 **MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Just Walk Out Library**

|  |  |
| --- | --- |
| **Group 3** | |
| **Group members** | Võ Hồng Hà – Team Leader – SE61394  Nguyễn Tuấn Anh – Team Member – SE61476  Đặng Nhật Thiên – Team Member – SE61357 |
| **Supervisor** | Mr. Kiều Trọng Khánh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | JWL |

- Ho Chi Minh City, 01/2017 -

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: from 02/01/2017…. To /2017…..

(\*) Profession: <Software Engineer> Specialty: <ES> <IS>

x

(\*) Kinds of person make registers: Lecturer Students

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Kiều Trọng Khánh |  | khanhkt@fpt.edu.vn | Mr. |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Võ Hồng Hà | SE61394 | 0949519894 | havhse61394@fpt.edu.vn | Leader |
| Student 2 | Nguyễn Tuấn Anh | SE61476 | 01692536559 | anhntse61476@fpt.edu.vn | Member |
| Student 3 | Đặng Nhật Thiên | SE61357 | 01678785551 | thiendnse61357@fpt.edu.vn | Member |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Just Walk Out Library

Vietnamese: Thư viện Mượn Sách Tự động

Abbreviation:

- JWL

**- Context:**

+ How can you borrow books without having to queue and contact a librarian to check out in libraries?

+ NFC (Near Field Communication) is a form of contactless communication between devices like smartphones or tablets. Contactless communication allows a user to wave the smartphone over an NFC compatible device to send information without needing to touch the devices together or go through multiple steps setting up a connection. utilizes electromagnetic radio fields while technologies such as Bluetooth and Wi-Fi focus on radio transmissions instead (<http://nearfieldcommunication.org/> ). NFC is supported in card or the smartphone

+ QR code (abbreviated from Quick Response Code) is the trademark for a type of [matrix barcode](https://en.wikipedia.org/wiki/Matrix_barcode) (or two-dimensional [barcode](https://en.wikipedia.org/wiki/Barcode)) first designed for the [automotive industry in Japan](https://en.wikipedia.org/wiki/Automotive_industry_in_Japan). A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and [kanji](https://en.wikipedia.org/wiki/Kanji)) to efficiently store data; extensions may also be used

+ RFID stands for Radio-Frequency IDentification. The acronym refers to small electronic devices that consist of a small chip and an antenna. The chip typically is capable of carrying 2,000 bytes of data or less. The RFID device serves the same purpose as a bar code or a magnetic strip on the back of a credit card or ATM card; it provides a unique identifier for that object. And, just as a bar code or magnetic strip must be scanned to get the information, the RFID device must be scanned to retrieve the identifying information (<http://www.technovelgy.com/ct/technology-article.asp> )

**- Building the application provides following services**

* Combine smartphone devices and NFC, QR code, RFID, …
* Building the application that determines the borrowed books automatically when users go out the library
* ...

**- Simulator**

* The borrower uses the application on a smartphone, which supported NFC or QR Code, to emulator.
* The borrower takes any books to his/her bag
* The borrower uses the application to emulator at the exit entrance, the application will show a list of borrowed books
* …

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):

* Student should apply the software development process and the UML
* Software artifacts include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages
* 3 tiers should be applied
* Server side technique:
  + Database design, OOA, OOD, OOP, MVC, Java or .Net technology, Restful API, …
* Client side technique
  + HTML5, CSS, JavaScript, jQuery, Ajax, Android**, iOS...**
* Communication technique
  + Exchange information and transfer data in effective in networks, communicating protocol between mobile devices, ...
* Research
  + Algorithms
  + NFC, QR Code, RFID
  + …

1. Program:

* Main functions
  + The application can allow the user to manage their borrowed books
  + Web Application for users
  + Mobile Application for users
  + Management Web Application for admins
  + ...

1. Other products:

* All of management functions of the system must be implemented to support the operating system in best

4. Other comment (propose all relative thing if have)

N/A

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM city, date 14/12/2016  **On behalf of Registers**  *(Sign and full name)* |

# Table of Contents

[Table of Contents v](#_Toc480114359)

[List of Tables vii](#_Toc480114360)

[A. Introduction 1](#_Toc480114361)

[1. Project Information 1](#_Toc480114362)

[2. Introduction 1](#_Toc480114363)

[3. Current Situation 1](#_Toc480114364)

[3.1 Ho Chi Minh City University of Social Sciences and Humanities library 1](#_Toc480114365)

[3.2 FPT University library 2](#_Toc480114366)

[4. Problem Definition 2](#_Toc480114367)

[5. Proposed Solution 2](#_Toc480114368)

[5.1 Technologies 4](#_Toc480114369)

[5.2 Featured functions 4](#_Toc480114370)

[5.3 Advantages and Disadvantages 5](#_Toc480114371)

[6. Functional Requirements 5](#_Toc480114372)

[7. Roles and Responsibilities 6](#_Toc480114373)

[B. Software Project Management Plan 6](#_Toc480114374)

[1. Problem Definition 6](#_Toc480114375)

[1.1 Name of this Capstone Project 6](#_Toc480114376)

[1.2 Problem Abstract 6](#_Toc480114377)

[1.3 Project Overview 7](#_Toc480114378)

[2. Project organization 12](#_Toc480114379)

[2.1 Software Project Model 12](#_Toc480114380)

[2.2 Roles and Responsibilities 13](#_Toc480114381)

[2.3 Tools and Techniques 14](#_Toc480114382)

[3. Project Management Plan 15](#_Toc480114383)

[3.1 Software development life cycle 15](#_Toc480114384)

[3.2 Phase Detail 15](#_Toc480114385)

[C. Software Requirement Specifications 15](#_Toc480114386)

[1. User Requirement Specification 15](#_Toc480114387)

[1.1 Unauthorized User Requirement 15](#_Toc480114388)

[1.2 Authorized User Requirement 15](#_Toc480114389)

[1.3 Borrower Requirement 15](#_Toc480114390)

[1.4 Librarian Requirement 16](#_Toc480114391)

[1.5 Admin Requirement 16](#_Toc480114392)

[1.6 Automatic Handler Requirement 16](#_Toc480114393)

[1.7 Emulator Requirement 16](#_Toc480114394)

[2 Software Requirement Specification 17](#_Toc480114395)

[2.1 External Interface Requirement 17](#_Toc480114396)

[2.2 System Overview Use Case 18](#_Toc480114397)

[2.3 List of Main Use Cases 19](#_Toc480114398)

[3 Software Requirement Specification 25](#_Toc480114399)

[4 Conceptual Diagram 26](#_Toc480114400)

[D. Software Design Description 28](#_Toc480114401)

[1. System Architectural Design 28](#_Toc480114402)

[1.1 Web Application Architecture Description 28](#_Toc480114403)

[1.2 API Application Architecture Description 30](#_Toc480114404)

[1.3 Mobile Application Architecture Description 31](#_Toc480114405)

[2. Component Diagram 32](#_Toc480114406)

[3. Detailed Description 33](#_Toc480114407)

[3.1 Class Diagram 33](#_Toc480114408)

[3.2 Class Diagram Explanation 34](#_Toc480114409)

[3.3 Interaction Diagram 35](#_Toc480114410)

[4. Database Design 45](#_Toc480114411)

[4.1 Entity Relationship Diagram (ERD) 45](#_Toc480114412)

[4.2 Data Dictionary 45](#_Toc480114413)

[5. Algorithms 46](#_Toc480114414)

[5.1 Check in Algorithm 46](#_Toc480114415)

[5.2 Automatic book - borrowing algorithm 50](#_Toc480114416)

[5.3 Interact with Estimote ibeacon algorithm 55](#_Toc480114417)

[5.4 Save book copy with RFID 56](#_Toc480114418)

[E. System Implement and Test 57](#_Toc480114419)

[1. Database Relationship Diagram 57](#_Toc480114420)

[1.1 Physical Diagram 57](#_Toc480114421)

[1.2 Data Dictionary 57](#_Toc480114422)

[F. Task sheet 58](#_Toc480114423)

[G. Appendix 63](#_Toc480114424)

# List of Tables

[Table 1: Roles and Responsibilities 6](#_Toc480113583)

[Table 2: Hardware requirement for Arduino Emulator 10](#_Toc480113584)

[Table 3: Hardware requirement for RFID reader 10](#_Toc480113585)

[Table 4: Hardware requirement for UART converter 11](#_Toc480113586)

[Table 5 : Hardware requirement for RFID tag/card 11](#_Toc480113587)

[Table 6: iBeacon 12](#_Toc480113588)

[Table 7: Roles and Responsibilities 14](#_Toc480113589)

[Table 8: Tools and Techniques 14](#_Toc480113590)

[Table 9: <Automatic Handler> Check in Borrower 21](#_Toc480113591)

[Table 10: <Automatic Handler> Check out Borrower 23](#_Toc480113592)

[Table 11: <Automatic Handler> Check out Borrow Books 24](#_Toc480113593)

[Table 12: Conceptual Diagram Data Dictionary 27](#_Toc480113594)

[Table 13: Component Dictionary 33](#_Toc480113595)

[Table 14: Class Diagram Dictionary 34](#_Toc480113596)

[Table 15: <Unauthorized User> Login 35](#_Toc480113597)

[Table 16: <Borrower> Check in with identification key by NFC 37](#_Toc480113598)

[Table 17: <Borrower> Check in with identification key by QR Code 38](#_Toc480113599)

[Table 18: <Borrower> Init check out 39](#_Toc480113600)

[Table 19: <Borrower> Init check out 40](#_Toc480113601)

[Table 20: <Borrower> Finish check out 41](#_Toc480113602)

[Table 21: ERD Data Dictionary 46](#_Toc480113603)

[Table 22: Check in device dictionary 48](#_Toc480113604)

[Table 23: Book - borrowing devices dictionary 51](#_Toc480113605)

[Table 24: Physical Diagram 57](#_Toc480113606)

**List of Figures**

[Figure 1: Borrower goes to borrow book 3](#_Toc480113607)

[Figure 2: Borrower borrows book with Librarian 3](#_Toc480113608)

[Figure 3: RFID Module Reader 10](#_Toc480113609)

[Figure 4: UART Converter 11](#_Toc480113610)

[Figure 5: RFID tag/card 11](#_Toc480113611)

[Figure 6: Estimote Beacon 12](#_Toc480113612)

[Figure 7: Scrum Process 13](#_Toc480113613)

[Figure 8: System Overview Use Case 18](#_Toc480113614)

[Figure 9: <Automatic Handler> Check in Borrower 19](#_Toc480113615)

[Figure 10: <Automatic Handler> Check out Borrower 21](#_Toc480113616)

[Figure 11: <Automatic Handler> Check out Borrow Books. 23](#_Toc480113617)

[Figure 12: Conceptual Diagram 26](#_Toc480113618)

[Figure 13: Web Application Architecture 28](#_Toc480113619)

[Figure 14: API Application Architecture 30](#_Toc480113620)

[Figure 15: Mobile Application Architecture 31](#_Toc480113621)

[Figure 16: Component Diagram 32](#_Toc480113622)

[Figure 17: Class Diagram 33](#_Toc480113623)

[Figure 18: Activity Diagram - <Unauthorized User> Login 35](#_Toc480113624)

[Figure 19: Activity Diagram - <Borrower> Request new identification key 36](#_Toc480113625)

[Table 20: Activity Diagram - <Borrower> Request new identification key 36](#_Toc480113626)

[Figure 21: Activity Diagram - <Borrower> Check in with identification key by NFC 37](#_Toc480113627)

[Figure 22: Activity Diagram - <Borrower> Check in with identification key by QR Code 38](#_Toc480113628)

[Figure 23: Activity Diagram - <Borrower> Init check out 39](#_Toc480113629)

[Figure 24: Activity Diagram - <Borrower> Scan book 40](#_Toc480113630)

[Figure 25: Activity Diagram - <Borrower> Finish check out 41](#_Toc480113631)

[Figure 26: <Librarian> Init borrow book copy 42](#_Toc480113632)

[Figure 27: <Librarian> Scan borrow copy 43](#_Toc480113633)

[Figure 28: <Librarian> Submit borrow copy 43](#_Toc480113634)

[Figure 29: <Librarian> Scan return copy 44](#_Toc480113635)

[Figure 30: <Librarian> Submit return copy 44](#_Toc480113636)

[Figure 31: Entity Relationship Diagram (ERD) 45](#_Toc480113637)

[Figure 32: Generate identification key flow chart 47](#_Toc480113638)

[Figure 33: Check in setup 48](#_Toc480113639)

[Figure 34: Check in with identification key flow chart 49](#_Toc480113640)

[Figure 35: Book-borrowing devices setup 51](#_Toc480113641)

[Figure 36: Init checkout flow chart 52](#_Toc480113642)

[Figure 37: Scan book flow chart 53](#_Toc480113643)

[Figure 38: Finish checkout flow chart 54](#_Toc480113644)

[Figure 39: Saving book copy with rfid 56](#_Toc480113645)

**Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| JWL | Just Walk out Library |
| JLib | JWL mobile application name |
| init | initiate |
| API | Application Programming Interface |
| RFID | Radio - Frequency Identification |
| NFC | Near Field Communication |
| QR Code | Quick Response Code |
| HTTP | Hyper Text Transfer Protocol |
| OS | Operating System |
| Admin | Administrator |
| URL | Uniform Resource Locator |
| UML | Unified Modeling Language |
| MVC | Model – View - Controller |
| ID | Identification |

# Introduction

## Project Information

* Project name: **Just Walk Out Library**
* Abbreviation: **JWL**
* Product Type: **Web Application & Mobile App**
* Start Date: **January 5th, 2017**
* End Date: **April 14­th, 2017**

## Introduction

For a long time, library has been a reliable place for people to study and borrow books. However, almost every library system nowadays still uses a traditional way for their borrowers to obtain books: a borrower has to carry all the books that he/she wants to the librarian, so that the librarian can record which books that the borrower will borrow; after that, the borrower signs in each form to commit that he/she is borrowing those books. Only after these steps, the borrower can take back his/her library card and bring the books home. Because one librarian can only manage one borrower at a time, these traditional steps can be very painful if the library is currently crowded with borrowers.

Therefore, we build a system to help libraries and their borrowers to solve those problems. Applying this system, they will not only save a huge amount of time in book-borrowing process, but they will also have a better way to manage books.

## Current Situation

In order to have a more thorough view on our current situation, we will look into the book-borrowing procedure in the libraries of Ho Chi Minh City University of Social Sciences and Humanities, and FPT University.

### Ho Chi Minh City University of Social Sciences and Humanities library

1. The borrower gives the library’s card/student’s card to the librarian.
2. The borrower receives the locker’s key to put the bag in.
3. The borrower walks into the room.
4. The borrower finds wanted books.
5. The borrower takes wanted books.
6. The borrower fills in the book-borrowing form.
7. The borrower brings the books and the form to the librarian’s desk.
8. The librarian checks the form and records the books.
9. The librarian gives the books to the borrower.
10. The borrower re-checks the books, takes back the bag, returns the locker’s key, and brings the books home.
11. The librarian arranges the book-borrowing forms in a specific order.

Reference: http://hcmussh.edu.vn/Resources/Docs/SubDomain/lib/TV.QT10.01%20-%20Quy%20trinh%20phuc%20vu%20muon%20tai%20lieu%20ve%20nha.pdf

### FPT University library

1. The borrower gives the student card to the librarian.
2. The borrower receives the locker key to put the bag in.
3. The borrower walks into the reading room.
4. The borrower finds and takes wanted books.
5. The borrower goes to the librarian to check out.
6. With each book, the student has to fill in a form with the book’s code, and his/her signature.
7. The librarian checks the forms and books.
8. The librarian scans each book’s barcode to record the student’s borrow list. The alarm is turned off for each scanned book.
9. The librarian returns the student’s card to the student.
10. The process finishes. The student can take the books home.

## Problem Definition

Because of current situation, we found that the traditional process has many advantages and disadvantages below:

* Advantages:
* No technical skills needed.
* Disadvantages:
* Borrowing books need to go through complicated steps, thus during the rush hour, it might take too much time to wait for checking out for both borrowers and librarians.
* The librarian should work very hard when having many borrowers.
* Paper records are hard to manage and easy to be damaged.
* Librarians may forget to remind borrowers about their deadline.
* Borrowers have to memorize the deadline of their borrowing books.
* It is hard for borrowers to know when a wanted book is available.
* All activities that relate to borrowing and returning books are manual.

## Proposed Solution

Our proposed solution is to build a system named Just Walk out Library (JWL), which uses NFC, QR Code, RFID technologies to create a more easily and efficiently book - borrowing process: Borrowers only need to use smartphones to check in the library and borrow books, the system will automatically match and save the right books to the right borrower. Therefore, our solution helps to reduce librarian’s work in book – borrowing process, let them have more time for other tasks. Furthermore, librarians and borrowers can manage the borrowing books better.

