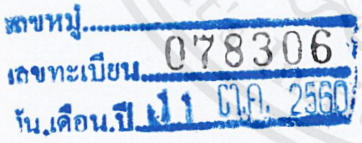
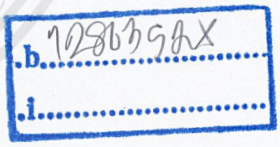
# Universal Locking System: InfiniLock (KMITL Bike)


Chatchaya Chanchua
Pattarawut Imamnuaysup
Sasawat Chanate




Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2016
KMITL-2017-IC-B-003-001

**Thesis — Academic Year 2016**

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

**Title:** Universal Locking System: InfiniLock (KMITL Bike)

**Authors:**


1. Ms. Chatchaya Chanchua — Student ID 56090004
2. Mr. Pattarawut Imamnuaysup — Student ID 56090014
3. Mr. Sasawat Chanate — Student ID 56090023


Approved for submission

(Assistant Professor Dr. Ronnachai Tiyarattanachai)

Project Adviser

Date 1 / 6 / 17

(Dr. Isara Anantavrasilp)

Project Co-adviser

Date 1 / 6 / 17

# Abstract

In any security scheme, physically sharing the access key could be really cumbersome. It lacked immediacy as well as risking key duplication after shared. InfiniLock aimed to solve this problem by introducing universal keyless sharable locking system.

Since the scope of this system was universal, this paper limited the work to one of the applicable system, bicycle-sharing system. Traditionally, bicycle-sharing system required station for lending the bicycle and respective key keeper. However, the main idea of this thesis, KMITL Bike, eliminated all these components.

The basis of KMITL Bike involved stationless lending process as well as keyless sharing system. Instead of using physical key, user was able to use any bike through mobile application, which was developed along with the system. After an authentication through the specified application, the user could then proceed to borrow the bike. The bike was secured by using a semi-automated lock controlled by an Arduino microprocessor. This allowed the bike to be locked anywhere anytime, removing the need for station. The application commanded the lock via Bluetooth module, therefore, no physical key was required.

# Acknowledgments

We would like to express our deep gratitude to Asst. Prof. Dr. Ronnachai Tiyarattanachai and Dr. Isara Anantavrasilp, our project advisers, for providing us with the knowledge and experiences along with funds and facilities, not to mentioned, countless of enthusiastic encouragement we need in order to complete the project. We would also like to thank Asst. Prof. Dr. Chaiwat Nuthong and Dr. Natthapong Jungteerapanich, for his advice in using different techniques and technologies in our project and showing us some of the related work he found that helps us to shape the idea for the project. A huge contribution was also made by Kajornsak Peerapathananont, our iOS lead and sole developer and the first team members who helped with the creation of the first prototype as a proof of concept, namely Boonnithi Jiaramaneepinit, Chaiwat Wattanapaiboonsuk, Pavit Noinongyao, and Puriwat Khantiviriya. A thank to Thitiwat Watanajaturaporn and the KMITL Bike Volunteers in helping us to regularly maintain the bicycles and services. Thank you Peeranut Chindanonda for providing us a 3D printer and other 3D printing related equipments along with advice and techniques in printing our prototypes. Our grateful thanks are also extended to Mr. Kiatkachorn Saripan in providing us valuable suggestions and comments on our project. Furthermore, we cannot forget Mr. Jamrat Anuchartchaikul and Mr. Nachatpong Somboonpun from Thai Vehicle Industry Co., Ltd. that assisted us in creating a prototype for this project. Finally, we wish to thank whoever reads this very copy of the report for spending valuable time to read the report, we are sure it is full of mistakes. Please do not hesitate to make corrections as well as suggestions to our report and kindly return your copy to us at the BiGGA Laboratory.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Security is increasingly becoming more prominent due to the result of economic inflation [1]. With regard to this, emerge varieties of means to implement security. One of the widely-used means of security is locking system. In this document, two usage types of locks are defined as: private and public.

Private locks are locks used with private properties, e.g., house, bicycle, and cabinet. This type of lock is used mostly by only a few trustful users (e.g. family members). Properties used with private locks are not focused on being shared and the owner is the main person who locks and unlocks it.

Public locks, on the other hand, are locks used with public properties, e.g., office, public bicycle, and public lockers. This type of lock is for public users to access it. Owner of the property are expected to have a lower degree of control on the usage of the property and user is not as trustworthy as in private; users might not have an incentive to secure the property after usage.

Even though locks have been created since the ancient Egypt, not much have changed [2]. People still have to own a physical key which must be carried at all times and could be easily lost or stolen. A combination lock allows user to use the lock without having to carry a key, but when sharing the combination once, the lock is forever shared. In a digital world, this problem can be solved by using a

virtual key that can be kept, share, or revoke at any time.

## 1.2 Problem Statement

Securing or locking is arguably the most troublesome part of using a product especially when it is shared between multiple users. Conventional method requires users lending their keys or giving away their combinations in order to give access to the person. However, the method is insecure and, in the case of lending key, laborious.

One of the largest issues of using a physical key is the ability to share especially for public usage. Sharing physical keys could mean giving away the key which can be easily duplicated and ultimately compromising security. However, there is some business that requires public sharing of key such as a public bicycle rental service.

A public bicycle should be made in such a way that is easy to access and return while being secure. In the present implementation, it is not that simple since users will have to go register at a physical booth for a key which is needed to be returned after each usage. This makes accessing the service troublesome and insecure.

Therefore, there is a need for the study to develop a lock system that can be easily shared for public use, while having suitable a security level. The system should also entail state-of-the-art design that can allow users to lock and unlock device via a mobile device.

## 1.3 Contribution

InfiniLock aims to provide a universal keyless sharable locking system which would be a generic system serving variety use cases for securing items.

KMITL Bike serves as the first implementation of the InfiniLock system. Its purpose is to encourage students and faculty members of King Mongkut's Institute of Technology Ladkrabang to use bicycles as a primary mode of transport instead of motorized vehicles. This could lead to less air pollution as well as reducing the

traffic within the institution.

## 1.4 Objectives

The objective of this study is to develop a lock system that can:

- Lock and unlock using user common owned device (e.g. mobile phone)
- Share lock to a specific person/group
- Revoke usage of the shared lock
- Track usage of each lock

## 1.5 Scope

KMITL Bike was selected as an initial implementation of InfiniLock. This is because at the moment there is a need for bicycle rental system in the campus to tackle the problem of traffic and pollution within the area. The scope of this thesis is to create a working software system of KMITL Bike and a working prototype hardware lock. Several real-world test flight should be executed to further refine the system to better suit the users' requirements and fix bugs and glitches.

| | Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | Project Initialization | 21d? | 01/08/2016 | 30/08/2016 | |
| 2 | Discuss the motive of the research topic | 1w? | 01/08/2016 | 10/08/2016 | |
| 3 | Discuss existing systems | 1w? | 10/08/2016 | 19/08/2016 | 2 |
| 4 | Discuss technology & systems to be use | 1w? | 19/08/2016 | 30/08/2016 | 2,3 |
| 5 | Requirements & System Design | 26d? | 30/08/2016 | 05/10/2016 | |
| 6 | List the requirements of the system | 1w? | 30/08/2016 | 08/09/2016 | 2,3,4 |
| 7 | Discuss & create use cases | 3d? | 08/09/2016 | 13/09/2016 | 6 |
| 8 | Create activity diagram | 3d? | 13/09/2016 | 16/09/2016 | 7 |
| 9 | Create class diagram | 1d? | 16/09/2016 | 19/09/2016 | 8 |
| 10 | Create sequence diagram | 3d? | 19/09/2016 | 22/09/2016 | 9 |
| 11 | Wireframes and mocoups | 7d? | 13/09/2016 | 22/09/2016 | 7 |
| 12 | Design database | 2d? | 19/09/2016 | 21/09/2016 | 9 |
| 13 | Design server responses | 2d? | 22/09/2016 | 26/09/2016 | 10 |
| 14 | Finalize the system design | 1w? | 26/09/2016 | 05/10/2016 | 2,6,7,8,9,10,11,1 |
| 15 | System Implementation | 49d? | 05/10/2016 | 13/12/2016 | |
| 16 | Implement database | 3d? | 05/10/2016 | 10/10/2016 | 14 |
| 17 | Implement server | 1w? | 05/10/2016 | 14/10/2016 | 14 |
| 18 | Implement app | 2w? | 05/10/2016 | 25/10/2016 | 14 |
| 19 | Implement Hardware Prototype #1 | 4w? | 05/10/2016 | 25/10/2016 | 14 |
| 20 | Hardware Implementation | 49d? | 05/10/2016 | 13/12/2016 | |
| 21 | Research existing models of lock | 1w? | 05/10/2016 | 14/10/2016 | 14 |
| 22 | Model the new lock | 4w? | 14/10/2016 | 23/11/2016 | 21 |
| 23 | Find suitable components for the new lock | 1w? | 23/11/2016 | 02/12/2016 | 22 |
| 24 | Create successful connection with Bluetooth module | 1w? | 02/12/2016 | 13/12/2016 | 23 |
| 25 | Logic programming on the microprocessor | 1w? | 02/12/2016 | 13/12/2016 | 23 |
| 26 | Internal System Tests & Debugging Process | 28d? | 25/10/2016 | 02/12/2016 | |
| 27 | Internal System Test #1 | 1w? | 25/10/2016 | 03/11/2016 | 16,17,18,19 |
| 28 | Debugging process | 1w? | 03/11/2016 | 14/11/2016 | 27 |
| 29 | Internal System Test #2 | 1w? | 14/11/2016 | 23/11/2016 | 28 |
| 30 | Debugging process | 1w? | 23/11/2016 | 02/12/2016 | 29 |
| 31 | Winter Break | 22.88d? | 14/12/2016 | 13/01/2017 | |
| 32 | Test Flight #1 | 28d? | 16/01/2017 | 22/02/2017 | |
| 33 | Ready system for Test Flight | 1w? | 16/01/2017 | 24/01/2017 | 31 |
| 34 | Conduct Field Test #1 | 3w? | 25/01/2017 | 22/02/2017 | 33 |
| 35 | Improve the new lock | 2w? | 09/02/2017 | 22/02/2017 | 34SS 1w |
| 36 | Improve server and app | 2w? | 09/02/2017 | 22/02/2017 | 34SS 1w |
| 37 | Test Flight #2 | 31d? | 23/02/2017 | 06/04/2017 | |
| 38 | Ready system for Test Flight | 3d? | 23/02/2017 | 27/02/2017 | 35,36 |
| 39 | Conduct Test Flight #2 | 3w? | 28/02/2017 | 28/03/2017 | 38 |
| 40 | Improve the new lock | 3w? | 09/03/2017 | 05/04/2017 | 39SS 1w |
| 41 | Improve server and app | 3w? | 09/03/2017 | 06/04/2017 | 39SS 1w |
| 42 | Final Test Flight | 24d? | 07/04/2017 | 10/05/2017 | |
| 43 | Ready system for Test Flight | 3d? | 07/04/2017 | 11/04/2017 | 40,41 |
| 44 | Conduct Final Test Flight | 3w? | 12/04/2017 | 10/05/2017 | 43 |
| 45 | Project Finalization | 28d? | 21/04/2017 | 30/05/2017 | |
| 46 | Finalized the new lock | 3w? | 21/04/2017 | 19/05/2017 | 44SS 1w |
| 47 | Finalized server and app | 3w? | 21/04/2017 | 19/05/2017 | 44SS 1w |
| 48 | Finalized report | 3w? | 02/05/2017 | 30/05/2017 | 44SS 2w |

Figure 1-1: Gantt Chart

# Chapter 2

# Literature Review

This chapter presents related work to the bicycle-sharing system. There are two sections that will be further discussed in this chapter: existing work and comparison.

## 2.1 Existing Work

There are many kinds of lock systems designed to serve the stated purposes, each with varying strengths and weaknesses. Below is a summary of some of the exceptional work in relation to this project.

### 2.1.1 MU's White Bike

One of the most primitive bicycle-sharing system, "White Bike" is provided by Mahidol University. It uses the concept of sharing physical keys to unlock the bike with no computerized system. Users are required to exchange their student ID card with a security guard nearby the station for the key to unlock the bike. After usage, the bike needs to be returned at the same station in order for an exchange back of key and ID card.

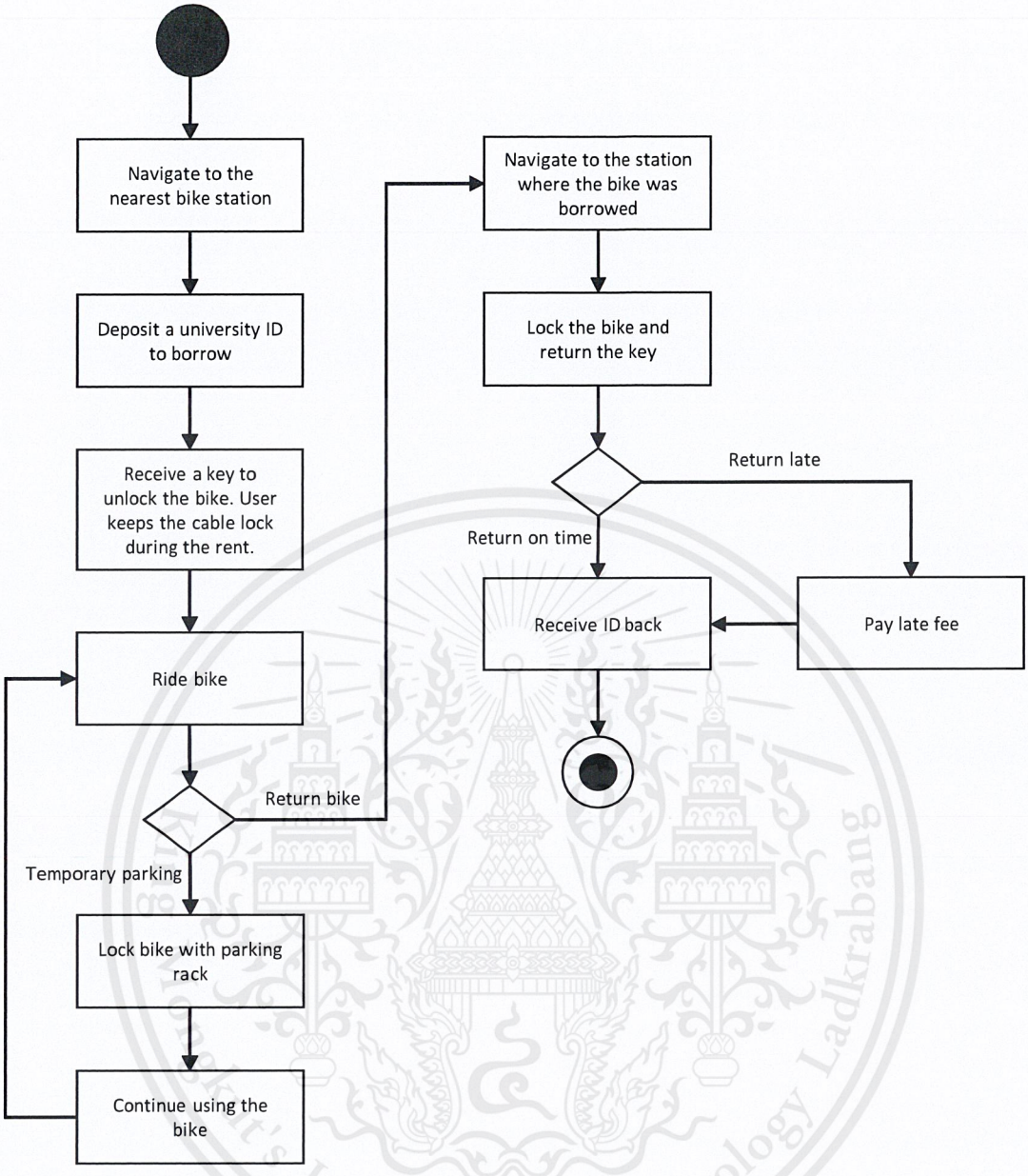



Figure 2-1: Workflow of MU's White Bike

### 2.1.2 CU Bike

CU Bike, provided by Chulalongkorn University is an implementation by Smoove, a French company which designs, manufacture, and install bicycle-sharing system solutions. An incremental step of MU's White Bike, instead of having a security guard to manage the keys, CU Bike uses a computerized locking station and bike. User will be required to have a membership card that uses RFID to tap and unlocks

the bike with a combination of personal PIN code. After finish using users will be able to return the bike at any station as opposed to the same station from the MU's White Bike.

