**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** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# 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 tool 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.

# 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.

# Proposed High level Requirements

* 1. 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).)

* + 1. 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

* + - 1. 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

* + - 1. Update: As an employee or administrator, I wanted to update my information so that everything is accurate.

Estimation: 18 to 24 hrs

* + - 1. 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 encryption

Explore Spring Security for encryption

* 1. Nonfunctional Requirements
     1. Staff workers would be able to submit forms/requests to administrators for data change approvals.

# Management Plan

## 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.

## 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.)

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.xlsx](https://docs.google.com/spreadsheets/d/1Tymvlvk6xtpKd6fv0JxV09BsPKSXD7JY/edit#gid=1482677559)

## 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 springboot framework for CRUD  Build shell project | Estimated: 24 hrs per person |
| 2 | Optional features: Spring security for encryption | Continue building basic project requirements  Learn about incorporating spring security to implement encryption | Estimated: 24 hrs per person |
| 3 | Desirable features: Login/Security for the user | Implement login encryption if time permits  Get project to final stages | Estimated: 24 hrs per person |

# Configuration Management Plan

## 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

* 1. 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.

## 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.

# Quality Assurance Plan

## 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 |
| # 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 |

* 1. Coding Standard

Java best practices(comments, readability..etc)

Maintain high readability, maintainability, availability

## 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.

## 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.

## 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.

# References

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

# Glossary

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