

---

University Project Management Web Application That  
Accounts for Social Isolation



UNIVERSITY OF  
BIRMINGHAM

By: Kaijian Zhang  
Student ID: 2047359

Supervisor: Vincent Rahil  
School of Computer Science, University of Birmingham  
For the degree of MSc Advanced Computer Science  
2020

---

## **Abstract**

There are many project management applications online, but most of them do not support sharing plans and communication between a large group of users. In the coronavirus pandemic, many university students are suffering from sharing their project information through the network with other students. This project aims to design an application that allows the student to share their project information with supervisors and other students and provide a place for the student to discuss their project problem. To achieve this, the project has implemented functions such as create the proposal, approve the proposal, sharing proposals, create a weekly schedule, sharing weekly schedules, create a forum, ask and answer questions in the forum, etc. This report will explain how these features are achieved, what the type of architecture has been used, and what kind of technologies have been used.

---

## Table of Contents

|       |   |    |
|-------|---|----|
| 1     | Introduction                                    | 4  |
| 2     | Project Specification                           | 5  |
| 2.1   | Problem Definition and Analysis                 | 5  |
| 2.2   | User Requirement (functional requirement)       | 6  |
| 2.3   | System Requirement (non-functional requirement) | 7  |
| 3     | Design  | 9  |
| 3.1   | High-level Architecture                         | 9  |
| 3.1.1 | MVC Architecture                                | 9  |
| 3.1.2 | Types of Architectures                          | 10 |
| 3.1.3 | Spring MVC                                      | 10 |
| 3.2   | Database and Data Structure Design              | 14 |
| 3.2.1 | MySQL   | 14 |
| 3.2.2 | Java Persistence API                            | 14 |
| 3.2.3 | Hibernate and MyBatis                           | 14 |
| 3.2.4 | Data Structure                                  | 15 |
| 3.3   | Interface Design                                | 19 |
| 3.3.1 | Bootstrap                                       | 19 |
| 3.3.2 | User Interface                                  | 19 |
| 4     | Project Management                              | 23 |
| 5     | Implementation                                  | 25 |
| 5.1   | Data Persistence (JPA)                          | 25 |
| 5.2   | Account type                                    | 26 |
| 5.3   | Approve proposal                                | 27 |
| 5.4   | Spring Security                                 | 29 |
| 5.4.1 | Spring Authentication                           | 29 |
| 5.4.2 | Spring Authorization                            | 30 |
| 5.5   | Validation and Testing                          | 31 |
| 5.5.1 | Validation                                      | 31 |
| 5.5.2 | Testing   | 32 |
| 6     | Evaluation                                      | 34 |
| 6.1   | Maintainability                                 | 34 |
| 6.2   | Performance                                     | 35 |
| 7     | Discussion                                      | 36 |
| 7.1   | Achievement                                     | 36 |
| 7.2   | Inadequacies                                    | 36 |
| 7.3   | Future works                                    | 37 |
| 8     | Summary and Conclusion                          | 38 |
| 9     | Reference                                       | 39 |
| 10    | Appendix  | 41 |

---

# 1 Introduction

During the coronavirus pandemic (COVID-19), the coronavirus pandemic has seriously affected students from completing their projects. Firstly, the University of Birmingham had to close due to the lockdown during the coronavirus pandemic, and the final year's students are compelled to do their graduation project at home. Compare with previous years, students have fewer or no opportunities to communicate and discuss their projects, ideas, and progress with other students. Also, this situation makes supervisors hard to manage and tracking students' progress. According to the research, there are many applications on the internet that allows users to create plans and track plans, but none of them allows huge group discussion and communication. Therefore, this project aims to develop a web application that can encourage communication between students, allow students to share their progress and idea, and help supervisors manage student's project progress such as proposal, feedback, and weekly objectives. In the system, students can share their project idea through create proposals, share progress through create weekly plans, and share resources and knowledge through post discussion in the forum. For supervisors, they can mark student proposals and track student weekly plans to track student's progress.

This project has two main challenges, the first challenge is to construct the relational database, especially, setting up the relationship between the user class. In this project the supervisor plays two roles, which the supervisor is also a second marker, this has made the project very challenging. The second challenge is to allow the supervisor and second marker to approve the proposal, although the logic behind is not very difficult the implementation of the logic is not easy. More detail of the challenges can be found in the implementation section.

This is a Java-based project that is implemented with the Spring boot framework and the MVC architecture. The front end is developed with the Bootstrap, JSP, JavaScript, jQuery, Ajax, and the Jakarta Standard Tag Library (JSTL). The database used the MySQL database. In the project, automated testing has been implemented on the database classes to ensure that the data are store correctly more information about the testing can be found in the testing section.

---

## 2 Project Specification

### 2.1 Problem Definition and Analysis

As mentioned in the introduction, the students from the University of Birmingham are compelled to study at home. For the final year students, they need to complete their final year project and dissertation at home. In the past, students can communicate, share resources, and discuss technical issues with their supervisor and classmates in the university, but right now, students are all over the world and this makes the communication between students become less and harder and also make supervisor hard to manage student's progress. According to the research, the university has a website called Canvas. This website allows students to find all project information released by the department, submit works, and consult questions about the project. Although this website has a discussion forum, it is not designed for academic discussion, it is more for announcing and consulting project information. So, there is no place for students to discuss their projects during the coronavirus pandemic.

On the other hand, there are many project management type applications online, such as the Pivotal Tracker. Pivotal Tracker is an online project management system. This system's design inspired the project a lot. For example, the UI design of the weekly plan page is inspired by the Pivotal Tracker. Appendix C is an example of the management system of Pivotal Tracker. In Pivotal Tracker, users can start creating plans in the icebox (in this project is called waiting box) and users can allocate the plans to the current iteration, which is the current week. In my application, students can do the same thing to create plans for the waiting box and allocate plans to the weeks they want. Pivotal Tracker has many more management features and is more powerful. However, the plans that users create are private and only can share between a small group of users, which is not suitable for the whole computer science department's students to use. Also, Pivotal Tracker does not provide any communication tools. Therefore, this project aims to design an application that is good at both tracking project progress and encourage discussions between a large group of users.

Reading other student's project proposal is a good way to understand their project and what they are trying to do. Also, students can often be inspired by other student's ideas, so the project decided to create features that allow students to create proposals and share with other students. However, consider the possibility of plagiarism, students can only view other student's proposals after they completed their proposals and the proposal is approved by their supervisor and the second marker. Also, the original plan of the project is to allow the entire university's students to use this application but considering that different major has different ways to do their project. For example, some major is dissertation based (writing only) and is project-based (do experiment), the proposal feature may not be suitable for all majors students to use, so this project eventually decided to narrow the target user group to computer science students. Reading other student's project plans is another good way to understand their project. Also, sharing progress between students can increase the student's motivation. For example, if a

---

student saw another student who is doing a similar project with him has achieved more features than him, that student is likely to be motivated and work harder to catch up. So this project also allows students to share their weekly plan and weekly progress with others. Also, supervisors and second markers can track their student's progress with this feature. To find a similar project manually is very time consuming, to help students save time from searching a similar project, the project implemented a tagging feature, which students can add tags to their project and another student can search the project according to these tags. Also, considering that some students may forget to add tags, so every student's project will be predefined with a tag related to their project type. Lastly, to encourage communication and discussion, a discussion forum has been implemented and the forums can also be filtered by tags.

## **2.2 User Requirement (functional requirement)**

### Login function

1. Student and supervisor can log in and sign out

### Proposal function

2. Students can create a proposal.
3. Students can edit proposals.
4. Students can submit a proposal.
5. Students can save the proposal.
6. Student can view the proposal
7. Students can view submitted proposal history.
8. Students can resubmit the proposal if the supervisor or second marker rejected the proposal.
9. Student can view comment when their proposal is rejected
10. Student can view other's student's proposal
11. Supervisor and second marker can view their supervised student's proposal
12. The supervisor and second marker can reject or accept their supervised student's proposal.
13. Supervisor and second marker can leave a comment when they reject the proposal
14. The supervisor and second marker can view all submitted proposal histories of their students.

### Plan function

15. Students can set number of weeks
16. Week tables can store weekly plans for that week.
17. Students can set a start date.
18. Students can create a plan in the waiting box.
19. Students can modify plans in both the waiting box and week tables.
20. Students can delete plans.
21. Students can delete multiple plans at once.
22. Students can change the status of the plan (Not started, start, finish).
23. Students can leave a repository link for tracking in each plan.
24. Students can move plans between different week table and waiting box.

- 
25. Students can move multiple plans at once.
  26. When plans move back to the waiting box, the status will change back to not start.
  27. Student can see the progress percentage from the progress bar on each week table
  28. Students can add notes in each plan.
  29. Students can view notes.
  30. Students can view another student's plan (Not include note and repository link).
  31. The supervisor and second marker can view their student's plans.

#### Tags function

32. Student can add tags to their project
33. Students can add multiple tags at once.
34. Student can delete tags
35. Student can delete multiple tags at once
36. Students, supervisors can search for other students according to the tags.

#### Discussion

37. Student can create a discussion forum
38. Students can leave discussions in the discussion forum.
39. Students can search for discussion by tags.

## **2.3 System Requirement (non-functional requirement)**

#### Login function

40. Validation on wrong password or wrong email address
41. Encrypt user login password with secure encryption
42. No repeated email in the database.

#### Proposal function

43. After students submit a proposal, the student cannot submit the proposal again, save only.
44. Only can view other student's proposals after the proposal is marked by both supervisor and second marker and the student has completed his proposal.
45. The proposal passed only when both supervisor and second marker accept the proposal.
46. The second marker can only mark the student's proposal after the supervisor accepted the proposal.

#### Plan function

47. Automatically generate week tables according to the number of weeks student set.
48. Students must enter a repository link when they change their status to finish.
49. After the start date is set, automatically calculate dates, and attach to each week's table.  
E.g. week number is 5 with a start date (22/08/2020 Saturday), create 5-week tables and the dates of the first-week table are (17/08/2020 Monday – 23/08/2020 Sunday), the second-week table is (24/08/2020 Monday – 30/08/2020 Sunday), etc.
50. The supervisor and second marker cannot modify the student's plan.

---

51. Students cannot modify another student's plan.

52. Students can modify the number of weeks.

1. If the new week number is larger than the old number add the week table.
2. If the new week number is smaller, remove the extra old week's table, if this week's table has plans inside, move all these plans to the waiting box.

Tags function

53. Tags are displayed in the order; the most used tags will be put on the top.

54. Tags usage will be calculated and display.

55. If tags already exist, it will only appear once.

56. All the tags student added will be shown on the people page' search function

Other function

57. **Authenticate** user identity

58. Give different types of users account with different levels of authority. E.g only allows students to access the student page, students can never access the supervisor's page.



---

## 3 Design

### 3.1 High-level Architecture

#### 3.1.1 MVC Architecture

As mentioned above, this project is implemented with the Spring Boot framework with MVC architecture. This section is going to provide some general ideas about what is the MVC architecture and how it works.

