PREMISE REAL-TIME OCCUPANCY INFORMATION SYSTEM

MUHAMMAD ZARIFUL HIMAM BIN RADZUAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**BORANG PENGESAHAN STATUS TESIS**

JUDUL: PREMISE REAL-TIME OCCUPANCY INFORMATION SYSTEM

SESI PENGAJIAN: 2018/2019

Saya: Muhammad Zariful Himam Bin Radzuan\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

mengaku membenarkan tesis (PSM/Sarjana/Doktor Falsafah) ini disimpan di Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dengan syarat-syarat kegunaan seperti berikut:

1. Tesis dan projekadalah hakmilik Kolej Universiti Teknikal Kebangsaan Malaysia.
2. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. \*\* Sila tandakan (/)

\_\_\_\_\_\_\_ SULIT (Mengandungi maklumat yang berdarjah

keselamatan atau kepentingan Malaysia seperti

yang termaktub di dalam AKTA RAHSIA RASMI

1972)

\_\_\_\_\_\_\_ TERHAD (Mengandungi maklumat TERHAD yang telah

ditentukan oleh organisasi/badan di mana

penyelidikan dijalankan)

\_\_\_/\_\_\_\_ TIDAK TERHAD

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(TANDATANGAN PENULIS) (TANDATANGAN PENYELIA)

Alamat tetap : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nama Penyelia

Tarikh : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Tarikh : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CATATAN: \*\* Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa.

^ Tesis dimaksudkan sebagai Laporan Projek Sarjana Muda (PSM)

PREMISE REAL-TIME OCCUPANCY INFORMATION SYSTEM

MUHAMMAD ZARIFUL HIMAM BIN RADZUAN

This report is submitted in partial fulfilment of the requirements for the

Bachelor of Computer Science in Software Development with Honours

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2019

# DECLARATION

I hereby declare that this project report entitled

PREMISE REAL-TIME OCCUPANCY INFORMATION SYSTEM

is written by me and is my own effort and that no part has been plagiarized

without citations.



STUDENT :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date : 30/5/2019\_\_\_\_

(Muhammad Zariful Himam Bin Radzuan)

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of Computer Science in Software Development with Honours.



SUPERVISOR : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: 30/5/2019\_\_\_

(MOHD HARIZ BIN NAIM @ MOHAYAT)

# DEDICATION

This project is wholeheartedly dedicated to my beloved parents, who have been my source of inspiration and gave me strength when I thought of giving up, who continually provide their moral, spiritual, emotional, and financial support. To my brothers, sisters, relatives, mentor, friends, and classmates who shared their words of advice and encouragement to finish this project.

# ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my supervisor En. Mohd Hariz Naim @ Mohayat who under his tutelage, provided me with the golden opportunity to do this wonderful project, who guided me in the direction to take for this project, provided me with the materials, and the necessary hardware for me to develop this system. For this, I am thankful.

Next, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame, and also for all the moral support that they provided that finally allowed me to push on and finish this project.

# ABSTRACT

The Premise Real-Time Occupancy Information System, which henceforth shall be known as the Quick! System, is a system utilising computer vision, to collect incoming and outgoing movement of people at a premise, thus determining its busyness or fullness. This system will then enable its users to make an informed decision using the data. The owner of the premise will also benefit from the strategic analysis provided by the system.

Table of Contents

[1. INTRODUCTION 1](#_Toc11344146)

[1.1. Introduction 1](#_Toc11344147)

[1.2. Problem Statements 2](#_Toc11344148)

[1.3. Objective 2](#_Toc11344149)

[1.4. Scope 2](#_Toc11344150)

[1.5. Project Significance 3](#_Toc11344151)

[1.6. Expected Output 3](#_Toc11344152)

[1.7. Conclusion 4](#_Toc11344153)

[2. LITERATURE REVIEW AND PROJECT METHODOLOGY 4](#_Toc11344154)

[2.1. Introduction 4](#_Toc11344155)

[2.2. Facts and Findings 4](#_Toc11344156)

[2.2.1. Domain 4](#_Toc11344157)

[2.2.2. Existing System 5](#_Toc11344158)

[2.3. Project Methodology 7](#_Toc11344159)

[2.4. Project Requirement 8](#_Toc11344160)

[2.4.1. Software Requirement 8](#_Toc11344161)

[2.4.2. Hardware Requirement 9](#_Toc11344162)

[2.4.3. Other Requirements 9](#_Toc11344163)

[2.5. Project Schedules and Milestones 9](#_Toc11344164)

[2.6. Conclusion 11](#_Toc11344165)

[3. ANALYSIS 11](#_Toc11344166)