The system consists of combination of short-range and long-range wireless communication. Each bike communicates with a nearby station via ZigBee while each station communicates to the central server via GPRS.

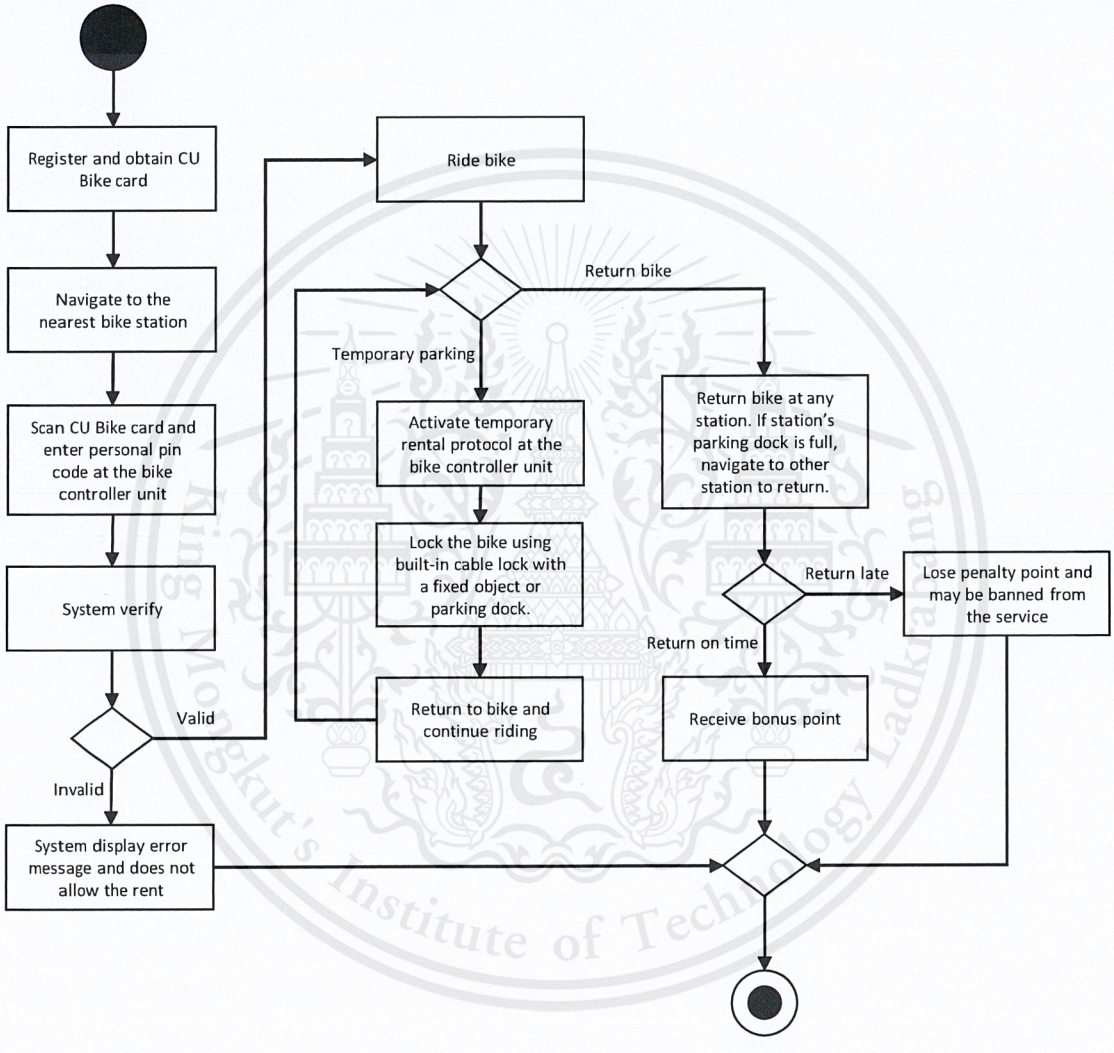



Figure 2-2: Workflow of CU Bike

### 2.1.3 weBike

This application runs on a scope of campus level. It focuses on the security of lending a bike to a student in the campus. With a simple process of retrieving a

passcode from the server and entering it onto a keypad, it accomplishes a certain level of security. However, with the current level of technology, this method is not sufficient as it lacks countermeasures against fake authorization as a student. Its simplicity reflects this flaw.

Additionally, the lock it uses, the U-Lock, requires a stationary bar or pole to be attached with. Looking ahead in case of rising numbers of bikes, there might not be enough pole or bar and would cost a lot to build more.

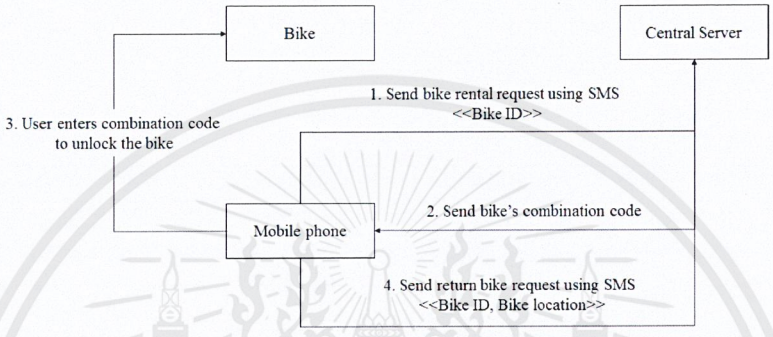



Figure 2-3: Workflow of weBike

The authentication methods that are being used in weBike are a keypad lock with an aid of SMS. SMS or Short Message Service is a text messaging service component of a telephone. User uses the SMS as a medium to communicate with weBike's server. By sending the identifier of the bike they want to rent via SMS, weBike server will send the bike's combination code back to the user. Even if a lot of people consider SMS to be old-fashioned, but it is still up and alive. It can be a very responsive method to connect with users and acts as an alternative to push notification. For that reason, weBike gains an advantage in that way. However, the use of SMS is likely prone to message modification, man-in-the-middle attack, unauthorized message access, and identity impersonation [3].

### 2.1.4 MuniBike

Interestingly, this application approaches security in several ways. This is not surprising with a city-wide scale it is running on. It provides different alternative

means to retrieve a key to unlock the lock and use the bike such as SMS, VoiceCall, Mobile Application, or just simply keypad. Although the means are diverse, they all lead to one security scheme, authorization with the server and receive a key. As a result, it, unfortunately, shares the same flaw as the previously mentioned application, weBike.

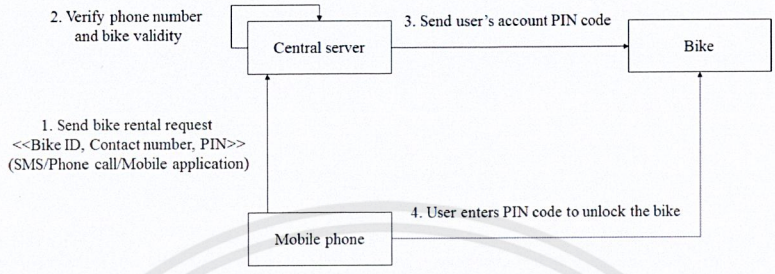



Figure 2-4: Workflow of MuniBike

### 2.1.5 BitLock

With a scope as wide as all end users is this application, BitLock. It similarly uses the U-Lock as its lock as the weBike application. However, comparing to the other two previously mentioned, Bitlock centers its focus differently. Unlike the two that have one owner of the system, a campus or a city as a whole, this application has many owners, each corresponds to the lock they bought. Therefore, only owner can use the lock, which leads to few questions on cases such that the owner wishes to share the lock or even sell out the lock. These matters are not answered with this application.

BitLock introduces a different security measure and communication technology approach than weBike and MuniBike by using Bluetooth. Bluetooth is one of the communication technology for short-range data transfers. It provides low power consumption and there are a lot of Bluetooth modules available on the market. Most devices such as smartphones or tablets support Bluetooth making it a popular choice for short-range communication standard. BitLock may be superior to others with its usage of Bluetooth. However, it does not support real-time GPS tracking, thus lacking a countermeasure against theft.

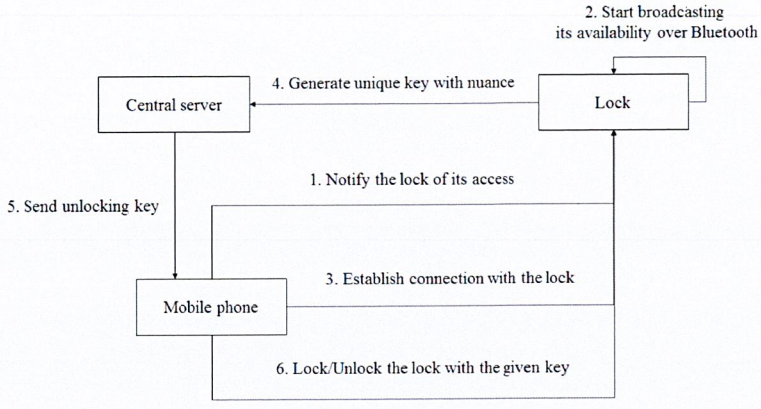



Figure 2-5: Workflow of BitLock

### 2.1.6 Noke (No-Key)

Noke runs in a similar manner to BitLock as it focuses on end users but with some twists. Noke answers the questions left by BitLock in a very satisfying way. It provides a way to share access to the lock and change of ownership. The shared access could be altered as pleased by the owner. Also, the lock does not limit to one specific usage. It could be used on many items such as door and bag. Regarding the security measures, Noke operates on the same design as BitLock, resulting in same strengths and weaknesses.

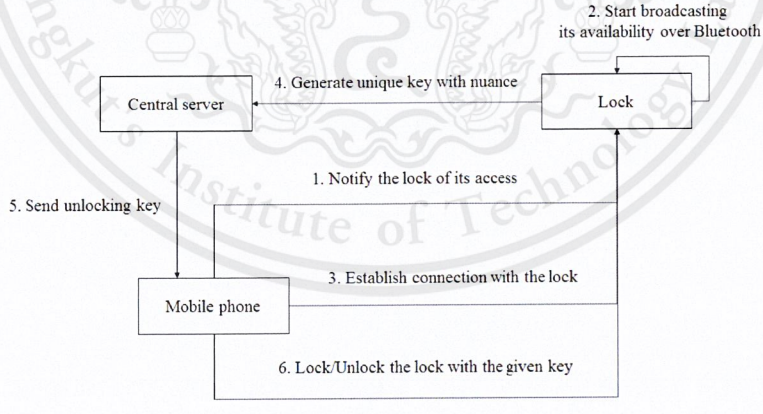



Figure 2-6: Workflow of Noke

## 2.2 Comparison

Below is the comparison of each work in many major aspects.

Table 2.1: Comparison between White Bike, CU Bike, weBike, MuniBike, BitLock, and Noke

| **Name** | **Target Group** | **Authentication Tools** | **Communication** |
|---|---|---|---|
| White Bike | University | Physical Key and Student ID card | None |
| CU Bike | University | Membership RFID card | GPRS, ZigBee |
| weBike | University | SMS with combination lock | None |
| MuniBike | City-wide | SMS, phone call, or mobile application with keypad lock | MAN/WAN |
| BitLock | End-users | Mobile application | Bluetooth |
| Noke | End-users and Enterprise | Mobile application | Bluetooth |

Table 2.2: Comparison between White Bike, CU Bike, weBike, MuniBike, BitLock, and Noke (continued)

| **Name** | **Bike-Locating** | **Anti-Theft Technology** | **Powering Method** |
|---|---|---|---|
| White Bike | None | None | None |
| CU Bike | None | Fork lock, cable lock | Rechargeable Battery |
| weBike | None | None | None |
| MuniBike | GPS | GPS tracking | Dynamo battery charger |
| BitLock | None | Phone's GPS and heat-treated, cut-resistant steel U-lock | Non-rechargeable battery |
| Noke | Phone's GPS | Anti-shim Padlock, U-lock | Rechargeable/ Swappable battery |

It can be inferred from the data in these tables that Bluetooth is widely used with mobile application. The reason for such popularity might be from the relatively low cost of the module and the fact that Bluetooth is ubiquitous on smart devices. Nonetheless, there are still only a few applications that really utilize mobile phone's potential such as MuniBike using keypad lock, or even weBike that only uses SMS.

# Chapter 3

# Requirements Analysis

This chapter provides detailed insights on the requirements resulting after the analysis of related existing works along with use cases of the system. Requirements are represented in a form of FURPS+ model (Table 3.1) which can aid in discovering potential needs that are both functional and non-functional.

## 3.1 Requirements

Table 3.1: Table of Requirements

| No | Requirement | Type |
|---|---|---|
| 1 | User shall be able to lock the bike. | Functional |
| 2 | User shall be able to unlock the bike. | Functional |
| 3 | User shall be able to unlock the bike using Android mobile devices. | Functional |
| 4 | User shall be able to unlock the bike using iPhone. | Functional |
| 5 | User shall be able to unlock using Bluetooth LE connection. | Functional |
| 6 | User shall be able to find the location of other bicycles in the area. | Functional |
| 7 | User shall be able to view past usage sessions. | Functional |
| 8 | User shall be able to login to the app. | Functional |

**Table 3.1 – continued from the previous page**

| No | Requirement | Type |
|---|---|---|
| 9 | User shall be able to logout from the app. | Functional |
| 10 | User shall be able to view the terms and conditions of the service. | Functional |
| 11 | User shall be able to report problems about the service. | Functional |
| 12 | System shall allow only KMITL students and faculty members to use the system. | Functional |
| 13 | System shall use I AM KMITL (KMITL Generation 2) account for verifying user's membership to KMITL. | Functional |
| 14 | System shall be able to keep track of users' usage sessions. | Functional |
| 15 | System shall be able to keep track of lock units' status. | Functional |
| 16 | System shall be able to securely send unlock command to the lock unit. | Functional |
| 17 | System shall prevent from unlocking the lock unit by using replay attacks by using cryptographic nonce. | Security |
| 18 | System shall uses RSA-2048 asymmetric encryption in unlocking the lock unit. | Security |
| 18 | System shall uses Bluetooth 4.0 LE in communicating between the client device and lock unit. | Security |

As for the mobile application workflow, the use cases of application are depicted below in Figure 3-1. Each use case is elaborated by using use case descriptions as shown in Table 3.2 - 3.8. Some technical details of main features will be described at the end of this subsection.

