Development Plan
Software Engineering

Team 2, SyntaxSentinals
Mohammad Mohsin Khan
Lucas Chen
Dennis Fong
Julian Cecchini
Luigi Quattrociocchi

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept 19	Lucas Chen	Start template
Sept 19	Mohammad Mohsin Khan	Appendix, Team Member Roles, Team Meeting Plan
Sept 23	Luigi Quattrociocchi	Add POC plan and change team member roles
Oct 7	Luigi Quattrociocchi	Change software license from MIT to GPLv3

Contents

1 Github Repository **3**

2 Copyright License **3**

3 Team Meeting Plan **3**
3.1 Meeting Agenda Template 4

4 Team Communication Plan **4**
4.1 Attendance Tracking 4
4.2 Issue Creation and Tracking 5

5 Team Member Roles **6**

6 Workflow Plan **7**
6.1 Workflow to merge changes 7
6.2 Additional project management tools 7
6.3 Coding Standard 8
6.4 Expected Technology 8

7 Proof of Concept Demonstration Plan **8**

8 Project Scheduling **9**

This report outlines the key components of the SyntaxSentials project plan, starting with the adoption of an MIT license for the project. The team's meeting structure is detailed, including weekly meetings and monthly check-ins with the industry advisor. Key roles, such as chair and minute-taker, are assigned, and a meeting agenda template is provided for structured discussions.

The communication plan revolves around GitHub for task management and issue tracking, with detailed steps for issue creation, labeling, and resolution. Team members are assigned roles, and a workflow for merging code changes, performing testing, and pushing updates to production is presented. Additionally, a proof of concept demonstration plan highlights potential project risks. Concluding with expected technologies, coding standards, and a high-level project schedule.

1 Github Repository

The project will be hosted on GitHub, found here. The repository will be used for version control, issue tracking, and documentation.

2 Copyright License

The project will be licensed under the GPLv3 license, ensuring that any derivative works must also be distributed under the same license, promoting software freedom and collaboration. Notably, the GPLv3 license disallows distributing closed source versions of our software. The license can be found here.

3 Team Meeting Plan

The team will meet for at least 4 hours every week, 2 hours during the allotted tutorial time on Monday at 2:30 pm. Precise location will be open but the meeting will be virtually on discord, otherwise on McMaster University campus. the other 2 hours allotted would be done virtually over the weekend.

Meetings will be scheduled with our industry advisor depending on their schedule virtually. The plan is to meet for at least 4 hours every month for guidance and to confirm that the team is on the right track. We hope to meet with our advisor for at least 1 hour at the end of every month for a recap. In person meetings with our advisor will be conducted depending on their availability.

Roles

- **Chair**: Dennis will chair all meetings.
- **Minute-taker**: Mohsin will take minutes during all meetings.

Meeting Preparation

Agendas will be collaboratively prepared by the team before meetings with the advisor to ensure we are focused and productive.

3.1 Meeting Agenda Template

Meeting Details

- **Date**: [Insert Date]
- **Time**: [Insert Time]
- **Location/Platform**: [Insert Location or Virtual Platform]
- **Attendees**: [List of Attendees]

Agenda Items

1. **Review of Progress Since Last Meeting**
    - Updates on tasks and milestones
    - Discussion of any challenges or roadblocks
2. **Discussion Points**
    - Key questions or topics for discussion
    - Feedback and suggestions from the supervisor
3. **Next Steps and Action Items**
    - Summary of key takeaways
    - Assignment of new tasks or responsibilities
    - Timeline for upcoming milestones
4. **Any Other Business (AOB)**
    - Additional concerns or questions
    - Future meeting scheduling

Meeting Minutes

- Minutes taken by: Mohammad Mohsin Khan

4 Team Communication Plan

4.1 Attendance Tracking

The attendance issue will be updated for each lecture and meeting that takes place. This issue can be found on the project GitHub page.

4.2 Issue Creation and Tracking

Tasks will be issued via creating a GitHub issue. This approach ensures all task-related discussions and updates are in a centralized place, providing a clear record of progress. Each issue must include a detailed explanation, enabling all team members to understand the task at hand without needing further clarification. Additionally, issues will be categorized using labels to specify their nature. The available labels are:

- **Documentation**: For tasks related to documentation.
- **Bug**: For resolving errors or malfunctions.
- **Chore**: For minor housekeeping issues such as typos.
- **Feature**: For implementing new functionalities.
- **Enhancement**: For enhancing existing features.
- **Low Priority**: For tasks that are not urgent.
- **Med Priority**: For tasks that need attention.
- **High Priority**: For tasks that need immediate attention.
- **Spike**: For tasks that require further clarification or research/feedback.
- **Lecture**: For tracking attendance and questions during lectures.
- **Meeting**: For tracking team meetings (with or without supervisor).
- **Review**: For tracking external reviews.

