Development Plan
CVT Simulator

Team #17, Baja Dynamics
Grace McKenna
Travis Wing
Cameron Dunn
Kai Arseneau

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept 24	All	Initial Draft
Nov 10	Grace McKenna	Updated Proof of Concept Demonstration Plan

This document provides a summary of how we plan to approach this project. It covers both the managerial and technical aspects of the project. On the management side, we discuss how we will communicate, organize ourselves and divide up responsibilities. On the development end, we list out our POC goals and what tools we expect to use. Overall, this document serves as a summary of how we will accomplish this project.

1 Confidential Information

This project does not contain any confidential information.

2 IP to Protect

We do not have any IP to protect.

3 Copyright License

4 Team Meeting Plan

The team will meet in person on Mondays between 2:30-4:30pm on a weekly basis. Additionally, the team will plan to meet virtually or in person as the team sees fit on Fridays at 1:30-2:30pm. Additional meetings may be scheduled as needed, based on the current status of the team's progress towards upcoming course deadlines. All meetings held will have corresponding GitHub issue templates that are to be filled out by the meeting chair. The issues will serve as documentation of the meeting and contain the meeting agenda. The role of the meeting chair will rotate depending on the content being covered during the meeting. The default role of meeting chair will be designated to Cameron Dunn.

5 Team Communication Plan

The team will use a structured Discord server with specific channels to organize conversations and content. This discord server has structured channels separating scheduling, resources, meetings and general questions and communication. Additionally, the team will communicate through the use of GitHub issues and commits. The team will enhance communication around GitHub issues by implementing a clear labeling system. Labels will help distinguish between different types of tasks, such as separating backend and frontend issues,

ensuring better organization and prioritization across the project. The team will also use GitHub Projects to communicate the status of issues.

6 Team Member Roles

The team has decided to give specific roles for both deliverables as well as meetings. The first role given to each team member corresponds to their meeting role and the second is their deliverable role.

Grace McKenna

- Note taker - The responsibility of this role is to take notes based on meeting discussion. These notes are then uploaded to the corresponding meeting GitHub Issue as a comment.
- Project Manager - The responsibility of this role is to manage the project timeline and resource allocation.

Travis Wing

- Issue manager - The responsibility of this role is to review and present the issues that are related to the meeting.
- Frontend Lead - This role is responsible for managing the implementation of the frontend of the application.

Cameron Dunn

- Meeting chair - The responsibility of this role is to facilitate meeting discussion and ensure all points on the meeting agenda are covered.
- Product Owner - The responsibility of this role is to ensure that the product meets it's requirements.

Kai Arseneau

- Reviewer - The responsibility of this role is to review the ideas discussed and evaluate their quality.
- Backend Lead - This role is responsible for managing the implementation of the backend of the application.

The roles listed above relate to decision making and organization, but all team members are expected to contribute to each aspect. This is to ensure all team members have knowledge about all parts of the project.

7 Workflow Plan

7.1 Branches

The team will use three categories of branches:

1. Main branch
   This branch is a unique branch which holds the production version of the application. Merging into this branch requires unanimous approval (mandatory 3 code reviews). Version updates will be merged into this branch from the develop branch.

2. Develop branch
   This branch is a unique branch which contains the most recent version of the project. Merging into this branch requires at least one approval from another team member. Version updates will come from merging releases from this branch into main.

3. Feature branches
   These branches relate to individual features that are currently being developed. These branches will branch off and merge into the develop branch.

7.2 Issues

The team will use issue tags to help organize issues by their category. The issue tags are:

- Bug
- Enhancement
- Refactor
- Documentation
- Testing
- Backend
- Frontend
- Meeting

Additionally, for making new issues we set up issue templates. The templates we will use:

- Bug report
- Feature
- Lecture

- Peer Review
- Supervisor Meeting
- TA meeting
- Team meeting

7.3 CI/CD

For this project, the team will use continous integration for automatic testing and code quality checks via Github Actions. Integration tests will be performed on full features and then run on each PR for regression testing.

