

**Development and Management of the Project: UWON Library System**

Team Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Student 1 | Student 2 | Student 3 | Student 4 |
| Name |  |  |  |  |

# Table of Content

[Table of Figures 4](#_TOC_250082)

[Table of Tables 5](#_TOC_250081)

[Abstract 7](#_TOC_250080)

[Project Plan 8](#_TOC_250079)

1. [Introduction 8](#_TOC_250078)
   1. [System-As-Is 8](#_TOC_250077)
   2. [System-to-Be 8](#_TOC_250076)
   3. [Project Deliverables 8](#_TOC_250075)
2. [Process Model 9](#_TOC_250074)
   1. [Activities to be Undertaken 9](#_TOC_250073)
   2. [Milestone to Measure Progress 10](#_TOC_250072)
3. [Organization of the Project 10](#_TOC_250071)
   1. [Information, services, resources, and facilities to be provided 10](#_TOC_250070)
   2. [Specification of the roles 10](#_TOC_250069)
4. [Methods and techniques 11](#_TOC_250068)
5. [Standards, guidelines, procedures 11](#_TOC_250067)
6. [Work packages 12](#_TOC_250066)
7. [Resources 12](#_TOC_250065)
   1. [Hardware 12](#_TOC_250064)
   2. [Personnel 12](#_TOC_250063)
8. [Effort and schedule 13](#_TOC_250062)
   1. [Effort 13](#_TOC_250061)
   2. [Schedule 19](#_TOC_250060)
9. [Delivery 20](#_TOC_250059)
   1. [Login procedure: 20](#_TOC_250058)
   2. [Reservation procedure: 20](#_TOC_250057)
   3. [Search procedure: 21](#_TOC_250056)
   4. [Intelligent Report procedure: 21](#_TOC_250055)

[Configuration Plan 22](#_TOC_250054)

1. [Introduction 22](#_TOC_250053)
   1. [Conventions: 22](#_TOC_250052)
   2. [Terms: 22](#_TOC_250051)
   3. [Abbreviations Identify Configuration Manager (s): 22](#_TOC_250050)
2. [Software Configuration Management 22](#_TOC_250049)
3. [Identify Configuration Items 23](#_TOC_250048)
4. [Identify Responsible for each Configuration Item 23](#_TOC_250047)
5. [Tools 23](#_TOC_250046)
6. [Environment and Infrastructure 23](#_TOC_250045)
7. [Configuration Policy 23](#_TOC_250044)
   1. [Project Plan: 24](#_TOC_250043)
   2. [Configuration Plan 24](#_TOC_250042)
   3. [Requirements Specification 24](#_TOC_250041)
   4. [Design Specification 24](#_TOC_250040)
   5. [Paper-based Prototype 25](#_TOC_250039)
   6. [Test Plan 25](#_TOC_250038)
8. [Document Version 25](#_TOC_250037)
9. [Evolution of the Version Number 25](#_TOC_250036)
10. [Configuration and Change Control 26](#_TOC_250035)
11. [System Building 26](#_TOC_250034)
12. [Release Management 27](#_TOC_250033)
13. [Contingency plan 27](#_TOC_250032)

[Requirements Specification 28](#_TOC_250031)

1. [High-Level Goals (Unique Identifiers) 28](#_TOC_250030)
2. [Primary and Secondary Actors 28](#_TOC_250029)
3. [Use Case diagram (s) 28](#_TOC_250028)
4. [Use Case Specification (s) 28](#_TOC_250027)
5. [Assumptions 36](#_TOC_250026)
   1. [Credentials 36](#_TOC_250025)
   2. [Security 36](#_TOC_250024)
6. [Domain Properties 36](#_TOC_250023)
7. [Functional Requirements 36](#_TOC_250022)
8. [Non-Functional Requirements 38](#_TOC_250021)
   1. [Product requirements: 38](#_TOC_250020)
   2. Organizational requirements: 39
   3. External requirements: 40
9. Traceability Matrix 40

[Design 42](#_TOC_250019)

1. [Entity Relationship Diagram (conceptual model) 42](#_TOC_250018)
2. [Logical Model 43](#_TOC_250017)
3. [Class Diagram 44](#_TOC_250016)
4. [Sequence Diagram 45](#_TOC_250015)
   1. [Make a Book Reservation 45](#_TOC_250014)
   2. [Manage Loan 46](#_TOC_250013)
5. [State Machine Diagram 46](#_TOC_250012)
   1. [Make a Book Reservation 46](#_TOC_250011)
   2. [Manage Loan 47](#_TOC_250010)
6. [System Architecture View and Style/Pattern 48](#_TOC_250009)

[Paper-based Prototype **49**](#_TOC_250008)

1. [Technical Description of the System 49](#_TOC_250007)

[Testing Plan 52](#_TOC_250006)

1. [Test Scenarios 52](#_TOC_250005)
2. [Unit Tests 52](#_TOC_250004)
3. [Integration Tests 57](#_TOC_250003)
4. [System/Acceptance Tests 60](#_TOC_250002)

[Discussion 63](#_TOC_250001)

[Reference 64](#_TOC_250000)

PROJECT PLAN

# Table of Figures

Figure 1. Evolutionary Prototype Model 11

Figure 2. Work Packages 12

Figure 3. SCREEN\_01: Library User Registration 13

Figure 4. SCREEN\_02: Book Reservation 15

Figure 5. SCREEN\_03: Self-guided Search 17

Figure 6. SCREEN\_04: Intelligent Report 18

Figure 7. PERT Chart 19

Figure 8. Project Information 19

Figure 9. Tasks Schedule 19

Figure 10. Resources Information 20

Figure 11. Gantt Chart 20

Figure 12. Resources Chart 20

CONFIGURATION PLAN

Figure 1. Version Evolution 25

Figure 2. Configuration and Change Control 26

Figure 3. Release Management 27

REQUIREMENTS SPECIFICATION

Figure 1. User Case Diagram 28

DESIGN

Figure 1. ER Diagram (conceptual model) 41

Figure 2. Logical Model 42

Figure 3. Class Diagram 43

Figure 4. Sequence Diagram (Make a Book Reservation) 44

Figure 5. Sequence Diagram (Manage Loan) 45

Figure 6. State Machine Diagram (Make a Book Reservation) 45

Figure 7. State Machine Diagram (Manage Loan) 46

Figure 8. MVC architecture Pattern 47

PAPER-BASED PROTOTYPE

Figure 1. SCREEN\_01: Library User Registration 48

Figure 2. SCREEN\_02: Book Reservation 49

Figure 3. SCREEN\_03: Self-guided Search 50

Figure 4. SCREEN\_04: Intelligent Report 50

PROJECT PLAN

# Table of Tables

Table 1: Project Deliverables Table 8

Table 2: Milestone to Measure Progress 10

Table 3: Specification of Roles 10

Table 4. SCREEN\_01 Function Points 13

Table 5. SCREEN\_02 Function Points 15

Table 6. SCREEN\_03 Function Points 17

Table 7. SCREEN\_04 Function Points 18

CONFIGURATION PLAN

Table 1: Configuration Item Responsibility 23

REQUIREMENTS SPECIFICATION

Table 1. Use Case: Log In 28

Table 2. Use Case: Verify Password 29

Table 3. Use Case: Display Login Error 30

Table 4. Use Case: Search Book 30

Table 5. Use Case: Make Reservation 31

Table 6. Use Case: Verify Book Status 31

Table 7. Use Case: Display Unavailable Status 32

Table 8. Use Case: Create Intelligent Report 32

Table 9. Use Case: Pay Loans 33

Table 10. Use Case: Check Loans 34

Table 11. Use Case: Display Deadline Warnings 34

Table 12. Use Case: Calculate Total Revenue 35

Table 13. Use Case: Self-guided Search 35

Table 14: Traceability Matrix 40

TESTING PLAN

Table 1: Test Scenarios 52

Table 2: UT\_01 52

Table 3: UT\_02 53

Table 4: UT\_03 53

Table 5: UT\_04 54

Table 6: UT\_05 54

Table 7: UT\_06 55

Table 8: UT\_07 55

Table 9: UT\_08 56

Table 10: UT\_09 56

Table 11: IT\_01 57

Table 12: IT\_02 58

Table 13: IT\_03 58

Table 14: IT\_04 59

Table 15: IT\_05 59

Table 16: ST\_01 60

Table 17: ST\_02 61

Table 18: ST\_03 61

Table 19: ST\_04 62

# Abstract

Bronco Recreation complex wants to develop a Bronco Recreation complex management system (BRCM) for the computerization of the customer visits, activities, billing and their services to ensure effective control, improved efficiency and reduce their operational costs. This system will replace their current maintenance of all the above-mentioned items by hand.

The purpose of this project is to develop a computerized system that can be used by customers to make reservations and to look at their previous history of their visits to the complex. The complex staff can use the system to manage recreational activities, users, order booking, price management including discounts, and to generate business intelligence.

Project was developed using Java in IntelliJ IDEA. Two components were created. One for desktop use and one for web use. We used Tomcat to develop a user-friendly and faster interface for the website component. Java FX was used to develop Desktop applications. For the central database, we used PostgreSQL and Hibernate for the maintenance of the information.

We had enough experience within our team in using the above-mentioned tools/software. We were able to develop a system that runs effectively and at the same time, interactive to the user.

Keywords: BRCM, Management system, booking

# Project Plan

**1.** **Introduction**  
**1.1** **System-As-Is**

In the current recreation complex system, all customers, visits and recreational activities are maintained in records by hand. For every single visit, the customers/users are provided with manual receipt in which the header information such as date, time, and customer name and the recreational activities, time, quantities, number of individuals, discounts, and total prices. Using the copies of these receipts, the transaction information is entered into spreadsheets ona computer. To no surprise, this system is complained to be

* Slowing down customers and recreational activity search process when the visits are to be created.
* Redundant and mixed info of professors and students.
* Consolidated reports of visits per customer
* Inaccurate pricing of activities

So, CPP has decided to develop an automated Bronco Recreation Complex Management System that would improve their operational control and reduce the operational cost.

**1.2** **System-To-be**

The new system is to be developed on a software-based solution. The new system should consist of two components, Desktop and web applications. Desktop application with GUI should provide online services such as customer registration, activity booking, visit recording and management, and business intelligence reports.

New Automated BRCM system is expected to meet the below stated requirements:

* Elimination of the redundancy in customer registration process
* Incorporation of different discount schemes for students and professors that also shown the percentage of the visits
* Access to price history timeline of the recreational activities
* Provide consolidated reports of visits per customer
* Also, generate business intelligence report to understand the revenue information as per customer and period.

These requirements would address all the complaints. In addition to the above requirements, it should also show the status which should show the live status and also the origin of the visit generation, whether it’s “counter”, “online-pending”, or “online-complete”.

### Project Deliverables

Table 1: Project Deliverables Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Deliverable Table** | | | | | |
|  | | | | | |
| **Project Name:** | Bronco Recreation Complex Management(BRCM) System | | **Project ID:** | Version 1.0 | |
| **Project Manager:** | Siva Charan Mallena | | **Status:** | 80% Complete | |
|  | | | | | |
| **Sr No** | **Deliverable Name** | **Description** | | **Owner** | **Status** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Prepare project plan | Design process Model, method and techniques. Provide guidelines, procedures and standards. Resources distributions and effort schedule. Demonstrate the procedures to be followed | Siva Charn Mallena | Complete |
| 2 | Prepare configuration Plan | Prepare software configuration management, identify configuration items. Determine environment and infrastructure, and provide versioning and other documents etc. | Priyatham Sai Chand Bazaru, Madhurima Budugu | Complete |
| 3 | Requirements Specification | Clear High-Level Goals, Use Case diagram and specifications. Provide Functional requirement and non-functional requirement. Traceability Matrix is also included. | Siva Charan Mallena, Madhurima Budugu | Complete |
| 4 | Design | ER diagram, Logical Model, Class Diagram, and other diagrams etc. | Rahul Nagarajan, Malem Thockchom | Complete |
| 5 | Prepare testing Plan | Prepare unit tests, integration tests, and system.Acceptance tests, and provide Quality Assurance plan | Priyatham Sai Chand Bazaru, Malem Thockchom | Complete |
| 6 | Design prototype | Design paper-based prototype(Mock ups)and provide technique description of the system | Malem Thockchom | Complete |
| 7 | Discussion | Analysis of the results obtained, and lessons learned | Siva Charan Mallena, Priyatham Sai Chand Bazaru, Rahul Nagarajan, Malem Thockchom, Madhurima Budugu | Complete |

## Process Model

### Activities to be Undertaken

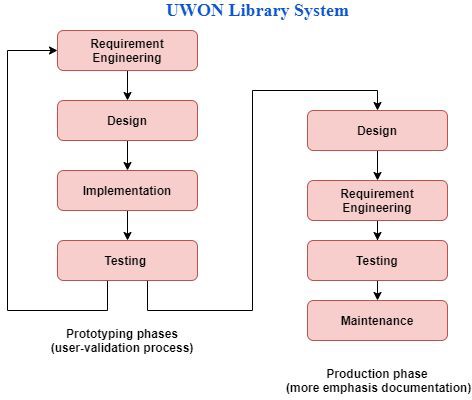


Figure 1. Evolutionary Prototype Model

The newly developed BRCM system will be using evolutionary prototype shown above in Figure 1.1. The reason for employing this third additional prototype is because it would enable us to cover all the requirements stated above. Before working on the production of the system, the prototypes are developed and tested extensively. Any identified bugs/errors are fixed prior to working on the production phase.

