Development Plan
Sandlot

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

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 24, 2024	NFI, JL, CG, AV	Initial Draft
October 13	JL	Updated GitHub Project Link, fixed typos, updated section 6 based on TA feedback
November 6	CG	Scaled back POC demo plans to remove menstions of system architecture and advanced UI based on TA feedback
November 11	JL	Updated POC demo plan with specific plan

This document contains our development plan for the 'Sandlot' project, a softball league scheduler. It includes team members, team meetings, workflows, standards, expected technologies, reflections and our team charter.

1 Confidential Information?

This project has no confidential information to protect from industry.

2 IP to Protect

This project has no intellectual property to protect.

3 Copyright License

We are adopting the GNU General Public License v3.0. It is located in the root of this repository: Sandlot

4 Team Meeting Plan

The team will have two scrum meetings a week on Tuesdays and Thursdays every week between lectures. We ask every team member to be in attendance unless a valid excuse is given. These meetings will be in person if possible.

During scrum meetings, the scrum master will ask each team member to describe work they have done since the last meeting, what they plan to work on, and any pain points or blockers the have ran into while working.

We will also have flexible meetings if a specific issue arrises. These meetings can be in person or online. We ask all group members to attend these meetings, but flexiblity is allowed if a member cannot attend.

Supervisor meetings will be held biweekly in the supervisor's office. If the team and supervisor agree a meeting is not necessary, it can be cancelled.

The meeting chair will head non-scrum meetings and will present the weekly agenda at the beginning of each meeting. The meeting chair will also create an issue for the meeting with the agenda on GitHub. The assigned team scribe will record the meeting minutes, and add them to the meeting issue.

5 Team Communication Plan

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 Discord including online meetings, admin details, and general unplanned discussions.

Call and text will be used for direct communication and will generally be reserved for high priority and urgent communication or campus location coordination.

6 Team Member Roles

- Jung Woo Lee
  - Scrum Master: Leads scrum meetings
  - Developer
- Alex Verity
  - Scribe: Takes meeting minutes of any team meetings
  - Developer
- Nicholas Fabugais-Inaba
  - Meeting Chair: Provides agenda for any team meetings
  - Developer
- Casra Ghazanfari
  - Domain Expert: Provides expertise on solution architecture
  - Developer

If a team member is unable fulfill their role for a time, the rest of the members will unanimously decide on a person to take their place.

7 Workflow Plan

Git is the most important part in managing issues, editing documentation, and developing features.

We are using two central branches named documentation and development for writing documentation and developing software respectively. The main branch will be merged and updated for milestones, proof of concepts, and final deployments. For new features, bug fixes or documents, a branch can be made from the documenation or development branch. Before any changes are be made, the local repository should be updated to the latest changes on the

remote branch.

Continuous integration will be implemented via GitHub Actions. When a pull request is made, automated tests will run against the code being tested. Each pull request will be assigned a GitHub label (i.e. documentation, bug). At least one approval will be required from a reviewer for each pull request. All pull requests to the main branch should have no obvious issues. Changes should be made to the development or documentation branch, and once quality is confirmed, the branch will be merged with main.

For each issue, a team member should be assigned to the issue on Github. The descriptions of the issues should be explained in detail and labeled appropriately with a Git label. New labels should be created, if none of the existing labels match the description of the issue in question. Issue templates should be used where possible for maintaining the quality and consistency of the issues.

Milestones and checklists will be used to help keep the team on track. The team will utilize these from the already existing templates within the Github 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 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', 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. '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.

GitHub project link

Project Schedule:

Revision 0	
Problem Statement, POC, Development Plan	September 24
Requirements Document	October 9
Hazard Analysis	October 23
V&V Plan	November 1
Proof of Concept Demonstration	November 11–22
Design Document	January 15
Revision 0 Demonstration	February 3–14

V&V Report	March 7
Revision 1	
Final Demonstration	March 24–30
EXPO Demonstration	April TBD
Final Documentation	April 2

9 Proof of Concept Demonstration Plan

To be a successful project, the project should be able to intake a set of availability data and generate an optimized schedule based on that data. Additionally, it should be able to display this schedule through a basic user interface. The main risks for the success of this project are whether we are able to optimize schedule generation to a point which satisfies our stakeholders, and our team's lack of domain knowledge on scheduling problems and web development.

