitor to update in this page.



**Create visitor** » Visitor » Create

**Visitor name\***

Visitor 1 | Name 1

+ Add visitor

**Representative phone\*** | **Representative car plate\***

Contact number | Car plate number

**Visitation purpose**

Describe the visitation purpose

Create | Cancel

**Submit by**

+ Add resident

Select a resident

Figure 5.21 Create new visitor page

In the create new visitor page above, a maximum limit of 10 visitors' name can be filled in for the visitor application. Besides, representation phone, car plate number and visitation purpose can be filled in also. As like feedback, community management user can help resident to create visitor application and assign the submit by field to the specific resident.

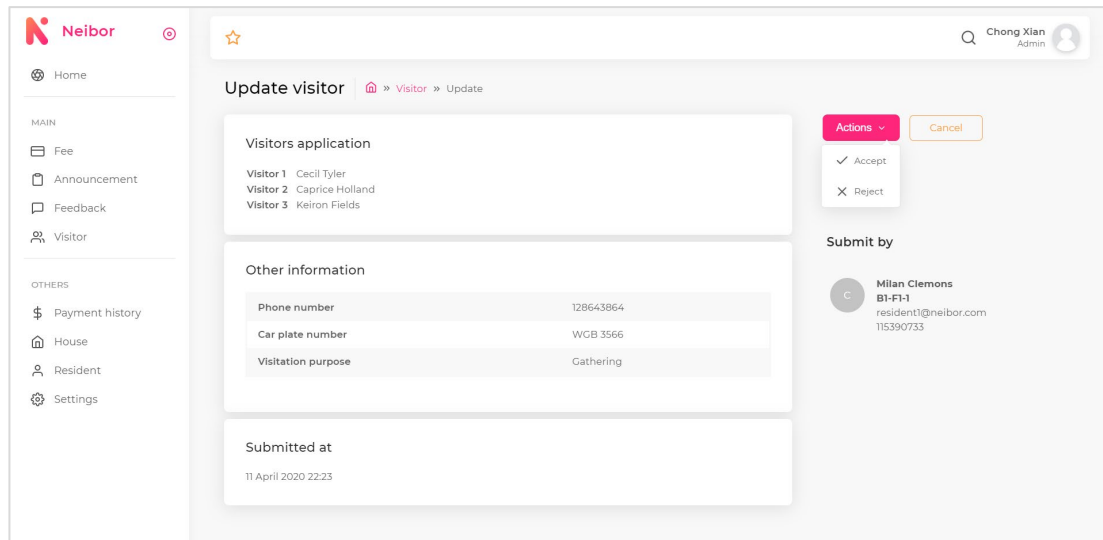


Figure 5.22 Update visitor page

Community management user can review the visitor application details such as names, visitation purpose etc. and decide to approve or reject the specific visitor application here.

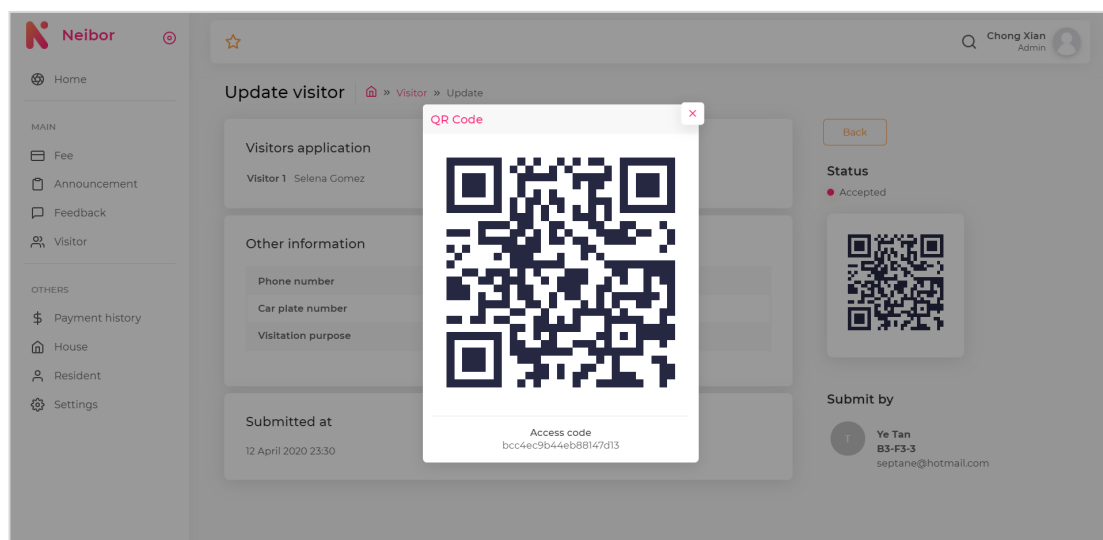


Figure 5.23 QR code generated for visitor

The figure above shown the QR code generated when the visitor application had been approved by the community management user.

REFERENCE ID	AMOUNT	METHOD	PAID BY	CREATED AT	ACTION
#111-00102	RM 87.20	CARD	B1-F1-1 Milan Clemons	11 April 2020	
#111-00103	RM 87.20	CARD	B1-F1-2 Tina Evans	11 April 2020	
#111-00104	RM 87.20	CARD	B1-F1-3 Billie Mccoy	11 April 2020	

Figure 5.24 Payment histories list in table

The figure above shows the payment histories of residents in table form. Community management user can select existing payment history to view its details in this page.

Transaction #111-00102	
Amount	RM 69.00
Payment method	CARD
Through	
Billing Details	
Address	C1-F1-1, Cypress Bandar Sungai Long 43000 Kajang Selangor
Email	resident1@neibor.com
Phone	115390733
Created at	

**Paid by**

**Milan Clemons**  
B1-F1-1  
resident1@neibor.com  
115390733

Figure 5.25 View payment history page

In this page, community management user can view the details of a specific payment history by resident. The details of payment history record may include amount, fees involved, billing information etc.

NO.	HOUSE NUMBER	BLOCK	FLOOR	UNIT	RESIDENT NAME	RESIDENT EMAIL	ACTION
1	B1-F1-1	1	1	1	Milan Clemons	resident1@neibor.com	
2	B1-F1-2	1	1	2	Tina Evans	resident2@neibor.com	
3	B1-F1-3	1	1	3	Billie Mccoy	resident3@neibor.com	
4	B1-F2-1	1	2	1	Michael Lim	resident4@neibor.com	
5	B1-F2-2	1	2	2	Magnum Mike	resident5@neibor.com	

Figure 5.26 House list in table

The figure above shows the houses of residence in table form. Community management user can create new house and select a house to view its details in this page.

**Create house** [Home](#) » [House](#) » Create

House numbering\*

Block  Numbering 1  Floor  Numbering 2  Unit  Numbering 3

**Resident**

+ Select existing resident  
or  
+ Add new resident's account

**Preview**

House ... will be created

Figure 5.27 Create house page

In this create house page, community management user can fill in the house numbering of the house to create new house. One of the house numbering examples is B1-F1-1. Besides, a resident can directly assign here either using existing resident or create new resident.

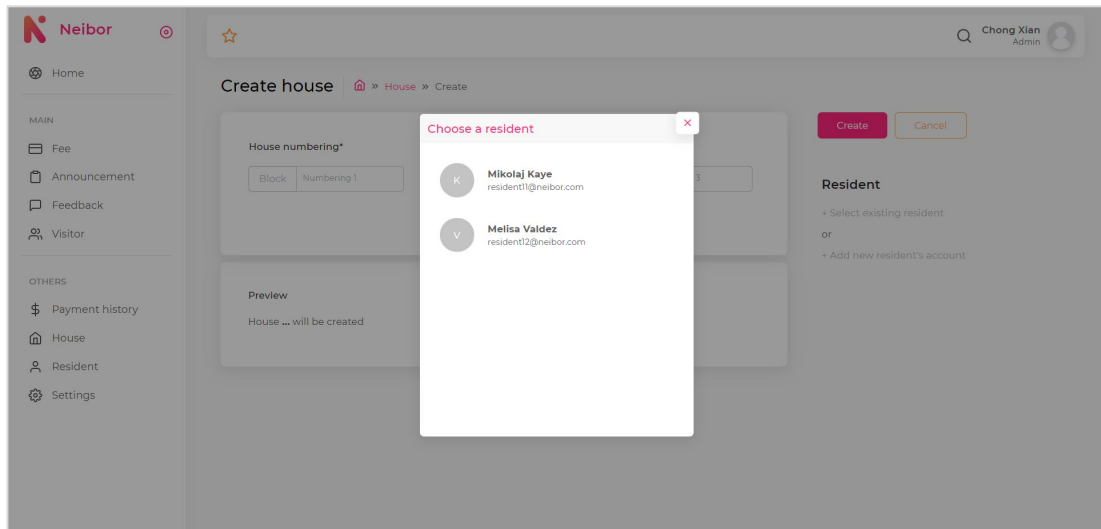


Figure 5.28 Select resident dialog

The select resident dialog will display the available resident to be assigned into the specific house.

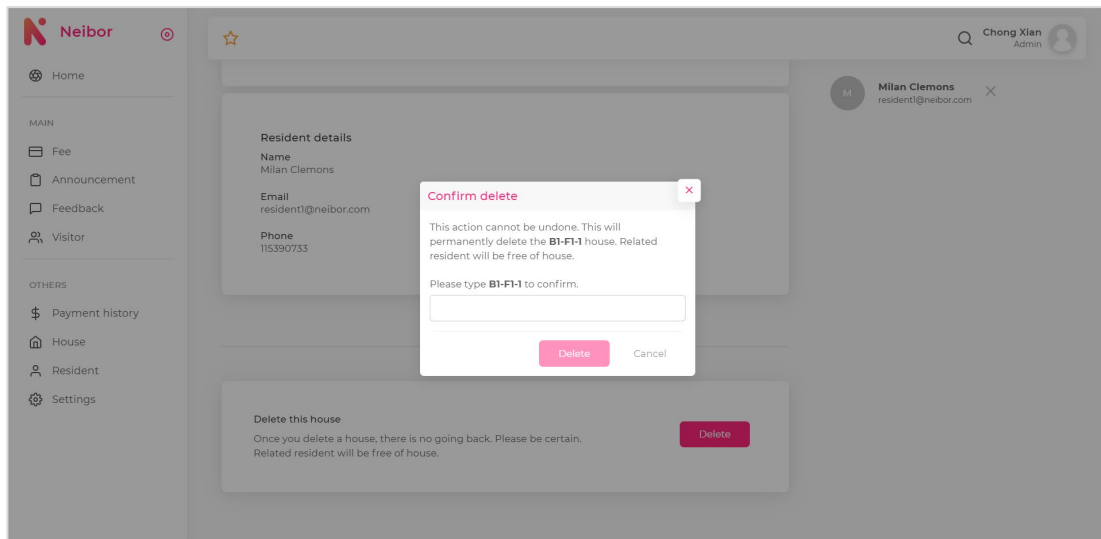


Figure 5.29 Delete house dialog

A confirmation dialog will pop up when community management user presses the delete house button located at the bottom of the view house page. The dialog is used to double confirm the delete operation of specific house.

NO.	NAME	HOUSE	EMAIL	PHONE	ACTION
1	Milan Clemons	B1-F1-1	resident1@neibor.com		
2	Tina Evans	B1-F1-2	resident2@neibor.com		
3	Billie Mccoy	B1-F1-3	resident3@neibor.com		
4	Michael Lim	B1-F2-1	resident4@neibor.com		
5	Magnum Mike	B1-F2-2	resident5@neibor.com		

Figure 5.30 Resident list in table

The figure above shows the residents of residence in table form. Community management user can add new resident account and select a resident to view its details in this page.

Figure 5.31 Create resident page

Community management user can fill in the resident's first name, last name and email to create a new resident account. A random generated password will be use as the default password for the account. In additions, house can be assigned to the specific resident by either select an existing house or create new house.

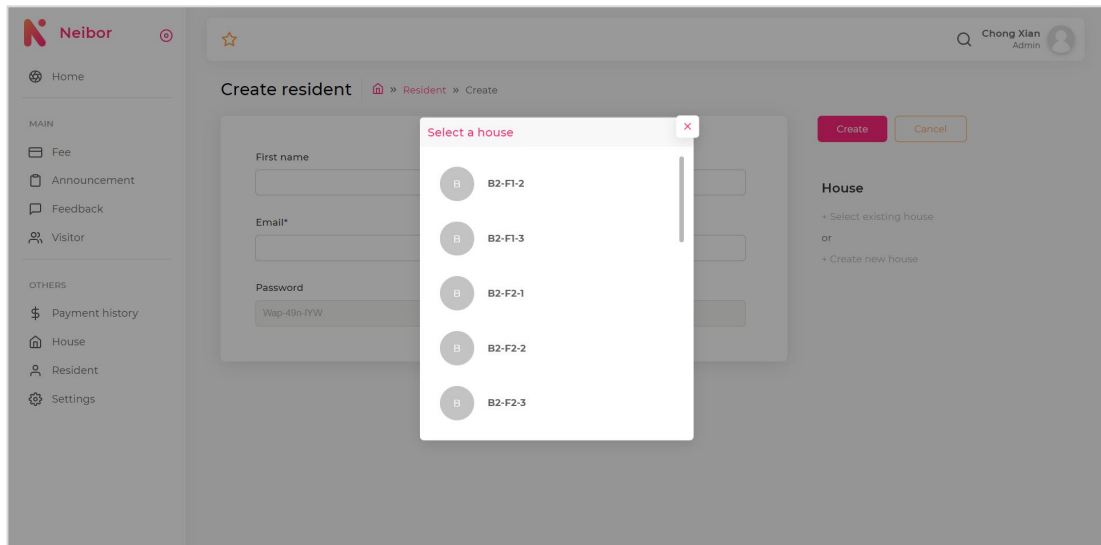


Figure 5.32 Select house dialog

The select house dialog will display the house available to be assigned to the resident.

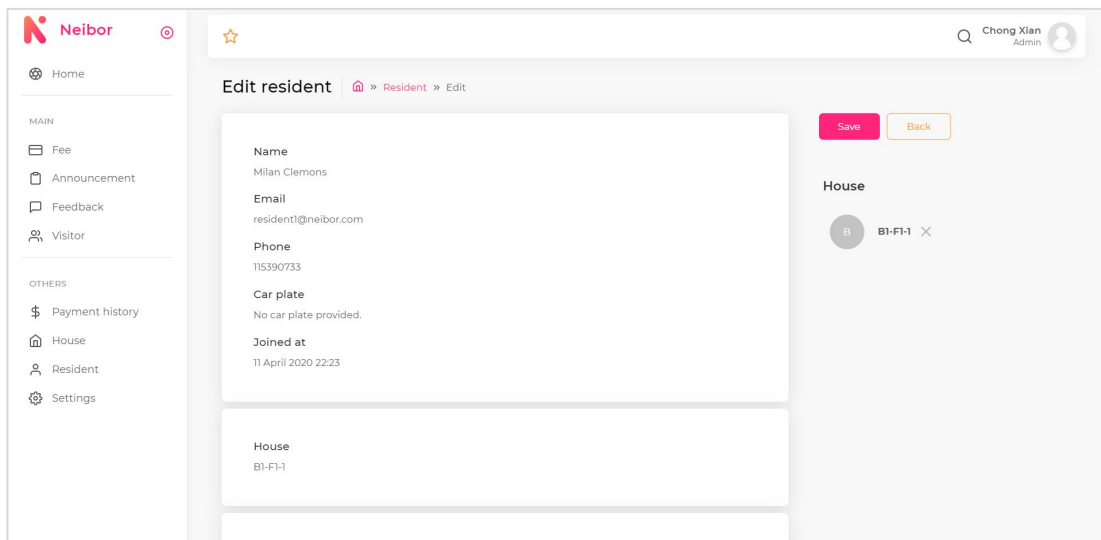


Figure 5.33 Edit resident page

In this page, community management user can view the personal information of resident such as phone number, car plate number, house, etc. Besides, the fees list involved will also displayed here. Community management user also can remove the house assigned to this resident in this page.

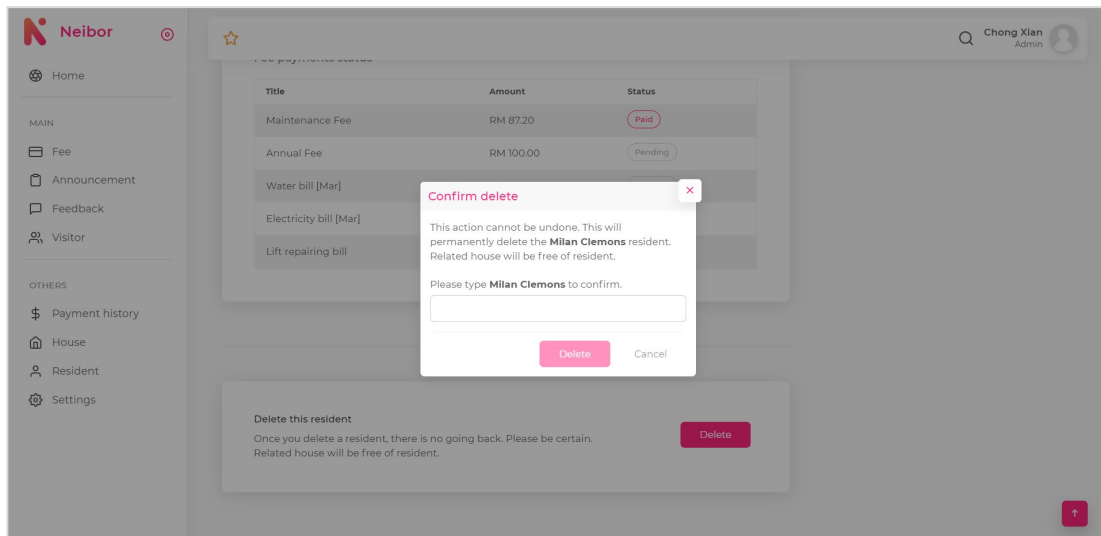


Figure 5.34 Delete resident dialog

A confirmation dialog will pop up when community management user presses the delete resident button located at the bottom of the view resident page. The dialog is used to double confirm the delete operation of specific resident account.

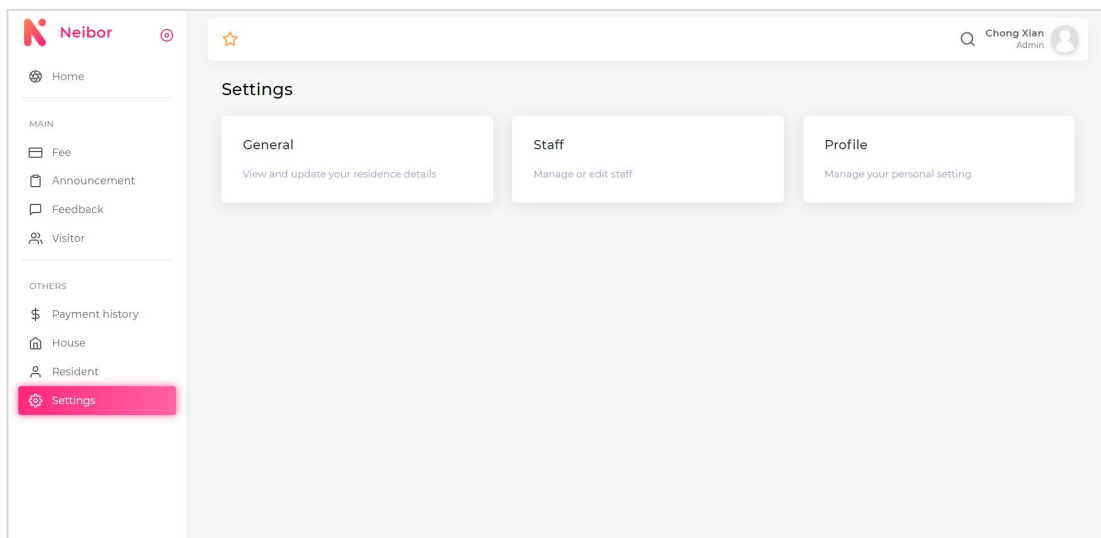


Figure 5.35 Settings page

Settings page is displaying three card buttons, which is about general information of resident, staff of residents and profile of specific user.



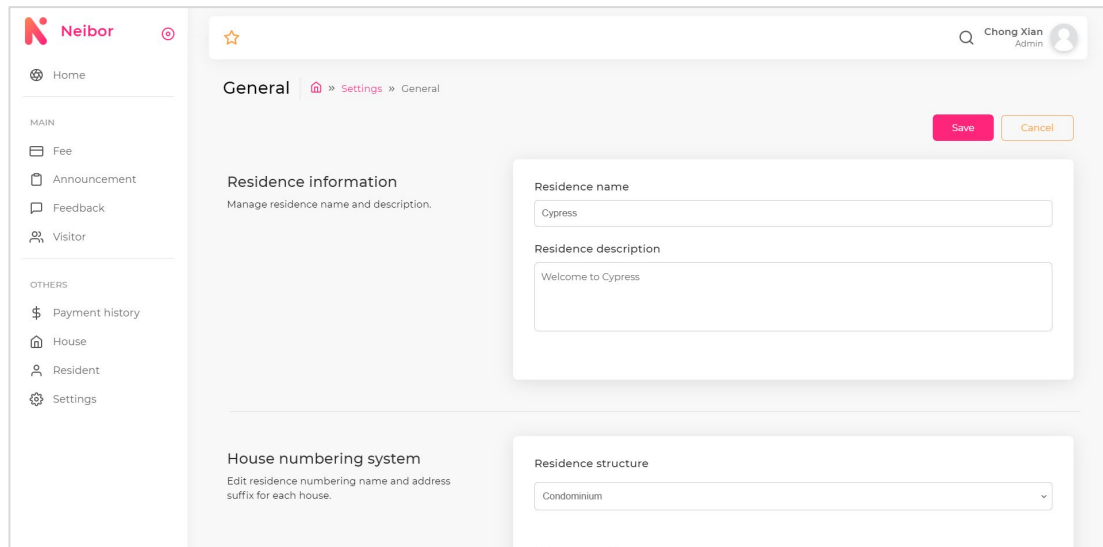


Figure 5.36 General information of residence page

This page allows community management use to view or edit the general information of residence such as name, description, etc.