[3.1. Introduction 11](#_Toc11344167)

[3.2. Problem Analysis 11](#_Toc11344168)

[3.3. Requirement Analysis 12](#_Toc11344169)

[3.3.1. Data Requirement 13](#_Toc11344170)

[3.3.2. Functional Requirement 15](#_Toc11344171)

[3.3.2.1. Mobile Application 15](#_Toc11344172)

[3.3.2.1.1 End Users 15](#_Toc11344173)

[3.3.2.1.2 Owner 16](#_Toc11344174)

[3.3.2.2. Web Service 16](#_Toc11344175)

[3.3.2.3. Microcontroller 17](#_Toc11344176)

[3.3.2.4. System Architecture Overview 18](#_Toc11344177)

[3.3.3. Non-Functional Requirement 18](#_Toc11344178)

[3.4. Conclusion 19](#_Toc11344179)

LIST OF FIGURES

[Figure 1 UCSD Waitz (now deprecated) 6](#_Toc11348127)

[Figure 2 Waitz's approach for multiple area in a premise 6](#_Toc11348128)

[Figure 3 Google's Popular Times 7](#_Toc11348129)

[Figure 4 Representation of the Feature Driven Development Model 8](#_Toc11348130)

[Figure 5 Current flow 12](#_Toc11348131)

[Figure 6 Flow with application 13](#_Toc11348132)

LIST OF DIAGRAMS

[Diagram 1 Milestone of this project 10](#_Toc11343063)

[Diagram 2 System architecture visualised 18](#_Toc11343064)

# LIST OF TABLES

[Table 1 Software Requirement 8](#_Toc11343087)

[Table 2 Hardware Requirement 9](#_Toc11343088)

[Table 3 Milestones of this project 10](#_Toc11343089)

[Table 4 Table for Place 13](#_Toc11343090)

[Table 5 Table for Owner 15](#_Toc11343091)

[Table 6 Table for Occupancy 15](#_Toc11343092)

[Table 7 End User Requirement 16](#_Toc11343093)

[Table 8 Owner Requirement 16](#_Toc11343094)

[Table 9 Web service requirement 16](#_Toc11343095)

[Table 10 Microcontroller Requirement 17](#_Toc11343096)

[Table 11 Non-Functional Requirement 18](#_Toc11343097)

# INTRODUCTION

## Introduction

In the hustle and bustle of urban living, time is of the essence, and with such a high average population density in the city, it is inevitable for people to waste time waiting instead of doing what they wanted to. Take for example, when going to a restaurant, and finding that it is packed to the brim. That is time wasted. And what about when a person is sick, and without knowing, went to a busy clinic, and ended up having to spend hours waiting to see the doctor. How about when the place is suddenly closed outside of its usual closing time?

So, what if instead, a person can first check the map and see that their favourite restaurant is busy, and then are able to decide to go to a different restaurant instead. The same goes to the sick person. When every important facilities and premises’ information are readily available to the public, maximum efficiency can be achieved by everyone. There will no longer be a situation where time is wasted going to a place that doesn’t have the capacity to serve you due to volume or having to deal with unexpected downtime.

With the Quick System, a premise can keep track of how many people are inside at any given time, and propagate that information freely to users, and also to push notifications to users that are tracking its occupancy state, therefore able to notify when a place is no longer busy. Additionally, for owners, they can easily change the attributes of their premise, such as the opening times, name, and maybe occasionally override the premise opening to closed for any unexpected situations, and also using the same notification system, able to directly send messages to their customers that chose to track their premise.

But best of all, with the Quick System Analysis and Dashboard, an owner can also see the trend for their customer density, presented visually, so they can manage their employees accordingly. This can increase productivity per employee as there will no longer be a day with too many employees and too little customers resulting in too much downtime, or conversely too little employees and too many customers resulting in reduced customer satisfaction.

All in all, with the information that can be provided by the Quick System, we hope it can improve productivity for both the end-user and the premise owner.

## Problem Statements

1. Urbanites living in the city are facing a problem with congestion. They unwittingly waste time when every place that they visit are busy and unable to or will take a long time to serve them.
2. Premise can also be closed unexpectedly, and without a direct link to the owner, customers will have no idea of knowing so.
3. Premise owners faces problem in seeing trends of customer density and arranging workforce accordingly.

## Objective

The objective for the development of the Quick System is to develop an app that implements an occupancy counter using Computer Vision to count the coming and going of patrons inside a premise, then updating it into a database.

## Scope

Project will be designed to include multiple types of premises, such as banks, clinics, government buildings (JPJ, JPN, JIM) and various kinds of stores.