### Milestone to Measure Progress

Table 2: Milestone to Measure Progress

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Duration** | **Constraints** | **Milestone** |
| **Requirements** | 6 |  |  |
| **Design** | 10 | Requirement finished | M1 |
| **Test plan** | 5 | Design finished | M2 |
| **Coding** | 14 | Design finished | M3 |
| **Test** | 10 | Coding finished, Test plan finished | M4 |

After requirements are finished, we will conduct project design and project design is also a constraint of the Testing plan so we set the Test plan as milestone M1. Project Design is also a constraint of Coding, and we will conduct Coding when Project Design finishes. We set the Coding as milestone M2. Coding and Test plan are constraints of Test and we set Test as milestone M3. All milestones make sure the project is on a good track and helps to keep it on time and on budget.

## Organization of the Project

### Information, services, resources, and facilities to be provided

### All copies of manually printed receipts for the customer visits, spreadsheets with all the data entered, and prior financial reports should be made available for the development and transition to the BRCM system. CPP should also allow access to all the customer information including students and professors. The new system would follow the same process as the old system, but the entire process is automated/computerized. It also provides consolidated reports of the visits of a customer/in a period and business Intelligence revenue reports which were not part of the older system.

### Specification of the roles

Table 3: Specification of Roles

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Responsibility** |
| Siva Charan Mallena | Project manager | Design process Model, method and techniques. Provide guidelines, procedures and standards. Resources distributions and effort schedule.  Demonstrate the procedures to be followed |
| Rahul Nagarajan, Malem Thockchom | Designer | Design ER-diagram, Logical Model, Class Diagram, and other |

|  |  |  |
| --- | --- | --- |
|  |  | diagrams, etc.  Design paper-based prototype and provide technique description of the system |
| Madhurima Budugu, Siva Charan Manager | Analyst | Clear High-Level Goals, Use Case diagram, and specifications. Provide Functional requirements and non-functional requirements.  Traceability Matrix is also included |
| Rahul Nagarajan, Priyatham Sai Chand Bazaru, | Programmer | Coding and debugging. Designing and testing computer structures. Troubleshooting system errors. Writing computer instructions.  Managing database systems. |
| Priyatham Sai Chand Bazaru, Malem Thockcham | Tester | Prepare unit tests, integration tests, and system. Acceptance tests. Reviewing software requirements and preparing **test** scenarios. Executing tests on software usability. Analyzing **test** results on database impacts, errors or bugs, and usability. |

## Methods and techniques

ER-diagram, Traceability Matrix, and Use diagram were designed on draw.io. in the phase of requirements engineering. For the completeness and clarity UML, logical diagrams and physical diagrams were created using draw.io and SQL Power Architect. Mock ups are created using Wire frames on Mock flow. Coding and testing were done using Java language in IntelliJ IDEA and PostgreSQL database was used for the storage of all data. User Interface design was done using JavaFX platform. Automation testing was performed using Selenium and Junit.

## Standards, guidelines, procedures

One of the teams constantly reviews requirement specifications, and makes sure to meet all the requirements. One of the teams will periodically inspect project progress and project quality before moving to subsequent phases correctly, after that teams will write verification processes. Since **Initial Level** is being currently implemented(CMMI), teams currently implement Requirements managements, Project planning, Project monitor and control and also working on Process and product quality assurance , Measurement and analysis and Configuration management.

**Quality- Priyatham**

**Requirements- siva, madhu**

**No CMMI**

## Work packages

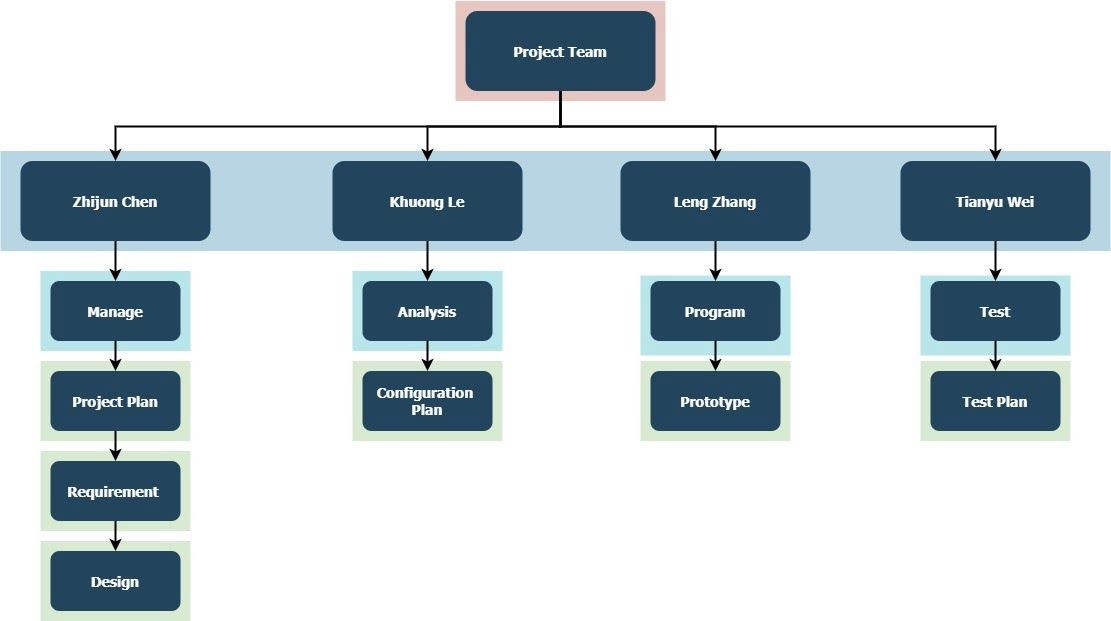


Figure 2. Work Packages

Project Team

Siva Charan Mallena –Manage, Project plan, Requirements,

Rahul Nagarajan-Design,Program

Malem Thockchom-Design, Testing

Madhurima Budugu-Analysis, Configuration Plan

Priyatham Sai Chand Bazaru-Configuration Plan, Testing

## Resources

### Hardware

The BRCM system uses Java framework Springboot for backend, Javascript,JavaFX for frontend, and PostgreSQL for database. The BRCM system can be installed and accessed on any of the operating systems (Windows, Mac OSX, Linux) that can support Javascript, Java, and PostgreSQL.

Expected minimal production environments: Quadcore (64-bit CPU and OS), 4 GB RAM, and 500 GB HDD. Additional hardware such as keyboard and mouse are required.

### Personnel

One project manager, two designer, One analyst, Three programmer, Two tester

## Effort and schedule

### Effort

SCREEN\_01

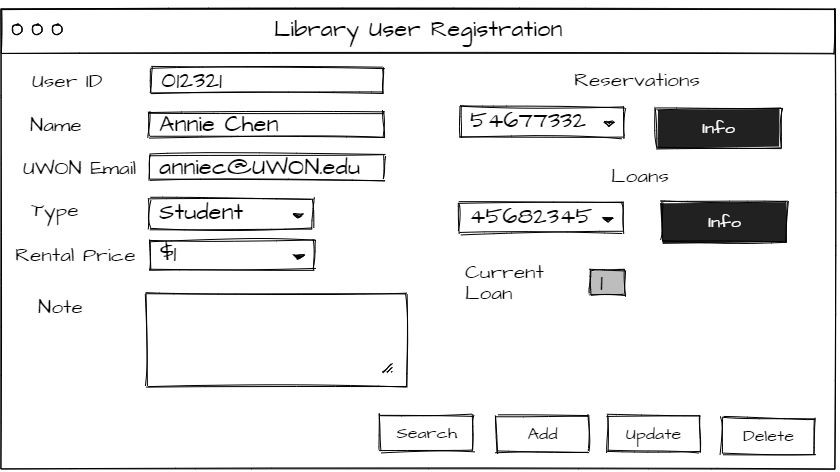


Figure 3. SCREEN\_01: Library User Registration Table 4. SCREEN\_01 Function Points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Elementary Process | Type | DET | RET/FTR | Complex | FP |
| Files | | | | | |
| User | ILF | 5 | 2 | Low | 7 |
| Reservation | EIF | 4 | 1 | Low | 5 |
| Loan | EIF | 6 | 1 | Low | 5 |
| Functions | | | | | |
| Buttons | | | | | |
| Search | EO | 9 | 3 | Average | 5 |
| Add | EI | 9 | 1 | Low | 3 |
| Update | EI | 9 | 1 | Low | 3 |
| Delete | EI | 9 | 1 | Low | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Reservation Info | EO | 3 | 1 | Low | 4 |
| Loan Info | EO | 3 | 1 | Low | 4 |
| Dropdown List | | | | | |
| Dropdown Type | EQ | 3 | 1 | Low | 3 |
| Dropdown Price | EQ | 3 | 1 | Low | 3 |
| Dropdown Reservation | EQ | 3 | 1 | Low | 3 |
| Dropdown Loan | EQ | 3 | 1 | Low | 3 |
| Calculated Box | | | | | |
| Current Loan | EO | 1 | 1 | Low | 4 |
| Total | | | | | 65 |

SCREEN\_02

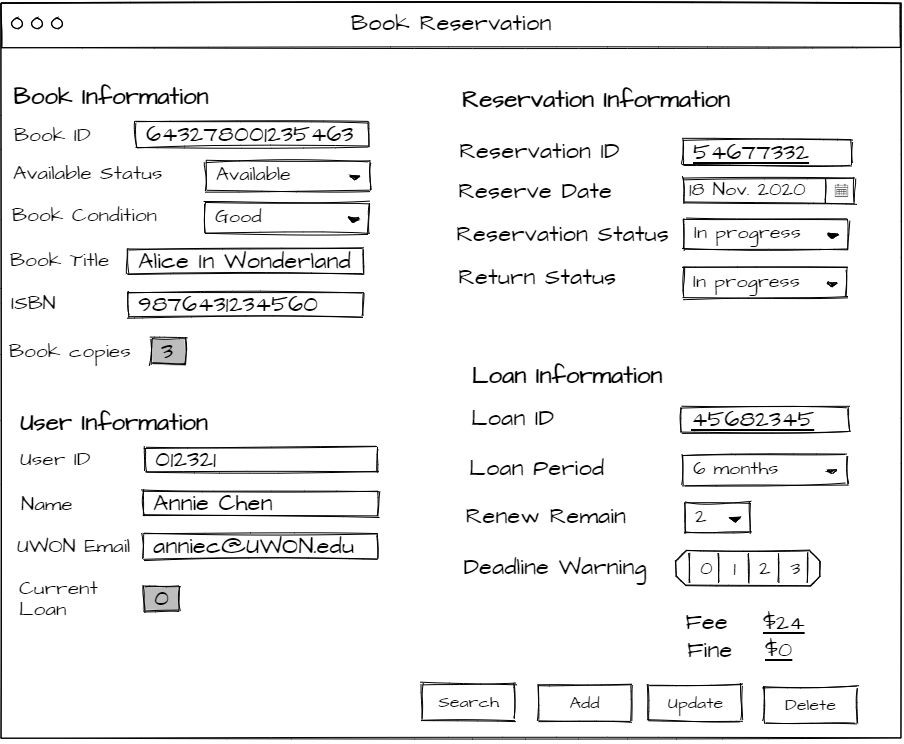


Figure 4. SCREEN\_02: Book Reservation Table 5. SCREEN\_02 Function Points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Elementary Process | Type | DET | RET/FTR | Complex | FP |
| Files | | | | | |
| Book Info | ELF | 6 | 2 | Low | 5 |
| User Info | EIF | 4 | 2 | Low | 5 |
| Reservation Info | ILF | 4 | 1 | Low | 7 |
| Loan Info | EIF | 6 | 2 | Low | 5 |
| Functions | | | | | |
| Buttons | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Search | EO | 20 | 4 | High | 7 |
| Add | EI | 20 | 4 | High | 6 |
| Update | EI | 20 | 4 | High | 6 |
| Delete | EI | 20 | 4 | High | 6 |
| Dropdown List | | | | | |
| Dropdown Available Status | EQ | 3 | 1 | Low | 3 |
| Dropdown Book Condition | EQ | 3 | 1 | Low | 3 |
| Dropdown Reserve Date | EQ | 3 | 1 | Low | 3 |
| Dropdown Reservation Status | EQ | 3 | 1 | Low | 3 |
| Dropdown Return Status | EQ | 3 | 1 | Low | 3 |
| Dropdown Loan Period | EQ | 3 | 1 | Low | 3 |
| Dropdown Renew Remain | EQ | 3 | 1 | Low | 3 |
| Calculated Box | | | | | |
| Book Copies | EO | 1 | 1 | Low | 4 |
| Current Loan | EO | 1 | 1 | Low | 4 |
| Deadline Warning | EO | 1 | 1 | Low | 4 |
| Fee | EO | 1 | 1 | Low | 4 |
| Fine | EO | 1 | 1 | Low | 4 |
| Total | | | | | 98 |