## 3.2 Use Case Diagram

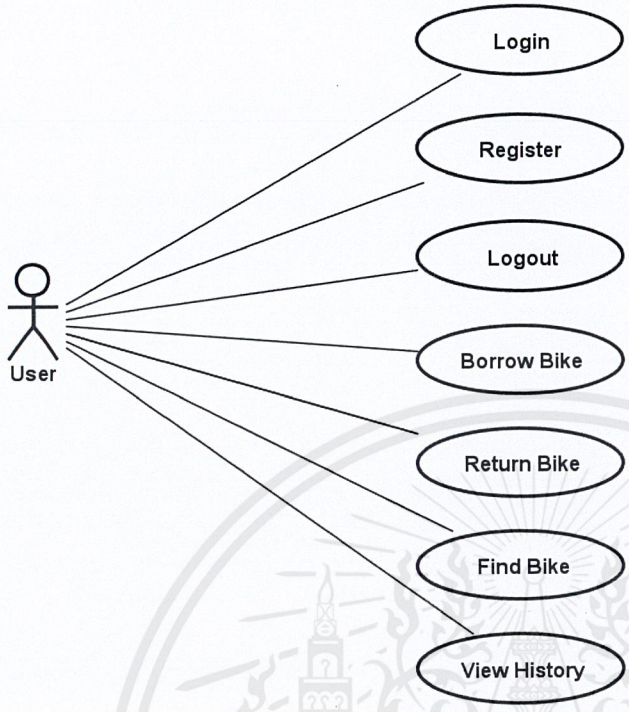



Figure 3-1: Use Case Diagram of KMITL Bike Application

## 3.3 Use Case Description

Table 3.2: Use Case Description - Login

| **Use Case** | **Login** |
|---|---|
| **Primary Actor** | All users |
| **Pre-condition** | User has the application on their device. User is registered. |
| **Post-condition** | User is logged in to the system. |
| **Flow of events** | 1) User opens the application |
| | 2) User enters their company username and password (ex: KMITL NAC) |
| | 3) Server recognizes that the user is already registered |
| | 4) Server verifies user's information |
| | 5) User is logged in to the system |
| **Alternative Flow** | **Condition: Wrong username/password** |
| | 3a) System notify user that the username/password is wrong |
| | **Condition: Unable to login** |
| | 4a) System notify the user and asks user to login again |

Table 3.3: Use Case Description - Register

| **Use Case** | **Register** |
|---|---|
| **Primary Actor** | All users |
| **Pre-condition** | User has the application on their device. |
| **Post-condition** | User is registered. User's information is recorded correctly into the server's database. |
| **Flow of events** | 1) User opens the application |
| | 2) User enters their company username and password (ex: KMITL NAC) |
| | 3) Server recognizes user never register |
| | 4) Application provide registration form |
| | 5) User enters personal information into the form provided in the application |
| | 6) Server verifies user's information |
| | 7) User is registered |
| **Alternative Flow** | **Condition: Invalid Information** |
| | 5a) Application asks user to re-enter the information into the application. |
| | **Condition: Unable to register** |
| | 6a) Application notifies the user and asks user to apply the registration again |

Table 3.4: Use Case Description - Logout

| **Use Case** | **Logout** |
|---|---|
| **Primary Actor** | Authorized user |
| **Pre-condition** | User has the application on their device. User is registered. User is logged in. |
| **Post-condition** | User is logged out from the system. |
| **Flow of events** | 1) User opens the application |
| | 2) User presses to "Logout" button on the application |
| | 3) Application performs logout for user |
| | 4) User is logged out from the system |
| **Alternative Flow** | **Condition: Unable to logout** |
| | 4a) System notify the user and asks user to logout again |

Table 3.5: Use Case Description - Borrow Bike

| **Use Case** | **Borrow Bike** |
|---|---|
| **Primary Actor** | Authorized user |
| **Pre-condition** | User is already logged in to the system through the application. The bike is in a locked state. |
| **Post-condition** | Bike is unlocked. Log is correctly saved into the system. Status of the bike is updated. |
| **Flow of events** | 1) User arrives at the bike with the application on mobile phone |
| | 2) User opens the application |
| | 3) User presses the switch on the bicycle box |
| | 4) User presses "Borrow Bike" button in the application |
| | 5) Application shows a list of bike usage plans |
| | 6) User selects a bike usage plan |
| | 7) User scans QR code that attached on the bike |
| | 8) Server verifies user's balance, status, and permission to borrow the bike |
| | 9) Application sends unlock key to the bike |
| | 10) Locking mechanism on the bike unlocks itself |
| **Alternative Flow** | **Condition: The bike is already borrowed** |
| | 9a) System notify the user that the bike is already borrowed by someone |
| | **Condition: Insufficient points to borrow** |
| | 9b) System notify the user that their points are insufficient to borrow |

Table 3.6: Use Case Description - Return Bike

| **Use Case** | **Return Bike** |
|---|---|
| **Primary Actor** | Authorized user |
| **Pre-condition** | User is already logged in to the system through the application. User is in a borrowing status. The bike is in an unlocked state. |
| **Post-condition** | Bike is lock. Log is correctly saved into the system. Status of the bike is updated. |
| **Flow of events** | 1) User opens the application |
| | 2) User presses "Return Bike" in the application |
| | 3) User scans QR code that attached on the bike |
| | 4) Application checks if the bike is locked |
| | 5) Server verifies user's status |
| | 6) User successfully returns the bike |
| **Alternative Flow** | **Condition: The bike is not locked properly** |
| | 5a) System notify the user that the lock is not locked properly |

Table 3.7: Use Case Description - Find Bike

| **Use Case** | **Find Bike** |
|---|---|
| **Primary Actor** | Authorized user |
| **Pre-condition** | User is already logged in to the system through the application. |
| **Post-condition** | - |
| **Flow of events** | 1) User opens the application |
| | 2) User press on "Find Bike" in the application |
| | 3) Application places bicycle "markers" on its map |
| | 4) User walks to the desire bike |

Table 3.8: Use Case Description - View History

| **Use Case** | **View History** |
|---|---|
| **Primary Actor** | Authorized user |
| **Pre-condition** | User is already logged in to the system through the application. |
| **Post-condition** | - |
| **Flow of events** | 1) User opens the application |
| | 2) User presses on "View History" tab in the application |
| | 3) Application lists all the borrowing sessions of user |
| | 4) User finds interested session |
| | 5) User presses on the interested session |
| | 6) Application displays an in-detail view of the interested session |

# Chapter 4

# Background Knowledge

This chapter will discuss background knowledge conducted in order to achieve the set objectives. Below are knowledge needed prior to fully understand the proposed solution.

## 4.1 Asymmetric Encryption

Asymmetric encryption is a form of cryptography and its sole purpose is to solve the age–the old problem of key sharing.

Prior to the asymmetric encryption, most encryptions were done symmetrically where both the sender and the reader need to know the key and that same key is used for both encryption and decryption of the message. The problem with symmetric encryption lies with the process of sharing the key between the sender and the reader. Since the key has to be kept secret, the key cannot be shared between the two parties securely without needing the two to be physically next to each other.

Asymmetric encryption was first thought as a form of "non-secret encryption" where the key does not need to be entirely secret [4]. Instead of having only one key to perform both the encryption and decryption of the message, asymmetric encryption has two different keys: a public key and a private key. The usage of two keys mentioned is called as public-key encryption [5]. Either of these keys

can be used for encryption or decryption but they must be used as a pair and performs different tasks. The difference between the private and public keys are as the name suggested, private keys are meant to be kept secret while public keys can be viewed by the public. To further clarify, public keys can be stolen but it will be useless since decrypting a message requires both public and private keys. Communication will be done by only the exchange of public keys, making making asymmetric encryption or public-key encryption a good candidate for a secure communication regardless of attempts on eavesdropping or interception.

## 4.2 Cryptographic Nonce

A cryptographic nonce is an arbitrary number that is made to be used once. Nonce consists of series of randomized number and is used in an encrypted message in order to prevent an event of a replay attack. Without nonce, every message that has the same content will appear to be the same even with encryption, attackers could save the message and replay it anytime.

## 4.3 Bluetooth Low Energy Module and Properties

Bluetooth low energy or BLE is a wireless personal area network technology with low energy functionality. BLE device can be run for long periods on power sources, such as coin cell batteries or energy-harvesting devices. It also comes with small size, low cost, and high compatibility for mobile phones and tablets, which is suitable for the Internet of Things [6]. In BLE devices, there is an extension of the classic Bluetooth stack that implements a specific Bluetooth profile known as the Generic Attribute Profile or GATT in short. BLE devices will use GATT to communicate with each other. Data are organized into nested objects called Profiles, Services, and Characteristics, as illustrated in Figure 4-1 [7].

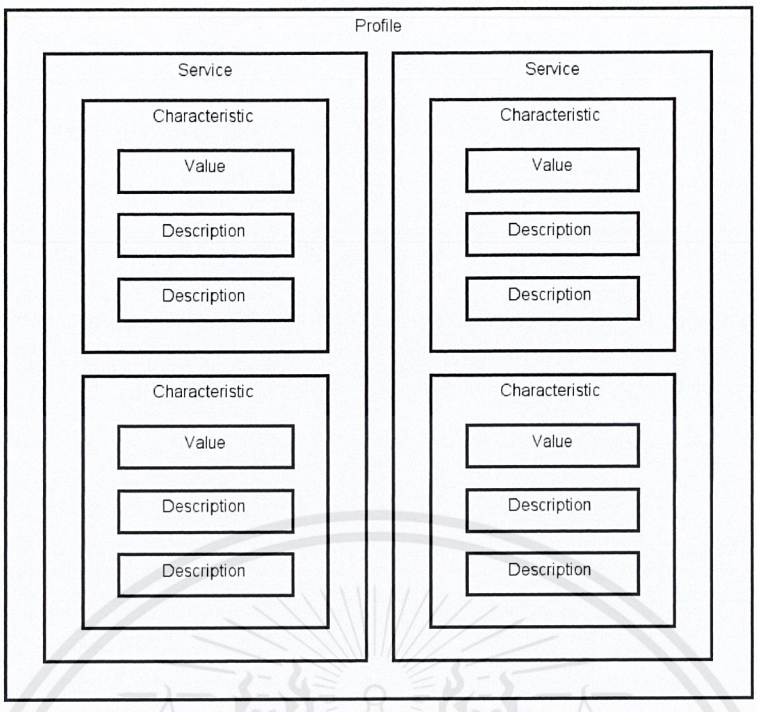



Figure 4-1: Overview of Bluetooth GATT

The GATT profile contains characteristics which are the data of BLE devices such as sensor data. Each characteristic is formed together into logical functions called service. Some characteristics are read-only, while others can be written for device configuration purposes. Inside each characteristic, there is a descriptor, which can be used to configure specific behaviors like notifications. Characteristic notifications allow configuration like pushing updates as depicted on schedule, or when the value of the characteristic changes. This is very efficient way to reduce the power usage since the host application is not required to connect to the remote peripheral all the time [7].

# Chapter 5

# System Design

This chapter provides an overview of the InfiniLock system design which is currently designed corresponding to the requirements of KMITL Bike. The design illustrates the system in a highly detailed manner ranging from the basis for implementation to the flow of system functions along with components that are used to construct the system.

## 5.1 System Architecture

The system consists of three main components: server, smartphone, and the lock unit. Since the lock unit does not contain a cellular network module, as shown in Figure 5-1, the smartphone primary serves as an interface between the lock unit and the server by communicating over Bluetooth low energy wireless connection. Users will also use the mobile application as the main controller in unlocking the bicycle.


Figure 5-1: Overview of the system design for KMITL Bike

### 5.1.1 KMITL Bike Server

In the server side there are few more important details kept compared to the overview shown earlier. Figure 5-2 illustrates the relationship of necessary data in the server. During the riding session, the server will keep track of the user, be it the borrow time, return time, and even the route the user took. All these information will be kept as user's histories which can be viewed at any time by them. Also, each session requires a deposit corresponding to the amount of time alloted. The model of the bike is recorded as well to each session. Finally, the server will keep track of application version, both on Android and iOS, to make sure user's application will always be up-to-date.

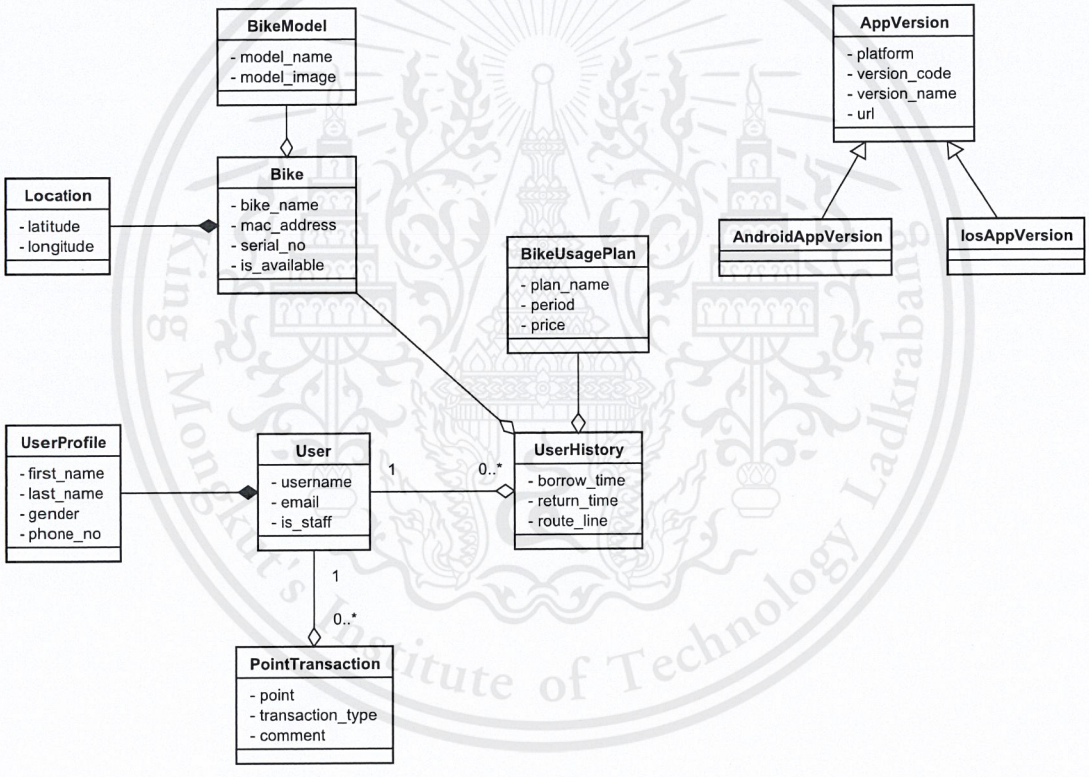



Figure 5-2: Simple class diagram of the server

### 5.1.2 KMITL Bike API

There are several services available on the server for the mobile application to use. The list of API and its description are described in Table 5.1.

