**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 Leader | *Siming Qian* | 09/19/2022 |
| Haiyang Wang | QA Leader | *Haiyang Wang* | 09/20/2022 |
| Lijian Yao | Configuration leader | *Lijian Yao* | 9/19/2022 |
| Alex Wang | Security Leader | *Alex Wang* | 9/19/2022 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **Iteration 0** | **Team 6** | **Sep 21.2022** |  |
| **Iteration 1** | **Team 6** | **Oct 20.2022** | 1. Motivation2.Related work description3.Functional Requirements4.Management Plan （time line）5.Quality Assurance Plan （metric，Testing Plan for IGroup） |
|  |  |  |  |
|  |  |  |  |

[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 around 100 to 200 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:**

#### Django (optional)

#### Pure Python(algorithm part)

#### **Management:**

#### Git/Github: Version control tool

* + Pivotal Tracker: Project Management

#### **Database:**

#### MySQL(optional)

* + .CSV file

#### 

# 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>

We can refer this work if we want to create the web application and have a established front end.

* Survey

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

The previous survey can give us a better reference to the types of questions

* Related Literature

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

The literature in this folder is the origin of our algorithm idea. The algorithm of our application is based on the algorithm of Team-maker

# 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 an instructor, so I can do operations.
        2. As an instructor, I want to add students into the system/class manually, so students who register late can get in.
        3. As an instructor, I want to remove students from the system/class manually, so students who dropped can get out.
        4. As an instructor, I want to see each student's survey answers, so I can manage the distribution manually if needed.
        5. As an instructor, I want to change attributes' priority, so the distribution will be more flexible.
        6. As an instructor, I want the algorithm to generate groups by attributes’ priority, so all teams come out in a reasonable form.
        7. As an instructor, I want to create teams manually, so I can manage the class.
        8. As an instructor, I want to remove teams manually, so I can manage the class.
        9. 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)
        10. 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)

NEW STUFF BELOW FOR THIS PART

* + - 1. As an instructor, I want to create the survey by clicking the “create” button, so that I can add questions into the survey.
      2. As an instructor, I want to save the survey by clicking the “save” button, so that I can generate a link to the survey and send it to the student.
      3. As an instructor, I want to login the system with my BU credential, so that I can access the system.
      4. As a student, I want to login to the system with my BU credential when I click the survey URL, so that I can take the survey.
      5. As a student, I want to submit the survey by clicking the “submit” button, so that my answers can be recorded and saved in DB.
      6. As a student, I want to rate the page (UI) after the survey, so that developers know how bad their work may be and ways to improve.
      7. As an instructor, I want to be able to check the status of survey by clicking
      8. As an instructor, I want to be able to check the group assigning result, so that I can let myself and students know.
    1. 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.

NEW STUFF BELOW FOR THIS PART

* + - 1. As an instructor, I want a dynamic progress metric shown at the homepage of the system, so that I can know how many students has finished the test and when to run the team dividing algorithm.
    1. 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. As a new user, I want to register as a student, so I can take the survey.
    2. Existing Features:
       1. The survey
  1. Security Requirements
     1. Essentials
        1. Users should create strong passwords without telling anyone about it.
        2. Use the http during the transmission and encrypt data in the DB (database)
        3. Logs of requests.
     2. Client side
        1. 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. Set up student instance 2. Set up class instance 3. adding/removing student 4. adding/removing teams | 1. Algorithm Brainstorming 2. Data structure design 3. Learning development skill | 100 |
| 2 | 1. Setting priority with user’s desire 2. Generate teams with algorithm 3. Testing enable   Optional:   1. adding/removing attribute with user’s desire 2. dissimilarity /similarity of attribute | 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 | Optional:   1. Student can see/hide their survey’s answer from each other 2. Finish testing 3. 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: intelliJ(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

(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 (LOC, # 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 |
| --- | --- |
| # of files |  |
| # of classes |  |
| # of method |  |
| # of commit |  |
| LOC (Length of code) |  |
| # of person hours used |  |
| # of user stories completed |  |
| Time that the program runs |  |

* 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 Plan for IGroup

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.

1. Objectives

1.1 Modules under test

We will basically use the python unittest module.

