Development Plan
Project Name: Course Buddy

Team #5, Overwatch League
Jingyao, Qin
Qianni, Wang
Qiang, Gao
Chenwei, Song
Shuting, Shi

Table 1: Revision History

Date	Developer(s)	Change
2023/9/28	Jingyao Qin, Qianni Wang	Initial draft of the document
Date2	Name(s)	Description of changes
...	...	...

The project is a time management tool designed to assist students in studying and completing coursework efficiently and in an organized manner.

1 Team Meeting Plan

Team meetings for Team 5 will occur at least three times a week, excluding SFWRENG 4G06 lectures and tutorials.Each meeting will be no longer than 50 minutes in length, with a preference for Microsoft Teams on-line meetings, and the rest of the meetings will take place off-line on the McMaster University campus, and will be held at the following times The meeting time will be chosen from the following time slots that have been agreed upon by the entire team:

- Tuesday 5:30 p.m. to 8:00 p.m.
- Wednesday 2:30 p.m. to 5:00 p.m.
- Thursday 7:00 p.m. to 8:00 p.m.
- Friday 7:00 p.m. to 8:00 p.m.
- Saturday 12:00 p.m. to 5:00 p.m.
- Sunday 12:00 p.m. to 5:00 p.m.

Meeting Types:

- **Sprint Planning Meeting:** This type of meeting is held every two weeks and is primarily used to discuss breaking down work tasks, assigning issue owners, setting issue deadlines, establishing short-term goals and plans, as well as defining the completion criteria and expectations for tasks. Team supervisor will attend this type of meeting.
- **Stand-up Meeting:** This type of meeting takes place three times a week, on Tuesday, Thursday and Friday, typically lasting no more than half an hour. It is primarily used for each team member to report on their work for the week, provide updates on progress, outline their next steps, discuss whether they need assistance, and identify any obstacles.
- **General Meeting:** The frequency of this type of meeting is random, and sometimes not all team members need to participate. It may involve one-on-one discussions. The primary purpose of this meeting is to quickly address specific details or engage in in-depth discussions about a single issue. Attendees should coordinate whether it's necessary to communicate the key meeting content to the rest of the team.

2 Team Communication Plan

WeChat is a platform used by the team on a daily basis and this is the quickest way to connect with team members.Z WeChat is mainly used to discuss task assignments, meeting times, meeting locations and meeting durations, brainstorming about the project, and planning extracurricular team building activities.

To better track project progress and work distributions, we use GitHub to manage issue assignments and deadlines, use GitHub issues to record team meeting notes, and also use GitHub Pull requests to review, modify, and approve PR requests that make changes to the project repository.

3 Team Member Roles

Member Name	**Roles**	
Jingyao Qin	Team Lead, Expert on LaTeX and Team Management	Developer, Tester, Reviewer, PR Approver, Issue Creator, Team Meeting Host and Scribe.
Qianni Wang	Team Co-Lead, Expert on Git and Project Management	
Shuting Shi	Project Board Manager	
Chenwei Song	Document Manager, Expert on Documentation	
Qiang Gao	Coordinator, Data Engineer	

Table 2: Team Roles

Team Member Responsibilities:

- **All Team Members:** Responsible for coding, conducting testing, reviewing pull requests, and approving them only after verifying and testing the modified documents. Additionally, responsible for creating issues with the appropriate issue template and

configuring project settings, assigning team members, and applying labels. Ensure the successful achievement of the project's goals by the end of the capstone course.

- **Roles Rotate within Team:**For every meeting, there will be a rotating host and scribe within the team, with the exception of sprint planning meetings.
- **Jingyao Qin:** responsible for liaising with supervisors, teaching assistants, and professors, scheduling team meetings, ensuring equitable distribution of work within the group, and tracking and ensuring that project milestones are within a reasonable process.
- **Qianni Wang:** Oversee GitHub repository management, ensure proper formatting and consistency of issues and pull requests, and maintain organization and structure within the project repository.
- **Shuting Shi:** responsible for hosting Sprint Planning meeting ensuring that all created issues are correctly labelled, assigned, and associated with the project, and ensuring that each issue is in the right status column on the project board view.
- **Chenwei Song:** for ensuring that all documents in the main branch of the GitHub project repository are updated to the latest version and free of typos and grammar issues.
- **Qiang Gao:** Responsible for assisting the team lead in coordinating project tasks, tracking project progress, facilitating internal team communication to ensure smooth and timely information flow, managing team financial management.