The scope of this project covers designing and implementing Computer Vision on a microcontroller, a web service to provide entry point for other platforms, and designing a mobile application for:

1. End users
2. To view premise current occupancy state, as in their busyness, visually.
3. Enable user to track, or subscribe to a premise, therefore enabling them to receive notifications regarding the premise, where that may be busyness updates, opening updates, or direct broadcast messages from the owner.
4. Owners
5. View the current information regarding their premise.
6. Update certain information, such as the opening and closing times, or to override it and set it to close or open.
7. View the information of customer density that is presented visually for them to strategize around.
8. To develop a program that will run on a microcontroller, currently set to be a RaspberryPi3, that will:
9. Count the coming and going of people in the range of the premise using a camera paired with computer vision.
10. Update the premise state in Firebase database.
11. Add an entry in MYSQL, known as a snapshot, with timestamp and current value, to be used in analysis.
12. Send notification to users when occupancy level drop below a certain percentage.
13. And finally, a web service to link all of them together that will:
    1. Enable the microcontroller to interface with both the MYSQL and Firebase database.
    2. Enable the push notification system that will be used by both the microcontroller and the applications.

## Project Significance

1. Users will be able to use the information provided by the application to make their decision. Thus, giving them the choice to look for another place that is less busy, or stay where they are until the application sends a notification that the place is no longer busy.
2. The owner themselves will be able to check their real-time occupancy and to declare business unavailability due to events or special bookings and whatnot.
3. Owners will also be able to directly communicate with their customers through the notification system, thus enabling better business interaction.
4. Premise owners will have more control of their premise and easily propagate information such as changed opening times, special off day, and etcetera to their customers.
5. Owners will also be able to strategize their employees’ shifts around the analysed busyness trend, thus increasing productivity.
6. The Quick system also does not store user location nor does it track its users, thus ensuring user privacy is protected.

## Expected Output

* 1. Occupancy counter will be developed and integrated with database.
  2. The data will be projected into a human readable information through the mobile application.
  3. Users will be able to use the information provided by the application to make their decision. User will also be able to track a business’ occupancy, getting a notification once it is not busy.
  4. Owners will be able to check real-time information of their premise on the application.
  5. Users will be able to get notifications when tracked premises are no longer busy.
  6. Premise owners will be able to control if their premise is available, thus overriding the occupancy counter.

## Conclusion

This application can potentially solve a lot of grievance that urbanites experiences in their daily life. For end users, by providing information, we are also giving them the freedom of choice. And the strategic information that can be used by the owner also can’t be underestimated, as when it is properly used, it can be used to significantly increase productivity. The simple no fuss nature of the app requires minimal input from both type of users to be able to function, thus increasing the possible coverage of user.

# LITERATURE REVIEW AND PROJECT METHODOLOGY

## Introduction

Planning is an important aspect of a software development cycle, and without proper planning, problems may arise later in development. This chapter will describe the methodology that was used to plan and develop the system, namely, the Feature Driven Development Methodology, and also the requirements, the relevant researches, schedules, development technique, and previous works.

## Facts and Findings

## Domain

This system is about automatically counting people using a camera utilising computer vision. The system will then aggregate the data inside a real-time database, allowing users to view real-time situation of premise around them. The system will also enable users to track a premise, and get notifications whenever there is a state change, e.g. from not busy to busy, busy to full, and busy to not busy. They will also be able to get notifications from premise owners themselves. The premise owners will also be able to view information regarding their premise, in addition to being able to change certain options such as current occupancy, maximum occupancy, and opening status on the fly.

## Existing System

People counting in a commercial perspective is not a new field. For events such as large-scale concerts, fairs and out-door art festivals, it is important to know the number of people attending. By counting people at the entrance, this information can be gained. Traditional method is by using manual counting, that is, by using humans. But manual counting is costly as humans can only work for a limited time, and not to mention that employing more people induces more cost. Placing a mechanical counter, such as a turn gate, might be impractical in some cases. Therefore, using cameras and camera vision has been suggested (Oosterhout, et al., 2012). By utilising computer vision, a discipline where Artificial Intelligence is implemented in a computer and then used to identify objects from a source, it is possible for a computer to automatically recognise and classify objects. Previous implementations of people counting includes using infrared sensors (Oosterhout, et al., 2012). Previously, this has been implemented in getting foot traffic in a retail store in order to maximise customer view of product by efficiently designing layouts according to traffic heatmap (Dwivedi, 2018).