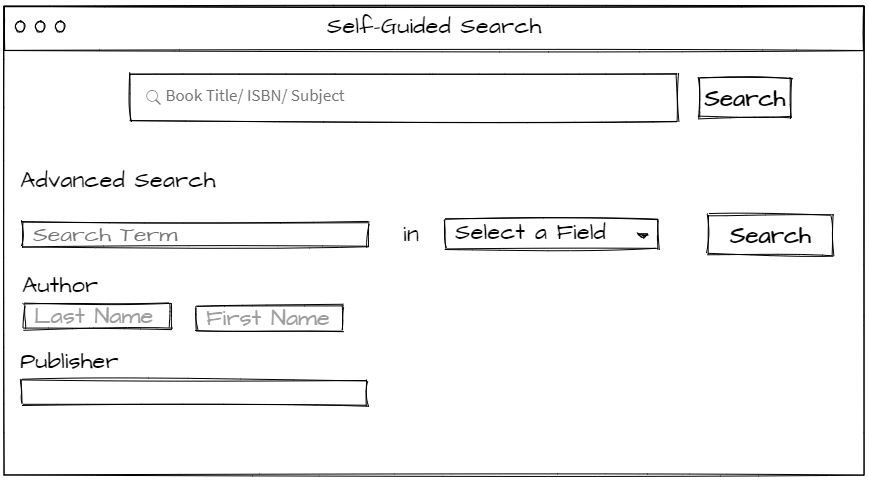
SCREEN\_03

Figure 5. SCREEN\_03: Self-guided Search Table 6. SCREEN\_03 Function Points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Elementary Process | Type | DET | RET/FTR | Complex | FP |
| Files | | | | | |
| Book Info | EIF | 6 | 2 | Low | 5 |
| Functions | | | | | |
| Buttons | | | | | |
| General Search | EQ | 1 | 2 | Low | 3 |
| Advanced Search | EQ | 5 | 2 | Low | 3 |
| Dropdown List | | | | | |
| Dropdown Field | EQ | 3 | 1 | Low | 3 |
| Total | | | | | 14 |

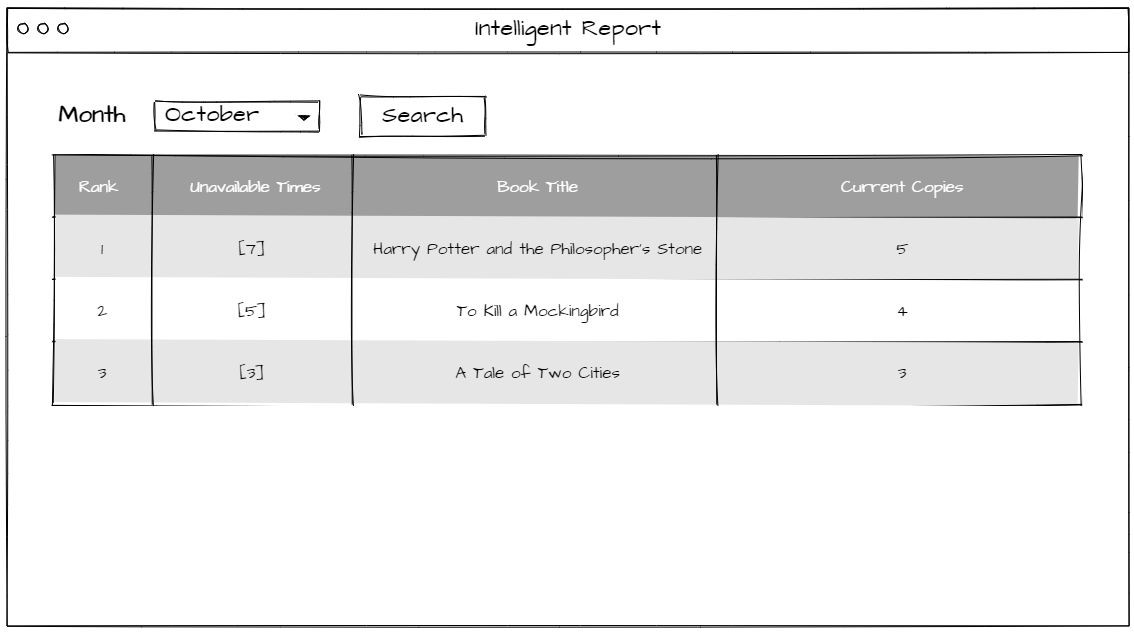
SCREEN\_04

Figure 6. SCREEN\_04: Intelligent Report Table 7. SCREEN\_04 Function Points

The total effort (measure in terms of number of function points) estimate for the project:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Elementary Process | Type | DET | RET/FTR | Complex | FP |
| Files | | | | | |
| Report | EIF | 3 | 2 | Low | 5 |
| Functions | | | | | |
| Buttons | | | | | |
| Search | EQ | 3 | 2 | Low | 3 |
| Dropdown List | | | | | |
| Dropdown Month | EQ | 3 | 1 | Low | 3 |
| Display | | | | | |
| Book List | EO | 4 | 2 | Low | 4 |
| Total | | | | | 15 |

65+98+14+15=192

### Schedule

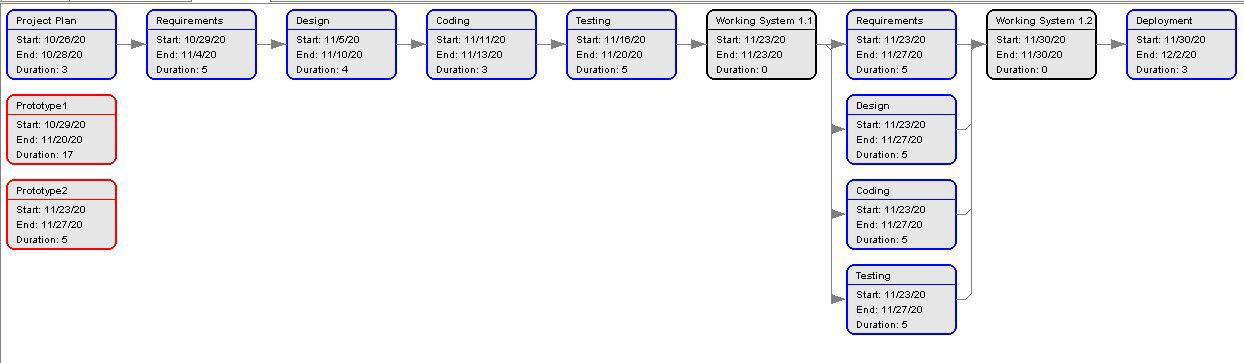


Figure 7. PERT Chart

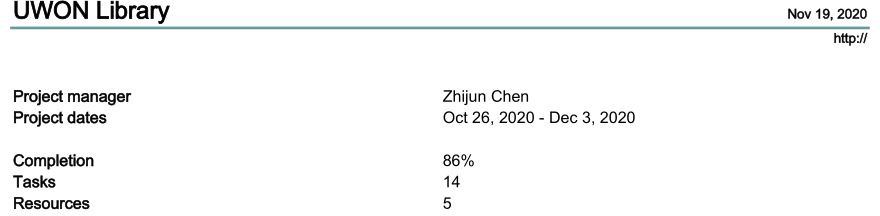


Figure 8. Project Information

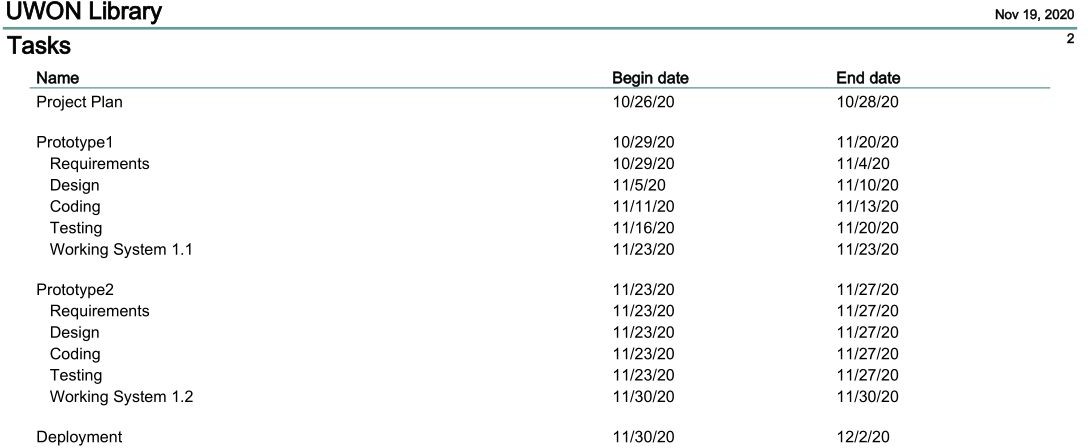


Figure 9. Tasks Schedule



Figure 10. Resources Information

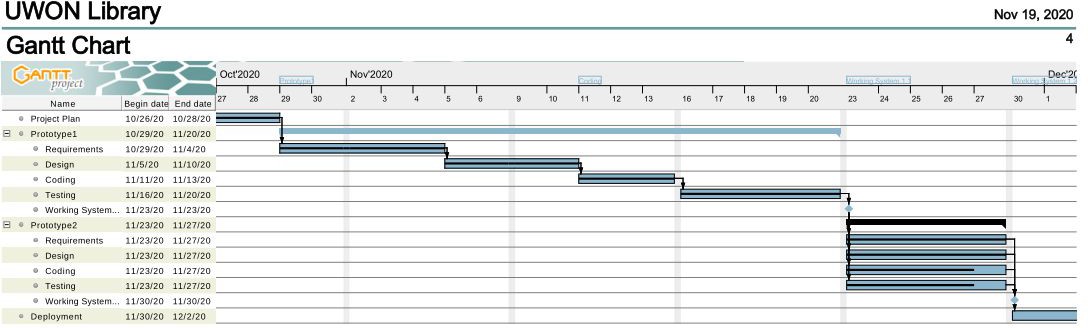


Figure 11. Gantt Chart

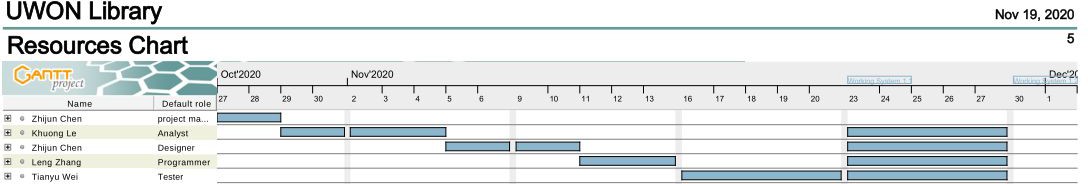


Figure 12. Resources Chart

## Delivery

### Login procedure:

* + - In order to use UWON library system, user must be a member of UWON and have a valid library account.
    - Must be a member of CPP must have valid account

### Appointment procedure:

Customer can be a student or professor

For Student/ Professor:

If student want to reserve an activity, they need to login to CPP account

Open desktop or web application

They can go to book appointment section, look at all available appointments,

Select an appointment and click on book appointment, and confirm book option

Appointment booked mssg will be shown

9.3 operational procedure

For employee:

Employee can add/update the student or proff by adding the details given by them,

Employee can delete after they no longer exist

* + - If user wants to take the book out of the library, s/he must go to the help desk and ask a librarian to make a reservation for that book. A user can only reserve/borrow books if they are available on the shelves.
    - While making the reservation, user must provide a valid library account and agree to pay for the loan for that reservation.
    - A loan will be U$1, U$2, U$3 per week for staff, student, professor respectively.
    - User cannot have more than 3 loans at the same time. In order words, a user may not borrow more than three books at the same time.
    - The maximum loan period of a loan is 6 months.
    - User can renew a loan up to 2 times. While approaching to deadline, UWON library system will

send out deadline warning through UWON email within 3 days.

* + - User fails to pay for the loan will be charged fine U$1 per day.
    - User will receive a message about the daily fines and fine being charged after the due date.
    - User can check their reservation and loan information in the library system. For librarian:
    - A book can be borrowed by a user only if at least one copy of the book is available.
    - After the reservation, the available status of that book with a unique book id will be marked unavailable and total copies of that book with the same ISBN will decrease 1.
    - Librarian should check the condition of the book (good/damaged) and record it before lending it to the user.
    - After the due date of a reservation, if the user returns the book and pays the fee, librarian can change the reservation status to “finish” and return status to “return”. If the user doesn’t return the book on time, change the reservation status to “past due” and return status to “no return”.