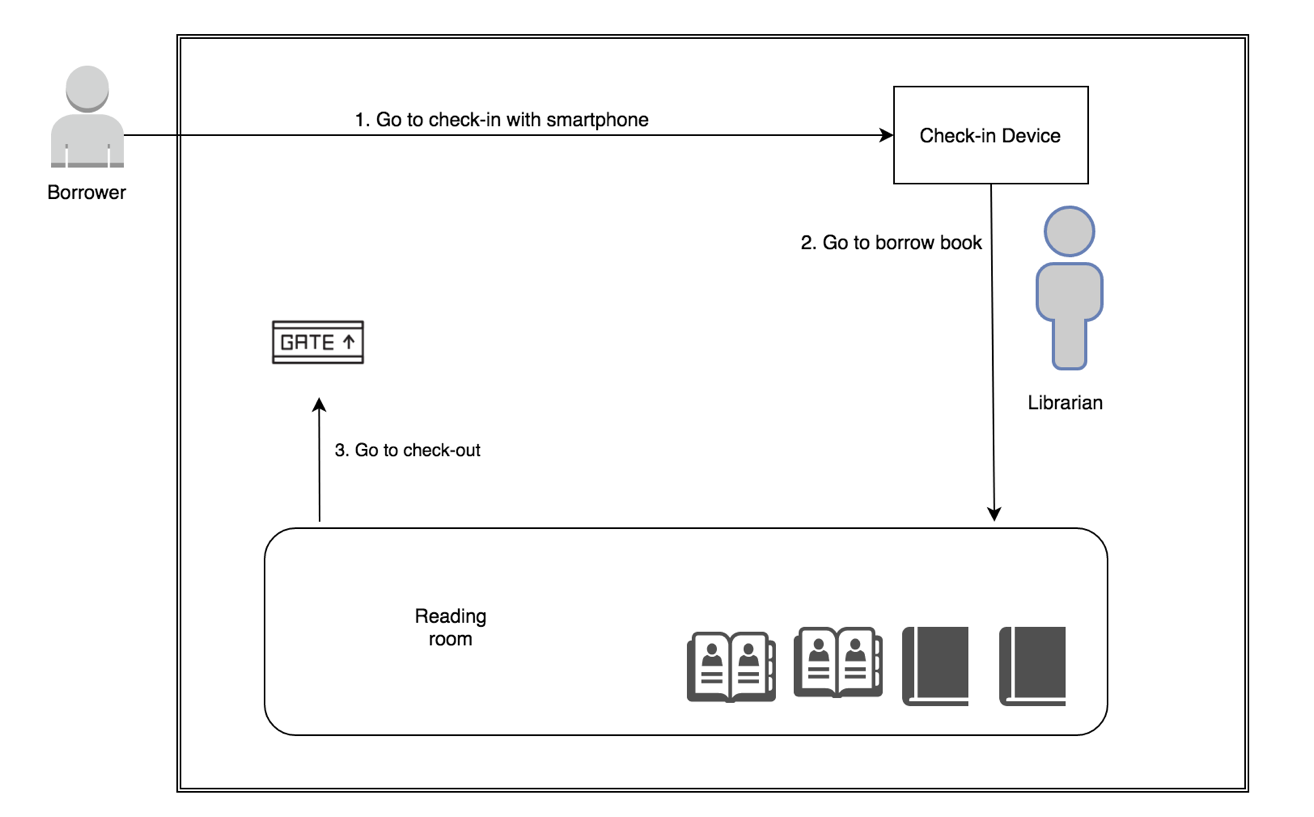


Figure 1: Borrower goes to borrow book

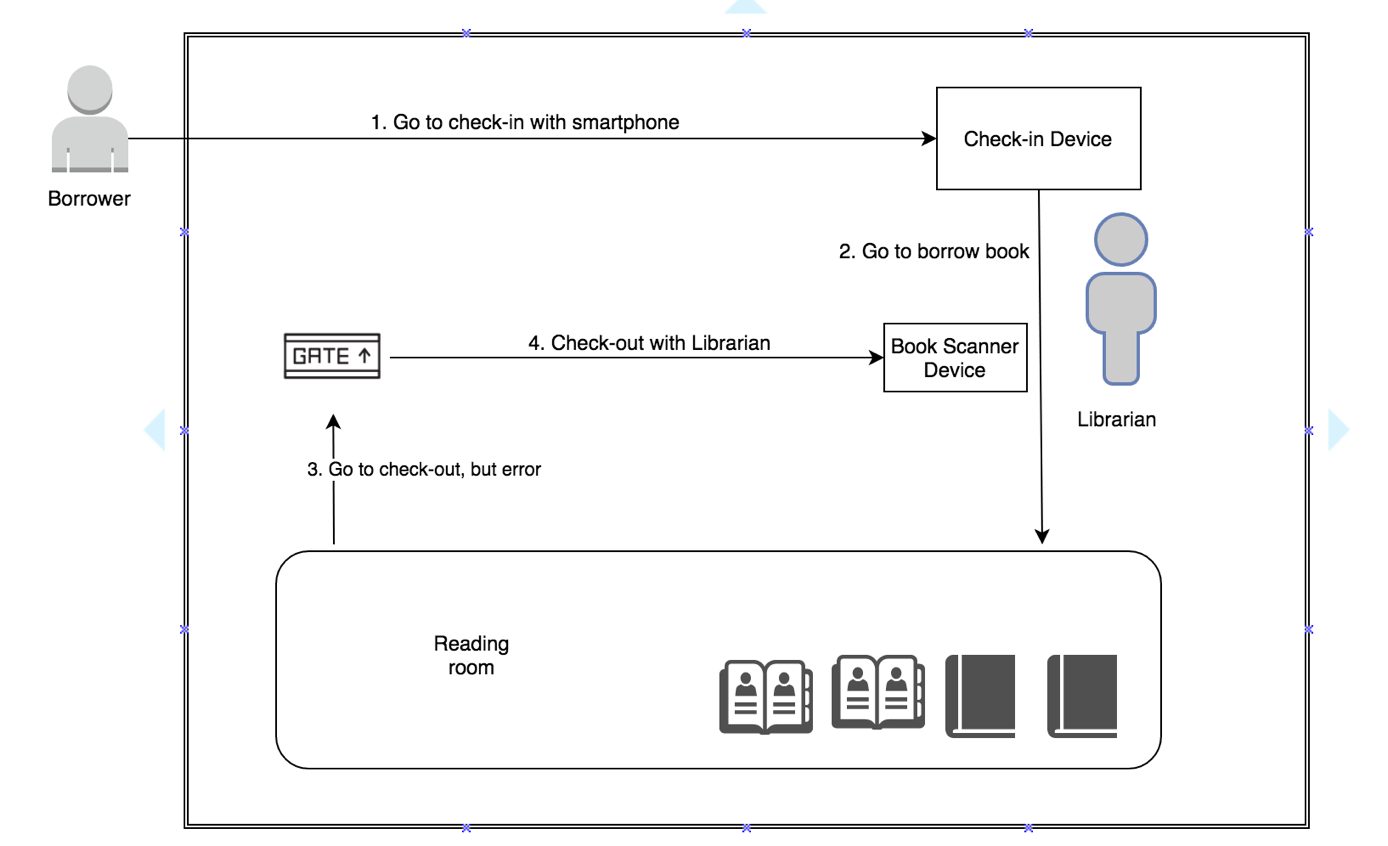


Figure 2: Borrower borrows book with Librarian

* 1. Technologies
  + **NFC (abbreviated from Near Field Communication):** is a form of contactless communication between devices like smartphones or tablets. Contactless communication allows a user to wave the smartphone over an NFC compatible device to send information without needing to touch the devices together or go through multiple steps to set up a connection. NFC utilizes electromagnetic radio fields while technologies such as Bluetooth and Wi-Fi focus on radio transmissions instead. NFC is supported in card or the smartphone (<http://nearfieldcommunication.org/>).
  + **QR code (abbreviated from Quick Response Code):** is the trademark for a type of [matrix barcode](https://en.wikipedia.org/wiki/Matrix_barcode) (or two-dimensional [barcode](https://en.wikipedia.org/wiki/Barcode)) first designed for the [automotive industry in Japan](https://en.wikipedia.org/wiki/Automotive_industry_in_Japan). A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and [kanji](https://en.wikipedia.org/wiki/Kanji)) to efficiently store data; extensions may also be used (<https://en.wikipedia.org/wiki/QR_code>).
  + **RFID (abbreviated from Radio-Frequency Identification):** The acronym refers to small electronic devices that consist of a small chip and an antenna. The chip typically is capable of carrying 2,000 bytes of data or less. The RFID device serves the same purpose as a bar code or a magnetic strip on the back of a credit card or ATM card; it provides a unique identifier for that object. And, just as a bar code or magnetic strip must be scanned to get the information, the RFID device must be scanned to retrieve the identifying information (<http://www.technovelgy.com/ct/technology-article.asp)>.
    - **iBeacon:** They are essentially tiny, low power computers attached to walls or objects in the physical world. Using proximity technologies, they detect human presence and behavior and trigger pre-programmed actions delivering contextual and personalized experiences (<http://estimote.com>).
  1. Featured functions
* **Check in:** JWL application supports both NFC and QR Code. To gain permission to the library, a borrower only needs to open the application on his/her phone, and scan it on our check - in device. He/she will be checked in automatically.
* **Borrow books and check out:** the borrower finds all the books he/she needs on the shelves, and carries them out. The RFID Reader Device and the iBeacons at the gate will automatically recognize that borrower and record the list of books he/she borrows.
* **Ring the alarm:** the alarm will go off if someone:
  + - * + Takes the books out without having JWL’s mobile application turned on and a logged in, valid account, or turn off Bluetooth, internet connection.
        + Exceeds the book limit that they can borrow.
        + Does not have enough money to borrow more books.
        + Checks in, or borrows books with invalid account.
* **Search:** a borrower can search for the books they want by title. The search result will show the book (if it is stored in the library) and its availability (if it is still available to be borrowed).
* **Make wish list:** a borrower can mark a book to his/her wish list so as to receive notification when the book is available again.
  + - **Manage borrowing books:** 
      * A librarian can manage borrowing books: get information of the books, their borrowers, and the borrow status (late or not), and confirm returned books from borrowers. A librarian can also help borrowers to borrow books.
      * A borrower can get information of his/her borrowing books, receive notification when a book is near its deadline, and renew a book’s borrow time.
* **Return book:** Librarian return book of borrower by using Book Scanner Device.
* **Renew book:** Extend the deadline of borrowing book’s borrower.
  1. Advantages and Disadvantages
     + **Advantages:**
       - Make the book-borrowing process faster and easier.
       - Reduce the management time.
       - Store the borrower’s record at ease and safely.
       - Quickly search for wanted books.
       - Notify borrower when a wanted book is available.
     + **Disadvantages:**
       - * Scanning issues: there maybe cases that make the RFID reader to scan books incorrectly.

## Functional Requirements

The functional requirements are listed as below:

* Borrower component:
* Get borrowing/borrowed books.
* Borrow books.
* Get notifications when borrowing books are about to be returned.
* Search books by title.
* Make wish list about an unavailable book, so that borrower can receive notification when the wanted book is back on shelf.
* Renew the borrowing books.
* Librarian component:
* Manage borrowing books: help borrower to borrow or return books.
* Manage borrowers: insert, update, activate, deactivate accounts.
* Tracking books that the borrower are borrowing.
* Search books by title.
* Search borrowers by ID.
* Admin component:
* Admin will have the ability to manage account.
* System component:
* Check in: the system will check the information of borrower before borrower go to inside library.
* Check out: the system will check the information of borrower when borrower go to outside library.
* Scan book: the system will scan the book that has RFID tag/card.
  + Notify: the system will notify borrower about the return date, available book, late deadline, recent list book borrowed.

## Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| **1** | **Kiều Trọng Khánh** | **Project Manager** | **Supervisor** | [**Khanhkt@fpt.edu.vn**](mailto:khanhkt@fpt.edu.vn) |
| **2** | **Võ Hồng Hà** | **Developer** | **Leader** | [**Havhse61394@fpt.edu.vn**](mailto:Havhse61394@fpt.edu.vn) |
| **3** | **Nguyễn Tuấn Anh** | **Developer** | **Member** | [**Anhntse61476@fpt.edu.vn**](mailto:Anhntse61476@fpt.edu.vn) |
| **4** | **Đặng Nhật Thiên** | **Developer** | **Member** | [**Thiendnse61357@fpt.edu.vn**](mailto:Thiendnse61357@fpt.edu.vn) |
| Table 1: Roles and Responsibilities | | | | |

# B. Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

* Official name: Just Walk Out Library
* Vietnamese name: Thư viện mượn sách tự động
* Abbreviation: JWL

### Problem Abstract

* We develop JWL system in order to help librarians reduce work overload and give borrowers a much more comfortable experience in acquiring books. The role of this system is to automate the book-borrowing process with the help of RFID, NFC, QR Code, and iBeacon. However, those great technologies could come with below difficulty:
* Those are complicated software and hardware which our team does not have any background knowledge. During the process of doing this research, we spend a good amount of time to study not only the operational principles but also detailed features and requirements. This is our main challenge.

### Project Overview

#### Current Situation

Below are the problems encountered in this project:

* **New technologies:** Our team has no experience in embedded system. Therefore, we need time to do research about each technology, which devices to choose, and how to use them.
* **Traditional habit:** Borrowers have been using traditional way to obtain books for a long time, so libraries should provide proper introduction and assistance for their users to get used to JWL.
* **Devices problem:** RFID/NFC devices may be crashed or damaged after long time using, so backup solution should be prepared and periodically maintenance should be carried out.

#### The Proposed System

* After doing many researches on technology for saving information, we choose RFID, NFC and QR code technologies as these technologies is very capable of resolving the current situations in borrowing book. The basic idea is to use a RFID tag that each tag contains a unique book ID.
* Our system includes four subsystems:
* Mobile application: for borrower, check in emulator.
* Web application: for admin, librarian.
  + API application: to serve API for web application, mobile application and communicate with Emulator application.
* Emulator application: for check in and scan books.

##### Web Application

* For admin:
* Manage accounts
* For librarians:
* Manage borrowers.
* Deactivate/activate borrowers.

##### Mobile Application

For borrower:

* Check in user through NFC (or QR Code).
* Get borrower’s information.
* Search books.
* Add books to wish list.
* View list of borrowing books.
* View history about borrowed books.
* View library’s policy.
* Renew books.
* Notify borrow about:
  + Recently borrowed books.
  + Book’s deadline.
  + Available books in the borrower’s wish list.

##### Emulator Application

For RFID Reader Gate:

* Reader RFID from scanned books.
* Send data from scanned books to the JWL server.
* Raise alarm when errors occur: robbery, borrowers do not meet requirement to borrow books, or system error.
* Stop alarm manually.

For RFID Reader for Librarian:

* Read RFID from scanned books.
* Send data from scanned books to the JWL server.
* Provide GUI control for librarian to help borrower borrow and return books.

For Check in Device:

* Read NFC/QR Code from mobile application.
* Send read data to JWL server.
* Play sound and show text message result of check in to librarian.

##### API Application

Main functions of backend site include

* Serve formatted data for both web and mobile.
* Communicate with Emulators.
* Communicate with third party service as: Firebase, upload.im.

#### Boundaries of the System

The system should do:

* + Allow borrowers to check in the library by scanning their phone to the Check-in Device of the library.
  + Allow borrowers to check out with their borrowing books by just walking out of the library.
  + Allow borrowers to search wanted books by title.
  + Allow borrowers to add a book to a wish list.
  + Allow librarians to manage borrowers and their borrowing book list.
  + Allow librarians to help borrowers borrow and return books.
  + Allow admins to manage all accounts.
  + Notify borrowers when their borrowing book near to deadline.
  + Notify borrowers when their borrowing book late of deadline.
  + Notify borrowers when their recent borrow books.
  + Allow system to automatic check out borrower with configured time.

The system should not do:

Our system is not a library management system, therefore, we do not handle any library-related tasks:

* + Do not manage human resource.
  + Do not manage stored books.

#### Future plans

With further research and development, the system can apply the following features:

* The system provides more thorough library management.
* The system provides more ways to search books: by category, book type, position, author, and publisher.
* The system separates RFID robbery scanner gate and RFID book scanner gate to provide a more comfortable experience for borrowers.
* The system provides better description when a borrower checks in so that librarian can identify a borrower better.
* Librarian can view the history:
  + When the borrower deposit.
  + When the borrower was penalty.
  + When the borrower renews books.
  + When the borrower returns books.
  + When the borrower borrows books.

#### Development Environment

##### Hardware requirement

* (References to main document, Section B – 1.3.5.1 for server, web application and mobile application)
* **For RFID Emulator:**

RFID Emulator consists of 2 part: RFID Reader and Arduino Emulator.

* + Arduino Emulator: we use laptop computer as Arduino Emulator.
  + RFID Reader: RFID module and UART converter.

|  |  |
| --- | --- |
| Arduino Emulator | Requirement |
| Communication | Wi-fi, USB, cable |
| Development Language | Python |

Table 2: Hardware requirement for Arduino Emulator

* **Module RFID**

****

Figure 3: RFID Module Reader

|  |  |
| --- | --- |
| RFID Module | Information |
| Module | RDM6300 |
| Frequency | 125 kHz |
| Baud Rate | 9600 |
| Interface Type | TTL Level RS232 format |
| Operating Voltage | DC 5V |
| Operating Current | >50mA |
| Reception Range | 20~50mm (depending on the antenna, the card and the surroundings) |

Table 3: Hardware requirement for RFID reader

*Reference:* [*http://www.dientubachviet.vn/498-rfid-rdm6300-125khz-uart-output.html*](http://www.dientubachviet.vn/498-rfid-rdm6300-125khz-uart-output.html)

* **UART converter**

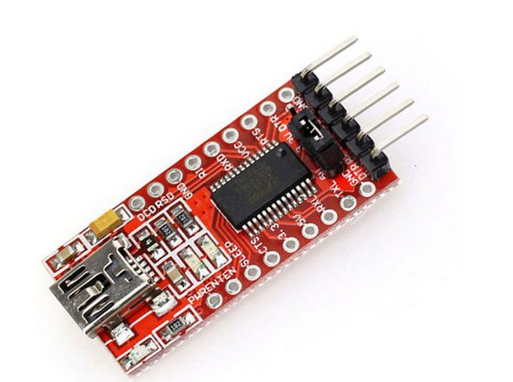
****

Figure 4: UART Converter

|  |  |
| --- | --- |
| RFID Converter | Information |
| Main IC | FT232RL from FTDI |
| Baud Rate | Custom |
| Description | Convert from UART TTL to USB |
| Operating Voltage | 5VDC from USB |

Table 4: Hardware requirement for UART converter

*Reference:* [*http://hshop.vn/products/mach-chuyen-usb-uart-ft232rl*](http://hshop.vn/products/mach-chuyen-usb-uart-ft232rl)

* **For RFID tag/card:**

****

Figure 5: RFID tag/card

|  |  |
| --- | --- |
| RFID Tag/Card | Information |
| Protocol | EM4100 |
| Operating Frequency | 125 kHz |
| Read Distance | Maximum to 10cm |
| Unique ID | 32 bit |
| Weigth | 8g |

Table 5 : Hardware requirement for RFID tag/card

*Reference:* [*http://hshop.vn/products/the-trang-rfid-125khz*](http://hshop.vn/products/the-trang-rfid-125khz)

