

**System Requirement Specifications**

**for**

**Library information appication**

**Version: 2.0**

**Date: 18/06/09**

**20143692 강경석**

**20144103 조유성**

**20145026 김상범**

**20141043 양준호**

**20146110 문태현**

**20143583 김기환**

Table of Contents

Index

[[Library information appliciation]](#_Toc480998875)

[Version 2.0](#_Toc480998877)

[June 9, 2018](#_Toc480998878)

[1. Product Description](#_Toc480998879)

[1.1 Functional Requirements](#_Toc480998880)

[1.2 Non-functional requirements](#_Toc480998880)

[1.3 Constraints](#_Toc480998880)

[2. UI(user interface) Diagrams](#_Toc480998881)

[3. use cases](#_Toc480998881)

[4. Product Description](#_Toc480998879)

[4.1 software tools](#_Toc480998880)

[4.2 job descriptions, work process](#_Toc480998880)

[4.3 schedule/Timeline](#_Toc480998880)

[4.4 Risk summary](#_Toc480998880)

1. **Product Description**
   1. **Functional Requirements**
2. Users must be able to know whether the book has benn rented.

* Users should know whether they can borrow the book now.

1. Users should know information of books to return.

* To return books on time

1. Administrator should know who borrowed the books.

* To manage books and change late fees

1. Administrator should able to add or remove books.

* To manage damaged or new books
  1. **Non-functional Requirements**

Product Requirement

* The change of the condition of the books (lost, new, borrowed) should be updated occasionally.

External Requirement

* There should be exact regulation for users who damage or lose borrowed books habitually.
  1. **Constraints**