Table 5.1: KMITL Bike REST API

| **Template** | **HTTP Verb** | **Description** |
|---|---|---|
| api/v1/auth/access_token | GET | Return credentials for a login session and identifies the user |
| api/v1/auth/login | POST | Login into the server, this will result a call to I AM KMITL account validation |
| api/v1/auth/logout | GET | Logout from the server |
| api/v1/auth/register | POST | Register a new account |
| api/v1/services/available | GET | Return a list of bikes that are currently available |
| api/v1/services/update_bike_location | POST | Update the current location of bike to the server |
| api/v1/user/status | GET | Return user's status whether he/she is still in riding session or not |
| api/v1/user/borrow | POST | Borrow the bike by bike ID |
| api/v1/user/return | POST | Return the bike |
| api/v1/user/history | GET | Return user's riding history |
| api/v1/user/update_user_location | POST | Update the current location of user, this will be invoked while the user is riding |

### 5.1.3 KMITL Bike Mobile Application

For mobile application, it was developed with MVP (Model-View-Presenter) architecture as a design pattern. Here, each view came with their own presenter that will handle all presentation logic, acting as a middle-man in the process between

model and view. Since model and view should not directly communicate with each other, having a presenter handling this task helps the system to achieve total independency. Additionally, in Figure 5-3, there are several services in the system. For communication, there are two services, Bluetooth and Location. Bluetooth service provides Bluetooth connection to the lock while Location service provides GPS location of the user. For any request from application to the server, API service is used. API service acts as a medium for server and mobile application to communicate. In the process, it gets response from server according to user's request.

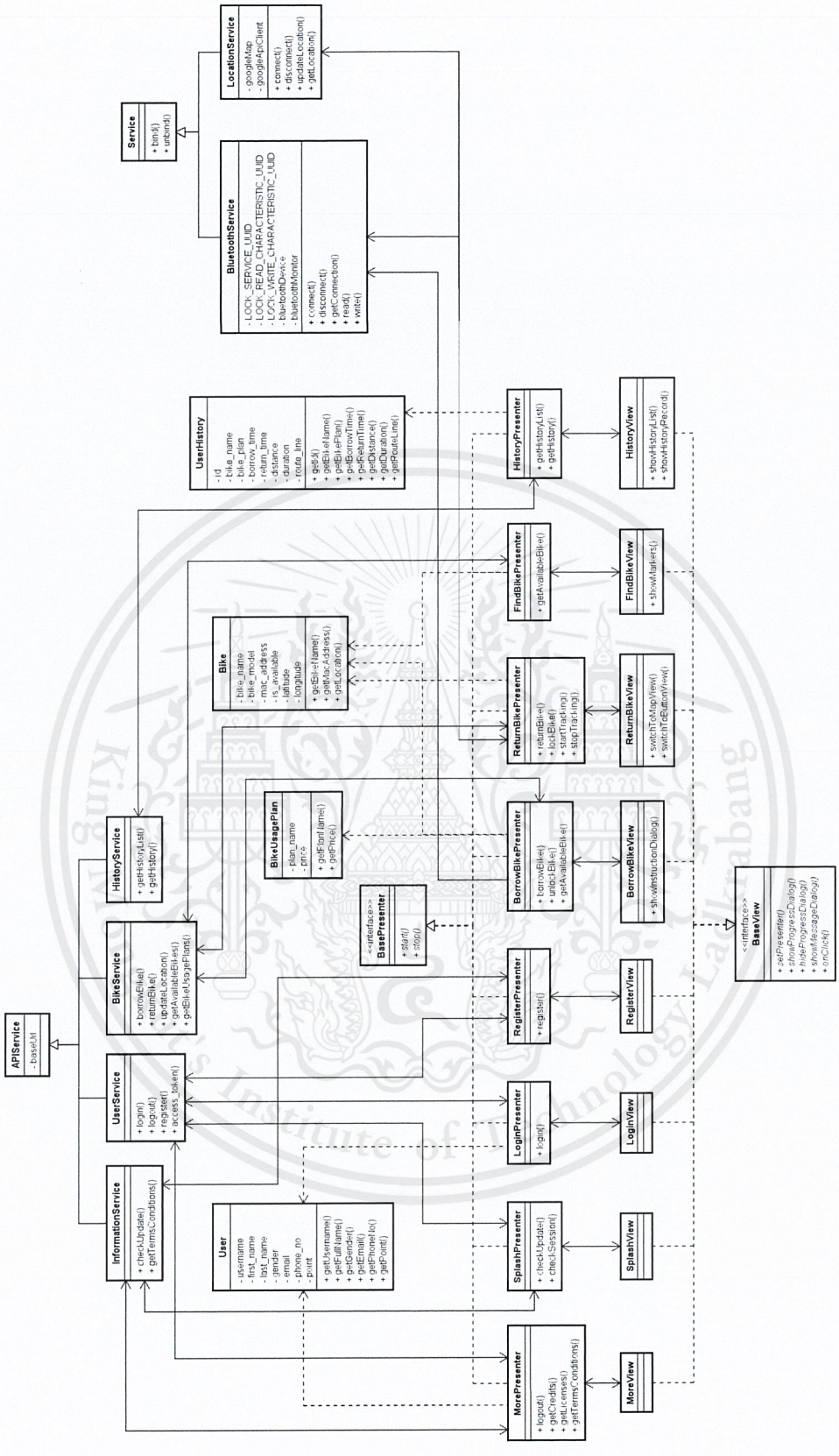



Figure 5-3: Class diagram of KMITL Bike mobile application

### 5.1.4 KMITL Bike Lock Unit

#### Hardware Architecture Overview

Figure 5-4 and Figure 5-5 show the major components of the lock unit and how they are wired. Here, Arduino acts as a central processing unit of the lock unit. It controls the connection of the Bluetooth LE module, toggles the solenoid, and checks the state of the proximity sensor.

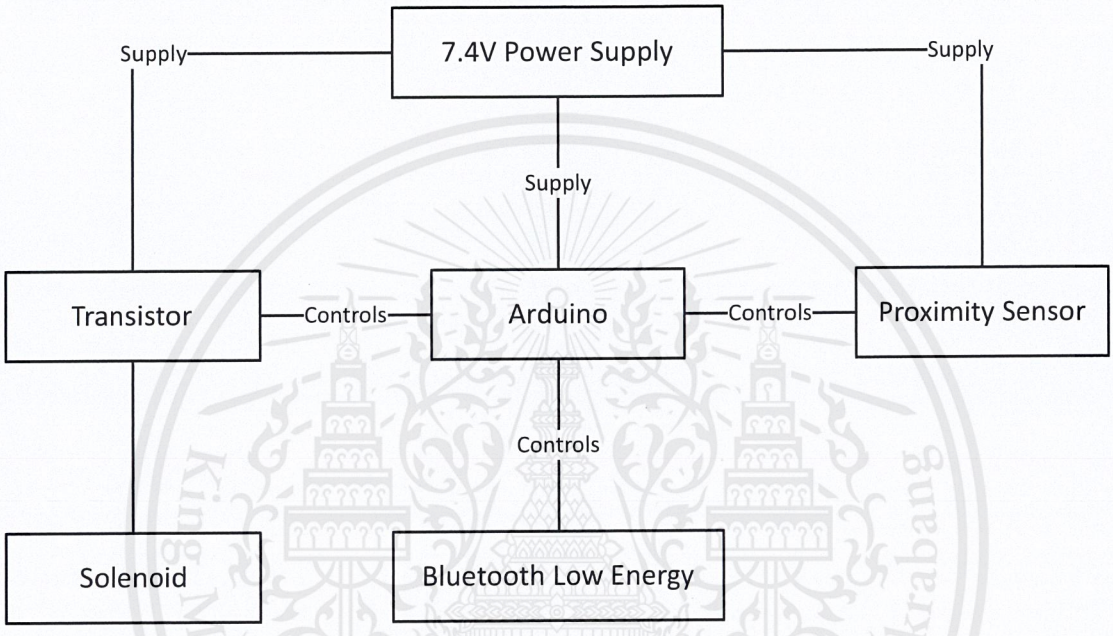



Figure 5-4: Simple block diagram of the Lock Unit

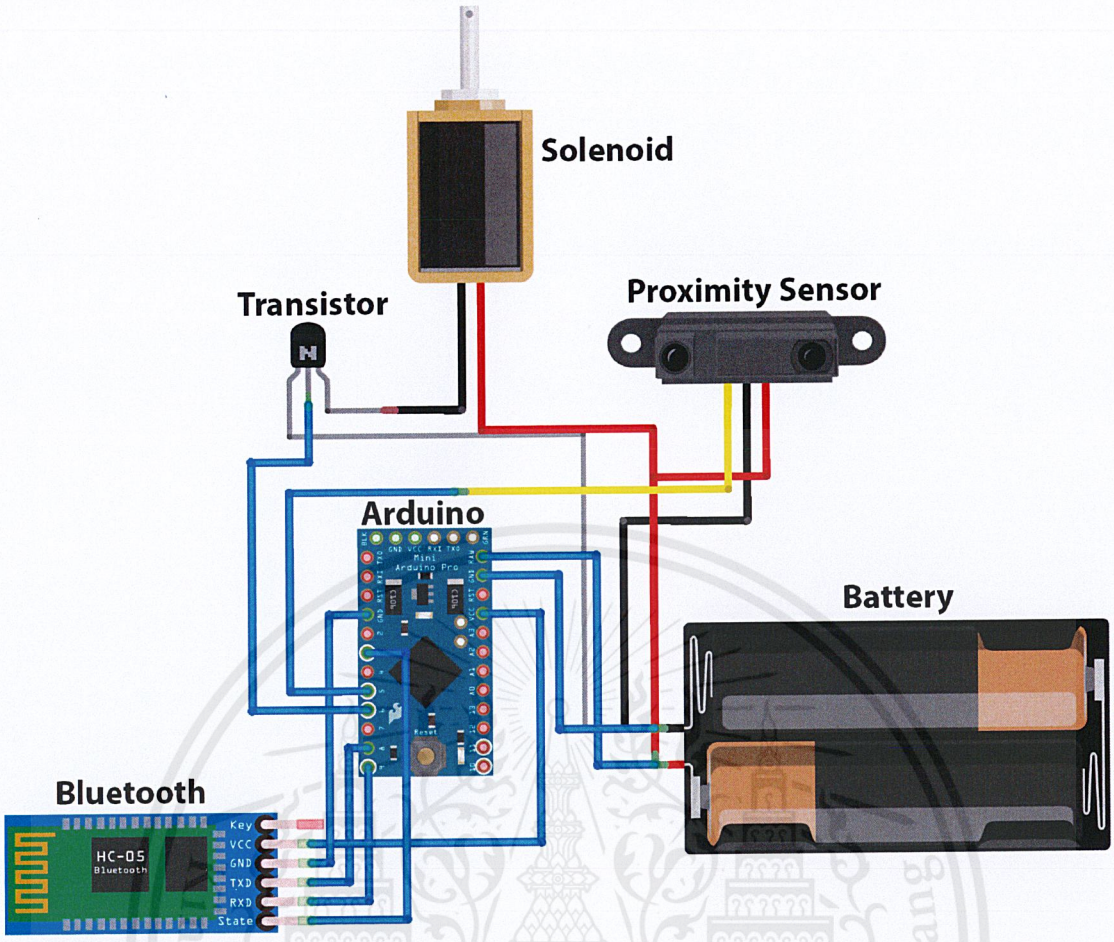



Figure 5-5: Block Diagram of the Lock Unit

### Hardware Class Diagram

The communication between each component can be further described using class diagram on Figure 5-6. During the communication between Arduino and mobile application via Bluetooth, all the messages are encrypted for security purpose. In order to unlock the lock, the application must correctly encrypt the command using the matching key pair with public key that is kept inside Arduino. If the process goes as intended, the Arduino decrypts the command and then signal solenoid to actuate corresponding to the received command. In case of returning, the Arduino reads the value of proximity sensor which reflect the state of the lock whether it is locked or not. It then sends the value to the application for the

application to process and decides whether user can return the bike or not.

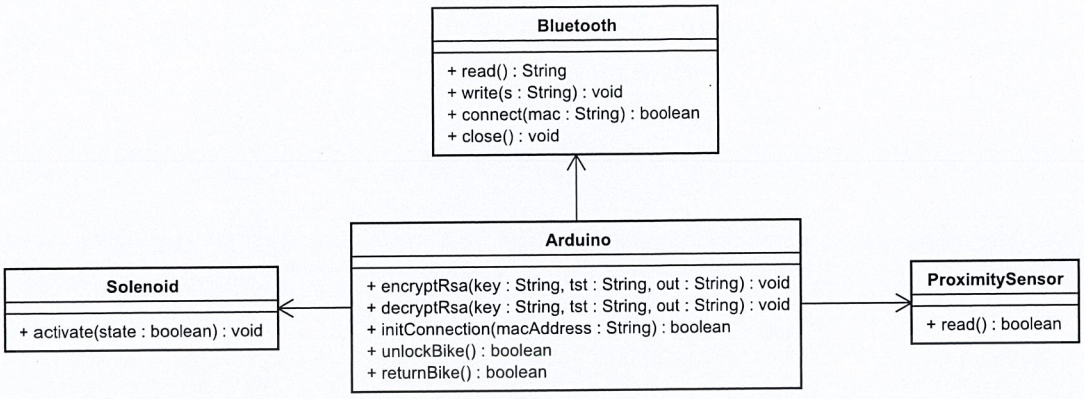



Figure 5-6: Simple class diagram of the lock unit

## 5.2 System Components

### 5.2.1 Server

Server serves as a bicycle management system. It consists mostly of two parts: database and service. The database collects information such as bikes locations, users' information, and usage sessions. The service checks for user's credentials, bike availability, and provide filtered information from the database to specific users.

### 5.2.2 Smartphone

Smartphone includes two types of operating system: Android and iOS. Since the lock unit does not contain any cellular network connection module (due to energy consumption), it cannot be directly connected with the server. Most smartphones are equipped with both Bluetooth LE and cellular network. This allows the lock to be able to communicate with the server via the smartphone. Smartphone also serves as the main interface for the user to interact with the system.

### 5.2.3 Arduino Pro Micro 5V/16MHz

Arduino is an open-source computer hardware which is widely adopted among hardware developers. It is usually served as a go-to hardware when starting out a project. Specifically, Arduino Pro Micro 5V/16MHz was chosen for the project due to its small footprint in terms of power consumption and physical size.

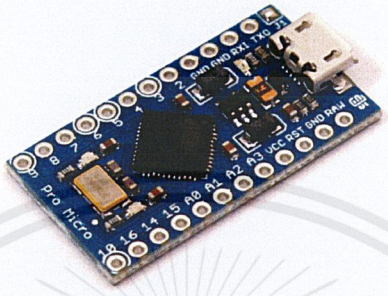



Figure 5-7: Arduino Pro Micro 5V/16MHz

### 5.2.4 HC-05 Bluetooth 4.0 Low Energy Module

HC-05 is a Bluetooth low energy (LE) module, primarily serves as a wireless communication between the Arduino and the smartphone. Bluetooth LE was chosen for its widely adopted standard for connecting with wireless peripheral devices as well as being energy efficient. A market study by IndustryARC Analysis reports that there could be over 8.4 billion units of Bluetooth LE shipped by 2020 [8].