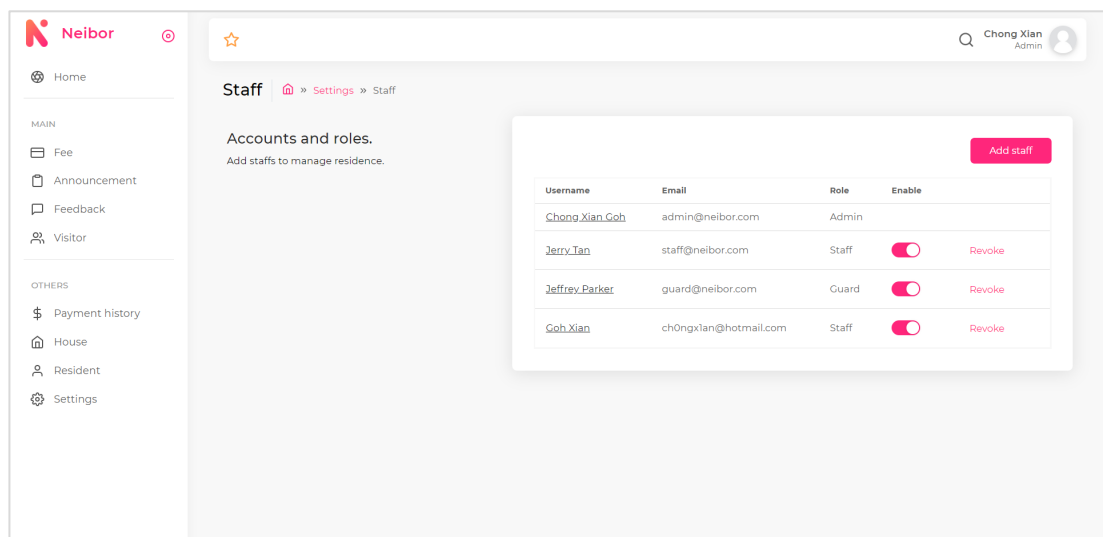


Figure 5.37 Staffs page

This page displayed the staffs of residence and community management user can add new staff, edit staff, enable or disable staff account, and revoke staff account in this page.

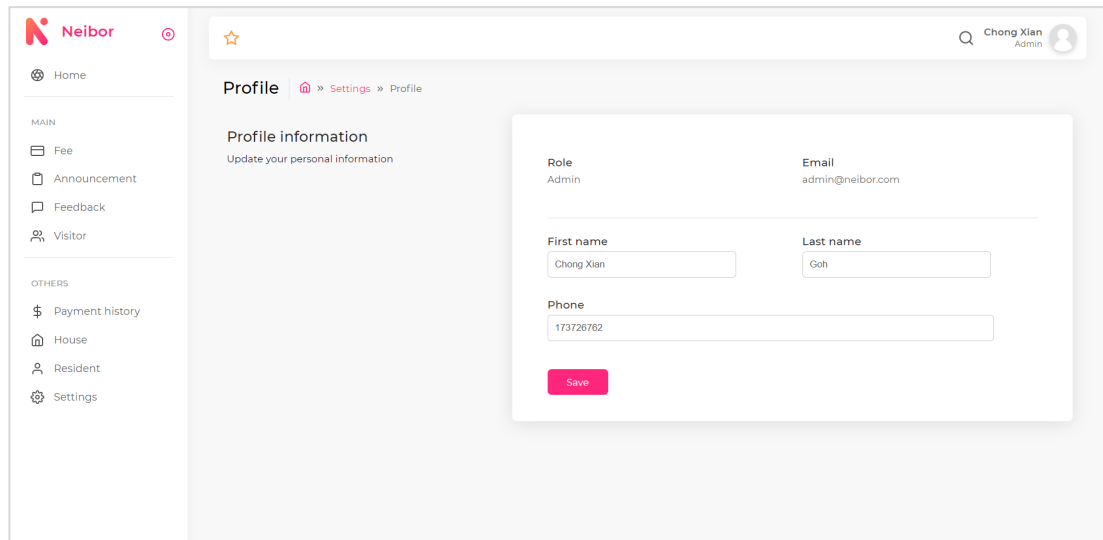


Figure 5.38 Profile page

This profile page will display the basic information of the specific user and allow he/she to edit the information.

## 5.5.2 Mobile Application for Community Resident

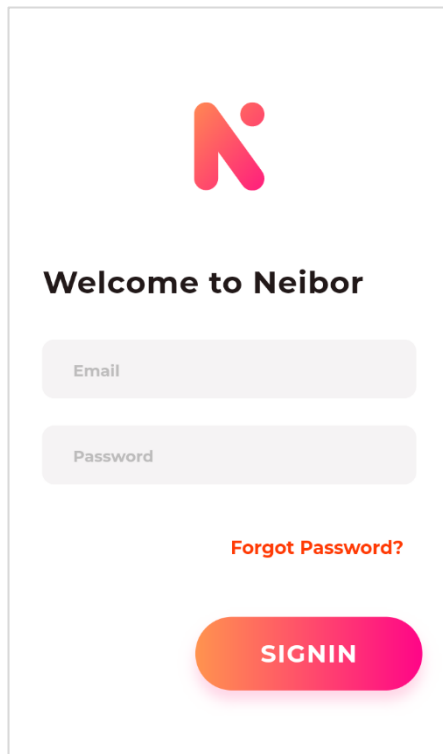


Figure 5.39 Login screen for mobile application

The figure 5.39 is the login screen for the mobile application. Residents will need to enter their email and password to login into the mobile application. Besides, there are one forgot password button to direct the resident to the recover password screen.

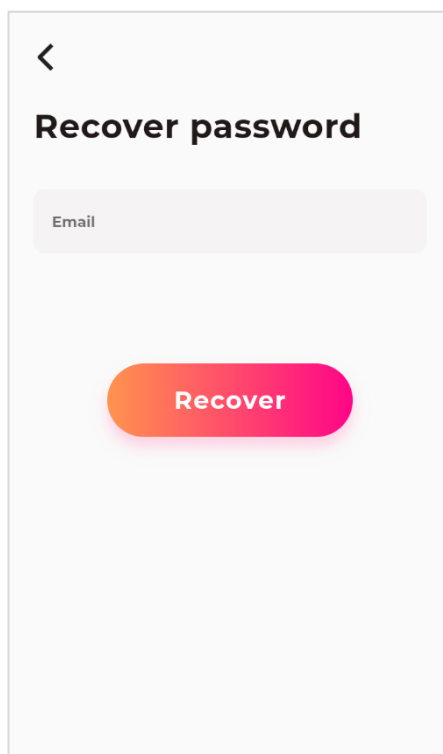


Figure 5.40 Recover password screen

The figure 5.40 is the recover password screen for the mobile application. Residents will need to enter their email, and received a recovery email to reset their forgotten password.

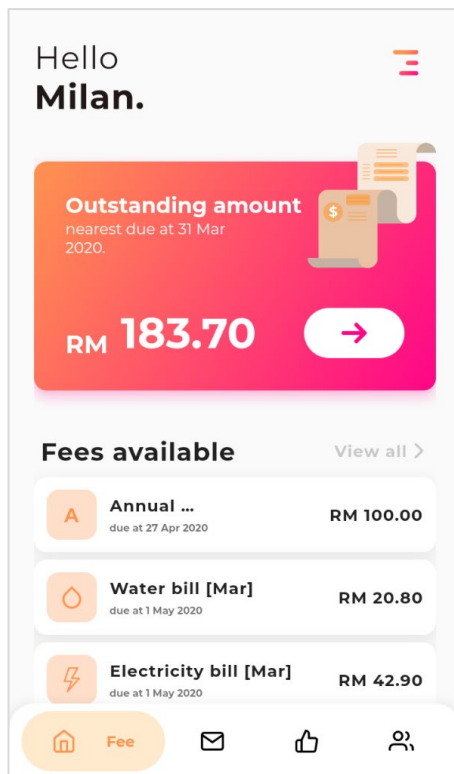


Figure 5.41 Fee main screen

The figure 5.41 is the main screen will be displayed to the residents after they login. In this screen, they can view their outstanding amount of the fee, view the up to 5 fees available and 5 payment histories.

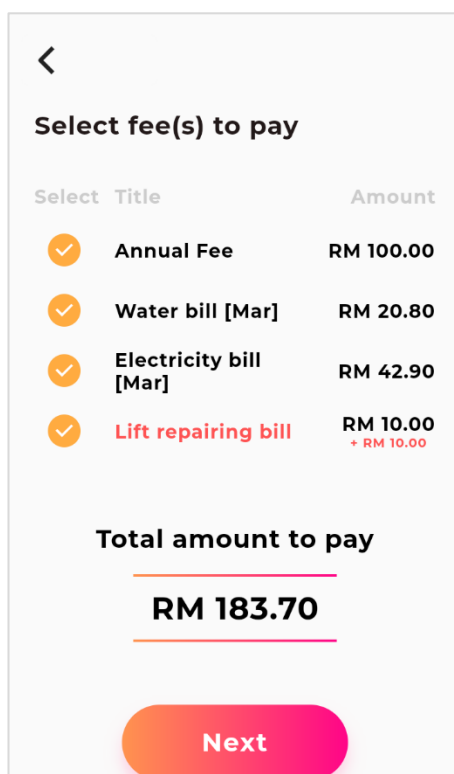


Figure 5.42 Select fee screen

The figure 5.42 is the select fee screen. Residents can select the fee they wished to pay and proceed to the payment process.



<

**Pay fee**

---

Total amounts  
**RM 183.70**

---

Card information

Card number\*

Expire\*      CVC\*

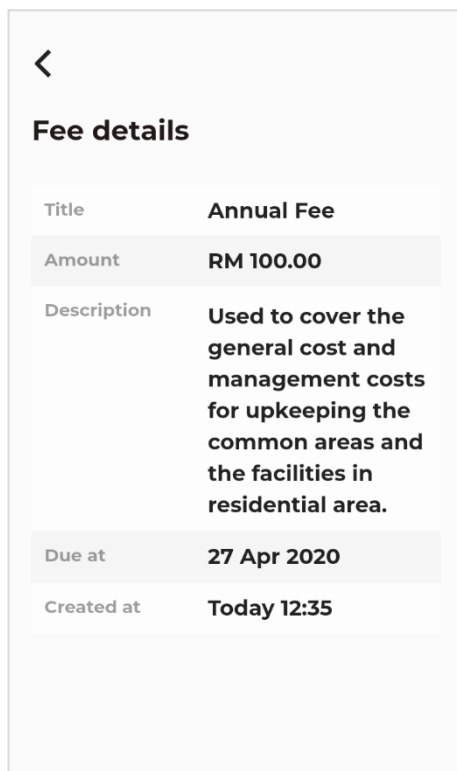
Card holder\*

Billing address (optional)

Line 1

Figure 5.43 Payment screen

The figure 5.43 is the fee payment screen. Resident need to filled in his/her debit or credit card details to pay the outstanding fees.



<

**Fee details**

Title	<b>Annual Fee</b>
Amount	<b>RM 100.00</b>
Description	<b>Used to cover the general cost and management costs for upkeeping the common areas and the facilities in residential area.</b>
Due at	<b>27 Apr 2020</b>
Created at	<b>Today 12:35</b>

Figure 5.44 View fee screen

The figure 5.44 is the view fee details screen. This screen will appear after user press a specific fee to view. It will show the fee's title, amount, description, etc.

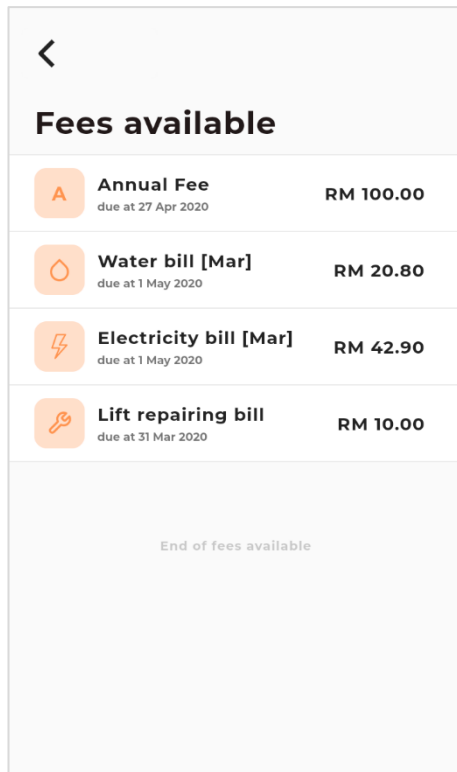


Figure 5.45 View all fees screen

The figure 5.45 is the view all fees screen. This screen contains a list view of all the fees available.

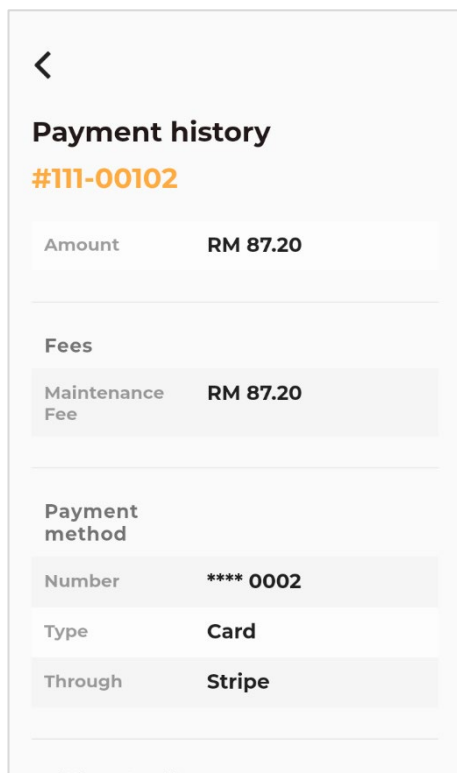


Figure 5.46 View payment history screen

The figure 5.46 is the view payment history screen. This screen will appear after user press a specific payment history to view. It will show the payment history amount, fees involved, payment method, billing details, etc.

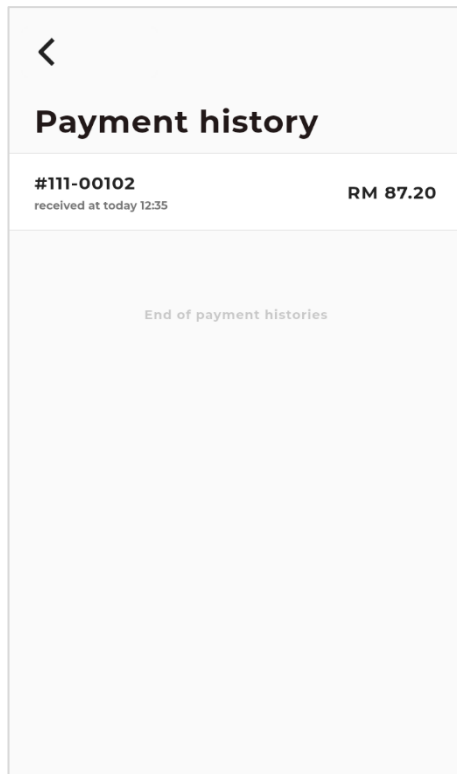


Figure 5.47 View all payment histories screen

The figure 5.47 is the view all payment histories screen. This screen shows all the payment histories in in list view.

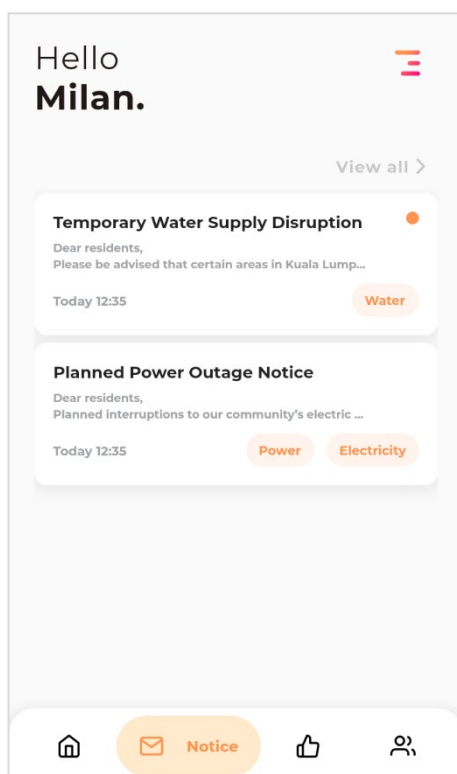


Figure 5.48 Announcement main screen

The figure 5.48 is the announcement main screen. This screen will show up to 5 announcements card. Residents can press a specific announcement card to view the details of the announcement.

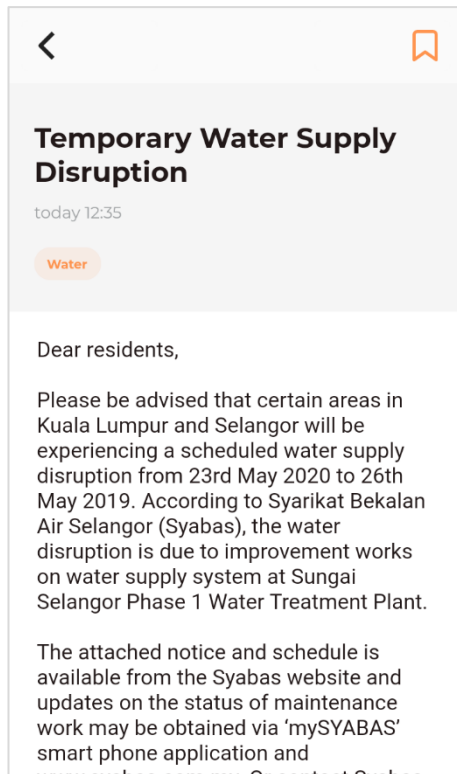


Figure 5.49 View announcement screen

The figure 5.49 is the view announcement screen. The content of the announcement will be display using HTML viewer, so special formatting such as bold, list, etc. will be retained. On the top right corner of the screen, residents can press the button to bookmark or un-bookmark the announcement. Bookmarked announcement will be shown at the top of the announcement screen.

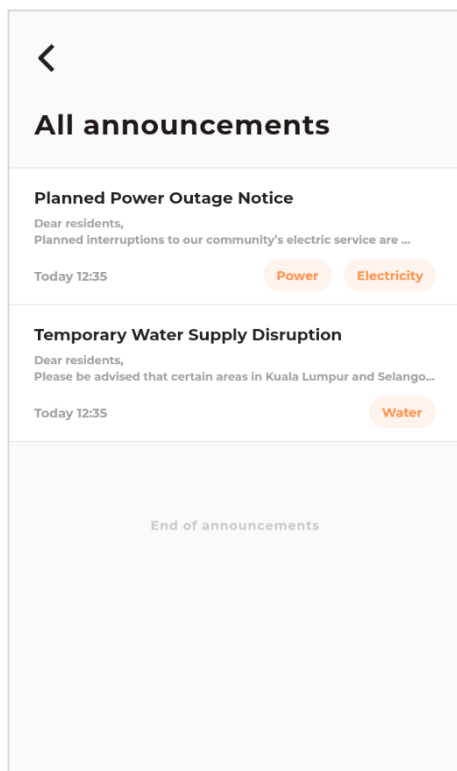


Figure 5.50 View all announcement screen

The figure 5.50 is the view all announcements screen. This screen shows all the announcements in in list view.



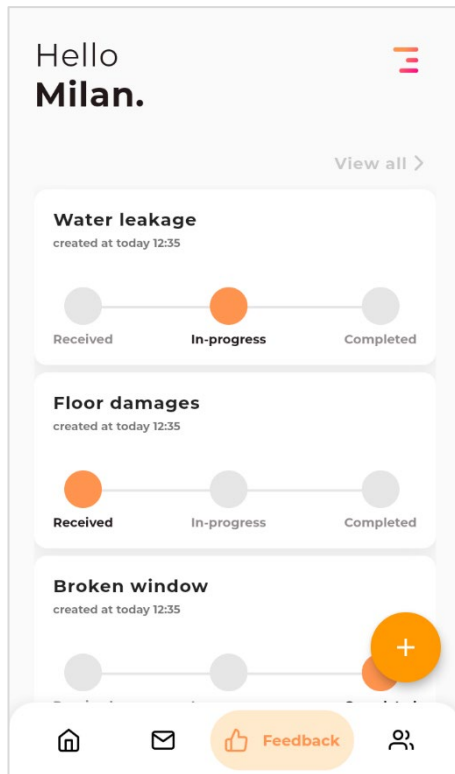


Figure 5.51 Feedback main screen

The figure 5.51 is the feedback main screen. Some preview information of each feedback case is displayed in this screen. The screen will show up to 5 feedback cases. The add floating button at right bottom button will directly resident to create new feedback case screen.

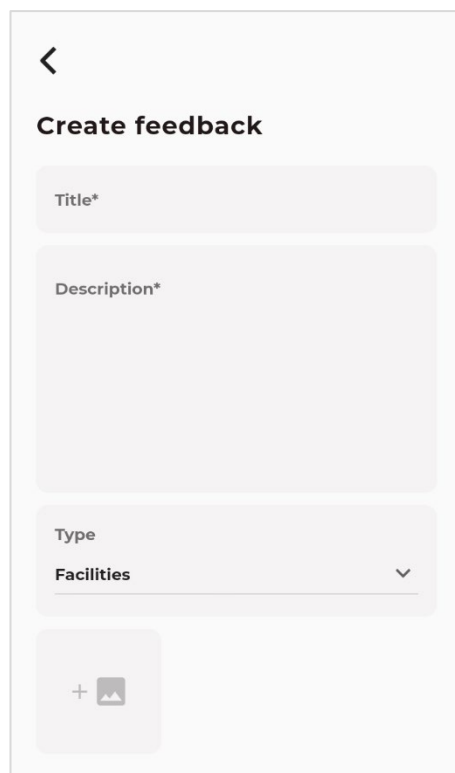


Figure 5.52 Create feedback screen

The figure 5.52 is the create feedback screen. Title, description, type, and images related to the feedback is allowed to filled in and submit as a feedback case.

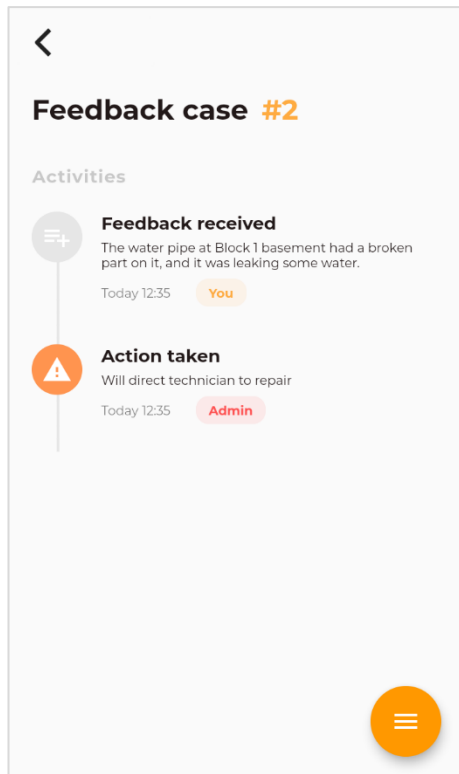


Figure 5.53 View feedback screen

The figure 5.53 is the view feedback screen. The progress of the feedback will be displayed here.