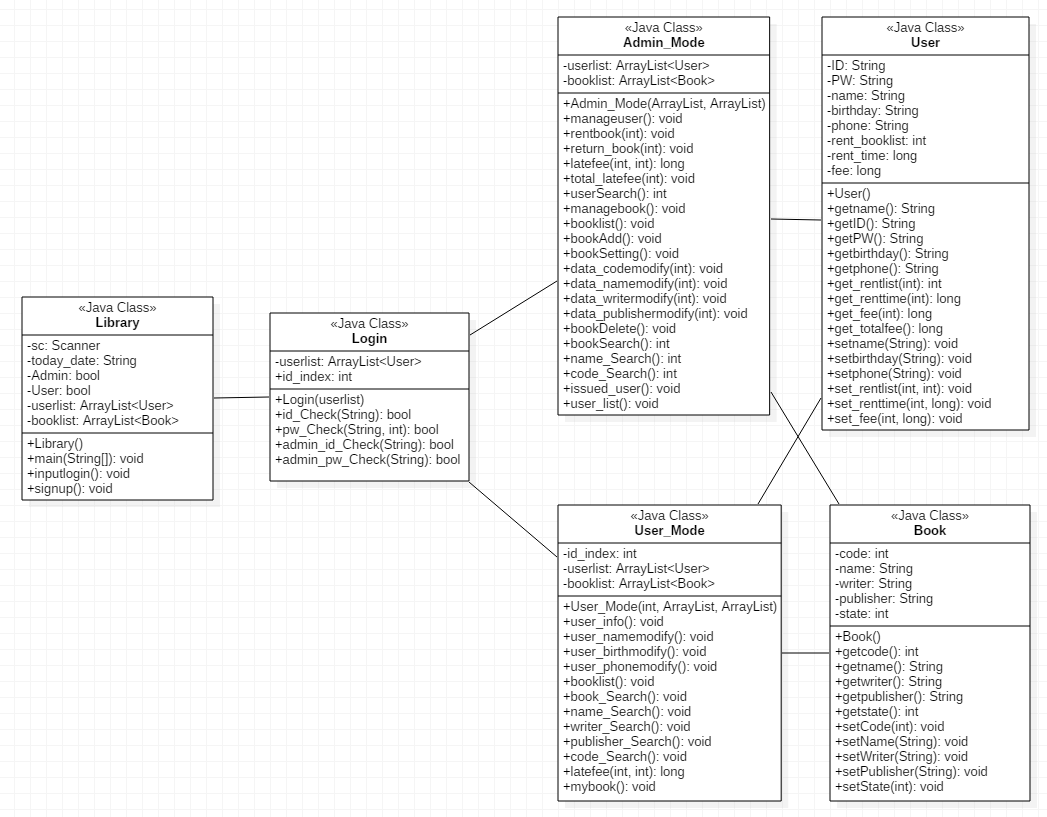
1. *Technical Constraints*

* Programming Language – We will basically work on the JAVA, however we wil also use some other software tools, like JAVAFX or JUNIT. All of our teammates have not used these software tools, so we have to study these tools to complete our program.
* Operating System or Platforms supported – We will basically work on Windows, but we don’t know whether our program satisfies the platform of Linux, iOS, or Qt on Solaris.

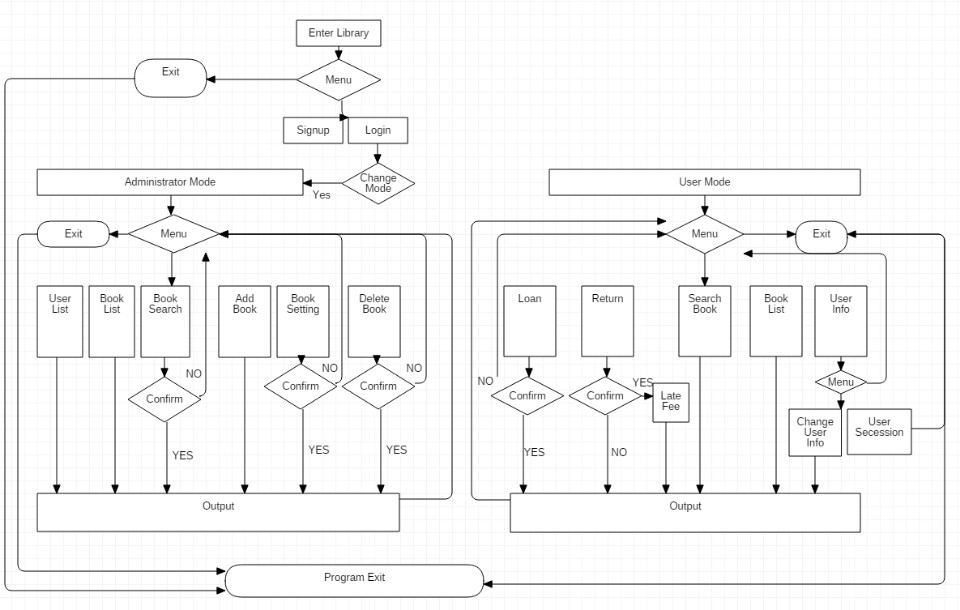
1. *Budget Constraints*

* Schedule – Our teammates all have our own individual schedule and work time. It can be hard to schedule our time to work on this project.
* Budget – We don’t need any budget to work on this project right now, however, when our program is published on the server, budget problem will be essential to be considered to maintain or update the program.