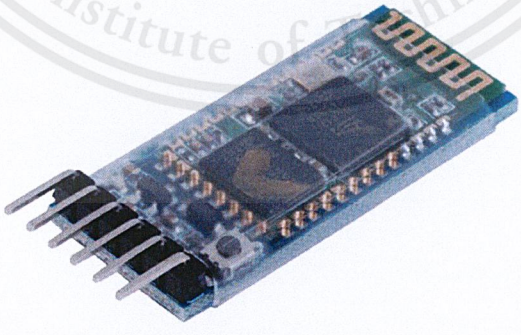


Figure 5-8: HC-05 Bluetooth 4.0 Low Energy

### 5.2.5 TAU-0826 Solenoid

TAU-0826 is a Pull-Type Linear Solenoid with a maximum keeping force of 20N while consuming only 6V. It is one of the core actuator used in the unlocking mechanism of the lock unit. The uniqueness of TAU-0826 is its small footprint of only 26 × 25 × 22mm while being powerful enough to pull the latch used for locking the lock's shackle which makes it compatible with the designed lock unit.


Figure 5-9: TAU-0826 Solenoid

### 5.2.6 TL-W5MC1 Inductive Proximity Sensor

TL-W5MC1 is an inductive proximity sensor capable of running on 5–36V DC. It is used for detecting various types of metal. Its sensing capability depends on the type of metal. Ferrous metals, such as iron and steel, allow for a longer sensing range, while nonferrous metals, such as aluminum and copper, can reduce the sensing range by up to 60 percent [9]. TL-W5MC1 has a detection range of 5mm according to the manufacturer specification. Its size of 30 × 18 × 10mm matches the size available in the lock unit.

Figure 5-10: TL-W5MC1 Proximity Sensor

# Chapter 6

# Development

This section elaborates the actual development processes of the project. The elaboration is divided into two main parts: the development tools and the development iterations.

## 6.1 Development Tools

The development tools used in this project are:

- Operating Systems: Android 4.4 KitKat (API Level 19), Ubuntu 12.04.5 LTS
- Programming Languages: C, Java, Python 2.7
- Database: PostgreSQL 9.3.16
- Integrated Development Environments: ARDUINO 1.8.2, Android Studio 2.3.2, PyCharm Professional 2016.3
- Libraries & Frameworks: Crashlytics, Django, Django REST Framework, MVBarcodeReader, Neatle
- Utility Softwares: Adobe Photoshop CC 2017, Adobe Illustrator CC 2017, Autodesk Fusion 360, MakerBot Desktop 3.10, pgAdmin III 1.22.2

## 6.2 Development Techniques

### 6.2.1 Securing Data and Communication with RSA and Cryptographic Nonce

In order to ensure a secure locking system, crucial communications between each party should be encrypted. In the current system, there are three main communication parties of the system: the Lock, the App, and the Server. However, there are several problems that can occur with ordinary symmetric encryption scheme since it requires the key to be completely secret. This means that all the three parties will have to be able to keep its source code which includes the key to be hidden. This is almost impossible since the application can be reverse engineered through the usage of decompilers. Android application, for example, can be easily decompiled by tools such as Android APK Decompiler [10]. Access to the source code also gives the attacker the access to the key. This is why the system should be using the asymmetric encryption scheme.

Since the most crucial commands such as locking and unlocking the lock are from the Server, Server will encrypt those commands with a private key. The message will then be passed on to the Lock through communication with the App, making the App to acts only a bridge connecting between the Server and the Lock. The lock will then decrypt the commands using a public key.

There is one more way an attacker could unlock the lock without the Server's authorization. That is after the attacker had legitimately unlocked the bike through Server's authorization, the attacker could record all communication that is being done between the App and the Lock and replays it later. This way, the Server will never know about the Lock being unlocked. Fortunately, the solution happens to be relatively easy, since the Server will just have to make sure that each message being sent will not be the same. The Lock and the Server could agree and add a nonce to the message which will change on every usage session.

To conclude, the usage of nonce will help prevent a replay attack and an asym-

metric encryption scheme will prevent the attacker from having complete control of the system after obtaining the key through means like code decompilation.

### 6.2.2 Underlying Communication in Unlocking Process

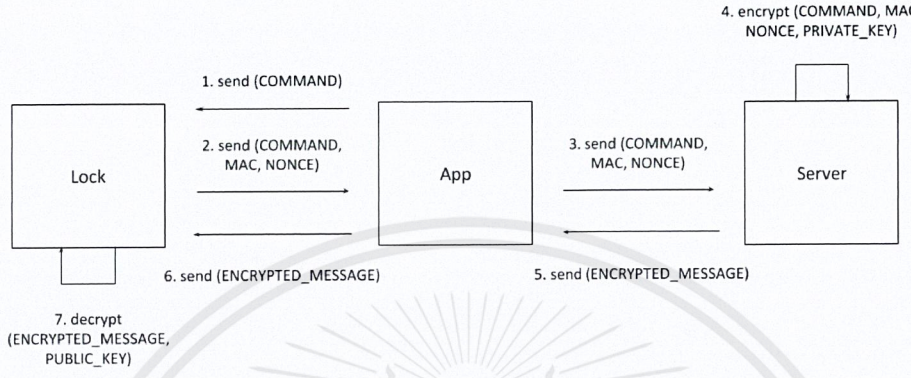



Figure 6-1: Overview of Unlocking Process

A flow of unlocking process can be described as shown in Figure 6-1. In order to initiate the connection between user's mobile phone and the lock, Bluetooth pairing procedure must be performed. To satisfy what was stated in the requirements, Bluetooth will broadcast itself at all time. During a broadcasting process, the bike will be visible to the user to unlock. When the user chooses the bike by scanning a barcode on it, the mobile application will request for unlock by sending a command to the lock. The lock will generate a message to be passed from the application to the server. After the server got the message, the message will be encrypted with a private key, that only known to the server, and pass back to the application. Once the application received the key from the server, user's mobile phone will connect to the lock and start discovering the services. After the process of discovery is completed, user's mobile can send a message to any characteristics available on the lock service as shown in Table 6.1. For the locking or unlocking process, the characteristic FFE0 is used as a destination for sending the unlock key that received from the server. If the key is valid then the bike will be unlocked and ready to ride. The procedure is also the same for locking the bike when the

user wants to return. However, the key validation is not required anymore.

Table 6.1: Bluetooth Characteristics Available on the Lock

| **Characteristic** | **Operation** | **Usage** |
|---|---|---|
| FFE0 | Read/Write | Send the command to the lock or unlock, Read the result after the command is executed |
| 180F | Read only | Read battery level of the lock |

### 6.2.3 Near Real-Time Location Tracking

Since the cost of 3G and GPS module is considerably high, the alternative way to track the bike is to use location service of user's mobile phone. By using mobile phone's Location service, the location of the bike, which represented as latitude and longitude, is updated to the central server every 10 seconds during the ride. The origin of this interval is based on an idea that the update interval should not be too frequently nor infrequently. If the update is invoked in short interval, it will consume data usage too much and if the interval is too long, then an action might not be taken in time in case the bike got stolen. Even if ten-seconds interval location tracking might not be real-time, but it is still acceptable as near real-time.

### 6.2.4 Searching Nearby Bikes on Google Maps

As the smart bicycle fleet does not require any stations to park, the bikes do not have a fixed position where the user can go there and grab them. If the user needs to use the bikes, then he or she will likely have to go through looking around the places to find them. For the convenience of the user, the mobile application provides the Google Maps view with the markers, which indicate the bikes' location, on it. The bikes that are already in use will not be shown to the user. With the find bikes feature, the user can easily locate nearby bikes and go there to borrow the bikes whenever he or she wants.

### 6.2.5 Activity Timeline for Bicycle Rental

In the rental business, keeping a track of user rental history is required. Each riding session contains an information of bike ID, route that the user takes, distance in kilometer unit, duration, and time-stamps when the user borrowing/returning the bike. The user can view his or her history, which included the information described above, via the mobile application.

## 6.3 Development Iterations

### 6.3.1 Software Development

**First Iteration: Bluetooth with Cable Lock (Hybrid)**

To fulfill the requirements stated in Chapter 3, the first version application was released. The application was implemented using Ionic, an open-source hybrid application development framework which can develop the native-like application made out of HTML, JavaScript, and CSS. However, as Ionic didn't provide native components like Bluetooth or Location, another library, which is called Cordova, was used here to add up a lack of these components. The tools provided in the framework were simple to use so there was not much difficulty in developing.

The usage of the application was straightforward. User can borrow the bike by selecting the preferred bike from the list appeared on the mobile application. This list came from scanning the perimeter for the application registered Bluetooth's MAC Address and filtering out any unrelated Bluetooth devices. Figure 6-2 shows the screenshots of the first version of KMITL Bike application.

**Features in this iteration:**

- Borrow bike
- Return bike
- Find bike

- View riding history (without route line)

**Frameworks & Libraries used in this iteration:**

- Ionic 1
- Cordova Bluetooth Serial
- Cordova Geolocation


Figure 6-2: 1st Iteration - KMITL Bike Application

**Second Iteration: Bluetooth with Cable Lock (Native)**

After experiencing some unknown issues with Bluetooth connection in the first application, it appeared that a hybrid application was hard to track down the issues that related to hardware components of mobile phone like Bluetooth or Location. As a response, the development of the application was completely changed from a hybrid application into a native application.

In this iteration, there was also an improvement to the design of the user interface to make it more intuitive and open for new features as shown in Figure 6-3, Figure 6-4, and Figure 6-5. Crashlytics is embedded into the app in order

to be able to keep track and assess crashes in real-time. Additionally, some new features were added to make the application more suitable for real-world usage.

**Features in this iteration:**

- Borrow bike
- Return bike
- Find bike
- View riding history
- Near real-time tracking
- View user profile

**Frameworks & Libraries used in this iteration:**

- Android Native Libraries
- Crashlytics

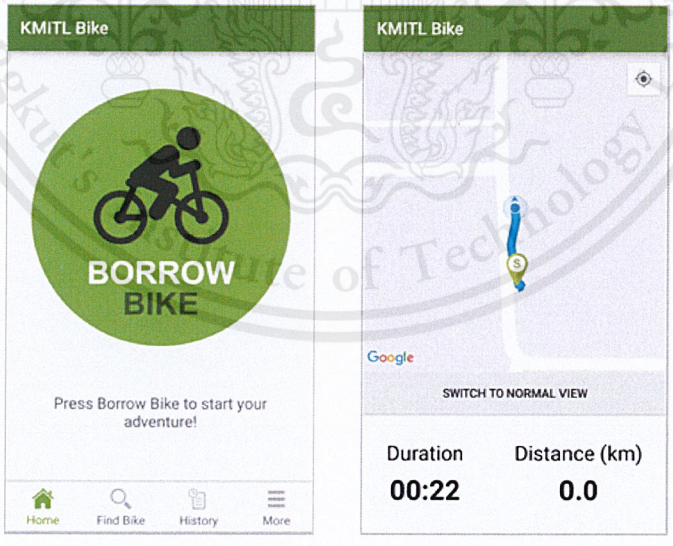



Figure 6-3: KMITL Bike application - 2nd Iteration

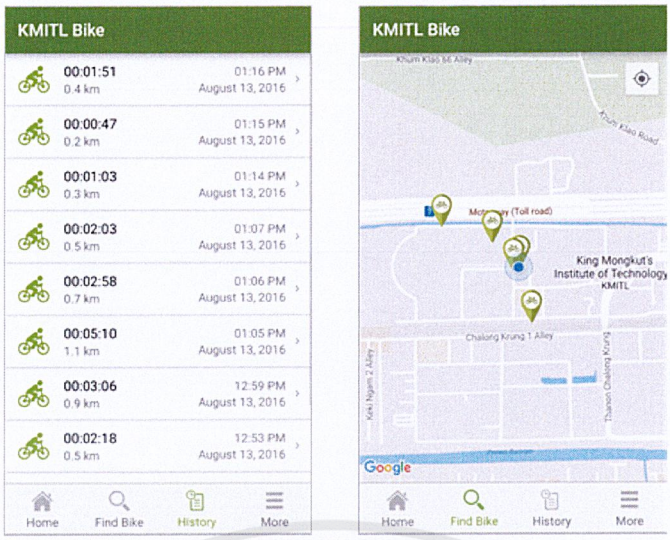



Figure 6-4: KMITL Bike application - 2nd Iteration (continue)

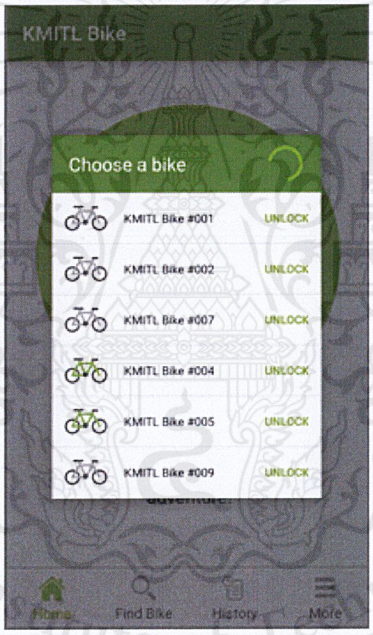



Figure 6-5: User can choose a bike from the list

### Third Iteration: Barcode Scanner with Passcode Lock (Native)

As a durability of the cable lock design in Section 6.3.2 was not enough to withstand the real-world environment (e.g., not rain-proof, internal circuit components could not handle rough terrain), the locking mechanism needed to be revised before deploying to an actual use. However, it was also necessary to conduct a test

flight for the purpose of collecting information. Therefore, a temporary solution was raised and it was to use manual passcode lock for a certain time along with barcode authentication. After the user scans the barcode on the bike, the corresponding passcode for the specific bike will show up and can then be used to unlock the lock. Not only that, using the barcode could indirectly prove the presence of the user near the bike as well.

**Features in this iteration:**

- Borrow bike
- Return bike
- Find bike
- View riding history
- Near real-time tracking
- View user profile

**Frameworks & Libraries used in this iteration:**

- Android Native Libraries
- MVBarcodeReader
- Crashlytics

**Fourth Iteration: Bluetooth with Semi-automated Lock (Native)**

After the run with cable lock and barcode scanning system, there were few observable good points. Barcode scanning successfully verify that the user was in the vicinity of the bike and prevent long distance borrowing. Since the Bluetooth module used in this project could receive connection as far as 10 meters, this system could cover this loophole [11]. However, the newly designed lock required Bluetooth connection, thus, barcode was combined with Bluetooth schema to perform

without loophole. Furthermore, in this iteration, there were minor improvements on the stability issues found while testing as well as Point System implementation. The Point System served as the protection to bike abuse problems. If any user misused the system in any way listed in the term and conditions, their credits will be deducted accordingly. If their credits were to reach zero, they may not use the system anymore.

**Features in this iteration:**

- Borrow bike
- Return bike
- Find bike
- View riding history
- Near real-time tracking
- View user profile
- Point system

**Frameworks & Libraries used in this iteration:**

- Android Native Libraries
- MVBarcodeReader
- Neatle
- Crashlytics

### 6.3.2 Hardware Development

**First Iteration: Cable Locking System**

While the goal of this project is to deploy the locking system and software system at the same time, the resources and time for them are not equal. The locking

system took a longer time to finalize the model design and create a final product whereas the software could be tested almost immediately after completion. As a result, it was better to create a mock locking system to support the finished software. In this first iteration, a metallic box and a cable were used as a lock. This lock will be placed on top on the back wheel of the bicycle and the cable is used to hold the wheel in place, restricting the ride. A test was conducted to prove the usability of the planned system. After the system went on two events: KMITL Science Exhibition Day 2016 (Figure 6-6) and Engineering Expo 2016: Engineering Innovation with Thailand 4.0 (Figure 6-7), the lock showed few issues. The lock could not detect if it was locked or not, which showed its vulnerability. Additionally, it was too fragile to external forces while riding and transporting. Fortunately, the software testing went well.

