**CS673 Software Engineering**

**Team 3 - Rhettoric**

# Project Proposal and Planning

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Magnus Urosev | Team Leader / Design and Implementation Leader | *Magnus Urosev* | 5/13/24 |
| Adrian Ortiz | Configuration Leader / Security Leader | *Adrian Ortiz* | 5/13/24 |
| Xi Zeng | QA Leader / Design and Implementation Leader (DBs) | *Xi Zeng* | 5/13/24 |
| Jack Cairns | Requirement Leader,  Design and Implementation Leader | *Jack Cairns* | 5/25/24 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Revision history

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **v0.01** | **Magnus** | **5/12** | **Added draft Overview and Related Work** |
| **v0.02** | **Magnus** | **5/13** | **Expanded more on project details and timeline** |
| **v0.03** | **Adrian** | **5/13** | **Added Configuration Management Plan, And relevant links pointing to corresponding topics** |
| **v1.0** | **Magnus Urosev** | **5/15** | **Added base description and text to overview, related work, requirements, calorimanagement plan, glossary, configuration management, QA plan, and references** |
| **v1.01** | **Magnus Urosev** | **5/20** | **Applying Feedback from Dr. Zhang** |
| **v1.02** | **Magnus Urosev** | **5/23** | **Revising feedback** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[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. )

* Survey and Analytics Service: At the end of every project or lesson, there will be a survey where users are asked to complete a series of questions to provide feedback to the course creators/facilitators.
  + Facilitator can retrieve completed surveys
  + Facilitators can get metrics such as “X participants rated Q1 as 3.5 average”

# Related Work

* 1. Google Forms
  2. Survey Monkey

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

The differences between those services and ours is that we are creating a survey application that solely focuses on the ease of creating and deploying surveys and just that. Things like survey monkey and google forms focus on integration with their other suite products while our’s is just on the survey.

# 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):

(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):

Facilitator:

* As a course facilitator, I want to be able to create new surveys and modify questions, So that I can gather feedback from users who completed the course/lesson.
* As a course facilitator, I want to be able to access the data of completed surveys, So that I can observe the feedback from users who completed the survey.

User:

* As a user who has completed a course, I want to be able to access and complete the survey related to the course, so that I can provide feedback on the course/project.
* As a user who has completed a survey, I want to be able to review my completed survey, So that I know how I responded to the survey.
  + 1. Optional Features (additional cool features that you want to have if there is time):

* 1. Nonfunctional Requirements
     1. Performance
        1. The system should provide exam scores within 10 minutes of submission.
        2. The system should be able to process up to 1,000 exam submissions per hour.

#### Scalability Requirements

* + - 1. Design the system to handle varying loads and an increasing number of users, ensuring it can scale without degradation in performance.

#### Security Requirements

* + - 1. Implement secure coding practices to protect user data and prevent common vulnerabilities like SQL injection and cross-site scripting (XSS)

# 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 )

* Be able to create exam items (all active) that have a difficulty measure (IRTb), supports Rasch (all discriminations are constant), support only fixed-form for now, allows creation and testing of new forms, return CI of theta estimate, provide Raw and Scaled score to professor or class facilitator
* Be able to pull exam and performance information from front-end teams and do analysis on backend then provide display/data to FE team

## 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 Link:**

* Unable to create model due to depreciating library
  + Could switch to R or manually create model from scratch (Magnus)
* Small group size due to dropped class
* Overlapping services with another team

## 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 | User should be able to create questions and surveys | We should be able to create placeholder survey and question models and be able to test them | 60 hours |
| 2 | User should be able to complete a survey and provide to users and have data stored in a database | We should be able to have question and survey data stored in a database and be able to be retrieved for analysis | 80 hours |
| 3 | Information on survey answers and completion should be sent to facilitator and user | We should be able to see the information from the survey and some type of report | 80 hours |

# Configuration Management Plan

Document Doc going in depth: [WIP - Configuration Management Plan](https://docs.google.com/document/d/1FCX7UnXCcizmWhpgYWI7EVShs9SpPc0qnszpmpefKLk/edit?usp=sharing)

## [Tools](https://docs.google.com/document/d/1FCX7UnXCcizmWhpgYWI7EVShs9SpPc0qnszpmpefKLk/edit#heading=h.knm5jdbzw6cx)

(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)

* Version Control - Git and GitHub
* IDEs - PyCharm and Visual Code Studio
* CI/CD - GitHub Actions
* Container Tools - Cocker
* Security - Bandit
* DevOps - Possibly Kubernetes

## [Code Commit Guideline and Git Branching Strategy](https://docs.google.com/document/d/1FCX7UnXCcizmWhpgYWI7EVShs9SpPc0qnszpmpefKLk/edit#heading=h.dijofkggyrto) (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/>

* Main - Stable production ready code
* dev-# = General branch for feature and bug fixes
* feature/<name> - Branch for adding new features
* bugfix/<name or ticket #> - Branch for fixing bugs
* release/<version #> - Branch for preparing new production release
* hotfix/<name> - Branch for hotfixes

## [Deployment Plan if applicable](https://docs.google.com/document/d/1FCX7UnXCcizmWhpgYWI7EVShs9SpPc0qnszpmpefKLk/edit#heading=h.n5gkklyd412d)