### Search procedure:

* + - User can search the book title/ISBN/subject in the search bar for general search.
    - User can also use advanced search with a specific term in a selected field, author’s name, and publisher.

### Intelligent Report procedure:

* + - Librarians can get an intelligent report about the popular book in the months.
    - If the system shows unavailability of a book 3 times in a month, this information should be included in an intelligent report for future acquisition of extra copies.

# Configuration Plan

## Introduction

### Conventions:

* + - Each developer works on own branch.
    - Pull request has to be created for merging new features or bug fixing to the dev branch and request at least 2 reviewers to approve merging.
    - Updates from the dev branch have to push to the beta server for testing and can be pushed to the production server after passing all tests in the beta server.

### Terms:

REQ - Requirements BLD - Builds

MN - Manuals

DEV - Development PROD - Production

DEVS - Development Server BETAS- Beta Server

PRODS- Production Server

### Abbreviations Identify Configuration Manager (s):

ZC - Student1 TW - Student2 KL - Student3 LZ - Student4

## Software Configuration Management

List of Process Activities

* Have a group meeting 3 times a week (Monday, Wednesday, Friday) to discuss what we are going to do and update on the current process.
* Identify High-Level Goals
* Identify Primary and Secondary Actors
* Identify Use Case Diagrams and Specifications
* Identify Requirements
* Identify Configuration Items
* Identify Responsibility for each Configuration Item
* Create Repository
* Commit
* Generate Versions and Baselines

## Identify Configuration Items

Configuration items that will be produced throughout the project life cycle

* Project Plan
* Configuration Plan
* Requirements Specification
* Design Specification
* Paper-based Prototype
* Test Plan
* Quality Plan

## Identify Responsible for each Configuration Item

Table 1: Configuration Item Responsibility

|  |  |
| --- | --- |
| Configuration Item | Responsible Person |
| Project Plan | TW |
| Configuration Plan | KL, LZ |
| Requirements Specification | ZC |
| Design Specification | ZC, KL,TW |
| Paper-based Prototype | ZC |
| Test Plan | LZ |
| Quality Plan | ZC, TW, KL, LZ |

## Tools

Version control tools that will be used.

* GitHub: Control versions of the apps
* Jira: Running sprints through this software. Managing planned tasks, in process tasks and testing tasks and finished tasks.
* IDE: Eclipse

## Environment and Infrastructure

* There are going to be 3 different servers: dev, beta, production: dev server is for developing, beta server is for internal testing, production server is for the releasing
* All the manual docs are stored on the dev server.

## Configuration Policy

Nomenclature for configuration items such as <project> - <type-artifact> - <name>, where

<project> is the name of the project, <type-artifact> is the configuration item, <name> is the name of the document

### Project Plan:

UWONLS-ProjectPlan-ProcessModel UWONLS-ProjectPlan-Organization

UWONLS-ProjectPlan-MethodsAndTechniques UWONLS-ProjectPlan-Standard

UWONLS-ProjectPlan-WorkPackages UWONLS-ProjectPlan-Resources UWONLS-ProjectPlan-Delivery

### Configuration Plan

UWONLS-ConfigurationPlan-Conventions UWONLS-ConfigurationPlan-Terms

UWONLS-ConfigurationPlan-SoftwareConfigurationManagement UWONLS-ConfigurationPlan-ConfigurationItems

UWONLS-ConfigurationPlan-Tool

UWONLS-ConfigurationPlan-EnvironmentAndInfrastructure UWONLS-ConfigurationPlan-ConfigurationAndChangeControl UWONLS-ConfigurationPlan-SystemBuilding

UWONLS-ConfigurationPlan-ReleaseManagement UWONLS-ConfigurationPlan-ContingencyPlan

### Requirements Specification

UWONLS-RequirementsSpecification-High-Level Goals UWONLS-RequirementsSpecification-PrimaryAndScondaryActors UWONLS-RequirementsSpecification-UseCaseDiagram UWONLS-RequirementsSpecification-UseCaseSpecification UWONLS-RequirementsSpecification-Assumptions

UWONLS-RequirementsSpecification-DomainProperties UWONLS-RequirementsSpecification-FunctionalRequirements UWONLS-RequirementsSpecification-NonFunctionalRequirements UWONLS-RequirementsSpecification-TraceabilityMatrix

### Design Specification

UWONLS-DesignSpecification-ERDiagram UWONLS-DesignSpecification-LogicalModel

UWONLS-DesignSpecification-SequenceDiagram UWONLS-DesignSpecification-StateMachineDiagram

UWONLS-DesignSpecification-SystemArchitectureViewAndStylePattern

### Paper-based Prototype

UWONLS-PaperBasedPrototype-TechnicalDescription

### Test Plan

UWONLS-TestPlan-Scenarios UWONLS-TestPlan-UnitTests UWONLS-TestPlan-IntegrationTests

UWONLS-TestPlan-SystemAcceptanceTests

## Document Version

Definition of a version numbering scheme for configuration items and baselines UWONLS.major.minor[.fix][-configuration]

UWONLS - University of Wonderland Library System

.major - Main Version with 5 or more new features

.minor - Minor Version with less than 5 new features or updating features [.fix] - bug fixing

[-configuration] - configuration label

## Evolution of the Version Number

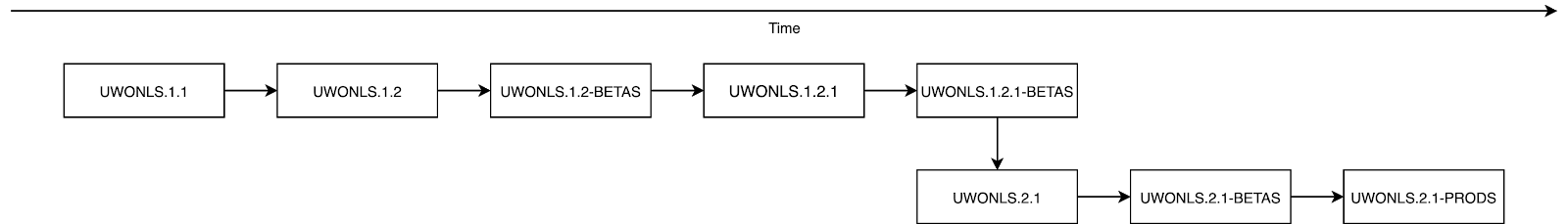
[](https://app.diagrams.net/?page-id=UHUzkEogl5E5jfNh7rj6&scale=auto&G1MQKIDWOJbCttQE_Pj448IadTg-6lnwXZ)

Figure 1. Version Evolution

UWONLS.1.1

Implemented User login by university email and password UWONLS.1.2

Implemented update password feature UWONLS.1.2-BETAS

Push version UWONLS.1.2 to beta server and test.

Bug\_01: After the user uses the update password feature, the user is not able to login with the new password.

UWONLS.1.2.1

Fixed Bug\_01 UWONLS.1.2.1-BETAS

Push version UWONLS.1.2.1 to beta server and test.

UWONLS.2.1

Implemented the self-guided tool which the logined user is able to search books by name, subject, ISBN, publisher, or author.

UWONLS.2.1-BETAS

Push version UWONLS.2.1 to beta server and test.

UWONLS.2.1-PRODS

Push version UWONLS.2.1-BETAS to production server.

## Configuration and Change Control

We are going to use Trello to update the progress of the project, ZC is the one who manages it.. There are five different columns to keep track of work.

First column is “Investigate”, the owner will create tasks in the “Problems” then our team will decide if we will reject, defer or approve it. If we feel that it’s not feasible, then we reject it. If we think it’s doable but need more confirmation then we request the owner to give me information. If all the information is given and it’s doable, we will set up a plan to finish that task.

Second column is “Schedule”, planning up all subtasks for needed architecture, system, codes, and time for the main task given by the owner.

Third column is “Implementation”, showing what tasks KL and LZ are currently working on. It will show the progress of each task and when will that one finish.

Fourth column is “Testing” to keep track of what tasks have been done by KL, LZ and those tasks are ready to be tested by TW.

Fifth column is “Done” to keep track of all the done tasks, and report to the owner if we finish all the big main task.

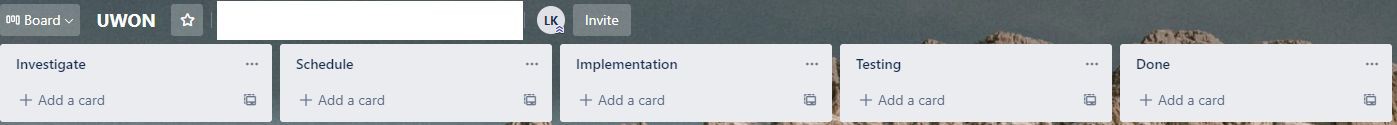


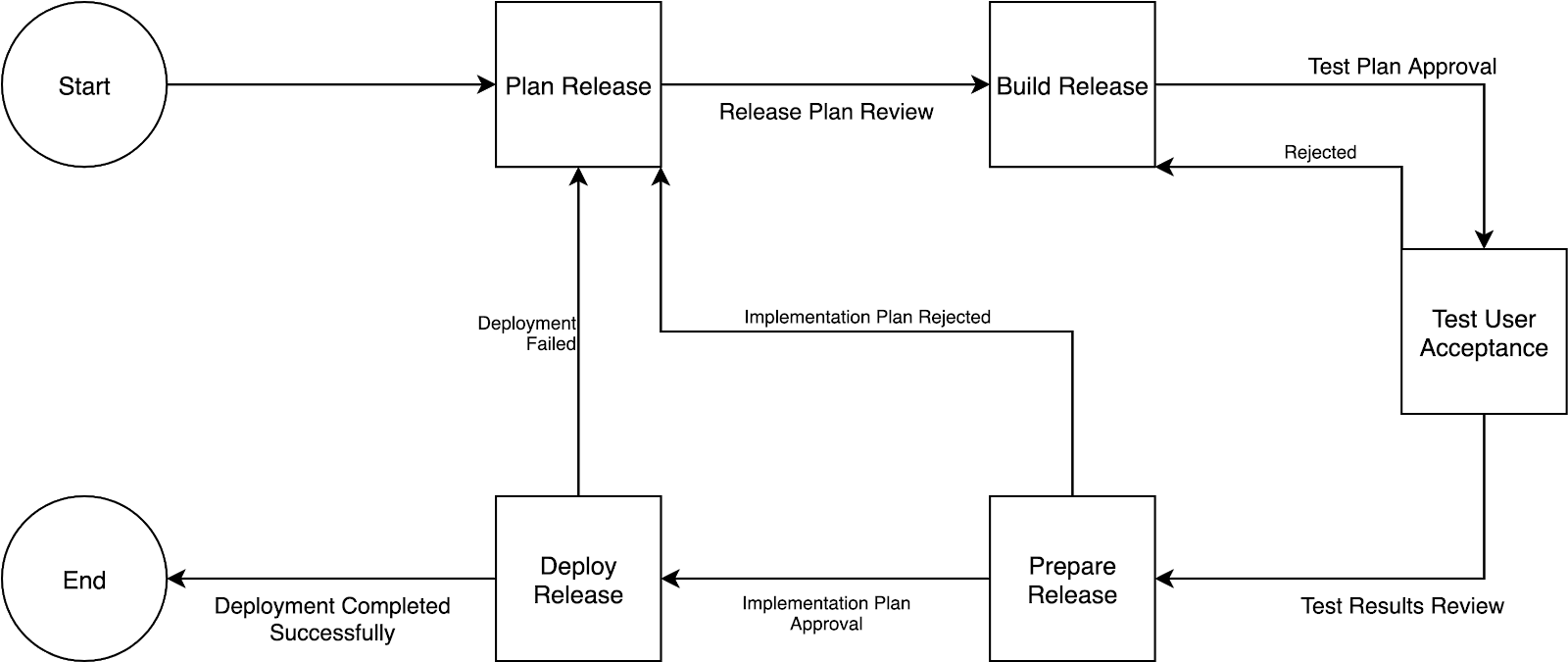
Figure 2. Configuration and Change Control

## System Building

1. Create repository
2. Configure Java development environment
3. Configure MySQL environment and manage data to match with UWON’s database
4. Follow use cases and implement those functionalities.
5. Follow test cases to validate and verify requirements
6. Compile the project and then write documentations on how to use it.

## Release Management

Strategy to system delivery, client system version management

[](https://app.diagrams.net/?page-id=cHRs2H6tyRlXQ9hHgzTW&scale=auto&G19sGoszaLZ8QFGPyOLbMuG1uwMH5Xetqi)

## Contingency plan

Figure 3. Release Management

Procedures to be executed in case of data loss or compromise

* Separate the development database and the production database
* Backup the production database every two weeks
* Before updating new version to production server, backup the production database
* Try to make clear and nice description of comments and pull requests, so developers can easy to find the cause of data loss or compromise
* Comments all changes and push to own branch daily
* Make logs for every database updates and save logs separate from the database
* When data loss or compromise, developers can try to rollback to the latest and working backup and update the backup by logs.