Figure 6-6: HRH Princess Sirindhorn
at KMITL Science Exhibition Day 2016


Figure 6-7: Team at Engineering Expo 2016

**Second Iteration: Automated Lock**

Improving on the previous iteration, a fully automated lock controlled by a mobile application via Bluetooth connection was designed. This lock had a button as a sensor for detecting locking state, which countered the issue with the first lock. As for the internal mechanism, the locking shaft was pushed and pulled by a horizontal gear driven by a motor, holding and freeing the back wheel of the bicycle respectively. Figure 6-8 below depict the design of this lock. This design was presented to the committee and was, unfortunately, doubted. There were issues with a gear-driven mechanism, since it was prone to dust and could not handle forceful opening from users. In the end, this lock was found faulty before it was produced.

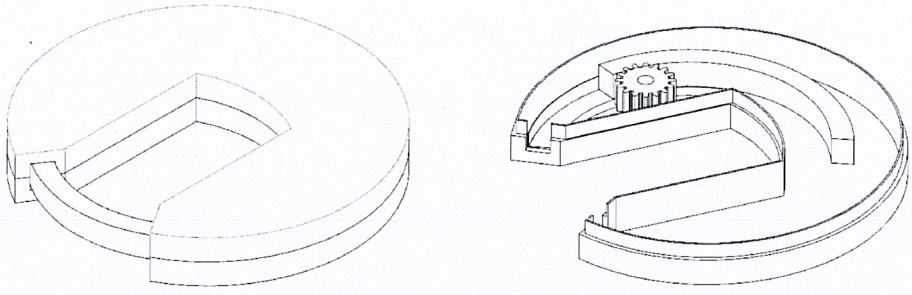


Figure 6-8: Locking Mechanism Design - 2nd Iteration

**Third Iteration: Semi-Automated Lock with Teeth-Based Mechanism**

To fix the previous issues with gear-driven mechanism, a new design using teeth-based mechanism was modelled. This model can be seen on Figure 6-9. The teeth on the locking shaft can only move one way through the teeth on the lock package, which is the act of pulling the locking shaft to hold the wheel in place. This is to ensure that at any moment, the lock can be locked without any need for battery. However, the unlocking process still require the battery to pull the teeth of the lock package away from the teeth of the locking shaft, creating a gap for the locking shaft to be pulled back by spring to its origin, freeing the back wheel of the bicycle. The teeth will be pulled by a servo's spinning motion to one side. Overall, it is a semi-automated process where user can unlocking the bike automatically using mobile application while manually locking the bike to handle security issues if in any circumstances the battery is dead.

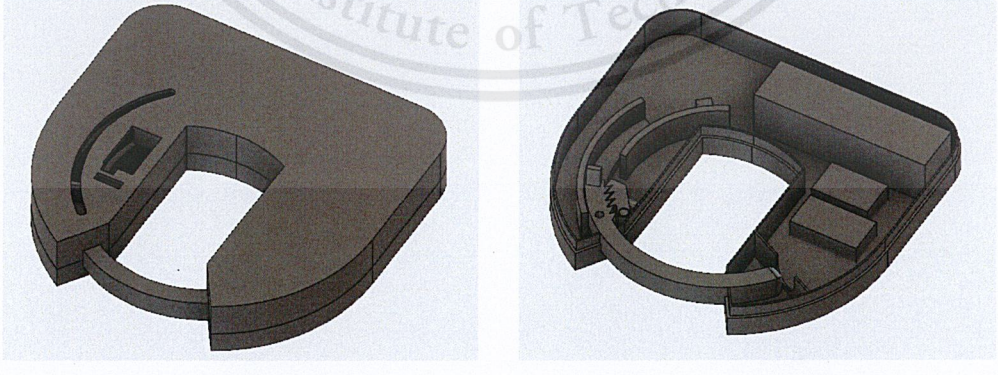


Figure 6-9: Locking Mechanism Design - 3rd Iteration

**Fourth Iteration: Semi-Automated Lock with Sliding-Latch-Based Mechanism**

Initially, the previous model appeared to be very promising. However, after the talk with Department of Mechanical Engineering at KMITL, there still were few stability issues with the lock. The choice of actuator for the locking system, the servo, was not suitable for harsh working environment such as constant shaking of the lock when riding on a rough surface. Moreover, the button that was used as a sensor to detect if the lock was successfully locked was unreliable. The right concept of designing a locking system was to involve as less mechanical parts as possible. The button required mechanical movement of the button surface to be contacted, which had high error rate depending on the amount of force and angle in contact. Not only that, the teeth-based mechanism could not hold on due to spring deformation and could be forcefully broken still. Therefore, a new model was designed and put through the prototype production. This model replaced the former mechanism with sliding-latch mechanism combined with solenoid to actuate it. It also used proximity sensor to remove the mechanical part of the button. Figure 6-10 represents the model of the lock.

Figure 6-10: Locking Mechanism Design - 4th Iteration

**Finalized Design: Semi-Automated Lock with Sliding-Latch-Based Mechanism (Optimized)**

The design presented in the fourth iteration is adequate for the basic functionality of the lock unit. However, throughout the development of second, third, and fourth iteration design, there are only testings on 3D printed plastics which are too fragile for real-world usage. In order to achieve full functionality testing, the lock unit needs to be made out of a stronger material such as aluminum. After discussions with several CNC machining factories, the design needs to be changed drastically to allow for the machining process possible at an appropriate cost and be more user-friendly while also concerning light raining conditions that might affect the inside electronics. Figure 6-11 represents the final design of the lock unit with its external casings made from aluminum 6061 (bottom piece which is shown on the right) and 6063 (top cover which is shown on the left). Aluminum 6061 and 6063 allows for high strength, good workability, weldability, and corrosion resistance of the lock unit package.

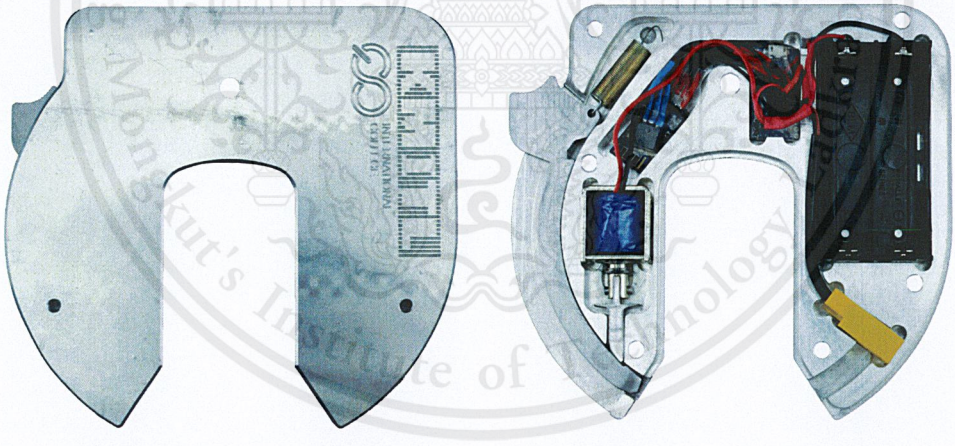


Figure 6-11: Locking Mechanism Design - Final Design

# Chapter 7

# Results and Evaluations

## 7.1 Experimental Method

The experiment for this system was initiated through a series of test flights. This test flight was conducted within a section of King Mongkut's Institute of Technology Ladkrabang campus. An application on both Android and iOS was implemented for the experiment under the name KMITL Bike. This application involved solely on the usage of bike as the focus of the test was on the bicycle aspect of the InfiniLock for the moment. The purpose of this is to gain feedback from users and use it to improve the system while promoting Green Campus campaign which is the underlying goal of the whole system.

## 7.2 Test Flight 1: The Beginning

In this test flight, the system was tested with actual users for its reliabilities, usabilities, and validity and lasted from January 18, 2017 to February 8, 2017 for a total of 21 days. However, due to the delay in hardware development, a substitute for the lock was necessary for this period of time. In the end, passcode lock was chosen for testing.

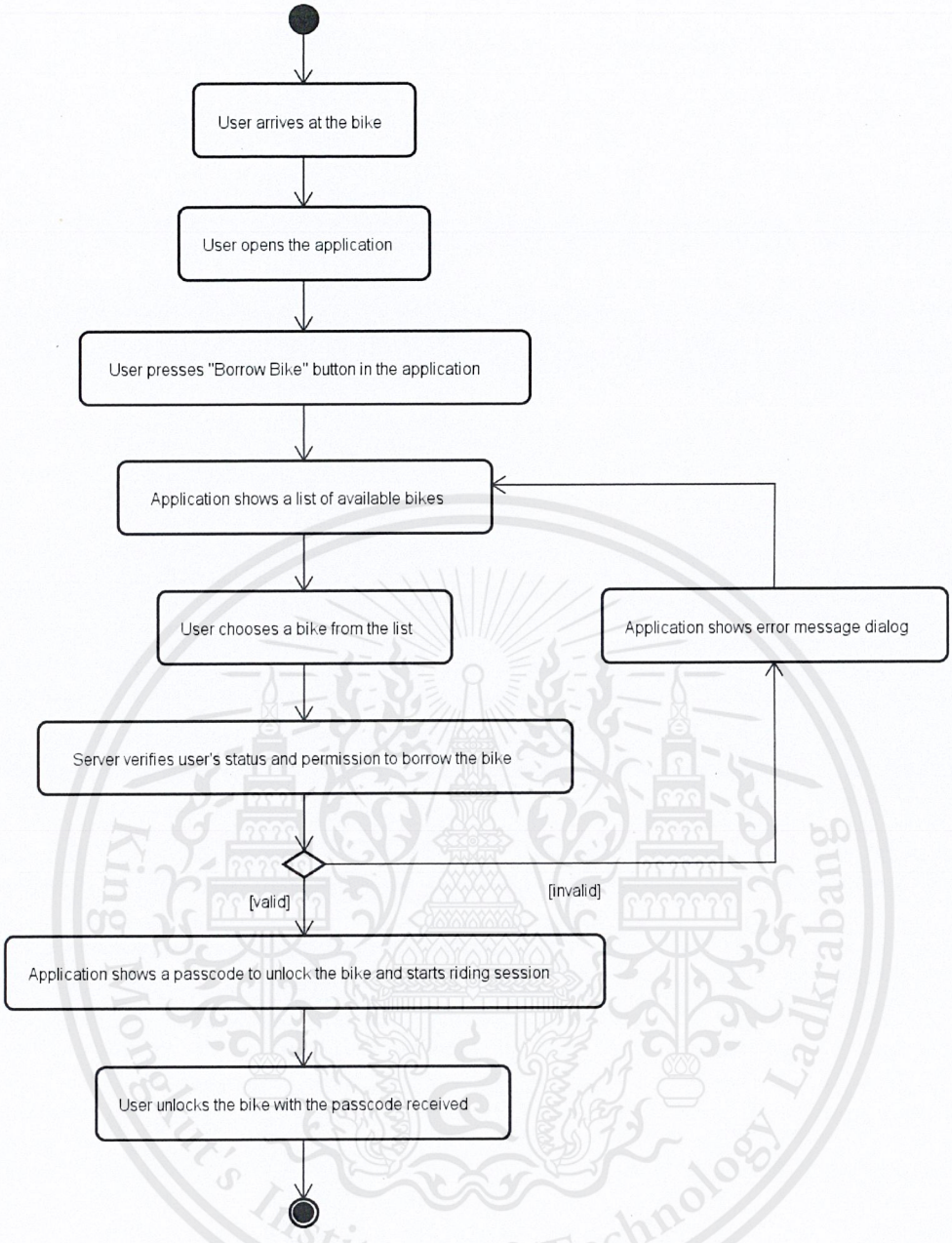



Figure 7-1: Borrowing a bike in TF1

The test flight was launched with two different types of bike: a city and commuting model. Table 7.1 lists the details of these models. The pictures of bikes can be seen in Figure 4-5 and Figure 4-6. The reason for the different types of bike is to accommodate both commuting and shopping for students and staff in KMITL. After the test ended, the number of usage in each type of bike will be counted and determine the ratio of future purchase for the bike.

Table 7.1: Bike Types

| **Model** | LA City Green | GIANT Escape 3 |
|---|---|---|
| **Type** | City Bike | Commuting Bike |
| **Size (Seat Tube)** | 16" | XS/15" and M/19" |
| **Number of seats** | 2 | 1 |
| **Price (THB)** | 3,500 | 8,800 |
| **Amount in service** | 3 | 5 |

Currently, KMITL students have two account used for access several institution facilities: generation 1 and 2 account. As for the authentication process of the system, users can use their KMITL account, both generation 1 and generation 2 account, to directly connect to the application. That way, users could be verified if they are actually students or staff in the university or not.

Figure 7-2: Picture of LA City Green bicycle

Figure 7-3: Picture of GIANT Escape 3 bicycle

### 7.2.1 Results

The graph in Figure 7-4 represents the number of application downloads from January 18, 2017 to February 8, 2017. On the first day, the application starts off with 43 downloads. The day after, the application was promoted in the "KMITL Green Campus" page on Facebook, causing the number of downloads to slightly increase until it reaches the peak at 163 downloads in January 20. After that, the number sharply drops and fluctuates moderately throughout the rest of this Test Flight. Overall, the total amount of downloads grows linearly.

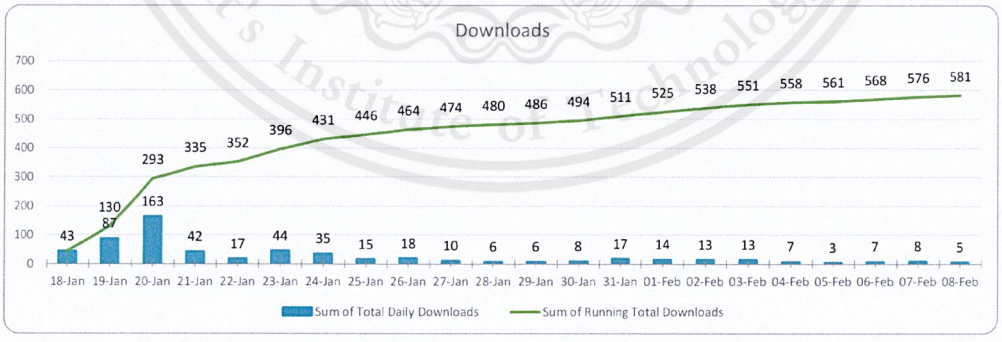



Figure 7-4: Number of downloads during TF1

Similar to the previous graph, the graph in Figure 7-5, which indicated the number of users, portrays the same growth. However, the actual numbers are

different. The number of Figure 7-5 is less than that of Figure 7-4 due to some users who downloaded the application out of curiosity might not belong to KMITL community.