The MVC architecture is a very commonly used architecture design pattern in the software development industry. The MVC stands for Model, View, and Controller. This design pattern divides an application into the three layers listed above, and the purpose of this is to help the developer keep their code organized and easy to manage.

The Model layer is responsible for defining and holding all the raw data of the project. The model layers do not contain any business logic, only contains the pure application data. For example, this layer only declares data type, create constructors, setting relationship with other model classes, and defines getter and setter. The data in this layer can be retrieved and presented by the Controller layers.

The View layer is responsible for handling the user interface, which is the layer for the front-end design. This layer can receive data from the controller layers and perform them to the user's screen in a human-readable way or collect data and response from users and send them to the controllers for further processing. In this project, the View layer is composed of the JSP, JavaScript, and CSS.

The Controller layer is the center layer that responsible for connecting the model layers and the view layers and is also responsible for handling the business logic such as processing data and user requests. For example, In this project, if a user wants to log in, he will need to enter his email address and the password to the view layer, and the view layer will send the email address and password to the controller layer for validation. When the controller layer received the email and the password, the controller layer will acquire all the login data from the model layer (database). Then compare the email and password from the view layer with all the email and password from the model layer, if there is a matching case, the controller layer will send the response to the view layer and tell the user login success, if there is no match case, the controller layer will send the response to view layer to tell the user login fails.

---

### 3.1.2 Types of Architectures

In web application development, many different types of architecture can be used. The MVC architecture used in this project is very similar to the three-tier architecture. The three-tiers architecture composes of a Presentation layer (View), a Business layer (Controller), and a Data Access layer (Model). The difference between the Spring MVC and the three-tiers architecture is that the three-tier architecture is a liner architecture that the Presentation layer can only communicate with the Data Access layer through the Business layer, and the layers in the Spring MVC communicates through a technology called DispatcherServlet. This technology makes the Spring MVC more flexible to use compared to the other three-tiers architectures. The two-tiers architecture consists of two layers, the Client layers, and the Server layers. The advantage of this type of architecture is that the communication is straight forward because there are only two layers, which means the communication is fast and it will be easier to implement, maintain, and modify. However, the performance of this architecture will be degraded when the number of users increases (Admin,2013). Therefore, it is not suitable to use in this project. Compare to the three tiers architecture and the Spring MVC, the N-tiers architecture is a more advanced and powerful architecture that often uses in large companies. This is because N-tiers architecture has many layers, which enables concurrent development. However, this is an individual project and do not require such complex architecture. Therefore, this architecture is not suitable either.

### 3.1.3 Spring MVC

The Spring boot is a mainstream Java web application development framework and it is often used with the MVC architecture in the development. Spring Boot is a very powerful framework that has a very convenient and fully functional ecosystem such as the Spring JDBC, Spring Data, and Spring Security (JournalDev.com). Its ecosystem can help developers to reduce a considerable amount of development time and help developers to develop flexible and loosely coupled applications (Tutorialspoint.com). Also, as Spring is a very popular and it is an open-source framework, so there are a countless number of libraries, plugins, and APIs on the internet, which makes the development even more convenient. The Spring MVC is composed of DispatcherServlet, Handler Mapper, View Resolver, and the MVC. This section is going will introduce the uses of these components and describe the flow of the entire Spring MVC.

#### DispatcherServlet

The DispatcherServlet is the core component of Spring MVC, in which the entire framework is designed around the DispatcherServlet (Tutorialspoint.com). The DispatcherServlet is responsible for receiving HTTP requests from the users and activate the correct controller that can respond to the request and activate the corresponding view page to users.

## Handler Mapper

The Handler Mapper is responsible for mapping the HTTP request to the corresponding controller. Every time the DispatcherServlet received an HTTP request, it will first send the request to the Handler Mapper, the Handler Mapper will filter through all the controller classes and look for the controller that can respond to the request. When the controller is located, the Handler Mapper will tell the DispatcherServlet which controller to activate.

## View Resolver

View Resolver is a Spring technology. The purpose of this technology is to resolve and simply the view path in the controller to make the development process more convenient. For example, figure 1 is the configuration function of the View Resolver that has implemented in this project, which the location ("/WEB-INF/jsp") and document type (".jsp") of all the view documents are specified by the setPrefix and setSuffix method. After implemented the View Resolver, the developer can simply return the view name (student) like the example shown in figure 2 line 28, instead of returning the whole path (/WEB-INF/jsp/student.jsp). The View Resolver is responsible for processing and matching the view name returning from the controller's request mapping methods to the corresponding JSP document.

```
21 @Bean
22 public InternalResourceViewResolver viewResolver() {
23     InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
24     viewResolver.setViewClass(JstlView.class);
25     viewResolver.setPrefix("/WEB-INF/jsp/");
26     viewResolver.setSuffix(".jsp");
27     viewResolver.setOrder(2);
28     return viewResolver;
29 }
30
31 }
```

Figure 1. View Resolver Configuration

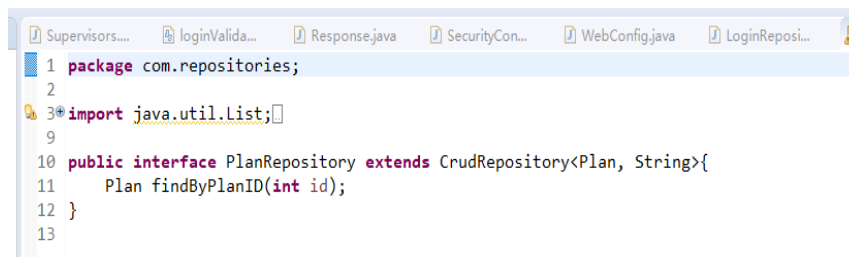


Figure 2. Project Controller example

---

## Repository

In Spring the repository is a mechanism that responsible for storing, retrieving, searching, deleting, and updating data. Spring provides an abstraction called the Spring data repository, in which this abstraction aims to implement the data access layers with less boilerplate code. For instance, the repository can be simply set up by calling the Crud Repository interface provided by Spring as shown in figure 3. The Crud Repository interface provides CRUD (Create, Read, Update, Delete) functionality for the model classes that are being managed (Docs.spring.io). With the Spring data repository, the controller classes can modify the model classes in the controller classes easily.

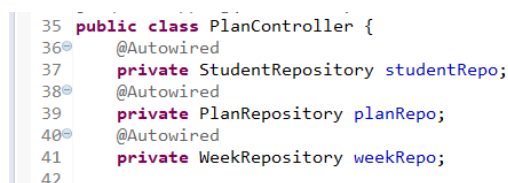
A screenshot of an IDE window showing a Java file named 'LoginReposi...'. The code defines a package 'com.repositories' and imports 'java.util.List'. It then defines a public interface 'PlanRepository' that extends 'CrudRepository<Plan, String>'. The interface has a single method 'Plan findByPlanID(int id);'.

```
1 package com.repositories;
2
3 import java.util.List;
4
5
6
7
8
9
10 public interface PlanRepository extends CrudRepository<Plan, String>{
11     Plan findByPlanID(int id);
12 }
13
```

Figure 3. Repository

## The Inverse of Control (IoC)

The Inverse of Control (IoC) is a principle that provides loose coupling in the Spring framework (Ashfaq, 2012). One of the most well-known features of the Inverse of Control is the dependency injection (a subset of IoC). The dependency injection is a process of injecting one object class to another object class. In Java development, if developers want to use a class A function in class B, the developers need to define an object of class A in class B. If it is a huge project, it will be difficult to maintain, so the dependency injection technique has been introduced. By using the dependency injection, class B can directly use the function in class A without defining an object of class A in class B. In Spring, the dependency can be injected with the Spring annotation “@Autowired”. For example, in this project, the repository interfaces are injected with the Spring annotation “@Autowired”, with these injections, the controller class can use the repository functions to search, retrieve, update, and delete data.

A screenshot of an IDE window showing a Java file named 'PlanController'. The code defines a public class 'PlanController' with four private fields: 'studentRepo', 'planRepo', and 'weekRepo'. Each field is annotated with '@Autowired'.

```
35 public class PlanController {
36     @Autowired
37     private StudentRepository studentRepo;
38     @Autowired
39     private PlanRepository planRepo;
40     @Autowired
41     private WeekRepository weekRepo;
42 }
```

Figure 4. IoC

## The Execution Flow of Spring Web MVC

This section is going to describe the entire execution process of the Spring Web MVC to provide a clearer picture of how the Spring Web MVC works. Firstly, when users interact with the application, HTTP requests will be sent out to the DispatcherServlet. For example, when a student trying to land on the student main page, an HTTP request <https://localhost8070/student> will be sent to the DispatcherServlet. However, the DispatcherServlet cannot identify the HTTP request, so it will send the request to the Handler Mappers that can recognize the request and find out the corresponding controller. The Handler Mapper will go through all the controller to check which controller's @RequestMapping value is equal to "/student" (the <https://localhost8070> part is fixed and will be ignored). Eventually, the Handler Mapper will locate the controller shown in figure 2 line 16. Once the Handler Mapper finds out the controller class, it will tell the DispatcherServlet which controller class should be invoked, and the DispatcherServlet will execute that controller class. After the controller layer finished processing the request (search the student data from the database and fetch to the model), it will return the processed data (model) with a view name ("student") to the DispatcherServlet, as shown in figure 2. Because the view name is simplified, so the DispatcherServlet cannot recognize the view name too. Therefore, the view name("student") will be sent to the View Resolver and the View Resolver will tell the DispatcherServlet the location of the view document (student.jsp). After the DispatcherServlet received the location, it will send the processed data (model) to the view layer (student.jsp file). The view layer will combine the processed data (model) with the font-end code and generate an updated web page. The DispatcherServlet will then display the updated web page (response) to the client-side (browser).