The most similar previous iteration of this project is the Waitz system developed by Waitz Inc. for the University of California San Diego. It uses small hardware devices to monitor Bluetooth and Wi-Fi signals (MAC addresses) in the area, and then using their proprietary normalisation technique, turns the data into occupancy information (Waitz Inc., 2018). This means that, even when not connected, a device will still send out random pings to nearby networks. This was used to then track individual devices and count them. However, starting in IOS8, iPhones uses a random MAC address when probing for new networks (O'Grady, 2014), while Android includes an option to enable it from the developer option starting from Android 9 (Pie) as it disables it by default (Alphabet Inc., 2019). What this means is that, it is no longer feasible to track devices through MAC address, as it will be randomised with each ping, resulting in a seemingly infinite number of devices. Although unmasking the anonymisation is not impossible (Martin, et al., 2017), it is unpractical and not worth it.

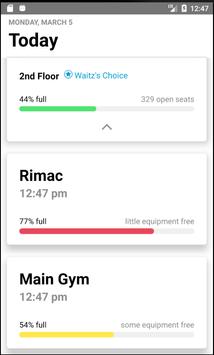


Figure 1 UCSD Waitz (now deprecated)

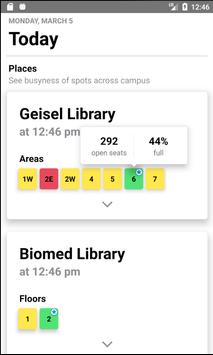


Figure 2 Waitz's approach for multiple area in a premise

Next, the Popular Times implemented by Google is also similar to the Quick system, in that both delivers information to customers to prevent them from wasting their time going to an establishment only to find that it is unable to serve them from being too busy. Popular Times by Google is implemented by getting location updates from users that installed Google Maps, then is projected into a graph showing both live and usual busyness. The main difference is that the Quick system does not require all the customers of a premise to own a smartphone, as counting will be done by analysing camera feed, which most places will already have, for example a security camera, with artificial intelligence. The Quick system also does not store user location nor does it track its users like Google does, thus ensuring user privacy is protected. But the biggest benefit for implementing the Quick system for the owners is that they will have a direct line to customers that subscribes to their premise, enabling them to push out a message, and enabling them to know if their premise is closed due to unexpected situation, or maybe to simply push out a message.

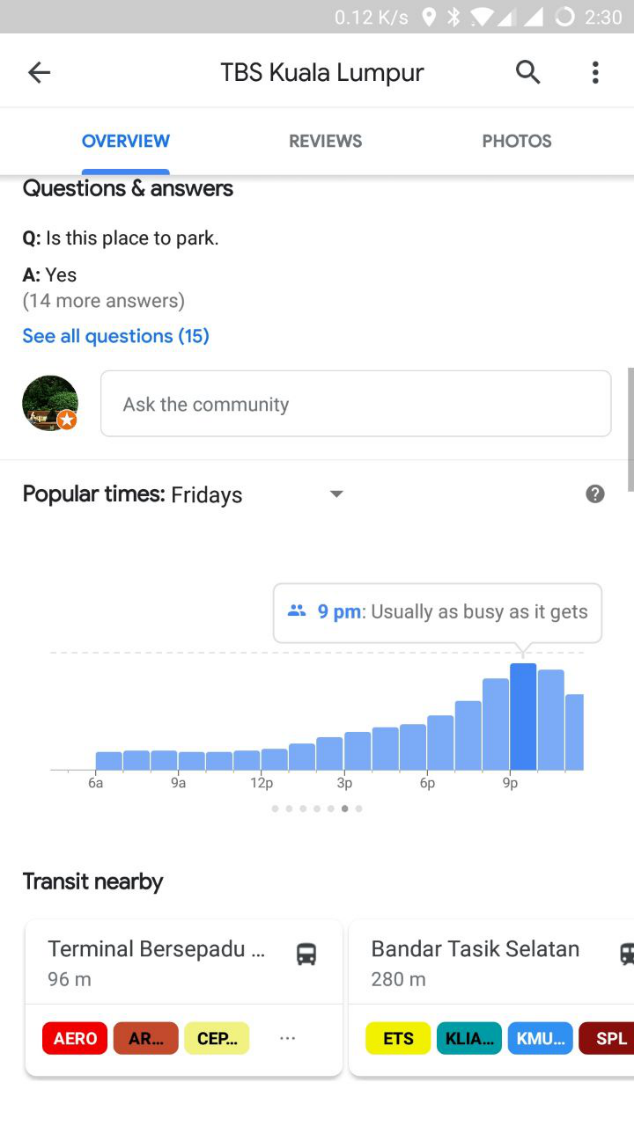


Figure 3 Google's Popular Times

## Project Methodology

The project will be developed bases on the Feature Driven Development Methodology (FDD). FDD is a subset of the Agile development models. It is described as having just enough process to ensure scalability and repeatability while encouraging creativity and innovation. The model was chosen due to the feature driven nature of the application. It consisted of discrete features that adds on to other features, and is not necessarily crucial. The system as a whole is also quite complex. Therefore, this model allows for the developer team to quickly develop by features, and push them out in increments, and not aim at a big full release. This gives the developers more space to innovate and act creatively, thus possibly increasing productivity and values for features.