Additional labels will be introduced as needed throughout the project. Issues will be assigned to the team member most suited to the task, based on their expertise. All relevant commits will be displayed under the issue, and it will automatically close once the associated pull request is merged into the main branch. Including a closing tag (e.g., "closes #12") in the pull request ensures proper closure of the issue. This structured process guarantees accountability and keeps tasks traceable.

Step-by-Step Process

1. **Open a New Issue**
    - Write a clear, detailed description to ensure the team fully understands what needs to be done without needing additional questions or clarification.
    - At this stage, the team can discuss what needs to be done, identify the problem, and determine the best course of action.

2. **Add a Label**

    - Choose one of the following labels based on the task type:
        (a) Bug
        (b) Chore
        (c) Feature
        (d) Enhancement
        (e) Low Priority
        (f) Med Priority
        (g) High Priority
        (h) Spike

3. **Assign the Issue**

    - Assign the issue to the most relevant team member.

Once the above steps are completed, the assignee should follow the established process workflow. After finishing the task and submitting a pull request, they must ensure the pull request provides sufficient detail to demonstrate the task has been resolved. This could include test cases, bug fixes, or other relevant changes. The pull request should also include a closing tag (e.g., "closes #12") to automatically close the issue and ensure proper tracking.

5 Team Member Roles

All parts of the project can and will be worked on by every member, however we plan to specialize in the following way.

- **Mohsin**: Mohsin is the Liaison and team leader. Among other things he will ensure that the team is on track and that each member is on track with their assigned tasks. Mohsin will also take notes during meetings to facilitate this. Mohsin will assist in all other technical aspects of the project.

- **Lucas**: Lucas will specialize in designing and prototyping the user interface(s) and UI/UX component of the project. Lucas will lead the user-facing aspects of the project, such a front end web interface. Lucas will also ensure that the git flow is followed within our project's github.

- **Dennis**: Dennis will chair all meetings (unless unavailable). Dennis specializes in machine learning and language models, he along with Julian will design the architecture and prescribe work to the team accordingly.

- **Julian**: Julian specializes in theoretical and practical applications of machine learning. He (with Dennis' help) will be the lead architect of the NLP component of the project.

- **Luigi**: Luigi specializes in programming lanugages and is the resident expert in traditional software plagiarism checkers (such as MOSS); Luigi will lead the portions of the project that involve compiler front-ends and normalization.

All members will contribute to documentation and code an equal amount - that is, no one member will be solely responsible for any piece of documentation, or of code.

6 Workflow Plan

6.1 Workflow to merge changes

1. Pull changes from the main branch
2. Run unit/E2E tests and any new test cases required for new code (Must pass a certain threshold to be approved)
3. Create a branch with the following template, branch type/SS/ticket number. (e.g. feature/SS-01, where branch type is feature or fix or spike, SS as SyntaxSentinel and 01 is the ticket number taken from GitHub Issues board)
4. Comment code with branch type followed by a descriptive message (e.g. feat: add database connection)
5. Open pull request
   (a) Must follow PR template, found on GitHub project page
   (b) Automated tests are run using GitHub Actions
   (c) Reviewers ensure that code meets acceptance criteria and does not regress any test cases
   (d) At least one approving review is required
6. Merge the branch into main

6.2 Additional project management tools

We plan to utilize GitHub Projects (specifically the Kanban board view) to efficiently track and manage issues throughout their lifecycle. Each column on the board represents a different stage of the workflow - "To Do," "In Progress," "In-Review" and "Done." Having a card for each issue will help us visualize progress, assign tasks, and update statuses in real-time.

6.3 Coding Standard

For this project, we will adhere to coding standards using flake8 and PEP8 to maintain high code quality and readability. Flake8 will help enforce style guidelines and detect errors, while PEP8 will ensure that our Python code is consistently formatted, making it easier for team members to collaborate and for others to understand the code. This approach will help prevent common mistakes and promote clean, efficient coding practices across the team.

6.4 Expected Technology

Tool	**Version**	**Explanation**
Git	any	Version control system
GitHub	N/A	A tool for version control and project management
Python	TBD	Main programming language for our AI development
VSCode	Any	Code editor
LaTeX	Any	Documentation tool

Table 2: Expected Technologies

7 Proof of Concept Demonstration Plan

The primary risk of failure for this project is that our idea of using NLP to improve upon existing alternatives (such as MOSS) does not the increase the accuracy of code plagiarism detection. It is possible that our solution will perform worse on average than traditional tools.

Our team has received some confirmation from professors that specialize in the subject matter that the idea of using NLP to assign a similarity score to two pieces of text is feasible. There is also literature that demonstrates the feasibility of using cosine similarity as a metric for plagiarism detection. Therefore, our minimal proof of concept for this project will be that - a tool that utilizes a LLM (large language model) to assign a similarity score to two pieces of code. For the sake of our proof of concept, just one source language will be supported, and only two text files will be compared at once.

