CS673 Software Engineering

Team 3 - Project Name

Project Proposal and Planning

Team Member	Role(s) Signature	Date
Jianle Xie	Team Leader	*Jianle Xie* 9/9/2023
Yuxuan Wang	QA leader	*Yuxuan Wang* 9/9/2023
Abigya Devkota	Design and Implementation Leader	*Abigya Devkota* 9/9/2023
Jingjing Tang	Configuration Leader	*Jingjing Tang* 9/10/2023
HaoLun Li	Requirement Leader	*HaoLun Li* 9/11/2023
WeiHao Mai	Security Leader	*Weihao Mai* 9/11/2023
Zhiwei Lin	Backup leader	*Zhiwei Lin* 9/11/2023

Revision history

Version	**Author**	**Date**	**Change**
1	**Jianle Xie**	**9/25/2023**	**Update Risk Management**
2	**Jianle Xie**	**10/09/2023**	**update timeline/risk management**

1. Overview

(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed software system, the basic functionality of the proposed software system and the possible technology stack to be used.)

We are embarking on a six-week project to create an Employee Management System using Java and Spring Boot, motivated by the need to showcase CRUD operations (Create, Read, Update, Delete) in a practical setting.

Our software will help organizations efficiently manage employee roles. Staff and administration workers will use it to create, read, update, and delete roles. The Following includes basic underlying potential users, functionalities, and technology stack. Additional users, functionalities, and tools may be added as this project progresses within our time and ability constraints.

Potential Users:

1. Staff Workers: To view roles and request updates.
2. Administration Workers: To create, update, and manage roles.

Basic Functionality:

Our system will focus on CRUD operations:
- **Create:** Add new roles.
- **Read:** View existing roles.
- **Update:** Modify role details.
- **Delete:** Remove unnecessary roles.

Technology Stack:

- **Java:** as the default language.
-**IntelliJ as the default IDE
- **Spring Boot:** To build a reliable and scalable application/framework

Our project aims to showcase CRUD operations while delivering a practical solution for role management.

2. Related Work

(Please describe any similar software systems that you have found through the online research, and the differences between your software and those software systems.)

Our application will mainly focus on the primary function of create, read, update, and delete first with basic user authentication. Modern day employee management systems (such as Workday, BambooHR, and ADP) have a wide range of HR functionalities. They can not only pull employee records, but can act as a benefit manager, payroll manager, time tracking, performance evaluation and much more.

This employee management system will serve primarily as an education tool for understanding and exploring CRUD operations via SpringBoot framework and React user interface.

3. Proposed High level Requirements Functional Requirements:

(For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)
i. Essential Features (the core features that you definitely need to finish):

1. Create: As an employee or administrator, I wanted to create my employee information so that it can be stored in the database.
Estimation: 18 to 24 hrs

2. Read: As an employee or administrator, I wanted to view all current employees so that we know how many employees are there.
Estimation: 18 to 24 hrs

3. Update: As an employee or administrator, I wanted to update my information so that everything is accurate.
Estimation: 18 to 24 hrs

4. Delete: As an administrator, I wanted to be able to delete records when employees leave the company.
Estimation: 18 to 24 hrs

(Note: estimation is based on everyone's daily schedule)
(For each essential features, please give a rough estimation in terms of person hours or an range of person hours)

ii. Desirable Features (the nice features that you really want to have too):

Login/Security for the user:

For enhanced security, we would like to implement a login page in the application to authenticate each user, distinguishing between workers and administrators.

iii. Optional Features (additional cool features that you want to have if there is time):

Spring security for hashing
Explore Spring Security

b. Nonfunctional Requirements

i. Staff workers would be able to submit forms/requests to administrators for data change approvals.

4. Management Plan

a. Objectives and Priorities

(Please describe your project objectives with highest priority first. Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc)

We want to build an employee management system with basic functionalities (create, read, update, delete) with a user interface and no known bugs.

b. Risk Management (need to be updated constantly)

(Please write a summary paragraph about the main risks your group identified and how you plan to manage these risks. Then use the separate google sheet for detailed risk management. The template is provided in the same folder with this file. Please provide the link to the sheet.)

Risk as of Iteration 2

As we progress through the project/iterations, 'integration risk' has been identified. During the earlier iterations, we decided to split the group into sub-teams that will handle different functionalities of our application. While this facilitates planning by dividing the application into parts and conquering each part individually, it can also isolate tech skills/expertise to only a few individuals. Team members who were in charge of the backend, become very knowledgeable about their parts, just as team members in the frontend or security. The risk comes when each individual part needs to be assembled and merged into one functioning application. Since every member only knows their part and not how other parts function, there becomes a gap: how does the backend connect to the frontend or how does the frontend connect to the backend if each part only knows about themselves. Factor in the limited

time that we have for the project, it is difficult for all team members to relearn other member's parts in a short span of time.