Continuous deployment does not apply to this project as it will be a standalone application. New releases will be released on Github.

8 Project Decomposition and Scheduling

8.1 Project Schedule

Team Formed, Project Selected	September 16
Problem Statement, POC Plan, Development Plan	September 24
Requirements Document Revision 0	October 9
Hazard Analysis 0	October 23
V&V Plan Revision 0	November 1
Proof of Concept Demonstration	November 11–22
Design Document Revision 0	January 15
Revision 0 Demonstration	February 3–February 14
V&V Report Revision 0	March 7
Final Demonstration (Revision 1)	March 24–March 30
EXPO Demonstration	April TBD
Final Documentation (Revision 1)	April 2
- Problem Statement	
- Development Plan	
- Proof of Concept (POC) Plan	
- Requirements Document	
- Hazard Analysis	
- Design Document	
- V&V Plan	
- V&V Report	
- User's Guide	
- Source Code	

8.2 GitHub Projects

The team will be using a GitHub project titled CVT Simulator Planner. This project will track items based current existing items to be assigned, items that are in progress and completed items. Each item is intended to have a description and, once in progress, the name of the team member/members who will be working on the item. The goal of this GitHub project is to ensure all team members are on the same page regarding what needs to be started for upcoming project deliverables and project features. Team members are also able to see which items are in progress, helping communicate what team members are actively working on. This will increase communication between team members, as everyone can see what their colleagues are currently working on. This project also has a section to show the items that have been completed.
GitHub Project Link: https://github.com/users/gr812b/projects/1

9 Proof of Concept Demonstration Plan

Our main concern for this project is handling the two-way communication between the mathematical model and the Unity animation, as well as working with unfamiliar tools such as ODE solvers and Unity. Our proof of concept will aim to have a GUI in Unity send and receive data from a python script, which plays the role of the mathematical model. The GUI will have a setting for a feature which will be sent to the python script. Once received, the python script will perform a calculation using the ODE solver and send the result back to Unity. When Unity receives the result, a corresponding animation will be displayed. This will be a sufficient proof of concept as once the team is able to complete this task the rest of the project will expand on this existing system.

One risk involved with this plan is ensuring the two way communication between the Unity frontend and the Python backend as these platforms require distinct handling protocols. To mitigate this we will implement a phased integration strategy starting with a simple data exchange and gradually incorporating more complex interactions. Additionally another risk is working with ODE solvers which would require the team to acquire new knowledge and skills. To mitigate this risk we plan to read documentation of existing Python libraries as well as set up meetings with Dr. Smith for any clarification as needed. We will also start with solving a simple ODE and incrementally increase the complexity. Both of these risks will require a large time investment and continuous attention to ensure our proof of concept is successful. However, the mitigation strategies discussed above will allow our team to have a successful proof of concept with minimal setbacks.

10 Expected Technology

- Python

  - Flake8
  - Black
  - Math
  - Numpy
  - Matplotlib
  - Scipy.integrate.solve_ivp
  - Unittest
- C#
  - SonarLint
  - StyleCop
  - Pythonnet
- Unity
  - Unity test framework
- GitHub
  - GitHub Projects
  - Version Control: git
- 3D model - CAD
  - Solidworks
- Microsoft Excel
- VS Code
- The Data Viewer - pre-existing website to graph and view data, designed by McMaster's Baja Racing Team.
- Specific unit testing framework
- Investigation of code coverage measuring tools

11 Coding Standard

The team will follow the PEP 8 coding standards for Python.
https://peps.python.org/pep-0008/
The team will also adhere to the C# coding standards outlined in the Unity standards.
https://unity.com/how-to/naming-and-code-style-tips-c-scripting-unity

Appendix — Reflection

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?

