Development Plan
Student Evaluation App

Team 29
Nicholas Fabugais-Inaba
Casra Ghazanfari
Alex Verity
Jung Woo Lee

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 23, 2024	NFI, JL, CG, AV	Initial Draft
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]

Our project has no confidential information from industry.

[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]

There is no IP to protect.

3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

We are adopting the GNU General Public License. It is saved in the root file under filename LICENSE.

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]

- Jung Woo Lee
  - Scrum Master
  - Developer
- Alex Verity
  - Developer
- Nicholas Fabugais-Inaba
  - Developer
- Casra Ghazanfari
  - Developer

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

Git is the most important part in managing issues, editing documentation, and developing features. Before any changes should be made, a "git pull" should be entered into the Github capstone directory for updating the local repository to the latest changes on the remote branch. Once this is complete, a branch should be created, from the main branch, for any commits involving documentation, development code, or any other changes present within the repository. With these commits, there should be comments detailing the changes that have been made. Furthermore, each commit should have the designated code name associated with it. After these commits are tested, committed, and then pushed from the local to the remote repository, a pull request should be made to merge the branch commit to the main branch. At least one approval will be required from a reviewer for each pull request to be merged with the main branch. Finally, when the commits have been successfully merged to the main branch, the branch the commits were made on will be deleted.

For managing merge conflicts, the developer should identify the conflicting changes, making sure to resolve any conflicts before new changes are made. Once all conflicts are resolved, the developer should be able to proceed in testing, committing, and pushing their changes to the remote repository.

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]

Sourced from the article:
A complete Guide to Coding Standards and Best Practices

- When using JavaScript:
  - Use camel case (exampleVariable) for variables and functions.
  - Use pascal case (ExampleClass) for classes.
  - Include semi-colons at the end of each statement.
- When using Python:
  - Use snake case (example_variable) for variables and functions.
  - Use pascal case (ExampleClass) for classes.
- When using any language:
  - Use four spaces when indenting.
  - Use whitespace to separate functions and code blocks for readability.
  - Include comments where applicable, in areas where code may be unclear.
  - Use variable and function names that describe the use of the function.
  - Limit the use of global variables wherever possible.

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?

Reflection – Alex Verity

It is important to create a development plan prior to starting a project to allow the team to get on the same page and form a line of communication. By setting standards and rules early it avoids unnessecary conflicts. It is also important as a way to fasttrack important decisions like a workflow plan and expected technologies.

Some advantages of continuous integration are that teams can find errors quickly and often, which can stop an error snowballing or hiding under the radar until it reaches the user. It makes the development of software more predictable and errors easier to find, as you are typically searching for them in smaller chunks of software rather than the entire code base. Some disadvantages of continuous integration are that it requires significant effort to set up and modify if needed. This overhead in some cases is not worth the gains, and it is up to the developer to decide this. As with all automated testing, some more complicated errors can fall through the cracks, so manual testing is still needed.

In most things the team was in agreement, but whether or not to use GitHub's label feature for issues was a small disagreement. To resolve this, we asked outside sources and the team agreed on a consistent style for issues.

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]

Our team's exepectations regarding meeting attendance are flexible, as long as everyone is doing a good share of the work. We ask all team members are on time and do not leave early during meetings, although we are flexible as long as missed meeting time is minimal. We ask all team members attend meetings whenever possible, and if circumstances prevent a member from attending a meeting they make an effort to be informed on what was discussed.

Acceptable Excuse

[What constitutes an acceptable excuse for missing a meeting or a deadline? What types of excuses will not be considered acceptable? —SS]

An acceptable excuse for missing a meeting would be illness, family emergency or other severe crises such as the passing of a loved one. If the excuse doesn't fall within a listed category, the entire team can unanimously accept the excuse. Excuses that will not be considered acceptable are missing a meeting due to an upcoming assignment due date or test, or due to recreational activity.

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]

We ask team members communicate how much work or how many meetings they will be missing, and make a plan to make up work. The team will be understanding but will also expect that once the emergency is resolved, the team member comes back ready and willing to get back on track.

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]

Our expectations are that every member come to meetings ready to show what they've acomplished in the time since last meeting. The deliverables should have no compilation errors and should have no obvious faults. Deliverables should follow the coding standard and should not be difficult to understand.

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]