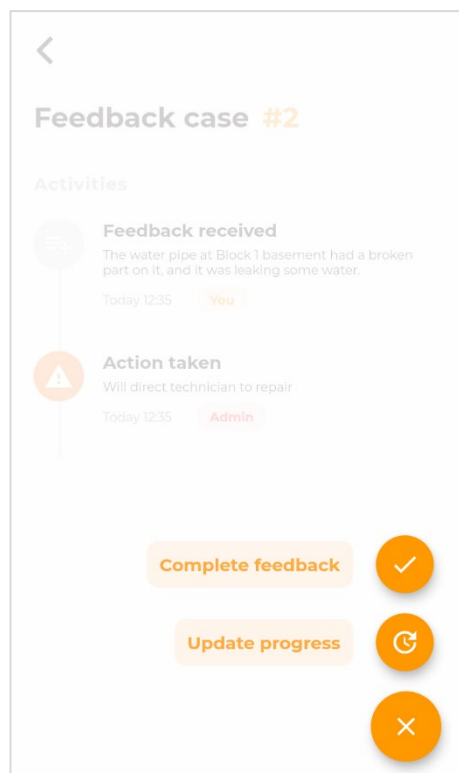
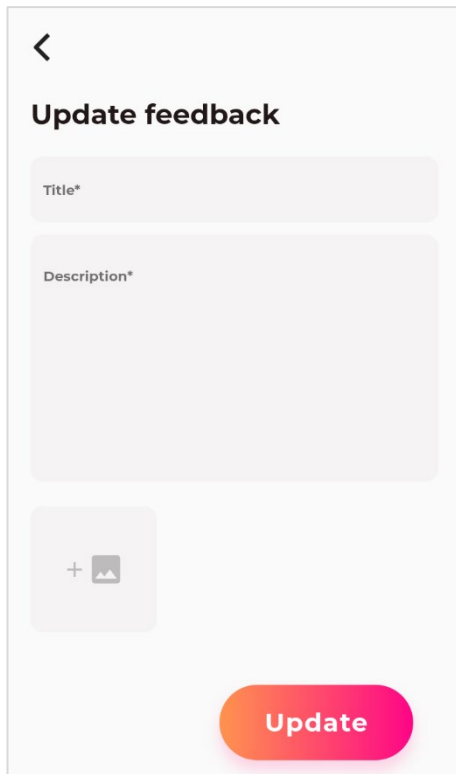


Figure 5.54 Action available for feedback

There are two action can be operated in this screen which is complete feedback and update progress for feedback.



The screenshot shows a mobile application screen titled "Update feedback". At the top left is a back arrow. Below it is the title "Update feedback". There are three main input areas: a "Title\*" text field, a "Description\*" text area, and an image upload button with a plus sign and a camera icon. At the bottom right is a prominent pink "Update" button.

Figure 5.55 Update feedback screen

The figure 5.55 is the update feedback screen. Title, description and image of the update can be filled in here.

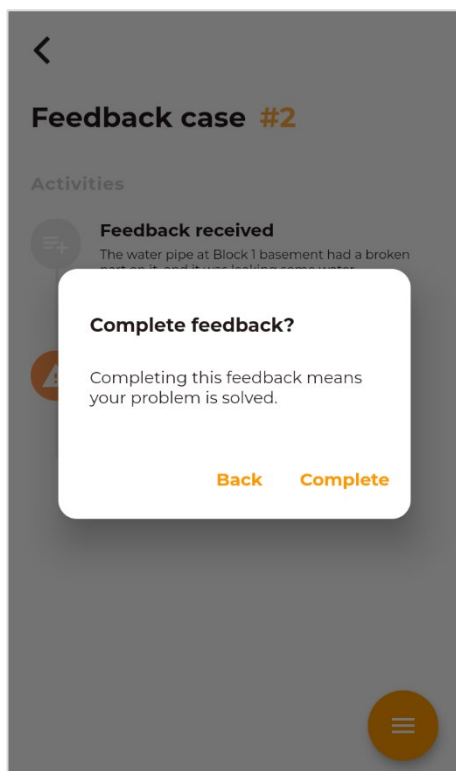


Figure 5.56 Complete feedback dialog

The figure 5.56 is the complete feedback dialog. The mobile application will prompt this confirmation dialog before completing feedback.

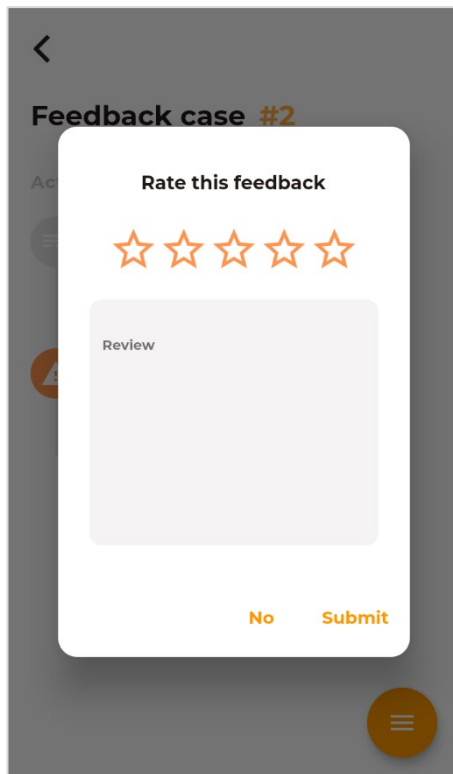


Figure 5.57 Rate feedback dialog

The figure 5.57 is the rate feedback dialog. The dialog will pop up to ask resident give rating and review about this feedback experience after user completed the feedback.

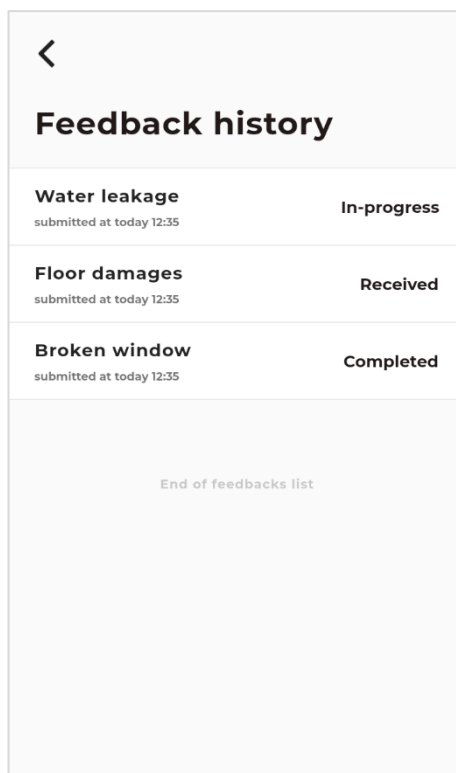


Figure 5.58 View all feedback screen

The figure 5.58 is the view all feedbacks screen. This screen shows all the feedbacks in in list view.

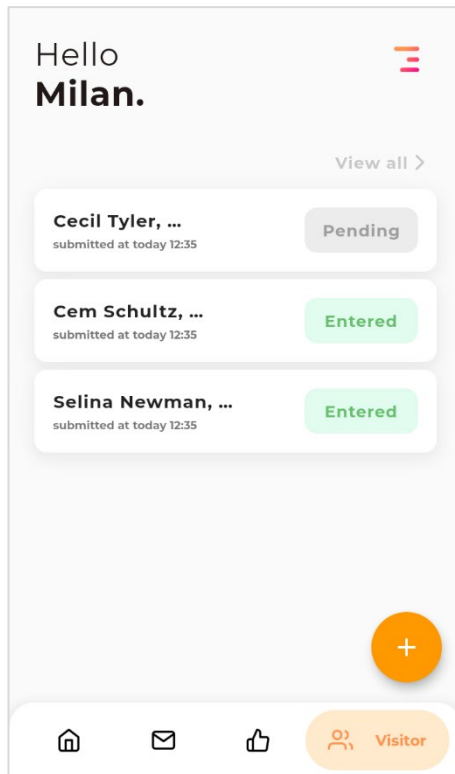


Figure 5.59 Visitor main screen

The figure 5.59 is the visitor main screen. Up to 5 visitor application will displayed in this screen including the details such as names of the visitor and status of the application. The floating action button at the bottom right corner will direct resident to the create visitor screen.

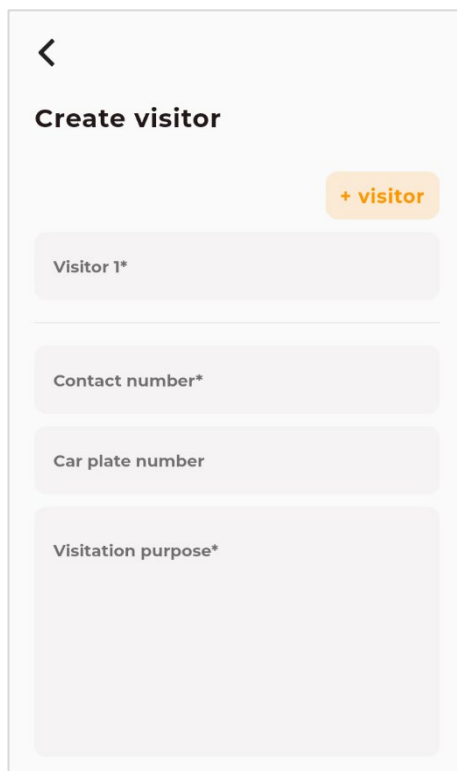


Figure 5.60 Create visitor screen

The figure 5.60 is the create visitor screen. Up to 10 visitors can be added in this application. Besides, phone number of the visitor's group's representative and visitation purpose is required to be filled in.

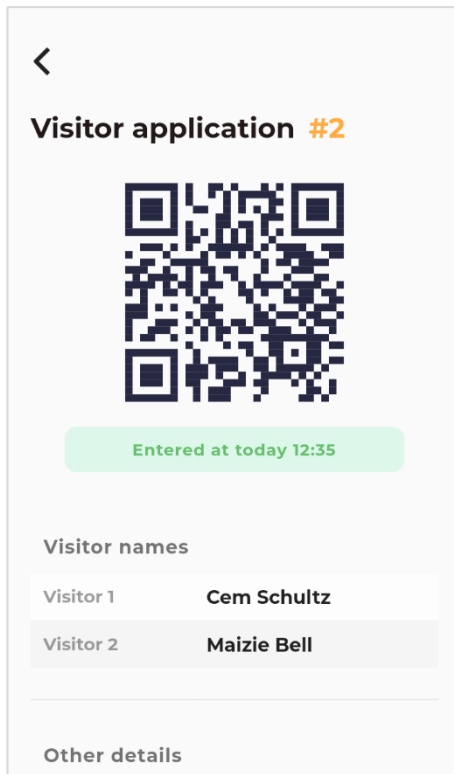


Figure 5.61 View visitor screen

The figure 5.61 is the view visitor screen. The QR code generated during the visitor application approved will be displayed here.

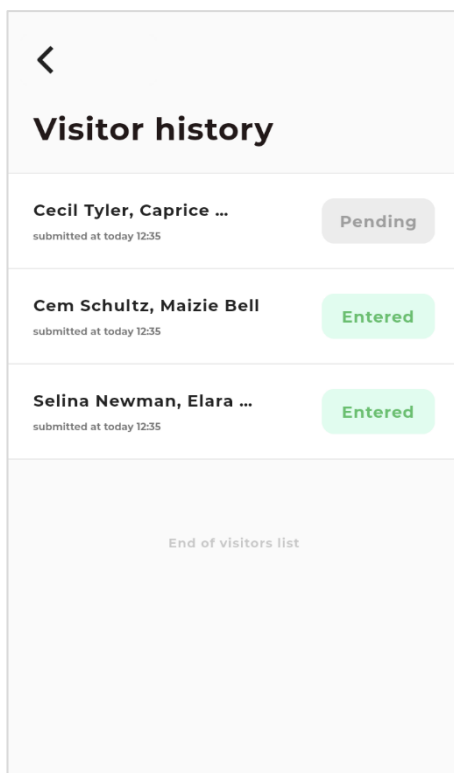


Figure 5.62 View all visitor screen

The figure 5.62 is the view all visitors screen. This screen shows all the visitors in in list view.

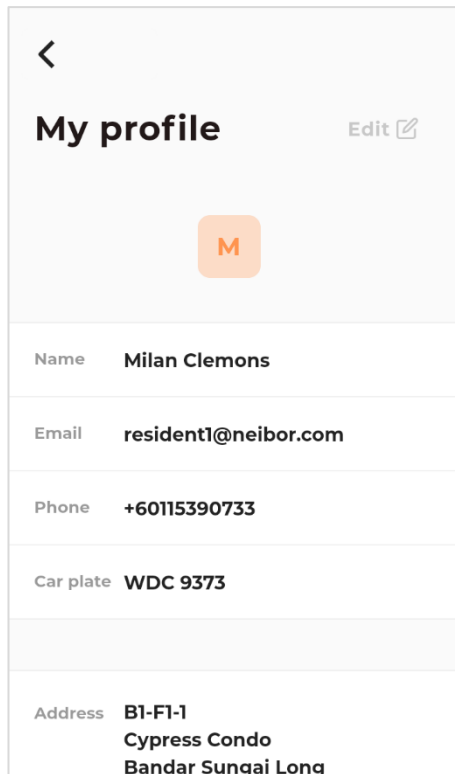


Figure 5.63 Profile screen

The figure 5.63 is the profile screen. It will display the personal information of resident, such as name, email, phone car plate, etc.

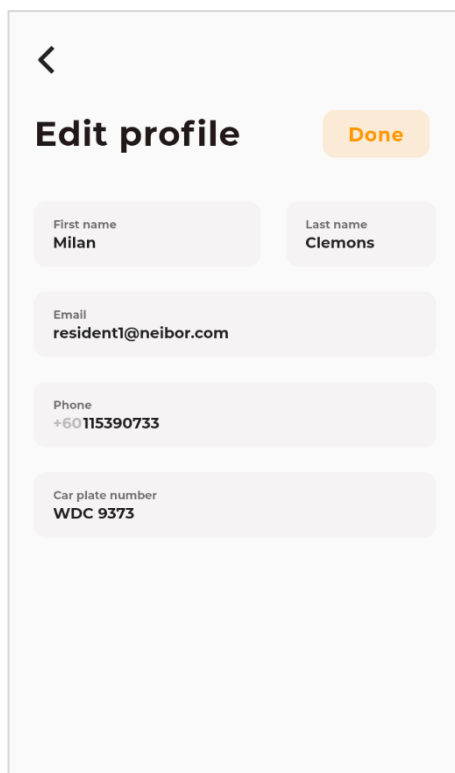


Figure 5.64 Edit profile screen

The figure 5.64 is the edit profile screen. It allows residents to update their personal information.

### 5.5.3 Mobile Application for Security Personnel

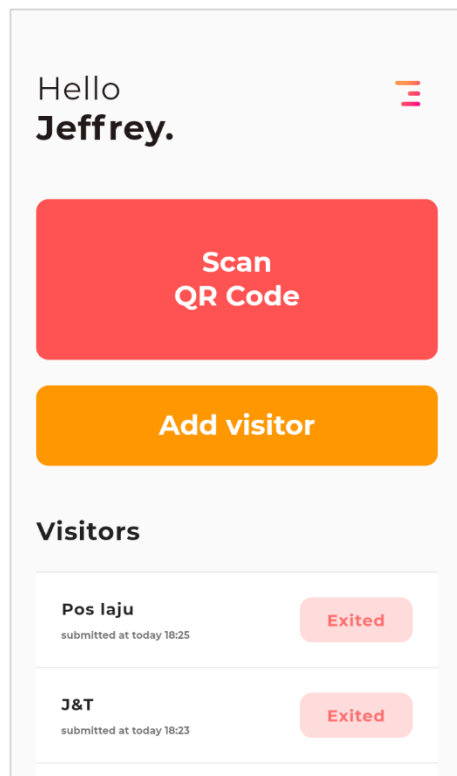


Figure 5.65 Guard home screen

The login flow of guard is same as resident. However, after login, different screen will be shown. The figure 5.65 is the guard mobile application home screen. The first button in the middle is used to scan the QR code presented by the visitor. The second button in the middle is used to add visitor with no QR code such as delivery personnel.

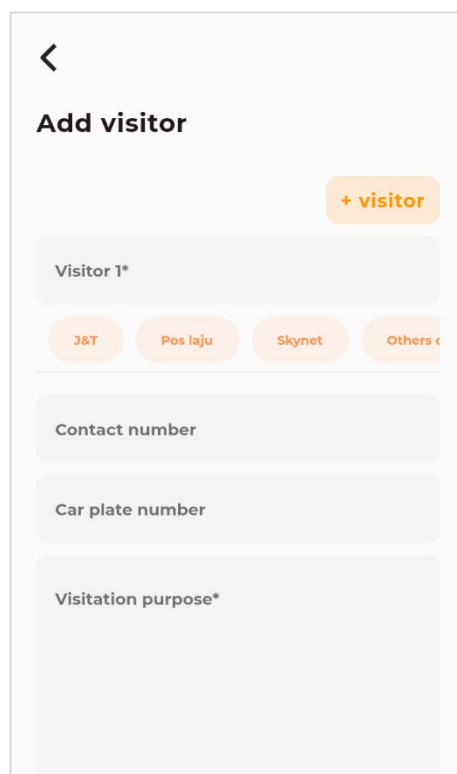


Figure 5.66 Guard add visitor screen

The figure 5.66 is add visitor screen. Security personnel can add new visitor with no QR code. A list of quick suggestions is displayed the name used to automatically fill in the visitor name and visitation purpose.



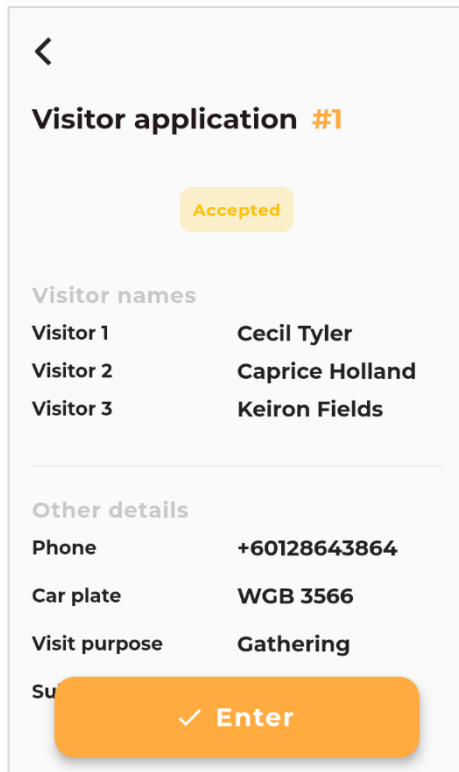


Figure 5.67 Visitor details screen

The figure 5.67 is the screen after QR code scanned. Visitor details will be displayed to allow guard to validate the visitors' details. An enter button is used to update the visitor status.

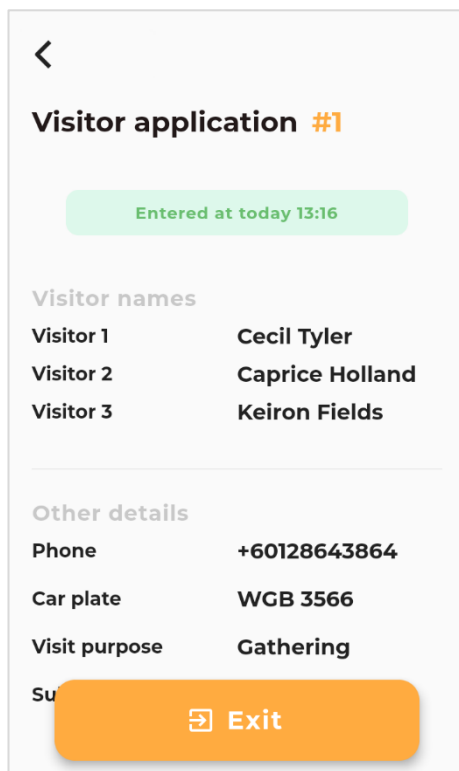


Figure 5.68 Entered visitor screen

The figure 5.68 is the entered visitor screen. It will appear when the entered visitor's QR code is scanned. An exit button is used to update the visitor status.

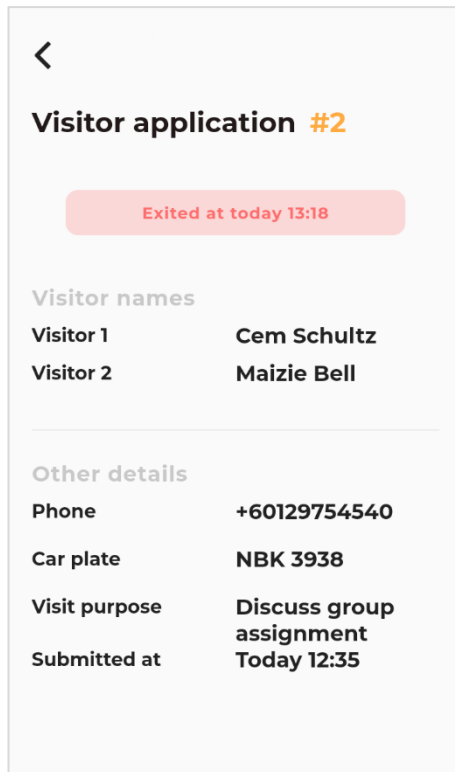


Figure 5.69 Exited visitor screen

The figure 5.69 is the exited visitor screen. It will appear when the exited visitor's QR code is scanned. Guard should not allow visitor to enter again.

## CHAPTER 6

### SYSTEM IMPLEMENTATION

#### 6.1 Backend Server

The Ruby on Rails backend server implemented Model View Controller (MVC) design pattern. In the whole system, the Controller layer and Model Layer is residing in the backend server. This section will compromise with an overview description of backend server and details explanation of each layers.

##### 6.1.1 Overview of Backend Server

This section will describe the overview flow of RESTful API request in the Ruby on Rails backend server created.