To mitigate such risk, team members who are well versed in their part should teach and demonstrate their part of the application. Demo recordings are posted onto google share folder for everyone to view. Also, team members should be encouraged to experiment with other member's code so that they can have an understanding of each other's code. In the future, dedicating a team member as 'integration specialist' would be a good idea too. Someone needs to understand all the functioning parts of the application and how they interact, but not necessarily need to code out all the components.

Risks as of Iteration 1

As we progress through the project, we have identified new and more compelling risks. Although, technology incompetence is always a risk, the degree of it's priority has lessened as group members assimilate to their roles. However, communication has become a greater risk threat now. Since our group has been subdivided into 3 components (frontend, backend, security), the likelihood of working on wrong components is possible (priority level 9), given that each sub group retains a semi autonomous state. When each sub-group solely focuses on their own part, it makes integration in the future harder. But having a sub-group facilitates the workload and allows each group to specialize in their own role. To mitigate such risks, team lead will have to continuously engage with each sub-group and gain a minimal understanding of each part, so that integration at a later point would be more smooth.

Another great communication risk that needs to be mentioned is inconsistent communication (priority level 9). Because meetings are hard to schedule due to everyone's different availability, assigned tasks are usually agreed upon through discord. Thus, it is critical that team members respond back as soon as possible whether to acknowledge or disagree with something. When a group member does not respond for an extended period of time, the work from the project will still be carried out. Workload will need to be restructured when the group member communicates back, and that can create additional stress/risks to the project. To mitigate such risks, the team lead will continuously engage with each team member privately on a regular basis to ensure that each team member is actively present and aware of all the changes to the projects.

Risks as of Iteration 0:

There are many risks involved in this project, but currently our top 3 highest priority risk categories are Technology Competence, Design and implementation, and Integration and deployment.

To mitigate Technology Competence risks, we will engage in self learning of SpringBoot, Java, Junit, and other technology stacks.

To mitigate Design and Implementation risks, all team members shall continuously review the application design and recommend/suggest improvement and constructive discussion with other team members/role leads.

To mitigate Integration and deployment risks, we will focus on the core functionality of the application and all team members will review additional features thru discussion/team meets and budget time

accordingly.

Risk Management Sheet Link: CS673_SPPP_RiskManagement

c. Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

Iteration	Functional Requirements(Essential/ Desirable/Option)	Tasks (Cross requirements tasks)	Estimated/real person hours
1	Essential features: CRUD(create, read, update, delete) system	Learn spring boot framework for CRUD Build shell project	Estimated: 168 hours Actual: 173 hours
2	Essential features: Spring security integration	Continue building basic project requirements Learn about incorporating spring security to implement encryption	Estimated: 168 hours Actual: 71 hours
3	Desirable features: Login/Security for the user	Implement login encryption if time permits Get project to final stages	Estimated: 168 hours

5. Configuration Management Plan

a. Tools

(In this project, we will use Git and Github as the version control tools. Please also specify any other tools to be used, e.g. IDE tools, CI/CD tools, container tools, SAST or DAST tools, and any other DevOps tools)

Tools:

Java SDK 17 or later
Maven
Intellij IDEA
Junit
Apache
GitHub
GitHub Issues
React
Pivotal Tracker
Figma
Postman
VScode

b. Code Commit Guideline and Git Branching Strategy
(Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies: https://www.flagship.io/git-branching-strategies/

Everyone should have the repository cloned to their local machine. For every iteration or change, we will create a new branch and work off of that branch. We will create a pull request to merge our changes to the main branch. The pull request will be reviewed by other team members to make sure there are no conflicts.

c. Deployment Plan if applicable
(If you plan to deploy your application (e.g. your web application), briefly describe how you plan to deploy your application).
If time permits, we will look into deploying our application.

6. Quality Assurance Plan

a. Metrics
(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (KLOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

Metric Name	Description
Line of codes	Measurement of code complexity
# of classes	Measurement of code complexity
# methods	Measurement of code complexity
# defect per KLOC	Measurement of code perfection rate
Metric Name	Description
Line of codes	Measurement of code complexity
# test cases	Measurement of code quality
# of person hours used	Time commitment to the project
Test cases pass rate	Rate percentage of the passed test cases
# of user stories	Measurement of incremental development needs

b. Coding Standard

Java best practices(comments, readability..etc)
Maintain high readability, maintainability, availability

c. Code Review Process

(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other's code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)

Any changes to be merged to the masters should be done through a pull request. All team members should review the code. Master branch should be protected.

d. Testing

(Both manual testing and automated testing should be considered. Both unit testing and integration testing should be considered. Briefly describe the testing tools/framework to be used, the

personnel involved (e.g. the QA leader will focus on the integration testing and each developer will unit test their own code), when and what types of testing will be performed, the testing objectives, etc)

Each developer will unit test their own codes before sending pull requests to Github, then the quality assurance leader will use Junit Testing/ integration testing to test the codes and review them.

e. Defect Management

(Describe the tool to be used to manage the defect (e.g github issues). The types of defects to look at. The actions or personnel for defect management.)

We will use github issues for defect management

7. References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard.)

8. Glossary

(Any acronym used in the document should be explained here)