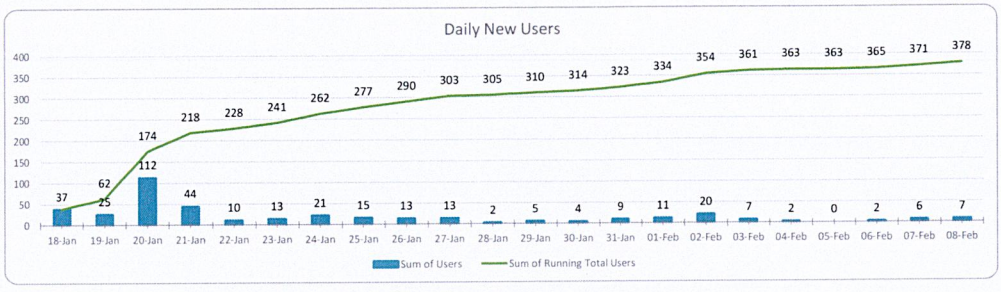



Figure 7-5: Number of users during TF1

As for the session, Figure 7-6 represents the usage of the application in each day during this Test Flight. The information between January 18 and January 24 is not available due to technical issues in the server that caused the loss of information in the database. The number of sessions during those time is approximately around 150 sessions. Nevertheless, the information in hand is still sufficient to arrive at an appropriate conclusion. It is possible to deduce from this graph that the day of week hold an impact on number of sessions as they are especially low on weekend. There are even a day on weekend with number of sessions as low as 5 on Feb 5.

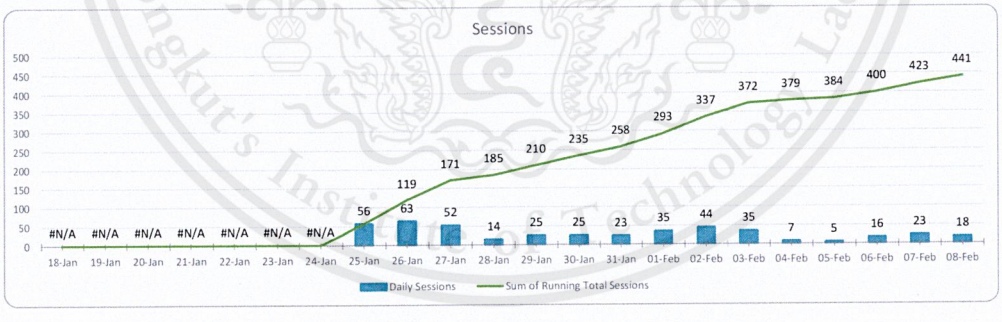



Figure 7-6: Number of sessions during TF1

## 7.3 Test Flight 2: Barcode

After gathering feedback and improving system upon the last test flight, the system was relaunched on February 17, 2017 until May 11, 2017. Few changes were made

for this test flight and all in regard to difficulties and incidents in the previous test.

With generation 1 accounts fading out, users registered with those will not be able to log into the system in the future. As a result, they were required to register again with the system using generation 2 account. Correspondingly, the system now limit new registration to only generation 2 accounts. However, since there still were unfamiliar users with generation 2 account, they will be prompted with texts and link to guide them through the steps to create a generation 2 account before creating an account for the system. Figure 7-7 compares the difference between previous test login screen and this test login screen.


Figure 7-7: Comparison between the login screen in TF1 and TF2


Figure 7-8: Sample of bike barcode

For location accuracy problem, barcodes were implemented in replacement to selection from bike list. Figure 7-8 shows a picture of the barcode that was on

one of the bike. Prior to this, user could potentially borrow the bike without actually being near the bike, causing the bikes' locations to be mislocated. Figure 7-9 illustrates the new flow of user borrowing the bike with barcode schema implemented.

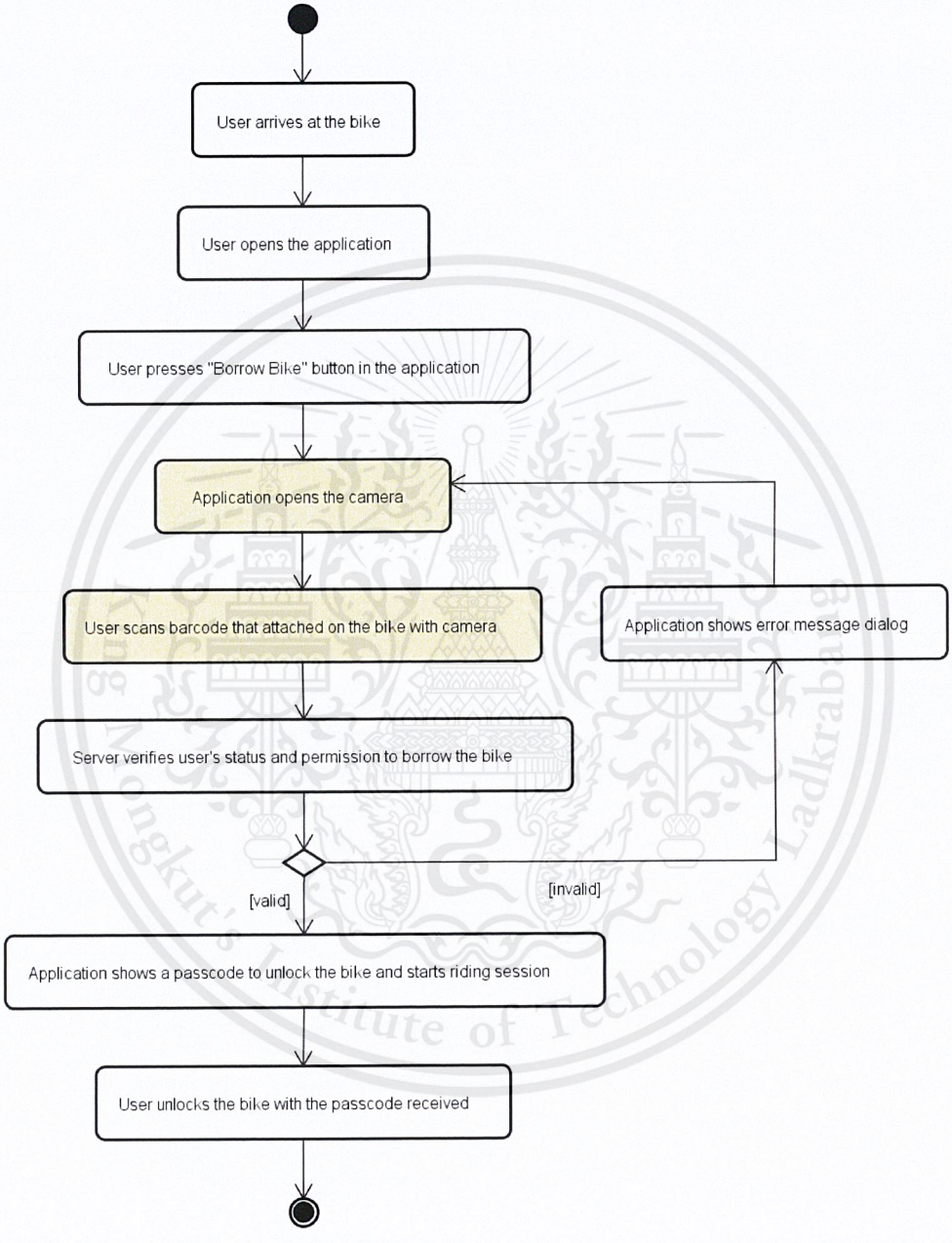



Figure 7-9: Borrowing a bike in TF2 (yellow indicates new changes)

At first, the solution for this problem was to filter users' location to those within a limited area around the bike. Unfortunately, due to inaccuracy of mobile

phones' GPS, it may cause unexpected event where users may not be able to borrow the bike that is in front of them. Thus, having barcode will prevent such an event from happening as well as verifying that user is in vicinity of the bike in order to borrow it. Figure 7-10 display an instruction dialog from the application, which is a change from bike selection screen in Figure 6-5.

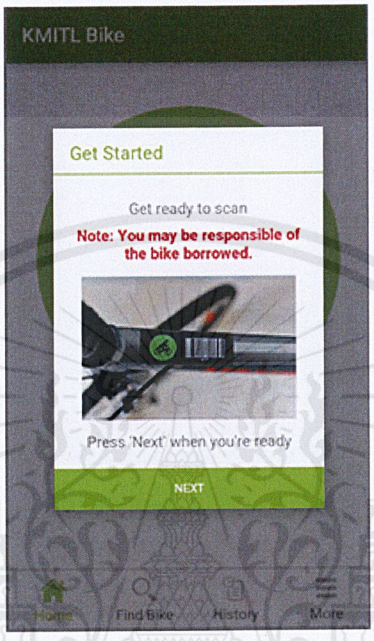



Figure 7-10: Instruction dialog in TF2

As for incidents where users rode the bike outside the specified area, terms and conditions and warnings were given in the application to restrict those kind of act. This will allow bike maintenance teams to easily take care and retrieve the bikes.

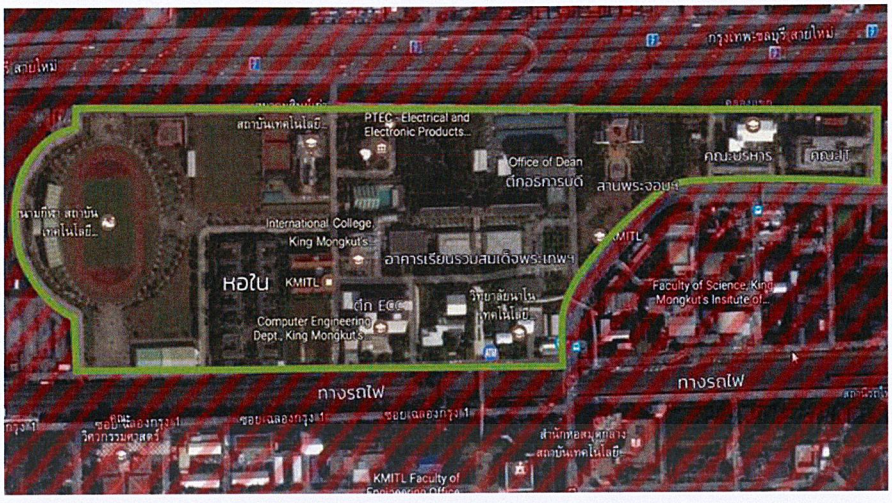



Figure 7-11: Test Flight Area

### 7.3.1 Results

The graph below in Figure 7-12 describes the number of downloads during Test Flight 2 which were held from February 17, 2017 to May 11, 2017. Continuing from the first Test Flight, the number of downloads stays low with a little oscillation of ups and downs. The highest amount of downloads are on February 20 with 10 downloads, which are considerably scarce compared to the initial launch of the first Test Flight. The reason behind this could be that daily downloads had reached its saturation point.

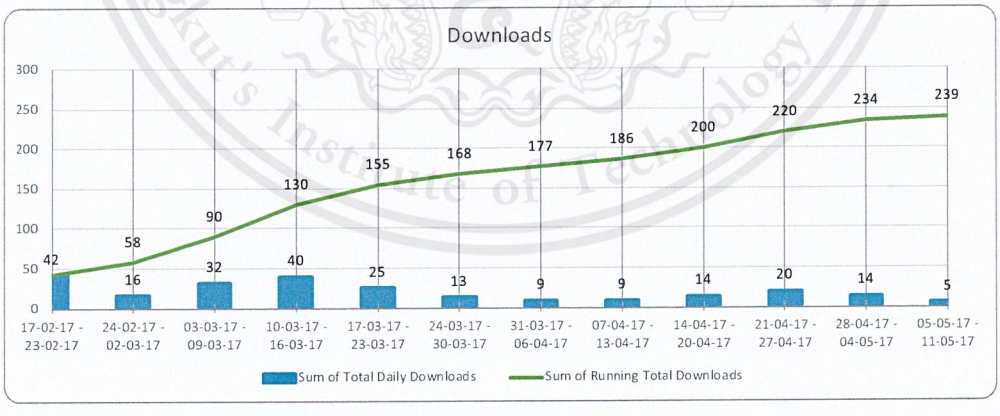



Figure 7-12: Number of downloads during TF2

Coincidentally, the graph in Figure 7-13 shows resemblance to Figure 7-12. The growth of both graphs are linear with identical slope. The actual numbers are

unexpectedly similar as well. Still, there are few users with multiple accounts with different ID from both Generation 1 and Generation 2 KMITL account mixed in the count, so the numbers could probably be lower. Overall, there is no significant change from the first Test Flight.

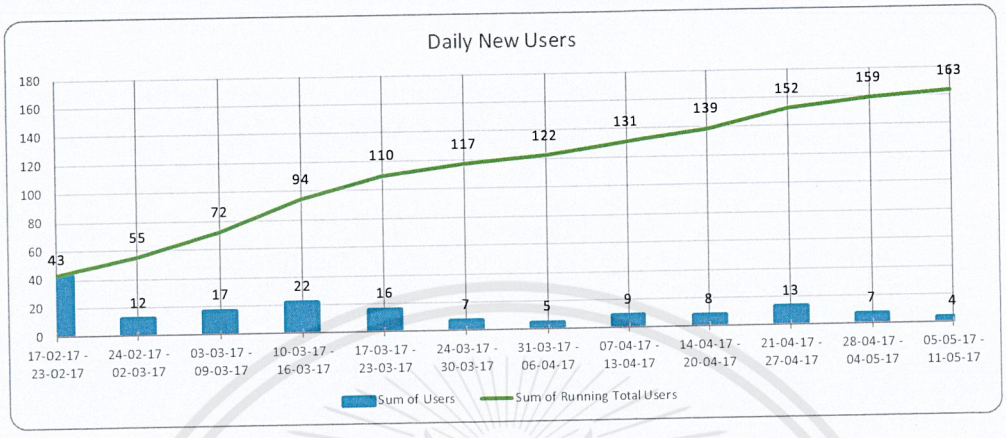



Figure 7-13: Number of users during TF2

Regarding the daily usage of the application during Test Flight 2, the graph in Figure 7-14 provides similar information to Figure 7-6 of first Test Flight except that the number is lesser. Nonetheless, the number of sessions appears to be higher than the number of downloads and the number of users, since there are regular users who use the service from time to time.

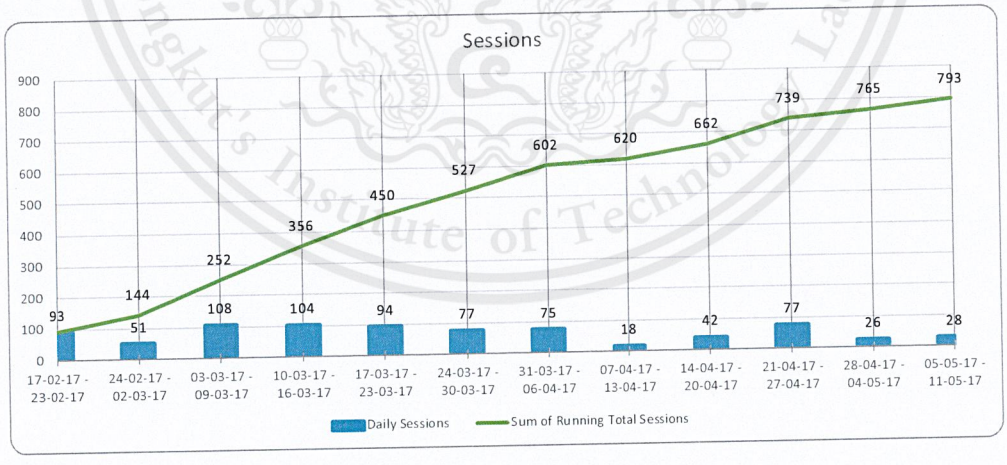



Figure 7-14: Number of sessions during TF2

## 7.4 Overall Discussion

Lots of data have been gathered during the two Test Flights. This section summarized the data since the beginning of the project until the end of Test Flight 2 (January 18, 2017 – May 11, 2017).

