Development Plan
Sandlot

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]

Team meetings will be held weekly on Tuesdays at 2:00pm at a Hatch conference room. However, team meetings are flexible, if a majority of the team (3/4 members) agrees to either cancel the weekly meeting, schedule an extra meeting, or hold the meeting virtually then the according changes will be made.

Supervisor meetings will be held every 2 weeks on Wednesdays at 3:00pm in JHE 373. Similar to team meetings, if either the supervisor requests or a majority of the team agrees to any of the previously stated changes, they will be made.

The meeting chair will head meetings and will present the weekly agenda at the beginning of each meeting which will cover the topics to be discussed. Each team member will provide a weekly summary covering the work they've done and outstanding issues they've been tackling. Discussion of these weekly agenda items and outstanding issues will be lead by the meeting chair. Every team member will leave the meeting with a list of TODO items that they plan to tackle next. The agenda, weekly summaries, outstanding issues, decisions made and TODOs will be recorded by the scribe in the meeting's assigned Github issue. Finally, meetings have a maximum duration of 1 hour.

5 Team Communication Plan

[Issues on GitHub should be part of your communication plan. —SS]

GitHub issues will be used to communicate technical information between group members.

Discord will be used as the main group communication channel for the team. A majority of group communication will be held through here including online meetings, admin details, and general unplanned discussions.

Call and Text will be used for direct communication between 2 team members and will generally be reserved for high priority and urgent communication or campus location coordination.

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

GitHub Projects will be used to keep track of the team's tasks. The 'board' feature will be primarily used for tracking issues and stories and during scrum meetings. This will ensure the team stays organized and understands what tasks to complete next. The board will contain the sections 'Todo', 'Backlog', 'In Progress', 'Review', and 'Done', with subsections open to be added. 'Todo' will contain planned tasks that have not been started. 'Backlog' will contain the previous sprint's items that carry over as well as any items that are currently on hold. 'In Progress' will hold items actively being worked on by team members. 'Review' will hold items that are awaiting a review from other team

members. 'Done' will contain items that have been completed. Tasks should be small in scope and based around features. For example, "Implement Module X", or "Document Y Section A.B". Tasks should be specific and measurable.

Team Formed, Project Selected	September 16	0%
Problem Statement, POC Plan, Development Plan	September 23	2%
Requirements Document Revision 0	October 9	5%†,‡
Hazard Analysis 0	October 23	3%†
V&V Plan Revision 0	November 1	5%†,‡
Proof of Concept Demonstration	November 11–22	5%*
Design Document Revision 0	January 15	5%†,‡
Revision 0 Demonstration	February 3–February 14	10%*
V&V Report Revision 0	March 7	5%†,‡
Final Demonstration (Revision 1)	March 24–March 30	20%*
EXPO Demonstration	April TBD	10%*
Final Documentation (Revision 1)	April 2	30%*,‡
- 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		

- 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

  The project can roughly be divided into 3 main components:

  1. A database where the bulk of the site's data will be stored.
  2. A webserver which will host the website's visual data and code.
  3. Middleware which will allow for communication between the webserver and database.

  The middleware will be written in Python due to its ease of use and familiarity among team members.

  We will either use PostgreSQL scripts or an ORM (like SQLalchemy) to implement the database, the language used will depend on the implementation chosen. PostgreSQL scripts would require using only SQL, while an ORM would likely live with the middleware and as such would be written in Python.

  The webserver will be written in Javascript using the React framework. This is because of its ease of use, team member experience, and large set of available libraries which will be useful for implementing the visual elements of the website. Additionally, it was expressed to us by stakeholders that the website should be easily maintainable. Implementing the website using an extremely popular and widespread framework such as React means that there are countless resources online to help future maintainers keep the project alive.

- Specific libraries

  Depending on the implementation, the middleware will use some combination of the FastAPI and SQLalchemy libraries. FastAPI will be used to implement HTTPS communication routers between the database and webserver. SQLalchemy will be used to implement the database if it is decided that it will be implemented using an ORM.