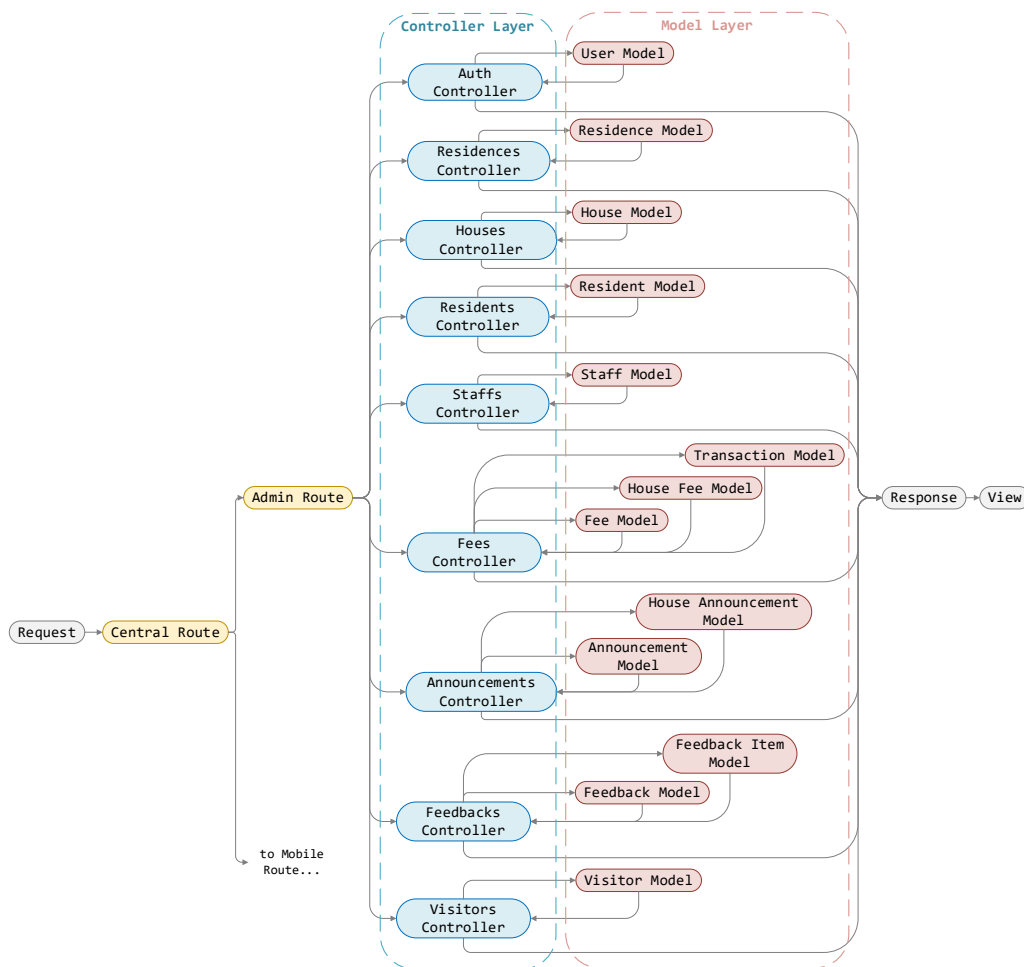


Figure 6.1 Flow of API request from web application being handle and respond

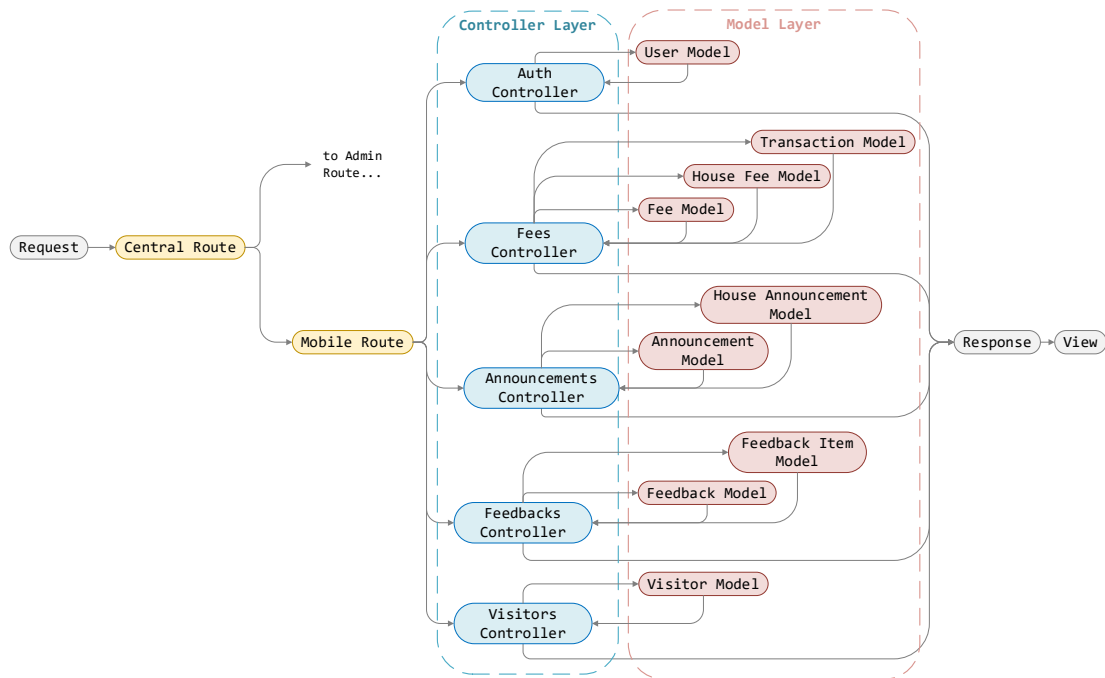


Figure 6.2 Flow of API request from mobile application being handle and respond

As the two figure diagrams shown, API requests from either mobile application or web application will be direct to admin route handler or mobile route handler respectively. After the route layer, the request and its payload will enter controller layer to its respective controller. For example, “Auth Controller” will handle request related to authentication. In controller, application logic on the input from request and interaction with the model will occurred. If every task or statement is completed successfully, response that contain the result or data should be return from controller to the source of input request for information rendering in view.

### 6.1.2 Controller Layer

In Controller layer, logical solutions that handle the request from client's application were defined. Controller will responsible to process or manipulate the data input and inform the Model for data retrieve, update, delete, etc. Reusability are emphasized in this layer to ensure the better organized architecture of this project. It is an essential approach as it can lead to better maintainability and readability of code (UKEssays, 2018). Inheritance is used to achieve the aims. The figure below is an overview of how the hierarchy of controller class inheritance in Ruby on Rails backend server.

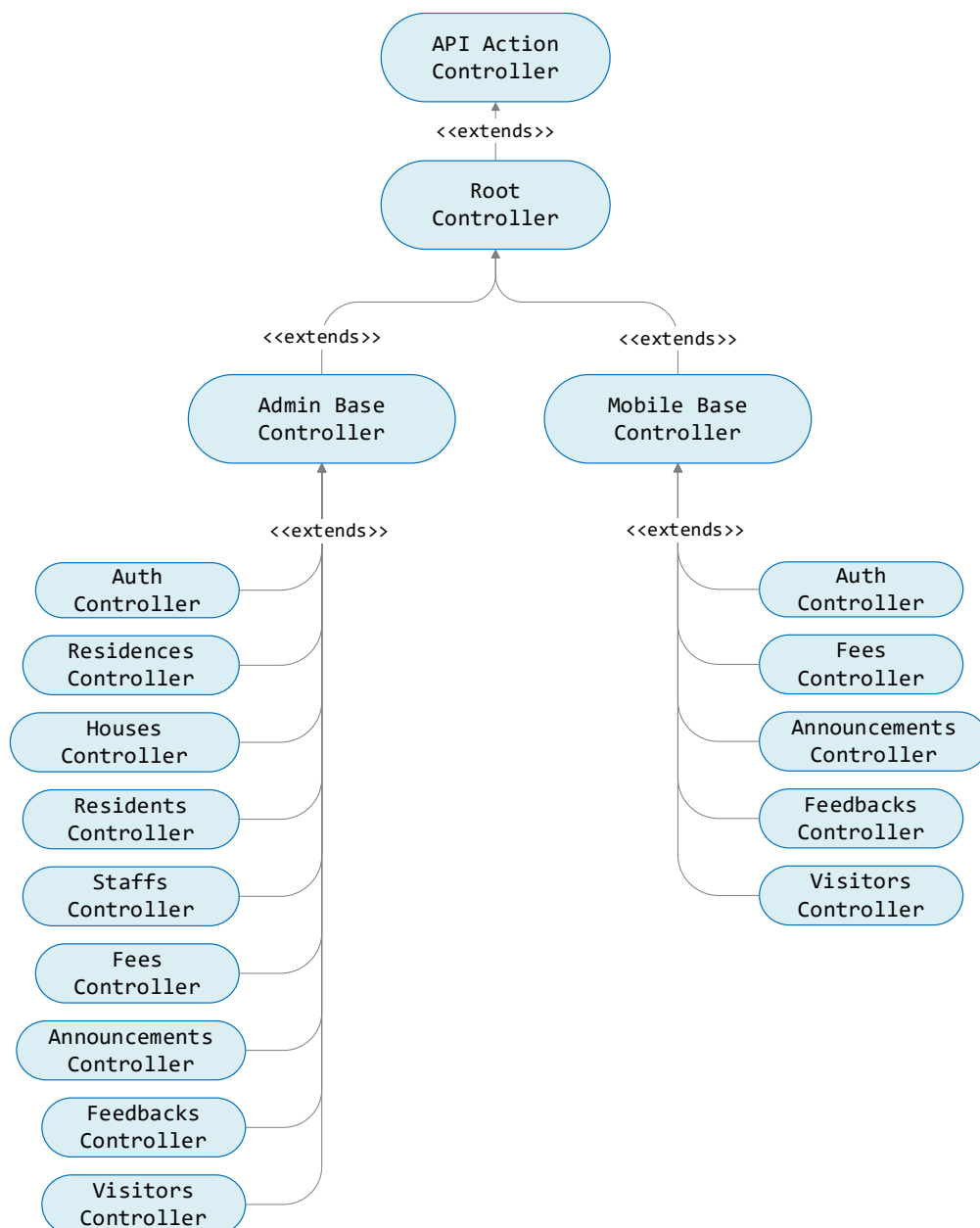


Figure 6.3 Hierarchy of Controller Inheritance

In the hierarchy of controller inheritance, multiple level of inheritance is involved. The highest level of super class, “API Action Controller” is Ruby on Rails built-in API controller. It provides only the essential feature, such as “render JSON” and “redirects” that required to build API endpoints. “Root Controller” will inherit the behaviour of “API Action Controller” in second level and act as the parent controller of “Admin Base Controller” and “Mobile Base Controller” to provide the interchangeable function needed for the child controller of both base controllers.

```

require "error"
require "ext/params"
require "ext/string"
require "ext/shout"
require "ext/true_class"
require "ext/false_class"
require 'jwt'
require 'rqrqcode'
require 'aws-sdk-s3'
require "tempfile"
require 'securerandom'

class RootController < ActionController::API

  rescue_from Error::Params,           :with => :parameters_exception
  rescue_from Error::Database,        :with => :database_exception
  rescue_from Error::Residence,       :with => :residence_exception
  rescue_from StandardError,          :with => :standard_error_handle

  before_action :configuration

  def configuration
    BCrypt::Engine.cost = 5
  end

  def render_json(response, status = :ok)
    return render status: status, json: response
  end

end

```

Figure 6.4 Root Controller Source Code

In “Root Controller”, the code statement to load general libraries, customized extension, customized error handling and basic configuration will reside here. Customized error handling can provide more details and full control of how the error will be present and respond.

```
class Admin::BaseController < RootController
  include AuthResidence
  include AuthStaff
end
```

```
class Mobile::BaseController < RootController
  include AuthResidence
  include AuthResident
end
```

Figure 6.5 Admin and Mobile Base Controller Source Code

As the figure above shown, “Admin Base Controller” and “Mobile Base Controller” had “Root Controller” as parent class. In each base controller, several authentication concerns had been included. Concerns is a core functionality in Ruby on Rails that create mix-ins for the class to use. “Auth Residence” that included will automatically check which residence the user’s access token belongs to and inject the residence data into the request’s parameters for controller use. “Auth Staff” and “Auth Resident” will verify the user’s access token and inject staff data or resident data into request’s parameters.

```
module AuthResidence extend ActiveSupport::Concern
  included do
    include AccessToken
    before_action :auth_residence

    def auth_residence
      residence = Residence.where(id: params[:residence_id]).first
      raise Error::Residence if residence.nil?
      params[:_residence] = residence
    end
  end
end
```

Figure 6.6 Auth Residence Concern Source Code

Using example of “Auth Residence”, figure 6.6 show how the residence being verify and data inject into request’s parameters. “Auth Resident” and “Auth Staff” had similar mechanism to perform authentication and data injection.

Apart from that, controller in this Ruby on Rails backend server normally have general function, which is revolve around the list, retrieve, create, update and delete resource operations. For example, fees controller will have list fees, get fee, create fee, update fee and delete fee function while announcements controller will have list announcements, get announcement, create announcement, update announcement and delete announcements function. Fees controller will be used as an example to describe the structure of controller.

```
class Admin::FeesController < Admin::BaseController
  def list_fees
  end

  def get_fee
  end

  def create_fee
  end

  def update_fee
  end

  def delete_fee
  end

  private
  def map_fee fee
  end
end
```

Figure 6.7 Admin Fees Controller Collapsed Source Code

The figure 6.7 above show the available function of “Admin Fees Controller”, which is revolve around list fees, get fee, create fee, update fee and delete fee. Using “Admin Fees Controller” as example, details of each function will be described.



```

def list_fees

  residence = params[:_residence]

  fees = residence.fees.map { |fee| map_fee(fee) }

  return render_json({
    data: {
      success: true,
      fees: fees
    }
  })
end

```

Figure 6.8 List fees function from Admin Fees Controller

Remapping of the fee attributes will be performed to control what attributes will be returned in list fees function.

```

def get_fee

  residence = params[:_residence]

  permitted = params.require_and_permit([
    :fee_id
  ], [])

  fee = residence.fees.where(id: permitted[:fee_id]).first

  return render_json({
    data: {
      success: false,
      error: "Invalid fee id",
    }
  }) if fee.nil?

  return render_json({
    data: {
      success: true,
      fee: map_fee(fee)
    }
  })
end

```

Figure 6.9 Get fee function from Admin Fees Controller

In get fee function, the fee will be found by the primary key of fee which is “fee\_id” provided, early error return exists to handle the resource not found situation. Same as list fees function, remapping of fee resource will be performed before return.

```

def create_fee

  permitted = params.require_and_permit([
    :title,
    :amount,
    :due_at
  ], [
    :description,
    :charge_to_numbering_1,
    :charge_to_numbering_2,
    :charge_to_numbering_3
  ])

  residence = params[:_residence]

  begin

    fee = residence.fees.create({
      title: permitted[:title],
      description: permitted[:description],
      amount: permitted[:amount],
      ...
    })

    target_houses = residence.houses

    #Filter houses

    registration_tokens = []

    target_houses.each{ |house|
      house.house_fees.create({fee_id: fee.id})
      registration_tokens << house.resident.registration_token if !house.resident.nil?
    }

    Resident.sendNotification({
      title: fee_notification_title,
      body: fee_notification_body,
      registration_tokens: registration_tokens
    })

    return render_json({
      data: {
        success: true,
        fee: fee,
      }
    })

  rescue => e
    return render_json({
      data: {
        success: false,
        type: "ActiveRecord::Rollback"
        error: e,
      }
    })
  end

end
end

```

Figure 6.10 Create fee function from Admin Fees Controller

In create fee function, the request's parameters will be validate using the custom function "require\_and\_permit" defined in "params" class extension. It will filter the parameter's body and verify the required content exists and remove unpermitted content. After that, creation of fee and house fees pivot will be performed. Lastly, a notification will be sent to residents to notify them about the new fee created by using the residents' device registration token.

```

def update_fee

  permitted = params.require_and_permit([
    :fee_id,
    :title,
    :amount,
    :due_at
  ], [
    :description,
    :charge_to_numbering_1,
    :charge_to_numbering_2,
    :charge_to_numbering_3
  ])

  residence = params[:_residence]

  fee = residence.fees.where(id: permitted[:fee_id]).first

  return render_json({
    data: {
      success: false,
      error: "Invalid fee id",
    }
  }) if fee.nil?

  fee.update({
    title: !permitted[:title].blank? ? permitted[:title] : fee.title,
    description: !permitted[:description].blank? ? permitted[:description] : fee.description,
    amount: !permitted[:amount].blank? ? permitted[:amount] : fee.amount,
    due_at: !permitted[:due_at].blank? ? permitted[:due_at] : fee.due_at,
    ...
  })

  return render_json({
    data: {
      success: true
    }
  })
end

```

Figure 6.11 Update fee function from Admin Fees Controller

In update fee function, same as get fee function, the specific fee will be found using its primary key. After the specific fee is found, update operation will be performed only to the attributes given at parameters. For example, if only description parameters are given in the request's parameters, only the description attribute will be updated.

```
def delete_fee

  residence = params[:_residence]

  permitted = params.require_and_permit([
    :fee_ids
  ], [])

  fee_ids = permitted[:fee_ids].split(",")

  fee_ids.each do |fee_id|

    fee = residence.fees.where(id: fee_id).first

    return render_json({
      data: {
        success: false,
        error: "Invalid fee id",
      }
    }) if fee.nil?

    fee.soft_delete

  end

  return render_json({
    data: {
      success: true
    }
  })

end
```

Figure 6.12 Delete fee function from Admin Fees Controller

In delete fee function, fee ID will be passed in and use to find the specific fee to perform delete operation. In addition, the delete fee function also support bulk delete. If a list of fee ID, such as “1,3,4” is passed in, three fee record will be found and soft delete. Soft delete is a custom defined function, which perform operation to update the “is deleted” attributes of fee to true and “deletant” attributes to current timestamp.

```

class Mobile::FeesController < Mobile::BaseController
  def list_fees
  end

  def get_fee
  end

  def pay_fee
  end
end

```

Figure 6.13 Mobile Fees Controller Collapsed Source Code

Mobile Fees Controller had two similar function to Admin Fees Controller which is list fees and get fee. However, there are no create fee, update fee and delete function in Mobile Fees Controller because the 3 operations stated is not the behaviours of this community association mobile application's end user, which is resident. There is one additional function in Mobile Fees Controller, pay fee function. Pay fee function will handle the fee payments related operation, such as validate the total amount of fees, communicate with Stripe Payment Method API and Stripe Intent API, etc.

```

Stripe.api_key = ENV["STRIPE_API_KEY"]
method = Stripe::PaymentMethod.create(method_params)
intent = Stripe::PaymentIntent.create(intent_params)

resident.transactions.create(transaction_params)

```

Figure 6.14 Part of the pay fee function source code from  
Mobile Fees Controller

Stripe Ruby Library is used here to provide easy access to Stripe API. Before performing any action, Stripe account secret key need to be configured into the Stripe Ruby library. Firstly, a new payment method will be created using resident's credit/debit card information and billing address provided. After a payment method successfully created, payment intent will be created to charge the total amount of fees from resident credit/debit card. Lastly, a successful transaction record will be created into database's transaction table or vice versa with failure reason.

### 6.1.3 Model Layer

Model layer responsible for storing the resource data that will be involved in the system. Active Record, a core functionality of Ruby on Rails will play the role of Model in the MVC architecture. It will serve as the Object Relational Mapping (ORM) technique to connect tables in a relational database with the object in system. Active Record as ORM act as the link bridge between database and object in system to provide a convenience approach to retrieve or update the database without writing SQL statements. In addition, relationship like has-many, has-one or belongs-to between multiple model will defined in the Active Record class. The figure below is an overview of how the hierarchy of model class inheritance in Ruby on Rails backend server.

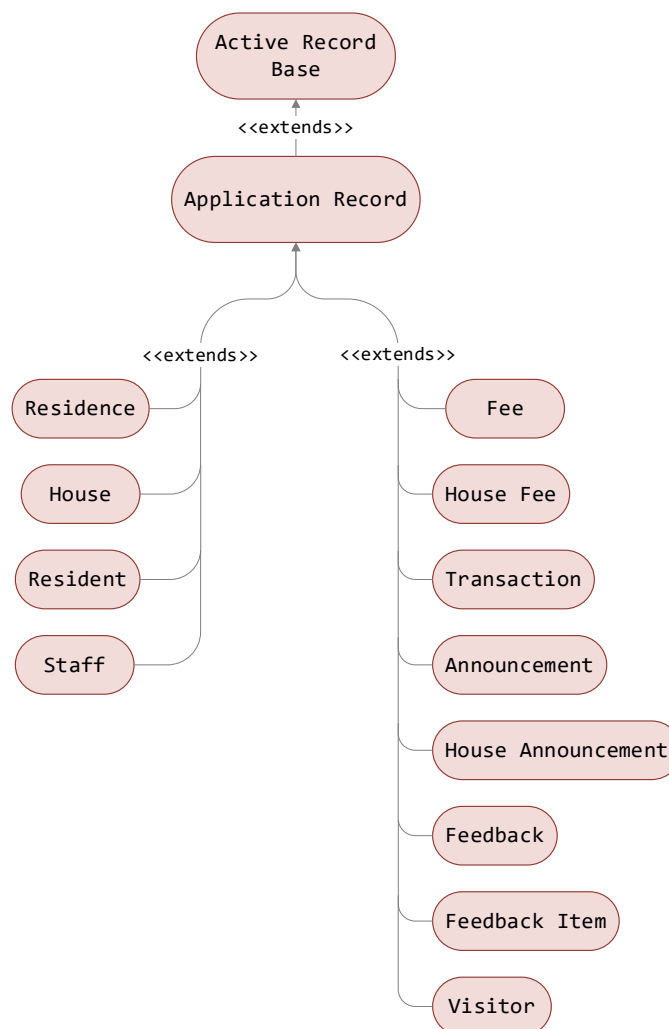


Figure 6.15 Hierarchy of Model Inheritance

In the hierarchy of model inheritance, multiple level of inheritance is involved. From top to bottom, the first and second level of super class, “Active Record Base” and “Application Record” are Ruby on Rails built-in base class to using Active Record, it provides basic function such as creation. All the custom defined Active Record model class like “Residence”, “Fee”, Announcement” will inherit “Application Record” class as parent class. Several custom defined class will be used to explain how the relationship between Model be create and manage.

```
class Residence < ApplicationRecord
  include Deletable

  has_many :staffs,      :class_name => 'Staff'
  has_many :residents,  :class_name => 'Resident'

  has_many :houses,     :class_name => 'House'
  has_many :fees,       :class_name => 'Fee'
  has_many :announcements, :class_name => 'Announcement'
  has_many :feedbacks,  :class_name => 'Feedback'
  has_many :visitors,   :class_name => 'Visitor'
end
```

Figure 6.16 Resident Active Record Class Source Code

The first example using will be Residence class. Residence class is the centralized class in this system as it had relationships with all other models. As previously described, Residence class will have “Application Record” class as parent class to enable the usage of Active Record. There is one custom defined concern included in Residence class which is “Deletable” concern that will inject several delete related function such as “soft\_delete” and “restore” into Residence class. Apart from that, there are several has-many relationship defined in Residence class. In Active Record class, has-many relationship indicates a one-to-many connection to another model. For example, residence has many residents, and residence has many announcements.

```

class Feedback < ApplicationRecord
  include Deletable

  belongs_to :residence, :class_name => 'Residence'
  belongs_to :resident, :class_name => 'Resident', foreign_key:'resident_id'

  has_many :items, :class_name => 'FeedbackItem'
end

```

Figure 6.17 Feedback Active Record Class Source Code

For second example, Feedback class is used to store the data of the residents' feedback case submitted. So, as the figure shown, feedback instance will be belonging to residence and resident. In additions, it will have many items which is used to store each update from either management staffs or resident.

```

class FeedbackItem < ApplicationRecord
  include Deletable

  has_many :images, :class_name => 'Image', as: :imageable
  belongs_to :feedback, :class_name => 'Feedback'
  belongs_to :authorable, :polymorphic => true
end

```

Figure 6.18 Feedback Item Active Record Class Source Code

Polymorphism relationship is implemented in Feedback Item class, as it can belongs to either Staff class or Resident class. Both of the class mentioned will act as the authorable class to the Feedback Item class. Polymorphism is a method in which multiple classes may be used to implement the same function. By doing so, when calling the authorable methods in Feedback Item instance, it may return either Staff instance or Resident instance which determined who created the Feedback Item instance.