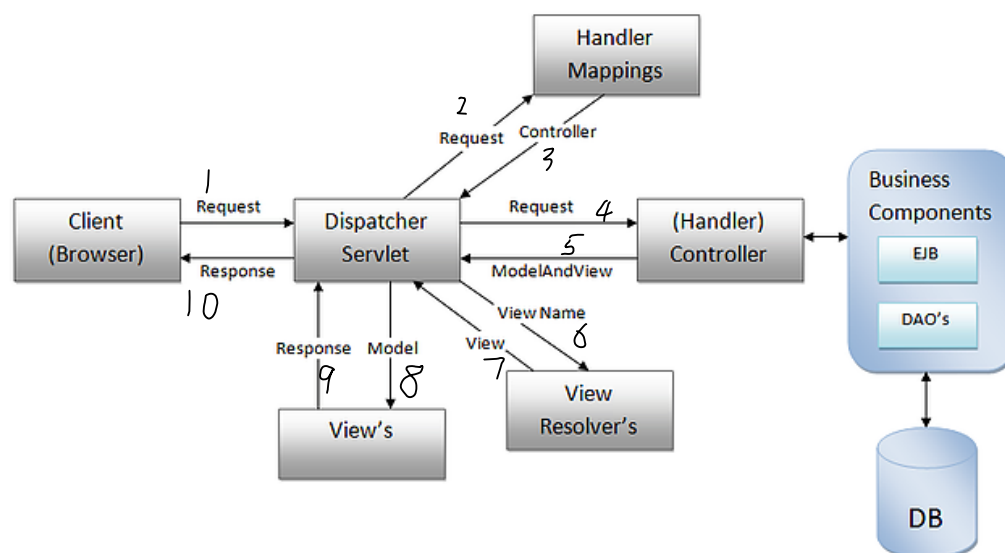


Figure 5. Spring Web MVC Flow image

Available at: <https://codedeal.wixsite.com/codedeal/single-post/Spring-MVC-Basic-Example-Without-Any-XML-Configuration>

---

## 3.2 Database and Data Structure Design

### 3.2.1 MySQL

The database used in this project is the MySQL database. This section will mention about why the project decided to use the MySQL database. Firstly, the MySQL database is one of the most popular open-source databases in the world and it is the leading choice database in the web development industry, which means supports such as how to set up a database connection can be easily found on the internet, and this can speed up the development process. Secondly, this is a relational database project, so it requires a relational database management system and the MySQL database also satisfies this requirement. Lastly, I have experience in setting up the MySQL database with the Spring project, so I finally decided to use the MySQL database instead of using something new that I am not familiar with and is less popular.

### 3.2.2 Java Persistence API

The Java Persistence API (JPA) is a Java programming interface specification that offers object-relational mapping (ORM) facilities to developers and supports developers to manage (store and retrieve) relational data (Fadatare, 2018). ORM is a tool that maps the application domain classes to the relational database tables. The JPA used in this project is supported by the Spring MVC, which is known as the Spring Data JPA (configured by the Hibernate). In the project, the Spring Data JPA provides many convenience annotations for managing and designing relational data. For example, it provides the `@Entity` that allows the project to persist the domain classes and the annotations such as the `@ManyToOne` that allow the project to design the relationship between objects. The use of JPA saves the project a huge amount of time from manually set up the ORM and the persistence. More detail about JPA can be found in the implementation section.

### 3.2.3 Hibernate and MyBatis

Hibernate is an open-source Java object-relational mapping framework (library) and it is one of the most mature and popular JPA implementations, so JPA and Hibernate are often conflated (Tyson,2019). The purpose of the Hibernate framework is to “simplify the development of Java application to interact with the database” by implementing the specification of JPA (Javatpoint.com).

MyBatis is an open-source persistence framework that uses for mapping the SQL statement to the Java objects. The main difference between the Hibernate and the MyBatis is the Hibernate is used for mapping Java classes to the database table and the MyBatis is mainly used for mapping the SQL statement to the Java method (Sharmi, 2020). In order words, the Hibernate is good at select queries(e.g. select all data related to the queried object) from the database

---

and the MyBatis is good at fetch queries, which just give you an answer instead of an entire object graph (fetch queries). In conclusion, hibernate is often used in object-centric types of projects (e.g. relational database project) and MyBatis is often used in data-centric types (query types) of projects such as online shop application. Therefore, this project decided to use Hibernate instead of MyBatis.

### **3.2.4 Data Structure**

Figure 8 is the relational database of the project. In the project, the Student class and the Supervisor class are the core of the entire structure, which are the two classes that responsible to define and store student's information such as the user's name, login email, password, etc. As the student class and the supervisor class are the core, this section is going to describe the structure begin with these two classes.

#### **Student Class – Project type Class**

When the student creates their proposal, they will need to decide which type of project they want to do (e.g. software engineering type). As each project can only have one type, so the relationship between the Student class and Project type class is Many To One in Student class and One to Many in Project type class, which many students can belong to the same project type, but one student can only have one project type.

#### **Student Class – Proposal Class**

At the beginning of the project, the relationship between the Student class and the Proposal class is a One to One relationship, which one student can only have one proposal and one proposal only belongs to one student, by using this relationship, every time the student update their proposal, the old version will be covered and cannot be retrieved anymore. At the middle stage of the development, my supervisor recommended me to implement a view history proposal feature, so I have changed the relationship to One to Many in Student class and Many to One in Proposal class so that one student can have many proposals under their account.

#### **Student Class – Week Class**

The Week class is used for storing each week's plans. In the application, students can set plans for different weeks, which means one student can have many week objects. Therefore, the relationship between the student class and the Week class is One to Many in the Student class and Many to One in Week class.

---

## Week Class – Plan Class

As the student can set many plans for each week but all the plans are unique, so it is impossible for one plan to appear in two week's table. In the project, students can move plans between different weeks but not appear in two different weeks at the same time, so the relationship between the Week class and the Plan class is also One to Many(Week class) and Many to One (Plan Class).

## Student class – Tag class

The Tag class is responsible for storing all the tags that students create. In the application, students allow adding as many tags to their projects as they want. The purpose of the tag function is to use the tag to search a project that has the same tag, which means one tag also belongs to many students so that when students selected a tag all other students who have the same tag will be filtered out. So, the relationship between the Student class and the Tag class is Many to Many, in which one student can have many tags and one tag can belong to many students.

## Student class – Forum class

In the application, students can create many forums to discuss their problems and the forum can only have one owner, so the relationship between the Student class and the Forum class is also One to Many and Many to One.

## Forum class – Content class

The content class is the class responsible for storing all the comments in the forum. Inside each forum, there can be many comments and all the comments only belong to the forum they in, so the relationship between the Forum class and the content class is One to Many and Many to One.

## Content class – Student Class

In a forum, there can have many comments from different users, but the forum can only store the name of the student who opens the forum, so to allow each comment to store the writer's name, I have to create a relationship between the Content class and the student class. In a forum, one student can leave many comments, so one student belongs to many comments, and each comment can only have one writer, so the relationship between the Content class and the Student class is Many to One and One to many.



## Student Class- Supervisor class

Both the Student class and Supervisor class are used to store the user's information. At the beginning of the project, it was decided to store both student and supervisor information together in one class, but as the supervisor is also the second marker of other students, so very difficult to construct a database that can store them together. Therefore, I have separated them into two classes. After setting up the Student class and Supervisor class, I have tried to set a Many to Many relationships, but by doing this I am not able to set multiple roles to the supervisor, then I tried to set two Many to One relationship in the Supervisor class and two One to Many in Student class like shown in figure 6 (line 37-38) and figure 7 (line 33-39), which one for mapping student to role supervisor and one for mapping student to role second marker, but this cause the duplication problem. Eventually, I have created a third class called SecondMarker to solve the problem which is used to map the relationship between student and supervisor and give different roles to the supervisor.

As a student need to set a relationship twice (one with supervisor, one with the second marker), so the relationship between the Student class and SecondMarker class is One to Many in Student class and Many to One in SecondMarker class.

Also, each supervisor will have many students, so a supervisor needs to set many relationships, hence, the relationship between the Supervisor class and SecondMarker class is also One to Many and Many to One.