* **For Estimote Beacon:**

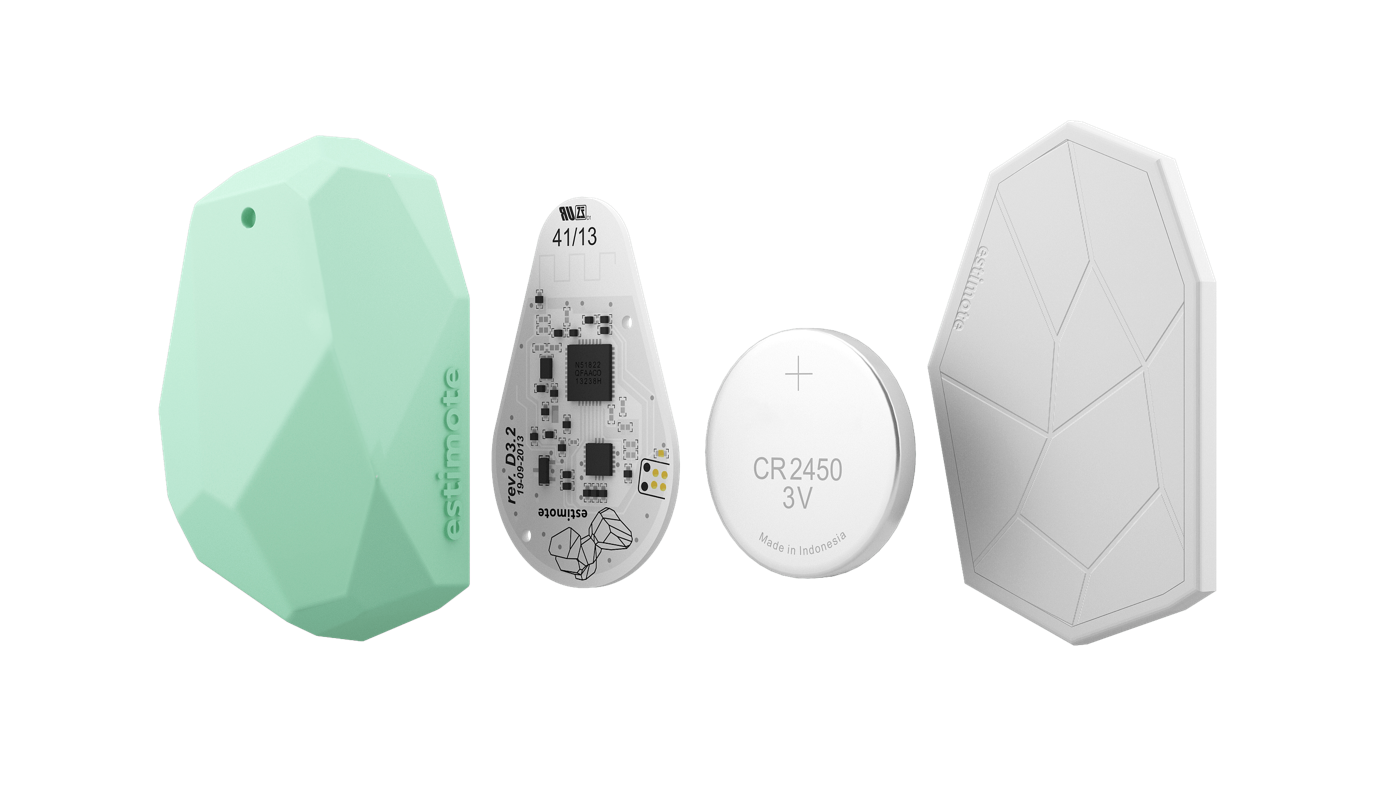
****

Figure 6: Estimote Beacon

Introduction: “iBeacon™ is a communication protocol developed by Apple on top of Bluetooth Smart technology. It allows developers to create mobile apps aware of location context provided by beacons. Estimote Beacons are compatible with iBeacon.”

|  |  |
| --- | --- |
| iBeacon | Information |
| Name | Estimote Proximity Beacon |
| Protocol | Bluetooth Low Energy |
| Range | Maximum to 70m |
| Lifetime | 2 years |
| Weight | 31g |

Table 6: iBeacon

*Reference:* [*http://estimote.com/*](http://estimote.com/)

##### Software requirement

(*References to main document, Section B - 1.3.5.2)*

## Project organization

### Software Project Model

This project is developed using Scrum Model. This model is appropriate under consideration of 3 following reasons:

* Those technologies used in this project (RFID, NFC, iBeacon, and QR Code) require much attention in studying. This takes time to concentrate on technologies understanding, we plan to apply them gradually to the system.
* Prototypes are delivered frequently for evaluation.
* Team members can involve more in the development process.



Figure 7: Scrum Process

(*References*:[*http://skytechnovation.com/scrum-development-model/*](http://skytechnovation.com/scrum-development-model/))

### Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Kiều Trọng Khánh | Product owner, Project Manager | * Specify user requirements * Control the development process   Provide technical and business analysis support |
| 2 | Võ Hồng Hà | Scrum leader, B.A, Developer, Tester | * Manage process * Prepare and combine documents * Design database * Clarify requirements * GUI design * Create test plan * Code * Test |
| 3 | Nguyễn Tuấn Anh | Team member,  B.A, Developer,  Tester | * Design database * Clarify requirements * Prepare documents * GUI design * Create test plan * Code * Test |
| 4 | Đặng Nhật Thiên | Team member,  B.A, Developer,  Tester | * Design database * Clarify requirements * Prepare documents * GUI design * Create test plan * Code * Test |

Table 7: Roles and Responsibilities

### Tools and Techniques

|  |  |  |
| --- | --- | --- |
|  | Tool | Techniques |
| Front-end | Webstorm | * HTML5 * CSS3 * ES6 * NodeJS * ReactJS |
| Back-end | IntelliJ | * RESTFul * JSON * Spring * Java * PostgresSQL |
| Web server | Heroku | N/A |
| Mobile application | Android Studio | * Java 8 * Android SDK |
| Arduino Emulator | Sublime | Python |
| Database management system | PgAdmin | N/A |

Table 8: Tools and Techniques

## Project Management Plan

### Software development life cycle

*(References to main document, Section B – 3.1)*

### Phase Detail

*(References to main document, Section B – 3.2)*

# Software Requirement Specifications

## User Requirement Specification

### Unauthorized User Requirement

Unauthorized User is the user who has not logged into this system. Unauthorized User only has one functions.

* Login.

### Authorized User Requirement

Authorized User is the user who has logged into system. Authorized User has two common functions.

* Get profile detail.
* Edit profile.
* Logout.

### Borrower Requirement

Borrower is the user who goes to the library to search for books. They can either read books at the library or borrow them home. Therefore, their main functions in the system are:

* + Check in.
  + Check out.
  + Borrow books.
* Add books to wish list.
* Search books.
* Get borrowing list.
* Get history of borrowed list.
* Get library policy.
* Renew book.

### Librarian Requirement

* Librarian is the user who manages the library. They have the following functions:
* Manage borrowers: (Optional usecase to simulate the library)
  + - * Add borrowers.
      * Update borrowers.
      * Activate/Deactivate borrowers.
* Manage books: (Optional usecase to simulate the library)
  + - * Add new books.
      * Update books.
      * Delete books.
* Check out borrowers.
* Get borrowing book list.
* Search books.
* Return books.

### Admin Requirement

Admin is the user who is responsible for managing accounts for the whole system, has the following function:

* Get accounts.
* Manage accounts: (Optional usecase to simulate the library)
  + - * Create account.
      * Update account.
      * Delete account.

### Automatic Handler Requirement

Automatic Handler is an intermediate system service which facilitates the flow of the whole system by automating the following functions:

* + Check in borrower by QR Code.
  + Check in borrower by NFC.
  + Record borrowers and their borrowing books.
  + Check out borrowers.
  + Push notification to borrowers.

### Emulator Requirement

#### RFID Reader Emulator

RFID Reader Emulator is the device which can scan RFID tag/card on each book, has the following function:

* Read books’ rfids.
* Call API and send data to server.

#### Check in Emulator

Check in Emulator is the device which can read NFC/QR Code on the mobile application, has the following function:

* Read NFC/QR Code.
* Call check in API and send data to server.

## Software Requirement Specification

### External Interface Requirement

#### User Interface

*(References to main document, Section C – 2.1.1)*

#### Hardware Interface

* Smartphone with:
  + - * BLE support (Bluetooth 4.0).
      * NFC support.
* RFID Reader and tags use 125 kHz.
* Estimate Beacon.

#### Software Interface

* + Mobile application : Android OS version 4.4 (or 5.0).
  + Estimate cloud for manage ibeacons.
  + Estimate SDK for connect between mobile application and ibeacons.

#### Communication Protocol

* + Use HTTP 1.1 to communicate between the web application and the API application.
  + Use HTTP 1.1 to communicate between the mobile application and the API application.
  + Use BLE protocol for communication between the mobile application and the Estimate ibeacons.
  + Use NFC protocol for communication between the mobile application and the emulator for check in.
  + Use RFID protocol with low frequency (125 kHz) for communication between RFID tags/cards and RFID reader.

### System Overview Use Case

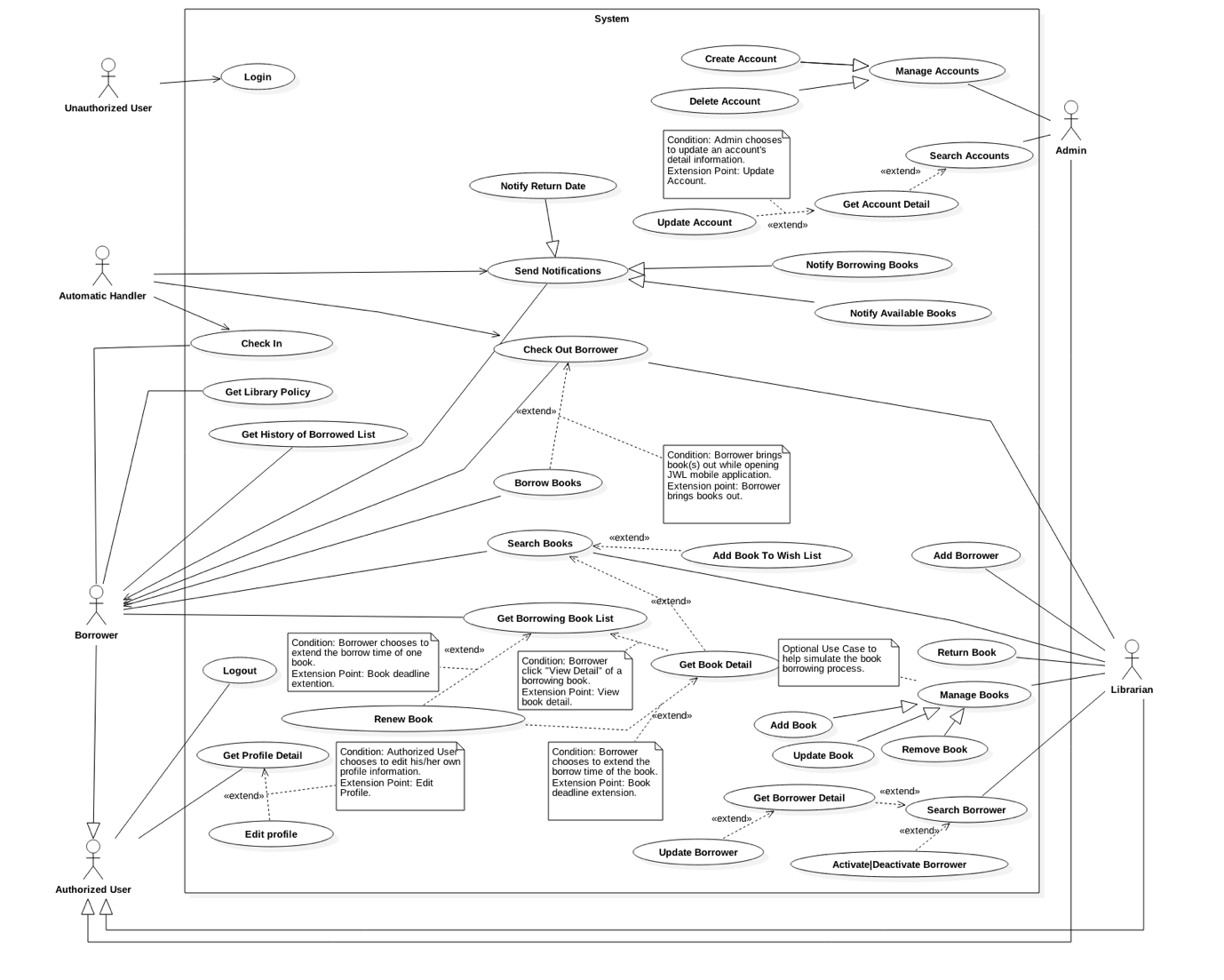


Figure 8: System Overview Use Case

### List of Main Use Cases

#### <Automatic Handler> Check in Borrower

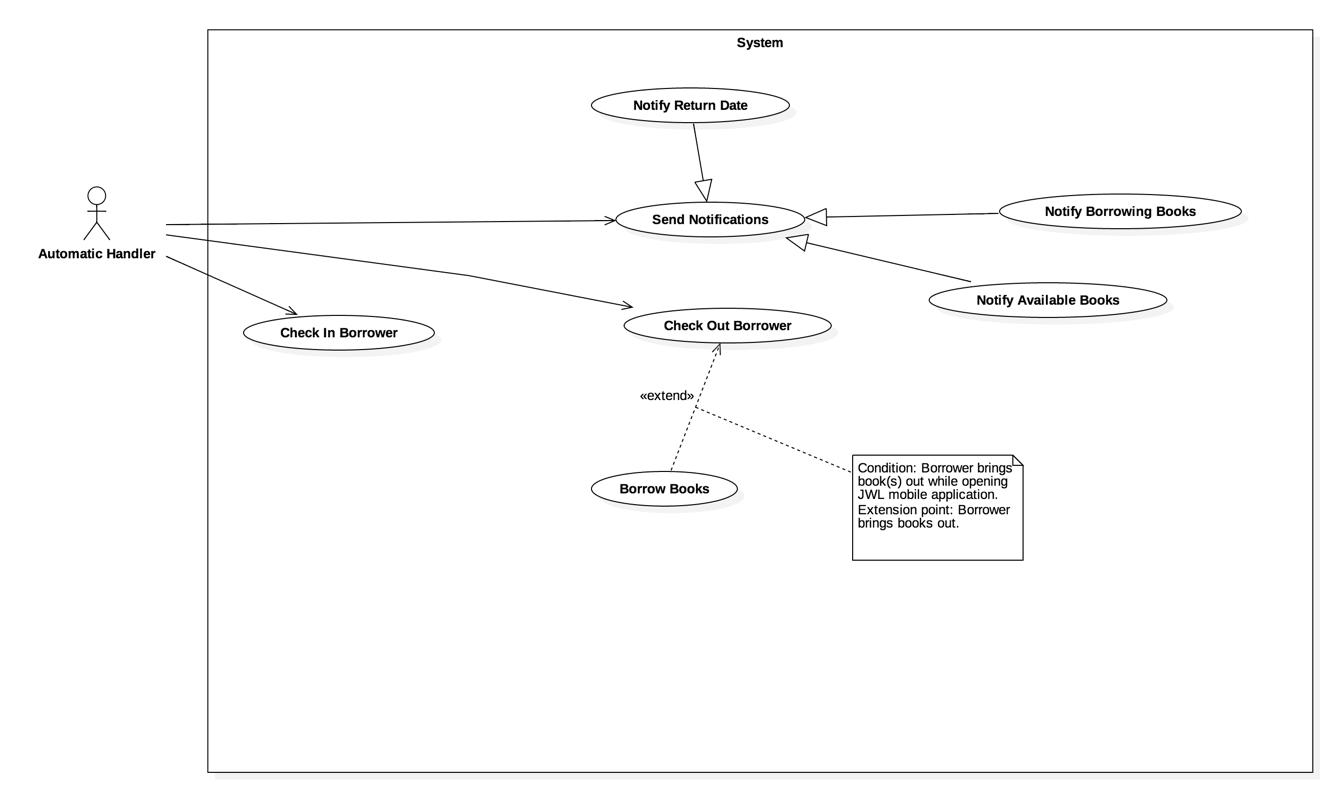


Figure 9: <Automatic Handler> Check in Borrower

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – JWL01** | | | |
| **Use Case No.** | JWL01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check in Borrower | | |
| **Author** | Vo Hong Ha | | |
| **Date** | February 13, 2017 | **Priority** | High |
| **Actor:**   * Automatic Handler   **Summary:**   * This use case allows Automatic Handler to verify a borrower’s identity.   **Goal:**   * Return message showing the borrower’s identity verification result.   **Triggers:**   * Automatic Handler checks the information of borrowers when they scan smartphone at the check in device.   **Preconditions:**   * NFC: borrower turns on NFC on mobile device. * QR Code: borrower keeps monitor bright on while showing the application. * Borrower keeps mobile phone approximately 10 – 15 cm far away from reader with QR Code check in or touch onto the reader with NFC check in.   **Post Conditions:**   * **Success:**   + The borrower’s status “in library” is updated.  + The message will show on the application.   * **Fail:** N/A   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1  2 | Automatic Handler gets the information from Check in Emulator.  Automatic Handler sends “Check in” command.  [ Exception 1] | System check the information of borrower. Then response the message of result:  “Chào mừng bạn tới thư viện.”  [Alternative 1, 2] |   **Alternative Scenario:**  Alternative 1:   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | The borrower is deactivated. | System response the message of result  “Tài khoản của bạn hiện đang bị khoá, vui lòng liên hệ thủ thư.” |   Alternative 2:   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | The information of borrower in mobile device is out of date. | System response the message of result  “Check-in không thành công, vui lòng làm mới key và thử lại.” |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Automatic Handler receives the response of the system, then plays sound and displays notification in comparable with that response on the Check in Emulator. * NFC and QR Code on mobile application: contains user ID and the identification key of borrower. | | | |