4 Workflow Plan

A general development workflow will be:

- **Opening an issue**

  Create an issue using *GitHub*'s issue tracker. Whenever there is a team meeting, lecture, or task, a *GitHub* issue should be created using the corresponding templates. An issue created for recording lecture attendance should follow the issue template created for lecture purposes. An issue created for assigning a task (bug fix, adding a new feature) should follow the issue template created for general purpose. An issue created for documenting the team meetings should follow the issue template created for meeting purposes. An issue should have at least one label (e.g. `bug`, `documentation`, `repo management`, etc), and at least one assignee, and project linked to it. After an issue is created, the assignee is responsible for updating the status of the issue inside the *GitHub* project board. There are four different statuses of issues inside the project board and each one of them should be put under the correct column:

  - A newly created issue should be put under `Todo` column.
  - An issue should be put under `In Progress` column when someone picks it up and starts working on it
  - An issue that the work has been done but needs a verification (e.g. when a PR is up and ready for a review) should be put under `In Verification` column.

  - An issue should be put under `Done` once it gets completed

- **Making code changes**

  The steps below show when a code change needs to be merged into the main branch:

  1. Clone the project *GitHub* repository from remote main to local main if the local system does not have it cloned. Otherwise, pull the latest changes from the remote main
  2. From *VS Code*, open the folder containing the repo, a Dev Container should be configured to run to provide a development environment. *Dev Container* should be used when doing development to ensure the development environment team members are using is consistent
  3. Create a feature branch follows the naming convention of '[issue number]-[short description]', for example, the branch name '12-create-pr-check-workflow' will be a feature branch designated for the issue #12 and the commits on this branch should be related to creating a workflow that checks the PR
     - Each commit will be checked by a workflow performed by `GitHub Actions`. The workflow will perform lint checks, automated testing, etc., to make sure code changes are aligned with our code standards and will not break anything. If the run of workflow fails, the PR containing the commits cannot be merged to the main
     - Each PR will have at least one label associated with it, and will have an issue linked to it, otherwise, the PR check workflow will fail and the PR cannot be merged to the main
     - If the branch of the PR is behind the main branch and there are conflicts that have not been solved, then a `git rebase` is needed to ensure the feature branch is up to date with the main branch, and the conflicts need to be resolved before the PR can be merged
  4. Checkout to the feature branch created and the development should happen inside this branch. Unit tests and integration tests should be developed or updated alongside the code
  5. Commit the code changes on this feature branch using a descriptive message
  6. Push the code changes to the remote feature branch
  7. A branch protection rule is created for the main branch, code changes cannot be merged to the main branch directly, it can only be done by creating a PR, and at least two team members need to review and approve the PR before it can be merged. A draft PR should be created when it is still under development, a regular PR should be opened only when it is ready for a review
     - Each commit will be checked by a workflow performed by `GitHub Actions`. The workflow will perform linting checks, automated testing, etc., to make sure code changes are aligned with our code standards and will not break anything. If the run of workflow fails, the PR containing the commits cannot be merged to the main

   - Each PR will have at least one label associated with it, and will have an issue linked to it, otherwise, the PR check workflow will fail and the PR cannot be merged to the main
8. Once the PR is approved and merged
   - The remote feature branch should be deleted
   - The issue linked to the PR should be closed

5 Proof of Concept Demonstration Plan

What will be the main risk for the success of the project?

The main risk of this project will be having too many features for an 8-month project, making it too hard to perform integration tests and manage/maintain. In our project, we need to:

- Manage a database
- Implement an algorithm that extracts date and task-related texts from a PDF
- Build a model to classify the priority of a task
- Construct a training pipeline to define the steps when training the model
- Design a user-centered and user-friendly interface
- Deploy a browser-based web app

This includes techniques and knowledge spanning databases, data engineering, machine learning, and UI design, which might require a large amount of time to learn, implement, etc. Due to the sheer amount of topics and systems, we wish to implement, this might result in not being able to be fully implemented and well-tested in an 8-month period.

Will a part of the implementation be difficult?

The part of the implementation that we find will be difficult is to construct a whole training pipeline for the task priority classification that involves many steps for example, data ingestion step, preprocessing step, training step, evaluation step, post-process step, etc. It might be difficult to set up the pipeline, define the input and output of each step, arrange all the steps and make them work together, and define the appropriate triggers for starting the pipeline. We could create a flow diagram to show that we have analyzed and defined the input/output of each step, and we could construct a simple training pipeline for a simple classification task: Iris classification, which involves several simple steps including the preprocessing step, training step, and verification step to demonstrate that this can be overcome for the task priority classification.