# Requirements Specification

## High-Level Goals (Unique Identifiers)

G1: Deliver a sufficient way to use the recreation center for professors and students

G2: Provide efficient methods to manage BRCM activity.

## Primary and Secondary Actors

Primary Actors: BRCM user (BRCM staff)

Secondary Actors: Student, Professor

## Use Case diagram (s)

## 

Figure 1. User Case Diagram

## Use Case Specification (s)

Table 1. Use Case: Add a new customer

|  |
| --- |
| **Use Case:** Log In |
| **Primary Actor:** BRCM User |
| **Secondary Actor:** None |

|  |
| --- |
| **Precondition:**  1. Actor has access to the BRCM Application |
| **Postcondition:**  1. Actor has successfully added customers to the customer database. |
| **Main Success Scenario:**   1. Actor selects the CUSTOMER option from the navigation pane   2a. Actor selects the appropriate customer option (STUDENT/PROFESSOR)   1. Actor clicks the button ADD NEW STUDENT/ADD NEW PROFESSOR 2. App directs the Actor to the ADD/UPDATE customer tab 3. Actor informs the customer information in the required input fields 4. Actor selects the option ‘SAVE CHANGES’ |
| **Alternate Course:**  2b – Actor selects the ADD/UPDATE CUSTOMER tab. Resumes at step 4. |
| **Exception Course:**  4. The customer details are not saved to the database if the user doesn’t click on the sa |

Table 2. Use Case: Verify Password

|  |
| --- |
| **Use Case:** Verify Password |
| **Primary Actor:** Staff |
| **Secondary Actor:** Library |
| **Precondition:** Credentials are submitted by the staff |
| **Postcondition:** Authentication is completed |
| **Main Success Scenario:**   1. Staff clicks the button LOGIN 2. Library system encrypts the informed credentials and compared them to the ones stored in the DB |

|  |
| --- |
| 3. A Boolean message with True is produced if the credentials match |
| **Alternate Course:**  3a – A Boolean message with False is produced if the credentials do not match |
| **Exception Course:** |

Table 3. Use Case: Display Login Error

|  |
| --- |
| **Use Case:** Display Login Error |
| **Primary Actor:** Staff |
| **Secondary Actor:** None |
| **Precondition:** Credentials were verified |
| **Postcondition:** A login error message is displayed |
| **Main Success Scenario:**   1. A Boolean message with False is produced by UC Verify Password 2. A login error message is generated 3. A login error message is displayed |
| **Alternate Course:** |
| **Exception Course:** |

Table 4. Use Case: Search Book

|  |
| --- |
| **Use Case:** Search Book |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor logged into the system |
| **Postcondition:** Redirect to the page that returns the search result. |
| **Main Success Scenario:** |

|  |
| --- |
| 1. Actor clicks on the search box 2. Actor types the title of the book or article 3. Actor presses on the search button or “ENTER” 4. Library system redirect to the search result page |
| **Alternate Course:**  3a – A Boolean message with False is produced if the book or article is not available |
| **Exception Course:** |

Table 5. Use Case: Make Reservation

|  |
| --- |
| **Use Case:** Make Reservation |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor is redirected to the search result pages. |
| **Postcondition:** Actor successfully reserves one or many books |
| **Main Success Scenario:**   1. Actor chooses the book they want 2. Actor clicks on button MAKE RESERVATION 3. Include UC verify book status 4. A book status success message is displayed 5. Actor successfully reserves one or many books |
| **Alternate Course:** |
| **Exception Course:**  5a – Extend UC Display Login Error when book is not available. Resumes at step 1. |

Table 6. Use Case: Verify Book Status

|  |
| --- |
| **Use Case:** Verify Book Status |
| **Primary Actor:** User |
| **Secondary Actor:** Library system |

|  |
| --- |
| **Precondition:** Book reservation has been made by user |
| **Postcondition:** Book status verification is completed |
| **Main Success Scenario:**   1. - Staff clicks the button “MAKE RESERVATION” 2. – Application encrypts the informed credentials and compared them to the ones stored in the DB 3. – A Boolean message with True is produced if the credentials match |
| **Alternate Course:**  3a – A Boolean message with False is the book is not available |
| **Exception Course:** |

Table 7. Use Case: Display Unavailable Status

|  |
| --- |
| **Use Case:** Display Unavailable Status |
| **Primary Actor:** User |
| **Secondary Actor:** None |
| **Precondition:** |
| **Postcondition:** |
| **Main Success Scenario:**   1. A Boolean message with False is produced by Library system when book is not available 2. An unavailable error message is generated 3. An unavailable error message is displayed |
| **Alternate Course:** |
| **Exception Course:** |

Table 8. Use Case: Create Intelligent Report

|  |
| --- |
| **Use Case:** Create Intelligent Report |

|  |
| --- |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor has made some previous reservations before. |
| **Postcondition:** Display the reservation report |
| **Main Success Scenario:**   1. Go to actor’s reservations. 2. Actor click on button CREATE REPORT 3. Library system starts to gather data 4. Library displays the report |
| **Alternate Course:** |
| **Exception Course:** |

Table 9. Use Case: Pay Loans

|  |
| --- |
| **Use Case:** Pay Loans |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor logged into the system and has made some previous reservations before |
| **Postcondition:** User finish paying their loans |
| **Main Success Scenario:**   1. Actor clicks on MANAGE LOANS in the menu 2. Library system redirected to Manage Loans screen 3. Actor clicks on PAY LOANS 4. Include UC Calculate total revenue 5. Actor confirms that they successfully pay for loans |
| **Alternate Course:**  3a. Actor is informed of the total revenue that they need to pay. Resume at step 4. |
| **Exception Course:** |

Table 10. Use Case: Check Loans

|  |
| --- |
| **Use Case:** Check Loans |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor logged into the system and has made some previous reservations before |
| **Postcondition:** Actor knows about their current reservations’ deadline |
| **Main Success Scenario:**   1. Actor clicks on CHECK LOANS in the menu 2. Library system redirected to Check Loans screen 3. Actor clicks on CHECK LOANS 4. Include UC Display deadline warnings |
| **Alternate Course:** |
| **Exception Course:** |

Table 11. Use Case: Display Deadline Warnings

|  |
| --- |
| **Use Case:** Display Deadline Warnings |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor clicks on CHECK LOANS in MANAGE LOANS section |
| **Postcondition:** UI display to show all the deadline for reservations |
| **Main Success Scenario:**   1. Library system starts to collect data inside the DB 2. Waiting screen is initiated while waiting 3. Display the deadlines to actor |
| **Alternate Course:** |
| **Exception Course:** |

Table 12. Use Case: Calculate Total Revenue

|  |
| --- |
| **Use Case:** Calculate Total Revenue |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor is navigated to PAY LOANS screen |
| **Postcondition:** Actor knows about their loan that new need to pay |
| **Main Success Scenario:**   1. Library system starts to collect data inside the DB 2. Library system apply formula to calculate the loan base on each user and deadline 3. Display the amount of loan user has to pay |
| **Alternate Course:** |
| **Exception Course:** |

Table 13. Use Case: Self-guided Search

|  |
| --- |
| **Use Case:** Self-guided Search |
| **Primary Actor:** User |
| **Secondary Actor:** Library |
| **Precondition:** Actor logged into the system and press on BIBLIOGRAPHIC RESEARCH |
| **Postcondition:** The systems show the research to actor |
| **Main Success Scenario:**   1. Actor inputs values on the bibliographic research 2. Library system gathered information and searched on DB 3. Library system returns the data to actor |
| **Alternate Course:** |
| **Exception Course:** |

## Assumptions

### Credentials

* + - User should be a member of the University of Wonderland (UWON).
    - User should have an active university email to register to the UWON library system.

### Security

* + - No user can take out the book without reservation.
    - No user can make a reservation without a loan.

## Domain Properties

* A book with an unique Book ID can not be borrowed by different users at the same time.
* A book can not be available and borrowed at the same time.
* A reservation can not exist with a loan.

## Functional Requirements

People Registration:

REQ\_1 - User Registration

Rationale: A USER is identified by User ID, and has the attributes Name, Type, Price, and Email.

REQ\_2 - Library Staff Registration

Rationale: A LIBRARY STAFF is identified by Staff ID and has an attributes Name.

Library Registration:

REQ\_3 - Book Registration

Rationale: A BOOK is identified by Book ID, and has the attributes Book Title, ISBN, Condition, and the calculated attributes Number of Copies, and Available Status.

REQ\_4 - Reservation Registration

Rationale: A RESERVATION is identified by Reservation ID, and has the attributes Reserve Date, Reservation Status, and Return Status.

Fund Registration:

REQ\_5 - Loan Registration

Rationale: A LOAN is identified by Loan ID, and has an attribute Loan Period, Number of Renew and the calculated attributes Fee, Fine, and Deadline Warming.

People Action Relationship:

REQ\_UL\_1 - Relationship between User and Loan

Rationale: Each USER can make at most 3 LOANS, and each LOAN must be made by only one USER.

REQ\_UR\_1 - Relationship between User and Reservation

Rationale: Each USER can have many RESERVATIONS, and each RESERVATION must belong to only one USER.

REQ\_UB\_1 - Relationship between User and Book

Each USER can use the self-guided tool to search many BOOKS, and each BOOK can be searched by the self-guided tool for many USERS. Each search by the self-guided tool has the attributes name, subject, ISBN, publisher, and author which inform searching properties of the BOOK.

REQ\_LSR\_1 - Relationship between Library Staff and Reservation

Rationale: Each LIBRARY STAFF can manage many RESERVATIONS, and each RESERVATION can be managed by many LIBRARY STAFF.

REQ\_LSR\_2 - Relationship between Library Staff and Reservation

Rationale: Each LIBRARY STAFF can get intelligent reports from many RESERVATIONS, each RESERVATION can generate an intelligent report for many LIBRARY STAFF. Each intelligent report has a calculated attribute Number of Extra Copies Needed.

REQ\_LSL\_1 - Relationship between Library Staff and Loan

Rationale: Each LIBRARY STAFF can manage many LOANS, and each LOAN can be managed by many LIBRARY STAFF.

Library Management Relationship:

REQ\_BR\_1 - Relationship between Book and Reservation

Rationale: Each BOOK can be reserved by many RESERVATIONS, and each RESERVATION must reserve a single BOOK.

REQ\_RL\_1 - Relationship between Reservation and Loan

Rationale: Each RESERVATION must belong to only one LOAN, and each LOAN must have only one RESERVATION.

## Non-Functional Requirements

### Product requirements:

NON\_REQ\_PR\_1 - Usability Requirement Rationale:

* + - The system shall allow the users to access the system from Library Computer User interface. The system uses a Swing GUI client as an interface. Since all users familiar with the usage of UI, no more training is required
    - A user should be a member of the University of Wonderland(UWON) and has an active university email which can be used to register a library account.
    - The system is user friendly and online help makes using the system easy and also the

product will support multiple languages such as Madarin, Japanese, Korean and English.

* + - The system shall generate bibliographical results when users conduct the self-guided search.
    - The system shall generate an Intelligent report when librarians send a request to the system.
    - The users borrow the books or copies at most 3 times simultaneously, which means users can make loans at most 3 times. Users shall return books or copies on time otherwise the library system will issue a fine per day after the deadline.
    - The system shall return a precise reservation status when users check the reservation status.
    - The librarians will make a reservation and manage the loans for each users correctly.

NON\_REQ\_PR\_2 - Availability Requirement Rationale:

* + - The library system is available for all the departments and users and is used 24 hrs and 365 days a year. The system should be operational 24 hrs a day and 7 days a week.
    - The information(Book and Copy, Reservation, Loan) is refreshed at regular intervals depending on whether some updates have occurred or not.
    - The format for bibliographical queries and answers shall be accessible to students from any department, responses to these queries shall take less than 3 seconds

NON\_REQ\_PR\_3 - Reliability Requirement Rationale:

* + - The library system should accurately provide real time bibliographical search, reservation, book, and loan’s information taking into consideration various concurrency issues. The system has to be 100% reliable and shall provide 100% access reliability.
    - Mean Time to Repair - Even if the library system falls, the system will be recovered back up in an hour or less.
    - Restarting the library system is acceptable when a failure event occurs
    - Failure rate less than or equals 2%
    - The system should send automatically a warning deadline email to users about book

reservation’s due date, and loan deadline warning.

NON\_REQ\_PR\_4 - Performance Requirement Rationale:

* + - The system shall respond to the user in less than 3 seconds from the time all kinds of requests such as bibliographical search, book reservation, intelligent report generation and etc are submitted. The responses to view results shall take no more than 4 seconds to appear on the UI page. The search result shall be complete and effective.
    - The library system shall support 20000 concurrent users and should be able to handle a large amount of data. Thus it shall accommodate a high number of books and users without fault.
    - The library system shall handle expected and unexpected errors in ways that prevent loss in information and downtime period.
    - The library system has strong traceability to previous borrowers when books are found to be damaged.