1.2 References

[unittest — Unit testing framework — Python 3.10.8 documentation](https://docs.python.org/3/library/unittest.html)

[Getting Started With Testing in Python – Real Python](https://realpython.com/python-testing/#testing-for-web-frameworks-like-django-and-flask)

2. Testing levels and methods

2.1 Testing levels

Unit test:

All the main functions will be tested, These will include the initialization function, add/delete functions, and the functions about algorithms. Here are the main classes that we will test.

* Answer Sheet class
* Question class

1. Single choice questions class
2. Multiple choice questions class

* Survey class
* Team class

1. Single\_choice\_scores (main algorithm)
2. Multiple\_choice\_scores (main algorithm)
3. Schedule\_scores (main algorithm)

* User class
* Main class (random team member displacement, score comparison)

These are the class files we have.

Integrate test:

Tests through parts and parts

* Combine our classes. Make sure our functions can work across classes.
* Combine scores evaluation function and random displacement function, test about random displacement of team members.
* Combine user classes to UI, make sure users can successfully login first.
* Combine UI and functions.
  1. UI test (student)

i. Able to login

ii. Able to take survey

iii. Able to submit

* 1. UI test (Instructor)

i. Able to login

ii. Able to set survey

iii. Able to Create Link

iv. Able to manually setting group

v. Able to get result

System test:

Some parts of it will be black box test. Here is the whole process of our application, we want to make sure there is no problem in the end:

1. User logins (here is the instructor)
2. User presses start group assign button, frontend send the request.
3. Backend receive request and run the functions
4. Backend gets the result successfully, sends the result to frontend.
5. Frontend browses result.
6. User rate results and experience (optional)

Random user data will be generated, some other tests might also be applied, for example: multi-task tests, critical point tests, interrupt tests (optional)

Acceptance test:

We might invite instructors and our friends to use our grouping application. We will get the feedback from them and improve the performance and the user experience of our application

as much as we can.

User Acceptance Testing:

To see whether our application fulfills the requirements of the customers (here, means user, like instructor and students).

Operational Acceptance Testing (OAT):

The performance will be tested (like running time). Also we want to ensure the stability of our application, like whether it is easy to maintain, to recovery. (Because our app is not big, we might only have few test in OAT part)

Regression test:

After we have done all the test above, if we still want to make

changes, we will do the regression tests. First we will see which part is to be modified, then we will choose the tests that can still be run in this condition and rerun some of them again. Then we will make new tests to test the new part. (Because we might not do modification in the end, this part is based on the actual situation)

Currently we are in the process of unit testing.

2.2 Testing Methods

Equivalence class testing for result evaluation.

Boundary value test for extreme values

Statement testing to ensure all the parts are executed once.

Load testing to see how many students we can handle.

Of course, we will make sure our functions output the right values based on the input we give.

3. Test Cases (not finished yet)

| Test ID | Test Case Description | Expected Result |
| --- | --- | --- |
| 1 | Test Answer Sheet class initialization | Class can be initialized successfully, initial parameters are stored. |
| 2 | Test Question class initialization | Class can be initialized successfully, initial parameters are stored. |
| 3 | Test Survey Class initialization | Class can be initialized successfully, initial parameters are stored. |
| 4 | Test team Class Initialization | Class can be initialized successfully, initial parameters are stored. |

## Defect Management

We might use:

ClickUp <https://clickup.com/>

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

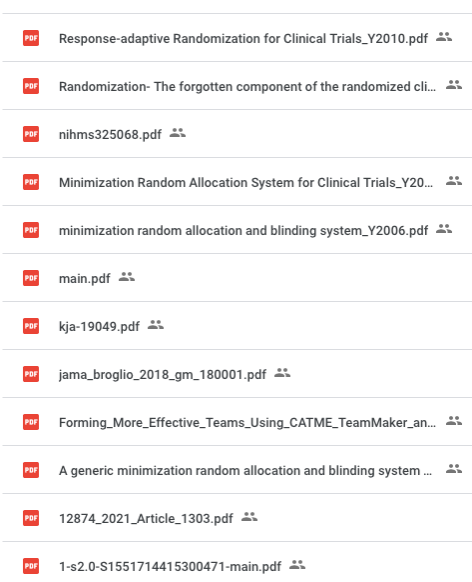
PivotTracker

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)