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

Meeting Frequency

- Regular weekly meetings:
  - One in-person meeting during scheduled tutorial time
  - One online meeting via Discord (time TBD by team availability)
- Additional ad hoc meetings scheduled as needed for deadlines or unblocking issues
- Bi-weekly supervisor meetings with Dr. Paul Rapoport (virtual or in-person)

Meeting Format

- Primary format: Virtual meetings via Discord (voice channel)
- Secondary format: In-person meetings during class/tutorials

- Meeting roles (rotating):
  - **Chair**: Responsible for
    - Preparing and sharing agenda (12+ hours in advance)
    - Managing time allocation
    - Facilitating inclusive discussion
  - **Note-taker**: Records
    - Key decisions
    - Action items with owners and deadlines
    - Risks and open questions

Attendance Policy

If unable to attend:

- Notify team via Discord in advance
- Review meeting notes independently
- Complete assigned tasks asynchronously

5 Team Communication Plan

We will use multiple coordinated channels to ensure clear, persistent, and efficient communication:

- **GitHub**: Version control, issue tracking, pull request review, and Kanban board management.
- **Discord**: Primary platform for quick text and voice communication. Daily informal updates, coordination of short-term tasks, and quick problem-solving occur here. Urgent blockers are first raised in Discord.
- **In-Person (Tutorial / Scheduled Check-ins)**: Used for structured milestone work, live demonstrations, and supervisor touchpoints.

6 Team Member Roles

Roles are rotational to ensure balanced workload distribution, broaden experience, and encourage shared ownership. Anticipated roles include:

- **Team Liaison (Ammar Sharbat)**: Primary point of contact with the supervisor and course staff; coordinates meeting requests and relays external feedback.
- **Developer (All members)**: Implements assigned features, writes and maintains tests, and updates related documentation.

- **Reviewer (All members)**: Performs code reviews for pull requests, checks alignment with coding standards, and leaves comments if needed before approval.
- **Meeting Chair/Note-taker (All members rotate)**: Facilitates meetings, ensures agenda is followed, and documents key discussion points and action items.

7 Workflow Plan

1. **Update Local Repository**: Before starting any work session, pull the latest changes from the ‘main‘ branch to ensure the local repository is up to date.
2. **Branching**: Create a new feature (or fix) branch from ‘main‘ for each unit of work (feature, bug fix, refactor). Follow a consistent naming convention such as: feature/short-description or fix/issue-id.
3. **Coding**: Implement modules and functions according to the design and requirements. Follow the agreed coding standards and document code with comments where necessary.
4. **Unit Testing**: Write and execute unit tests for the newly implemented modules/functions to verify expected behaviour early. Tests must pass locally before proceeding.
5. **Commit and Push**: Commit changes with descriptive messages. Push to the remote feature branch.
6. **Review and Merge**: Open a pull request into ‘main‘. Another team member reviews for correctness, style, test coverage, and documentation updates. The reviewer will leave comments and suggestions if needed. After at least one approval, the branch is merged into the main branch.

All project tasks are tracked through GitHub Issues and a Kanban board using GitHub Projects (Backlog / In Progress / Review / Done).

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

[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

- **Meeting Preparation:**
  - Read the agenda and relevant issues/PRs prior to the meeting.
  - Come prepared with (a) a concise progress update, (b) explicit blockers, (c) questions to ask.
  - For design/architecture discussions, read any documentation shared at least 12 hours before the meeting.
- **Expectation for a code task / issue:**
  - Code compiles/runs locally without new warnings or linter errors.
  - Automated tests added or updated: cover new logic and edge cases.
  - All tests pass locally.

  - Documentation updated: inline comments for non-obvious logic; README/module documentation updated if behaviour, interfaces, or run steps change.
  - Peer review completed (minimum one approval) with review comments addressed or explicitly deferred via a follow-up issue.

- **Timeliness:** Work items should be completed within their originally estimated iteration window.

Attitude

- **Respect and Inclusion:** Listen actively and avoid interrupting. Credit ideas to originators; disagreements focus on the idea, never the person.
- **Communication Responsiveness:** Weekday Discord messages acknowledged within 24 hours (a reaction or short reply). If unavailable (midterms, travel), communicate ahead of time if possible.
- **Constructive Feedback:** Use specific, friendly, actionable language and pair it with reasoning.
- **Conflict of Ideas:** Healthy debate is encouraged, and once a decision is recorded, team members support it unless new evidence emerges.
- **Inclusivity:** Zero tolerance for harassment, discrimination, or disparaging language. Maintain a commitment to inclusivity across backgrounds and experience levels.

Conflict Resolution Process

1. *Direct Discussion:* Involved members attempt a private, respectful conversation to clarify intent and desired outcome.
2. *Mediated Conversation:* If unresolved, bring the issue to the next meeting or request a teammate to facilitate a short discussion.
3. *Escalation:* If behaviour breaches the Code of Conduct, escalate to the supervisor / TA. Persistent issues may be elevated to the instructor.

Stay on Track

The team will use clear metrics and regular check-ins to ensure progress and accountability.

Metrics Tracked

- Meeting attendance (tutorials, scheduled meetings, supervisor check-ins)
- Number of pull requests made and contributions to reviewing PRs

Recognition Positive, on-time, high-quality contributions are acknowledged verbally in weekly meetings. Reusable good practices or discoveries are shared for team-wide adoption.

Managing Underperformance

1. Check-in to clarify blockers (scope, skill gap, time constraints).
2. Adjust: refine issue scope, pair program, or provide targeted resources.
3. If no improvement and no proactive communication: document an action plan with achievable concrete milestones.
4. Continued gaps will need to be escalated to TA / instructor for guidance.

Consequences If deadlines are repeatedly missed without notice:

- Action plan logged (issue comment or team log) with dates.
- Escalation path: (1) team check-in, (2) written plan, (3) supervisor/TA, (4) instructor if unresolved.

Team Building

Lightweight cohesion practices:

- Open casual discussion on any topic for the first 5 minutes of each meeting.
- Celebrate small wins (merged PRs, resolved issues) in meetings.
- Share interesting articles, tools, or tips related to the project in Discord.

Decision Making

1. Open discussion aiming for consensus.
2. If no consensus, move on to a simple majority vote.
3. Record decision (issue comment, PR, or decision log); set as the final decision until new evidence justifies a change.