1. **UI Diagrams**
2. *Class Diagram*

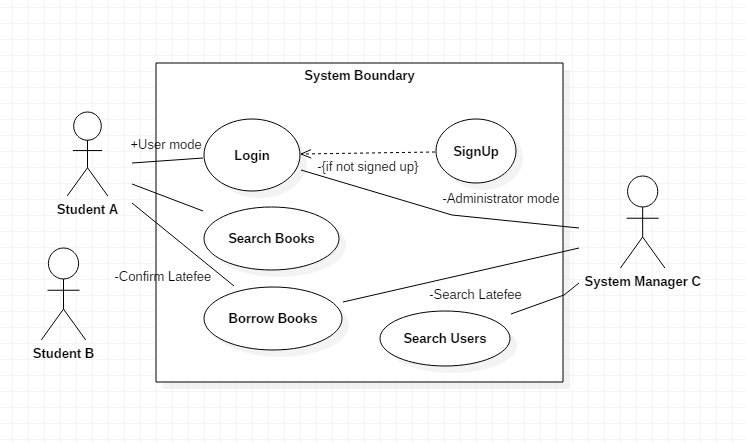
****

1. *Flow Diagram*



1. **Use Cases**
2. Student A wants to check if the book is in the library, so he turns on the ‘library.app’. After confirming the availability of the book, he goes to the library to borrow the book.
3. Student B wants to see how much time he has left for returing books. After confirming the return period of the book, he is relieved by finding out the deadline is still alive.
4. System manager C wants to check which student borrowed the book and broke the deadline. By login into the administrator mode, he can find out which student did not return untill the due date.

**Ex)**

****

1. **Process Description**
   1. **Software Tools**
2. **Development Tools**
   1. **JAVA**

[](http://www.google.co.kr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi14ciorMbbAhUHy7wKHcV-B7wQjRx6BAgBEAU&url=http://m.blog.naver.com/pgh7092/221130579329&psig=AOvVaw3b0iEAIG_ZjmE2oxjrH-BU&ust=1528625614631873)

We will use object-oriented language java to implement the functions of this program divided into class units.

To use the java language efficiently we will use eclipse as a program development tool.

Eclipse has a huge plug-in. In the eclipse Plug-in, we will use ‘junit’ to test our program. This test is mentioned in detail below “build and test”.

We use javafx for GUI. Using eclipse can be a great help in linking our program with javafx

And eclipse provede us many libraries, which will benefit program development

* 1. **JAVAFX**

[](https://www.google.co.kr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi0t9q4rMbbAhXIxbwKHQowCnMQjRx6BAgBEAU&url=https://www.javatpoint.com/javafx-tutorial&psig=AOvVaw1x8ZFeE2JZHFwwViqIZg3B&ust=1528625637721373)

In order to improve user’s convenience and environment, our team will implement GUI using javajx.

There is an example of a GUI that we will implement in “how to use software”.

All the GUI necessary for the program as well as the example above will be implemented using javafx

1. **Test Tools**

**2.1) Maven, GITHUB**

Our team manage source codes in GitHub. By interlocking GitHub and Jenkins, which is automated build tools, we will build automated daily build environment.

There are 3 things for preparations.

- Maven Tool

- GitHub account

- Project to build in GitHub

Maven should build automatically by reading the results automatically pushed by GitHub public storage. And it is also Eclipse Plug in.

**2.2) JUNIT**

[](https://www.google.co.kr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj3qeDNrMbbAhUJvrwKHdI_DCYQjRx6BAgBEAU&url=https://junit.org/junit4/&psig=AOvVaw0gESYZ2W1Na4O1T9ZTwAvi&ust=1528625686390151)

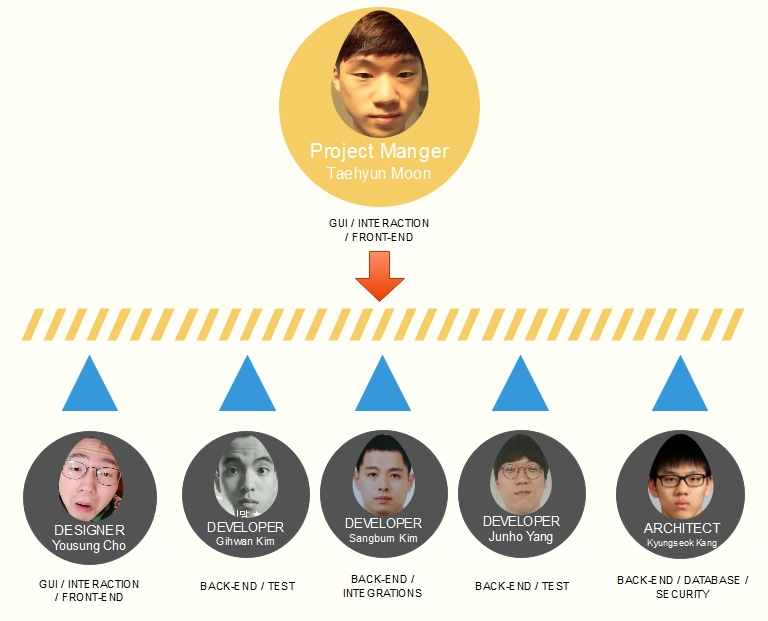
We will use JUnit to implement automated test.