### 7.4.1 Downloads, New Users, Sessions

In the first week, the system received a tremendous amount of interest from the several announcements on social media and posters throughout the testing site. However, from Figure 7-15, it seems that the system have reached its saturated point in the first week making the downloads in the later weeks to be relatively low when compared to the first week. As for the amount in Figure 7-16, most of the new users appeared during the first week then leveled off in the following weeks for the same reason. Figure 7-17 shows the usage sessions of the application per week. It can be deduced that a lot of people were hype about the system in the beginning then began to loose interest later. Additionally, problems of inaccurate location of the bicycle also leads to frustration and unsatisfied users. It is also important to note that prior to the barcode schema implemented in Test Flight 2 it is very likely that users press borrow and return bike without actually using it as well. This is the underlying reason behind misleading high amount of usage before Test Flight 2.

Please note that the system was closed for maintenance during February 9, 2017 – February 16, 2017.

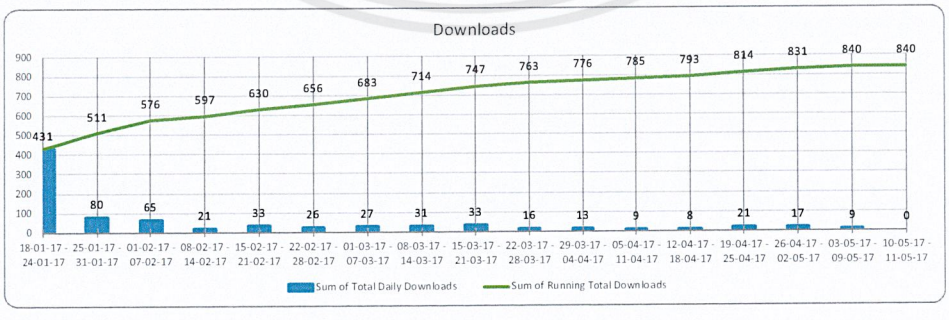



Figure 7-15: Summary of Downloads

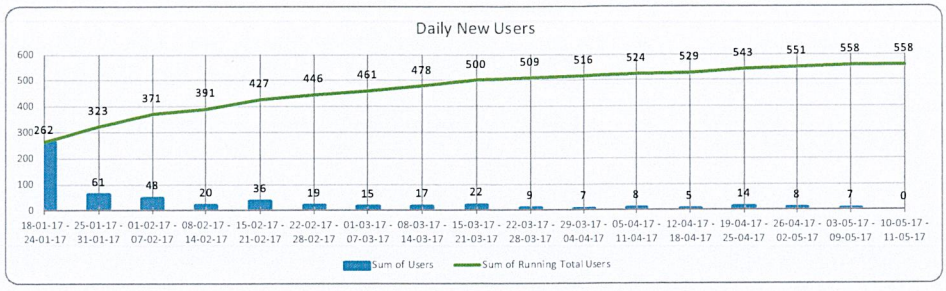



Figure 7-16: Summary of New Users

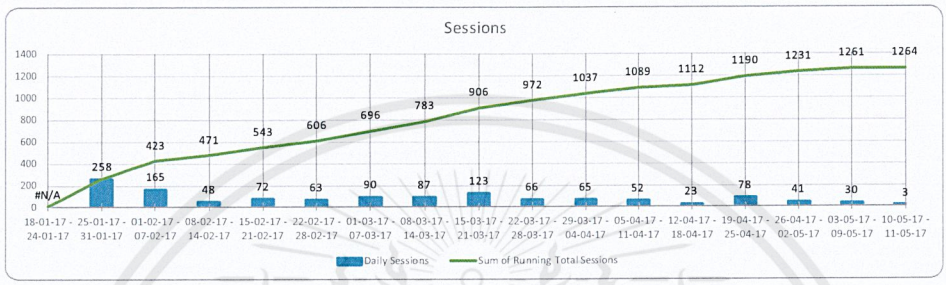



Figure 7-17: Summary of Usage Sessions

### 7.4.2 Users' Preferences

Since the launch of KMITL Bike, there are over 1200 sessions of borrowing bicycles. It is vital for the team to understand users' preferences in order to further improve the system towards users' demands.

Figure 7-18 and 7-19 visualizes the usage of each type of bike categorized by their gender. Figure 7-18 shows that over 78 percent of male users use the Escape 3 bikes while only 60 percent of female users use them. Two possible explanations are:

1. The design of Escape 3 bikes as seen on Figure 7-3 contains a top tube which make them difficult for female users wearing skirts to get on the bike.

2. Female users prefers a two seats model. From the team's observation, LA City Green bikes were usually rode by two passengers.

The reason why female users use more Escape 3 might be because of the lower availability of LA City Green bikes.

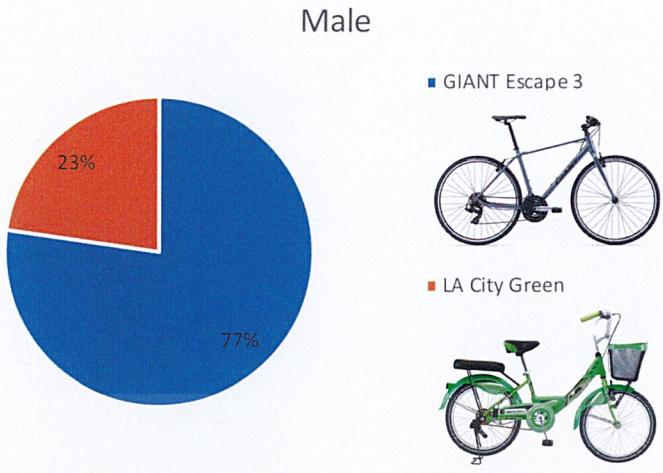



Figure 7-18: Male usage of different bike

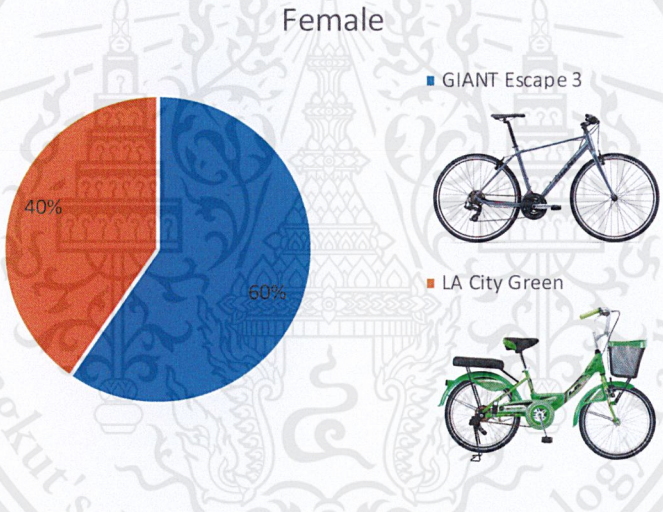



Figure 7-19: Female usage of different bike

Observing from Figure 7-20, the top three users were from the Faculty of Engineering, International College, and Faculty of Science, respectively. It could be inferred that the high usage came from the short distance between the main deployment area of the system. Still the distance is not the only factor for the difference in usage. With College of Data Storage Innovation and College of KMITL Nanotechnology being considerably near the testing site, it still had comparatively low usage to International College of the same distance. There seemed to be other

influences on the usage. It was probably due to the size of each department as well, since those two departments had lowest amount of students among all other departments.

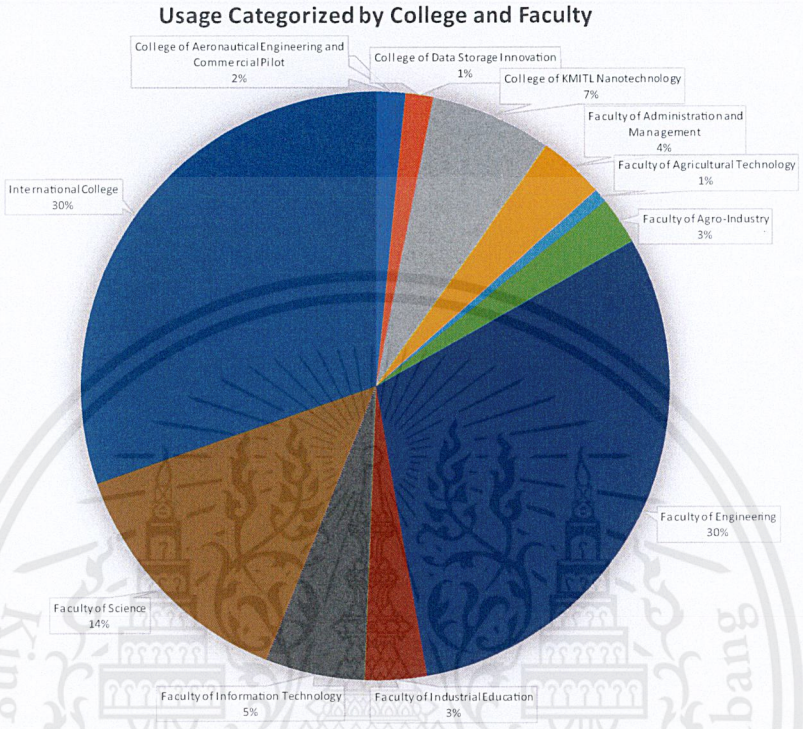



Figure 7-20: Usage by College and Faculty

# Chapter 8

# Conclusion

## 8.1 Summary

KMITL Bike was proven to be a difficult task when introduced into real-world testing. Several bugs and glitches were quickly found in the system since the first week of testing as there were over 150 sessions estimated usage sessions in just 6 days. Many students and faculty members from different faculties all over the institution showed their interest in the service. Several students post on social media and blogs review and feedback of the system. As of May 11, 2017 there are a total of 840 downloads, 1264 sessions, and 558 accounts from Test Flight 1 and 2 proved that KMITL students and faculty members are interested in the KMITL Bike service. However, due to technical difficulties the next test flight with the new lock unit will not be able to happen within the time of the submission of this document. This project can be considered a successful run.

## 8.2 Problems and Lesson Learned

Throughout the development of the project, there are several issues along the way that cause the project to take too much time than what was expected. In the process of solving them, there were lessons taught along the way.

### 8.2.1 Inexperience

Most of the problem are caused by the low experience in many new fields. For example, modeling 3D models poses a challenge for a team that are all Software Engineering students. However, trying to follow standard coding conventions and design patterns for better future code maintainability and scalability also requires going through several books and examples. Overall, time management must be done fairly well to cope with the inexperience in this field.

### 8.2.2 Design for Real-World Use

Designing the app for using by the public with different platforms proves to be a challenge. Each platform has a different design language (i.e., Android uses Google's Material Design and iOS uses Apple's iOS Human Interface Guidelines). It is essential to make the app intuitive for the users of each platform to follow the design language and guidelines set for their platform. Aside from following the guideline, users' feedbacks are also important and fixes are made accordingly.

### 8.2.3 3D Prototyping

In the hardware development process, over fifty prototypes were produced. This could not be achieved without a 3D printer. However, that does not mean it is nontrivial. 3D printing technology is not as matured as traditional 2D-based printing. It is not possible to select all the models and print them all at once. Each model must be exported to a printable file separately. The 3D printer's speed and temperature of both the nozzle and printing bed must be manually set and tweak occasionally in order to print some specific models. It takes several attempts of trial and error in the beginning and ever so often tweak for a specific model. Printing speed is also not that fast to ensure minimum error. It took at least 8 hours to print one prototype package which takes up time.

### 8.2.4 Moving to Metal

3D printing is a technology that was taken for granted. It is capable of printing complex shapes and sharp edges, something that traditional milling could not. This raises a problem when changing from plastic model to metal model. To lower the cost of CNC milling, the design needed to be drastically changed to have no sharp or narrow corners and take as much infill as possible.

### 8.2.5 Finding & Ordering Parts

Due to the limited space available to put the lock on the bike, the lock size must also be considerably small. This cause components in the lock unit to be hard to find due to the small size. In a case of the solenoid used in the latest prototype, the part was brought online and shipped from China which took several weeks to arrive.

## 8.3 Future Work

Regarding the aspect of the lock unit in KMITL Bike, here are some of the things that can be improved:

- Display status on the lock unit for diagnostic
- Increase battery life
- Integrate alarm buzzer to alert user and surrounding in case of theft
- Integrate real-time GPS tracking
- Optimize rainproof sealing and design
- Optimize design for mass production (e.g. reduce size, weight, and cost)
- Use dynamo to recharge the unit

Regarding the aspect of the application service in KMITL Bike, here are some of the things that can be improved:

- Add user feedback report
- Add user summary usage dashboard (e.g. social media related features, user's fitness level)
- Add ride history statistics (e.g. show graph of usage per month, show usage heat map)
- Add system near real-time statistics report generator in the backend system for system administrator
- Enable user to use the bike outside of the campus while ensuring that the user will always secure the bike and return the bike within the service area
- Increase service area
- Refactor code to Model-View-Presenter based design

# References


[1] C. Tang and H. Lean, "Will Inflation Increase Crime Rate? New Evidence from Bounds and Modified Wald Tests", Global Crime, vol. 8, no. 4, pp. 311–323, 2007.

[2] M. Ceccarelli, International Symposium on History of Machines and Mechanisms. Dordrecht: Kluwer Academic, 2004.

[3] Medani, A., Gani, A., Zakaria, O., Zaidan, A. A., Zaidan, B. B., "Review of mobile short message service security issues and techniques towards the solution", Scientific Research and Essays, 6(6), 19. doi: 10.5897/SRE11.107, 2011.

[4] P. Sawer, "The unsung genius who secured Britain's computer defenses and paved the way for safe online shopping", Telegraph.co.uk, 2016. [Online]. Available: http://www.telegraph.co.uk/history/12191473/The-unsung-genius-who-secured-Britains-computer-defences-and-paved-the-way-for-safe-online-shopping.html. [Accessed: 03-Dec-2016].

[5] W. Stallings, Cryptography and Network Security: Principle and Practice, 5th ed. Boston: Prentice Hall, 2011, p. 267.

[6] "Bluetooth Low Energy | Bluetooth Technology Website", Bluetooth.com, 2016. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy [Accessed: 03-Dec-2016].

[7] "Bluetooth Smart For Android", Possiblemobile.com, 2016. [Online]. Available: https://possiblemobile.com/2013/12/bluetooth-smart-for-android/ [Accessed: 03-Dec-2016].

[8] IndustryARC Analysis, "Bluetooth Smart/Bluetooth Low Energy Market: Applications (Consumer Electronics, Healthcare, Sports & Fitness, Retail, Automotive, Security); By Technology [Discrete Modules, Integrated Modules (Single & Dual Mode)]-Forecast(2017–2022)", 2017.

[9] F. Lamb, Industrial Automation: Hands On, 1st ed. McGraw-Hill Education, 2013, pp. 74–75.

[10] "APK Decompiler - Decompile Android .apk", Javadecompilers.com. [Online]. Available: http://www.javadecompilers.com/apk. [Accessed: 03-Dec-2016].

[11] "arduino-info - Bluetooth-HC05-HC06-Modules-How-To", Arduino-info.wikispaces.com, 2016. [Online]. Available: https://arduino-info.wikispaces.com/Bluetooth-HC05-HC06-Modules-How-To. [Accessed: 10-May-2017].