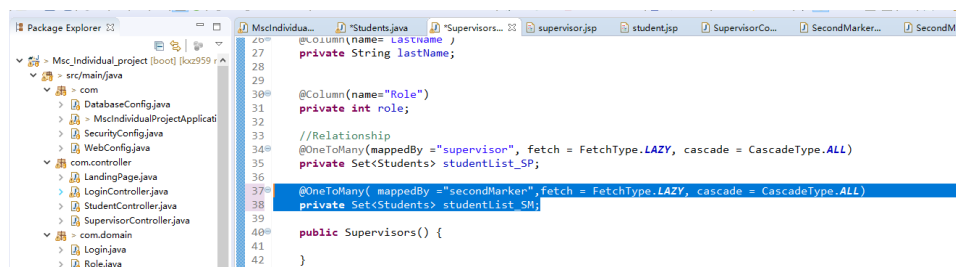


Figure 6. Old structure

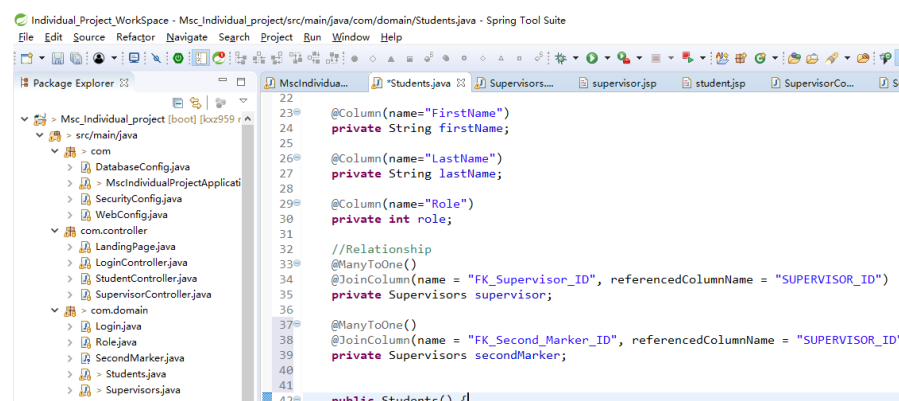


Figure 7 Old structure

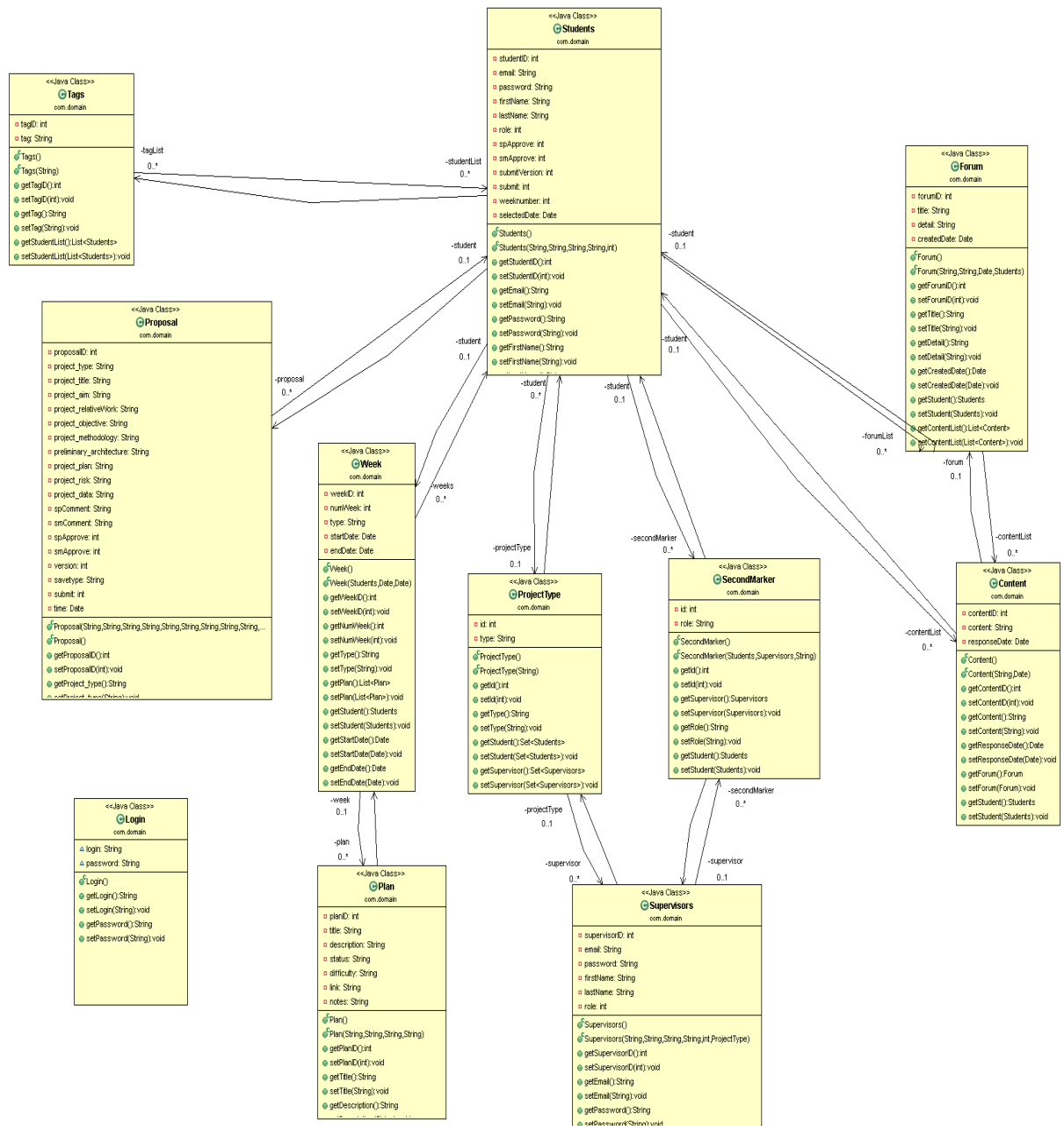


Figure 8

---

## 3.3 Interface Design

### 3.3.1 Bootstrap

Bootstrap is an open-source front-end (CSS) framework that provides many ready to use UI components and built-in CSS classes such as form, navigations, buttons, table, modal, etc. that would make the webpage development much easier. I chose to use Bootstrap for several reasons. Firstly, Bootstrap is famous for its grid system. The grid system divides the webpage into 12 columns and developers can simply assign the UI components e.g. button, forms into these 12 columns instead of writing a bunch of CSS code to adjust the positions of the components. Appendix B is an example of how the grid system is implemented. In the example, I have assigned 4 columns for each component by simply calling the “class=col-md-4” in the div tag without writing any CSS code. The grid system saved a great amount of time for the project from adjusting the UI. Secondly, Bootstrap is a very mature framework that has the largest ecosystem among front-end frameworks (Thakur, 2020). This means there are plenty of supports and examples on the internet which can give the project UI design plenty of inspiration. Also, the majority popular web browser supports the Bootstrap. Lastly, I have experience in using Bootstrap, so I eventually decided to use Bootstrap.

### 3.3.2 User Interface

In user interface design, the application must be easy to use. Although the target users are students and supervisors from the computer science department who are good at using this kind of application, it is still necessary to make the user interface as simple as possible. An easy to navigate interface would increase the user’s work efficiency and happy to use the application. The following part will reveal the user interface design of the project. We are going to begin with, the login page shown in figure 9. The login page is the first page that the users will interact with. As can be seen, the design of the Login page is very simple which the only login form for users to interact with. This makes users know what they need to do immediately when they are landing on this page. To let users know they did not enter the wrong website, I have put a large picture of the University of Birmingham beside the login form, so when users first land on this page, they will know they landed on the right page.



Figure 9 login page

After user login succeeded, they will be directed to their main page as shown in Figures 10 and 116. Figure 10 is the student main page and figure 11 is the supervisor's main page. Because the student's account has many features and each feature have many functions, So the features have been displayed in boxes with different color and icon. The idea of this design is to help students to recognize the features they want to use easier. For example, all the features related to the create project information such as create the proposal, create the weekly plan, and add the tag to the project, their boxes are blue, and the feature box of searching people with a similar project is in color green and the discussion forum feature box is in color yellow. So, when students want to edit project information, they know they should look for blue boxes. For the supervisor's main page, because the supervisor's features are all related to view student's project and mark student's project, so when the supervisor login, they will see the student table straight away, and in the table supervisor is given the option to view student's proposal and weekly plan. However, if there are more features implement to the supervisor account in the future, the supervisor's main page will also design in the same way as the student main page. Also, on the supervisor page, the supervisor can switch roles between supervisor and second marker simply through the side navigation bar.

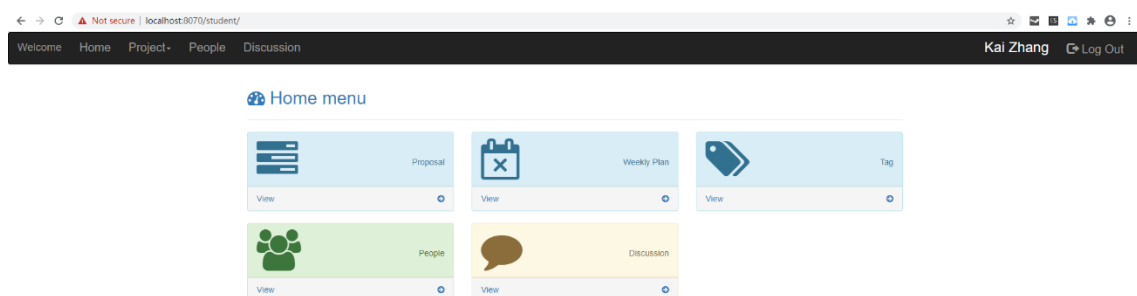


Figure 10 Student main page

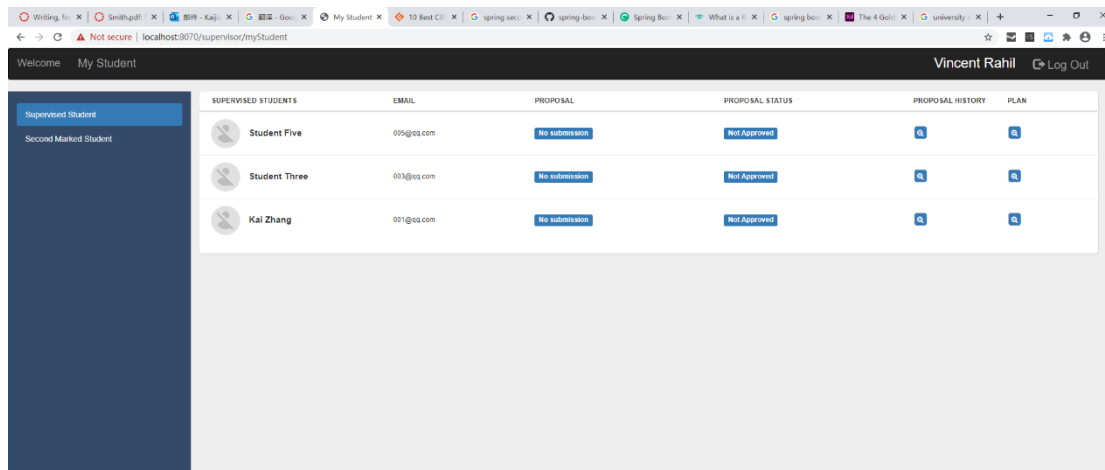


Figure 11 Supervisor main page

In the application, students can access a function in two ways. The student can access any function with the navigation bar on the top of the web page like all other web applications have or access through the boxes on the main page. For example, when students click the weekly plan box, they will be direct to the weekly plan page as shown in figure 12. And after landing on this page, the student will be provided with side navigation with more features that are related to the weekly plan. In the weekly plan function, the progress bar has been implemented, the use of a progress bar can help the student to calculate how many plans they have done instead of calculating all the completed plans by themselves. For the progress bar, there are three types of color, green for 100% completed, blue for 1%-99%, and orange for 0%. Different colors will help students to easily recognize whether they have achieved all the plans or not. Also, all the buttons in the application are the same as the buttons that are shown in figure 12, which the dangerous buttons such as the delete function's button are all filled with the color red and the adding function's buttons are in color green, and edit function's buttons are in color blue. The color of buttons is defined with the Bootstrap. By doing this, students are less likely to click the wrong button even they did not read the button label carefully.

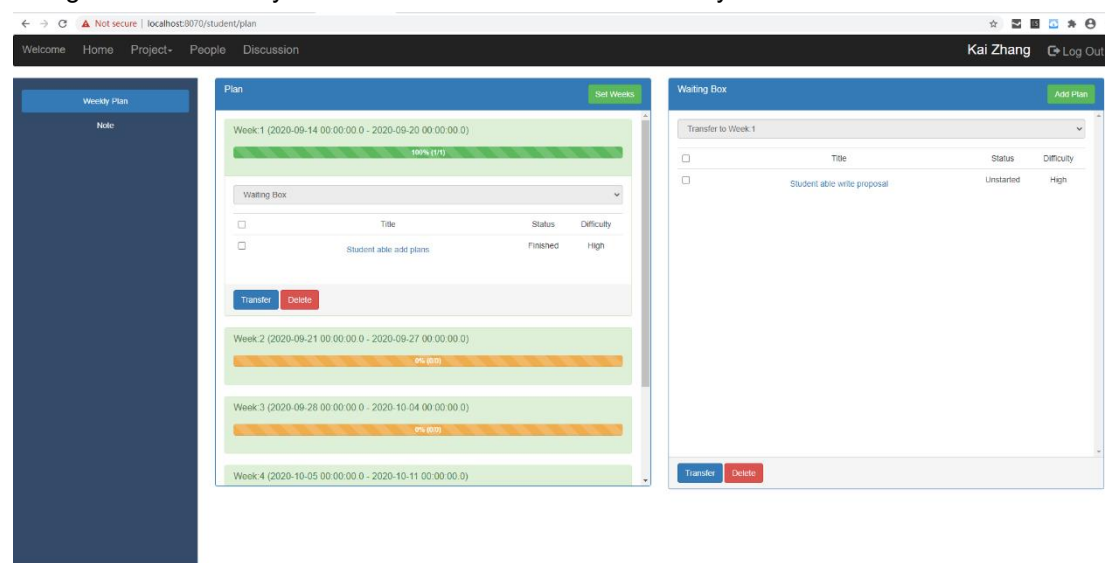


Figure 12 Example of user interface design

Lastly, in the application, I have implemented the modal on all the adding and edit functions (except adding and edit the proposal). The modal is a UI component provide by the Bootstrap that allows developers to create multiple web pages within one web page. Figure 13 is an example of the implementation of modal on the add plan function of a weekly plan. In figure 13, the modal component opened the adding plan page as a pop-up box instead of a direct student to a new page. The advantage of doing this is when users want to switch between two pages back and forth, the modal does not require refresh, but the open new page does, and sometimes loading a new page frequently will make users feel annoyed. So, using modal instead of the open new page could improve usability too.