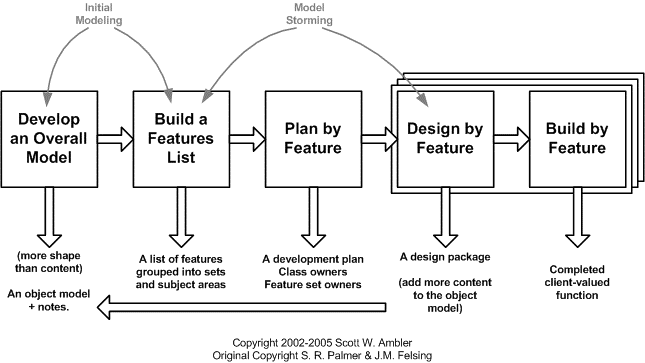


Figure 4 Representation of the Feature Driven Development Model

FDD has proven to be a highly effective way to rescue complex projects because it addresses so many of the problems that so commonly afflict developers. The entire feature list is built to the priorities of business users and its fast approach – using two-week increments – giving users the chance to use the application before it has been finished.

## Project Requirement

## Software Requirement

Software requirement for this project is as follows:

Table 1 Software Requirement

|  |  |
| --- | --- |
| **Software** | **Description** |
| Android Studio | Platform to build and code mobile applications |
| StarUML | Platform to draw diagrams such as flowchart and use case |
| Microsoft Visio | Platform to draw diagrams DFD and ERD |
| Python IDE | Platform to code the python program that is used by the microcontroller |
| Python Shell | Program to build and run the python program |
| Firebase | Platform for real-time database, user credentials and premise information |
| MYSQL | Platform for database of snapshots of premise occupancy after updates. Used for analysis. |
| XAMPP | Local server deployment platform for local testing |

## Hardware Requirement

Hardware requirement for this project is as follows:

Table 2 Hardware Requirement

|  |  |
| --- | --- |
| **Hardware** | **Description** |
| Computers | To develop the necessary programs and services. |
| Servers | To host the relational database and the web service. |
| Android device | For development and testing purpose of the application. |
| RaspberryPi 3B | To run required scripts to process image and update count value. |
| PiCamera | To get image and pass to the microcontroller |

## Other Requirements

Other than the ones mentioned above, this project also uses some open source codes. Namely, the Computer Vision algorithm used to detect people. The codes used in this project has been heavily modified, for example, the python script is now thread based, and instead of using video files, it will instead use PiCamera. The count value is also separated and getters defined to enable the updater to get the value. In essence, the counter is now wrapped around the updater, which is also wrapped around the main python script which will determine the premise identification.

## Project Schedules and Milestones

This section will list out the entire schedule while developing the project from the start until the end of the project. Milestone are needed to help the developer to know exactly the date to start or finish a certain process in the project. This can help the developer to finish the project on time. The project planning is important to plan the flow and the timeline of the project. The purpose is to ensure that every part of the project can be completed and do not delay the whole project. Thus, with the milestone the project plan will help the developer to manage the time. The following shows the schedule of this project;