NON\_REQ\_PR\_5 - Supportability(including maintainability and portability Rationale:

* + - Any changes (new user registration, database changes, reservation change, book database) must be verified once per day at least by library staff. The system should provide automatic notification email to users about book’s overdue, and loan deadline warning.
    - The library system shall be ported in 64-bit linux operating system and Android operating system.
    - The library system shall be maintained by library staff and shall be updated once a month

### Organizational requirements:

NON\_REQ\_QR\_1 - Implementation Requirements Rationale:

Deploying and commissioning the library system shall take one month or less, and the implementation involves library system installation, maintenance, library staff and user training. Ability to deal with any department library system branch, users, user categories, library staff, book categories, books, loans and other data. Easy to use search and tracking policies, strong user management, intuitive navigation, and extensive administrative permission for staff accounts.

NON\_REQ\_QR\_2 - Software Constraints Rationale:

* + - The quality of the database is maintained in such a way that it can be very user friendly to all the users of the database. The library databases shall be MySQL.
    - To develop the web server of the library system we will use Java and Java Swing GUI

to design UI pages. Other related software associated with the library system shall be written by using Java, to comply with University of Wonderland(UWON) library’s

policy.

NON\_REQ\_QR\_3 - Hardware Constraints Rationale:

The system requires a database in order to store persistent data. The database should have backup capabilities.

### External requirements:

NON\_REQ\_ER\_2 - Security Requirements Rationale:

* + - The databases may get crashed at any certain time due to virus or operating system failures. Therefore the database shall be backed up by library staff and it shall be recovered within fifteen minutes when the failure happened.
    - There are different categories of users namely library staff, members(student, faulty, professor), administrator etc. Depending on the category of user the access rights are different. If the user is a staff or administrator then he is able to modify the data, delete, reserve books and manage loans for members. All the other users other than library staff and administrators only have the rights to retrieve the information about the database.
    - All system data must be backed up every 24 hours and the backup copies stored in a secure location which is not in the same building as the system.

NON\_REQ\_ER\_3 - Legal Requirements Rationale:

User information shall be protected. The development process to be used must be explicitly defined and must be conformant with ISO 9000 standards.

## Traceability Matrix

Table 14: Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Design | Implementation | Test Case |
| REQ\_1 | User | SCREEN\_01 | TS\_01, TS\_02, UT\_01, UT\_02, UT\_03, UT\_04, UT\_05, UT\_06, UT\_07, UT\_08, UT\_09, IT\_01, IT\_02, IT\_03, IT\_04, IT\_05, ST\_01, ST\_02, ST\_03, ST\_04 |
| REQ\_2 | LibraryStaff | SCREEN\_01 | TS\_01, UT\_01, UT\_02, UT\_03, UT\_04, UT\_05, UT\_06, IT\_01, IT\_02, IT\_03, ST\_01, ST\_02 |

|  |  |  |  |
| --- | --- | --- | --- |
| REQ\_3 | Book | SCREEN\_02 | UT\_07, UT\_08, UT\_09, IT\_04, IT\_05, ST\_03, ST\_04 |
| REQ\_4 | Reservation | SCREEN\_02 |  |
| REQ\_5 | Loan | SCREEN\_02 |  |
| REQ\_UB\_1 | SelfGuidedSearch | SCREEN\_03 | UT\_07, UT\_08, UT\_09, IT\_04, IT\_05, ST\_03, ST\_04 |
| REQ\_LSR\_1 | IntelligentReport | SCREEN\_04 |  |

# Design

## Entity Relationship Diagram (conceptual model)

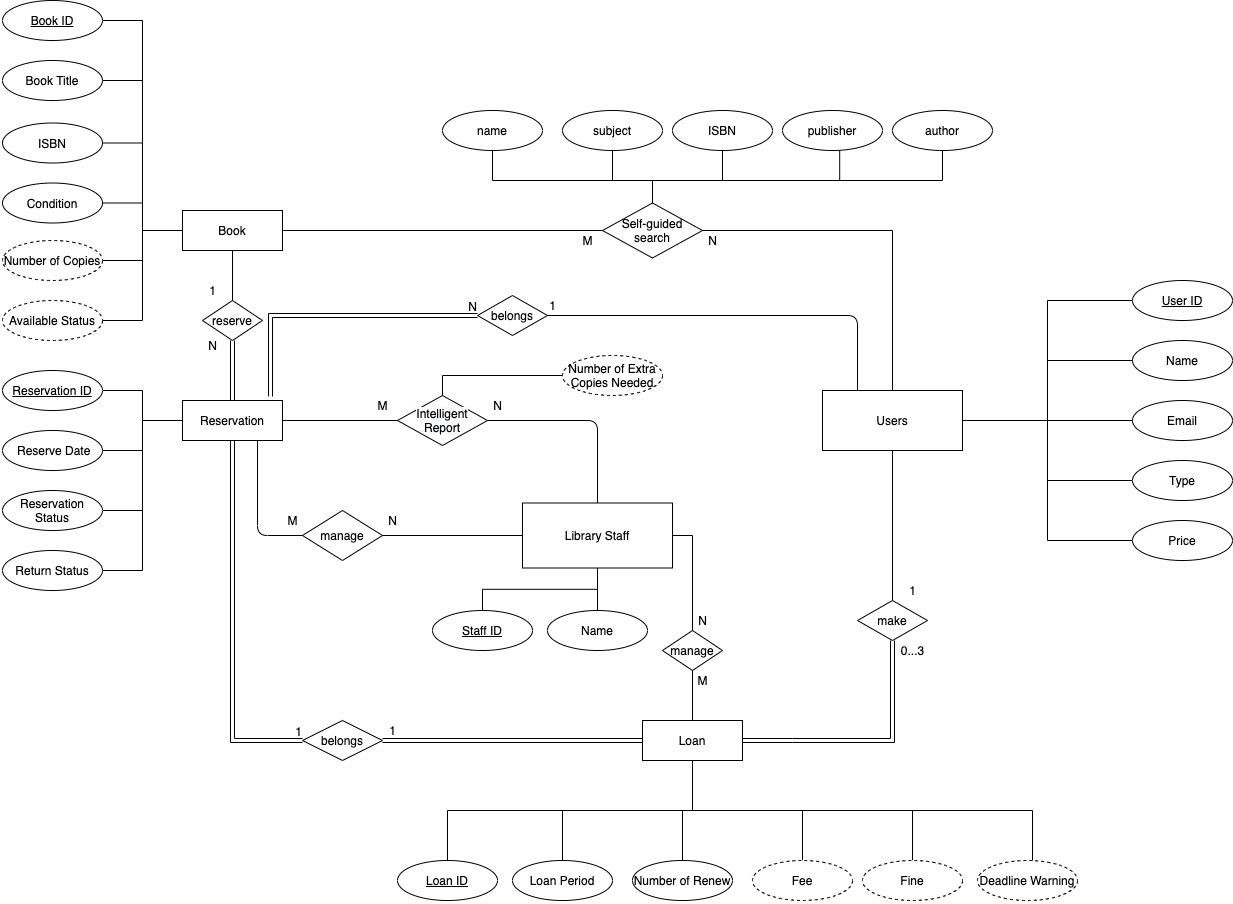


Figure 1. ER Diagram (conceptual model)

## Logical Model

Figure 2. Logical Model

## Class Diagram

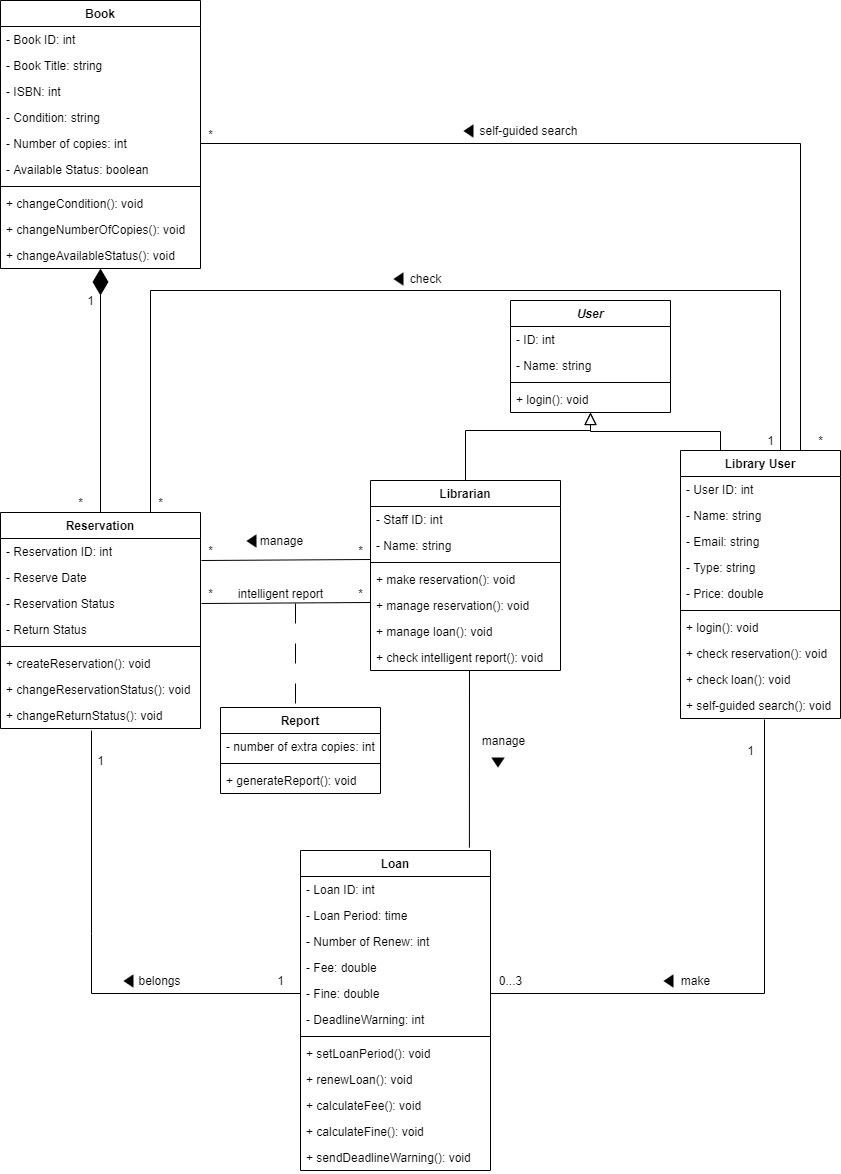


Figure 3. Class Diagram

## Sequence Diagram

### Make a Book Reservation

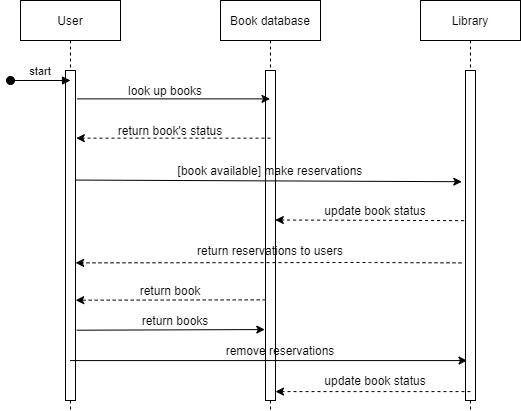


Figure 4. Sequence Diagram (Make a Book Reservation)

### Manage Loan

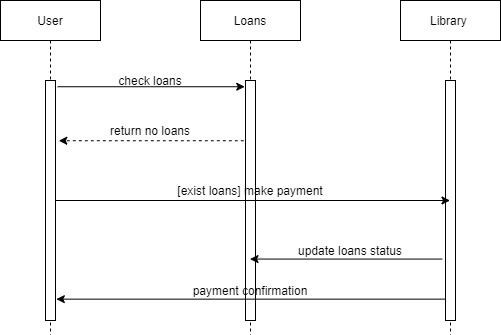


Figure 5. Sequence Diagram (Manage Loan)

## State Machine Diagram

### Make a Book Reservation

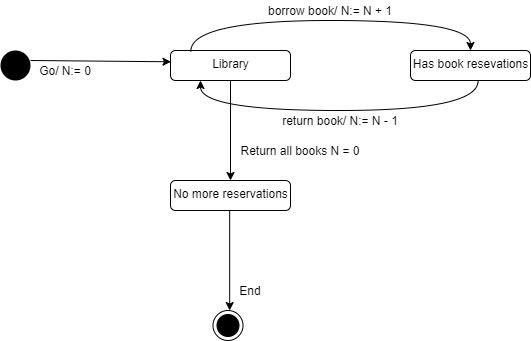


Figure 6. State Machine Diagram (Make a Book Reservation)