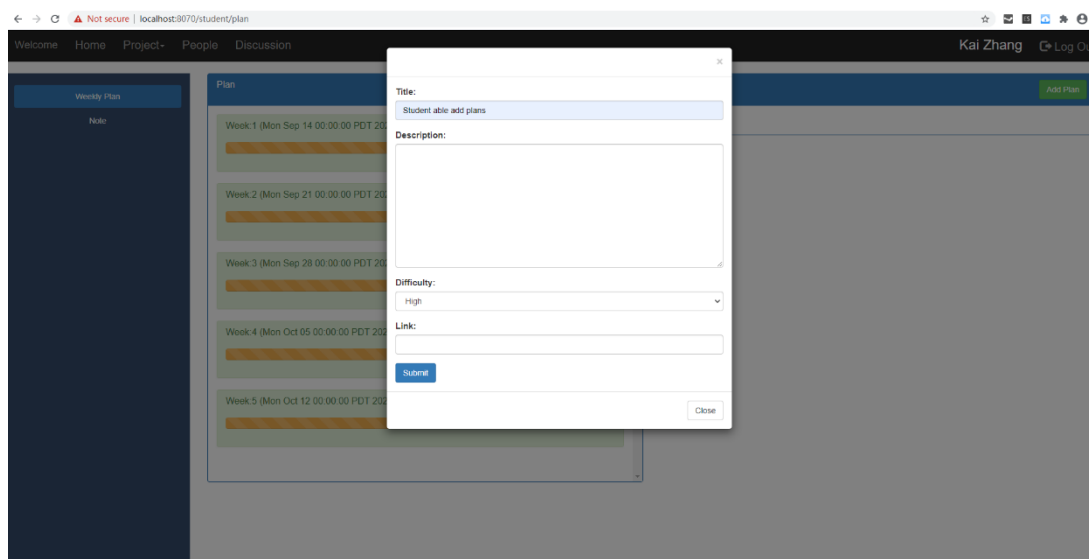


Figure 13. Example of user interface design

---

## 4 Project Management

In this project, I am responsible to complete the whole application individually, which means all the functions can only be done one by one, so I have used a Waterfall model as the project methodology rather than the Agile model. An Agile model is an approach that divides the development processes into many short life cycle stages that can be executed concurrently, which is more suitable to use in a team project that requires many people to work concurrently. The Waterfall model is also an approach that divides the development processes into many small phrases, but only one phrase can be executed at once. As this is an individual project, the linear and sequential Waterfall model will be more appropriate to use. In the project, the Gantt Chart has been used to list the tasks and milestones and reflected the project objectives/deliverables.

Figure 14 is the Gantt chart that I created at the beginning of the project. In the original plan:

- The first week (Gantt week 26, project week 3) plan is to set up the framework configuration and construct a relational database.
- In the 2nd and 3rd weeks (Gantt week 27-28, project week 4-5), the plan is to create a proposal page that allows students to enter their proposal and allow the supervisor to accept it.
- For week 4 (Gantt week 29, project week 6), the plan is to implement search similar project function.
- For week 5 (Gantt week 30, project week 7), the plan is to implement the chat/ discussion function.
- In week 6 and week 7 (Gantt week 31-32, project week 8-9), the plan is to implement the weekly plan function.
- In week 8 (Gantt week 33, project week 10), the plan is to complete the user interface design and the login function. If there are additional features and there is enough time, then implement additional features, if no additional features, then do some manual test.
- Week 9 (Gantt week 34, project week 11) is the demonstration week.

However, due to the version update of the Spring framework, the time required to spend on the framework configuration (set up dependencies) exceeded my expectations, and the plan of connecting database and construct relational databases has been delay for a few days. In the second week (project week 4), I started to connect the database and construct the database structure. When implementing the relational database between the Student class and the Supervisors class, the double identity of the Supervisor class cause data duplication in the database (more detail can be found in the implementation section), because the Supervisor class and the Student class is the core of the project, so this problem must be fixed, and I have spent one week on fixing this problem. When fixing this problem, I realized that the Login function must be implemented first because all the following functions require authorizations, so the Login function has been implemented in the second week (project week 4). After fixing the database problem and implementing the Login function, the project is far behind the plan (a week behind). When I started to do the proposal, weekly plan, and the search function, I

thought of many features that I didn't think of at the planning stage, such as the student cannot submit a new proposal during the supervisor and the second marker is marking (student can save update and submit after marking is completed), view proposal history function, help student manage notes and search by tags. After implemented these features, I realized there is not enough time for implementing both discussion function and chat function. In comparison, I think the discussion function is more useful in terms of encouraging communication than the chat function, so the chat function has been removed from the project. Also, there is not enough time for implementing tests on the whole project, so most of the project is test manually and the automated test are implemented to the database.

Furthermore, as mentioned in the database structure section the project has divided the user class into two classes, By doing this, the project needs to construct twice of the relationship in the student class and the supervisor class with other classes, which doubled the workload. Also, in coding aspect, every time supervisors want to view their student's proposal or plan, the controller will need to retrieved the student objects to search the related plans and proposals, instead of search the plan and proposal repository directly. As there are no time left for the project to construct the relationship twice, so the relationships are only set up in the student class, which means for the supervisors, they are not able to use some of the functionalities such as adding tags, create forum and write comment in the forum.

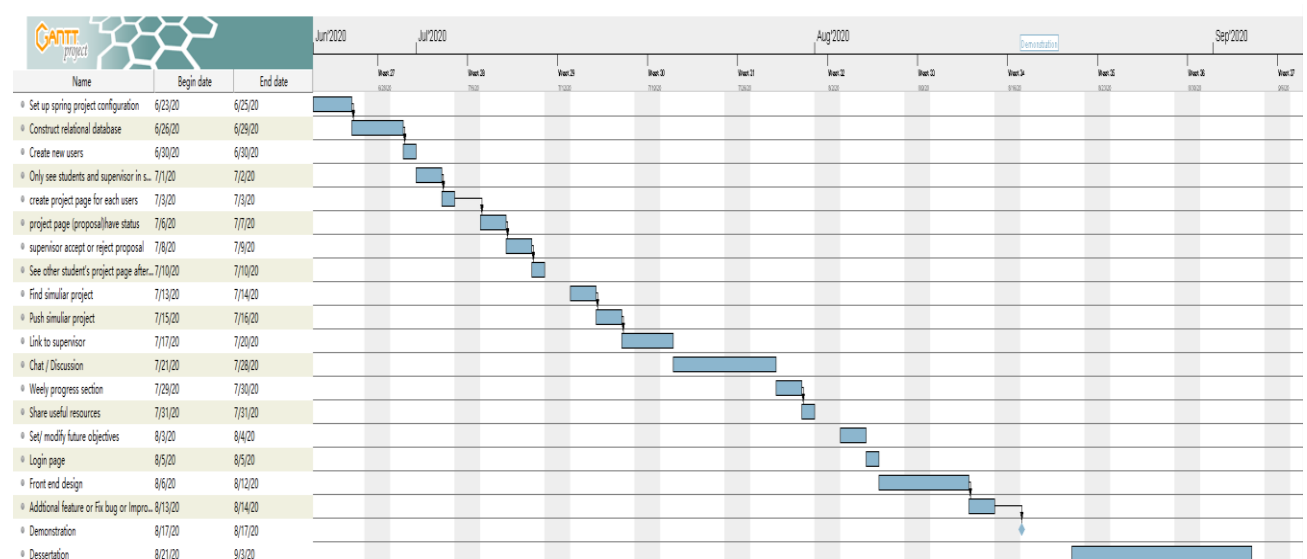


Figure 14 Gantt Chart



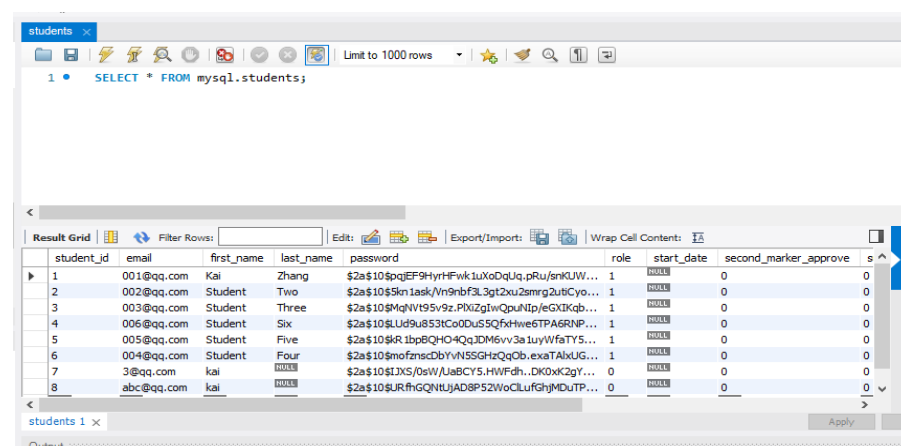
# 5 Implementation

## 5.1 Data Persistence (JPA)

This section is going to explain the implementation of the Java Persistence API (JPA). As mentioned in the design section, the JPA implementation is very simple. Figure 15 is an example of the implementation of the `@Entity` JPA in the project. As shown in the example, the project just needs to import the persistency API and add the annotation `@Entity` on top of the class name (line 14), the entire class will be persisted. The persisted class will be stored displayed in the database. The `@Table`, `@Id`, `@GeneratedValue`, and `@Column` are also the annotation provided by the JPA, the `Table` annotation defines the name of the database table, the `Id` annotation sets the "studentID" variable as the primary key of the table, the `GeneratedValue` annotation generates new id to new object created automatically, and the `Column` annotation defines the column in the table as shown in figure 16.

```
1 package com.domain;
2
3 import java.util.ArrayList;
4 import java.util.Date;
5 import java.util.List;
6 import java.util.Set;
7
8 import javax.persistence.*;
9 import javax.validation.constraints.NotNull;
10
11 import org.hibernate.validator.constraints.NotEmpty;
12 import org.springframework.format.annotation.DateTimeFormat;
13
14 @Entity
15 @Table(name="students")
16 public class Students {
17     @Id
18     @GeneratedValue(strategy=GenerationType.IDENTITY)
19     @Column(name="STUDENT_ID", nullable=false)
20     private int studentID=-1;
21
22     @Column(name="Email", unique=true, nullable=false)
23     private String email;
24
25     @Column(name="Password", nullable=false)
26     private String password;
27
28     @Column(name="FirstName")
29     private String firstName;
```