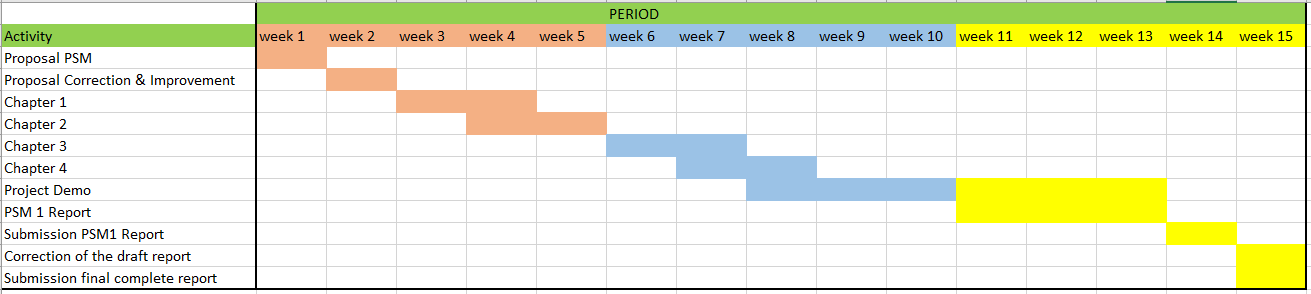


Diagram 1 Milestone of this project

Table 3 Milestones of this project

|  |  |  |  |
| --- | --- | --- | --- |
| WEEK | ACTIVITY | START DATE | END DATE |
| 1 | Proposal PSM: Discussion Proposal assessment & verification | 18 February 2019 | 18 February 2019 |
| 2 | Proposal correction/improvement | 25 February 2019 | 01 March 2019 |
| 3 | Proposal Presentation & Submission via PSM Online System  - Chapter 1 (System Development Begins) | 04 March 2019 | 08 March 2019 |
| 4 | Chapter 1  Chapter 2 | 11 March 2019 | 25 March 2019 |
| 5. | Chapter 2  Chapter 3 | 25 March 2019 | 29 March 2019 |
| 7 | Chapter 3 | 01 April 2019 | 05 April 2019 |
| 9 | Demonstration project to supervisor | 13 May 2019 | 17 May 2019 |
| 11 | Final Presentation to supervisor and evaluator | 30 May 2019 | 30 May 2018 |
| 12 | Final report submission | 30 May 2019 | 30 May 2019 |

## Conclusion

Literature review and methodology has been detailed in this chapter. The summary of the research papers has also been discussed at this chapter. A proper software development planning and model can make the development process a success. The next chapter will explain about the analysis phases and how the system going to be developed. It includes the problem analysis and requirement analysis.

# ANALYSIS

## Introduction

This chapter details the analysis that was done on this system. Analysis was done by analysing the existing system, finding out the problems, and implementing new ideas and innovation as solutions. This is done to define the goals and objectives.

## Problem Analysis

The current system is the Popular Times implemented by Google. It does not provide direct control to the owner, and users will also have to check periodically to know that the place is not busy.

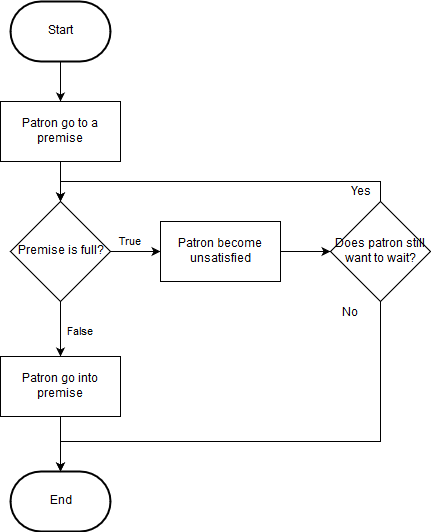


Figure 5 Current flow

## Requirement Analysis

The requirement analysis for this chapter comprises of functional, software, hardware, and network requirements

Below is the proposed flow of patrons after implementation and usage of application.

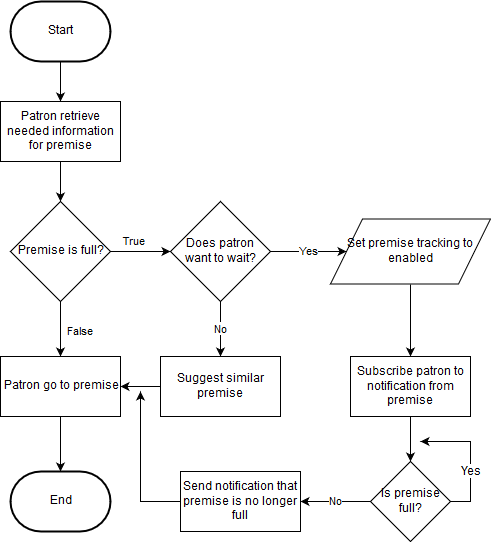


Figure 6 Flow with application

## Data Requirement

This section will describe the data that the system should input and output, as well as the data that is needed to be stored in the database.