(If you plan to deploy your application (e.g. your web application), briefly describe how you plan to deploy your 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 |
| --- | --- |
| Lines of Code | Number of lines of code in project total |
| Number of files | How many files are in the project |
| Number of classes | Number of classes in project |
| Cyclomatic Complexity | Measure of the complexity of the control flow |
| Defect Rate | Numbers of defects per kilolines of code |
| Number of test cases | Total num of test cases written |
| Test Case Pass Rate | Percentage of passed test case divided by number of test cases |
| Execution Time | Time taken for the model to score exams using Rasch implementation |
| Memory Usage | Amount of memory using during scoring process |

## Coding Standard

(Describe any coding standard to be used) - [Code Commit Guideline and Git Branching Strategy](https://docs.google.com/document/d/1FCX7UnXCcizmWhpgYWI7EVShs9SpPc0qnszpmpefKLk/edit#heading=h.dijofkggyrto)

* PEP8 for Python
* APA 7th ed. Styling for any Statistics or Math output
* Trunk-Based Development For Smaller Teams
* Committing straight to the trunk

## 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?)

Future pull requests will need to be reviewed and follow this template before merging onto main. The follow members will need to identify these features of a Pull Request before being Merged by either the Team Leader or Configuration Leader and confirming PR:  
  
Requirement Leader - Ensure code meets project requirements and user stories

Design and Implementation Leader - Ensure code follows sound design and implementation patterns

Team leader / Design and Implementation - Ensure statistical and machine learning models are correctly implemented and ensure code follows best practices. Review merging.

Configuration / Security Leader - Oversee configuration and security of codebase

QA Leader / DB - Oversee quality assurance and database design implementation  
  
# Pull Request

## Description

<!-- Please include a summary of the changes and the related issue. -->

## Checklist

- [ ] My code matches all coding standards.

- [ ] I included documentation updates to the coding standards, when needed.

- [ ] I ran CI Tests and Unit Tests locally before submitting.

- [ ] My code resolved all of the task's acceptance criteria.

## Changes Proposed in this PR

- \*\*Updated existing component:\*\*

- <!-- Updated places where said component was used -->

## Acceptance Criteria Scenarios

List all relevant new/existing Cucumber scenarios from the specs folder. Insert your specs folder path here.

1. [ ] [Scenario 1: Show Sprint reconcile header state]

2. [ ] [Scenario 2: Tap reconcile header's "move back" button]

3. [ ] [Scenario 3: Tap reconcile header's "merge into sprint" button]

4. [ ] [Scenario 4: Tap reconcile header's close button]

## Visuals

Include any relevant changes to visual/animations/UX flows.

- All visual changes MUST include before & after screenshots.

- All UX flow changes MUST include an animated video.

- If no visuals are possible, write "N/A" in the visual section.

<!-- Add before & after screenshots here -->

## Additional Information

<!-- Any other relevant information or comments. -->

Future issues will need to follow this template

Inspired by Chihack Night and MIT

## 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)

Unit Testing will use Django’s build in test along with Pythin’s unittest if needed. These tests will be executed and reviewed by the QA Leader. Manual testing of statistic models will be examined using exploratory testing and regression testing by the Team Leader or QA leader.

GitHub Action will be used to automatically run tests with the help of the QA and configuration leaders.

The objectives are to verify that individual units of code are working as expected and to ensure different application components are working as expected. The Security Leader will work with QA and Configuration to mitigate security vulnerabilities. Bandit and safety can be used to verify security. Team leader will use Pandas, numpy, and exploratory testing to ensure model performance.

## 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. )

Issues will be identified and logged with GitHub issues and will need to follow the below template when reported

# Issue Name

## Description

<!-- Provide a detailed description of the issue. Include any relevant context or background information. -->

## Checklist

- [ ] I have searched for existing issues that might be related to this one.

- [ ] I have included a clear and descriptive title for the issue.

- [ ] I have included steps to reproduce the issue.

- [ ] I have included the expected result.

- [ ] I have included the actual result.

- [ ] I have included the environment details (OS, browser, application version).

## Steps to Reproduce

1. [ ] Step 1: <!-- Describe the first step -->

2. [ ] Step 2: <!-- Describe the second step -->

3. [ ] Step 3: <!-- Describe the third step -->

## Expected Result

<!-- Describe what you expected to happen. -->

## Actual Result

<!-- Describe what actually happened. -->

## Environment

- \*\*OS:\*\* [ ] <!-- e.g., Windows 10, macOS Catalina -->

- \*\*Browser:\*\* [ ] <!-- e.g., Chrome 89, Firefox 86 -->

- \*\*Application Version:\*\* [ ] <!-- e.g., 1.0.0 -->

## Screenshots/Logs

<!-- Attach any relevant screenshots or logs that can help diagnose the issue. -->

## Additional Information

<!-- Include any additional information that might be relevant to the issue. -->

Adapted from the Odin Project

# References

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

* Django Documentation
* GitHub Documentation
* Bandit Documentation
* Py-Test Django Documentation
* The Odin Project GitHub - Issue Reporting
* Chihack Night - Open Source Project Template
* MIT - Open Source Guidelines
* PEP 8 - Style Guide for Python Code
* APA 7th ed. Sterling

# Glossary

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

* FE - Front End
* QA - Quality Assurance
* Docs - Documentation