JUnit is a unit testing tool. We do not have to debug inconveniently for using system.out by writing external test program(case). It reduces time for program test and it is in Eclipse for plugin form.

The test result does not remain in just a text file, but remains in Test class so that it can produce test methods and the history of class to the developers. Also, simply uninherited Test may be classified as a unit test.

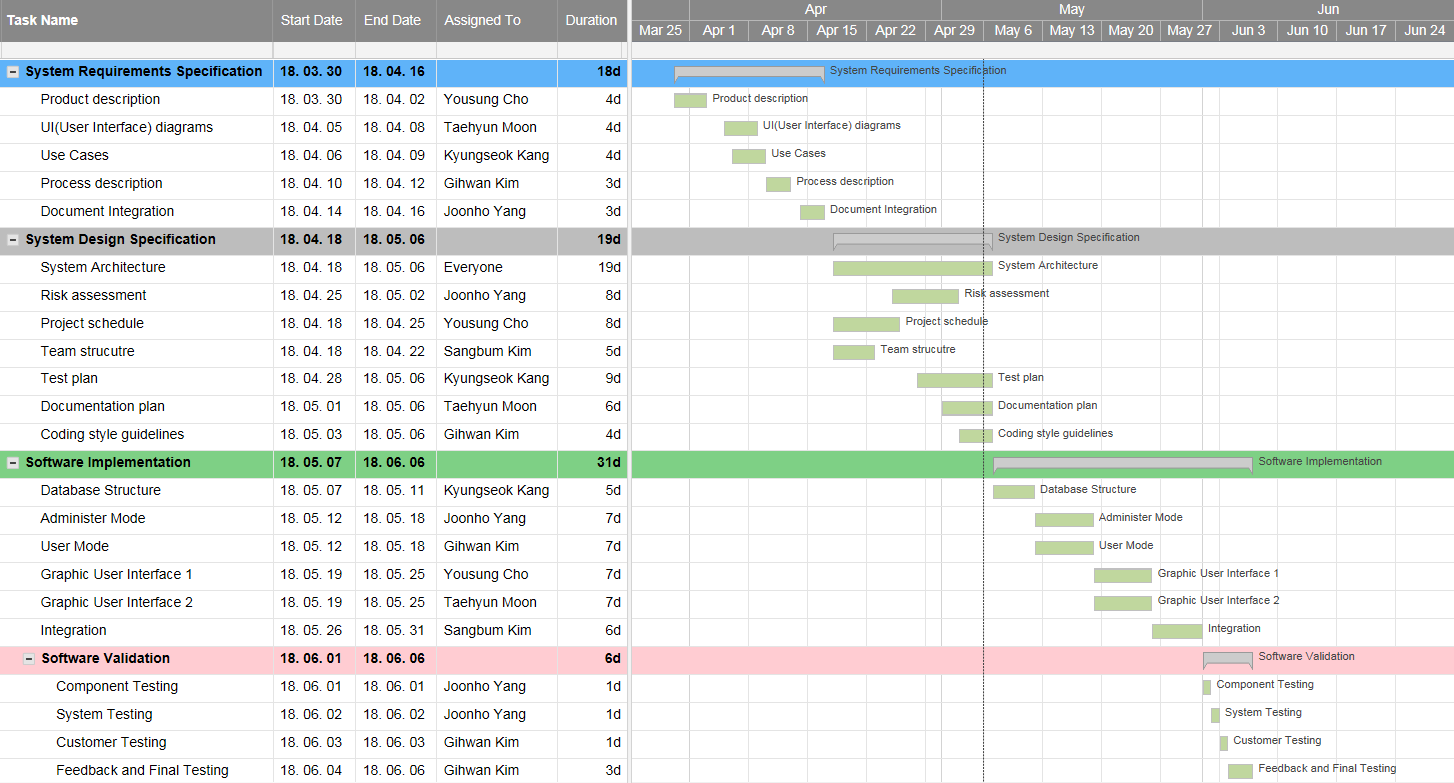
For test classification implement, the settings are like below.

