**CS673 Software Engineering**

**Team 4 - TeamBuilder**

**Project Proposal and Planning**

****

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Jake Kelly | Team and Requirement Leader | *Jake Kelly* | 9-9-24 |
|  |  |  |  |
| Raymond Chen | Configuration Leader | *Raymond Chen* | 9/9/2024 |
| Vishal Sundaram | QA Leader | *Vishal Sundaram* | 9-9-24 |
| Wenli Gai | Security Leader | *Wenli Gai* | 9/9/2024 |
| Max | Design and Implementation Leader | *Max Zink* | 9/12/24 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **0.0** | **Team 4** | **9-9-24** | **Proposal 0** |
| **0.1** | **Raymond** | **10/6/24** | **Deployment Plan** |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

Github: https://github.com/BUMETCS673/seprojects-cs673olf24team4

Jira: https://bu-cs673olf24-team4.atlassian.net/jira/software/projects/TB/boards/1

# Overview

TeamBuilder is a team assignment tool to help create teams based on a few simple, but important questions that users will answer. The motivation and purpose to create this tool is to make a team where collaboration, inclusivity, efficiency, and flexibility are emphasized to help users become more productive and engaging, improving overall team performance. The potential users are any group of people who want to form separate teams. The application will work as so: the user will get on the website, they will answer a few questions, and they will be placed into a team according to their answers and the answers of others.

(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. )

# Related Work

We tried searching for sites that would help build a team for a group of people, but not really finding anything closely related to what our website will be offering. Most searches come up with team building exercises or articles related on how one could successfully build a good team.

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

# 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).)
    - Essential Features (the core features that you definitely need to finish):

(For each essential features, please give a rough estimation in terms of person hours or an range of person hours)

* + - Desirable Features (the nice features that you really want to have too):
    - Optional Features (additional cool features that you want to have if there is time):
  + Essential features:
  + Creating a good team for users - User Tracker:

As a user, I want to be able to add my name and email (or other way of communication) to the TeamBuilder Questionnaire, so that I can be notified when I am placed in the correct team (knowing TeamBuilder only builds teams up to 10 people). Given users that answer questions when they use the TeamBuilder application then they will be placed into fitting teams accordingly. Est. hours: 30

* Quick use of the application. Notification System:

As a user, I want to receive a notification of my team assignment being completed, so that I can know who my teammates are as soon as possible. Given a questionnaire when users use the TeamBuilder application then they will be notified when the teams are created immediately. Est. hours: 5

* Creating code that helps. Easy to navigate UI:

As a developer, I want to create a friendly UI for the questionnaire, so that users can easily complete it. Given multiple people that have answered a few questions in different ways when they use the TeamBuilder application, then they are placed in the best teams possible. Est. hours: 5

* **Automated Team Formation**
  + As a developer, I want to build an algorithm that analyzes user responses (key words) and assigns users to teams, so that teams are formed efficiently.
  + Estimate: 20
  + Desirable Features (the nice features that you really want to have too):
  + **Multiple Group Creation**
    - As an admin, I want to create multiple sets of teams for different groups, so that I can manage diverse user groups.
  + Optional Features (additional cool features that you want to have if there is time):
  + **AI-Enhanced algorithm**
    - As a developer, I want to use machine learning to improve team matching over time, so that team formation becomes smarter and more optimized.

# 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 )

The primary objective of the TeamBuilder project is to develop a team assignment tool that improves collaboration, inclusivity, and team performance. The top priorities are: Completing all essential features as outlined in the functional requirements (e.g., assigning users to teams based on questionnaire responses). Deploying the tool successfully with zero critical bugs. Ensuring the application is user-friendly, with quick navigation and a responsive GUI. Providing flexibility for users to form teams with varying group sizes. Maintaining high-quality performance, allowing smooth functionality even with large datasets. Constantly refining the system for optimal team matching based on user inputs.

## 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.)

The project faces several key risks that require careful management. First, Data Management Risk arises from the possibility of inaccurate data matching due to improper handling of questionnaire responses. To mitigate this, strict validation rules for input data should be implemented, and a robust algorithm must be developed to sort and group users effectively. Another significant risk is System Overload, which could occur when handling large datasets or experiencing high traffic, potentially slowing down the system or causing errors.

To address this, the backend should be optimized for scalability, and the system should undergo stress testing under high load conditions. The project also faces a User Experience Risk, where complicated user flows may frustrate users and lead to low adoption rates. Mitigating this risk involves prioritizing ease of use through minimalistic design, conducting user testing, and ensuring fast response times for team assignments. Lastly, Feature Creep presents a risk, as the addition of non-essential features during development could delay the project timeline. To prevent this, the team should adhere strictly to the defined scope of essential and desirable features, only considering optional features if there is ample time available.