Table 9: <Automatic Handler> Check in Borrower

#### <Automatic Handler> Check out Borrower

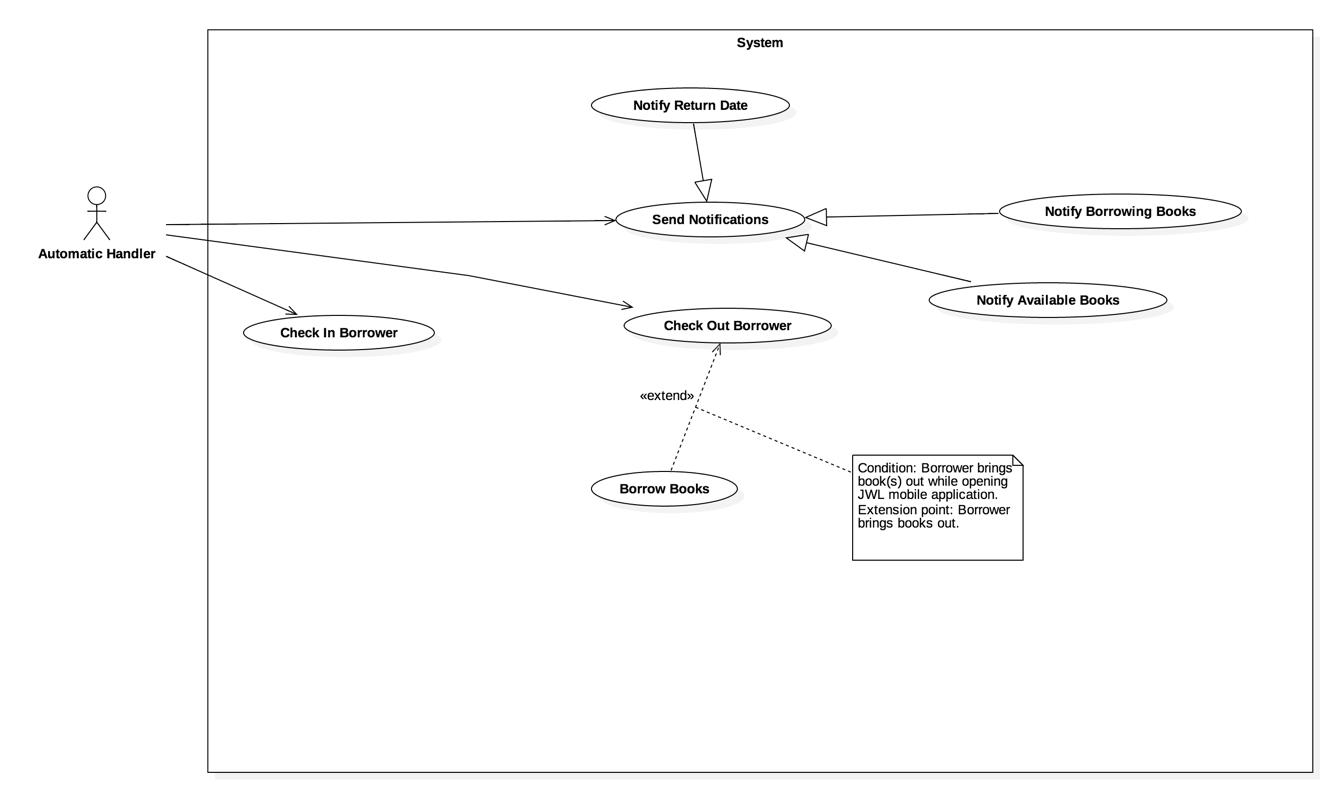


Figure 10: <Automatic Handler> Check out Borrower

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – JWL02** | | | |
| **Use Case No.** | JWL02 | **Use Case Version** | 1.0 |
| **Use Case Name** | Check out Borrower | | |
| **Author** | Vo Hong Ha | | |
| **Date** | February 13, 2017 | **Priority** | High |
| **Actor:**   * Automatic Handler   **Summary:**   * This use case allows Automatic Handler to check out the borrower.   **Goal:**   * Check out the borrowers who go out the library.   **Triggers:**   * Borrower goes into the range of init check out iBeacon.   **Preconditions:**   * Borrower not turn off the mobile application and the turn on Bluetooth, internet connection. * Borrower keeps go out the library.   **Post Conditions:**   * **Success:** Borrower receive the success message. * **Fail:** N/A   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1  2  3  4 | Borrower walks into the range of init  check out iBeacon.  Automatic Handler sends “Init  Check out” command.  Borrower walks into the range of  finish check out iBeacon.  Automatic Handler sends “Finish Check out” command. | System response message to borrower:  “Bạn có thể mượn sách”.  System response message to borrower:  “Cám ơn bạn đã sử dụng thư viện”. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Extend Borrow Books: Borrower can borrow book while check out.  **Business Rules:**   * After “Finish Check out” command successful, the status of borrower “in library” must be change to “false”. * When “Init Check out” command: * The system only allows one borrower to check out at a time. * The system creates new borrow cart and calculates usable balance. | | | |

Table 10: <Automatic Handler> Check out Borrower

#### <Automatic Handler> Check out Borrow Books

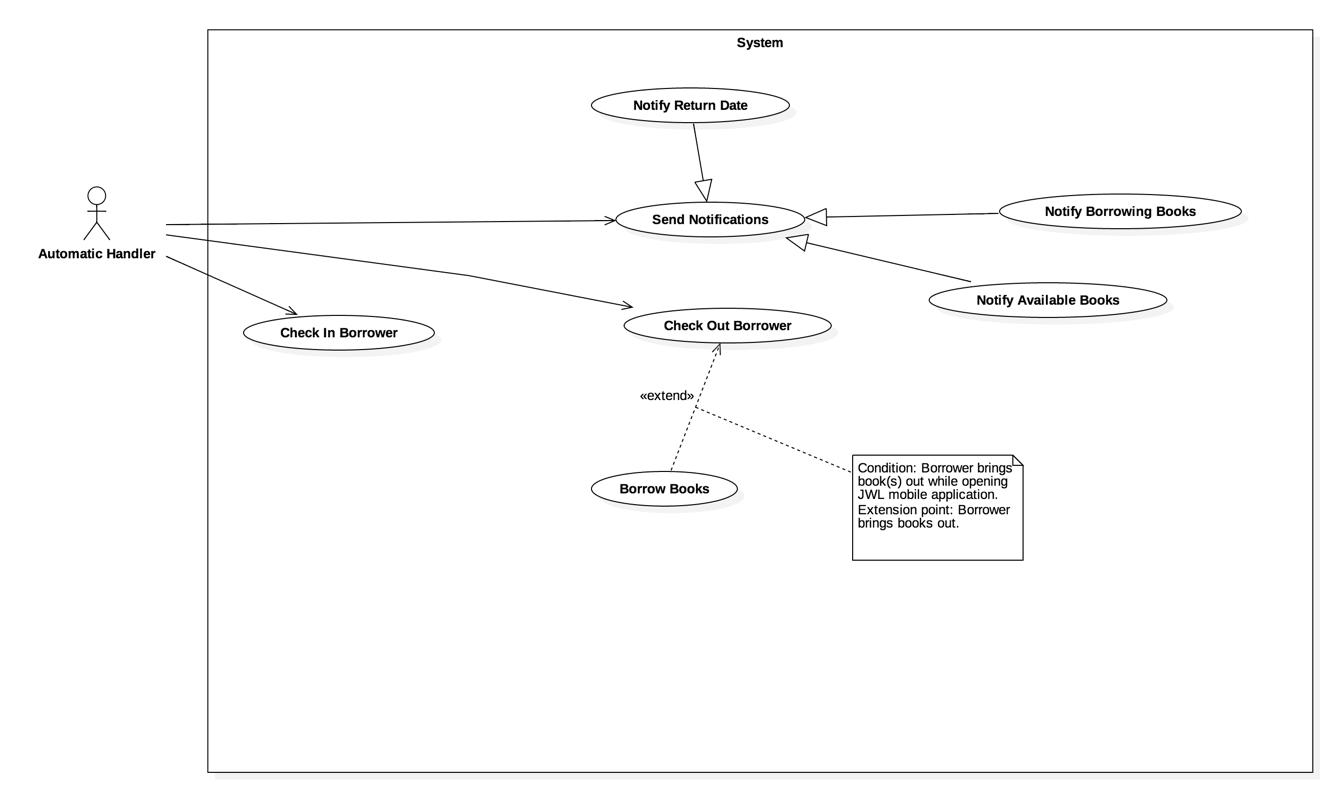


Figure 11: <Automatic Handler> Check out Borrow Books.

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – JWL03** | | | |
| **Use Case No.** | JWL03 | **Use Case Version** | 1.0 |
| **Use Case Name** | Borrow Books | | |
| **Author** | Vo Hong Ha | | |
| **Date** | February 13, 2017 | **Priority** | High |
| **Actor:**   * Automatic Handler   **Summary:**   * This use case allows Automatic Handler to scan and save borrower’s books while the borrower is walking out library.   **Goal:**   * Automatically borrow books for borrower.   **Triggers:**   * The Borrower goes through RFID Reader Gate.   **Preconditions:**   * RFID Reader Gate is installed and working correctly. * Each book has RFID tag.   **Post Conditions:**   * **Success:**   + System records all books that borrower borrows.  + Borrower receives notification from system.   * **Fail:** N/A   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1  2  3  4 | Borrower goes through RFID  Reader Gate.  Automatic Handler scans books.  Automatic Handler sends “Add New  Scanned Books” command. | System plays successful sounds.  [Exception 1] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Borrower does not have enough money to borrow more books. | System raises alarm. | | 2 | Borrower exceeds the number of books quota. | System raises alarm. | | 3 | Borrower fails to initiate checkout. | System raises alarm. | | 4 | Borrower has not checked in. | System raises alarm. |   **Relationships:** Extend from Check Out.  **Business Rules:**   * Borrower has initiated check out. * Borrower has enough money to borrow the books he/she is carrying. * Borrower has not exceeded number of books quota. | | | |

Table 11: <Automatic Handler> Check out Borrow Books

## Software Requirement Specification

* 1. **Usability**
* The Android application for emulator requires 10 minutes training for librarian.
* The Android application for borrower will take 15-20 minutes to get used to the mobile system completely.
* Admin and Librarian interface are written in English include all dialogs and messages.
* Borrower interface is written in Vietnamese includes all dialogs and message.
* Emulator interface is written in English
  1. **Reliability**
* The number of sending notification failure is 1 time per 1000 notifications.
* The scheduler tasks run at configured time with 100% execution rate.
* RFID Reader has 100% accuracy.
  1. **Availability**
* System is available 24 hours per day and 7 days per week.
* System should take at most 5 hours per month for backup or repairing.
  1. **Security**
* Privacy: Each role of user has a specific permission to interact with system.
* Only admin can grant permission to other roles.
* System always check the status of borrower before doing anything.
  1. **Maintainability**
* The system is divided into separated modules.
* The code is easy to maintain and upgrade.
  1. **Portability**
* The web application can be used in any platform that use Chrome (v25 or above) and can connect to the Internet.
* Mobile application for borrower and emulator runs on Android API greater than 4.4.
  1. **Performance**
* Web appication handles the task within 5 seconds.
* Request from mobile application are responded in less than 10 second at 4Mbps.

## Conceptual Diagram

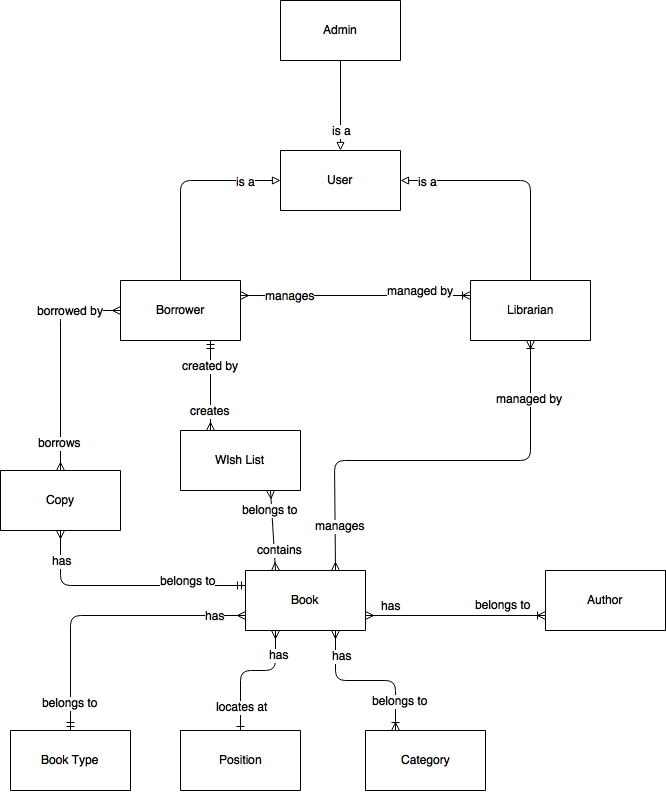


Figure 12: Conceptual Diagram

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| User | * Contains all information of users in the library. |
| Admin | * Person who manages all accounts. |
| Librarian | * Person who manages all borrowers and books. |
| Book | * Contains all information of books in the library. |
| Copy | * Contains all information of copies of books. |
| Author | * Contains all information of authors of books in the library. |
| Category | * Contains all information of book’s categories, for example, Math, Literature, Physics, Software Engineering, … |
| Book Type | * Contains all information of book’s types, for example, text book or reference book. |
| Position | * Contains all information of physical positions of books in the library. |
| Wish List | * Contains all information of books wished by borrowers. |

Table 12: Conceptual Diagram Data Dictionary

# Software Design Description

## System Architectural Design

### Web Application Architecture Description

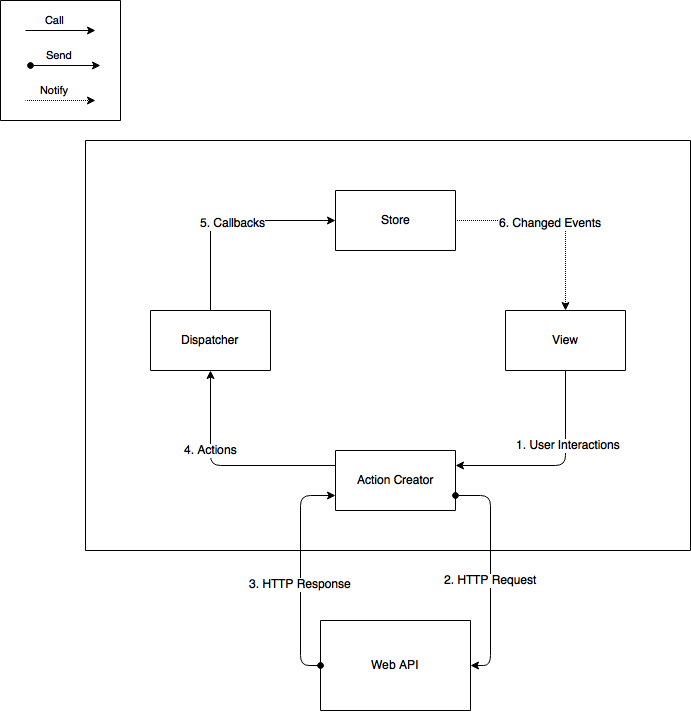


Figure 13: Web Application Architecture

The web application uses Flux (<https://facebook.github.io/flux/>) as an architecture. We choose this architecture for a couple of reasons:

* Performance: The web application’s render performance is dramatically improved thanks to React. React renders the web application with the help of its virtual DOM. Each React view component has a render method, which will update the virtual DOM. React then compares the updated virtual DOM with a virtual DOM snapshot that was taken right before the update. By doing so, React figures out exactly which virtual DOM objects have change, and only updates those different objects on the real DOM.
* Simplicity: the whole web application is simple to develop and maintain, because the data always flows in one direction. There is no need to handle event listener for each web component like in MVC application. This architecture also separates frontend from backend, so developers are able to work independently.
* Reusability and Testability: The architecture divides the application into components. This increases reusability and testability.

### API Application Architecture Description

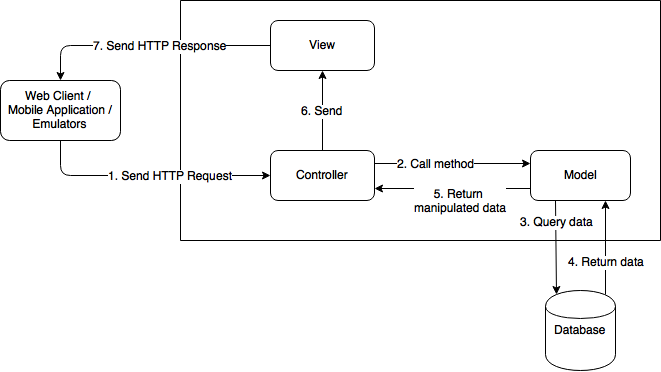


Figure 14: API Application Architecture

The API Application Architecture uses Model – View - Controller (MVC) architecture. We choose MVC for the following advantages:

* The application will be divided into three clear parts: Model – View – Controller.
* Developing: it is easier for developers to divide work and develop the system.
* Testing, Maintaining, and Extending: thanks to dividing the system into clear components, we decrease the dependence and complexity of the developing process. Therefore, testing, maintaining and extending work become much easier.

### Mobile Application Architecture Description

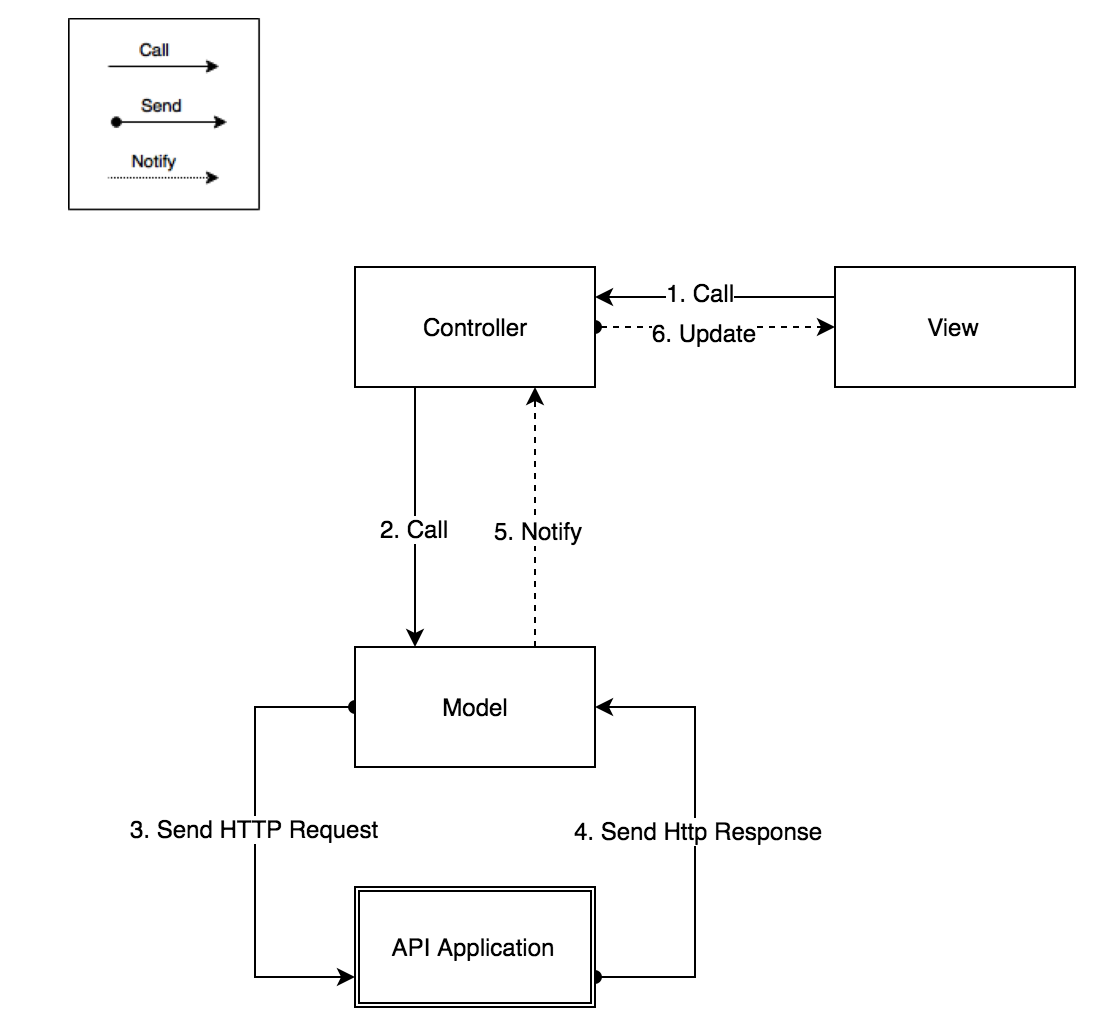


Figure 15: Mobile Application Architecture

Mobile application architecture will use MVC. Therefore, we can organize the code better for maintainability, extensibility, and reusability