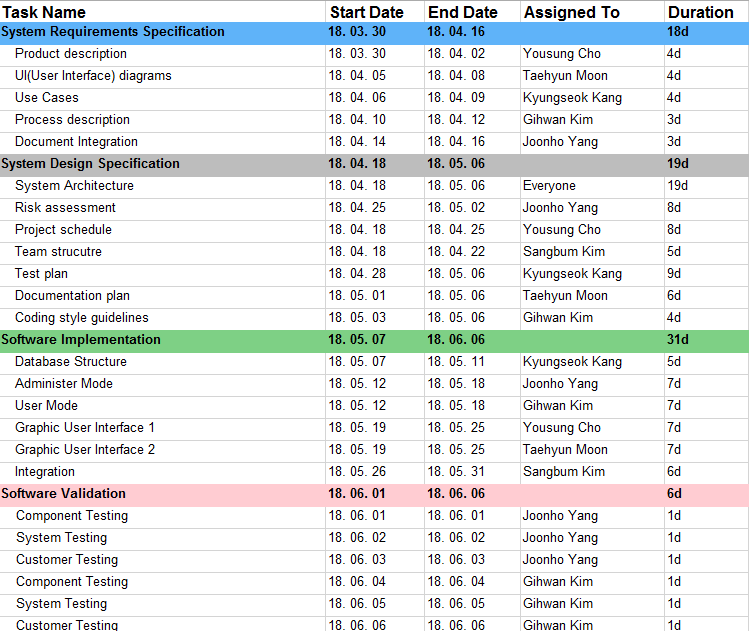
* 1. **Job Descriptions, Work Process**

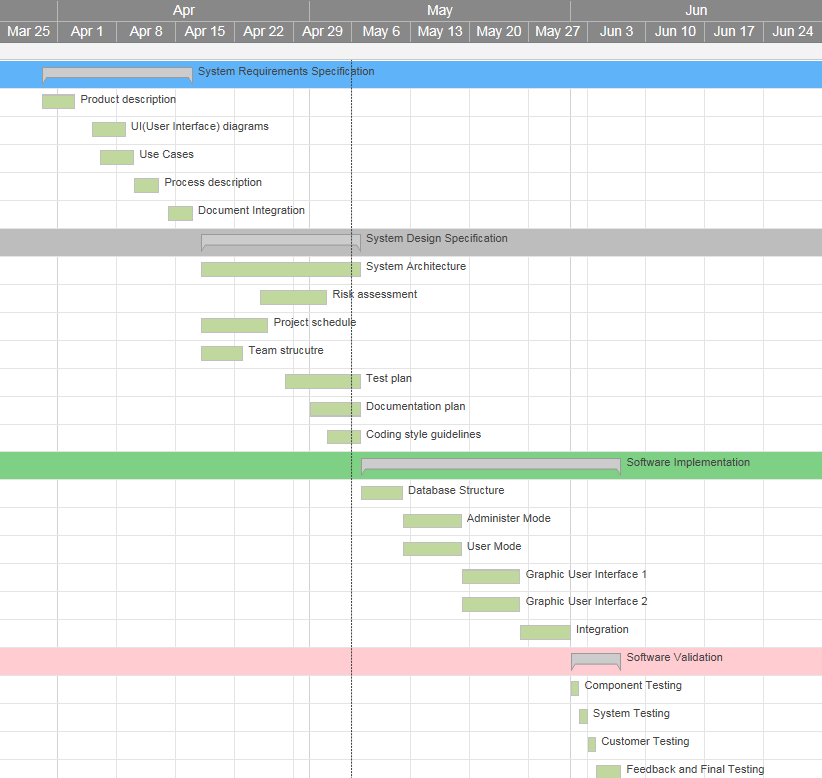


|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Group** | **Role** | **Responsibility** |
| Taehyun Moon | Front-end Team | Core Team leader (Project manager) | •Leading & managing Project Team. •Schedule authority for project. •Splits the tasks among the members of our group •Also support Front-end Part. |
| Yousung Cho | Front-end Team | Designer | •Developing the GUI. •Translate a project engineer’s ideas into the appropriate software. •Design decisions will be made through group discussion and Designer makes a final review. |
| Kyungseok Kang | Back-end Team | Architect | •Creating H/W & S/W architecture. •Make sure the software and system architectures are in synchronization •Manage risk identification and risk mitigation strategies associated with the architecture. |
| Sangbum Kim | Back-end Team | Developer | •Developing the solutions. •Integration with other modules. •Review whether team have coded the code guidelines •Risk management that can occur when integrating |
| Junho Yang | Back-end / Test team | Developer | •Developing the solutions •Defining the appropriate tests required and any associated Test Data •Gathering and managing the Test Data and evaluating the outcome of each test cycle. |

* 1. **Schedule/Timeline**







