**CS673 Software Engineering** 

**Team 6 - iGroup**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Shawn Hu | Team leader | *Shawn Hu* | 09/19/2022 |
| Dawei Yin | Design and Implementation Lead | *Dawei Yin* | 09/20/2022 |
| Siming Qian | Req & Sec Leader | *Siming Qian* | 09/19/2022 |
| Haiyang Wang | QA Leader | *Haiyang Wang* | 09/20/2022 |
| Lijian Yao | Configuration leader | *Lijian Yao* | 9/19/2022 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **Iteration 0** | **Team 6** | **Sep 21.2022** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[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)

# 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. )

* Our project is making a team assignment tool that can automatically assign students to a team, which involves the concepts of front-end, back-end, database and so on. We named it "iGroup". The main idea of the Team assignment tool is to help create teams based on students preferences and background. Algorithmically, we weight different preferences to create the most appropriate team.

**High-level Description:** A Web-based Application for team assignment

**Motivation:** Help professors automatically assign teams based

on the survey while focusing on Class Size which is no more than 50 students.

**Goal:** Efficiently form teams and visualize the results on web page

Tech Stack and Frameworks:

#### **Front End:**

HTML: Front End building block

CSS: Front End building block

Javascript: used for more dynamic elements in front end

#### **Back End:**

**Java**

Spring Boot (optional)

#### **Git(Github):**

Source control version repository.

#### **Database:**

MySQL(optional)

# 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.)

* Try to comprehensive the previous student project

<https://github.com/BUMETCS673/BUMETCS673A1F21P3>

* Survey

<https://docs.google.com/forms/d/e/1FAIpQLSfkskHnSJLweJSulYyeBenhNPSyzHkQTRW6wzakM_Ffb3gJFA/viewform>

* Team Sign

<https://docs.google.com/document/d/1PzFrv9GLrKaXZP46fl1S_kaAev4Gb6BcfAxBUfSvNSs/edit>

* Related Literature

https://drive.google.com/drive/u/1/folders/1BBafMsvOnvsn76p2TktR9ZJ1NIbP4Eg5

# 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. As a new user, I want to register as a student, so I can take the survey.
        2. As a new user, I want to register as an instructor, so I can do operations.
        3. As an instructor, I want to add students into the system/class, so students who register late can get in.
        4. As an instructor, I want to remove students from the system/class, so students who dropped can get out.
        5. As an instructor, I want to see each student's survey answers, so I can manage the distribution manually if needed.
        6. As an instructor, I want to change attributes' priority, so the distribution will be more flexible.
        7. As an instructor, I want the algorithm to generate groups by attributes’ priority, so all teams come out in a reasonable form.
        8. As an instructor, I want to create teams manually, so I can manage the class.
        9. As an instructor, I want to remove teams manually, so I can manage the class.
        10. As an instructor, I want to add members into the group manually, so students who register late can get in. (so I can adjust the teams if needed)
        11. As an instructor, I want to remove members out of the group manually, so students who dropped can get out. (so I can adjust the teams if needed)
     2. Desirable Features (the nice features that you really want to have too):
        1. As an instructor, I want to add attributes into the system, so the team distribution will be more universal.
        2. As an instructor, I want to remove attributes from the system, so the team distribution will be less universal.
        3. As an instructor, I want the algorithm to process each attribute by dissimilarity or similarity as I prefered, so the team distribution will be more reasonable.
     3. Optional Features (additional cool features that you want to have if there is time):
        1. As a student, I want to see other’s survey answers, so I can know my classmates better. (same group only?)
        2. As a student, I want to show my survey answers to other students (same group only?)
        3. As a student, I want to hide my survey answers from other students (same group only?)
     4. Existing Features:
        1. The survey
  2. Security Requirements
     1. Essentials
        1. Users should create strong passwords without telling anyone about it.
        2. Encrypted during connection and rest in DB.
        3. Logs of requests.
     2. Client side
        1. Student may not how the algorithm works
        2. Change password periodically
     3. Fancy things that may never needed
        1. Auto logout after exact period without actions
        2. IP block
        3. Honey pot
        4. Versioning
        5. Load balancer
        6. Alarm of suspicious activity
        7. Incident interpretation
        8. Psychical protection of DB
        9. Penetration test
     4. Reserved spaces

# 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 )

**Highest priority, deploy a valid algorithm for group allocation; next, achieve modularity; no known bugs; maintain high user experience.**

## 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 Management Sheet**](https://docs.google.com/spreadsheets/u/1/d/1oXajwHWtP0rqrXSfCWsf2NTtd7TqJNTJ5b6tWfP5csk/edit) **(click it!)**