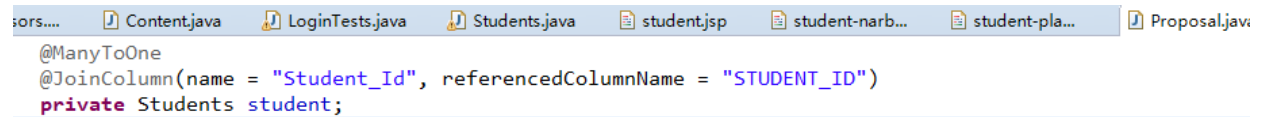
Figure 15. Example of JPA implementation



| student_id | email      | first_name | last_name | password                                    | role | start_date | second_marker_approve |
|------------|------------|------------|-----------|---|------|------------|-----------------------|
| 1          | 001@qq.com | Kai        | Zhang     | \$2a\$10\$pgEF9HhFwk1uXoDqUq.pRu/snkUW...   | 1    | NULL       | 0                     |
| 2          | 002@qq.com | Student    | Two       | \$2a\$10\$9m1aak/n9nbf3L3gt2xu2mrg2u8Cyo... | 1    | NULL       | 0                     |
| 3          | 003@qq.com | Student    | Three     | \$2a\$10\$MqVY95v9z.PNzgtwQpuNp/eGXKqb...   | 1    | NULL       | 0                     |
| 4          | 006@qq.com | Student    | Six       | \$2a\$10\$LU9u853ICoDu5SQf8Hwe6TPA6RNP...   | 1    | NULL       | 0                     |
| 5          | 005@qq.com | Student    | Five      | \$2a\$10\$R1bp8QH04QqJDM6vv3e1uyWf6TYS...   | 1    | NULL       | 0                     |
| 6          | 004@qq.com | Student    | Four      | \$2a\$10\$nofmzcdbyVNISSGhzQCb.exaTAkUg...  | 1    | NULL       | 0                     |
| 7          | 3@qq.com   | kai        | NULL      | \$2a\$10\$IXS/0eW/UaBCYS.HWfFah..DK0xk2g... | 0    | NULL       | 0                     |
| 8          | abc@qq.com | kai        | NULL      | \$2a\$10\$URfHGGNlUAD8P52WoCLufGhYMDuTP...  | 0    | NULL       | 0                     |

Figure 16 Database example

On the other hand, the JPA also provides annotations that support relationship mapping. For example, the `@ManyToOne` annotation in figure 17 defines that the Proposal class is a Many to One relationship to the Student class, and the `@JoinColumn` takes the student primary key from the Student class as the foreign key in the Proposal class table. To respond to the relationship, the Student class just needs to add the `@OneToMany` annotation.

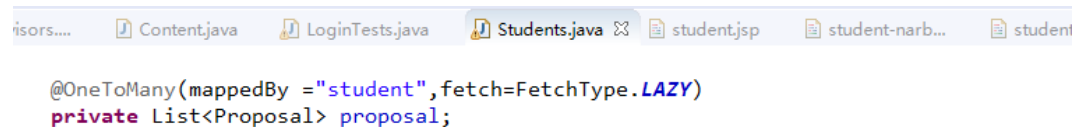


```

@ManyToOne
@JoinColumn(name = "Student_Id", referencedColumnName = "STUDENT_ID")
private Students student;

```

Figure 17 Proposal class



```

@OneToMany(mappedBy = "student", fetch=FetchType.LAZY)
private List<Proposal> proposal;

```

Figure 18 Student Class

## 5.2 Account type

Implement one type of account plays two roles is one of the most challenging parts of the project. As mentioned in the data structure part, several ways have been tried to solve the duplicate entry problem in the database. In the project, students and supervisors need to use email to log in their accounts, which means duplicate email addresses will cause exceptions during login. Also, the student's class stores many status information such as proposal submission status, if there are duplicate account, the status data may be updated to the wrong one, so sometimes even the students submitted their proposal, it will show that is not submitted, furthermore, the table generated on the web page will show duplication. To solve the duplication problem, I have tried many ways to construct the database. In a relational database, when storing a parent object, the child object under this parent will also be stored into the database, in the project, the supervisor object was the parent and the student object is the child. For example, student A is supervised by supervisor B and second mark by supervisor C, when the database store supervisor B, student A will be store to the database too. At this time, the database continues to store supervisor C, student A will be store one more time which causes duplication. To solve the duplication, I have tried to use Many to Many relationships because Many to Many relationship structures will not cause duplication. However, Many to Many relationships cannot set two roles to one supervisor at the same time (new role will replace the old role, so supervisors can only either be a supervisor or a second marker). Through testing many different ways, I found a way to keep the Many to Many relationships and also able to add multiple roles to one supervisor, which I created a third table as shown in figure 19, figure 20, and figure 21 highlighted part.

```

1 package com.domain;
2
3 import java.util.Set;
4
5 @Entity
6 @Table(name="Second_Marker")
7 public class SecondMarker {
8     @Id
9     @GeneratedValue(strategy=GenerationType.IDENTITY)
10    @Column(name="Pair_ID", nullable=false)
11    private int id;
12
13    @ManyToOne(fetch=FetchType.LAZY, cascade = CascadeType.PERSIST)
14    @JoinColumn(name = "Supervisor_Id")
15    private Supervisors supervisor;
16
17    @ManyToOne(fetch=FetchType.LAZY, cascade = CascadeType.PERSIST)
18    @JoinColumn(name = "Student_Id")
19    private Students student;
20
21 }

```

Figure 19 Second Marker class (Class for setting role and relationship)

```

1 package com.domain;
2
3 import javax.persistence.*;
4
5 @Entity
6 @Table(name="supervisors")
7 public class Supervisors {
8     @Id
9     @GeneratedValue(strategy=GenerationType.IDENTITY)
10    @Column(name="SUPERVISOR_ID", nullable=false)
11    private int supervisorID=1;
12
13    @Column(name="Email", unique=true, nullable=false)
14    private String email;
15
16    @Column(name="Password", nullable=false)
17    private String password;
18
19    @Column(name="FirstName")
20    private String firstName;
21
22    @Column(name="LastName")
23    private String lastName;
24
25    @Column(name="Role")
26    private int role;
27
28    //Relationship
29    @OneToMany(mappedBy = "supervisor", fetch=FetchType.LAZY, cascade = CascadeType.PERSIST)
30    private Set<SecondMarker> secondMarkers;
31
32 }

```

Figure 20 Supervisor class

```

1 package com.domain;
2
3 import javax.persistence.*;
4
5 @Entity
6 @Table(name="Students")
7 public class Students {
8     @Id
9     @GeneratedValue(strategy=GenerationType.IDENTITY)
10    @Column(name="STUDENT_ID", nullable=false)
11    private int studentID=1;
12
13    @Column(name="Email", unique=true, nullable=false)
14    private String email;
15
16    @Column(name="Password", nullable=false)
17    private String password;
18
19    @Column(name="FirstName")
20    private String firstName;
21
22    @Column(name="LastName")
23    private String lastName;
24
25    @Column(name="Role")
26    private int role;
27
28    @Column(name="Supervisor_Approve")
29    private int spApprove;
30
31    @Column(name="SecondMarker_Approve")
32    private int smApprove;
33
34    private int submitVersion;
35    private int submit;
36    private int weeknumber;
37
38    @DateTimeFormat(pattern = "yyyy-MM-dd")
39    @Column(name="Start_Date")
40    private Date selectedDate;
41
42    //Relationship
43    @OneToMany(mappedBy = "student", fetch=FetchType.LAZY, cascade = CascadeType.PERSIST)
44    private Set<SecondMarker> secondMarkers;
45
46 }

```

Figure 21 Student class

## 5.3 Approve proposal

Approve the proposal is one of the challenges of the project. The reason I think it is challenging is that there are many situations in this functionality. In the beginning, I have not constructed the tree diagram shown in figure 22 and tried to implement the logic straight away, and the result of this is the logic is very messy, so I decided to draw a tree diagram. In the project, there

are three states for each role. For example, a supervisor account will have three states for role supervisors and three states for the role second marker. The state is default as null when the application first time execute. When a student submits a proposal, their supervisor's state and second marker's (supervisor and second marker are a different person) state will be set to 0 in the database. In the application, the supervisor must mark before the second marker, so if the supervisor state is equal to 0 second marker cannot view and marker proposal. If the supervisor accepted the proposal, the supervisor state will change to 1 and the second marker will then be able to view and start marking. However, if the supervisor declines the proposal, then the supervisor state will change to 2, when the supervisor state equals 2, the second marker will be notified about the decline and still cannot view the proposal and the student will be required to resubmit a new proposal and the cycle will start again from the first stage in figure 22.

After the supervisor accepted the proposal, the supervisor is not able to mark anymore and the second marker can start to mark the proposal. When the second marker is marking the second marker state is also 0. If the second marker declines the proposal (second marker state becomes 2), then the student needs to resubmit the proposal and the cycle will start all over again from the first stage in figure 22. If the second marker accepted the proposal, then the proposal will be passed.

In the student, supervisor, and second marker page, all the status will be updated automatically, for example, when the supervisor state is 1 and the second marker state is 2, the student side will be notified that the supervisor approved and the second marker declined and required resubmission. The supervisor side supervisor will be notified that the second marker rejects the proposal and is waiting for the student to submit a new proposal. In the second marker side, the second marker will be notified that the project is decline waiting for a new submission. The logic behind this function is not very difficult but the combination of states and different user types are many, so I think this is a challenge to the project.

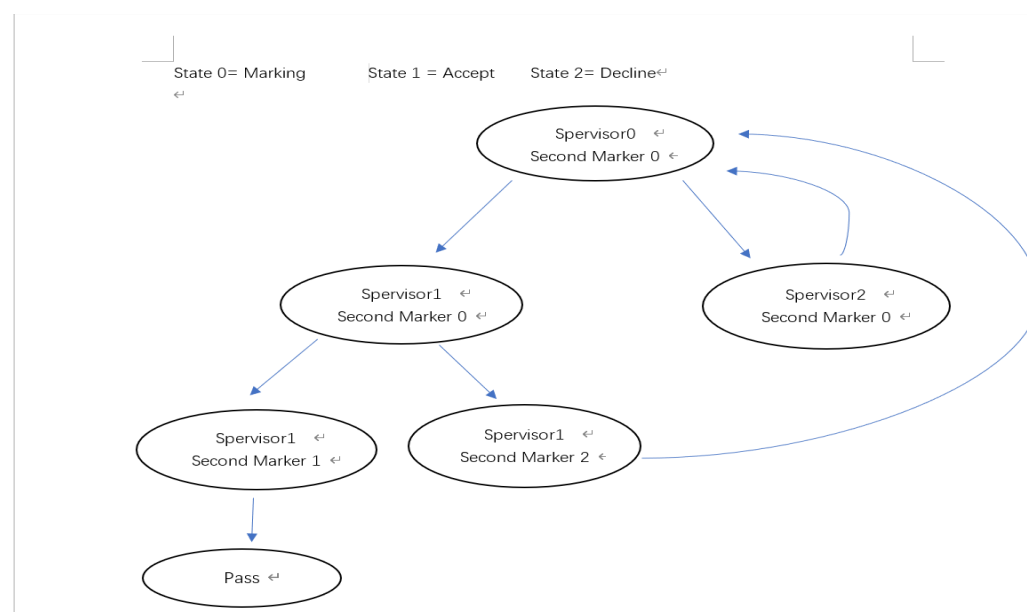


Figure 22. Tree diagram of approve proposal function.