## 6.1.4 Special Integration

There are several integrations to be highlighted had integrated in to the Ruby on Rails backend server to achieve more advanced functionalities in this project. For examples, Stripe, Firebase Cloud Messaging, Sendinblue, RQRCode etc.

### 6.1.4.1 Stripe Payment

Stripe is payment processing platform that integrated to enable the ability to receive the residents' payment for the fee's charges that through online. In another word, through the integration of Stripe, resident can pay their fee charge using their debit or credit card in their mobile application installed.

```
Stripe.api_key = ENV['STRIPE_API_KEY']

method = Stripe::PaymentMethod.create({
  type: 'card',
  card: card_params,
  billing_details: billing_details_params
})

intent = Stripe::PaymentIntent.create({
  amount: amount,
  description: fees_title,
  currency: 'myr',
  payment_method_types: ['card'],
  payment_method: method[:id],
  confirm: true
})
```

Figure 6.19 Fee Payment using Stripe

First and foremost, a unique Stripe API key is required to set before any actions of Stripe API client. After the key is set, a payment method needs to be created before the payment charge. The payment method is created using the card information including card number, expiry month, expiry year, CVV/CVC, card holder name passed in and the billing details including billing address, phone and email. After a payment method is created, the ID of the method will then pass in in to the Stripe's Payment Intent create function. A Stripe Payment Intent is an encapsulated detail that contain the details of the payment methods, amount to collect and the currency to charge. Besides of the payment method's ID, amount to collect, description of payment, and currency using will also pass in for the creation of payment intent. After all, if a payment intent is successfully created, the fee payment amount will be deducted from the residents' debit or credit card. The whole payment process will be complete.

### 6.1.4.2 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is integrated to enable the ability to send real time mobile notification to the residents' devices. Notification will be sent to the resident, whenever there was a new fee, new announcement, feedback case update or visitor application status update, etc.

```
def self.sendNotification envelope

  fcm = FCM.new(ENV['FCM_API_KEY'])

  options = {
    notification: {
      title:  envelope[:title],
      body:   envelope[:body],
    },
    data: {
      click_action: "FLUTTER_NOTIFICATION_CLICK",
      sound: "default",
      status: "done",
    },
    android: {
      ttl: 3600 * 1000,
      notification: {
        color: '#000000',
        sound: 'default'
      },
    },
  }

  fcm.send(envelope[:registration_tokens], options)

end
```

Figure 6.20 FCM Send Notification Function

As the figure shown, a send notification function is defined at Resident class. The function will be received an envelope that contained the notification title, body text and the user devices' registration tokens as parameter. After that, each respective value will be mapped into the format required. At last, the notification is sent through the FCM API client instance created.

```
Resident.sendNotification({
  title: "[Notice] #{announcement.title}",
  body: "#{Nokogiri::HTML(announcement.content).text[0..10]}...",
  registration_tokens: registration_tokens
})
```

Figure 6.21 Send Notification when Announcement Creation

For example, the figure above shown the code segment to send notification to residents when a new announcement was made.

### 6.1.4.3 Sendinblue SMTP Email

Sendinblue is an email marketing service provider. It is integrated in this project to provide the ability to send email to the resident during the process of recovering forgotten password and send invitation email to resident when his/her account is ready to be use.

```
class NeiborMailer < ApplicationMailer
  default from: 'admin@neibor.com'

  def send_to envelope

    # envelope mapping

    sender = Sendinblue::Mailin.new(
      "https://api.sendinblue.com/v2.0",
      ENV['SENDINBLUE_API_KEY'],
      10
    )

    payload = {
      to: {"#{email}" => "#{name}"},
      from: ["admin@neibor.com", "Neibor"],
      subject: subject,
      html: html_content
    }

    sender.send_email(payload)

  end
end
```

Figure 6.22 Neibor Mailer with Sendinblue

The figure above is the Neibor Mailer class defined. It consists of one function that responsible to send email to resident by using the Sendinblue API client.

```
html_content = Nokogiri::HTML(File.open("storage/templates/invitation.html"))
html_content.at_css('[id="email"]').content = resident.email
html_content.at_css('[id="password"]').content = permitted[:password]

NeiborMailer.send_to({
  email: resident.email,
  name: resident.first_name,
  subject: "Your Neibor Account is ready!",
  html_content: html_content.to_s
}).deliver_now
```

Figure 6.23 Send Invitation Email to Resident

The figure above shown the code segment to send the invitation email to resident by using the Neibor Mailer defined.

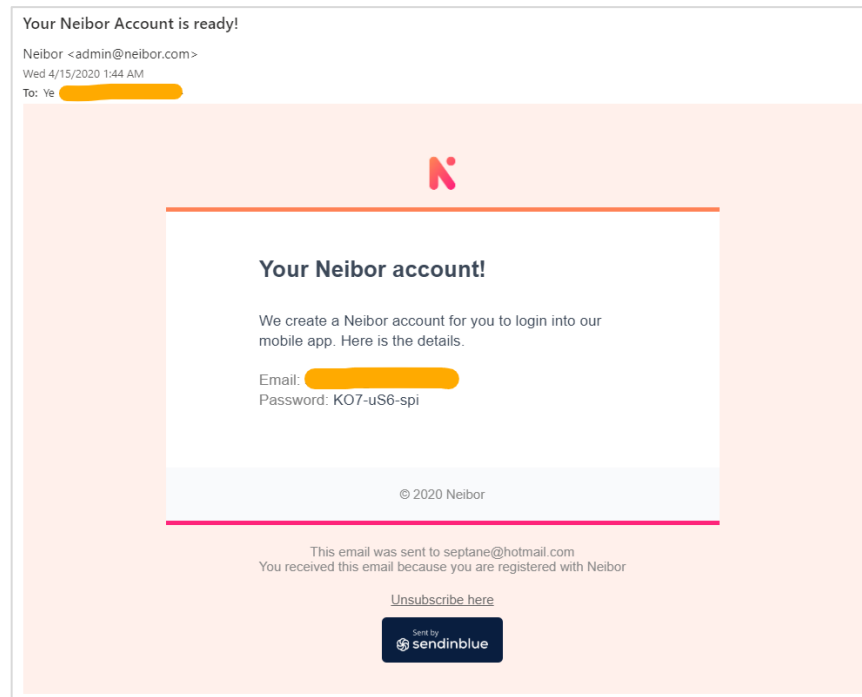


Figure 6.24 Invitation email

The figure above is the invitation email. It will be sent to residents when their account is ready to be used.

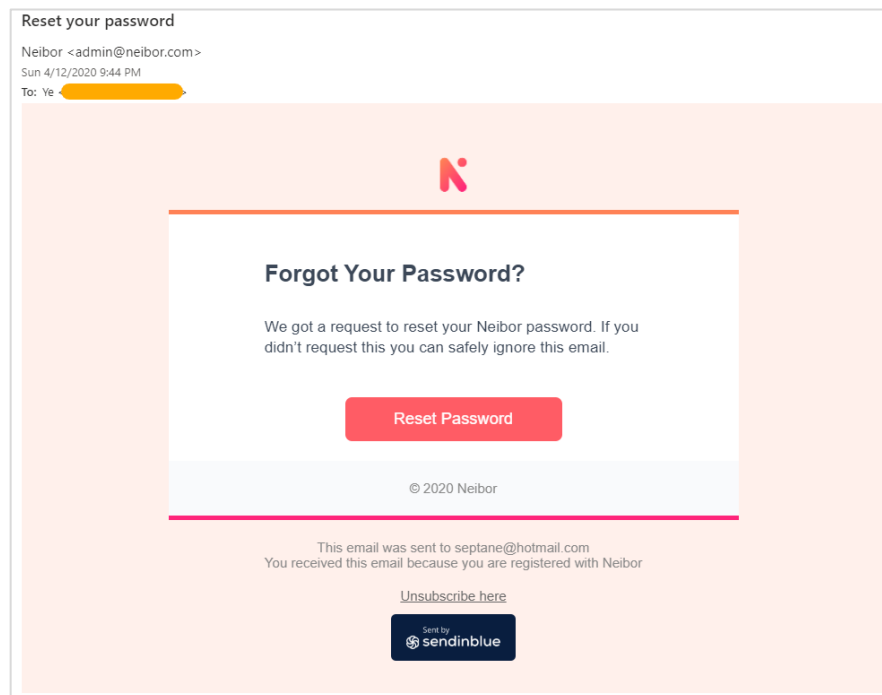


Figure 6.25 Password recovery email

The figure above is the password recovery email. It will be sent to user when they wished to recover their forgotten password.

#### 6.1.4.4 Amazon Simple Storage Service

Amazon Simple Storage Service or known as AWS S3 is integrated into this project to provide the ability to store the image or user data on cloud. For examples, the images uploaded in the feedback process and QR code images generated in the visitor application approve process will be store at the AWS S3 bucket.

```
s3 = Aws::S3::Resource.new({
  region: "ap-southeast-1",
  credentials: Aws::Credentials.new(ENV['AWS_API_KEY'])
})

bucket = s3.bucket('bt-neibor')

images_url = images_json.map.with_index { |image_json, index|

  file = Tempfile.new("#{image_json["name"]}.jpg")
  file.binmode
  file.write Base64.decode64(image_json["base64_image"])
  file.rewind

  destination = bucket.object("feedback/#{feedback.id}/#{image_json["name"]}.jpg")
  destination.upload_file(File.open(file), {
    acl: "public-read",
    content_type: "image/jpg"
  })

  file.close
  destination.public_url

}

images_url.each { |image_url|
  feedback_item.images.create({
    url: image_url
  })
}
```

Figure 6.26 Code Segment of Feedback's Images Upload

The code segment above is located at the image upload part of the feedback process. Initially, a S3 resource object is created using the AWS API key and region the bucket located at. After that, the specific bucket object which is the destination of the image upload will be created also. Temporary image files will be created for the upload purpose. Those image files will then upload to the specific bucket directory path defined. After the upload process completed, the URL where the image located will be return and store into database for the later retrieve.

#### 6.1.4.5 RQRCode QR Code Generator

RQRCode is a Ruby library that use to convert data into QR code. It is integrated into this project to generate QR code for the visitor entry access into residential area. The QR code will be generated when a visitor application is approved.

```
qrcode = RQRCode::QRCode.new(visitor_access_code)

svg = qrcode.as_svg(
  offset: 0,
  color: '262842',
  shape_rendering: 'crispEdges',
  module_size: 6,
  standalone: true
)

bucket = s3.bucket('bt-neibor')
destination = bucket.object("visitor-qr/#{visitor_id}")

file = Tempfile.new('svg-qr')
file.write(svg)
file.close

destination.upload_file(File.open(file), {
  acl: "public-read",
  content_type: 'image/svg+xml'
})
```

Figure 6.27 QR Code Generation when Visitor Application Approved

The figure above shown the code segment of the QR code generation in visitor admin controller. Firstly, the visitor access code generated in used to convert into the QR code image. The QR code image will then uploaded into AWS S3 bucket, so residents can retrieve the QR code image in their mobile devices. In their mobile devices, residents can screenshot the QR code image and send it to their friend, which is the visitor. Visitor can present the QR code image to the security personnel of the residential area to grant entry access. Security personnel can use their mobile devices to scan the QR code and retrieve the visitor information. When the visitor wished to exit the residential area, he/she will need to show the QR code to the security personnel again.

### 6.1.5 Available Endpoints

There are total 70 endpoints available in this Ruby on Rails API server, which can be split into 2 major parts which are “Admin” and “Mobile”.

#### 6.1.5.1 Admin Endpoints

Admin endpoints will be responsible to serve the request from the admin panel that is used by residential community management. For admin endpoints, it consists of 51 endpoints. All the admin endpoints will have a prefix which is “`{{api_url}}/admin/residences/1`” except for the residence-related routes.

Table 6.1 Admin Endpoints Listing

Authentication			
No.	Method	Route	Description
1	POST	auth	To verify the access token and persist the login state of admin.
2	POST	auth/login	To verify the email and password in login process.
3	POST	auth/recover_password	To initiate the recover forgotten password process, send email to admin.
4	POST	auth/verify_token	To verify the recovery token for the password recovery link.
5	POST	auth/reset_password	To reset the password of user.
Residence			
No.	Method	Route	Description
6	GET	residence/:residence_id	Retrieve the residence details
7	PUT	residence/:residence_id	Update the residence details
Staff			
No.	Method	Route	Description
8	GET	staffs	List all the staffs
9	POST	staffs	Create new staff
10	GET	staffs/:staff_id	Retrieve specific staff
11	PUT	staffs/:staff_id	Update specific staff

12	DELETE	staffs/:staff_id	Delete specific staff
Resident			
No.	Method	Route	Description
13	GET	residents	List all the residents
14	POST	residents	Create new resident
15	GET	residents/:resident_id	Retrieve specific resident
16	PUT	residents/:resident_id	Update specific resident
17	PUT	residents/:resident_id/remove_house	Remove house from specific resident
18	DELETE	residents/:resident_id	Delete specific resident
House			
No.	Method	Route	Description
19	GET	houses	List all the houses
20	POST	houses	Create new house
21	GET	houses/:house_id	Retrieve specific house
22	PUT	houses/:house_id	Update specific house
23	PUT	houses/:house_id/remove_resident	Remove resident from specific house
24	DELETE	houses/:house_id	Delete specific house
25	POST	houses/bulk-create	Bulk create houses
Fee			
No.	Method	Route	Description
26	GET	fees	List all the fees
27	POST	fees	Create new fee
28	GET	fees/:fee_id	Retrieve specific fee
29	PUT	fees/:fee_id	Update specific fee
30	DELETE	fees/:fee_ids	Delete specific fee(s)
Payment history			
No.	Method	Route	Description
31	GET	payment_histories	List all the payment histories
32	GET	payment_histories/:payment_history_id	Retrieve specific payment histories
Announcement			



No.	Method	Route	Description
33	GET	announcements	List all the announcements
34	POST	announcements	Create new announcement
35	GET	announcements/:announcement_id	Retrieve specific announcement
36	PUT	announcements/:announcement_id	Update specific announcement
37	DELETE	announcements/:announcement_ids	Delete specific announcement(s)
<b>Feedback</b>			
No.	Method	Route	Description
38	GET	feedbacks	List all the feedbacks
39	POST	feedbacks	Create new feedback
40	GET	feedbacks/:feedback_id	Retrieve specific feedback
41	PUT	feedbacks/:feedback_id	Update specific feedback progress
42	PUT	feedbacks/:feedback_id/complete	Complete specific feedback
<b>Visitor</b>			
No.	Method	Route	Description
43	GET	visitors	List all the visitors
44	POST	visitors	Create new visitor
45	GET	visitors/:visitor_id	Retrieve specific visitor
46	PUT	visitors/:visitor_id	Approve or reject specific visitor
<b>Analytic</b>			
No.	Method	Route	Description
47	GET	analytics/fee	Retrieve specific fee payment analytics
48	GET	analytics/transaction	Retrieve overall transaction payment analytics
49	GET	analytics/announcement	Retrieve overall announcement analytics

50	GET	analytics/feedback	Retrieve overall feedback analytics
51	GET	analytics/visitor	Retrieve overall visitor analytics

### 6.1.5.2 Mobile Endpoints

Mobile endpoints will responsible to serve the request from mobile application that used by resident and security personnel. For mobile endpoints, it consists of 19 endpoints. All the admin endpoints will have a prefix which is “*{{api\_url}}/mobile/residences/1*”

Table 6.2 Mobile Endpoints Listing

Auth			
No.	Method	Route	Description
1	POST	auth	To verify the access token and persist the login state of mobile user.
2	POST	auth/login	To verify the email and password in login process. To record the device registration token for notification.
3	POST	auth/logout	To delete the device registration token from database.
4	POST	auth/recover_password	To initiate the recover forgotten password process, send email to mobile user.

Profile			
No.	Method	Route	Description
5	PUT	profile	Update the mobile user profile
6	PUT	password	Change password
Fee			
No.	Method	Route	Description
7	GET	fees	List all the fees available, transaction history of the specific resident
8	POST	fees	Make payment for the fee by resident
Announcement			
No.	Method	Route	Description
9	GET	announcements	List all the announcements available for the specific residents
10	PUT	announcements	Update the read status of the announcement, bookmarked or unbookmarked the announcement
Feedback			
No.	Method	Route	Description
11	GET	feedbacks	List all the feedbacks
12	POST	feedbacks	Create new feedback
13	PUT	feedbacks	Update feedback
14	PUT	feedbacks	Complete feedback, give rating or review
Visitor			
No.	Method	Route	Description
15	GET	visitors	List all the visitors
16	POST	visitors	Create new visitor

Guard			
No.	Method	Route	Description
17	GET	guard/visitors/access	Retrieve the details information of specific visitor
18	PUT	guard/visitors/:visitor_id/enter	Record specific visitor enter time
19	PUT	guard/visitors/:visitor_id/exit	Record specific visitor exit time

### 6.1.6 Deployment

The Ruby on Rails API server is deployed to AWS Elastic Beanstalk environment, at AWS EC2 server instance. The platform of the EC2 instance using is Puma with Ruby 2.6 on 64bit Amazon Linux 2.11.4. The API server URL is “http://neibor-api-dev.ap-southeast-1.elasticbeanstalk.com”

The screenshot displays the AWS Elastic Beanstalk console for the environment 'NeiborApi-env'. The environment is in a healthy state, indicated by a green checkmark icon and the text 'Ok'. The running version is 'app-dfc7-200412\_213238'. The platform is 'Puma with Ruby 2.6 running on 64bit Amazon Linux/2.11.4'. The console also shows a list of recent events, including application updates and deployments.

Time	Type	Details
2020-04-12 21:34:47 UTC+0800	INFO	Environment health has transitioned from Info to Ok. Application update completed 55 seconds ago and took 62 seconds.
2020-04-12 21:33:49 UTC+0800	INFO	Environment update completed successfully.
2020-04-12 21:33:49 UTC+0800	INFO	New application version was deployed to running EC2 instances.

Figure 6.28 Deployment of Ruby on Rails API server

## 6.2 Web Application for Community Management

The web application built with Vue.js, will be acted as the admin panel for the community management to perform the daily operation to manage residents. This Vue.js web application is one of the front-end clients of the overall system. As single page application nature of Vue.js, the web application is rendering in a main layout and the content of the main layout will be replaced by new data directly without refreshing the whole web page.

### 6.2.1 Overview of Web Application

In this project, a “Main.vue” is created to utilize the single page feature and the content of the web application will be resided in “Main.vue”. For examples, as the figure below shown. The “Main.vue” consists of navigation bar area and content area. The content wrapper will be used to display each page’s content while the navigation bar area will be static and always displayed.

The navigation area allows community management user to navigate within the navigation bar’s items defined. The content area will be use to display content like fees page, announcement page, create feedback page, etc. Majority of the users’ activities will be taking place within the content area.

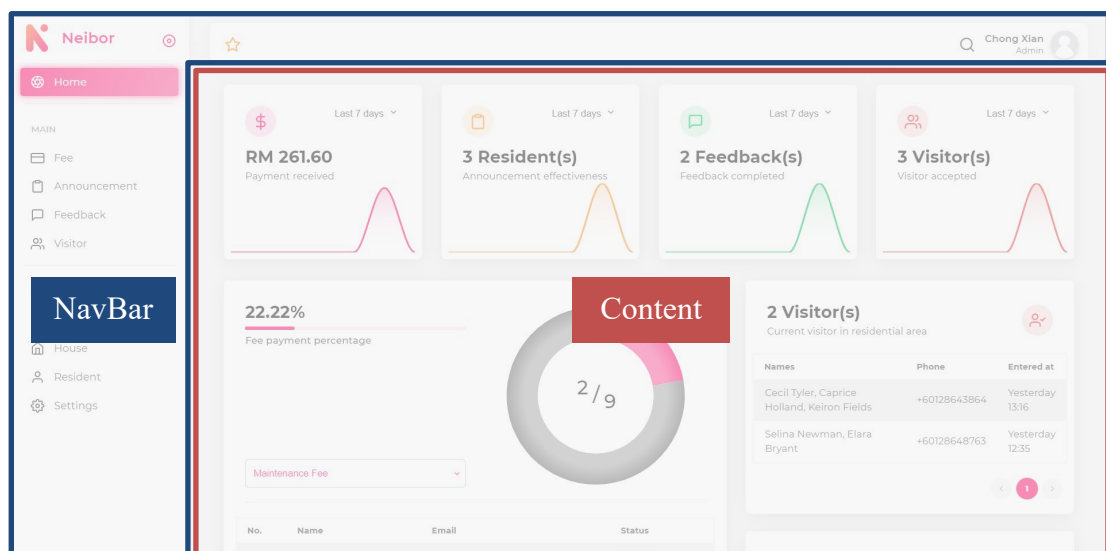


Figure 6.29 Web application main layout

```

<template>
  <div...
  >
    <v-nav-menu :navMenuItems="navMenuItems" title="Neibor"/>

    <div id="content-area">...
  </div>
</div>
</template>

```

Figure 6.30 Collapsed code of Main.vue

As the figure above shown, Main.vue consists of two main components which is “v-nav-menu” and a division with ID of “content-area”. The “v-nav-menu” will used the “navMenuItems” array of menu items’ information passed in to render out the buttons in the left navigation bar.

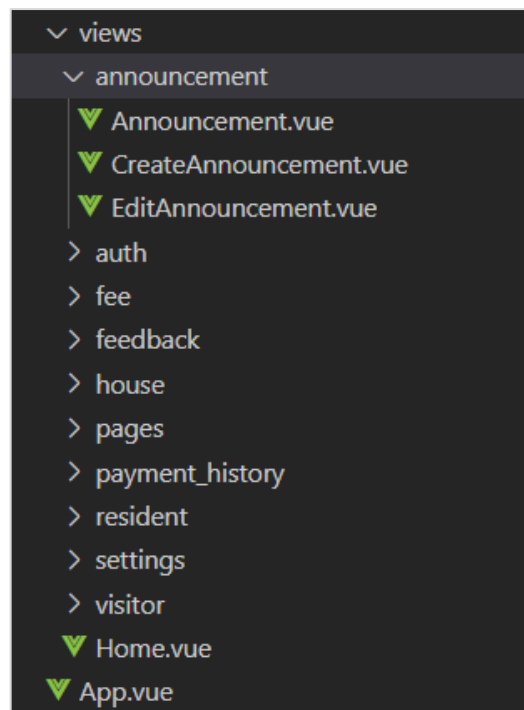


Figure 6.31 Views folder for content area

The views folder consists of 10 mains folder which contains the .vue file to be rendered inside the content area. For example, announcement folders contained “Announcement.vue” used to render the announcements in table view, “CreateAnnouncement.vue” will be used to render the create announcement page.

## 6.2.2 Pages Hierarchy

This section will describe the web application's pages hierarchy.