### Manage Loan

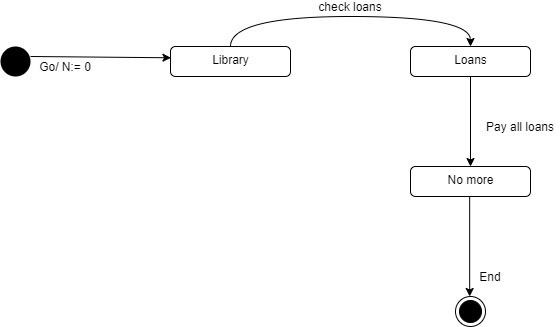


Figure 7. State Machine Diagram (Manage Loan)

## System Architecture View and Style/Pattern

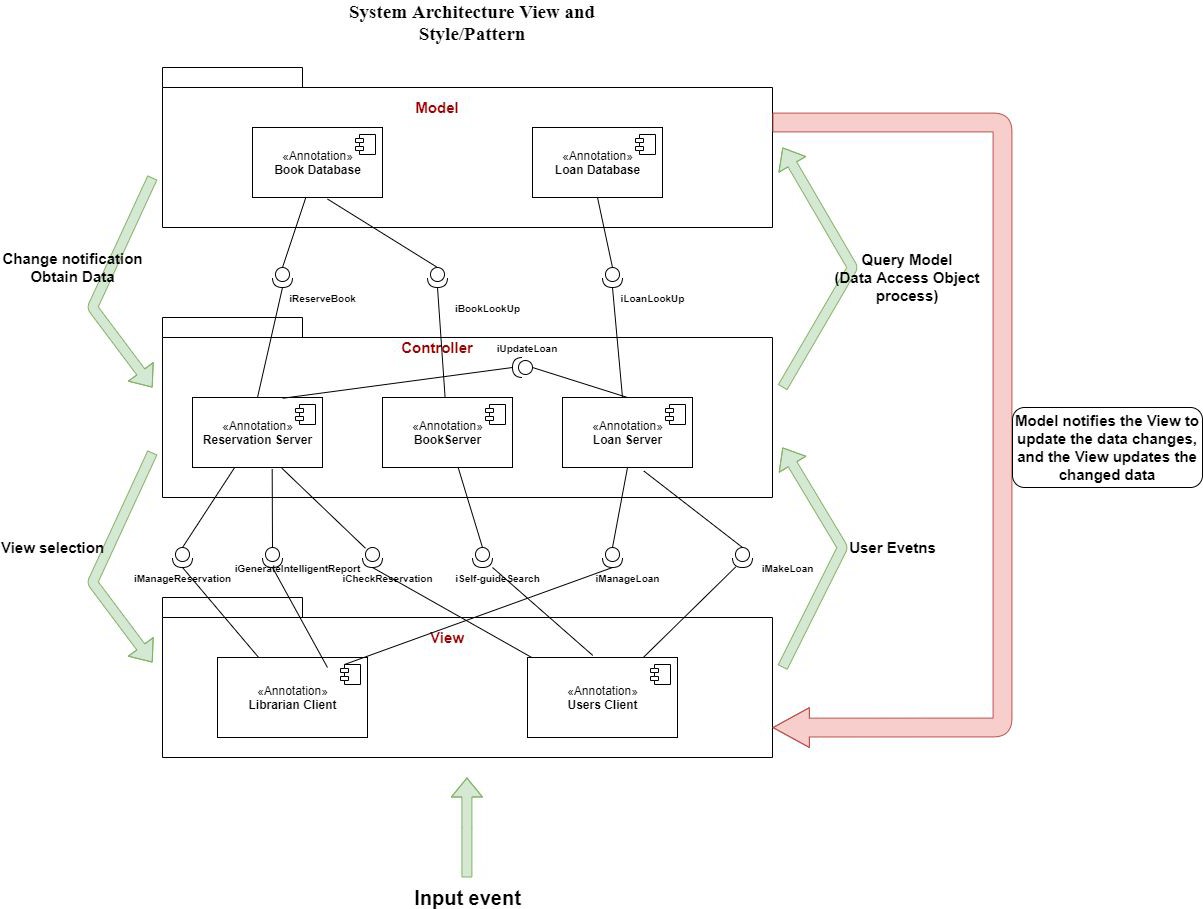


Figure 8. MVC architecture Pattern

We will use MVC System Architecture Pattern to design this project, as we know, **the Model layer** is business data, **View layer** is user interface and **Controller** is used to handle dispatcher events. When the View layer accepts a user interface request(input event) such as self-guided search, makes a loan etc, it will transfer requests to Controller, and Controller will operate Model layer and manipulate the database such as **Add**, **Delete**, **Update**, **Search** functions. Finally, the Model layer will notify the View layer to update the data changes, and View updates changed data.

# Paper-based Prototype

## Technical Description of the System

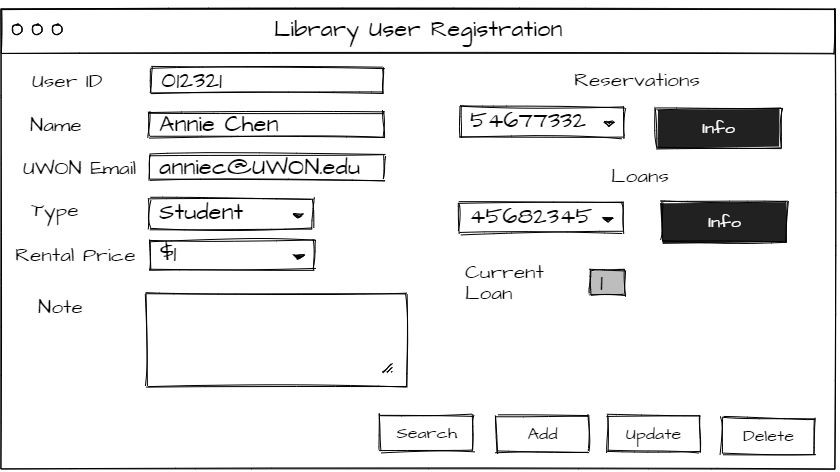
SCREEN\_01

Figure 1. SCREEN\_01: Library User Registration

SCREEN\_02

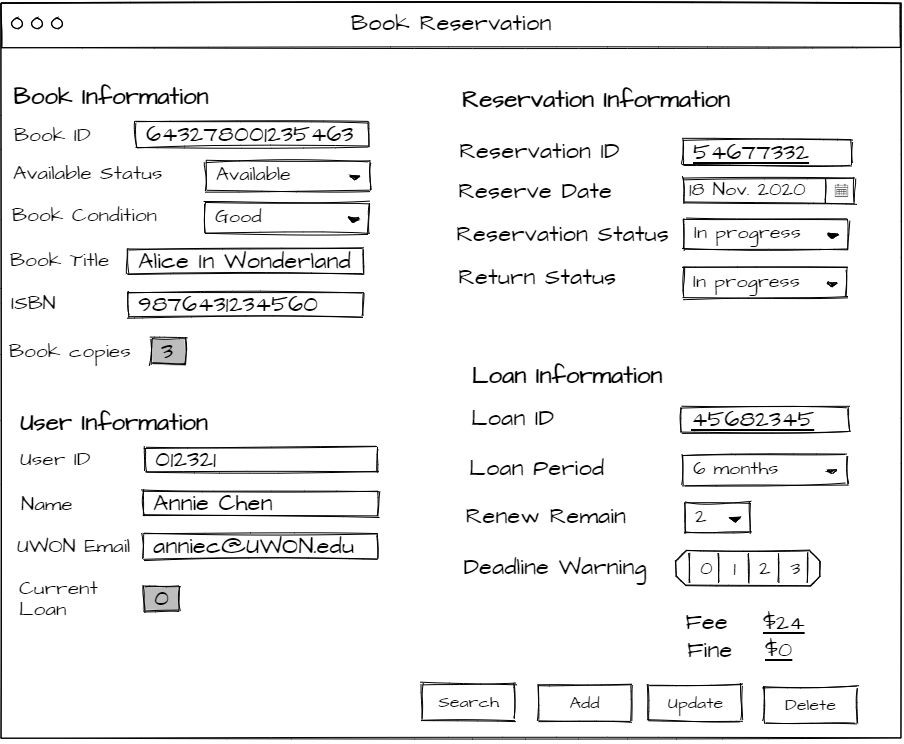


Figure 2. SCREEN\_02: Book Reservation

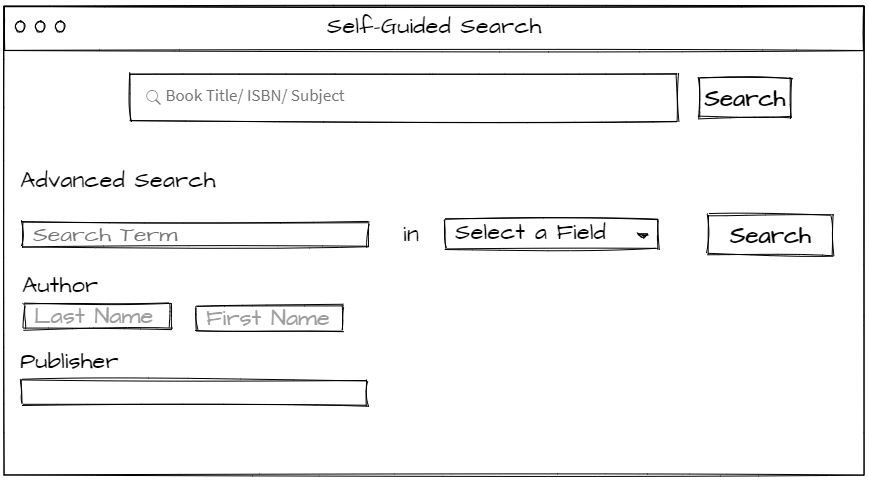
SCREEN\_03

Figure 3. SCREEN\_03: Self-guided Search

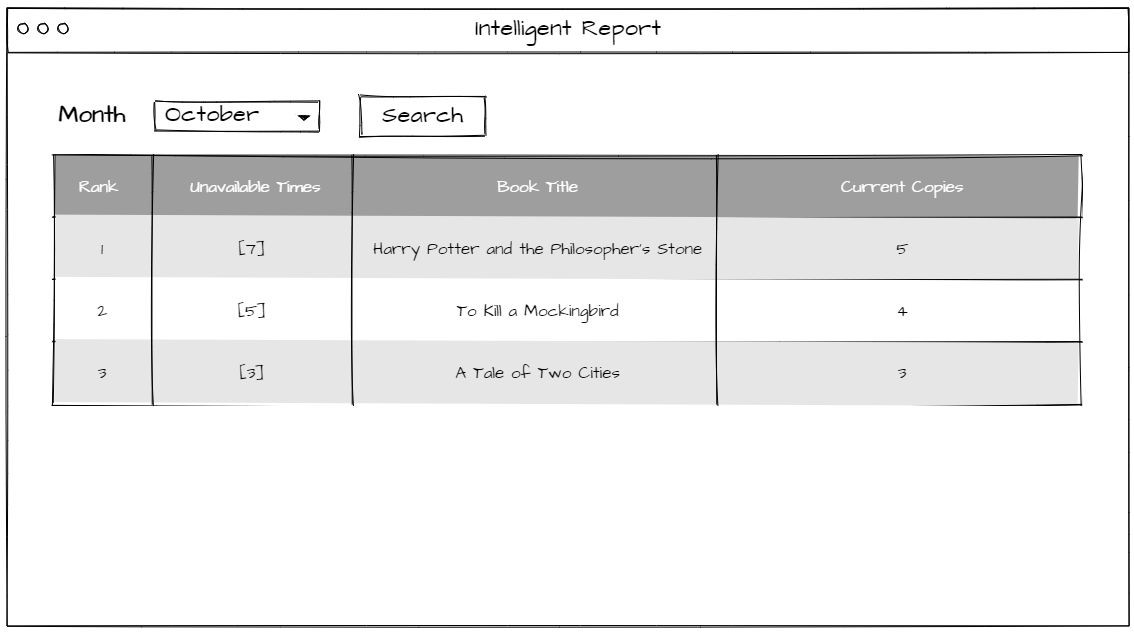
SCREEN\_04

Figure 4. SCREEN\_04: Intelligent Report

# Testing Plan

## Test Scenarios

Table 1: Test Scenarios

|  |  |  |
| --- | --- | --- |
| Test Scenarios | | |
| Scenario ID | Title | Description |
| TS\_01 | Log In | Users are able to login to the system by user’s university email and password on the login screen, and then the application redirects to the next screen based on the type of the user. |
| TS\_02 | Search Books | Library users are able to search books by name, subject, ISBN, publisher, or author of the book on the self-guided search screen. |

## Unit Tests

Table 2: UT\_01

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_01 |
| Test Case | Login email format is invalid |
| Pre-conditions | 1. The “Log In” screen loaded up. 2. Email field and password is empty. 3. The “Login” button is disabled. |
| Instructions (Steps) | 1. Enter a random string which does not match email format on the email field |
| Expected Outputs | 1. The email field turns red, and an error message “email is invalid” displays under the email field. 2. The “Login” button is disabled. |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies |  |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 3: UT\_02