   It is important to create a development plan prior to the start of large team projects, as it helps set the tone for the project, organize and breaks it down into smaller tasks and outlines individual responsibilities. The development plan also highlights key deadlines and identifies potential risks, ensuring that mitigation strategies are discussed in advance. Additionally, a development plan aids in the success of team communication by specifying the frequency of meetings, methods of communication, and other measures to enhance collaboration. Ultimately, a development plan acts as a roadmap that keeps the team on track throughout deliverables, supports effective problem solving, and supports risk management.

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

   The advantages of using CI/CD are for large projects where the needs of the users are constantly changing because they makes it easier to test and deploy code quickly and efficiently. CI helps to automate testing and integration, while CD helps to automate the deployment of code. This helps on large projects where the can be many systems interacting with each other in complex and unpredictable ways. On the other side, the disadvantages are more felt on smaller projects where the needs of the users are more static. In these cases, the overhead of CI/CD slows down development more than it helps. Additionally since these projects are smaller, it is less helpful due to the system being more predictable.

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

   For this deliverable we had some disagreements about how we should update the Revision History. We had an even split 50/50 on whether we should update it on each commit or each major revision. We resolved it

by asking the professor for clarification at our next meeting. The professor recommended that we update it on each major revision, so we decided to go with that.

Appendix — Team Charter

External Goals

- The team wants to be able to score higher at Baja competitions.
- The team wants to be successful within the course.
- The team wants to have an interesting presentation at the Capstone EXPO.
- The team wants to be able to show their employers they can solve complex problems on a team within a short time span.

Attendance

Expectations

All team members are to communicate regarding their attendance of team meetings. If a team member is unable to attend a meeting, the meeting will either be rescheduled, depending on the importance of the content being covered in the meeting, or the team member will be debriefed by another team member. Team members are expected to communicate with one another if they will be late or need to leave the team meeting early.

Acceptable Excuse

Acceptable excuses for missing a meeting or deadline without prior notice include:

- Sudden illness or injury requiring medical attention
- Family emergency needing immediate attention
- Unexpected technical difficulties
- Other extreme unforeseen circumstances

With prior notice (>24 hours before), it is acceptable to miss a meeting for any scheduled event or obligation that cannot be rescheduled such as:

- Medical appointments
- Planned family events or support
- Academic obligations (exams, projects, presentations, etc.)
- Other scheduled events

It is unacceptable to miss a meeting or deadline outside of the above circumstances.

In Case of Emergency

In a case of emergency where a team member cannot deliver on their responsibilities, the team will handle redistributing that work. In the case of a meeting, one or more team members will assume the responsibilities of the missing team member and will debrief them afterwards. In the case of a deliverable, the team will redistribute what cannot be completed by the missing team member to the rest of the team.

Accountability and Teamwork

Quality

The process for addressing teammates who do not meet expectations will be as follows. The team will meet with the individual who did not meet the expectations and address why the expectations have not been met yet. The team member will be given an additional day to complete the work if this is time permitted.

Attitude

Team members will always treat one another with respect and have a positive attitude towards one another. All team members must be respectful when listening to others ideas.

Stay on Track

Team meetings as well as the GitHub project board, issues and commit history will be used to monitor each team member's progress. At each meeting, the team will discuss the progress of each member and evaluate if they are on track to meet the project deadlines. If a team member is found to not be meeting their expectations, the team will discuss what can be done to get them back on track in the regular meeting. If the issue continues, a special meeting will be scheduled to understand that member's problems and how to resolve them. If after this the issues still persist then the team will meet with a TA or the professor to address the team member's failure to meet their responsibilities.
To incentivize performance, control over decision making of team celebration will be given proportionally to the quality and timeliness of work completed by each team member.
The metrics used to measure a team performance will be their attendance at meetings, the subjective quality of their work and how well they are able to meet deadlines. Other quantitative metrics such as commits and lines of code will only be used to supplement the above metrics or in extreme cases such as having no commits or lines of code.

Team Building

The team will celebrate their accomplishments through team dinners or other social events.

Decision Making

When there are disagreements, the team will meet to discuss the issue until an absolute majority (>50%) of the team agrees on a solution.