## 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 |
| --- | --- | --- | --- |
| 1 | 1. Finish algorithm design 2. Build initial project code | 1. Algorithm Brainstorming 2. Data structure design 3. Learning development skill | 100 |
| 2 | 1. Finish development 2. Write some Unit Tests. 3. Start to do the integration testing. 4. Start to check security | 1. Everyone finishes their coding. 2. Do some simple tests. 3. Verify part-to-part linkage 4. Analyze security issues 5. Carry out security inspection | 120 |
| 3 | 1. Finish testing 2. Finish checking security | 1. Make sure our application can run as a whole. 2. Reduce the probability of occurrence of bugs. 3. Make sure our app gives the right results. 4. Reduce the secure risks. | 110 |

# Configuration Management Plan

## Tools

1. Version Control: Git,Github
2. Front end (optional): VS Code(HTML,CSS)
3. Back end: intellJ(IDE)(Java), framework(optional):Spring Boot,Maven
4. Database(optional):DataGrip(PostgreSQL relational database) :
5. CI/CD:TeamCity(optional)

(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)

* 1. **Code Commit Guideline and Git Branching Strategy**

GitHub-Flow

GitHub Flow is a very suitable Git Branching Strategy for a small team like us. It allows us to all develop together effectively. and the pull request will be reviewed by every team member, especially tech lead before merge.

## Deployment Plan if applicable

Use Maven to pack our Spring Boot project to an executable jar file or in other ways.

# Quality Assurance Plan

## Metrics

| Metric Name | Description |
| --- | --- |
| Quality | Clean and clear. Bug rate. |
| Reliability | Rate of providing exact service. |
| Performance | Time to run |
| Usability | User-friendly or not |
| Correctness | Rate of correct results |
| Maintainability | Time of maintenance |
| Security | Defense of attacks |

* 1. Coding Standard

Please refer to this link: <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>

Our standards will be based on the contents in this link.

## Code Review Process

Team members review each other’s code. We will make sure we all review code that is not from ourselves. We will use pull requests for the code review, to make sure there is no conflict. Recently we don’t have a checklist that everyone must do, but we do require the reviewer to give feedback about cleanliness, clarity and bugs.

## Testing

Our testing will be based on the AAA model (Arrange, Act, Assert), and our test will use frames like JUnit. We will always use asserts() to achieve the test goals. In a test case, use only a minimum number of assertions; preferably only one assertion per test method. This ensures that our tests are unique and correspond to only one feature of our app.

About unit testing, at first we will test the very basic functionality, make sure our methods can run at the first. In this part, the contents in asserts() will be really simple, but having enough amount. These contents include things like adding, returning, etc. Here we want to make sure our code can give us the results we want. Also the contents may cover some simple special situations, just like inputting zeros or negative values. Our teammates are recommended to write some simple unit tests by themselves. The QA leader will continue to add more unit tests about special inputs and situations until we all think it is ok.

About the integration test, it will basically be done by our QA leader, but other teammates can also help if needed. Here, enough random data that represents the students should be generated. The QA leader will combine the previous unit tests to form more integration tests. These integration tests will test the coordination of our various parts to ensure that the combination of contents will not cause problems. This will ensure that every module of our app runs with only few errors or even without errors. At the same time, we will use these integration tests to minimize the possibility of appearances of bugs. In this case, either black box testing or white box testing will likely be used.

About the system testing, we will test all parts of the entire application together, including the environment in which it runs. This part will be done by the QA leader only, and it will always be black box testing. This part is to ensure that defects and problems related to the internal and external interfaces of the entire system are resolved, and the performance and load of the whole project will also be tested. We will simulate real cases as much as possible to ensure a good user experience， and finally get a product that meets our specified requirements.

## Defect Management

We might use:

ClickUp <https://clickup.com/>

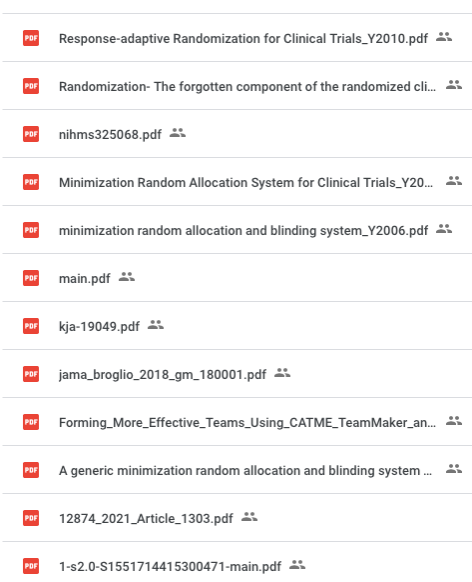
Bugzilla <https://www.bugzilla.org/>

These tools will help us upload bugs we find and eventually get them fixed.

# References

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

[Here are All References !](https://drive.google.com/drive/u/1/folders/1pURHLwjMS-YIpG8iNqkOSN9_c2rAcr6k)



# Glossary

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