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

The team will adopt consistent coding standards to ensure readability, maintainability, and effective collaboration. For JavaScript code, the team will follow the Airbnb JavaScript Style Guide, which provides well-established conventions for formatting and best practices. Code formatting will be automated with **Prettier**, and linting will be enforced with **ESLint** to reduce inconsistencies across contributions.

Naming conventions will remain uniform across the stack:

- `camelCase` for variables and functions
- `PascalCase` for React components and NestJS classes
- `snake_case` for PostgreSQL tables and column names

Beyond style rules, the team will apply **Clean Code principles** to keep the codebase simple and easy to understand. This means writing small and focused functions, choosing descriptive names, avoiding duplication, and keeping modules cohesive. Comments will be added only where necessary to clarify complex logic, with the preference being self-explanatory code.

For version control, all work will be done in feature branches, with pull requests reviewed before merging into `staging` or `main`. Commit messages will follow the imperative style (for example, "Add payment processing logic") to maintain a clear project history.

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

[borrows from University of Portland Team Charter —SS]

External Goals

[What are your team's external goals for this project? These are not the goals related to the functionality or quality fo the project. These are the goals on what the team wishes to achieve with the project. Potential goals are to win a prize at the Capstone EXPO, or to have something to talk about in interviews, or to get an A+, etc. —SS]

Attendance

Expectations

[What are your team's expectations regarding meeting attendance (being on time, leaving early, missing meetings, etc.)? —SS]

Acceptable Excuse

[What constitutes an acceptable excuse for missing a meeting or a deadline? What types of excuses will not be considered acceptable? —SS]

In Case of Emergency

[What process will team members follow if they have an emergency and cannot attend a team meeting or complete their individual work promised for a team deliverable? —SS]

Accountability and Teamwork

Quality

[What are your team's expectations regarding the quality of team members' preparation for team meetings and the quality of the deliverables that members bring to the team? —SS]

Attitude

[What are your team's expectations regarding team members' ideas, interactions with the team, cooperation, attitudes, and anything else regarding team member contributions? Do you want to introduce a code of conduct? Do you want a conflict resolution plan? Can adopt existing codes of conduct. —SS]

Stay on Track

[What methods will be used to keep the team on track? How will your team ensure that members contribute as expected to the team and that the team performs as expected? How will your team reward members who do well and manage members whose performance is below expectations? What are the consequences for someone not contributing their fair share? —SS]

[You may wish to use the project management metrics collected for the TA and instructor for this. —SS]

[You can set target metrics for attendance, commits, etc. What are the consequences if someone doesn't hit their targets? Do they need to bring the coffee to the next team meeting? Does the team need to make an appointment with their TA, or the instructor? Are there incentives for reaching targets early? —SS]

Team Building

[How will you build team cohesion (fun time, group rituals, etc.)? —SS]

Decision Making

[How will you make decisions in your group? Consensus? Vote? How will you handle disagreements? —SS]