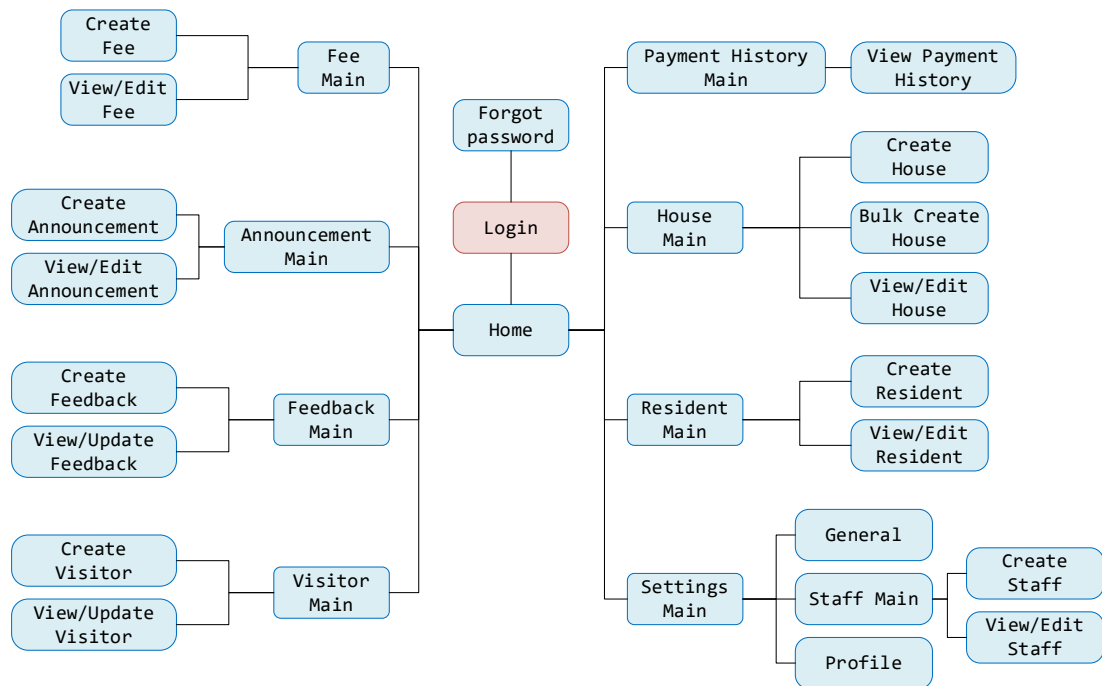


Figure 6.32 Web application pages hierarchy

The entry page of the web application is the Login Page. After login, it will enter the Home Page, which will display the analytics of the residence information. In Home Page, user can navigate to 8 screens, including the Fee Main Page, Announcement Main Page, Feedback Main Page, Visitor Main Page, Payment History Main page, House Main Page, Resident Main Page and Settings Main Page. Majority of the main page can be navigated to respective resource operations page, such as create and view. For example, Fee Main Page can navigate to Create Fee Page and View/Edit Fee Page.

### 6.2.3 API Client

An API client is created to be communicate with the Ruby on Rails API server. The API client is built based on Axios, a promise-based HTTP client for the browser and server. It supports multiple HTTP method such as GET, POST, PUT, DELETE.

```
apiGet(url, params, config) { this.uiHandler('get', url, params, config) },
apiPost(url, params, config) { this.uiHandler('post', url, params, config) },
apiPut(url, params, config) { this.uiHandler('put', url, params, config) },
apiDelete(url, params, config) { this.uiHandler('delete', url, params, config) },
```

Figure 6.33 Code segment of ApiClient.js

The figure above shown a part of the code of the ApiClient.js. There are 4 methods defined to handle the 4 types of HTTP request. The uiHandler method will be called to map the API client format to the real Axios method. The API client will auto get the API server access token and add it into headers of HTTP request. Besides, it also handles and map parts of the URL path passed in into the full API server URL path.

```
this.apiGet(
  '/analytics/feedback?duration=${duration}',
  params,
  {
    success: res => {
      // Success handler
    },
    error: err => {
      // Error handler
    },
    finally: () => {
      // Finally handler
    }
  }
)
```

Figure 6.34 Code segment of ApiClient.js usage

The figure above shown a part of code in the Home.vue, the “apiGet” function defined in ApiClient.js is called. In the function, the first parameter passed in is the API URL path, the second parameter is the params of the HTTP request, and the last parameters is the handlers object.



## 6.2.4 Deployment

The Vue.js web application is deployed to Heroku web hosting platform. It enabled with the continuous deployment process. Whenever the Git upstream had new commit, it will automatically use the latest push of the master branch to deploy a new version of web application. The deployed web application can be accessed at: <http://neibor-admin.herokuapp.com/>.

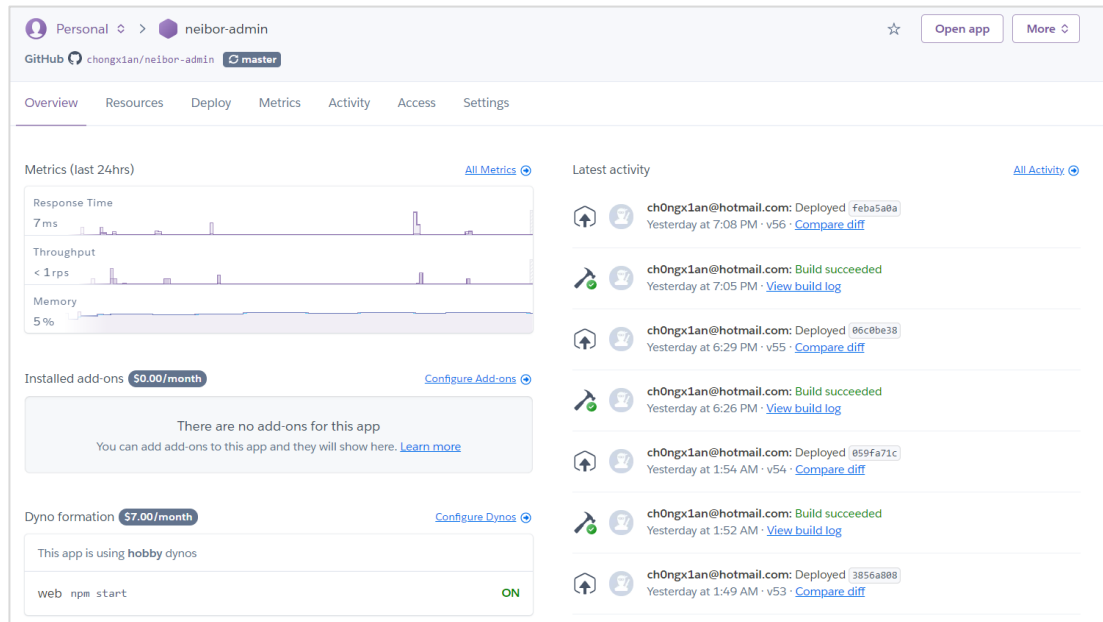


Figure 6.35 Deployment of Vue.js Web Application

### 6.3 Mobile Application for Community Residents and Security Personnel

The mobile application built with Flutter, is one of the front-end clients of the overall system. The target end user of this mobile application will be community residents and security personnel.

#### 6.3.1 Overview of Mobile Application

In this mobile application, major screen can be divided into two portions which are content area and bottom navigation bar area. The bottom navigation bar is applied to ease the navigation from screen to screen. The bottom navigation bar will have four items which can navigate to Fee Main Screen, Announcement (Notice) Main Screen, Feedback Main Screen and Visitor Main Screen.

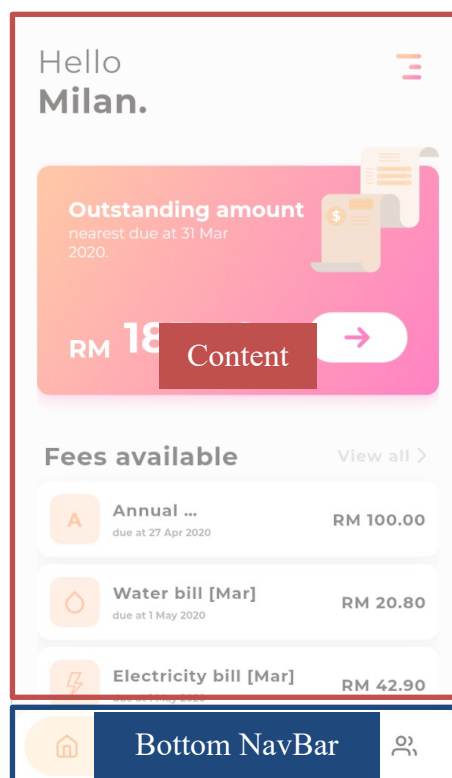


Figure 6.36 Mobile application main layout

The figure above shows the screen that will be displayed after a resident is logged in. If no account is logged in, the user will be directed to the login screen. This process will be handled by a component defined as "AuthWrapper.dart". Inside AuthWrapper.dart, it consists of a switch statement on the AuthStatus of the mobile application, based on the value of AuthStatus, different screens will be returned.

```

switch (_userProvider.authStatus) {
  case AuthStatus.notSignedIn:
    return LoginScreen();
    break;

  case AuthStatus.residentSignedIn:
    return NavigationScreen();
    break;

  case AuthStatus.guardSignedIn:
    return GuardHomeScreen();
    break;

  default:
    return Scaffold(
      backgroundColor: Colors.white,
      body: Center(
        Logo()
      ),
    );
    break;
}

```

Figure 6.37 Code segment of AuthWrapper.dart

As the figure shown, if AuthStatus is not signed in, login screen will be return and displayed to users. If residents were signed in, the navigations screen which the resident main screen will be returned. Besides, if guards were signed in, the guard home screen will be returned. Lastly, if none of the stated condition met, a default splash screen consist of the mobile application logo will be return.

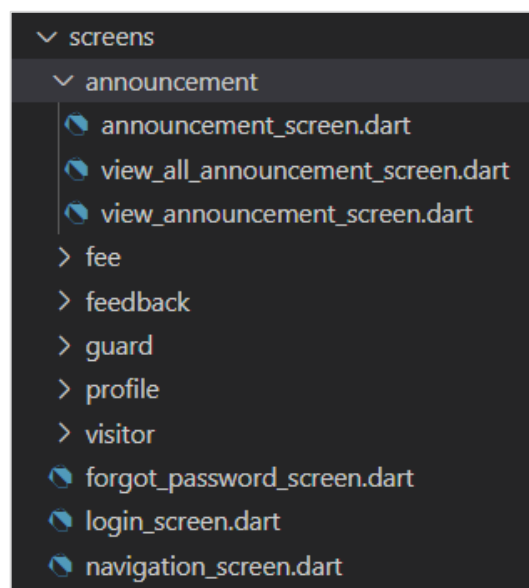


Figure 6.38 Views folder of mobile application

The views folder of mobile application consists of 6 mains folder which contains the .dart file to be rendered inside the content area.

### 6.3.2 Screen Hierarchy

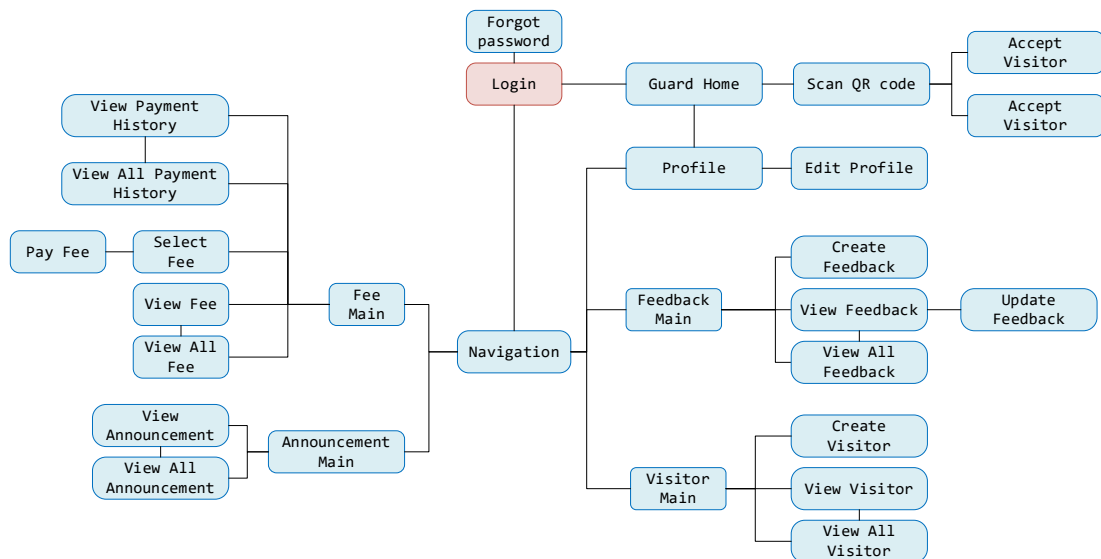


Figure 6.39 Screen hierarchy of mobile application

The entry screen of the mobile application is the Login Screen. After resident's login, it will be route to the Main Navigation Screen which consists of 4 tabs, fee, announcement (notice), feedback and visitors. In Fee Main Screen tab, fee details and payment history details will be displayed. Besides, residents can navigate to select fee screen to pay their fees. At Announcement Main Screen tab, residents can select a specific announcement to view its content. At Feedback Main Screen tab, residents can create new feedback case, view specific feedback, or view all feedbacks. At Visitor Main Screen tab, residents can create new visitor application, view specific visitor, or view all visitors. Apart from that, if there was a security personnel account signed in, the Guard Home Screen will be displayed instead of Main Navigation Screen.

### 6.3.3 State Management

Flutter is a declarative framework, each of the widgets will have their own state which containing the data used to render. If the application is simple, multiple states of several widgets might be still handleable to developer. However, when the application grows along with the functionalities implemented, the large amount of states of widgets might be a hard nut to crack. The application will become more complex and unable to maintain. With the built-in state management of Flutter like combination of Stateful widget and “setState” function, numerous redundant codes will be produced.

Provider State Management, which is one of the recommend state management approaches in the official documentation of Flutter is implemented in this project to solve the state management issues stated previously. Provider State Management consists of three main components which is:

- ChangeNotifier
- ChangeNotifierProvider
- Consumer

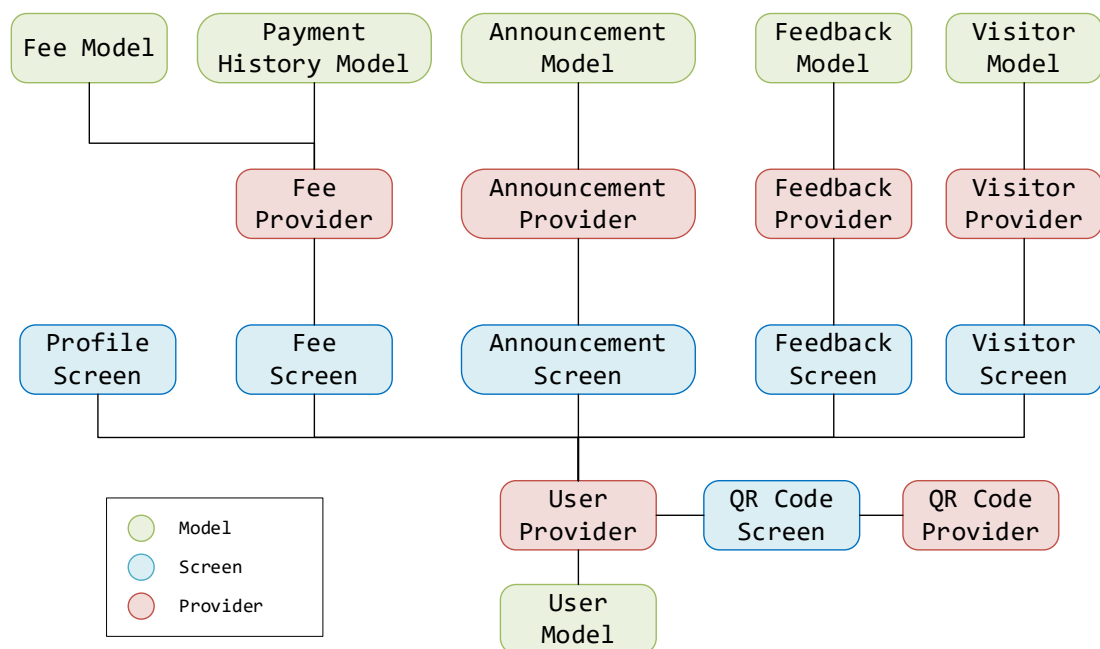


Figure 6.40 Overview of Provider State Management

The figure above shown the overview design of Provider State Management of this mobile application. Six providers are created to handle the states of multiple

screens. The providers here will act like the controller in the Model View Controller software pattern which implemented in the Ruby on Rails API server of this project. Provider will handle the request from screens, communicate with the API server and manipulate the model data with state.

Fee related state management will be used to described the implementation of Provider State Management in this project. ChangeNotifier is a simple class included in the Flutter SDK which notifies its listeners about changes. In provider, the application state can encapsulate by ChangeNotifier. The ChangeNotifier's child provider will act as the centralized state for the screens. For example, the figure below shows the FeeProvider. The state of fee-related screens will be access and modify by using the FeeProvider defined.

```
class FeeProvider extends ChangeNotifier {
  // Attributes
  List<dynamic> feesList;
  List<dynamic> paymentHistoriesList;
  ///...

  // Constructor
  FeeProvider() {
    this.feesList = [];
    this.paymentHistoriesList = [];
  }

  // Methods
  Future<void> getFees() async {}
  ///...
}
```

Figure 6.41 Code segment of Fee Provider

ChangeNotifierProvider is the intermediate widget that provides an instance of a ChangeNotifier to its descendants, which normally is screens. As the FeeProvider is defined, the connection between the FeeProvider and screens can be constructed by ChangeNotifierProvider widget. All the screens wrap in the child of ChangeNotifierProvider widget will have access to the Provider instance. For example, figure below, ChangeNotifierProvider of FeeProvider type is defined. In additions, MultiProvider is used to wrap multiple ChangeNotifierProvider widgets into the application.

```

void main() async {
  runApp(
    MultiProvider(
      providers: [
        // ChangeNotifierProviders
        ChangeNotifierProvider<UserProvider>(
          create: (context) => UserProvider(),

        ChangeNotifierProxyProvider<UserProvider, FeeProvider>(
          create: (_) => FeeProvider(),
          update: (_, user, fee) => fee..update(user),
        ),
        //...
      ]
    ),
    child: App()
  );
}

```

Figure 6.42 Code segment of implementation of ChangeNotifierProvider

As the states of FeeProvider is provided to the widgets in screens, FeeScreen will be the consumer in this case. FeeScreen will make use of the FeeProvider's states and render the user interface.

```

class _FeeScreenState extends State<FeeScreen> {
  @override
  Widget build(BuildContext context) {
    // Providers
    UserProvider _userProvider = Provider.of<UserProvider>(context);
    FeeProvider _feeProvider = Provider.of<FeeProvider>(context);

    // Methods
    void _viewFee(fee) async {
      Navigator.pushNamed(context, '/fee/view', arguments: fee);
    }

    // Render
    return Scaffold(
      //...
    );
  }
}

```

Figure 6.43 Code segment of FeeScreen

By using “Provider.of”, provider will be instantiated to the screen local state and accessible by the widgets.

### 6.3.4 Deployment

The Flutter mobile application is deployed by using the flutter command. The “flutter build apk” command will generate a release “.apk” file for installation.

```
C:\flutter_project\neibor-mobile>flutter build apk
Running Gradle task 'assembleRelease'... Done           109.8s
✓ Built build\app\outputs\apk\release\app-release.apk (23.7MB).
```



## CHAPTER 7

### SYSTEM TESTING

#### 7.1 Introduction

This chapter will discuss about the various type of testing involved in the project, such as unit testing and integration testing. Automated testing is applied in this project to validate the requirement specification are achieved. Several Ruby libraries was integrated to perform the testing of this project. Firstly, a testing library, RSpec was used to develop the unit testing and integration testing. Besides, a data mock-up library, Faker was used to generate sample data in the testing. Unmoderated remote usability testing was also conducted with 10 participants through online to gain user insight on the mobile application developed.

#### 7.2 Unit Testing

Unit testing will be focused on the Ruby on Rails API server as it contains most of the application logic and request handling from clients. In this Model View Controller software pattern, the unit testing will be performed as controller-basis. The whole unit testing will be separate into two major modules, which is “Admin” and “Mobile”. The “Admin” parts will be responsible to test the controller of “Admin” module while “Mobile” parts will be responsible to test the controller of “Mobile” module.

##### 7.2.1 Admin Module

In “Admin” module, there are total 51 test cases of unit testing involved.

