# Development Plan
# Software Engineering

Team #11, OKKM Insights
Mathew Petronilho
Oleg Glotov
Kyle McMaster
Kartik Chaudhari

Table 1: Revision History

| **Date** | **Developer(s)** | **Change** |
|---|---|---|
| 9/18/2024 | Mathew Petronilho | Added Member Roles and Coding Standards |
| Date2 | Name(s) | Description of changes |
| ... | ... | ... |

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

[Additional information on the development plan can be found in the lecture slides. —SS]

## 1 Confidential Information?

[State whether your project has confidential information from industry, or not. If there is confidential information, point to the agreement you have in place. —SS]

[For most teams this section will just state that there is no confidential information to protect. —SS]

## 2 IP to Protect

[State whether there is IP to protect. If there is, point to the agreement. All students who are working on a project that requires an IP agreement are also required to sign the "Intellectual Property Guide Acknowledgement." —SS]

## 3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

## 4 Team Meeting Plan

The team will meet weekly on Mondays from 11:30 until 13:20, in a study room booked by Kyle McMaster. The schedule to meet with our supervisor will be determined when our supervisor is confirmed.

Weekly meetings will be chaired by Kyle, who will prepare an agenda to be sent in advance of team meetings. Team members can add additional agenda items as needed. Team members are also able to request additional meetings if necessary. In this case, it is their responsibility to chair the meeting, organize a meeting location, and provide an agenda. Following all meetings, the meeting chair will prepare a list of action items.

## 5 Team Communication Plan

[Issues on GitHub should be part of your communication plan. —SS]

## 6 Team Member Roles

| **Member Name** | **Roles** | |
|---|---|---|
| Mathew Petronilho | Document Manager, Front-End Development Expert | Developer, Tester, PR Reviewer, Issue Creator, Meeting Participant, and Note Taker. |
| Oleg Glotov | Team Lead, Project Manager | |
| Kyle McMaster | Meeting Chair, Back-End Development & AI Expert | |
| Kartik Chaudhari | Customer Relations Manager, Git Expert | |

Table 2: Team Roles

- **All Team Members:** Every team member is responsible for developing code and creating tests for the code. Everyone is also responsible for reviewing open pull requests and providing feedback if necessary. Additionally, all members will be tasked with creating issues using the appropriate templates, tracking issue status, and updating issues assigned to them with relevant information. Each member is expected to attend meetings punctually and contribute ideas to discussions, while maintaining respectful and concise communication. The role of meeting note taker will rotate among members in each meeting. The note taker should keep track of meeting attendance, issues discussed, decisions made, and action items. These notes should be well-maintained and easily accessible to all team members.
  If we encounter challenges, we may consider switching roles to maintain progress and improve team performance. More specific roles can be assigned as the project evolves and implementation details become clearer.

- **Mathew Petronilho:** Responsible for ensuring that all documents are formatted consistently, that all necessary components are included, and that there are no grammatical or spelling errors. Also responsible for assisting team members with the front-end and taking the lead in implementing this component of the project.

- **Oleg Glotov:** Responsible for liaising with the supervisor, teaching assistant, and professor. Coordinates project tasks among team members, organizes meetings, ensures equitable distribution of work, and monitors deadlines to ensure they are met.

- **Kyle McMaster:** Responsible for creating meeting agendas, guiding discussions, managing meeting time, and resolving conflicts. Also responsible for assisting team members with the application's back-end logic, deploying services, and contributing expertise in machine learning and artificial intelligence to integrate advanced data processing, and intelligent system functionalities.

- **Kartik Chaudhari:** Responsible for contacting potential customers and managing customer relationships. Oversees the GitHub repository by organizing files and ensuring it is updated to reflect project progress. Provides guidance to team members on resolving issues related to Git.

[You should identify the types of roles you anticipate, like notetaker, leader, meeting chair, reviewer. Assigning specific people to those roles is not necessary at this stage. In a student team the role of the individuals will likely change throughout the year. —SS]

## 7 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classification, etc.?
- Use of CI/CD

### 7.1 Git Workflow

### 7.2 Issue Management

### 7.3 Usage of CI/CD

#### 7.3.1 Tex Files

To ensure the PDFs found in our repository are consistent with the tex files they are generated from, we have developed a CI workflow in GitHub actions. This workflow detects when a push has been made to the *docs* folder in the *main* branch. When this happens, the workflow automatically regenerates the relevant documents and pushes the new PDFs to the repository. This will ensure that our PDFs are always the most recent version.

#### 7.3.2 Linting & Static Checks

As discussed in sections 10 and 11, we expect we will use Python to develop the backend components of our software. To improve the clarity of our code, especially when collaborating, we have decided to use type hinting. To enfore this requirement and reap further benefits of type hinting, we will use a static type checker as part of our CI/CD workflow. This is possible with a package called mypy.

For both the front and back ends of the project, we have selected coding standards as described in section 11. To ensure these standards are met, we will include a linting step before pull requests can be merged into the main branch.