## Component Diagram

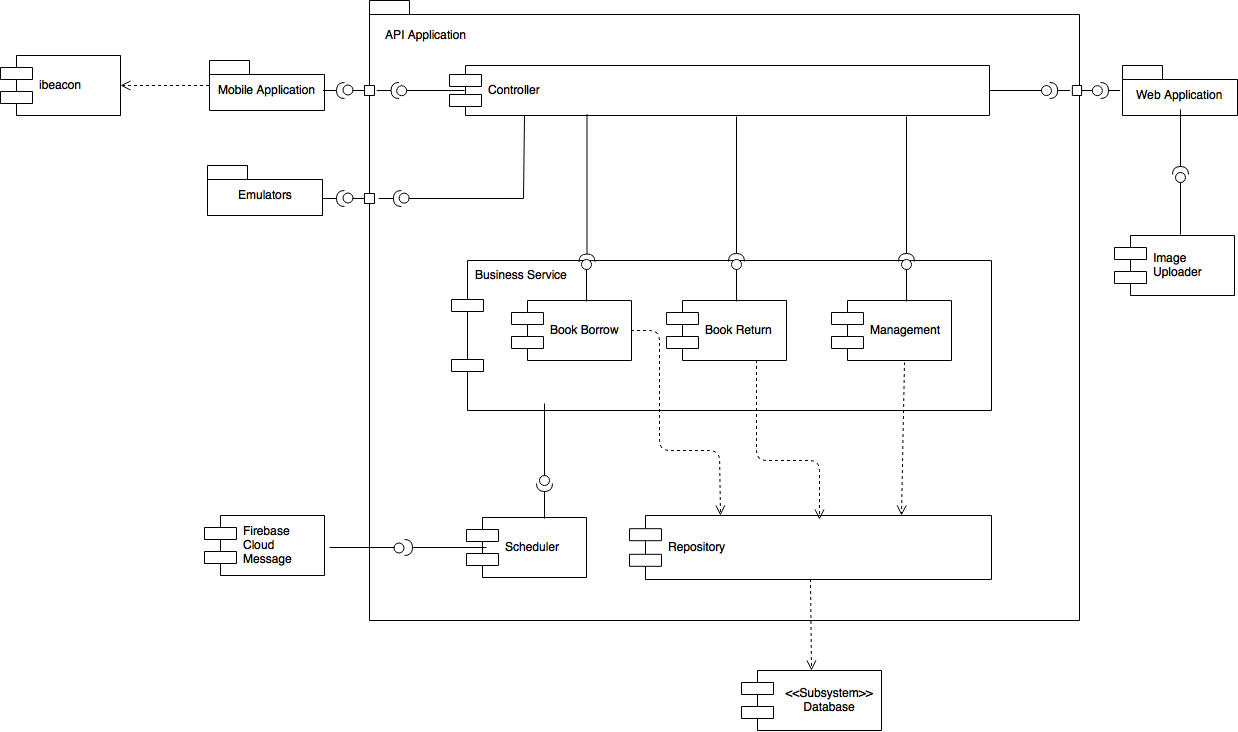


Figure 16: Component Diagram

|  |  |
| --- | --- |
| **Name** | **Description** |
| API Application | Provide API for Mobile application, Web application, and Emulators |
| Controller | Handles HTTP requests and calls services |
| Business Service | Handles system’s business operations |
| Management | Manages system data |
| Book Borrow | Handles borrowing books process, including: initiate borrow process, scan books, check out |
| Book Return | Handles return books process by librarian |
| Scheduler | Runs background job and call Notification |
| Repository | Abstract data layer to interact with database system |
| Database System | Stores data of the system |
| Emulators | Handles books scanning and borrower’s identification scanning. Includes Check in device, RFID Reader Gate, and Book scanner for librarian |
| Mobile Application | Provides graphical user interface for borrower |
| Web Application | Provides graphical user interface for librarian and admin |
| Firebase Cloud Message | External Component to send messages to Mobile application |
| Image Uploader | External Component to upload image and create image url |
| iBeacon | Triggers event on Mobile Application to call checkout APIs |
| Table 13: Component Dictionary | |

## Detailed Description

### Class Diagram

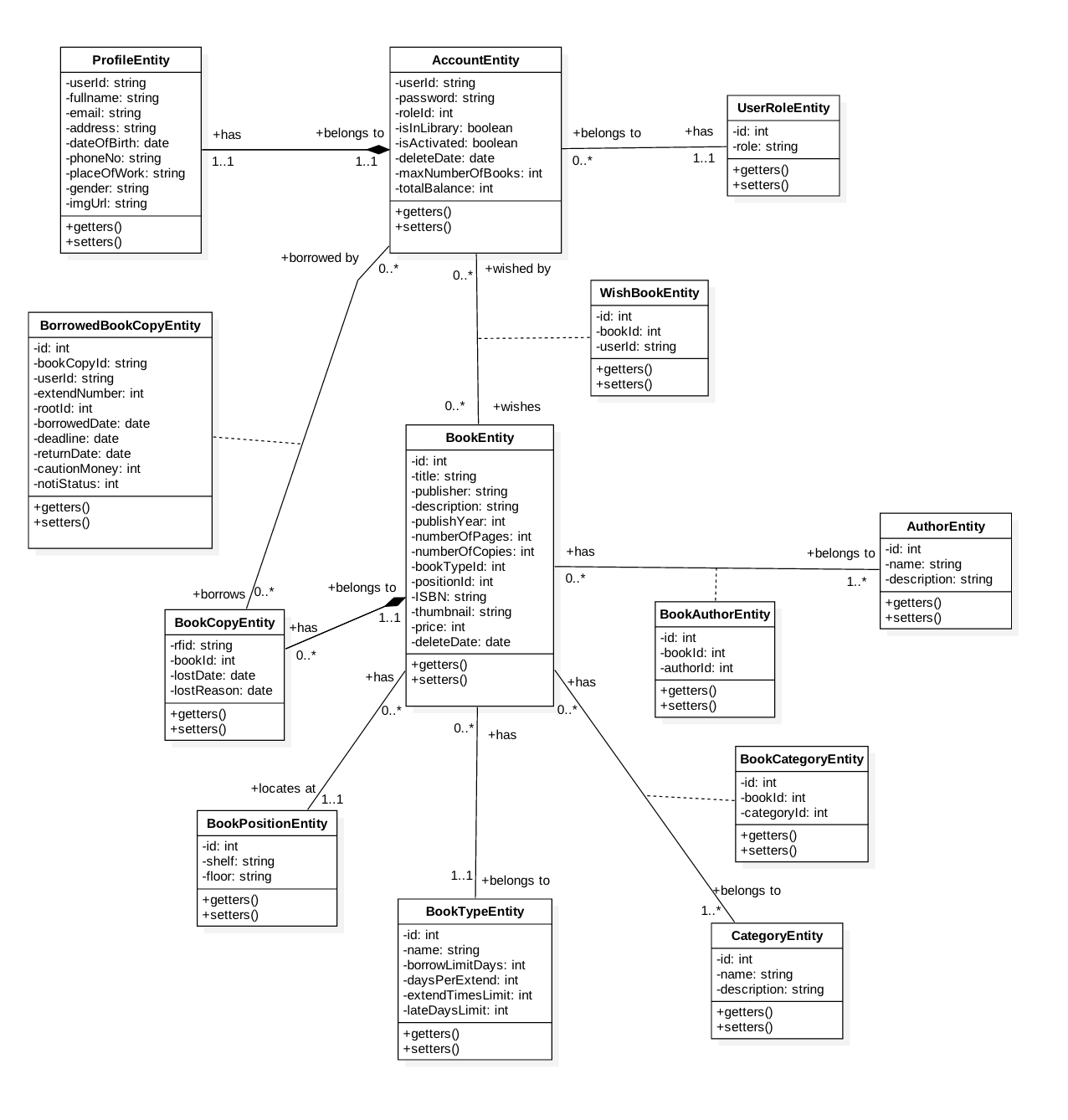


Figure 17: Class Diagram

|  |  |  |
| --- | --- | --- |
| **Class name** | **Mapping column with conceptual diagram** | **Description** |
| ProfileEntity | N/A | Contains the detail information of accounts of the system’s users. |
| AccountEntity | User | Contains the frequently accessed information of accounts of the system’s users. |
| UserRoleEntity | N/A | Contains the information of users’ roles. |
| BookEntity | Book | Contains the information of books in the system. |
| BorrowedBookCopy Entity | N/A | Contains the information of copies that are borrowed by system’s users. |
| BookCopyEntity | Book Copy | Contains the information of copies of the system’s books. |
| WishBookEntity | N/A | Contains the information of books that the system’s users put in their wishlist. |
| AuthorEntity | Author | Contains the information of author in the system. |
| BookAuthorEntity | N/A | Contains the relational information of BookEntity and AuthorEntity |
| CategoryEntity | Category | Contains the information of category of the system’s book. Example of category are math, literature, or IT. |
| BookCategoryEntity | N/A | Contains the relational information of BookEntity and CategoryEntity |
| BookTypeEntity | Book Type | Contains the information of type of the system’s book. Example of book type are reference book, or textbook. |
| BookPositionEntity | Position | Contains the information of book position in the library. Example of book position is floor 1, shelf A. |

Table 14: Class Diagram Dictionary

### Class Diagram Explanation

*(References to main document, Section D – 4.2)*

### Interaction Diagram

#### Mobile Application

##### Unauthorized User

###### **<Unauthorized User> Login**

Summary: these diagrams show the process of unauthorized user login into the system using mobile application

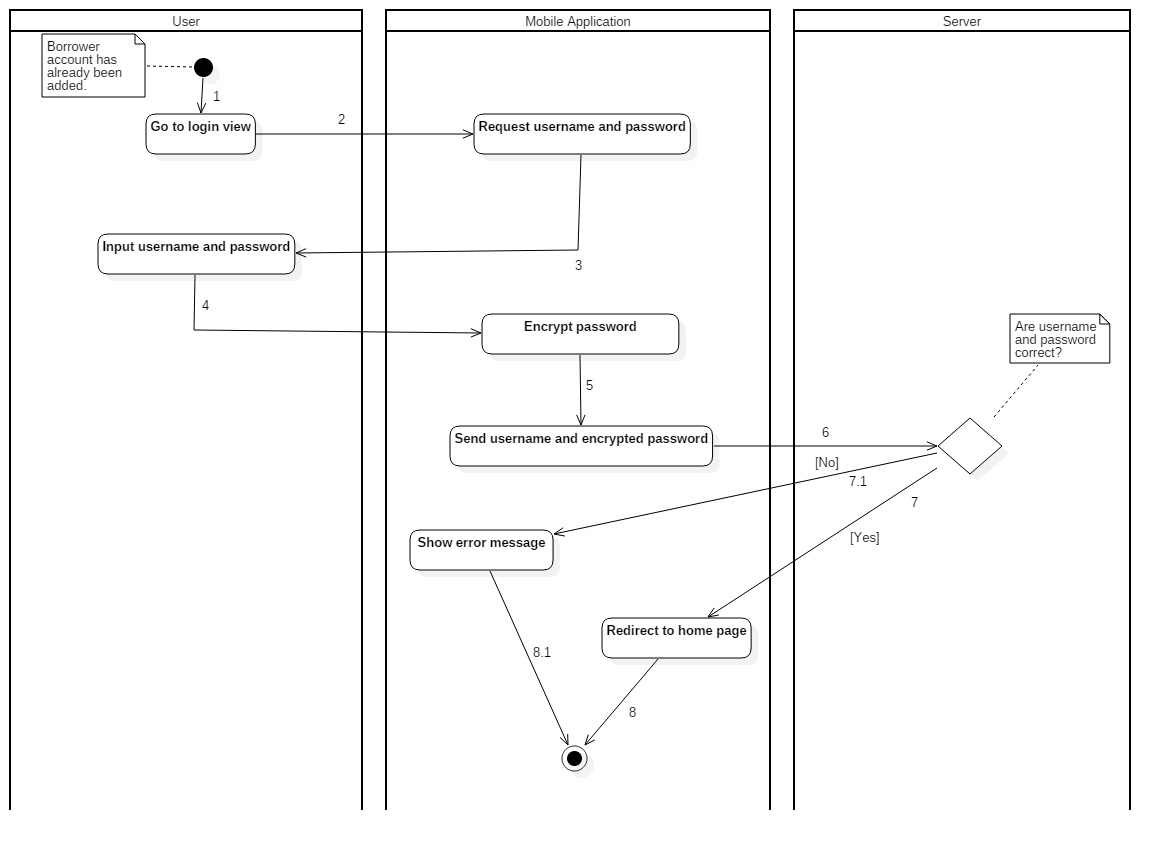


Figure 18: Activity Diagram - <Unauthorized User> Login

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Go to login view. | 3. Request username and password. |
| 4. Input username and password. | 5. Encrypt password. | 6. Send username and encrypted password. |
| 7. Username and password are correct. | 8. Redirect to home page. | 7.1. Username and password are not correct. |
| 8.1. Show error message. |  |  |

Table 15: <Unauthorized User> Login

##### Borrower

###### **<Borrower> Request new identification key**

Summary: this diagram shows the process of borrower request new identification key

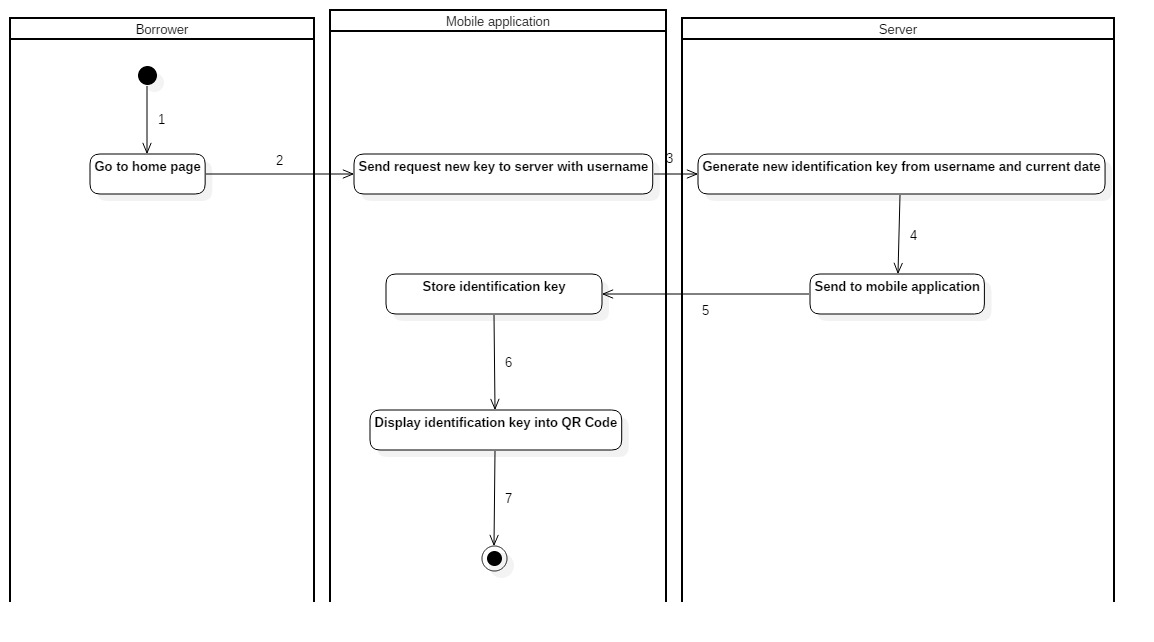


Figure 19: Activity Diagram - <Borrower> Request new identification key

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Go to home page. | 3. Send request new key to server with username. |
| 4. Generate new identification key from username and current date. | 5. Send to mobile application. | 6. Store identification key. |
| 7. Display identification key into QR Code. |  |  |

Table 20: Activity Diagram - <Borrower> Request new identification key

###### **<Borrower>** **Check in with identification key by NFC**

Summary: This diagram shows the process of borrower checking in the library by their smart phone with NFC turned on.

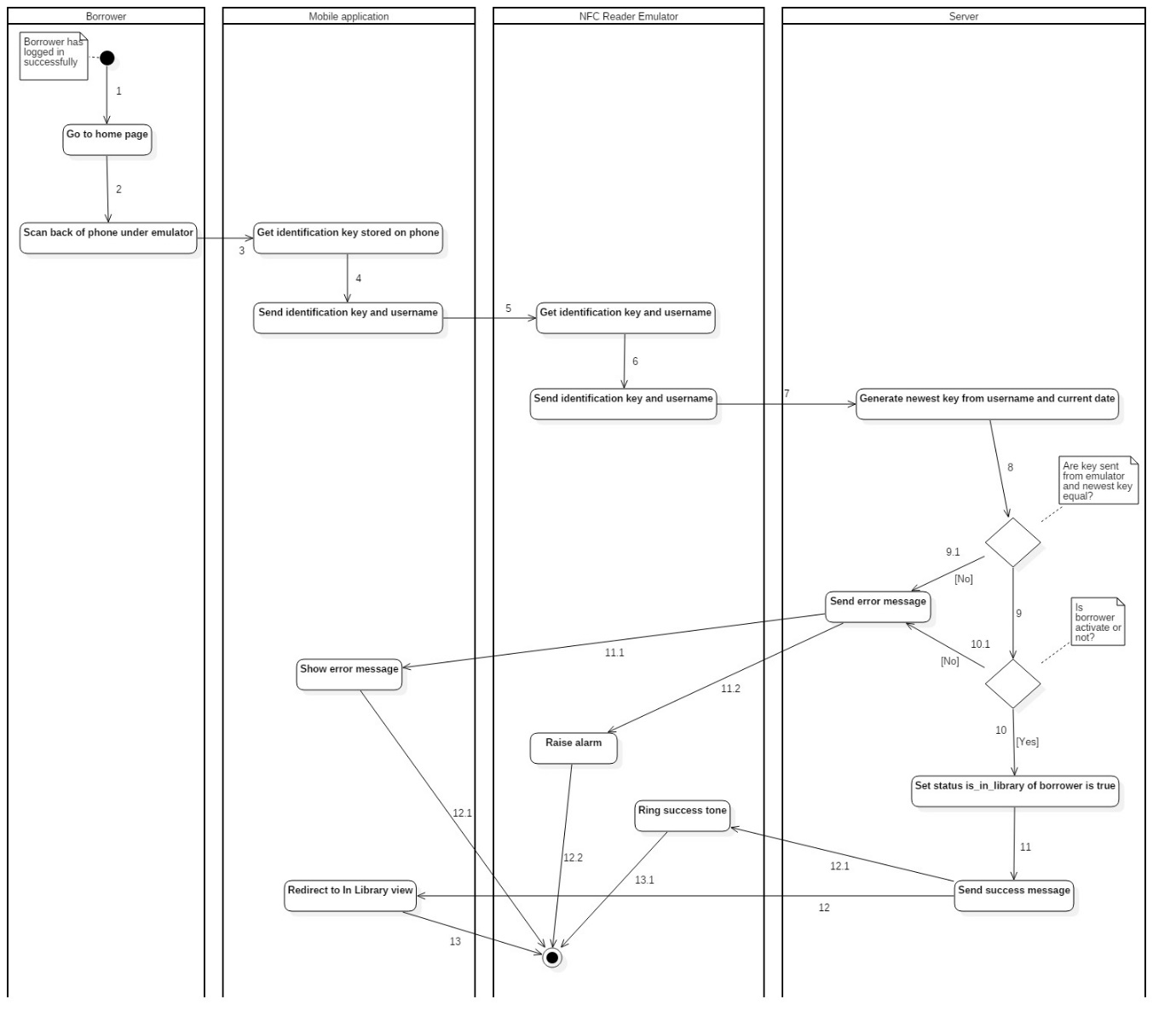


Figure 21: Activity Diagram - <Borrower> Check in with identification key by NFC

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Go to home page. | 3. Scan back of phone under emulator. |
| 4. Get identification key stored on phone. | 5. Send identification key and username. | 6. Get identification key and username. |
| 7. Emulator send identification key and username. | 8. Generate newest key from username and current date. | 9. Key sent form emulator and newest key are equal. |
| 10. Set status is\_in\_library of borrower is true. | 11. Send success message. | 12. Redirect to In Library view. |
| 9.1. Key sent form emulator and newest key are not equal. | 11.1. Send error message to mobile app. | 12.1. Mobile app show error message. |
| 10.1. Borrower is not activated. | 11.2. Send error message. | 12.2. Raise alarm. |

Table 16: <Borrower> Check in with identification key by NFC

###### **<Borrower> Check in with identification key by QR Code**

Summary: This diagram shows the process of borrower checking in the library by QR Code.