  The webserver will use a large set of both functional and visual React libraries. For example, react-navigation will be used for its page traversal functionality while libraries such as react-datepicker and react-calendar

will be very helpful when implementing the visual elements of a scheduling system. Additionally, the axios library will be used to form and send the HTTPS requests to the middleware from the webserver.

- Pre-trained models

  This project does not include an AI component and will not use a machine learning model.

- Specific linter tool (if appropriate)

  ESLint will be used for linting Javascript code while flake8 will be used to lint Python code. We chose these linters due to our previous experience using them and because they provide our preferred formatting style.

- Specific unit testing framework

  The pytest framework will be used to create unit tests for the middleware code. We chose pytest over other python testing frameworks due to its simplicity, small amount of boilerplate code, and plugins which can add useful functionalities like coverage reporting. We plan to incorporate these pytest unit tests as a part of our CI plans for the project via Github actions.

  Testing the database will likely be done using a dummy / development PostgreSQL database prior to making any changes to the production database to ensure that minimal migrations are required during development.

  Our plans for testing our webserver's React code are not decided yet, but we're currently investigating potential testing options.

- Investigation of code coverage measuring tools

  The Coverage.py Python library will be used to measure the code coverage of our middleware program. For the webserver's React code, Jest is included by default when using the "create-react-app" command and will be used to measure the code coverage of the webserver.

- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done

- Specific performance measuring tools (like Valgrind), if appropriate

  The webserver will be hosted on azure's web app services, meaning that azure's suite of performance measurement tools and metrics will be used as the webserver's main performance measurement system. Additionally, we plan to do practical performance tests with stakeholders by having them use the website casually to ensure that performance during regular use is up to their standards / expectations.

  Similarly, the database will be hosted on an azure container, meaning that azure's suite of performance measurement tools and metrics will once

again be used as the databases main performance measurement system. Additionally, performance of queries will be timed throughout development to determine what indices should exist on the database for practical performance.

Finally, the middleware will use the profile library included with Python to measure the performance of its HTTPS routes.

- Tools you will likely be using?

  There are a number of tools/programs/services that we plan to use that were either mentioned in passing previously or not mentioned at all.

  Azure containers and/or virtual machines will be used to host both the database and middleware components of this project. Additionally, the webserver will be hosted using azure's web app services. This allows all main components of the project to be hosted in one place.

  Node.js will be the server environment which we run our webserver on and npm will be what we use to manage our Javascript packages.

  Git/GitHub will be used for version control on the project. GitHub projects will be used as a general project management tool to help keep track of issues, work done, and available tasks. Finally, GitHub actions will be used for CI of tests as the repository is modified.

[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.

Reflection – Nicholas Fabugais-Inaba

The importance of the development plan is to help the team understand the standards that we have set ourselves, so that throughout the development process, everyone is aligned with the objectives that need to be accomplished at a given time. Roles, communication methods, workflows, and more, remind team members of the structure that takes place to keep all of us and the project organized.

There are many advantages that come with CI/CD being implemented in the development process. The biggest role is its ability to aid developers in making sure that everything compiles correctly and any errors are highlighted to the user. This allows the developer to fix their mistakes that may be fatal to what the user may interact with when the changes are eventually deployed. One disadvantage may be when the developer relies too much on the usage of CI/CD as although it may catch some errors it might not catch all of them, even the ones that the developer may have missed. As a developer, it is important to not only use automated testing, especially with larger code bases that are constantly experiencing changes, but also conduct their own testing to make sure the features that they are implementing, work as intended.

A disagreement that the group had in this deliverable was about the structure of pull requests and the naming conventions associated with them. After some deliberation, the group collectively came to the conclusion that the best method to organize our pull requests would be to not only state what the commits or changes were about, but also attach a label in Github, indicating the category the pull request focuses on (i.e. documentation, bug).

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]