Mar 30 ~ April 16

- System Requirements Specification

April 18 ~ May 6

- System Design Specification

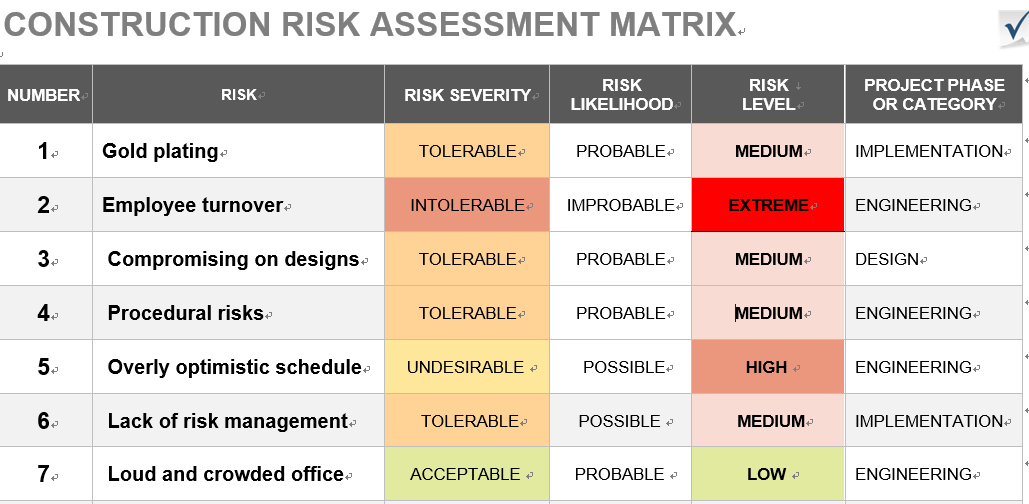
May 7 ~ May 31

- System Implementation

June 1 ~ June 10(change)

- System Validation

* 1. **Risk Summary**

****

**1. Gold plating**

Developers sometimes like to show off their skills by adding unnecessary features. For instance, a developer might add Flash to a basic login module to make it look ‘stylish’. Again, this is a waste of programming hours.

**2. Employee turnover**

Every project has a number of developers working on it. When a developers leaves, he or she may take critical information with him/her. This can delay, and sometimes derail an entire project.

**3. Compromising on designs**

In order to get stuck into the next ‘real’ tasks, developers tend to rush the design-process. This is a waste of programming hours, as designing is the most critical part of software development.

**4. Procedural risks**

Day-to-day operational activities might hamper due to improper process implementation, conflicting priorities, or a lack of clarity in responsibilities.

**5. Overly optimistic schedule**

Time goes unexpectedly to fast.

**6. Lack of risk management**

Projects are always potentially error-prone.

**7. Loud and crowded office**

Loud and crowded office can decrease efficiency of work.