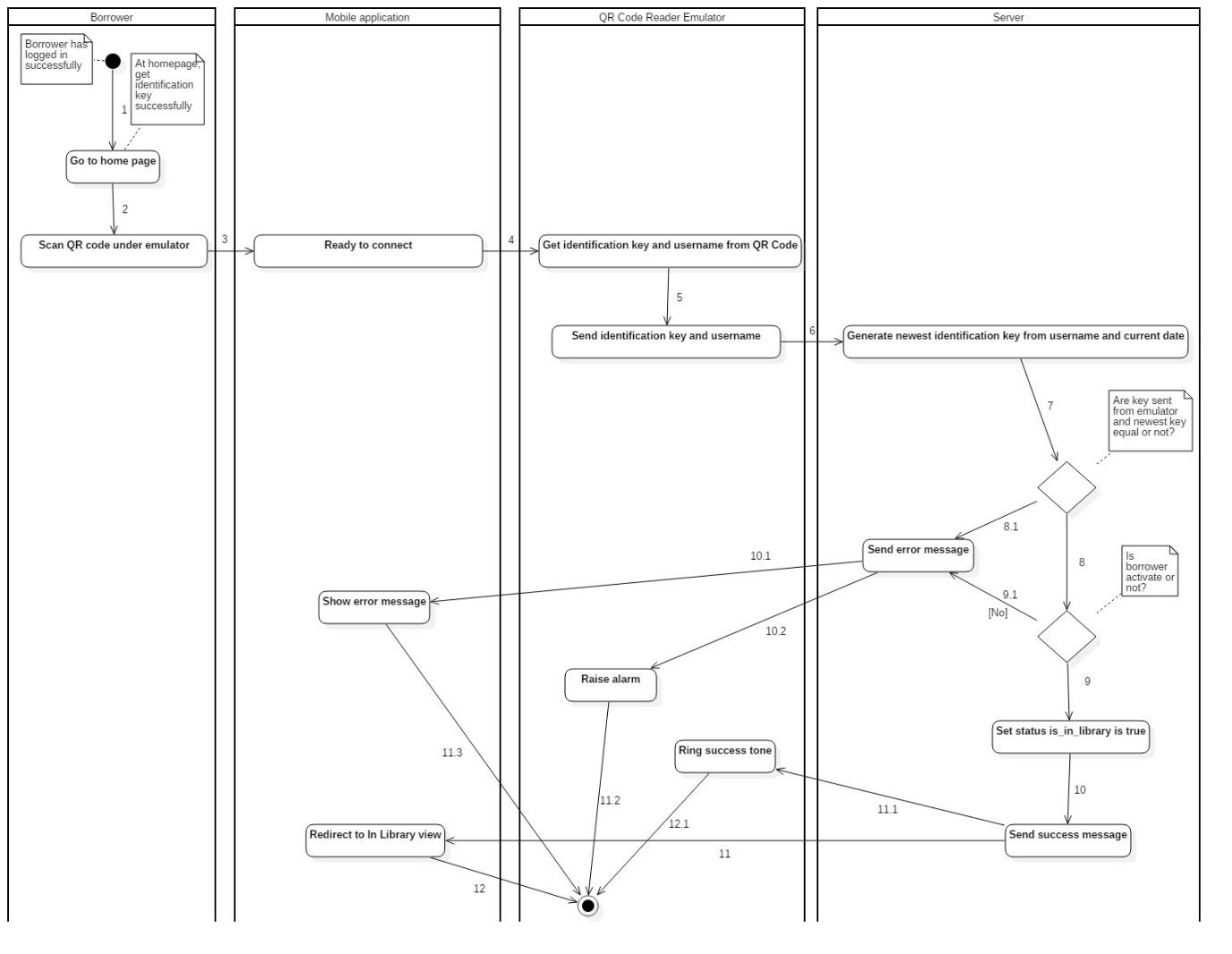


Figure 22: Activity Diagram - <Borrower> Check in with identification key by QR Code

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Go to Check in view. | 3. Send user Id to server. |
| 4. Generate newest key from user id. | 5. Receive key from server. | 6. User scan phone under emulator. |
| 7. Borrower phone display key into QR Code. | 8. Emulator connect, get key then send to server. | 9. Generate newest key from user Id. |
| 10b. Key from user and server are not equal. | 10.1b. Notify check in fail to borrower. | 10a. Key from user and server are equal. |
| 10.1a. Borrower phone redirect to In Library view. |  |  |

Table 17: <Borrower> Check in with identification key by QR Code

###### **<Borrower> Init check out**

Summary: This diagram shows the process of borrowers initiating check out.

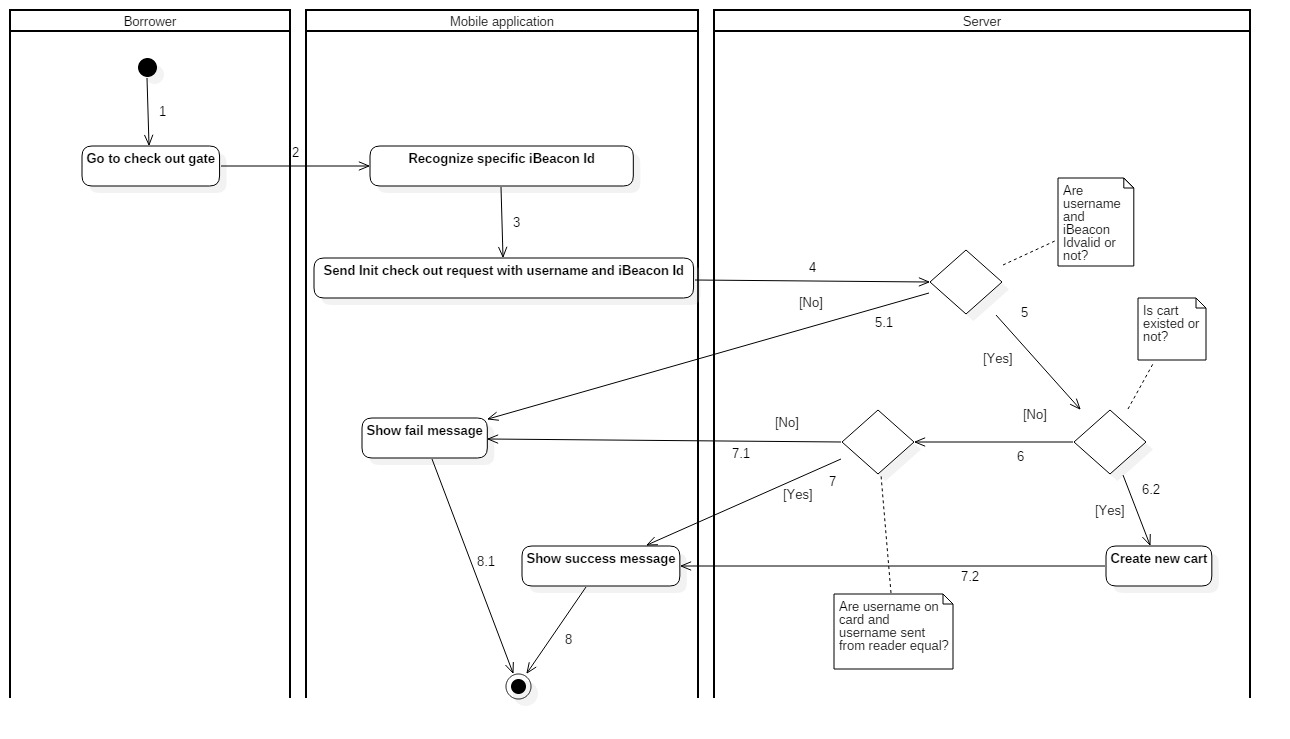


Figure 23: Activity Diagram - <Borrower> Init check out

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Go to check out gate. | 3. Recognize specific iBeacon Id. |
| 4. Send Init check out request with username and iBeacon Id. | 5. Username and iBeacon are valid. | 6.2. Cart is existed. |
| 7.2. Create new cart. | 8. Mobile show success message. | 5.1. Username or iBeacon Id is not valid. |
| 6.1. Show fail message. | 6. Cart is existed. | 7. Username on card is equal with username sent from reader. |
| 8. Mobile show success message. | 7.1 Username on card is not equal with username sent from reader. | 8.1 Mobile show fail message. |

Table 18: <Borrower> Init check out

###### **<Borrower> Scan book**

Summary: This diagram shows the process of borrowers scanning book.

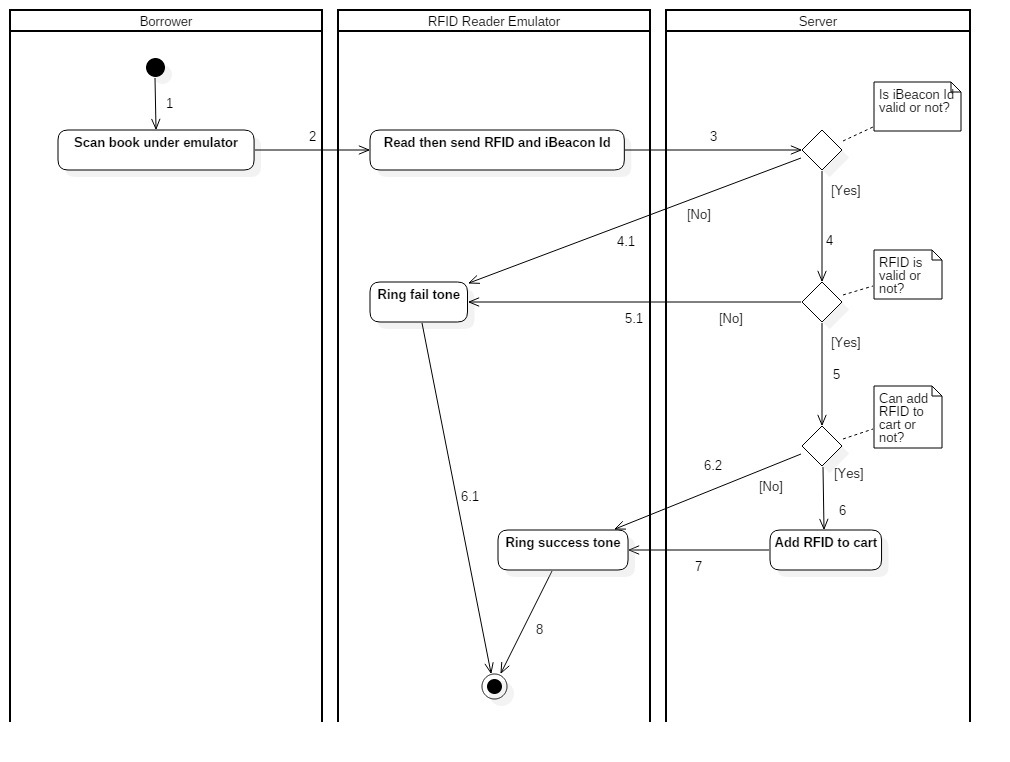


Figure 24: Activity Diagram - <Borrower> Scan book

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Borrower scan book under emulator. | 3. Emulator read then send RFID, iBeacon Id. |
| 4. iBeacon Id is valid. | 5. RFID is valid. | 6. Can add RFID to cart. |
| 7. Add RFID to cart. | 8. Mobile application ring success tone. | 4.1. iBeacon Id is not valid. |
| 5.1. RFID is not valid. | 6.1. Ring fail tone. | 6.2. Can not add RFID to cart. |

Table 19: <Borrower> Init check out

###### **<Borrower> Finish check out**

Summary: This diagram shows the process of borrowers finishing check out.

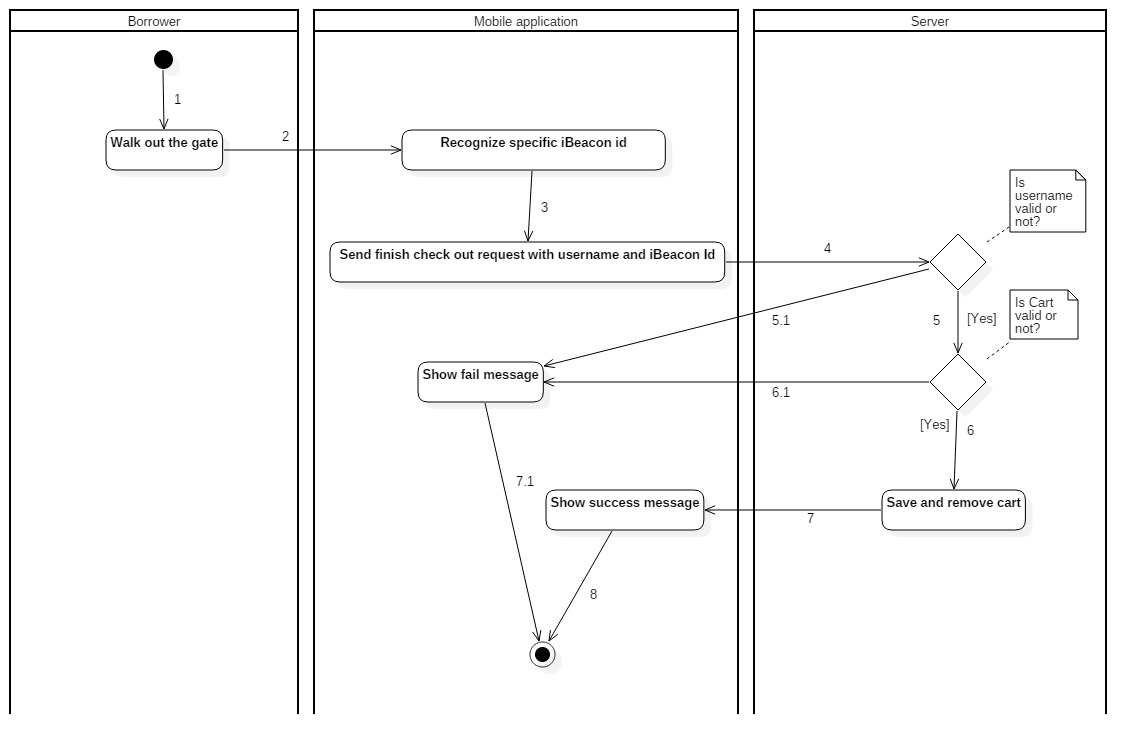


Figure 25: Activity Diagram - <Borrower> Finish check out

|  |  |  |
| --- | --- | --- |
| 1. Start. | 2. Walk out the gate. | 3. Recognize specific iBeacon Id. |
| 4. Send finish check out requeset with username and iBeacon Id. | 5. Username is valid. | 6. Cart is valid. |
| 7. Save and remove cart. | 8. Show success message. | 5.1. Username is not valid. |
| 6.1. Cart is not valid. | 7.1. Show fail message. |  |

Table 20: <Borrower> Finish check out

#### Web Application

##### Librarian Borrow Copies for Borrower

Summary: The diagrams show the process of librarian helping borrower to borrow book copies.

###### **<Librarian> Init borrow book copy**

Summary: This diagram shows the process of librarian initiating the borrow book copy procedure.

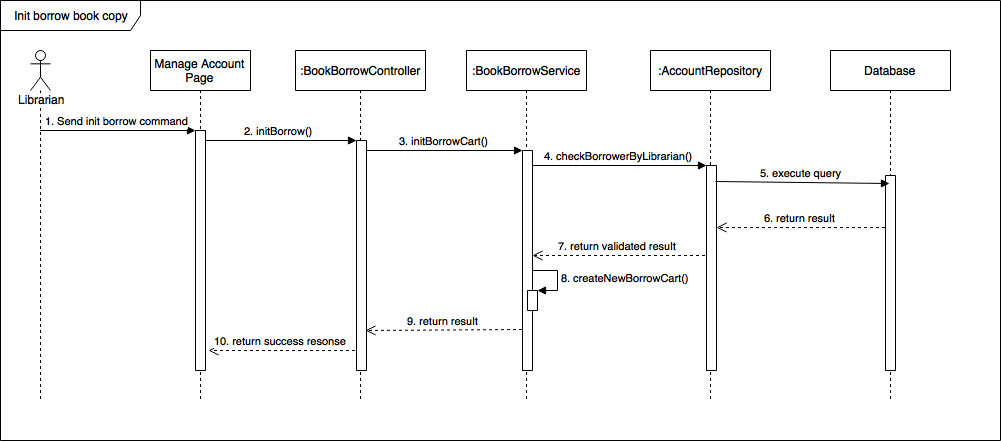


Figure 26: <Librarian> Init borrow book copy

###### **<Librarian> Scan borrow copy**

Summary: This diagram shows the process of librarian scanning borrow book copies for borrower.

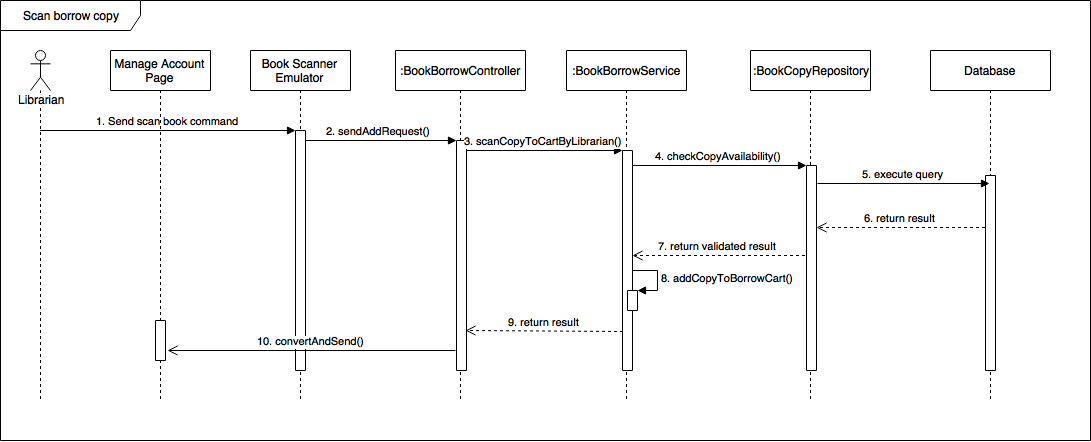


Figure 27: <Librarian> Scan borrow copy

###### **<Librarian> Submit borrow copy**

Summary: This diagram shows the process of librarian submitting the borrow copies for borrower.