---

## 5.4 Spring Security

Spring Security is a Java framework that provides highly customizable authentication, authorization, and other security features to web applications. This project has implemented both authentication and authorization provided by Spring security. This section will explain how the Spring Security implements in this project.

### 5.4.1 Spring Authentication

Authentication is the process of verifying the user's identity, so it is significant in any applications that involve a login function. The project has used the authentication configuration provided by Spring Security. When a user enters their email and password in the login page (URL: "/login/"), the email and password will be sent to the usernameParameter and the passwordParameter is shown figure 23 (line 65,66) to authenticate user identity, if the authentication passed, the system will execute Success URL "/success-login". The "/success-login" URL is responsible for sending users to the landing page they suppose to go. For example, the student will land on the student main page, and the supervisor will land on the supervisor's main page. The Code of user allocation is shown in figure 24 (line 100-105). If the login failed, the "access-denied" URL will be executed, which also can be seen in figure 24 (line 111-113), the "access-denied" URL will let users stay on the login page.

```
58         .anyRequest().authenticated()
59
60     .and()
61     .formLogin()
62     .failureForwardUrl("/error")
63     .loginPage("/login/")
64     .usernameParameter("email")
65     .passwordParameter("password")
66     .defaultSuccessUrl("/success-login", true)
67     .loginProcessingUrl("/login")
68     .failureUrl("/error")
69     .permitAll()
70
71     .and()
72     .logout()
73     .invalidateHttpSession(true)
74     .permitAll()
75
76     .and()
77     .rememberMe()
78
79     .and()
80     .exceptionHandling().accessDeniedPage("/access-denied");
81 }
82 }
```

Figure 23. Authentication configuration

```

90 @RequestMapping(value = "/success-login", method = RequestMethod.GET)
91 public String authenticate() {
92
93     User authUser = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
94     authUser.getAuthorities().stream().forEach(c -> System.out.println (c));
95
96     System.out.println("logging in as " + authUser.getUsername());
97
98     String view = "welcome";
99
100     if(studentRepo.findByEmail(authUser.getUsername()) !=null) {
101         view = "redirect:/student/";
102     }
103     if(supervisorRepo.findByEmail(authUser.getUsername()) !=null) {
104         view = "redirect:/supervisor/myStudent";
105     }
106
107     return view;
108 }
109
110
111 @RequestMapping("/access-denied")
112 public String accessDenied() {
113     return "login";
114 }
115
116
117 }

```

Figure 24

## 5.4.2 Spring Authorization

Authorization is the process of giving different types of users with different levels of accessibility. For example, the supervisor cannot visit the student page to modify the student's proposal and plan and students cannot visit the supervisor page to approve their proposal. The Spring Security also provides configurations for authorization. Figure 25 is the authorization configuration of the project. The URL with permitAll means all types of users can access these URLs. The URL with hasRole means only the role of user specified in the hasRole can access the URL. For instance, in figure 25 (line 53,54), only allow accounts with the role "SUPERVISOR" to access the URL with the prefix "/supervisor" and students can only access the URL with the prefix "/student". Therefore, when designing the functions, all the student function's URLs are beginning with "/student" e.g. "/student/plan" is for plan functions and "/student/proposal" is for proposal function.

```

36
37 protected void configure(HttpSecurity http) throws Exception {
38     http
39         // AUTHENTICATION
40         .csrf().disable() // if csrf is enabled, /logout must be a POST
41
42         .authorizeRequests()
43         .antMatchers("/").permitAll()
44         .antMatchers("/welcome").permitAll()
45         .antMatchers("/resources/**").permitAll()
46         .antMatchers("/jsp/**").permitAll()
47         .antMatchers("/js/**").permitAll()
48         .antMatchers("/img/**").permitAll()
49         .antMatchers("/loginValidate").permitAll()
50         // AUTHENTICATION AND AUTHORISATION
51         .and()
52         .authorizeRequests()
53         .antMatchers("/supervisor/").hasRole("SUPERVISOR")
54         .antMatchers("/supervisor/**").hasRole("SUPERVISOR")
55         .antMatchers("/student/").hasRole("STUDENT")
56         .antMatchers("/student/**").hasRole("STUDENT")
57

```

Figure 25. Authorization configuration

## 5.5 Validation and Testing

### 5.5.1 Validation

In this project, validations have been implemented in the Login function and the other function that requires filling in forms such as edit proposal and add plans functions. For example, the login function is validated with Ajax. Ajax is a client-side technology that allows the developer to create asynchronous web applications. By using the Ajax, the validate message can be displayed without refresh the web page. When users submit their email and password in the login page, the “loginAjax” function shown in figure 26 will be executed, and the password and email will be sent to the request mapping URL “/loginValidate” (line 11). Then the controller will execute the corresponding request mapping method shown in figure 27 (line 43). In the controller, the password from the client-side will be encrypted and then compared with the password and email address in the database. If the validation pass, the controller would return “Success” to the Ajax function, if the validation failed, the controller will return “Failed” and the result value (e.g. wrong password result =password is incorrect). If the Ajax receives Success, the “send” in figure 26 line 15 will become true, and the user will be direct to the next page. If the Ajax receives Failed, Ajax will run the else statement in figure 26 (line 18-27) and the result value return from the controller will be displayed to the user with the line of code “\$('#error').html(errorInfo)”. Because is using ajax, so updates will not refresh the web page.

```
1 function loginAjax() {
2   var email = $('#email').val();
3   var password = $('#password').val();
4   console.log(email);
5   var send = false;
6   var token = $("meta[name='_csrf']").attr("content");
7   var header = $("meta[name='_csrf_header']").attr("content");
8   $.ajax({
9     async: false,
10    type: "POST",
11    url: "/loginValidate",
12    data: "&email="+ email + "&password=" + password,
13    success: function(response){
14      if(response.status == "SUCCESS"){
15        send = true;
16      }
17    }
18    else{console.log(response.result);
19      errorInfo = "";
20      for(i=0 ; i < response.result.length ; i++){
21        errorInfo += "<br>" + (i + 1) + ". " + response.result[i].code;
22      }
23
24      $('#error').html(errorInfo);
25      $('#info').hide('slow');
26      $('#error').show('slow');
27    }
28  })
29  return send;
30 }
31 }
```

Figure 26 Ajax implementation

```

43* @RequestMapping(value="/loginValidate", method = RequestMethod.POST)
44 public @ResponseBody Response LoginValidate(@ModelAttribute("userLogin")Dto dto, BindingResult result, Model m
45 Response res = new Response();
46
47 BCryptPasswordEncoder pe = new BCryptPasswordEncoder();
48
49 Students student = studentRepo.findByEmail(dto.getEmail());
50 Supervisors supervisor= supervisorRepo.findByEmail(dto.getEmail());
51
52 if(student != null || supervisor != null) {
53     if(student != null) {
54         if(!pe.matches(dto.getPassword(), student.getPassword()) && !dto.getPassword().isEmpty()) {
55             result.reject("password is incorrect");
56         }
57         if(dto.getPassword().isEmpty()) {
58             result.reject("Please enter your password");
59         }
60     }
61     if(supervisor != null) {
62         if(!pe.matches(dto.getPassword(), supervisor.getPassword()) && !dto.getPassword().isEmpty()) {
63             result.reject("password is incorrect");
64         }
65         if(dto.getPassword().isEmpty()) {
66             result.reject("Please enter your password");
67         }
68     }
69 }
70 else {
71     result.reject("Invalid email.");
72 }
73 if(dto.getEmail().isEmpty()) {
74     result.reject("Please enter an email");
75 }
76
77
78
79 if(!result.hasErrors()) {
80     res.setStatus("SUCCESS");
81 }
82 else{
83     res.setStatus("FAIL");
84     res.setResult(result.getAllErrors());
85 }
86
87 return res;
88 }

```

Figure 27

## 5.5.2 Testing

As mentioned in the project management section, the project progress has been delayed, so there was not much time left for the project to implement testing for the whole project, so I have done some simple unit tests on the database and the rest of the project were tested manually. As mentioned above, for the student database and the supervisor database, there cannot have duplicate users, so I decided to implement tests on the database. Firstly, the testing tools that I use is called JUnit, which is a very popular testing framework in the Spring. Figure 28 is an example of testing on the supervisor database, in the example, I mocked the process of creating a new supervisor and store it to the database if the database store the user twice, then the test will return an exception, if the database only stores the user once, then the test will past. Also, if the database is not storing objects or lost connection, it will return exceptions too. I have implemented the test on the major databases.



---

```
@Test
public void SupervisorRepoTest() throws Exception{
    // given
    int pre =(int) superRepo.count() ;
    BCryptPasswordEncoder bcpe = new BCryptPasswordEncoder();
    Supervisors s = new Supervisors();
    String password ="Password";
    s.setFirstName("kai");
    s.setEmail("4@qq.com");
    s.setPassword(bcpe.encode(password));
    superRepo.save(s);

    // when
    Supervisors found = superRepo.findByEmail(s.getEmail());

    // then
    assertThat(found.getEmail())
        .isEqualTo(s.getEmail());
    assertThat(superRepo.count()-pre)
        .isEqualTo(1);
}

@Test
```

Figure 28

---

# 6 Evaluation

## 6.1 Maintainability

As mentioned in the design section, this project is implemented with the Spring Boot framework with MVC architecture. The greatest advantage of using MVC architecture is that the project business logic, data, and view are implemented separately, which makes the code of the project more readable and reusable. For example, all the functionality codes are in the controller, so when I want to modify some functions, I can easily find the function in the controller classes. If I want to change the user interface, I can easily find the view document in the view package. Furthermore, when I implement different types of functions, I will generate a new controller for it. For example, as can be seen in figure 29, when I design the proposal functions, I have created a proposal controller which only use to handle proposal functionalities. Also, I have a planning controller for plan functionalities, a login controller for login functions, etc. By doing this I can easily find the controller I want and makes the project easy to maintain. Furthermore, to further improve the code readability and reusability, I have written the commonly used codes in separate functions, so instead of copy and paste the common codes, I can simply call these functions in other functions. This improves the code readability and reusability (Example in figure 30). However, due to time reason, I did not put these common functions in separate classes, which this sometimes make it difficult to find these functions and this would be something that can be improved next time. Overall, maintainability is good.