#### 7.3.3 Testing

We will use a suite of tests to ensure our code base continues to satisfy our requirements before integrating changes into the code base. These tests will be run when a new pull request is created, before a code review is conducted.

#### 7.3.4 Deployment

Depending on how we decided to host our server, we will also investigate if it would be beneficial to develop a continuous deployment workflow when changes are pushed to the main branch. This will be determined in the future when our requirements are more clear.

## 8 Project Decomposition and Scheduling

- How will you be using GitHub projects?
- Include a link to your GitHub project

[How will the project be scheduled? This is the big picture schedule, not details. You will need to reproduce information that is in the course outline for deadlines. —SS]

## 9 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

## 10 Expected Technology

[What programming language or languages do you expect to use? What external libraries? What frameworks? What technologies. Are there major components of the implementation that you expect you will implement, despite the existence of libraries that provide the required functionality. For projects with machine learning, will you use pre-trained models, or be training your own model? —SS]

[The implementation decisions can, and likely will, change over the course of the project. The initial documentation should be written in an abstract way; it should be agnostic of the implementation choices, unless the implementation choices are project constraints. However, recording our initial thoughts on implementation helps understand the challenge level and feasibility of a project. It may also help with early identification of areas where project members will need to augment their training. —SS]

Topics to discuss include the following:

- Specific programming language

- Specific libraries
- Pre-trained models
- Specific linter tool (if appropriate)
- Specific unit testing framework
- Investigation of code coverage measuring tools
- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done
- Specific performance measuring tools (like Valgrind), if appropriate
- Tools you will likely be using?

[git, GitHub and GitHub projects should be part of your technology. —SS]

## 11 Coding Standard

Our back-end code, written in Python, will adhere to PEP 8 for code formatting and PEP 484 for type annotations. For our front-end code, written in JavaScript, we will follow the JavaScript Standard Style. [What coding standard will you adopt? —SS]

## Appendix — Reflection

1. Why is it important to create a development plan prior to starting the project?

   Starting with a development plan was useful for our team for two reasons. First, it helped us develop a clearer picture of what our project will look like. In order to make development plan decisions, we had to discuss the actual project itself. This gave us a chance to clear up misconceptions we were having, and discuss what our overall mission is. More importantly though, working on this document (and problem statement/goals) was a dry run for future deliverables. As we worked on this doc, we were able to put our plans of how we thought we would work together to the test. We found that our git workflow needed a couple of refinements and clarifications. For example, we needed to create more descriptive pull request titles and descriptions so that reviewers know what they are looking to review before approving. Otherwise, it was very hard to get feedback on the *Team Charter* section if the pull review was called '8-expectations'.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

   We are already feeling the benefits of CI/CD in our workflows. Kyle built a tex workflow to compile our latex documents every time there is a push to the main branch. This makes sure that the .tex file and its associated PDF never become out of sync. In the past, we have all experienced challenges working on text files in git, as it is very easy to upload a code change and much harder to upload a pdf (usually requires fixing a merge conflict). This means that we probably just would do one big compile and upload when the document is 'finished'. The problem with that becomes when you are creating living documents and it never really gets finished and the PDF then becomes hard to trust as a source of truth.

   This workflow also exposed us to a disadvantage of CI/CD. It is very easy to spend a lot of time on it. Kyle spent close to 3 hours just getting the git workflow to work and that is not even a very complicated deployment. It is exciting to have such a hands on and interactive part of the code base, where changes are easy to see in real time. This makes it risky that we dedicate a lot of time to CI/CD, when it's not really necessary. We also intend to use CI/CD for regression testing. We have to be careful that we don't let the lovely green check mark showing that a PR passes all the tests, replace thorough code review. When people are pressed for time, it's easy to be tempted to skip reading the code if the tests pass. Although we will try to make a comprehensive test suite, there is always value to reading the code too.

3. What disagreements did your group have in this deliverable, if any, and

how did you resolve them?

We only really had one disagreement for this deliverable and it had to do with our collaboration style on documentation. Some members of the team would prefer to first write sections in a shared google doc, and then upload all of the changes at once when complete and reviewed. This would make it easier to contribute to other sections, not bog us down with pull requests, and allow others to read the document in its entirety. The other members prefer using git issues and pull requests to improve the traceability of contributions to the project. We have had a few discussions regarding this and in general they have been productive. We conduct brainstorming and task allocation in a shared google doc, so that everyone has access to the task delineation. Before discussing the issue further as a group, we decided to confirm with Dr. Smith whether the first option is even permissible. Following this response, we will meet again to finalize our approach. For now, we are using the git issues and pull requests to ensure traceability, as a small sacrifice to the ease of collaboration.

# Appendix — Team Charter

## External Goals

As a team, we all see this project as an opportunity to demonstrate the extent of our skills and knowledge after several years of training. As such, our main goal is to develop a project we can talk with excitement to future recruiters and for some of us, graduate school admissions committees. We intend to work hard to make a project we are very proud of. We all expect this effort will translate into an A+ in the course, although that is somewhat secondary. Our effort will not be limited by the expectations of the capstone course, and instead by the limits of our abilities.