Table 7.1 Test Cases for Admin Module

Auth Controller				
TC ID	Test case description	Parameters	Expected output	Result
1	Authentication for persisting the login state	<ul style="list-style-type: none"> <li>Access token</li> </ul>	It should respond with 200 success status code and the user data.	Pass
2	Login process	<ul style="list-style-type: none"> <li>Email</li> <li>Password</li> </ul>	It should respond with 200 success status code and the user data.	Pass

3	Initial recover password process	<ul style="list-style-type: none"> <li>Email</li> </ul>	It should respond with 200 success status code and send recovery email.	Pass
4	Verify recovery token	<ul style="list-style-type: none"> <li>Recovery token</li> </ul>	It should respond with 200 success status code and the user email.	Pass
5	Reset the forgotten password	<ul style="list-style-type: none"> <li>Email</li> <li>Password</li> <li>Recovery token</li> </ul>	It should respond with 200 success status code.	Pass
<b>Residence Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result
6	Retrieve the residence details	<ul style="list-style-type: none"> <li>Residence ID</li> </ul>	It should respond with 200 success status code and the residence data.	Pass
7	Update the residence details	<ul style="list-style-type: none"> <li>Name</li> <li>Description</li> <li>Structure</li> <li>Numbering name 1</li> <li>Numbering name 2</li> <li>Numbering name 3</li> <li>Numbering name format</li> <li>Address postfix</li> </ul>	It should respond with 200 success status code and the residence data.	Pass
<b>Staff Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result
8	List all the staffs		It should respond with 200 success status code and array of staffs' data.	Pass
9	Create new staff	<ul style="list-style-type: none"> <li>Email</li> <li>Password</li> <li>Role</li> <li>First name</li> <li>Last name</li> </ul>	It should respond with 200 success status code and the staff data.	Pass
10	Retrieve specific staff	<ul style="list-style-type: none"> <li>Staff ID</li> </ul>	It should respond with 200 success status code and the staff data.	Pass

11	Update specific staff	<ul style="list-style-type: none"> <li>• Staff ID</li> <li>• Password</li> <li>• Role</li> <li>• First name</li> <li>• Last name</li> <li>• Phone</li> <li>• Is enabled</li> </ul>	It should respond with 200 success status code and the staff data.	Pass
12	Delete specific staff	<ul style="list-style-type: none"> <li>• Staff ID</li> </ul>	It should respond with 200 success status code.	Pass
<b>Resident Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result
13	List all the residents		It should respond with 200 success status code and array of residents' data and their house data.	Pass
14	Create new resident	<ul style="list-style-type: none"> <li>• Email</li> <li>• Password</li> <li>• First name</li> <li>• Last name</li> <li>• House ID</li> </ul>	It should respond with 200 success status code, the resident data and his/her house data if applicable.	Pass
15	Retrieve specific resident	<ul style="list-style-type: none"> <li>• Resident ID</li> </ul>	It should respond with 200 success status code, the resident data and his/her house data if applicable.	Pass
16	Update specific resident	<ul style="list-style-type: none"> <li>• Resident ID</li> <li>• House ID</li> </ul>	It should respond with 200 success status code, the resident data and his/her house data if applicable.	Pass
17	Remove house from specific resident	<ul style="list-style-type: none"> <li>• Resident ID</li> <li>• House ID</li> </ul>	It should respond with 200 success status code and resident data.	Pass
18	Delete specific resident	<ul style="list-style-type: none"> <li>• Resident ID</li> </ul>	It should respond with 200 success status code.	Pass
<b>House Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result

19	List all the houses		It should respond with 200 success status code and array of houses' data and their resident data.	Pass
20	Create new house	<ul style="list-style-type: none"> <li>• House numbering 1</li> <li>• House numbering 2</li> <li>• House numbering 3</li> <li>• Resident ID</li> </ul>	It should respond with 200 success status code, the house data and its resident data if applicable.	Pass
21	Retrieve specific house	<ul style="list-style-type: none"> <li>• House ID</li> </ul>	It should respond with 200 success status code, the house data and its resident data if applicable.	Pass
22	Update specific house	<ul style="list-style-type: none"> <li>• House ID</li> <li>• Resident ID</li> </ul>	It should respond with 200 success status code, the house data and its resident data if applicable.	Pass
23	Remove resident from specific house	<ul style="list-style-type: none"> <li>• House ID</li> <li>• Resident ID</li> </ul>	It should respond with 200 success status code and house data.	Pass
24	Delete specific house	<ul style="list-style-type: none"> <li>• House ID</li> </ul>	It should respond with 200 success status code.	Pass
25	Bulk create houses	<ul style="list-style-type: none"> <li>• Number of house numbering 1</li> <li>• Number of house numbering 2</li> <li>• Number of house numbering 3</li> </ul>	It should respond with 200 success status code.	Pass
<b>Fee Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result
26	List all the fees		It should respond with 200 success status code and array of fees data.	Pass

27	Create new fee	<ul style="list-style-type: none"> <li>Title</li> <li>Amount</li> <li>Due at</li> <li>Description</li> <li>Charge to numbering 1</li> <li>Charge to numbering 2</li> <li>Charge to numbering 3</li> </ul>	It should respond with 200 success status code and the fee data.	Pass
28	Retrieve specific fee	<ul style="list-style-type: none"> <li>Fee ID</li> </ul>	It should respond with 200 success status code and the fee data.	Pass
29	Update specific fee	<ul style="list-style-type: none"> <li>Fee ID</li> <li>Title</li> <li>Amount</li> <li>Due at</li> <li>Description</li> <li>Charge to numbering 1</li> <li>Charge to numbering 2</li> <li>Charge to numbering 3</li> </ul>	It should respond with 200 success status code and the fee data.	Pass
30	Delete specific fee(s)	<ul style="list-style-type: none"> <li>Fee IDs</li> </ul>	It should respond with 200 success status code.	Pass
<b>Payment History Controller</b>				
<b>TC ID</b>	<b>Test case description</b>	<b>Parameters</b>	<b>Expected output</b>	<b>Result</b>
31	List all the payment histories		It should respond with 200 success status code and array of payment histories data.	Pass
32	Retrieve specific payment histories	<ul style="list-style-type: none"> <li>Payment history ID</li> </ul>	It should respond with 200 success status code and the payment history data.	Pass
<b>Announcement Controller</b>				
<b>TC ID</b>	<b>Test case description</b>	<b>Parameters</b>	<b>Expected output</b>	<b>Result</b>
33	List all the announcements		It should respond with 200 success status code and array of announcements data.	Pass

34	Create new announcement	<ul style="list-style-type: none"> <li>Title</li> <li>Content</li> <li>Send to numbering 1</li> <li>Send to numbering 2</li> <li>Send to numbering 3</li> <li>Tags</li> <li>Is published</li> </ul>	It should respond with 200 success status code and the announcement data.	Pass
35	Retrieve specific announcement	<ul style="list-style-type: none"> <li>Announcement ID</li> </ul>	It should respond with 200 success status code and the announcement data.	Pass
36	Update specific announcement	<ul style="list-style-type: none"> <li>Announcement ID</li> <li>Title</li> <li>Content</li> <li>Tags</li> <li>Is published</li> </ul>	It should respond with 200 success status code and the announcement data.	Pass
37	Delete specific announcement(s)	<ul style="list-style-type: none"> <li>Announcement IDs</li> </ul>	It should respond with 200 success status code.	Pass
<b>Feedbacks Controller</b>				
<b>TC ID</b>	<b>Test case description</b>	<b>Parameters</b>	<b>Expected output</b>	<b>Result</b>
38	List all the feedbacks		It should respond with 200 success status code and array of feedbacks data.	Pass
39	Create new feedback	<ul style="list-style-type: none"> <li>Title</li> <li>Type</li> <li>Description</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass
40	Retrieve specific feedback	<ul style="list-style-type: none"> <li>Feedback ID</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass
41	Update specific feedback progress	<ul style="list-style-type: none"> <li>Feedback ID</li> <li>Title</li> <li>Description</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass
42	Complete specific feedback	<ul style="list-style-type: none"> <li>Feedback ID</li> </ul>	It should respond with 200 success status code and the completed feedback data.	Pass

Visitor Controller				
TC ID	Test case description	Parameters	Expected output	Result
43	List all the visitors		It should respond with 200 success status code and array of visitors' data.	Pass
44	Create new visitor	<ul style="list-style-type: none"> <li>• Names list</li> <li>• Representative phone</li> <li>• Representative car plate</li> <li>• Visitation purpose</li> <li>• Resident ID</li> </ul>	It should respond with 200 success status code and the visitor data.	Pass
45	Retrieve specific visitor	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code and the visitor data.	Pass
46	Approve or reject specific visitor	<ul style="list-style-type: none"> <li>• Visitor ID</li> <li>• Status</li> <li>• Rejected reason</li> </ul>	It should respond with 200 success status code and the visitor data.	Pass
Analytic Controller				
TC ID	Test case description	Parameters	Expected output	Result
47	Retrieve specific fee payment analytics	<ul style="list-style-type: none"> <li>• Fee ID</li> </ul>	It should respond with 200 success status code and analytics result of fee payment, including the percentage of paid resident, percentage of unpaid resident and the resident involved.	Pass
48	Retrieve overall transaction payment analytics	<ul style="list-style-type: none"> <li>• Duration</li> </ul>	It should respond with 200 success status code and overall analytics result of transaction payment including total amount of transactions payment received in certain period and the daily received amount in certain period.	Pass
49	Retrieve overall announcement analytics	<ul style="list-style-type: none"> <li>• Duration</li> </ul>	It should respond with 200 success status code and overall analytics result of	Pass

			announcement, including number of residents read announcements, number of residents haven't read announcement and the number of residents read announcement each day in certain period.	
50	Retrieve overall feedback analytics	<ul style="list-style-type: none"> <li>• Duration</li> </ul>	It should respond with 200 success status code and overall analytics result of feedback, including the number of incomplete feedback case, number of completed feedback case and number of completed feedback case each day in certain period.	Pass
51	Retrieve overall visitor analytics	<ul style="list-style-type: none"> <li>• Duration</li> </ul>	It should respond with 200 success status code and overall analytics result of visitor, including the number of accepted visitors, number of reject visitor, number of accepted visitors each day in certain period and total number of visitors inside the residential area currently.	Pass



All the admin module's test cases will be performed using RSpec library. One example, which is "Auth process" test case will be described in below about the implementation of the automated unit test.

```
@email = Faker::Internet.email
@password = Faker::Internet.password

@staff = @residence.staffs.create(
  email: @email,
  password_digest: BCrypt::Password.create(@password),
  role: 'admin'
)
```

Figure 7.1 Code segment to generate mock staff account

The mock data will be generated using Faker library. For examples, the figure 7.1 shown the staff's email and password was generated using the Faker's Internet module.

```
post :auth, params: {
  use_route: "residences/#{@residence.id}/admin/auth",
  token: @staff.token
}

it "should returns status code 200 and staff" do
  result = JSON.parse(response.body, :symbolize_names => true)

  flunk "internal server error" if result[:data].nil?
  success = result[:data][:success]
  staff = result[:data][:staff]

  expect(response).to have_http_status(:success)
  expect(success).to match(true)
  expect(staff).to include_json({
    "email": @staff.email,
    "id": @staff.id,
    # Staff data to assert
  })
end
```

Figure 7.2 Code segment to assert the response

After the mock data generated, the Auth Controller's function will be called using the input data and the response will be assert and validate the result returned. If the status code and data of test result returned was match with the expected output, the "Auth process" unit test will be pass.

```

C:\rails_project\neibor-api>rspec ./spec/unit/admin
.....
Finished in 30.61 seconds (files took 16.91 seconds to load)
51 examples, 0 failures

```

Figure 7.3 Automated unit tests performed

Total of 51 unit tests in the admin module will be performed using the command in the figure above. As the figure shown, the 51 unit tests used about 30 second to finish.

### 7.2.2 Mobile Module

In “Mobile” module, there are total 19 test cases of unit testing involved.

Table 7.2 Test Cases for Admin Module

Auth Controller				
TC ID	Test case description	Parameters	Expected output	Result
1	Authentication for persisting the login state	<ul style="list-style-type: none"> <li>Access token</li> </ul>	It should respond with 200 success status code and the user data.	Pass
2	Login process	<ul style="list-style-type: none"> <li>Email</li> <li>Password</li> </ul>	It should respond with 200 success status code and the user data.	Pass
3	Logout process		It should respond with 200 success status code and delete the registration token.	Pass
4	Initial recover password process	<ul style="list-style-type: none"> <li>Email</li> </ul>	It should respond with 200 success status code and send recovery email.	Pass
Resident Controller				
TC ID	Test case description	Parameters	Expected output	Result
5	Update the mobile user profile	<ul style="list-style-type: none"> <li>First name</li> <li>Last name</li> <li>Phone</li> <li>Car plate</li> </ul>	It should respond with 200 success status code and the user data.	Pass

6	Change password	<ul style="list-style-type: none"> <li>• Password</li> </ul>	It should respond with 200 success status code and the user data.	Pass
Fee Controller				
TC ID	Test case description	Parameters	Expected output	Result
7	List all the fees available, transaction history of the specific resident		It should respond with 200 success status code, array of fees available and array of transaction histories.	Pass
8	Make payment for the fees	<ul style="list-style-type: none"> <li>• Fee IDs</li> <li>• Amount</li> <li>• Card details</li> <li>• Billing details</li> </ul>	It should respond with 200 success status code.	Pass
Announcement Controller				
TC ID	Test case description	Parameters	Expected output	Result
9	List all the announcements available for the specific residents		It should respond with 200 success status code and array of announcements data.	Pass
10	Update the read status of the announcement, bookmarked or unbookmarked the announcement	<ul style="list-style-type: none"> <li>• Announcement ID</li> <li>• Is read</li> <li>• Is bookmarked</li> </ul>	It should respond with 200 success status code and the announcement data.	Pass
Feedback Controller				
TC ID	Test case description	Parameters	Expected output	Result
11	List all the feedbacks		It should respond with 200 success status code and array of feedbacks data.	Pass
12	Create new feedback	<ul style="list-style-type: none"> <li>• Title</li> <li>• Description</li> <li>• Type</li> <li>• Images</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass
13	Update feedback	<ul style="list-style-type: none"> <li>• Feedback ID</li> <li>• Title</li> <li>• Description</li> <li>• Images</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass

14	Complete feedback, give rating or review	<ul style="list-style-type: none"> <li>• Feedback ID</li> <li>• Rating</li> <li>• Review</li> </ul>	It should respond with 200 success status code and the feedback data.	Pass
<b>Visitor Controller</b>				
TC ID	Test case description	Parameters	Expected output	Result
15	List all the visitors		It should respond with 200 success status code and array of visitors' data.	Pass
16	Create new visitor	<ul style="list-style-type: none"> <li>• Names list</li> <li>• Representative phone</li> <li>• Representative car plate</li> <li>• Visitation purpose</li> </ul>	It should respond with 200 success status code and the visitor data.	Pass
17	Retrieve the details information of specific visitor	<ul style="list-style-type: none"> <li>• Visitor access code</li> </ul>	It should respond with 200 success status code and array of visitor data.	Pass
18	Record specific visitor enter time	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code and the visitor data.	Pass
19	Record specific visitor exit time	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code and array of visitors' data.	Pass

All the mobile module's test cases will be performed using RSpec library. One example, which is "List all fees available" will be described in below about the implementation of the automated unit test.

```
@fee = @residence.fees.create({
  title: Faker::Commerce.product_name,
  description: Faker::Commerce.product_name,
  amount: Faker::Number.decimal(l_digits: 2),
  due_at: Faker::Date.forward(days: 14),
  charge_to_numbering_1: '0',
  charge_to_numbering_2: '0',
  charge_to_numbering_3: '0'
})
```

Figure 7.4 Code segment to generate mock fee

The mock data will be generated using Faker library. For examples, the figure 7.4 shown the fee's title, description, amount and due date was generated using the Faker's Commerce module, Faker's Number module and Faker's Date module.

```
get :list_fees, params: {
  use_route: "residences/#{@residence.id}/fees",
  residence_id: @residence.id
}

it "should returns status code 200 and fees" do
  result = JSON.parse(response.body, :symbolize_names => true)

  flunk "internal server error" if result[:data].nil?
  success = result[:data][:success]
  fees = result[:data][:fees]

  expect(response).to have_http_status(:success)
  expect(success).to match(true)
  expect(fees).to include_json([
    {
      "id": @fee.id,
      "title": @fee.title,
      "amount": @fee.amount.to_s,
      # Data to assert
    }
  ])
end
```

Figure 7.5 Code segment to assert the response

After the mock data generated, the Fee Controller's function will be called using the input data and the response will be assert and validate the result returned. If

the status code and data of test result returned was match with the expected output, the “List all fees available” unit test will be pass.

```
C:\rails_project\neibor-api>rspec ./spec/unit/mobile
.....
Finished in 20.62 seconds (files took 18.09 seconds to load)
19 examples, 0 failures
```

Figure 7.6 Automated unit tests performed

Total of 19 unit tests in the mobile module will be performed using the command in the figure above. As the figure shown, the 19 unit tests used about 20 second to finish.

### 7.3 Integration Testing

As like unit testing, the integration testing in this project was implemented using the RSpec library and Faker library also. There are total 7 integration test suites developed. Each integration test suite will contain multiple unit test case stated previously and serve as the purpose to ensure that the integration points between components unit are exercised and validated.

Table 7.3 Test Suites for Integration Testing

Staff test suite				
Test suite ID: 1				
Step no.	Step description	Parameters	Expected output	Result
1	Admin create new staff	<ul style="list-style-type: none"> <li>• Email</li> <li>• Password</li> <li>• Role</li> </ul>	It should respond with 200 success status code and staff account data.	Pass
2	Staff use account created to login	<ul style="list-style-type: none"> <li>• Email</li> <li>• Password</li> </ul>	It should respond with 200 success status code and the staff account data.	Pass

3	Staff use token to persist login state	<ul style="list-style-type: none"> <li>• Token</li> </ul>	It should respond with 200 success status code and the staff account data.	Pass
House test suite				
Test suite ID: 2				
Step no.	Step description	Parameters	Expected output	Result
1	Admin create new house	<ul style="list-style-type: none"> <li>• House numbering 1</li> <li>• House numbering 2</li> <li>• House numbering 3</li> </ul>	It should respond with 200 success status code and house data.	Pass
2	Admin create new resident account	<ul style="list-style-type: none"> <li>• First name</li> <li>• Last name</li> <li>• Email</li> <li>• Password</li> </ul>	It should respond with 200 success status code and the resident account data.	Pass
3	Admin attach the house to the resident	<ul style="list-style-type: none"> <li>• Resident ID</li> <li>• House ID</li> </ul>	It should respond with 200 success status code, the resident account data and house data.	Pass
Fee test suite				
Test suite ID: 3				
Step no.	Step description	Parameters	Expected output	Result
1	Admin create new fee	<ul style="list-style-type: none"> <li>• Title</li> <li>• Amount</li> <li>• Due date</li> </ul>	It should respond with 200 success status code and fee data.	Pass
2	Resident pay fee	<ul style="list-style-type: none"> <li>• Fee IDs</li> <li>• Amount</li> <li>• Card details</li> <li>• Billing details</li> </ul>	It should respond with 200 success status code.	Pass
Announcement test suite				
Test suite ID: 4				
Step no.	Step description	Parameters	Expected output	Result

1	Admin create new announcement	<ul style="list-style-type: none"> <li>Title</li> <li>Content</li> </ul>	It should respond with 200 success status code and announcement data.	Pass
2	Resident read announcement	<ul style="list-style-type: none"> <li>Announcement ID</li> </ul>	It should respond with 200 success status code and announcement data.	Pass
3	Resident bookmark announcement	<ul style="list-style-type: none"> <li>Announcement ID</li> </ul>	It should respond with 200 success status code and announcement data.	Pass
Feedback test suite				
Test suite ID: 5				
Step no.	Step description	Parameters	Expected output	Result
1	Resident create new feedback	<ul style="list-style-type: none"> <li>Title</li> <li>Description</li> <li>Type</li> </ul>	It should respond with 200 success status code and feedback data.	Pass
2	Admin update feedback progress	<ul style="list-style-type: none"> <li>Feedback ID</li> <li>Title</li> <li>Description</li> </ul>	It should respond with 200 success status code and feedback data.	Pass
3	Resident update feedback progress	<ul style="list-style-type: none"> <li>Feedback ID</li> <li>Title</li> <li>Description</li> </ul>	It should respond with 200 success status code and feedback data.	Pass
4	Resident complete feedback	<ul style="list-style-type: none"> <li>Feedback ID</li> <li>Rating</li> <li>Review</li> </ul>	It should respond with 200 success status code and feedback data.	Pass
Visitor test suite				
Test suite ID: 6				
Step no.	Step description	Parameters	Expected output	Result
1	Resident create new visitor	<ul style="list-style-type: none"> <li>Names list</li> <li>Representative phone</li> <li>Representative car plate</li> <li>Visitation purpose</li> </ul>	It should respond with 200 success status code and visitor data.	Pass



2	Admin accept visitor	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code, visitor data and access code generated.	Pass
3	Guard verify the access code generated	<ul style="list-style-type: none"> <li>• Access code</li> </ul>	It should respond with 200 success status code and visitor data.	Pass
4	Visitor enter	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code and visitor data.	Pass
5	Visitor exit	<ul style="list-style-type: none"> <li>• Visitor ID</li> </ul>	It should respond with 200 success status code and visitor data.	Pass
Profile test suite				
Test suite ID: 7				
Step no.	Step description	Parameters	Expected output	Result
1	Admin create new resident account	<ul style="list-style-type: none"> <li>• First name</li> <li>• Last name</li> <li>• Email</li> <li>• Password</li> <li>• House ID</li> </ul>	It should respond with 200 success status code, resident data and house data.	Pass
2	Resident login	<ul style="list-style-type: none"> <li>• Email</li> <li>• Password</li> </ul>	It should respond with 200 success status code and resident data.	Pass
3	Resident update profile	<ul style="list-style-type: none"> <li>• First name</li> <li>• Last name</li> <li>• Phone</li> <li>• Car plate</li> </ul>	It should respond with 200 success status code and resident data.	Pass

Total of 7 integrations test suites will be performed. One integration test suite will be used to described the implementation of this integration testing using RSpec library and Faker library. The visitor test suites will be described in the section below.

```
@names = [Faker::Name.name, Faker::Name.name, Faker::Name.name].to_json
@representative_phone = Faker::PhoneNumber.phone_number_with_country_code[0..9]
@representative_car_plate = Faker::Vehicle.license_plate
@visitation_purpose = Faker::Lorem.sentence
```

Figure 7.7 Mock data of visitor

The figure above shown the code segment to generate the visitor data used to pass in to the controller function as parameters. As different from unit testing, in this integration testing, multiple function will be called and test their integrity.

```
post "/mobile/residences/#{@residence.id}/visitors", params: {
  residence_id: @residence.id,
  names: @names,
  representative_phone: @representative_phone,
  representative_car_plate: @representative_car_plate,
  visitation_purpose: @visitation_purpose
}, headers: {
  'Neibor-Access-Token': @resident_token
}

# Assert visitor data

put "/admin/residences/#{@residence.id}/visitors/#{@visitor.id}", params: {
  residence_id: @residence.id,
  visitor_id: @visitor.id,
  status: 1
}, headers: {
  'Neibor-Access-Token': @admin_token
}

# Assert visitor data

get "/mobile/residences/#{@residence.id}/guard/visitors/access?code=#{@visitor_code}", params: {
  residence_id: @residence.id
}, headers: {
  'Neibor-Access-Token': @guard_token
}

# Assert visitor data

put "/mobile/residences/#{@residence.id}/guard/visitors/#{@visitor.id}/enter", params: {
  residence_id: @residence.id,
  visitor_id: @visitor.id,
}, headers: {
  'Neibor-Access-Token': @guard_token
}

# Assert visitor data

put "/mobile/residences/#{@residence.id}/guard/visitors/#{@visitor.id}/exit", params: {
  residence_id: @residence.id,
  visitor_id: @visitor.id,
}, headers: {
  'Neibor-Access-Token': @guard_token
}

# Assert visitor data
```

Figure 7.8 Code segment of Visitor test suite

```

C:\rails_project\neibor-api>rspec ./spec/integration
.....
Finished in 1 minutes 28 seconds (files took 23.16 seconds to load)
7 examples, 0 failures

```

Figure 7.9 Integration testing performed

Total of 7 integration tests will be performed using the command in the figure above. As the figure shown, the 7 integration tests used about 1 minutes 28 seconds to finish.

#### 7.4 Usability Testing

In this project, unmoderated remote usability testing was conducted to gain better insight on the interaction between user and the system. Unmoderated remote usability testing, is a type of usability testing that participated tester complete a set of given tasks without the supervision of a moderator. It is suitable with the need of this project as it requires low cost, fast and accurate result. The unmoderated remote usability testing was conducted through online using web meeting tools like, Microsoft Teams and Skype based on participants' preference. The participant is instructed to perform a list of activities like the table below.