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_02 |
| Test Case | Login password length is too short. |
| Pre-conditions | 1. The “Log In” screen loaded up. 2. Email field and password are empty. 3. The “Login” button is disabled. |
| Instructions (Steps) | 1. Enter a random string which the length is less than 8 on the password field |
| Expected Outputs | 1. The password field turns red, and an error message “password is too short” displays under the password field. 2. The “Login” button is disabled. |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies |  |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 4: UT\_03

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_03 |
| Test Case | Login password length is too long. |
| Pre-conditions | 1. The “Log In” screen loaded up. 2. Email field and password are empty. 3. The “Login” button is disabled. |
| Instructions (Steps) | 1. Enter a random string which the length is more than 13 on the password field |
| Expected Outputs | 1. The password field turns red, and an error message “password is too long” displays under the password field. 2. The “Login” button is disabled. |
| Type of Implementation | MANUAL |

|  |  |
| --- | --- |
| Phase | Implementation |
| Dependencies |  |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 5: UT\_04

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_04 |
| Test Case | Login email and password are valid |
| Pre-conditions | 1. The “Log In” screen loaded up. 2. Email field and password are empty. 3. The “Login” button is disabled. |
| Instructions (Steps) | 1. Enter an university email on the email field 2. Enter a password on the password field |
| Expected Outputs | 1. The “Login” button is enabled. |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies |  |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 6: UT\_05

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_05 |
| Test Case | When the login success message is displayed, clicking the “OK” button can trigger the *redirect()* function. |
| Pre-conditions | 1. The login success message is displayed with a “OK” button. 2. Mock the *redirect()* function which is triggered by clicking the “OK” button to print “redirecting...” on the console |
| Instructions (Steps) | 1. Click the “OK” button |
| Expected Outputs | 1. The message “redirecting…” is printed out on the console |

|  |  |
| --- | --- |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies | IT\_02 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 7: UT\_06

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | UT\_06 |
| Test Case | When the login error message is displayed, clicking the “OK” button can close the error message. |
| Pre-conditions | 1. The login error message is displayed with a “OK” button. |
| Instructions (Steps) | 1. Click the “OK” button |
| Expected Outputs | 1. The login error message disappears. |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies | IT\_03 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 8: UT\_07

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | UT\_07 |
| Test Case | A dropdown list of subject fields shows up after the “Select a Field” selection field is clicked. |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. |
| Instructions (Steps) | 1. Click on the “Select a Field” selection field. |
| Expected Outputs | 1. A dropdown list shows up and displays a list of subject fields such as “Computer Science”, “Biology”, “Physics”, and etc. |

|  |  |
| --- | --- |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies | ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

Table 9: UT\_08

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | UT\_08 |
| Test Case | The “Select a Field” selection field displays the selection after the subject field is selected. |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. |
| Instructions (Steps) | 1. Click on the “Select a Field” selection field. 2. Select “Computer Science” on the dropdown list. |
| Expected Outputs | 1. The dropdown list disappears. 2. “Computer Science” shows on the “Select a Field” selection field under Advanced Search section |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies | UT\_07, ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

Table 10: UT\_09

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | UT\_09 |
| Test Case | All text fields are able to take inputs from user input and show on text fields. |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. |

|  |  |
| --- | --- |
| Instructions (Steps) | 1. Enter “a” on the search bar 2. Enter “b” on the “Search Term” field 3. Enter “c” on the “Last Name” field under Author section 4. Enter “d” on the “First Name” field under Author section 5. Enter “e” on the text field under Publisher section |
| Expected Outputs | 1. “a” shows on the search bar 2. “b” shows on the “Search Term” field 3. “c” shows on the “Last Name” field under Author section 4. “d” shows on the “First Name” field under Author section 5. “e” shows on the text field under Publisher section |
| Type of Implementation | MANUAL |
| Phase | Implementation |
| Dependencies | ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

## Integration Tests

Table 11: IT\_01

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | IT\_01 |
| Test Case | The *login()* function can be triggered by the “Login” button. |
| Pre-conditions | 1. Mocking the *login()* function takes email and password as arguments and return an array [email, password] 2. The “Log In” screen loaded up. |
| Instructions (Steps) | 1. Enter an valid university email on the email field 2. Enter a valid password on the password field 3. Click the “login” button |
| Expected Outputs | The *login()* function is fired and returns an array as *result* which *result[0]* is equal to the value in the email field and *result[1]* is equal to the value in the password field. |
| Type of Implementation | MANUAL |
| Phase | Testing and Integration |
| Dependencies | UT\_04 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 12: IT\_02

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | IT\_02 |
| Test Case | Login success message displays. |
| Pre-conditions | 1. Mock the login api responds a mock login credential 2. The “Log In” screen loaded up. |
| Instructions (Steps) | 1. Enter an university email on the email field 2. Enter a password on the password field 3. Click the “login” button |
| Expected Outputs | 1. The application received the login credential from login api response and stored the login credential. 2. A success message is displayed with a “OK” button. |
| Type of Implementation | MANUAL |
| Phase | Testing and Integration |
| Dependencies | UT\_04, IT\_01 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 13: IT\_03

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | IT\_03 |
| Test Case | Login error message displays. |
| Pre-conditions | 1. Mock the login api responds an error 2. The “Log In” screen loaded up. |
| Instructions (Steps) | 1. Enter an university email on the email field 2. Enter a password on the password field 3. Click the “login” button |
| Expected Outputs | 1. The application received the error response. 2. An error message is displayed with a “OK” button. |
| Type of Implementation | MANUAL |

|  |  |
| --- | --- |
| Phase | Testing and Integration |
| Dependencies | UT\_04, IT\_01 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 14: IT\_04

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | IT\_04 |
| Test Case | The general search button which is behind the search bar can trigger the  *generalSearch()* function with the value in the search bar. |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. 3. Mock *generalSearch()* function to take *searchStr* as argument and print the *searchStr* to console. |
| Instructions (Steps) | 1. Enter “a” on the search bar 2. Enter “b” on the “Search Term” field 3. Selection “Computer Science” on the “Select a Field” selection field under Advanced Search section 4. Enter “c” on the “Last Name” field under Author section 5. Enter “d” on the “First Name” field under Author section 6. Enter “e” on the text field under Publisher section 7. Click the “Search” button after the search bar. |
| Expected Outputs | 1. A message “a” is printed out on the console. |
| Type of Implementation | MANUAL |
| Phase | Testing and Integration |
| Dependencies | UT\_08, UT\_09, ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

Table 15: IT\_05

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | IT\_05 |
| Test Case | The advanced search button which is under the Advanced Search section can trigger *advancedSearch()* function with values of all text fields and the |

|  |  |
| --- | --- |
|  | selection field on the self-guided search screen. |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. 3. Mock *advancedSearch()* function to take *searchStr*, *subject*, *subjectField*, *lastName*, *firstName*, and *publisher* as arguments and print an array [searchStr, subject, subjectField, lastName, firstName, publisher] to console. |
| Instructions (Steps) | 1. Enter “a” on search bar 2. Enter “b” on the “Search Term” field 3. Selection “Computer Science” on the “Select a Field” selection field under Advanced Search section 4. Enter “c” on the “Last Name” field under Author section 5. Enter “d” on the “First Name” field under Author section 6. Enter “e” on the text field under Publisher section 7. Click the “Search” button under the Advanced Search section. |
| Expected Outputs | 1. A message [“a”, “b”, “Computer Science”, “c”, “d”, “e”] is printed out on the console. |
| Type of Implementation | MANUAL |
| Phase | Testing and Integration |
| Dependencies | UT\_08, UT\_09, ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

## System/Acceptance Tests

Table 16: ST\_01

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | ST\_01 |
| Test Case | Successfully login by student |
| Pre-conditions | 1. The “Log In” screen loaded up. |
| Instructions (Steps) | 1. Enter an university email of a student on the email field 2. Enter a correct password on the password field 3. Click the “login” button 4. Click the “OK” button after the login success message is displayed |
| Expected Outputs | 1. The application redirects to the self-guided search screen. |

|  |  |
| --- | --- |
|  | 2. The self-guided search screen loads up. |
| Type of Implementation | MANUAL |
| Phase | After Integration Test |
| Dependencies | UT\_04, IT\_01， IT\_02 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 17: ST\_02

|  |  |
| --- | --- |
| Test Scenario ID | TS\_01 |
| Test Scenario | Log In |
| Test Case ID | ST\_02 |
| Test Case | Successfully login by librarian |
| Pre-conditions | 1. The “Log In” screen loaded up. |
| Instructions (Steps) | 1. Enter an university email of a librarian on the email field 2. Enter a correct password on the password field 3. Click the “login” button 4. Click the “OK” button after the login success message is displayed |
| Expected Outputs | 1. The application redirects to the librarian dashboard screen. 2. The librarian dashboard screen loads up. |
| Type of Implementation | MANUAL |
| Phase | After Integration Test |
| Dependencies | UT\_04, IT\_01, IT\_02 |
| Requirement Tested | REQ\_1, REQ\_2 |

Table 18: ST\_03

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | ST\_03 |
| Test Case | General Search |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. |

|  |  |
| --- | --- |
| Instructions (Steps) | 1. Enter “Software Engineering” on the search bar 2. Click the “Search” button after the search bar. |
| Expected Outputs | 1. The application redirects to the screen which displays the list of found books. 2. All titles of books on the screen include keywords “Software Engineering”. |
| Type of Implementation | MANUAL |
| Phase | After Integration Test |
| Dependencies | IT\_04, ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

Table 19: ST\_04

|  |  |
| --- | --- |
| Test Scenario ID | TS\_02 |
| Test Scenario | Search Books |
| Test Case ID | ST\_04 |
| Test Case | Advanced Search |
| Pre-conditions | 1. Library user is logged in. 2. The self-guided search screen loaded up. |
| Instructions (Steps) | 1. Enter “Software Engineering” on the search bar 2. Enter “Wiley” on the text field under the Publisher section. 3. Click the “Search” button under the Advanced Search section. |
| Expected Outputs | 1. The application redirects to the screen which displays the list of found books. 2. All books on the screen have titles which include keywords “Software Engineering” and publishers which are “Wiley”. |
| Type of Implementation | MANUAL |
| Phase | After Integration Test |
| Dependencies | IT\_05, ST\_01 |
| Requirement Tested | REQ\_1, REQ\_3, REQ\_UB\_1 |

# Discussion

The UWON Library System project helps us review and understand the knowledge covered in the class such as software life cycle, configuration management, team organization, software testing, and design patterns. We understand how to apply those principles of software engineering to the design, development, maintenance, testing, and evaluation of computer software when we work on this project and fully integrate software engineering principles into the real situation. The UWON Library System project consists of 6 parts as Project plan, Configuration Plan, Requirements Specification, Design, Paper-based Prototype, and Testing Plan, and some of them require us to overcome certain difficulties to accomplish. Let me show you the challenges that we have encountered in this project and how we solved them finally.

In the Project Plan, the difficulty is how to determine our process model. As we know, the Evolution Prototype is used to obtain the requirements of some aspect of the system. The development of one or more prototypes may help to better understand the requirements of this project and then we can get some feedback from users when prototypes are tested intensively so that we can modify and perfect our requirements and coding and then move to the production phase. That’s the reason why we select Evolution Prototype as our process model. In addition, our teams also have to do research for the work package to find out what it means and its format, what’s more, we also need to search what is perfect hardware requirements to support the library system.

In the Identify configuration items, we thought we needed to select some tools which can be used for the project at the beginning, finally, we figured out that configuration items are those documents we generated to control the project. This is one of the tricky parts we have encountered.

ER-Diagram is a necessary and important part of our project and many other diagrams and Assumptions and functional requirements should be based on it. We were working on ER-Diagram so hard together for 4 hours and we used Google docs and Zoom to get everybody on the same page and to design it. Some teams live in different States and we are in different time zones but we still find some time to set up weekly meetings together no matter how difficult it is. The only goal for our teams is to finish our UWON Library System project on time and do our best to perfect it. We have accumulated lots of valuable experience in the UWON Library project, which will provide references for our future work.

# Reference

1. JGraph Ltd(2005). Draw.io Diagram (V 7.1.9)[Diagram Modeling software] Published in 2005. Retrieved from <http://app.diagrams.net/>
2. Kenji Hiranabe(1999). Astah Professional JUDE(V 1.4)[UML Modeling software] Published in 1999. Retrieved from <http://astah.net/>
3. Best of BI (2009) SQL Power Architect (V 1.0.7)[Data modeling Software] Published in 2009. Retrieved from <http://www.bestofbi.com/>
4. Produle System(2008). MockFlow(Basic)[UI mockup Software]Published in 2008. Retrieved from <https://mockflow.com/#login>
5. Textbook: Vliet H. V., “Software Engineering: Principles and Practice”, 3th Edition, Wiley.