Table 4 Table for Place

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Description** | **Data Type** | **Sample Data** | **Constraint** |
| placeId | Identification for place | String | 2OwMV6VJBteLogHtlhe0K9tFRNd2 | Primary key |
| ownerId | Identification for owner of place | String |  | Not null |
| placeLatitude | Latitude of place | Double | 2.273756 | Not null |
| placeLongitude | Longitude of place | Double | 102.285736 | Not null |
| overrideStatus | Override status from owner | Integer | -1 | Not null |
| placeName | Name of place for display | String | McDonald’s MITC | Not null |
| lastUpdated | Occupancy last updated time | String | 12:30 |  |
| currentOccupancy | Current number of people in premise | Integer | 100 |  |
| maxOccupancy | Maximum number of people in the premise | Integer | 120 | Not null |
| address | Address of place | String | Lingkaran MITC | Not null |
| openingHours0Opening | Opening of Monday | String | 08:00 |  |
| openingHours0Closing | Closing of Monday | String | 21:00 |  |
| openingHours1Opening | Opening of Tuesday | String | 08:00 |  |
| openingHours1Closing | Closing of Tuesday | String | 21:00 |  |
| openingHours2Opening | Opening of Wednesday | String | 08:00 |  |
| openingHours2Closing | Closing of Wednesday | String | 21:00 |  |
| openingHours3Opening | Opening of Thursday | String | 08:00 |  |
| openingHours3Closing | Closing of Thursday | String | 21:00 |  |
| openingHours4Opening | Opening of Friday | String | 08:00 |  |
| openingHours4Closing | Closing of Friday | String | 21:00 |  |
| openingHours5Opening | Opening of Saturday | String | 00:00 |  |
| openingHours5Closing | Closing of Saturday | String | 00:00 |  |
| openingHours6Opening | Opening of Sunday | String | null |  |
| openingHours6Closing | Closing of Sunday | String | null |  |

Table 5 Table for Owner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Description** | **Data Type** | **Sample Data** | **Constraint** |
| email | Email of owner | String | zariful.radzuan@gmail.com | Unique |
| password | Password of owner | String | 123 | Not null |
| UID | Use id of owner | String | h5vGJlkhUsgjlV4fuVF6iiQ7HoJ3 | Not null, Unique |

Table 6 Table for Occupancy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Description** | **Data Type** | **Sample Data** | **Constraint** |
| idOccupancy | Id of occupancy state | Int(11) | 1 | Primary  key |
| idPlace | Id of place | Varchar(32) | 2OwMV6VJBteLogHtlhe0K9tFRNd2 |  |
| timestamp | Timestamp of occupancy snapshot | Timestamp | 2019-05-29 11:52:07 |  |
| valueAtTime | Value at snapshot | Int(11) | 120 |  |

## Functional Requirement

This section defines and describes the functional requirements of the Quick system. These requirements will be split into three section to account for the platform that it is developed on.

## Mobile Application

The mobile applications, one for end users and another for the owner are developed on the Android operating system.

## End Users

Table 7 End User Requirement

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Place Browse | Enable user to browse list of places. |
| Map Interface | Enable user to quickly glance at an area and know the condition. Markers in the map will be colour coded. |
| Tracking | Enable users to track/subscribe to an establishment, thus enabling them to receive updates notifications. |
| Filtering | Enable users to sort/filter places by types. (Clinics/Restaurants/Gyms) |

## Owner

The premise owner will have a separate mobile application for them to quickly and easily access information regarding their premise and to change relevant information.

Table 8 Owner Requirement

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Dashboard | Display glance information to owner, such as current occupancy, open status, and ability to override open status |
| Push Message | Enable owner to push messages to subscribing end users. |
| Analysis | Enable owners to see the occupancy trends of their occupancy in the form of a graph. |

## Web Service

The web service will be hosted online, and used to interface with the database. The system will use both Firebase, which is a NoSQL database, and the MYSQL database, a relational database. The reason why will be described shortly.

Table 9 Web service requirement

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Send notification | Able to push notifications to end users that subscribes to any premises.  Example of notifications include:  If premise fullness is equal to 100%,  “<PlaceName> is at full capacity”  If premise fullness is >69%  “<PlaceName> is getting busy”  If premise fullness is <51%  “<PlaceName> is not busy anymore.” |
| Update databases | Enable microcontrollers to handle the data sent from the microcontroller and update the Firebase and MYSQL database simultaneously.  The data received from the microcontroller is first parsed, making sure that it does not exceed maximum and minimum value, which is the maximum occupancy of the premise and 0 respectively. |

## Microcontroller

The microcontroller will receive stream from camera, and detects the coming and going of patrons before updating it to the database.

Table 10 Microcontroller Requirement

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Send updates | Send data at 5 minutes interval to the server using POST. Parameter is ‘placeid’, and ‘valueToAdd.’ |
| Count people | Using OpenCV computer vision, count the patrons going in and out, then send the data to the web service. |

## Non-Functional Requirement

Non-functional requirements considered important. They ensure that the system function as expected and perform well without any error, and is of certain quality.