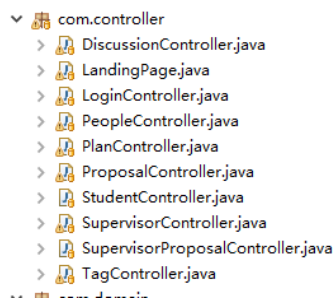


Figure 29. Controller classes of the project

```

279     }
280     //-----Shared Method-----
281     public List<Week> sortWeekList(List<Week> list) {
282         List<Week> weekList = new ArrayList<>();
283         for(Week w:list) {
284             if(w.getType()!=null) {
285                 weekList.add(w);
286             }
287         }
288         Collections.sort(weekList, new Comparator<Week>() {
289             @Override
290             public int compare(Week w1, Week w2) {
291                 // TODO Auto-generated method stub
292                 return Integer.compare(w1.getWeekID(), w2.getWeekID());
293             }
294         });
295         return weekList;
296     }
297 }
298
299 //reference: w3resource.com/java-exercises/datetime/java-datetime-exercise-9.php
300 public List<Date> start_end_date(Date selectedDate) {
301     List<Date> dateList = new ArrayList<>();
302     Calendar cal = Calendar.getInstance();
303     cal.setTime(selectedDate);
304
305     // Set the calendar to monday of the current week
306     cal.setFirstDayOfWeek(Calendar.MONDAY);
307     cal.set(Calendar.DAY_OF_WEEK, Calendar.MONDAY);
308
309     // Print dates of the current week starting on Monday

```

Figure 30. Share method.

## 6.2 Performance

Before evaluating the performance of the project, it is important to define what is meant by performance in web development. In web development, the performance refers to the speed of loading a web page and display results to the user's web browser. In the project, during the manual testing, the speed of loading all the web pages are fast, which all the web pages can be loaded in an instant. However, due to the condition constraints, the project is not able to ask many people to test the webpage because this will require to upload the project to the server. Also, due to the time constraints, the project is not able to implement a performance test on the maximum usage, so this project cannot guarantee that the application can still load fast when the number of users increases, but there is certainly a limit. Further, the application also works well under poor internet conditions. The application has run under the mobile cellular network (3G) to test the speed of loading web page. In the test, the speed of loading a new web page is also in an instant, so the internet has very little influence on the performance for the application. To conclude, as far as manual testing is concerned, the performance of the application is good.

---

# 7 Discussion

## 7.1 Achievement

This project has achieved all the features list in the requirements section. This section will summarize the core features that have been achieved. The following are the core feature achieved in the project.

For student:

- Log in/ Log out
- Create, edit, submit and view proposals.
- View proposal history
- Create, edit, share, delete and view weekly plan
- View other's proposal and weekly plan
- Search for a similar project with tag
- Add and delete tags
- Generate note from plans
- Create forums and leave comments in forums.

For Supervisor (second marker)

- Read, approve and decline the student proposal
- View student's proposal history
- View the student's weekly plan.
- Search not supervised students with tags.

## 7.2 Inadequacies

This section will explain the inadequacies of the project. Firstly, as the user account has been divided into two types of account this makes the project need to spend twice the amount of time on setting up data relationships. This slows down the progress and made the supervisor account has a very limited amount of features, which the supervisor can only view and mark the proposal and track the student's weekly plan. As there is not enough time, the project has only set up the relationship between the student class and the forum class, therefore the only student can use the forum. Also, in the forum, students can leave comments but not leaving comments under another comment, to leave a comment under another comment it requires adding more relationships which are a time-consuming process. Further, the application does not implement the modification function for the forum feature, so students are not able to modify the forum they post if they find mistakes. Moreover, the proposal does not support file upload function, so students are compelled to create proposals on the application, which thus limits the degree of freedom of the application. Lastly, supervisors can view student's weekly plan but they are not able to accept plans or give suggestions to the students. This weakened the communication between students and supervisors.

---

On the coding aspect, the project did not consider too much about the maintainability at the beginning of the project. Although the common methods are all coded separately from the function, they are still in the same file, which sometimes makes the project hard to maintain.

## 7.3 Future works

All the core features of the project have been implemented to meet the objectives. However, this application is still far from complete. If had had more time, I would implement the following functionalities.

- Able to reset password
- Supervisor able to leave a comment in student' forum
- Students can submit a dissertation or proposal through the file upload function.
- Students and supervisors can download the uploaded file.
- Supervisors can add tags.
- Student can search supervisor with tags too.
- Supervisors can also approve the student's weekly plan.
- Able to edit forum.
- Able to delete self comments.
- Able to leave comments under other comments.
- Enable individual chat function.

---

## 8 Summary and Conclusion

This report has explained the detail of the design and development of a project management system that accounts for social isolation. The project aims to develop an application that can encourage students to share their project information with other students, encourage academic discussion, and allow supervisors to track their student's progress. It is believed that the application has fulfilled these objectives, with a user-friendly user interface design. Also, the choice of the Spring framework and the MV architecture has ensured that the application has good maintainability. This project has encountered some challenges such as construct a large relational database with two accounts and three roles, and the approved proposal function. Fortunately, all the challenges are solved. The development can never be completed. This project still has plenty of room for future development.

---

## 9 Reference

JournalDev.com (No date) "Spring Boot Tutorial" Available at: <https://www.journaldev.com/7969/spring-boot-tutorial#:~:text=Advantages%20of%20Spring%20Boot%3A,-It%20is%20very&text=It%20reduces%20lots%20of%20development,Spring%20Data%2C%20Spring%20Security%20etc.> (Accessed: 23/08/2020)

"TutorialsPoint.com. (No date) "Spring - MVC Framework" Available at: [https://www.tutorialspoint.com/spring/spring\\_web\\_mvc\\_framework.htm](https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm) (Accessed: 23/August/2020)

Docs.spring.io. (No date) "Working with Spring Data Repositories" Available at: <https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html> (Accessed: 25/ August /2020)

Ashfaq, U. (2012) "What Is Dependency Injection/Inversion of Control And How Does It Help In Achieving Loose Coupling?" Available at: <https://medium.com/eastros/what-is-dependency-injection-inversion-of-control-and-how-does-it-helps-in-loose-coupling-f2143e58dbf2#:~:text=Spring%20framework%20contributes%20in%20loose,creating%20an%20object%20is%20inverted.> (Accessed: 26/ August /2020)

Karia, B. (2018). "A quick intro to Dependency Injection: what it is, and when to use it" Available at: <https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/> (Accessed: 26/ August /2020)

Admin, S (2013). "What is Difference Between Two-Tier and Three-Tier Architecture?" Available at: <https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/> (Accessed: 27/ August /2020)

Tyson, M.(2019) "What is JPA? Introduction to the Java Persistence API" Available at: <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html> (Accessed: 27/ August /2020)

Fadatare R, (2018) "What Is the Difference Between Hibernate and Spring Data JPA?" Available at: <https://dzone.com/articles/what-is-the-difference-between-hibernate-and-spring-1#:~:text=Hibernate%20is%20a%20JPA%20implementation,solution%20to%20GenericDao%20custom%20implementations.&text=question%20at%20StackOverflow.,Hibernate%20provides%20a%20reference%20implementation%20of%20the%20Java%20Persistence%20API,with%20benefits%20of%20loose%20coupling.> (Accessed: 28/ August /2020)

---

Javatpoint.com, (No Date). "Hibernate Tutorial" Available at:  
[https://www.javatpoint.com/hibernate-tutorial#:~:text=Hibernate%20is%20a%20Java%20framework,Persistence%20API\)%20for%20data%20persistence](https://www.javatpoint.com/hibernate-tutorial#:~:text=Hibernate%20is%20a%20Java%20framework,Persistence%20API)%20for%20data%20persistence). (Accessed: 27/ August /2020)

Sharmi, N. (2020) "Hibernate vs MyBatis " Available at:  
<https://www.performatix.com/hibernate-vs-mybatis/> (Accessed: 27/ August /2020)

Thakur, A (2020). "10 Best CSS Frameworks for Front-End Developers" Available at:  
<https://geekflare.com/best-css-frameworks/> (Accessed: 29/ August/2020)



# 10 Appendix

Git Repository: <https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2019/kxz959>

Set up information: Import the project into Eclipse or Spring IDE as a Gradle project.

Before running the project, please connect the project with the MySQL database. Database configuration file is in path :Msc\_Individual\_project\src\main\resources\application.properties.

Run the project with Gradle boot run.

Predefined account:

- supervisor account: ([www@qq.com](mailto:www@qq.com), password: 1) ([aaa@qq.com](mailto:aaa@qq.com) password: 1) ([bbb@qq.com](mailto:bbb@qq.com) password: 1)
- Student account: ([001@qq.com](mailto:001@qq.com) password:1) ([002@qq.com](mailto:002@qq.com) password:1) ([003@qq.com](mailto:003@qq.com) password:1) ([004@qq.com](mailto:004@qq.com) password:1) ([005@qq.com](mailto:005@qq.com) password:1)

## Appendix A. Additional testing example of the project

```
61 @Autowired
62 TagRepository tagRepo;
63 @Test
64 public void student_supervisor_create() throws Exception{
65     long pre = studentRepo.count();
66     long pre2 = superRepo.count();
67     long pre3 = smRepo.count();
68     BCryptPasswordEncoder bcpe = new BCryptPasswordEncoder();
69     // given
70     Students student = new Students();
71     String password = "Password";
72     student.setFirstName("kai");
73     student.setEmail("3@qq.com");
74     student.setPassword(bcpe.encode(password));
75
76     Supervisors s1 = new Supervisors();
77     s1.setFirstName("S1");
78     s1.setEmail("2@qq.com");
79     s1.setPassword(bcpe.encode(password));
80
81     Supervisors s2 = new Supervisors();
82     s2.setFirstName("S2");
83     s2.setEmail("1@qq.com");
84     s2.setPassword(bcpe.encode(password));
85
86     SecondMarker relation1 = new SecondMarker(student, s1, "Supervisor");
87     SecondMarker relation2 = new SecondMarker(student, s2, "SecondMarker");
88
89     Set<SecondMarker> studentlist = new HashSet<>();
90     studentlist.add(relation1);
91     studentlist.add(relation2);
92     student.setSecondMarker(studentlist);
93
94     Set<SecondMarker> s1list = new HashSet<>();
95     s1list.add(relation1);
96     s1.setSecondMarker(s1list);
97
98     Set<SecondMarker> s2list = new HashSet<>();
99     s2list.add(relation2);
100    s2.setSecondMarker(s2list);
101
102    smRepo.save(relation1);
103    smRepo.save(relation2);
104
105    // when
106    long studentNumber = studentRepo.count() -pre;
107    long supervisorNumber = superRepo.count() -pre2;
108    long sm_repo =smRepo.count() -pre3;
109    // then
110    assertThat(studentNumber)
111        .isEqualTo(1);
112    assertThat(supervisorNumber)
113        .isEqualTo(2);
114    assertThat(sm_repo)
115        .isEqualTo(2);
116    }
117 }
```

## Appendix B.

```
1 <div class="col-md-12 col-sm-12 col-xs-12">
2   <div class="container bootstrap snippets bootdey">
3     <div class="col-md-12">
4       <h2 class="text-primary">
5         <i class="fa fa-tachometer"></i> Home menu
6       </h2>
7       <hr>
8       <div class="row">
9         <div class="col-md-4">
10        <div class="col-md-4">
11        <div class="col-md-4">
12        </div>
13        <div class="row">
14          <div class="col-md-4">
15            <div class="col-md-4">
16              <div class="panel panel-warning">
17            </div>
18          </div>
19        </div>
20      </div>
21    </div>
22  </div>
23 </body>
24 </html>
```

## Appendix C Pivotal Tracker example