Once this is accomplished, we can evaluate if the similarity score is a good metric for determining if two pieces of code are plagiarised or not. When compared to MOSS, our POC solution should be able to produce similar detection rates. Our similarity scores should also make sense when validated by a human reviewer.

If the POC is unsuccessful, it demonstrates that an NLP-based approach is not a viable solution for source-code based plagiarism detection. If this turns out to be the case, the team will pivot to more traditional algorithm-based techniques for our source-code plagiarism detection project, which might entail improving upon existing tools like MOSS.

8 Project Scheduling

Task	Date
Team Formed, Project Selected	September 16
Problem Statement, POC Plan, Development Plan	September 24
Requirements Document Revision 0	October 9
Hazard Analysis 0	October 23
V&V Plan Revision 0	November 1
Proof of Concept Demonstration	Nov 25
Design Document Revision 0	January 15
Revision 0 Demonstration	February 3–February 14
V&V Report Revision 0	March 7
Final Demonstration (Revision 1)	
EXPO Demonstration	April TBD
Final Documentation (Revision 1)	April 2
- Problem Statement	
- Development Plan	
- Proof of Concept (POC) Plan	
- Requirements Document	
- Hazard Analysis	
- Design Document	
- V&V Plan	
- V&V Report	
- User's Guide	
- Source Code	

Appendix — Reflection

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

   A development plan provides a structured roadmap for the project, outlining key components like team roles, communication strategies, and workflow processes ensuring everyone is aligned on the goals, deliverables and timelines. Furthermore, it prevents scope creep by clearly defining the project's scope and allows for better resource allocation and risk management.

   It also promotes accountability and transparency among team members, as each member has a clear understanding of their responsibilities and the project's progress. Moreover, it serves as a guide that can be revisited throughout the project and be modified if needed ensuring that the project stays on the right track regardless of any challenges.

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

   One major benefit is that it enables rapid testing and deployment of changes, allowing the team to identify and rectify issues early in the development process. Automated testing ensures the code is working as expected and that the code quality is being mantained. This leads to faster feedback loops and higher productivity.

   However, it also comes with some challenges such as being complex to setup and mantain especially for large projects. If not managed well, it can lead to disruptions in the development process and cause delays. Although uncommon, another disadvantage is the over-reliance on automated testing, which leads to neglecting more thorough manual tests.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

   Our team didn't have any major disagreements which we needed to resolve. Some minor ones included disagreements about the structure and format of the documentation, specifically in the way certain processes were described and wether specific technologies should be used. They were resolved quickly though through open discussion where each team member talked about their concerns and we all collaboratively came to a consensus.

   In the end, we were satisfied with our final result.

Appendix — Team Charter

External Goals

Our team's external goals for this project are:

To create a state-of-the-art code plagiarism detector using ML and NLP that stands out from traditional tools like MOSS. To showcase the project at the Capstone EXPO and win a prize or recognition. To leverage this project as a highlight in job interviews to showcase our technical and teamwork skills.

Attendance

Expectations

All team members are expected to attend meetings on time. Leaving early or missing meetings should only occur in exceptional circumstances, and prior notice must be given whenever possible. Virtual attendance is allowed if in-person is not feasible.

Acceptable Excuse

Acceptable excuses include health issues, family emergencies, academic conflicts with prior notice, or unforeseen personal commitments. Unacceptable excuses are forgetfulness, lack of motivation, or missing meetings due to non-academic social activities.

In Case of Emergency

If a team member encounters an emergency and cannot attend a meeting or complete their assigned tasks, they should immediately inform the team via group chat (or other agreed communication tools) and arrange to cover or reschedule their tasks with another member if possible.

Accountability and Teamwork

Quality

Each team member is expected to come to meetings prepared, having completed their assigned work. The work brought to the team should be of high quality, thoroughly tested, and reviewed where applicable.

Attitude

Team members should approach each task with a positive attitude and be open to suggestions and feedback. Collaboration, respect, and a willingness to listen to others are key expectations. Everyone's ideas should be considered equally.

Stay on Track

To stay on track, the team will use weekly task management tools (such as Trello, Jira or GitHub Issues) to monitor individual progress. Regular check-ins will help ensure members are meeting their targets. If someone consistently underperforms, they will be required to explain their challenges to the team, and corrective actions will be taken, such as redistributing tasks or providing support. Consequences may include minor penalties like bringing coffee to meetings or scheduling a session with the TA or instructor. If a member performs exceptionally well, they may be rewarded with team recognition or other informal incentives like selecting the next team activity.

Team Building

The team will engage in fun activities like online gaming sessions or casual group lunches to build cohesion. Periodically, the team can also celebrate milestones with group outings or social events.

Decision Making

We will aim for consensus in decision-making. However, if disagreements arise and cannot be resolved quickly, we will proceed with a majority vote. For critical decisions, we will ensure that everyone has a chance to voice their opinion, and any unresolved issues can be escalated to the instructor or TA if necessary.