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

7.1 Git Workflow

Git will be the main version control system for this project. The main code of the Tangled Program Graphs (TPG) framework is stored in GitLab. However, due to the nature of the Capstone project, the team will be cloning the repository as a subtree of the Github repository to implement the two main goals of this project: support software engineering best practices and the integration of the machine learning framework into another agent-environment. The Gitlab version will be pulled into Github and the Github version changes will be pushed into Gitlab frequently to keep the repositories' commit history in sync. To redirect the development of TPG to support standard software engineering principles, the Github repository will use Github Actions for running Continuous Integration checks, different kinds of comprehensive testing, and generative documentation.

The general outline of the workflow is as follows:

1. Pull latest changes from the **main** branch
2. Create a feature branch
3. Implement code changes
    (a) Add and perform unit and integration testing
    (b) If applicable, add comments and documentation
4. Commit changes with descriptive messages
5. Open a pull request with the **main** branch as base
    (a) Provide detailed description of changes and testing performed
    (b) CI pipeline will be triggered via GitHub Actions
        i. If CI pipeline fails, fix the issue and revert back to Step 3
    (c) Other team members check if changes satisfies the acceptance criteria
6. Merge changes using **Squash and Merge** method

7.2 Issues

Github Issues will be used to monitor tasks and their progress. New templates have been created within the project to better distinguish each issue. These templates can be used to maximize team collaboration, task breakdown and scheduling. The workflow for leveraging issues is as follows:

1. Create an issue via Github's Issue dashboard
2. Once work begins for the issue, update its status to 'In Progress'
3. If the task requires a code change, follow **7.1 Git Workflow** section
4. Once issue is ready for review, update its status to 'In Review'
5. After the issue meets the acceptance criteria, update its status to 'Closed'

7.3 Milestones

Github Milestones will be used to group issues and pull requests to track the overall progress of the project. This would allow the team to analyze small increments of their progress and determine if the project is on track to be completed. Since Github Milestones can have deadlines, the team will assess every milestone once the deadline has been reached to gather feedback for iterative improvement.

7.4 Labels

Github Labels will be used to group different types of tasks and pull requests. These can be used to further categorize the priority and nature of each task, allowing the team members to distinguish their criticality throughout the duration of the project.

The following are labels that will be used in the future. These labels may be modified or replaced throughout the lifecycle of the project:

- feature - related to new features
- bug - related to software bug fixes
- documentation - related to documentation
- high-priority - important task and requires immediate attention
- medium-priority - important task but not urgent
- low-priority - not important task and lowest in priority

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

For the project's proof of concept demonstration plan, some significant risks may occur while in development and hinder the project's success. Here are the most significant risks:

- **GitHub / GitLab integration:** As of now, the TPG repository is currently on GitLab; however, GitHub is required to be utilized for this project. This TPG repository is still currently being contributed by both Dr. Kelly and graduate students. Failure to be able to integrate both changes could deal with some serious merge conflict issues.
- **Code Robustness:** Due to the large codebase and current unfamiliarity of the code, difficulties in testing may occur and the team may be unable to gather large code coverage within our testing suite. This will delay the goal of establishing a CI/CD environment for TPG and cause major hurdles in the remainder of the project.
- **Time-Consuming Refactoring:** A part of this project includes refactoring the code to follow standard software engineering principles, as this code is currently being modified throughout the project, refactoring may be a significant time-consuming task, preventing the completion of the other project goals.
- **Programming Language Learning Curve:** As most of the team for this project have not had experience with the C++ programming language, it will require some additional learning. This may slow down the progress of our project and prevent the proof of concept demonstration plan from being achieved.
- **Unable to create compatibility with the MuJoCo environment:** The project goal in the end is to allow the TPG code to be simulated within the MujoCo environment. If we are somehow unable to do this, we may need to reroute the focus of our project.

If these significant risks can be prevented or overcome for this project, then the proof of concept demonstration shall include the following to demonstrate the overcoming of the risk:

1. The overcoming of TPG integrated into a MuJoCo simulation environment by a basic demonstration of CartPole.

2. Walkthrough the CI/CD pipeline of the TPG project.

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]