Development Plan
ProgName

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

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?

[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

[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

- 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

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

We aim for our project to be well documented and easy to understand. Git will be used for version control alongside GitHub for project management and CI/CD. For Git commit messages, we will follow the conventional commits specification at https://www.conventionalcommits.org/en/v1.0.0/. The provided issue templates will be used and additional templates will be added for bug reports and feature requests to ensure consistency in issue reporting. Since the TPG project is primarily written in C++, we will adopt the Google C++ Style Guide. For LaTeX documents, we will be following the file structure provided with the template and maintain consistent formatting conventions. Unit tests must be provided for all non-trivial functions and classes and integrated with the CI/CD pipeline.

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?

   Creating a development plan prior to starting the project is important because it helps in outlining the project's scope, objectives, and deliverables. It ensures that all team members are aligned with the project's goals and provides a roadmap for the project's execution. This planning phase helps in identifying potential risks and challenges early on, allowing the team to devise strategies to mitigate them. Additionally, a well-structured development plan facilitates better resource management and timeline estimation, which are crucial for the successful completion of the project.

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

   The advantages of using CI/CD include automated testing, which helps in identifying bugs early in the development process, and ensuring that the codebase remains stable and functional. CI/CD also promotes repeatable and consistent testing environments, reducing the chances of discrepancies caused by different development setups. For example, our team members own devices with different operating systems, so having a CI/CD pipeline ensures that the project can be tested consistently across all devices. Moreover, it can block pull requests until tests pass, ensuring code quality. However, the disadvantages include the potential for CI/CD configurations to become complex and time-consuming to set up and maintain. Additionally, if not managed properly, CI/CD processes can consume significant computational resources, leading to increased costs and longer build times.

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

Appendix — Team Charter

External Goals

Our team has several external goals for this project. Firstly, we aim to present a standout project at the Capstone EXPO that showcases our technical abilities and innovative thinking. Winning a prize at the EXPO would be a significant achievement and a testament to our hard work. Secondly, we want to develop a comprehensive project that serves as a strong talking point in future job interviews, demonstrating our skills and experience to potential employers. Lastly, we aim to achieve an A+ in this course, reflecting our dedication and the quality of our work.

Attendance

Expectations

Our team expects all members to arrive on time for meetings, with a grace period of up to 5 minutes. Consistent lateness should be explained, and team members are expected to attend at least 80 percent of all meetings. Attendance will be tracked on GitHub, including team meetings, meetings with our supervisor, and TA. Active participation in discussions and decision-making processes is more important than mere attendance.

Acceptable Excuse

Acceptable excuses for missing a meeting or a deadline include medical emergencies, pre-approved commitments (with prior notification to the team), and technical difficulties such as internet outages or device failures. Unacceptable excuses include lack of interest in the project, poor time management, and procrastination.

In Case of Emergency

In case of an emergency, the team member must inform the team as soon as possible and provide an acceptable excuse. They should also give an estimated time frame for resolving the emergency and propose a contingency plan for covering their responsibilities.

Accountability and Teamwork

Quality

Our team expects high-quality preparation for meetings and deliverables. Each member should come prepared with their assigned tasks completed to the best of their ability, ensuring that all contributions are thorough and well-researched. Moreover, we will check each other's work through assigning reviewers to each

other's pull requests. The review process will be based on the provided rubric to ensure that all contributions meet the required standards.

Attitude

Team members are expected to contribute positively to the team environment, respecting each other's ideas and cooperating fully. We will adopt McMaster's student code of conduct to ensure respectful interactions. For example, Appendix A in the Code of Student Rights and Responsibilities requires students to refrain from using derogatory or offensive language, maintain a positive and respectful attitude, and refrain from engaging in any form of harassment or discrimination. If a team member violates these rules, the team will address the issue through a formal appeal process. We expect all team members to adhere to these standards at a bare minimum. What we hope to foster is an environment where team members feel inspired to contribute their best work and support each other in doing so.

Stay on Track

To keep the team on track, we will use project management metrics such as attendance, commits, and issue tracking. These metrics will be used as a rough indicator of team member's contributions. We recognize that these metrics are not perfect. For example, deleting 100 lines of code is often more productive than writing 100 lines of code, but this is not reflected in the commit count. We will also consider other factors such as the complexity of the changes made and the number of reviews a team member has conducted. Nonetheless, members who consistently meet or exceed their targets will be recognized and rewarded, while those who fall short will face consequences such as bringing coffee to the next meeting or scheduling a meeting with the TA or instructor. We will set clear targets for attendance and commits, and ensure that everyone is aware of their responsibilities.

Team Building

To build team cohesion, we will schedule regular fun activities and group rituals, such as team lunches or game nights. These activities will help strengthen our bond and improve our collaboration.

Decision Making

Our team will make decisions through consensus whenever possible. If consensus cannot be reached, we will hold a vote. In case of disagreements, we will follow our conflict resolution plan, which includes discussing the issue openly and seeking input from our supervisor or TA before making a final decision.