Development Plan
ScoreGen

Team #7, Tune Goons
Emily Perica
Ian Algenio
Jackson Lippert
Mark Kogan

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

Issue Management Workflows
When creating an issue in the Capstone Kanban Board, there are two possible workflows:

1. Meeting issue is created (using the relevant template) prior to a scheduled meeting.
    (a) Issue is given the 'meeting' label and assigned the 'Meeting Minutes' status.
    (b) The notetaker (see section 6) will add a comment to the issue, outlining the meeting minutes and action items of each attendee.
    (c) Issue is closed once all action items have been addressed.
2. Project work issue is created using a blank issue.
    (a) Issue begins in 'To Do'
    (b) One or more appropriate labels are assigned - 'documentation', 'deliverable', 'enhancement', 'bug', and potentially others as the need arises.
    (c) If the issue does not have the 'deliverable' label AND is not time sensitive, it will be moved to 'Backlog'.
        i. The issue will move out of 'Backlog' and into 'To Do' if the issue becomes time sensitive or the team has the time/resources to take it on.
    (d) The issue moves to 'In Progress' once it is assigned and being worked on. It may be assigned to more than one person, depending on the scope of the issue.
        i. At this point, an issue pertaining to the codebase will branch to the Git Workflow (below).
    (e) The issue moves to 'In Review' once a PR has been made and approval is requested from the team.
    (f) The issue moves to 'Done' and can be closed once all associated tasks have been completed, including all related PRs being merged/closed.

In addition to the above usage of the Kanban board, all course deliverables are documented as milestones in the repository and will have issues assigned as necessary. Each milestone has a due date and a description of the tasks to be completed.

The Project Manager (see section 6 for team member roles) is responsible for making issues and keeping the project on track. As well, they will run monthly backlog grooming sessions with the team to ensure any stale issues are taken care of.

Git/GitHub Workflow
A new developer on the project will begin at step 1; otherwise step 1 can be skipped:

1. Developer may choose to fork or clone the repo. A fork is recommended so that contributors can display the project directly in their GitHub profile.
2. Main is a protected branch - any commits must be made in a separate branch before being pushed to a PR. There is no specific naming scheme for branches.
3. Commit and PR names will both start with 'Issue #: '. This will be enforced through a GitHub Action.
   (a) Each PR will only address a single issue, but an issue may be spread out across multiple PRs.
   (b) Squash commits are enforced to ensure there is a single commit per PR.
4. At least one reviewer must approve the PR before it can be merged into main.

CI/CD Workflow
Actions on Push/Pull Request:

- LaTeX to PDF converter
- Code packaged as an executable
- Linting
- Unit tests
- PR naming convention enforcer

Note that a PR may not be merged if any failures are present in the pipeline.

8 Project Decomposition and Scheduling

Due Date	Deliverable
Sept 23/24	Problem Statement, POC Plan, Development Plan
Oct 9/24	Requirements Document Revision 0
Oct 23/24	Hazard Analysis
Nov 1/24	V&V Plan Revision 0
Nov 11/24*	Proof of Concept Demonstration
Jan 15/25	Design Document Revision 0
Feb 3/25*	Revision 0 Demonstration
Mar 7/25	V&V Report Revision 0
Mar 24/25*	Final Demonstration (Revision 1)
Apr ##/25 - TBD	Expo Demonstration
Apr 2/25	Final Documentation (Revision 1)

*The demonstration should be prepared by this date, but this will not necessarily be the date of the demo itself.

Ideally, progress on a deliverable will begin at least 3 days before the previous deliverable is due (i.e., D2 work should begin on or before Sept 20). The above schedule will be enforced through the use of Milestones in the project repository, such that each deliverable (including its due date and all relevant information) is its own Milestone. The only exception to this is the Expo Demonstration, which is not expected to require any specific issues.

Each deliverable will be split into issues based on dependencies and relevant tasks. For example, when writing this document a single issue consisted of sections 7 and 8. They both cover project management-related plans, so completing one section provides the writer with the needed knowledge to write the other section. An alternative proposed method was to create an issue for each section of the document, but the granularity of this strategy may make the document feel more disjointed to the reader.

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

[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]