Table 11 Non-Functional Requirement

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Standard | * Use standard coding for interface designing, implementing and data coding. * Interface designing, implementing and data coding based on standard use for mobile application and programming. |
| Security Requirements | * No unregistered user should have access. * All data inside database need to be secure to ensure the integrity of the data and unauthorized user cannot access data without privilege. |
| Performance requirements | * Perform under any condition. * Database should be online 24 hours. |
| Usability | * This system is easy to use and user easy to understand the flow of system. |
| Reliability | * The database update process must roll back all related updates when any update fails. |
| Simple GUI | * The interface should be easy to understand to the user and the navigation through the system interface is user friendly. |

## Conclusion

In this analysis phase system requirement was gather in order to ensure the functionality of the system before process to the next stages and in order to fit to the system scope. This chapter has provided several flow charts as early visual of the quick system. By visualizing through the flow chart, the system process can be seen clearly.

# DESIGN

## Introduction

This chapter will discuss the system design and the activities involved. The activities involved includes converting information, functional, and non-functional requirements that was identified in the analysis phase into the design specification. This chapter will also discuss about defining the result of the analysis of the preliminary and detailed design.

## High-Level Design

High-Level design (HLD) explains the architecture that will be used for the development of the Real-Time Premise Occupancy Counter (Quick!) application. This architecture defines the software structure which will fulfil the requirements needed by the user.

## System Architecture

Below is the visual representation of the system architecture.

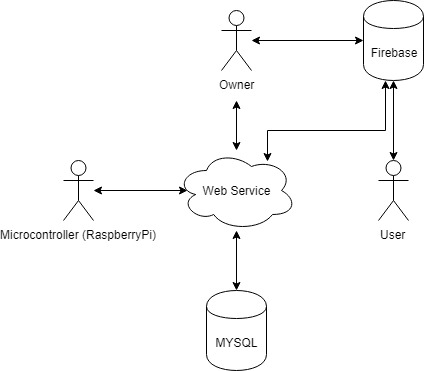


Diagram 2 System architecture visualised

The system uses a dual database methodology. Firebase provides the real-time database functionality, acting as the core database of the system while MYSQL provides the logging functionality which is used for analysis purposes.

## Flowchart Diagram

Below are the flowcharts representing core activities and actions in each application and environment.

## Firebase data retrieval

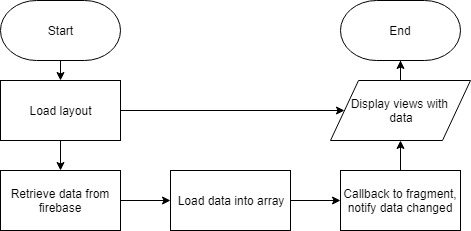


Figure 7 Typical flow of retrieving data from Firebase

As Firebase data request are processed asynchronously, it is important that there is a callback to the calling function to load the retrieved data. A normal assignment (e.g. data=getFromFirebase()) is not possible when using Firebase as it will return a null value.

## End-User Tracking

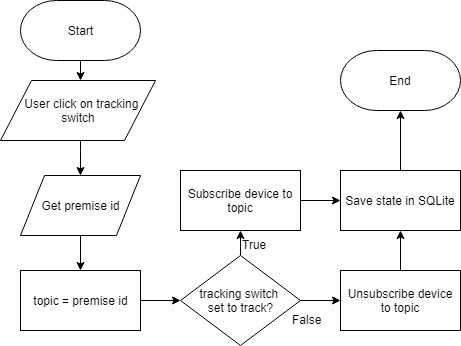


Figure 8 Tracking of premise in end-user application

Firebase provides a topic-based subscription feature for notification. In this application, a premise’s unique id is used as the topic, thus allowing easy dispatch of notifications to the subscribing (tracking) devices. The state saved in SQLite are only used to set the tracking status in the premise detail.

## Chart Generating

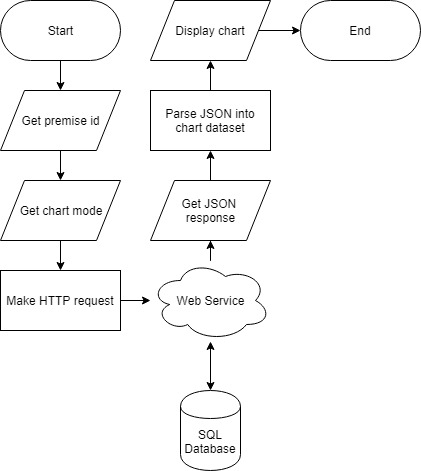


Figure 9 Generating chart

Charts in the owner application are generated from data collected by the microcontroller. Every time an update is done, the new states are saved in a separate database, namely the SQL database. SQL is used due to its ease in creating complex queries, something Firebase lacks in. Charts are generated in two types, or modes. Weekly and daily. Daily charts display the trends of patrons by hours, while weekly displays the same but in the range of days in a week (Monday-Sunday). The data are limited to the latest 30 days.