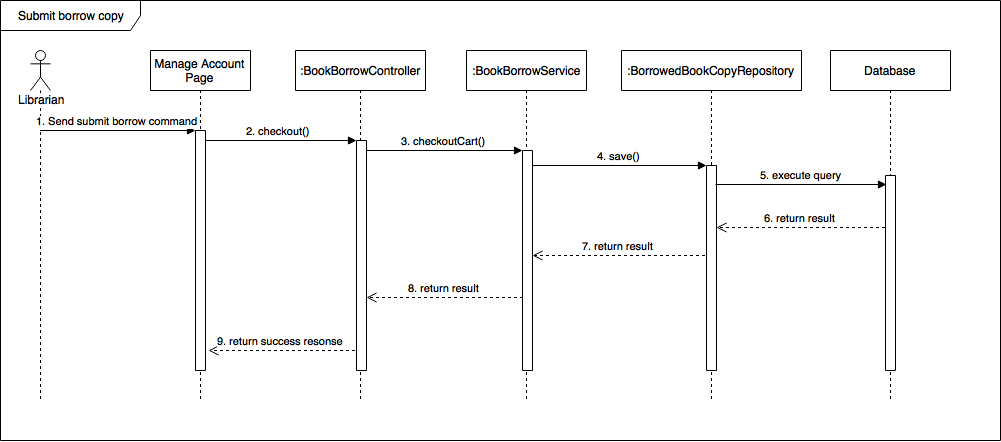


Figure 28: <Librarian> Submit borrow copy

##### Librarian Borrow Copies for Borrower

Summary: The diagrams show the process of librarian returning book copies from borrower to the library.

###### **<Librarian> Scan return copy**

Summary: This diagram shows the process of librarian scanning return book copies from borrower.

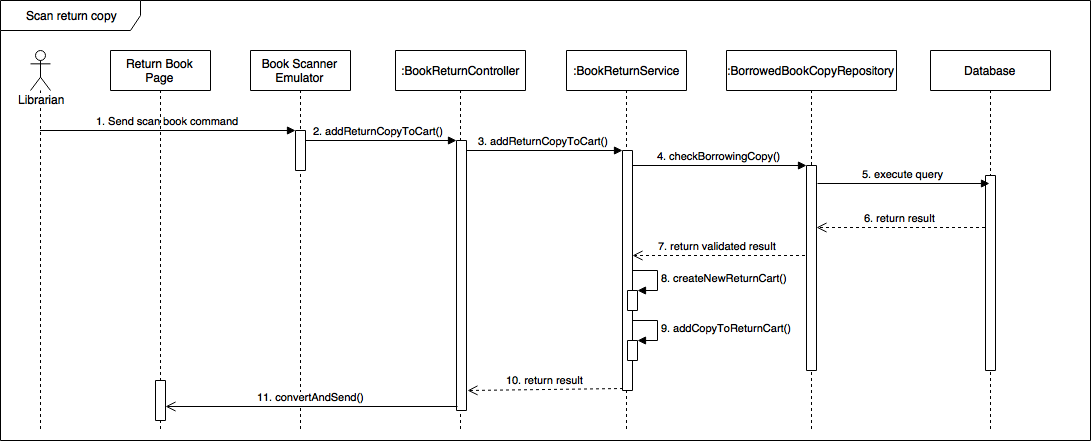


Figure 29: <Librarian> Scan return copy

###### **<Librarian> Submit return copy**

Summary: This diagram shows the process of librarian submitting the return copies from borrower.

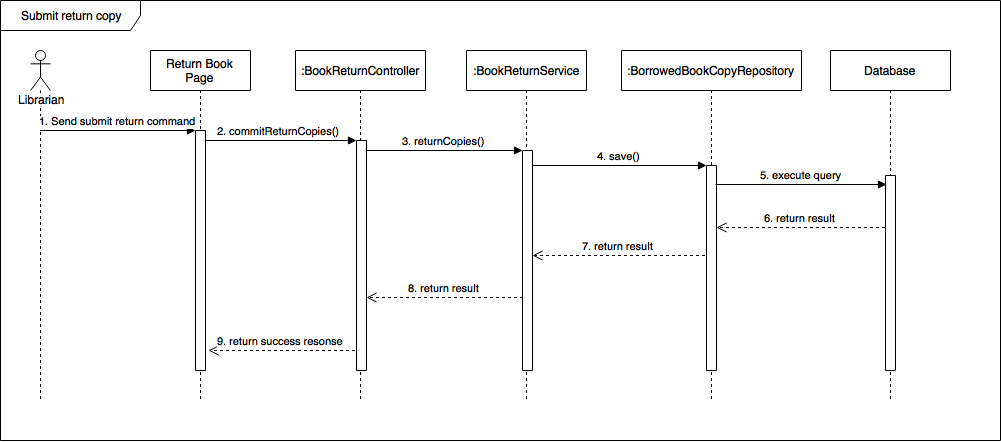


Figure 30: <Librarian> Submit return copy

## Database Design

### Entity Relationship Diagram (ERD)

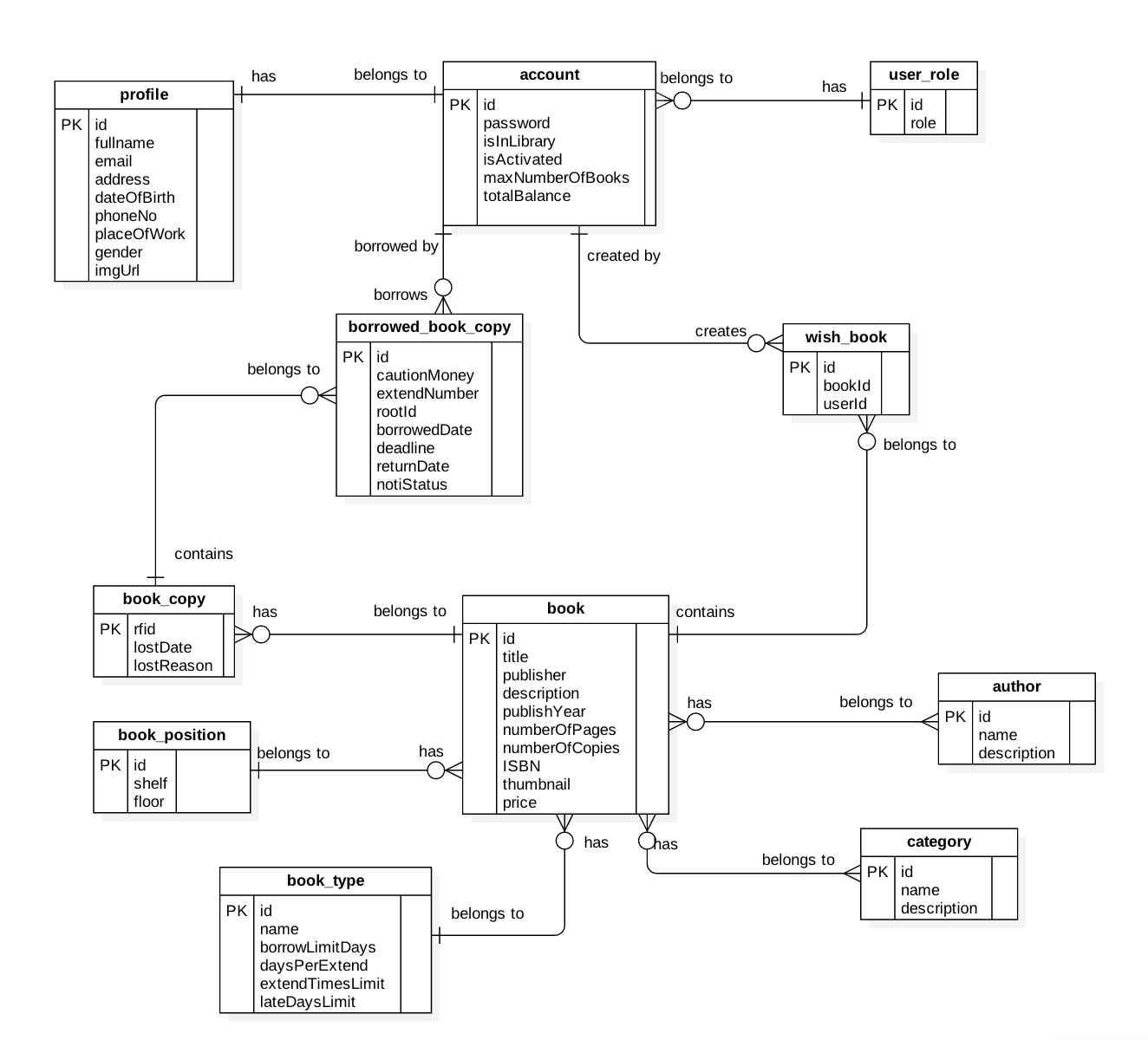


Figure 31: Entity Relationship Diagram (ERD)

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data Dictionary: describe contents of all entities** | |
| **Entity Name** | **Description** |
| Account | Contains frequently accessed information of all accounts in the system. |
| Profile | Contains information of detail profile of accounts in the system. |
| UserRole | Contains information of all role in the system. |
| WishBook | Contains a list of books that the accounts in the system put in their wishlist. |
| BorrowedBookCopies | Contains a list of borrowed book copies that the accounts in the system borrowed. |
| BookCopy | Contains information of all book copies of books in the system. |
| Book | Contains information of all books in the system. |
| BookPosition | Contains information of all book positions in the system. |
| BookType | Contains information of all book types in the system. |
| Category | Contains information of all book categories in the system. |
| Author | Contains information of all book authors in the system. |

Table 21: ERD Data Dictionary

## Algorithms

### Check in Algorithm

#### Definition

* The check in algorithm proposes method to help the system be aware of the present of a borrower in the library.
* Checking in has two meaning to the system:
  + - Help librarian to identify a borrower before giving the access right to that borrower.
    - Record the present of the borrower to the system, to enable the book – borrowing method for that borrower.

#### Define the problem

* To prevent robbery, book damaging, and other delinquencies that may happen in the library, a borrower needs to check in so that the librarian and the system can verify his/her authentication.
* To check in, each borrower needs a unique identification key.
* Someone may take another’s identification key to have long-term access right to the library.
* With tradition library card, a librarian must check borrower’s information by himself/herself. The job can be inaccurate if the librarian has a long working day.

#### Solution

To solve those problems, the check in algorithm has the following features:

* Use a check in device that has both sound and text message to assist librarian in identifying a borrower.
* Use the borrower’s userID and server’s current date to encode a unique identification key for that borrower in the current date. Therefore, an identification key is:
  + - Unique for each user
    - Valid for only one day
* For encoding, the system uses **Secure Hash Algorithm 1.**
* Borrower can use either QR Code or NFC in his/her smart phone to check in with the check in device. The check in device will interpret the information in borrower’s phone and send the system’s server for validation.

To be able to provide those features, the check in algorithm includes two children processes:

* + - Generate identification key for borrower.
    - Check in with the given identification key.
      * 1. **Generate identification key**

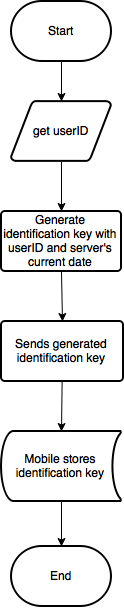
****

Figure 32: Generate identification key flow chart

**Check in with identification key**

* **Setup:**

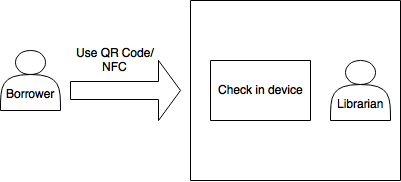
****

Figure 33: Check in setup

|  |  |
| --- | --- |
| **Name** | **Description** |
| Check in device | Device that can scan both QR Code and NFC, interpret the scanned data to borrower’s identification key, then send the key to server for validation. |

Table 22: Check in device dictionary

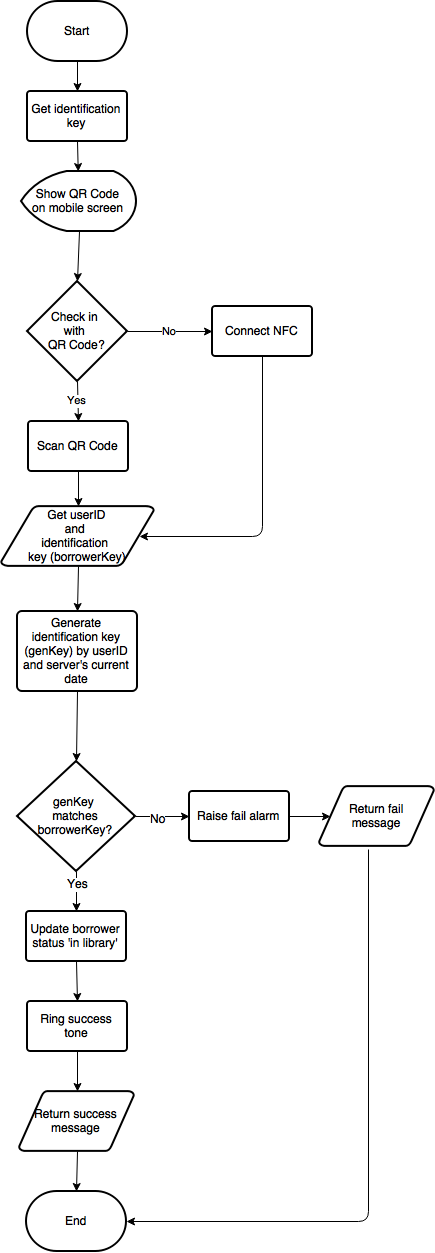


Figure 34: Check in with identification key flow chart

#### References

* Secure Hash Algorithm 1: <https://en.wikipedia.org/wiki/SHA-1>

### Automatic book - borrowing algorithm

#### Definition

* The automatic book - borrowing algorithm proposes method to automate the book – borrowing process in the library.
* Automatic book - borrowing algorithm has two meaning to the system:
  + - Help librarian reduce the work load.
    - Provide borrower an easier and faster way to borrow books.

#### Define the problem

* Traditional book – borrowing process can be time – consuming and tiring for both librarian and borrower.
  + In order to borrow books, a borrower has to follow a long process, including bringing books to the librarian and fill in borrow forms.
  + A librarian has to manually check a borrower’s forms and books.
* Traditional book – borrowing process may lead to long waiting lines if there are many borrowers.

#### Solution

To solve those problems, the book - borrowing algorithm needs the following setup:

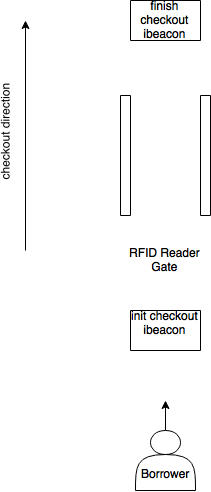


Figure 35: Book-borrowing devices setup

|  |  |
| --- | --- |
| **Name** | **Description** |
| Init checkout ibeacon | Ibeacon to trigger a server – call from borrower’s mobile phone, so that the server knows which borrower is walking out to create a borrow cart for that borrower. |
| RFID Reader Gate | RFID Reader Gate to scan books from borrower and send to system’s server. The Gate also sends init checkout ibeacon’s id for the server to find the borrow cart of the current borrower. |
| Finish checkout ibeacon | Ibeacon to trigger a server – call from borrower’s mobile phone, so that the server call necessary services to save the borrower’s current borrow cart to database. |

Table 23: Book - borrowing devices dictionary

#### Process

The book – borrowing algorithm needs three child processes:

* Init checkout
* Scan book
* Finish checkout

#### Flow Chart

#### Init checkout

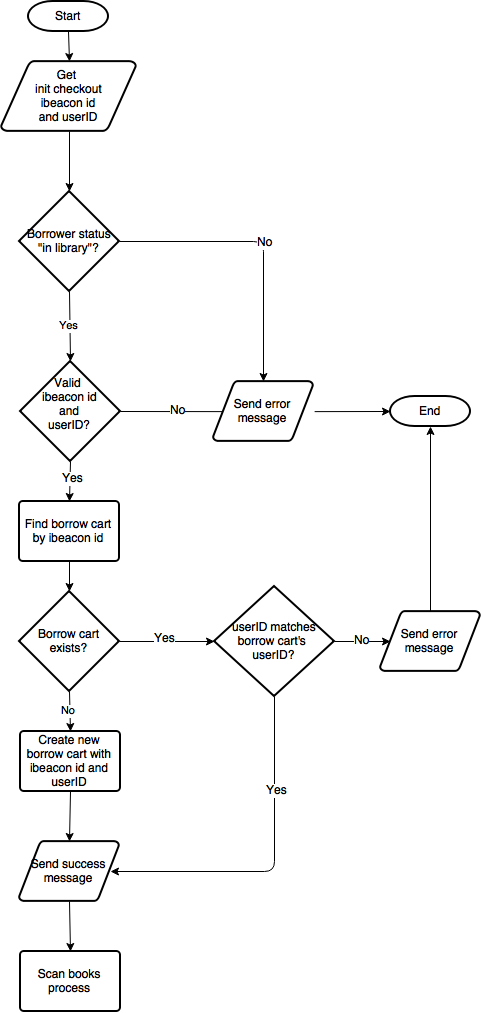


Figure 36: Init checkout flow chart

#### Scan book

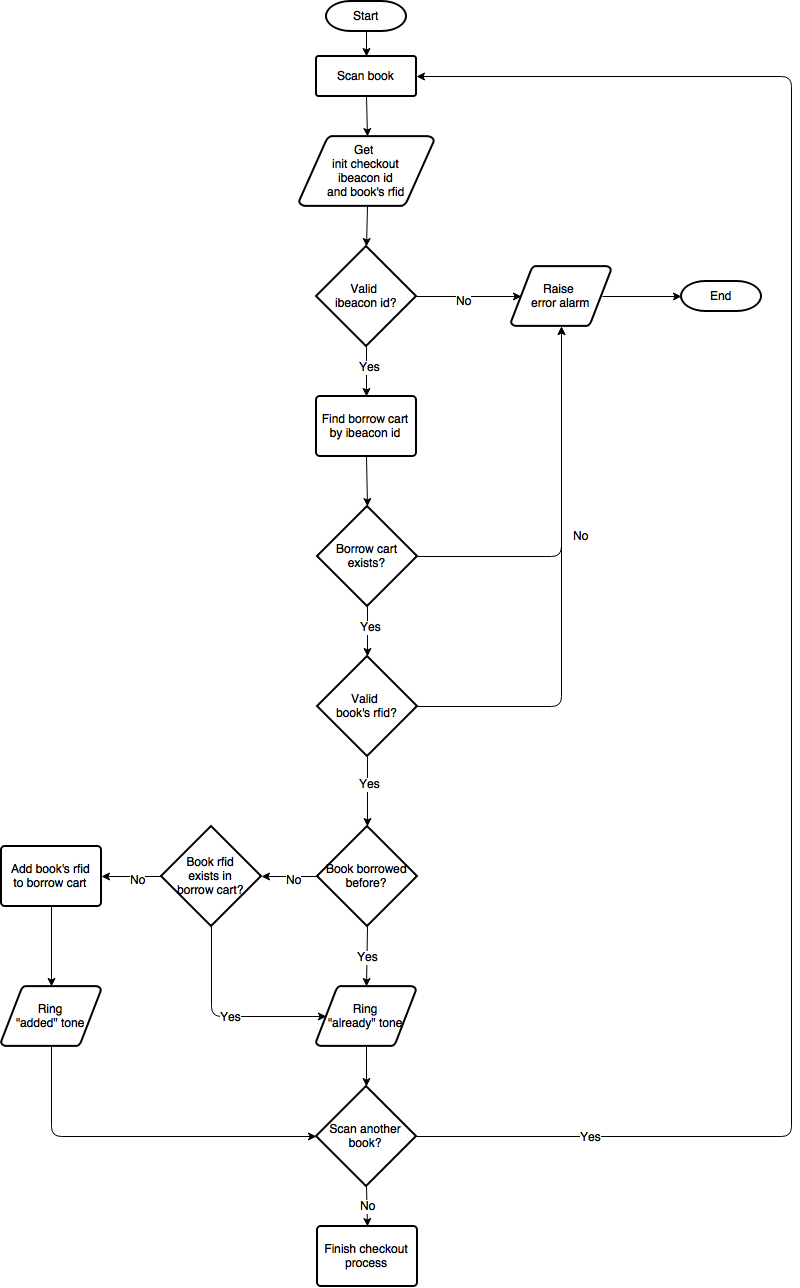


Figure 37: Scan book flow chart

#### Finish checkout

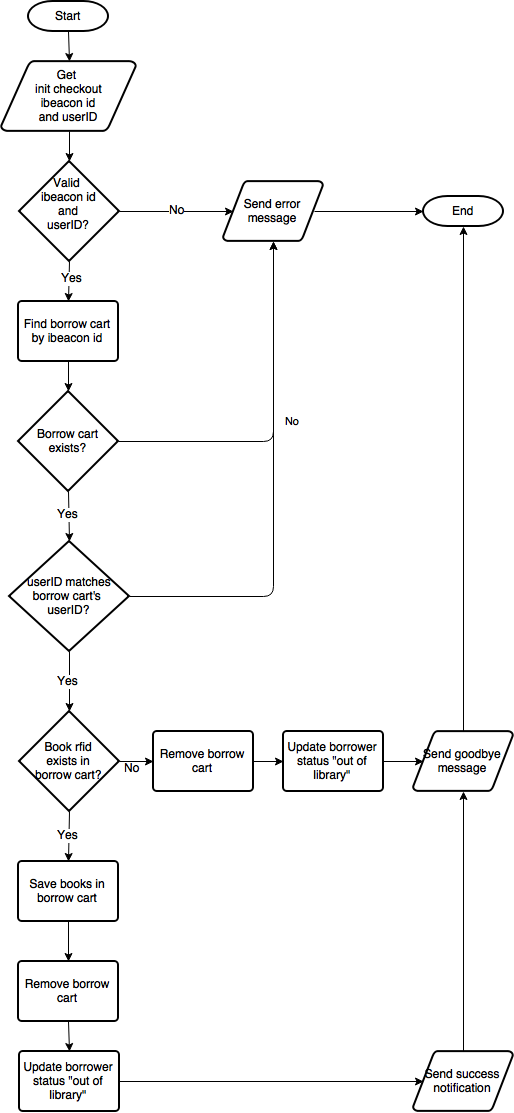


Figure 38: Finish checkout flow chart

### Interact with Estimote ibeacon algorithm

#### Definition

* Interact with Estimote ibeacon algorithm proposes method to help JWL’s mobile application interact with ibeacon in order to commit needed server-call described in book – borrowing algorithm.

#### Define the problem

As described in book – borrowing algorithm, the system needs to know:

* When a borrower starts to check out, along with needed information to create a borrow cart for that borrower.
* When the borrower finishes check out along with needed information to save the books in his/her borrow cart.

#### Solution

Interact with Estimote ibeacon algorithm make use of Estimote ibeacon’s features:

* An ibeacon constantly broadcast a tiny packet of data, including its:
  + Unique MAC address: the system will use this field as ibeacon id.
  + Major number: the system will use this field to identify an ibeacon’s type. Init checkout ibeacon has major number = 1, and finish checkout ibeacon has major number = 2.
* Any phone with Bluetooth Low Energy turned on can read data and trigger an action.

Therefore, the algorithm solves the problems defined in 7.3.2 as below:

#### Init checkout

Borrower enters the init checkout ibeacon’s range.

Borrower’s phone, with JWL’s mobile application (JLib) turned on, detects the init ibeacon (major number = 1).

JLib saves the init ibeacon id on the phone’s storage.

JLib calls the init checkout api and sends the init ibeacon id and borrower’s userID to server.

#### Finish checkout

Borrower enters the finish checkout ibeacon’s ranges.

Borrower’s phone, with JLib turned on, detects the checkout ibeacon (major number = 2).

JLib calls the finish checkout api and sends the saved init ibeacon id and borrower’s userID to server.

#### Reference

Estimote ibeacon: <http://developer.estimote.com/ibeacon/>

### Save book copy with RFID

#### Definition

* Radio-frequency identification (RFID) uses electromagnetic fields to identify tags that attached to objects. RFID Reader does not need to have the tags in its line of sight.

#### Define the problem

To automate the book – borrowing process, the system needs to have a way to identify each book quickly and exactly.

#### Solution

To solve the problem, we apply the advantages of RFID technology to our system. The system has:

* RFID readers set up
  + as a “gate” to recognize books that borrowers borrow (described in book – borrowing algorithm),
  + as a “book scanner” for librarian to help borrowers to borrow and return books.
* RFID tags that stick to each book copy to act as a copy’s unique identification. Thus, book copies are saved in the system database as below:

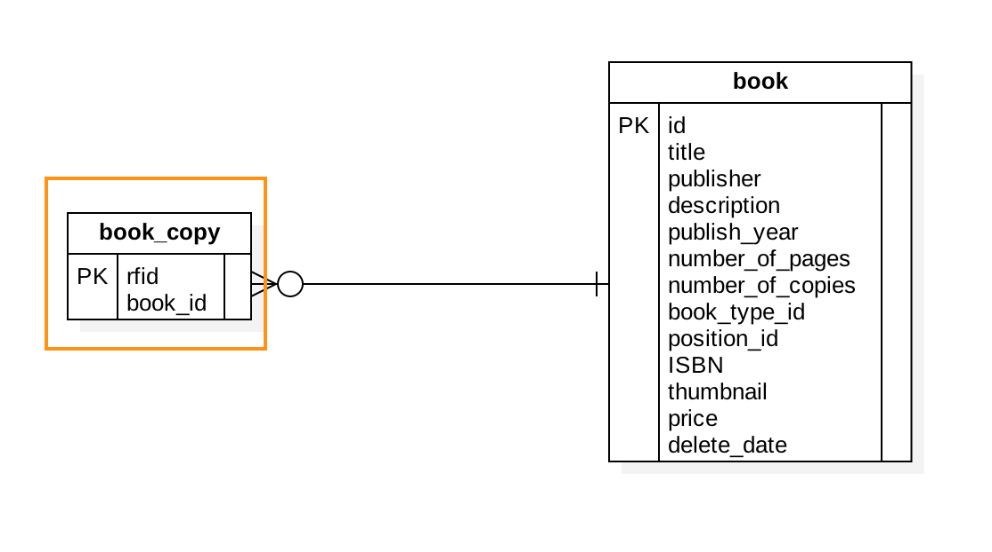


Figure 39: Saving book copy with rfid

Therefore, each time a book copy is scanned to a RFID reader, the system recognizes it immediately and takes proper action according to the situation.

#### Reference

RFID: <https://en.wikipedia.org/wiki/Radio-frequency_identification>

# System Implement and Test

## Database Relationship Diagram

### Physical Diagram



Table 24: Physical Diagram

### Data Dictionary

*(References to main document, Section E – 2.2)*

# Task sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Product Deliverables** | **Task** | **HaVH** | **AnhNT** | **ThienDN** | **Size** |
| 1 | Report1 - Introduction | Project Information |  | **O** |  |  |
| Introduction |  | **O** |  |  |
| Current Situation |  | **O** |  |  |
| Problem Definition |  |  | **O** |  |
| Proposed Solution |  |  | **O** |  |
| Functional Requirements | **O** |  |  |  |
| Roles and Responsibilities | **O** |  |  |  |
| 2 | Report2- Software Project Management Plan | Problem Definition |  |  |  |  |
| Project organization |  |  |  |  |
| Project Management Plan |  |  |  |  |
| Coding Convention | **O** |  |  |  |
| 3 | Report 3- Software Requirement Specification | [User Requirement Specification](#_Toc479512100) |  |  |  |  |
| [Unauthorized User Requirement](#_Toc479512101) | **O** |  |  |  |
| [Authorized UserRequirement](#_Toc479512102) | **O** |  |  |  |
| [Staff Requirement](#_Toc479512103) | **O** |  |  |  |
| [Borrower Requirement](#_Toc479512104) | **O** |  |  |  |
| [Librarian Requirement](#_Toc479512105) | **O** |  |  |  |
| [Admin](#_Toc479512106) Requirement | **O** |  |  |  |
| [A](#_Toc479512107)utomatic Handler Requirement | **O** |  |  |  |
| Emulator Requirement | **O** |  |  |  |
| [Software Requirement Specification](#_Toc479512108) |  |  |  |  |
| [External Interface Requirement](#_Toc479512109) | **O** |  |  |  |
| [System Overview Use Case](#_Toc479512110) | **O** |  |  |  |
| **List of usecase** |  |  |  |  |
| [<Unauthorized User>](#_Toc479512179) Login |  | **O** |  |  |
| [<Authorized User>](#_Toc479512180) Get Profile Detail |  | **O** |  |  |
| [<Authorized User>](#_Toc479512180) Logout |  | **O** |  |  |
| [<Authorized User>](#_Toc479512180) Edit |  | **O** |  |  |
| [<Admin>](#_Toc479512183) Create Account |  | **O** |  |  |
| [<Admin>](#_Toc479512184) Get Account Detail |  | **O** |  |  |
| [<Admin>](#_Toc479512184) Update Account |  | **O** |  |  |
| [<Admin>](#_Toc479512184) Delete Account |  | **O** |  |  |
| [<Librarian>](#_Toc479512187) Search Book |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Add Book |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Get Book Detail |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Update Book |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Remove Book |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Add Borrower |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Search Borrower |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Get Borrower Detail |  |  | **O** |  |
| [<Librarian>](#_Toc479512187) Update Borrower |  |  | **O** |  |
| [<Borrower>](#_Toc479512198) Search Book |  |  | **O** |  |
| [<Borrower>](#_Toc479512199) Add Book To Wish List |  |  | **O** |  |
| [<Borrower>](#_Toc479512201) Get Borrowed Book |  |  | **O** |  |
| [<Borrower>](#_Toc479512203) Renew Book |  |  | **O** |  |
| [<Automatic Handler>](#_Toc479512204) Check in Borrower | **O** |  |  |  |
| [<Automatic Handler>](#_Toc479512204) Notify Return Date | **O** |  |  |  |
| [<Automatic Handler>](#_Toc479512204) Notify Borrowing Books. | **O** |  |  |  |
| [<Automatic Handler>](#_Toc479512204) Notify Available Books. | **O** |  |  |  |
| [<Automatic Handler>](#_Toc479512204) Check out Borrower | **O** |  |  |  |
| [<Automatic Handler>](#_Toc479512204) Check out Borrow Books | **O** |  |  |  |
| [Software System](#_Toc479512112) Attribute | **O** |  |  |  |
| [Conceptual Diagram](#_Toc479512113) |  | **O** |  |  |
| 4 | Report 4- Software Design Description | [Design Overview](#_Toc479512115) |  | **O** |  |  |
| [System Architectural Design](#_Toc479512116) |  |  |  |  |
| [Web Application Architecture Description](#_Toc479512117) |  | **O** |  |  |
| [API Application Architecture Description](#_Toc479512118) |  | **O** |  |  |
| [Mobile Application Architecture Description](#_Toc479512119) | **O** |  |  |  |
| [Component Diagram](#_Toc479512120) |  | **O** |  |  |
| [Detailed Description](#_Toc479512121) |  |  |  |  |
| [Class Diagram](#_Toc479512122) |  |  | **O** |  |
| [Class Diagram Explanation](#_Toc479512123) |  |  | **O** |  |
| [Interaction Diagram](#_Toc479512124) |  |  | **O** |  |
| [Interface](#_Toc479512125) |  |  |  |  |
| [Component interface](#_Toc479512126) |  |  | **O** |  |
| [User Interface Design](#_Toc479512127) |  |  | **O** |  |
| [Database Design](#_Toc479512128) |  |  |  |  |
| [Entity Relationship Diagram (ERD)](#_Toc479512129) | **O** |  |  |  |
| [Data Dictionary](#_Toc479512130) | **O** |  |  |  |
| [Algorithms](#_Toc479512131) |  |  |  |  |
| [Check in Algorithms](#_Toc479512132) |  | **O** |  |  |
| [Automatic Book - Borrowing Algorithm](#_Toc479512133) |  | **O** |  |  |
| [Interact with Estimote iBeacon Algorithm](#_Toc479512134) |  | **O** |  |  |
| [Save Book Copy with RFID Algorithm](#_Toc479512135) |  | **O** |  |  |
| 5 | Report 5 - Software Implementation and Test Document | [Introduction](#_Toc479512140) |  |  |  |  |
| [Overview](#_Toc479512141) | **O** |  |  |  |
| [Test Approach](#_Toc479512142) | **O** |  |  |  |
| [Database Relationship Diagram](#_Toc479512143) |  |  |  |  |
| [Physical Diagram](#_Toc479512144) |  |  | **O** |  |
| [Data Dictionary](#_Toc479512145) | **O** |  |  |  |
| [Implementation](#_Toc479512146) |  |  |  |  |
| [API Application Implementation](#_Toc479512147) |  |  | **O** |  |
| [Mobile Application Architecture Implementation](#_Toc479512148) | **O** |  |  |  |
| [Web Architecture Implementation](#_Toc479512149) |  | **O** |  |  |
| Emulator Implementation | **O** |  |  |  |
| [Test Plan](#_Toc479512151) |  |  |  |  |
| [Communication Diagram](#_Toc479512152) | **O** |  |  |  |
| [System Testing Test Case](#_Toc479512153) |  |  |  |  |
| [Web Application Test Cases](#_Toc479512154) | **O** |  |  |  |
| [Mobile Application Test Cases](#_Toc479512155) |  | **O** | **O** |  |
|  | Database | Create database schema |  | **O** | **O** | 3 |
| Insert default data schema |  | **O** | **O** | 1 |
|  | Framework | Design server code structure |  | **O** |  | 5 |
| Design mobile code structure | **O** |  |  | 5 |
| Design web code structure |  | **O** |  | 5 |
|  | **Algorithm** | **Implement check in algorithm** |  |  | **O** | **5** |
| **Implement automatic book – borrowing algorithm** | **O** |  |  | **5** |
| **Implement interact with ibeacon algorithm** | **O** |  |  | **5** |
| **Implement save book copy with RFID algorithm** | **O** |  |  | **5** |
|  | Web | Implement add account |  | **O** |  | 1 |
| Implement return book |  | **O** |  | 1 |
| Implement borrow book |  | **O** |  | 1 |
| Implement get account detail |  | **O** |  | 3 |
| Implement search account |  | **O** |  | 1 |
| Implement get book detail |  | **O** |  | 2 |
| Implement update borrower balance |  | **O** |  | 1 |
|  | Server | Implement book borrow component |  |  | **O** | 1 |
| Implement book return component | **O** |  |  | 3 |
| Implement management component |  |  | **O** | 3 |
| Implement scheduler component | **O** |  |  | 3 |
|  | Mobile | Implement borrow book | **O** |  |  | 1 |
| Implement get borrowing book list | **O** |  |  | 2 |
| Implement get borrowed book list | **O** |  |  | 2 |
| Implement renew book | **O** |  |  | 1 |
| Implement check in | **O** |  |  | 1 |
| Implement checkout | **O** |  |  | 1 |
| Implement add to wishlist | **O** |  |  | 1 |
| Implement notification | **O** |  |  | 2 |
|  | Emulator | Implement check in |  | **O** |  |  |
| Implement scan book |  | **O** |  |  |
| 6 | Report 6 - Software User's Manual | [Installation Guide](#_Toc479512157) |  |  |  |  |
| [Setting up environment at server side](#_Toc479512158) |  |  | **O** |  |
| [Web Services Application Deployment Process](#_Toc479512159) |  | **O** |  |  |
| [Mobile Application Deployment Process](#_Toc479512160) | **O** |  |  |  |
| Emulator | **O** |  |  |  |
| [User Guide](#_Toc479512161) |  |  |  |  |
| [Mobile application](#_Toc479512162) |  |  | **O** |  |
| [Web application](#_Toc479512163) |  | **O** |  |  |
| Emulator |  | **O** |  |  |

# Appendix

1. NFC:

<http://nearfieldcommunication.org/>

1. QR Code:

<https://en.wikipedia.org/wiki/QR_code>

1. RFID:

<http://www.technovelgy.com/ct/technology-article.asp>

<https://en.wikipedia.org/wiki/Radio-frequency_identification>

1. RFID Module Reader:

<http://www.dientubachviet.vn/498-rfid-rdm6300-125khz-uart-output.html>

1. UART Converter:

<http://hshop.vn/products/mach-chuyen-usb-uart-ft232rl>

1. RFID tag/card:

<http://hshop.vn/products/the-trang-rfid-125khz>

1. iBeacon:

<http://estimote.com>

<http://developer.estimote.com/ibeacon/>

1. Flux

<https://facebook.github.io/flux/>

1. Spring

<https://spring.io/docs>

1. SHA-1

<https://en.wikipedia.org/wiki/SHA-1>