Table 7.4 Usability Testing Participant Checklist

Usability test				
No	Task	Description	Time used (second)	Comments
1	Login	Use the account details from email and login into mobile application		
2	View fee	Select any fee and view its details		
3	Pay fee	Select any fee(s) to pay through the card information given		

4	Bookmark announcement	Select any announcements and bookmark it		
5	Submit a feedback	Submit a new feedback with any title, any description and 1 image		
6	Update the feedback	Update the newly created feedback progress with any title, any description and 0 image		
7	Complete and review the feedback	Complete and rate the specific feedback		
8	Create new visitor	Create a new visitor application with 2 friends' names		
9	Update profile	Update your profile by editing your personal information like name or phone number		
10	Logout	Logout from the mobile application		

After the participants complete the task stated in the checklist, a post-study usability questionnaire adopted from “Usability Test Plan Prepared for : Better World Books” by Dubois and Purcell was conducted to gain more insight about the testing.

Table 7.5 Post-study Usability Questionnaire

No	Questions	Strongly disagree		Strongly agree		
		1	2	3	4	5
1	The system is intuitive and simple to use.					
2	I was able to complete the tasks in the checklist quickly using the system.					
3	I felt comfortable using the system.					

4	The system gave error message that clearly told me how to fix problems.					
5	I was able to find the information I needed easily.					
6	The user interface of this system is pleasant.					
7	I felt very confident using the system.					
8	I think that the system's functions were well integrated and organized.					
9	The system is consistent in term of logic and user interface.					
10	Overall, I am satisfied with the system.					

In this project, there were 10 participants involved in this usability testing. The overall result was gathered and analysed. The 2 tables below showed the average time usage for participant to complete each task and the average score of the post-study usability questionnaire.

Table 7.6 Average Time Usage of Task

Usability test result			
No	Task	Description	Average time usage
1	Login	Use the account details from email and login into mobile application	15.70s
2	View fee	Select any fee and view its details	2.92s
3	Pay fee	Select any fee(s) to pay through the card information given	34.89s
4	Bookmark announcement	Select any announcements and bookmark it	5.43s
5	Submit a feedback	Submit a new feedback with any title, any description and 1 image	16.70s
6	Update the feedback	Update the newly created feedback progress with any title, any description and 0 image	9.95s

7	Complete and review the feedback	Complete and rate the specific feedback	7.63s
8	Create new visitor	Create a new visitor application with 2 friends' names	17.77s
9	Update profile	Update your profile by editing your personal information like name or phone number	10.36s
10	Logout	Logout from the mobile application	3.90s

Table 7.7 Post-study usability questionnaire average score

No	Questions	Average score
1	The system is intuitive and simple to use.	4.1
2	I was able to complete the tasks in the checklist quickly using the system.	4.0
3	I felt comfortable using the system.	4.3
4	The system gave error message that clearly told me how to fix problems.	3.8
5	I was able to find the information I needed easily.	3.9
6	The user interface of this system is pleasant.	4.5
7	I felt very confident using the system.	4.4
8	I think that the system's functions were well integrated and organized.	4.0
9	The system is consistent in term of logic and user interface.	4.2
10	Overall, I am satisfied with the system.	4.5
	Total average score	4.17

Based on the result of the usability testing, some objective achievement can be proved here such as the time required of fee payment is definitely shorter than the traditional method as the average time of pay fee activities is 34.89 second. The post-study usability questionnaire had total average score of 4.17. This reflect that the users were generally satisfied with the system as it is easy to use, well organized user interface and integrated functionalities.

## CHAPTER 8

### CONCLUSION AND RECOMMENDATION

#### 8.1 Conclusion

This project, “A Mobile App for Community Association” had been designed and developed according to the project plan to tackle the problem stated in this project. Throughout the project execution, Kanban methodology was practiced and used to schedule the work task. As a result, a system including a Ruby on Rails API server, a Vue.js web application and a Flutter mobile application is developed and deployed. In additions, validation and testing are performed to ensure the developed system are met with requirement specification and achieved the objectives stated:

1. To identify the current problem of the existing community association management system.
2. To simplify the existing community association management operations by automating repeated tasks.
3. To develop a system that can used to manage the community association that build community loyalty and enhance the residential image.

The developed system will contribute to the community association in several aspects. Firstly, community management can save time and cost compared to the traditional approach which involving more handiwork. Besides, residents have more flexible payment method to pay their residential fees securely through online. Through the developed system, the announcement effectiveness had been increasing and didn't require any hard-copy paper printing. Furthermore, using the mobile application, resident can have an official channel to submit their feedback and track the status. Leakage of personal information can be prevented as the personal information of visitors and residents was collected through the system and without exposure to public.

However, the developed system still has some limitations. The system might have a learning curve for aged community management staff. They may need more time to learn and utilize the system. Others than that, residential management need

internet connection to use the system as the admin panel is a cloud-based web application. Security personnel need to equip with mobile device with camera to scan the QR code present by the visitor.

## **8.2 Future Implementation**

There are some future enhancements can be implemented to this project. Firstly, the system produced can be fine-tuned and further developed into a Software as a Service product. The system architecture and database design of this project are fully support with this enhancement. Despite the current system is only using by 1 residential area, after the system is turned into a Software as a Service product, it can be used by multiple residential area parallely.

Secondly, the current resident registration flow in this system can be improved in the future. A registration screen can be developed in the mobile application, and the resident account registration can be reviewed by the community management.

Lastly, the API server and web application deployed are currently using HTTP protocol. The both applications can integrate with an SSL certificate in the future and upgrade to HTTPS protocol to provide more secure data transmission.



## REFERENCES

Ahmad, M.O., Markkula, J. and Oivo, M., 2013. Kanban in software development: A systematic literature review. In: *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*. IEEE Computer Society. pp.9–16.

Alonso, J., 2006. *Security and Property Management System*. [online] Available at: <<https://patents.google.com/patent/US20060064305A1/en>>.

Alqudah, M. and Razali, R., 2018. A comparison of scrum and Kanban for identifying their selection factors. In: *Proceedings of the 2017 6th International Conference on Electrical Engineering and Informatics: Sustainable Society Through Digital Innovation, ICEEI 2017*. [online] Malaysia. pp.1–6. Available at: <<https://ieeexplore.ieee.org/document/8312434>> [Accessed 28 Jul. 2019].

Amazon Inc., 2018. *AWS Elastic Beanstalk – Deploy Web Applications*. [online] Available at: <<https://aws.amazon.com/elasticbeanstalk/>> [Accessed 19 Mar. 2020].

Anon 2020. *Comparison with Other Frameworks — Vue.js*. [online] Available at: <<https://vuejs.org/v2/guide/comparison.html>> [Accessed 19 Mar. 2020].

Benson, J., 2010. Personal Kanban : Optimizing the Individual Coder. In: *Øredev Conference*. [online] Malmo, Sweden. Available at: <<https://vimeo.com/16917928>>.

Dubois, A. and Purcell, K., n.d. Usability Test Plan Prepared for : Better World Books. [online] Available at: <[www.betterworldbooks.com](http://www.betterworldbooks.com)> [Accessed 22 Mar. 2020].

Dutta, P. and Dutta, P., 2019. Comparative Study of Cloud Services Offered by Amazon, Microsoft and Google. *International Journal of Trend in Scientific Research and Development*, Volume-3(Issue-3), pp.981–985.

Fries, D., 2019. A brief history of agile project management with Kanban. [online] pp.1–4. Available at: <<https://www.mindjet.com/blog/2019/03/brief-history-agile-project-management-kanban/>>.

Google Inc., 2020. *Products & Services | Google Cloud*. [online] Available at: <<https://cloud.google.com/products>> [Accessed 19 Mar. 2020].

Groenendijk, K., Guild, E. and Barzilay, R., 2000. The Legal Status of Third Country Nationals who are Long-Term Residents in a Member State of the European Union. *Community Relations series*, [online] (April), pp.1–116. Available at: <<http://cmr.jur.ru.nl/cmr/docs/status.pdf>>.

Huang, O., 2019. *Amazon AWS, Google Cloud, Alibaba Cloud and now Microsoft*

*Azure - Why does Salesforce want to run its software on all of them? - Olive Huang.* [online] Gartner Inc. Available at: <<https://blogs.gartner.com/olive-huang/amazon-aws-google-cloud-alibaba-cloud-now-microsoft-azure-salesforce-want-run-software/>> [Accessed 19 Mar. 2020].

Jin, S.-S., Wu, W.-J. and Huang, X.-M., 2017. Improving Property Service for the City Residential Community. *ITM Web of Conferences*, [online] 12, p.03049. Available at: <[https://www.itm-conferences.org/articles/itmconf/pdf/2017/04/itmconf\\_ita2017\\_03049.pdf](https://www.itm-conferences.org/articles/itmconf/pdf/2017/04/itmconf_ita2017_03049.pdf)>.

Laaziri, M., Benmoussa, K., Khouilji, S. and Kerkeb, M.L., 2019. A Comparative study of PHP frameworks performance. In: *Procedia Manufacturing*. Elsevier B.V. pp.864–871.

Lee, Y.T., Hsiao, W.H., Huang, C.M. and Chou, S.C.T., 2016. An integrated cloud-based smart home management system with community hierarchy. *IEEE Transactions on Consumer Electronics*. [online] Available at: <<https://ieeexplore.ieee.org/abstract/document/7448556>> [Accessed 28 Jul. 2019].

Li, Y., Cao, L., Qian, Y., Shi, W. and Liu, A., 2010. A property management system using WebGIS. In: *ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings*. [online] Available at: <<https://ieeexplore.ieee.org/document/5485699>> [Accessed 29 Jul. 2019].

Madsen, J.J., 2007. Technology 's Impact on the Property Management Profession ( al ). *Buildings Magazine*. [online] Available at: <<https://www.buildings.com/article-details/articleid/4836/title/technology-s-impact-on-the-property-management-profession-al->>.

Muslihat, D., 2018. 7 Popular Project Management Methodologies And What They're Best Suited For. *Zenkit*, [online] pp.1–25. Available at: <<https://zenkit.com/en/blog/7-popular-project-management-methodologies-and-what-theyre-best-suited-for/>>.

Nan, S., Zhou, M. and Li, G., 2018. Optimal residential community demand response scheduling in smart grid. *Applied Energy*, [online] 210, pp.1280–1289. Available at: <<https://www.sciencedirect.com/science/article/pii/S030626191730819X>> [Accessed 15 Aug. 2019].

Poortinga, W., Calve, T., Jones, N., Lannon, S., Rees, T., Rodgers, S.E., Lyons, R.A. and Johnson, R., 2017. Neighborhood Quality and Attachment: Validation of the Revised Residential Environment Assessment Tool. *Environment and Behavior*, [online] 49(3), pp.255–282. Available at: <<https://journals.sagepub.com/doi/full/10.1177/0013916516634403>> [Accessed 28 Jul. 2019].

Powell-Morse, A., 2017. Iterative Model: What Is It And When Should You Use It? *Airbrake*, [online] pp.3–5. Available at: <<https://airbrake.io/blog/sdlc/iterative->

model>.

R.Raval, R. and M. Rathod, H., 2013. Comparative Study of Various Process Model in Software Development. *International Journal of Computer Applications*, [online] 82(18), pp.16–19. Available at: <[https://www.researchgate.net/publication/260632268\\_Comparative\\_Study\\_of\\_Various\\_Process\\_Model\\_in\\_Software\\_Development](https://www.researchgate.net/publication/260632268_Comparative_Study_of_Various_Process_Model_in_Software_Development)> [Accessed 15 Jul. 2019].

Rahman, M.S., Hussain, B., Uddin, A.N.M.M. and Islam, N., 2015. Exploring residents' satisfaction of facilities provided by private apartment companies. *Asia Pacific Management Review*.

Rose, J.D. and Survesh, V.R.L., 2017. A case analysis of Node.js I/O performance under Linux environment in various storage media. In: *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*. Institute of Electrical and Electronics Engineers Inc.pp.1967–1973.

Rouse, M., 2005. *Prototyping Model*. [online] Available at: <<https://searchcio.techtarget.com/definition/Prototyping-Model>> [Accessed 15 Jul. 2019].

Schwaber, K. and Beedle, M., 2001. *Agile Software Development with Scrum*. Upper Saddle River, NJ, United States: Pearson Education.

Sharma, Y. and Gupta, S., 2020. A Study of Flutter and React Native for Mobile App Development. In: *Our Heritage. 5th International Conference On 'Innovations in IT and Management'*. [online] pp.691–698. Available at: <<https://archives.ourheritagejournal.com/index.php/oh/article/view/2568>> [Accessed 19 Mar. 2020].

Stackify, 2017. What is Agile Methodology? How It Works, Best Practices, Tools. [online] pp.1–14. Available at: <<https://stackify.com/agile-methodology/>>.

Tachibana, Y., Kon, J. and Yamaguchi, S., 2018. A study on the performance of web applications based on RoR in a highly consolidated server with container-based virtualization. In: *Proceedings - 2017 5th International Symposium on Computing and Networking, CANDAR 2017*. Institute of Electrical and Electronics Engineers Inc.pp.580–583.

Thakur, R.N. and Pandey, U.S., 2019. The Role of Model-View Controller in Object Oriented Software Development. *Nepal Journal of Multidisciplinary Research*, 2(2), pp.1–6.

UKEssays, 2018. *Reusability of Object Oriented Interfaces in UML Diagrams*. [online] Available at: <<https://www.ukessays.com/essays/it-research/measure-the-reusability-of-object-oriented-interfaces-in-uml-diagrams.php>> [Accessed 22 Feb. 2020].

Wiegand, S., 2018. *Personal Kanban Part 1 — Why Todo-Lists don't work*. [online] Hackernoon. Available at: <<https://hackernoon.com/personal-kanban-part-1-why-todo-lists-don-t-work-3b5c6dc78708>> [Accessed 15 Jun. 2019].

Wu, W., 2018. React Native vs Flutter, cross-platform mobile application frameworks. [online] (March), pp.1–7. Available at: <<https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1>>.

Yan, Y., 2018. Research on the Innovation of Residential Property Management Under Internet Thinking. In: *2018 International Conference on Economy, Management and Entrepreneurship (ICOEME 2018)*. [online] Available at: <<https://www.atlantispress.com/proceedings/icoeme-18/25905637>> [Accessed 29 Jul. 2019].

## APPENDICES

### APPENDIX A: Interview Question

1. What do you live?

Kajang

2. What type of residence you live?

Apartment

3. Do you encounter any problem in your residential area, such as fee payment, facilities issue etc.?

Ya, I always need to personally go to guard house to take the receipt.

4. Can you explain more details about it?

Sure, after I transfer the money through e-banking to the management account, I have to find a time to go to guard house for the receipt. But the problem is the office only open from 10.00 a.m. to 5.00 p.m., it's difficult to find a timeslot as I always have class at weekdays and overlapped with the office opening hours.

## APPENDIX B: Visitor Registration Item at Cypress Condominium

MORNING. CHIFFY			731311	8442	5221			
DATE	NAME	PASSPORT-IC-NO	PASS-NO	ROOM-NO	CAR-NO	IN-TIME	OUT-TIME	VISITOR-
14-08-19			1	C-2-5-3	WMP 9221	09:06	12:16	visitor
14-08-19			2	C-5-7-3	WNR 1698	09:45	16:27	visitor
14-08-19			4	C-1-1-1	AJR 8622	09:55	10:36	visitor
14-08-19			13	C-3-8-3	WTU 8476	10:13	12:44	visitor
14-08-19			7	C-2-5-3	WTB 9061	10:25	10:47	visitor
14-08-19			2	C-3-4-1	BMV 7802	10:21	17:06	visitor
14-08-19			2	C-3-6-2	VAQ 6097	10:49	17:13	visitor
14-08-19			4	C-4-9-3	SU 1073W	12:01	18:00	visitor
14-08-19			X	C-5-4-3	VU 6026	12:11	12:00	TEL.COM
14-08-19			10	C-4-9-4	EFM 6090	12:25	14:04	visitor
14-08-19			9	C-3-6-2	VV 9585	12:28	12:50	WORI
14-08-19			3	C-3-6-2	VEH 9585	12:29	17:13	CONTR
14-08-19			7	C-4-1-4	AGC 8015	13:13	15:09	visitor
14-08-19			9	C-3-6-2	WAD 240	13:26	16:56	CONTR
14-08-19			1	C-3-6-2	WNS 6992	13:57	16:53	CONTR
14-08-19			10	C-5-6-5	VEG 4278	14:06	15:07	visitor
14-08-19			7	C-6-2-2	WLY 4739	15:21	13:56	visitor
14-08-19			6	C-2-5-3	WTB 9061	15:55	18:00	visitor
14-08-19			8	C-2-7-1	WAB 6267P	16:08	22:32	visitor
14-08-19			4	C-3-8-2	WTU 8476	16:19	17:57	visitor
14-08-19			10	C-1-8-4	WE-36	18:06	22:23	visitor
14-08-19			1	C-5-3-1	WSS 9118	18:19	19:24	visitor
14-08-19			13	C-5-7-4	BW 8229	20:23	21:18 PM	visitor
14-08-19			12	C-4-7-3	WTU 6476	21:09	21:28 PM	visitor
14-08-19			7	C-3-6-3	BSA 3509	21:56		visitor
14-08-19			6	C-5-7-3	WU 1898	22:01		visitor
14-08-19			11	C-4-4-3	WTU 6476	22:35		visitor



# Residential Community Association Management Questionnaires

People always seek for higher standard of comfort, productivity and quality of life. Today's residential management are found to be fraught with inefficiencies and they are generally backwardness in term of technology. It is expected to be more advance and integrate with smart technologies in current decade. A better community association management system is required to let the residential life to be more convenience and optimal.

NEXT

Page 1 of 8

Never submit passwords through Google Forms.

## Basic Information

What kind of residence are you living? \*

- Condominiums
- Terrace/Link house
- Apartment
- Other: \_\_\_\_\_

Does your residence have its own community association application? \*

- No
- Yes

If yes, please specify the app name

Your answer

BACK

NEXT

Page 2 of 8

Never submit passwords through Google Forms.

## Security/Maintenance Fee Payment

Do you face any problem(s) related to payment of fees before? \*

- No
- Yes

If yes, please specify the problem(s)

Your answer

Generally, how many times you pay the fee in one year? \*

- 0
- 1
- 2
- 3
- More than 3

What type of payment methods you prefer to do the payment of general fees? \*

- Cash
- E-payment

What is your prefer way to receive receipt of the payment? \*

- Hard-copy receipt
- Soft-copy receipt

BACK

NEXT

Page 3 of 8



## Residents Information Update

What is your prefer way to register/update your information to residence's management? \*

- Hard-copy form
- Online form

BACK

NEXT

Page 4 of 8

Never submit passwords through Google Forms.

## Feedback

Do you face any problems listed below while you submit complaint/feedback to the residence's management? \*

- Anonymity
- Lost track of status of complaint/feedback
- Did not receive any update
- Problem did not solve

What type of complaint/feedback you more probably to submit? \*

Your answer

If the feedback process will keep you anonymous, do you more likely to submit a feedback? \*

- No
- Yes

BACK

NEXT

Page 5 of 8

Never submit passwords through Google Forms.

## Announcement

Do you ever miss any announcement that clipped in the residence noticeboard? \*

- No
- Yes

Do you agree online announcement is better than announcement at noticeboard? \*

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

BACK

NEXT

Page 6 of 8

Never submit passwords through Google Forms.

## Visitors System

How many visitors normally you have in one week? \*

- 0-2
- 3-5
- 6-8
- More than 9

Do you feel the current visitor system is a hassle process? \*

- No
- Yes

BACK

NEXT

Page 7 of 8

Never submit passwords through Google Forms.

## Others

In your opinion, what is the weakness of current existing community association management system? \*

Your answer \_\_\_\_\_

If there is a community association mobile application, who will responsible to use the application in your house? What is the age of the person? \*

Your answer \_\_\_\_\_

BACK

SUBMIT

 Page 8 of 8

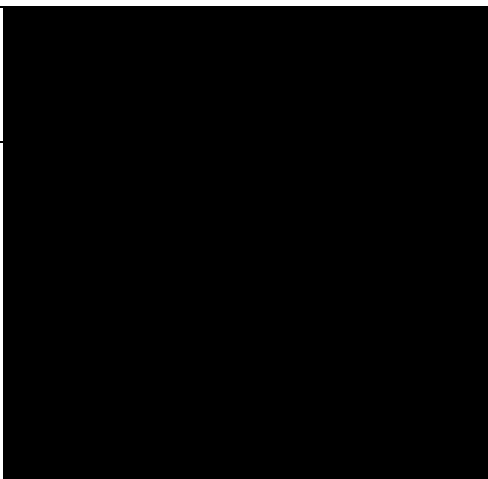
Never submit passwords through Google Forms.

APPENDIX D: Supervisor and moderator comments on project plan

<b>Project title:</b>	A Mobile App for Community Association
<b>Student Name</b>	GOH CHONG XIAN
<b>Supervisor</b>	Ms Chean Swee Ling
<b>Moderator</b>	Ts Dr Madhavan Nair

<b>Key Assessment for Project Proposal</b>	<b>Supervisor Comments/Remarks</b>	<b>Moderator Comments/Remarks</b>
<b>Project Description</b> - Is the problem or need to be addressed clearly presented? - Is the proposed approach or solution clearly presented and justified?	None.	The term community association management is too vague. The term residential management or Joint Management Committee (JMC) are the commonly used terms. Reference should be included for presented facts.
<b>Project Scope and Objectives</b> - Is the scope of the project clearly defined? - Are the objectives of the project clearly specified? - Are the project scope and objectives appropriate for a final year project?	None.	The objectives should be clearly written and measurable at the end of the project. Some rephrasing is required. Suggest to number each objective.
<b>Literature Review / Fact Finding for Benchmarking / Verification of Project</b> - Are sources for literature review / fact finding appropriate? - Is information from literature review / fact finding relevant and adequate?	Suggest study cost for hosting service.	More review is needed for similar applications available in the market. The features available should also be presented to make comparisons.

<ul style="list-style-type: none"> <li>- Is information from literature review / fact finding clearly presented and discussed?</li> </ul>		
<p><b>Research/Development Methodology and Development Tools</b></p> <ul style="list-style-type: none"> <li>- Is the methodology for the project clearly described and discussed?</li> <li>- Are the required development tools clearly described and discussed?</li> <li>- Are the stated methodology and development tools appropriate?</li> </ul>	None.	Good
<p><b>Project Plan</b></p> <ul style="list-style-type: none"> <li>- Are the phases and tasks of the project properly defined and planned?</li> <li>- Are the phases and tasks consistent with the methodology of the project?</li> </ul>	None.	Good
<p><b>Initial Deliverables</b></p> <ul style="list-style-type: none"> <li>- Are deliverables (e.g. use case diagrams and descriptions) of initial phases of the project plan included in the report?</li> </ul>	<p>Need to add role of security guard in use case diagram.</p> <p>Need to capture all/more possible cases, such as the visitor reaches guard house without barcode.</p>	Should consider the security personal as one of the actors in the USE CASE diagram.
<p><b>Report Structure and References</b></p> <ul style="list-style-type: none"> <li>- Is the report organised in a logical structure?</li> <li>- Are references listed</li> </ul>	None.	

in accordance to Harvard format?		
<b>Language and Clarity of Writing</b> - Are the sentences concise and understandable? - Are there spelling and grammar issues?	None.	

## APPENDIX E: Kanban board (Trello) progress

