Development Plan
Software Engineering

Team #12, Streamliners
Mahad Ahmed
Abyan Jaigirdar
Prerna Prabhu
Farhan Rahman
Ali Zia

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
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?

This project does not involve the use or disclosure of confidential information from any industry partner. All requirements, specifications, and implementation details have been developed by the student team. As there is no industry-provided proprietary data or trade secrets involved, no Non-Disclosure Agreement (NDA) is required.

2 IP to Protect

All intellectual property (IP) generated as part of this project is owned by the student team, in accordance with McMaster University's policies on ownership of student work. Since the project is not sponsored by an external industry partner, no Intellectual Property Guide Acknowledgement form is required. To enable the intended use of the system, the student team will grant the McMaster Engineering Society a non-exclusive license to operate and maintain the software for its events and activities. Ownership of the underlying source code remains with the student developers.

3 Copyright License

All software developed by the student team is protected under copyright law. To clarify permitted use, the team has adopted the MIT License, which is included in the project repository here. This license allows reuse and modification of the code, provided that the original copyright notice and license terms are preserved. This ensures proper attribution to the development team.

4 Team Meeting Plan

[How often will you meet? where? —SS]

[If the meeting is a physical location (not virtual), out of an abundance of caution for safety reasons you shouldn't put the location online —SS]

[How often will you meet with your industry advisor? when? where? —SS]

[Will meetings be virtual? At least some meetings should likely be in-person. —SS]

[How will the meetings be structured? There should be a chair for all meetings. There should be an agenda for all meetings. —SS]

5 Team Communication Plan

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

6 Team Member Roles

[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

7.1 Git Usage

- Github will be used as the primary version control system (VCS).
- All commits must be made to a non-master branch and submitted via a pull request before they can be merged.
- Branching Strategy:
  - Main branch: Stable production-ready code waiting for a release.
  - Documentation branches: Used to maintain and update project documentation and requirements.
    - Notation: `docs/SL-<Issue#>`
  - Development branch: Integration branch where all new features and bug fixes are merged. This branch will contain the latest features with the most up-to-date code.
    - Notation: `dev`
  - Feature branches: Used to develop and test new features before merging into the development branch.
    - Notation: `feature/SL-<Issue#>`
  - Bugfix branches: Used to fix defects.
    - Notation: `bugfix/SL-<Issue#>`

7.2 Issues and Project Management

- Github issues will serve as the primary issue/ticket system
- All subsections of each document/report will have its own issue
- All new tasks and/or features will have its own issue

- Issues will be classified to using labels (e.g., `documentation`, `feature`, `bug`).
- Github's Kanban board will be used to monitor progress

7.3 CI/CD Usage

- A CI/CD pipeline will be implemented using Github Actions.
- Continuous Integration **(CI)**:
  - Automatically build the project and run tests on every pull request.
- Continuous Deployment **(CD)**:
  - Deploy staging builds after merging to development branch.
  - Deploy production builds after merging to main branch.

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

[What coding standard will you adopt? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

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

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

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

Appendix — Team Charter

External Goals

- Secure a grade of A+ in 4G06.
- Gain real-world experience in areas of project management, teamwork and shipping projects.
- Create a strong portfolio project that has a large user base and can be discussed in interviews.

Attendance

Expectations

- All members are expected to attend all scheduled meetings either in-person or online.
- All members are expected to be on time for meetings and remain present for the duration of the meeting.
- All members are expected to notify the team 24 hours beforehand if they are unable to attend meetings.

Acceptable Excuse

Acceptable excuses for missing meetings or deadlines include the following:

- Family emergency
- Health issues
- Medical appointments

Unacceptable excuses for missing meetings or deadlines include the following:

- Forgetting about meetings
- Oversleeping

In Case of Emergency

- If the emergency affects the members ability to attend a meeting, they must notify the group immediately or as soon as possible after the emergency.
- If the emergency affects completion of work, the member must communicate explicitly what work is done and what remains so the team can decide on the division of labour or a plan to catch-up.

Accountability and Teamwork

Quality

- All members must come to meetings prepared with progress updates on their tasks.
- Deliverables should be well-organized and reviewed before they are shared with the team.
- Contributions in the code repository must follow team standards for readability and documentation.

Attitude

- All members must listen actively to all other members and avoid passive or dismissive behavior.
- All members should respect all ideas and viewpoints without judgment.
- All members are encouraged to share their ideas openly.
- All members must adopt a simple code of conduct revolving on principles of respect, communication and collaboration.
- Conflicts must be resolved respectfully and if unresolved will be escalated to the TA.

Stay on Track

- Weekly progress check-ins where members discuss what they completed and next steps will be used to keep the team on track.
- GitHub Issues and commits will be used to ensure members contribute as expected.
- Metrics:
  - An attendance rate of 100% is expected from all members unless an excuse is deemed valid and acceptable by the team.
  - Consistent GitHub activity is expected per week with issues in progress or items being committed for review.
  - On-time delivery of tasks on assigned deadlines.
- Rewards:
  - Members who reach targets early or who do well overall will be rewarded with acknowledgments of being strong contributors.
- Consequences:

  - First time incident will be a verbal reminder.
  - Second time will result in a discussion with the TA.
  - Third time and more will result in escalation to the course instructor.

Team Building

- Start meetings with small talk and conversations to get to know each other and maintain team connection.
- Doing a group ritual of small activities like board games or lunch to celebrate milestone deliveries.

Decision Making

- Initial plan will be to reach consensus when possible.
- If consensus cannot be reached, opt for a majority vote.
- If disagreements still exist within the team if consensus and majority vote leaves members unsatisfied, involve the TA for handling disagreements.