## Attendance

### Expectations

- Team members are expected to attend every meeting in person. If attendance in person is not possible, they will notify the team at least 24 hours in advance so the team may discuss mkaing the meeting virtual or rescheduling.

- If attendance is not possible in person or virtually, they will notify the team at least 24 hours in advance. Each team member will be permitted to have 2 absences per semester without penalty. Additional absences, or absences without proper warning, will have the following diciplinary action:

  **1-2** Bring snacks to following meeting.

  **3-4** Bring snacks to following meeting & a message will be sent to TA.

  **5+** Message is sent to course instructor. Meeting is scheduled with team to discuss absences. If meeting is not attended, student receives 2.5% deduction in final course grade per offence.

### Acceptable Excuse

- Acceptable excuses are typically limited to exceptional circumstances that are urgent and could not have been foreseen. Examples of such events are medical emergencies of oneself or of family members, severe weather, or other risks for personal safety.

- Unacceptable excuses can be reasonably foreseen or expected by the member of the group. Examples of unacceptable excuses are getting busy with other courses, missing the bus, or over sleeping.

- If a team member is unable to meet a deadline due to other responsibilities, they are to alert the group with at least 7 days of notice. Each team member will be permitted to request 1 reduction of work per semester, provided other team members are able to take over the task and 7 days of notice is provided.

- Excuses are to be judged by group consensus on a case by case basis. If an excuse for absence is deemed unacceptable, penalties described above will be applied.
- Missing a deadline with an unacceptable excuse will be met with an immediate penalty of a message sent to the TA describing the situation for the first offence. All other missed deadlines will result in a message sent to the instructor and a 2.5% penalty on course grade.

### In Case of Emergency

- In the event of an emergency, team members must contact group as soon as reasonably possible if a meeting or deadline might be missed. Team members should avoid completing work at the last minute to lessen the effect of emergencies on the group.

## Accountability and Teamwork

### Quality

Based on our expectations for the project, the team is required to meet a very high standard of quality. Each addition to the project should be crafted with high attention to detail, and reviewed thoroughly by the creator. We will expect that every pull request is adding code or text that is ready to be submitted to the best of our abilities. To help each other ensure we meet this standard, we will require every addition to be reviewed by at least one other team member. This reviewer will approach the review as if they are the last line of defense for the quality of our codebase.

Every team member will complete expected action items and review the provided agenda before each meeting. They will provide additional agenda items when necessary.

### Attitude

- Team members will treat other with respect at all times
- Team members will be friendly and collaborative
- Team members will remember that we are all working towards a common goal, to develop a high quality product
- Team members will not act hostily towards eachother. Offenders to this will face diciplinary action as outlined in the absence penalty structure

- Team members will openly communicate
- Team members will be open to team member ideas. If they do not agree, they are welcome to provide reasons as to why not. It is not permitted to decline an idea because it is 'not the way we do things'
- Team members will try to work together to make work load equitable
- Team members will try to accommodate external responsibilities, but understand it might not always be possible around large deadlines
- Team members will answer questions and provide additional information to team members who are interested in developing skills others are proficient in
- **We will be kind**

**Stay on Track**

The team will work well in advance to assign work to team members. The team will schedule time in weekly meeting to have a 'stand up' meeting, where they discuss progress and blockers. This will ensure there are no surprises when a deadline is approaching. We will also define earlier internal deadlines to provide enough time for someone to cover in the unlikely event that a team member is not able to meet a deadline.

We will track team member progress by the amount of deliverables they complete in each milestone. Typlically, one deliverable will correspond to one or two GitHub issues. We believe this metric will be more useful than lines of code or commits, both of which can be manipulated and do not always capture the amount of value contributed. If planned correctly, each issue should contain roughly the same amount of effort to complete (although this won't always be the case).

We will also track attendance at meetings and lecture. Attendance at meetings is very important and penalties for absence are discussed above. Attending lecture is also important to the team, but less so. As long as we are able to find one person to attend, we will not worry much about lecture attendance. We will track lecture attendance so if we need to force someone to attend, we can choose the person with worst attendance.

**Team Building**

We already have a strong start with building rapport within the team. Everyone is very friendly towards each other and is aligned to the mission. We tend to sit together in lecture and are active in our group chats. After a large deliverable, the team has agreed to celebrate at a location of their choosing. Kyle might even host a party if there is interest within the team.

## Decision Making

In general, we will strive for team consensus. If it is not possible, opposing parties will prepare arguments for their proposal. Other members of the team can then ask further questions, or announce their preference. To make a change to the current path, a majority is needed. Ties will result in the status quo. If at any time a group member has *new* arguments, they may bring the issue up again. It is important that team members are poilte and rational, but also provide a realistic indication of how deeply they feel about their arguments. If someone could go either way on an issue, and we are stuck in a stalemate, learning that they do not have a deep stake might be enough to help the group move forward.

Regardless of how passionate team members are about certain issues, they will follow our code of conduct and continue to be kind.