**Risk Management Sheet Link:https://docs.google.com/spreadsheets/d/1z8Uh\_lGYK7jILRBZHF5IywkVpfIQcxV5IDGY7EMKKGI/edit?usp=drive\_link**

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

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 0 | Complete project proposal 0 | Write up plan, set up development environments | 8 |
| 1 | Setup project structure Implement method to categories users | Set up Spring Boot backend  Set up Angular frontend  Create database schema  Implement basic user interface  Develop API endpoints | 40 |
| 2 | Implement team assign algorithm | Develop algorithm for team assignment based on user input | 50 |
| 3 | Implement multiple group functionality | Update frontend to allow creation and management of multiple groups Conduct thorough testing of the entire system | 50 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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)

In this project, we will use several tools to assist development, testing, and deployment processes:

* **Git/Github**: Git will be used as the version control. Github will serve as the remote repository for tracking changes and collaboration.
* **Jira**: Jira will be used to manage projects such as issue tracking, sprinting planning, organizing tasks, and timeline.
* **IntelliJ IDEA**: IntelliJ IDEA will be the main IDE tool for writing, debugging, and testing.
* **Spring Boot and H2 Database**: We will use Spring Boot for backend development with H2 database for local data storage in the project. H2 database is an embedded database with quick and simple setup to minimize configuration process.
* **JUnit**: Junit will serve as the testing framework for write and run unit tests to ensure project’s reliability.
* **Angular**: Angular will be used as the frontend framework for developing the user interface. It will integrate to the Spring Boot backend via REST APIs.
* **Playwright**: Playwright will be used as end-to-end testing. It serves the purpose of simulating user behaviors.
* **CI/CD tools**: GitHub Actions could be used with automating testing and deployment for both frontend and backend.
* **Docker**: Docker will be used for containerizing both the backend (Spring Boot) and the frontend (Angular) applications. This will ensure consistency across environments and simplify the deployment process.
  + 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/>

**Code Commitment**: Each commit will contain a completion of a task.

**Pull Request process**: Each code contribution will be reviewed via pull request before merge into main. The pull request will include:

* Passing tests(Junit and Playwright)
* Code review by at least one other team member.

**Branching Strategy**: We will adopt Gitflow branching strategy.

* **main**: This branch will contain a stable and production-ready code. No direct commits will be allowed on main.
* **Iteration branches**: Each iteration will be developed in a new branch created from main. Once complete and tested, it will be merged into main through pull request.
* **Feature branches**: Each feature will be developed in its own branch created from the Iteration branch. Once complete and tested, it will be merged into Iteration through pull request. Example: feature/user-questionnaire.
* **Task branches**: Each task under the feature will be divided into its own branch. Example: task/setup-database.

**Release**: Release will be created from the Iteration branch after testing and bug fixes are completed.

* 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).

**Deployment Architecture**

Backend: Spring Boot application host on Render.

Frontend: Angular application serve on GitHub Pages.

Database: H2 database for development.

Version Control: GitHub for source code management.

**Deployment Tools**

CI/CD Tools: GitHub Actions for continuous integration and deployment.

Containerization: Docker for containerizing the Spring Boot application.

Build Tools: Maven for the backend; Angular CLI for the frontend.

# Quality Assurance Plan

## Metrics

| **Metric** | **Description** | **Measurement** | **Target** |
| --- | --- | --- | --- |
| Lines of Code (LOC) | Measures the size of the codebase | Total number of lines of code | - |
| Cyclomatic complexity | Measures the complexity of the code logic | Count of independent paths through the code | ≤ 10 for methods |
| Test coverage | Measures the percentage of code exercised by tests | (Lines of code executed ÷ total LOC) × 100 | ≥ 80% |
| Number of defects | Measure of accuracy of code | Number of bugs reported | - |
| Sprint/iteration velocity | Tracks the number of user stories completed per sprint/iteration | Number of user stories completed per sprint | - |
| Customer Satisfaction | Measures how satisfied users are with the software | Surveys or feedback ratings (scale of 1-5) | ≥ 4 |

* + Coding Standard
* Naming conventions
  + Use upper Camel case for Class names e.g. StudentProfile
  + Use lower Camel case for method and variable names e.g. myString
  + Use upper Snake case for constants e.g. MAX\_RETRIES
  + Use lower case for package names e.g. edu.bu.csmet.teambuilder
  + Use Kebab case for Angular component/file names e.g. student-profile.component.ts
* Typescript guidelines
  + Use strict types (string, number, boolean, etc.) and avoid using any
  + Null and Undefined: Handle null and undefined values explicitly to avoid runtime errors
  + Async/Await: Prefer async/await for handling asynchronous code instead of callbacks or promises

## Code Review Process

1. Author creates a Pull Request for the newly developed feature
2. Initial automated testing (unit, integration, E2E) and static code analysis to catch basic issues early on through CI/CD pipeline
3. Peer Review:
   1. One or more reviewers are assigned based on expertise
   2. Local testing and verify feature requirements
   3. Code readability and maintainability
   4. Should adhere coding standard
4. Reviewer gives feedback directly on the PR and author makes necessary changes
5. Once reviewer approves changes, code is merged to main branch

## Testing

* Testing framework
  + JUnit for unit tests on server side
  + Playwright for Acceptance/E2E tests
  + Jasmine/Karma for UI unit testing
* Manual testing to test for usability, consistency of look & feel, performance
* Developer will unit test their own code
* QA Leader is responsible for integration and end-to-end testing

## Defect Management

* Bugs are tracked through project management tool - JIRA
* Treat bugs the same way as features
* Bug classification based on priority:
  + Critical
  + High
  + Medium
  + Low

# References

# Glossary