Proof of Concept Demonstration

There might be many moving parts to our idea, but once the basis for them is set up, a lot of them will be managed a lot more smoothly moving forward. In order for us to feel as though our project is moving forward at an acceptable rate, there are a few basic milestones we would like to reach:

- Have a simple data storage, even if it's as basic as reading and writing from a test file for now
- A workflow that can extract basic data from a user's syllabus
- Preliminary model that accepts syllabus and schedule data
- Template pipeline with steps and framework chosen to handle the model training
- Host a local-based web app that can handle some basic communication between the model and the database
- Minimal user interface designed simply for functionality

Setting these up will be the bulk of the learning curve for technologies that aren't our main focus. Setting up a form of simple storage database, website, and data extraction workflow, are all processes that don't need to be altered too much while working with the data. Getting all these done and out of the way at this point will let us be able to focus on the lengthy process that will be fine-tuning the model and training pipeline to try to get the best results from our data.

Will testing be difficult?

Testing the model's performance will not be difficult as some of us have worked on AI projects that perform cross-validation (K-fold) tests using *scikit-learn* library. For those members that are not familiar with cross-validation testing or *scikit-learn* library, online documentations and tutorials can be found easily for example, accessing *scikit-learn* website from here, reading cross-validation documentation from here. Testing web apps will not be difficult as we have worked with *Python* projects that are using *Streamlit/Flask* as the web framework and *PyTest* as the testing framework from previous co-op work terms and have learned and used those frameworks from projects of previous courses like *SFWRENG 3X03*. As a result, we will be able to perform unit tests and integration tests for the web apps.

Is a required library difficult to install?

No, the required library is not difficult to install, using tools like *pip* can easily install the library required very easily by running a command line.

Will portability be a concern?

No, the portability will not be a concern since this project is building a browser-based web application so any device that has a browser will be able to run the application.

6 Technology

The following tools and packages will be used for the development and management of the project:

- *GitHub*: For version control, tracking issues, and keeping code and documents
- *Visual Studio Code*: Main IDE for code development, debugging, etc.

- *Python*: Primary language for developing models, setting up pipelines, and building up web apps
- *PyTest*: A testing framework for *Python* that will be used to construct unit tests for web-based features
- *Streamlit/Flask* Web frameworks for *Python* that will be used for web development
- *Pandas*: A *Python* library that will be used for data manipulations, data visualization, and data analysis
- *numpy*: A *Python* library that will be used for numerical computing such as performing operations between arrays
- *scikit-learn*: A machine learning library for *Python* that will be used to develop models, set up pipelines, and evaluate the performance of a trained model
- *Flake8*: Linter tool that checks the format of the code against the PEP8 coding standard

7 Coding Standard

The project will follow the *PEP 8* coding standards as *Python* is the primary language that the project will be using. There is also a linting tool called *Flake8* that will be used in one of the workflows for the main branch to ensure the code being merged to the main is following *PEP 8*.

8 Project Scheduling

In this section, we encompass significant milestones and associated deadlines. The project will be partitioned into 6 phases, each marked by a specific collection of actions and deliverables.

8.1 Project Phases

The project will be divided into the following phases:

1. **Initiation Phase** (Duration: [September 5] - [September 25])
    - Team building and project selection.
    - Identify the project supervisor.
    - Create a GitHub project repository.
2. **Planning Phase** (Duration: [September 26] - [October 11])
    - Create a Development Plan.
    - Define the problem and scope of the project.
    - Identify stakeholders and investigate their needs.
    - Create SRS Documentation.

3. **First Implementation Phase** (Duration: [October 12] - [November 24])
    - Address the implementation challenges of the project.
    - Perform Hazard Analysis.
    - Create the Verification and Validation Plan.
    - Present Proof of Concept Demonstration.
4. **Second Implementation Phase** (Duration: [November 25] - [January 17])
    - Implement the main features of the project.
    - Perform testing.
    - Optimize the implementation details of the project.
    - Create the Design Document (Version 0).
5. **Evaluation Phase** (Duration: [January 18] - [March 6])
    - Evaluate the achievement of project goals.
    - Perform risk and security assessments.
    - Conduct user end testing assessments.
    - Demonstrate Revision 0.
    - Prepare the Verification and Validation Report.
6. **Closure Phase** (Duration: [March 7] - [April 4])
    - Conduct the final demonstration.
    - Get ready for the Expo Demonstration.
    - Complete all documentation.

8.2 Milestones and Deadlines

The following are milestones and their respective deadlines for this project:

- **Team Formed, Project Selected** - [September 18]
- **Problem Statement, POC Plan, Development Plan** - [October 4]
- **Requirement Document Revision 0** - [October 11]
- **Hazard Analysis 0** - [October 20]
- **Verification and Validation Plan Revision 0** - [November 3]
- **Proof of Concept Demonstration** - [November 13-24]
- **Design Document Revision 0** - [January 17]
- **Revision 0 Demonstration** - [February 5-16]
- **Verification and Validation Report Revision 0** - [March 6]
- **Final Demonstration (Revision 1)** - [March 18-29]
- **EXPO Demonstration** - [April TBD]
- **Final Documentation (Revision 1)** - [April 4]