In order to demonstrate that these obstacles can be overcome, the goal of this proof of concept demonstration is to develop an algorithm to generate highly optimized schedules based on team availability data. Furthermore these optimized schedules should be displayed through a basic UI hosted on a webvserver.

Demonstration Plan:

- Quick overview of the POC demonstration

  Scheduling algorithm and UI visualizer

- Explanation of scheduling algorithm

  Problem formulation

  Algorithm to solve

  Algorithm details and other notes

- Demonstration of POC

  Show UI

  Generate schedule

  Show that each constraint is satisfied

10 Expected Technology

- Programming languages

  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 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 user interface 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.

- 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

- Linter tools

  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.

- Unit testing frameworks

  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.

- 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.

- Performance measuring tools

  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.

- Other tools

  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.

11 Coding Standard

Borrows from:
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 variable or function.
  - Limit the use of global variables wherever possible.

Appendix — Reflection

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).

Reflection – Jung Woo Lee

A development plan is crucial prior to beginning the project to lay out necessary information for the team to follow. The information it provides guides each team member's actions regarding the project, ensuring the quality of their work, work ethic, and the project as a whole. The development plan being created beforehand further reduces risks of disagreements on fundamental details of project management and operations occurring in the middle of the project, where it would be more cumbersome to adapt. It can also simplify the team's work, as ambiguities regarding certain aspects of the project should be clarified in this document.

An advantage of CI/CD is that it can quickly and automatically perform tests on one's code in a standardized method, such that a certain defined quality is met. This can catch common problems and also might reduce the need of each developer testing every aspect of their code. It can also integrate with GitHub to automatically create issues, saving on a lot of boiler-plate actions; tests can be imported easily as well. A disadvantage is that setup may be complicated, and for an efficacious CI/CD system, a lot of thought and time needs to be put into it. There may also be a psychological aspect where developers may be led into a false sense of security when passing the automated tests. Lastly, automated tests do not pertain to usability aspects of the project, which requires manual testing and stakeholder interaction.

Slight disagreement regarding use of labelling and tagging occurred but was resolved by consulting with a TA. Slight disagreement regarding meeting schedule and times was resolved through discussion and consensus. There was significant disagreement about the design of the scrum board about the sections. This was resolved through discussion and agreeing to changes later if needed.

Reflection – Casra Ghazanfari

It is important to create a development plan before starting a project because of the number of benefits it provide to a project's development cycle. One of its main benefits is increased clarity on the objectives of the project for those involved. Development plans achieve this by identifying and defining the project's goals/objectives early in the development cycle, meaning that those involved in the project will more easily and quickly understand the project's goals greatly

reducing potential misunderstandings. Additionally, it acts as a fantastic tool for risk management as it allows you to identify any risks or challenges associated with the project at an early stage. Furthermore, they allow you to develop strategies to mitigate these risks/challenges from causing any failure in the project in the future.

Two of the biggest advantages of CI/CD are early detection of bugs and decreased manual errors. Through the automation of testing every code change using CI/CD, bugs can be caught much earlier than if tests were manually done by humans. Additionally, when automating pipelines using CI/CD a large amount of manual processes are eliminated, greatly reducing the number of points of potential manual error and greatly increasing reliability of the system. However, CI/CD comes with its share of downsides, the two biggest being the cost of setup and maintenance overhead. Setting up CI/CD is not cheap, it takes a large amount of time when unfamiliar with the process or requires experience to implement it quickly. Additionally, not only is CI/CD costly to set up but also costly to maintain. CI/CD systems require continuous maintenance through updating tools, dependencies and other means. Overall CI/CD comes with many disadvantages and advantages that can weigh differently depending on the project and team.

The group disagreed on whether our proof of concept demonstration plan should include a UI element. Some members argued that UI elements would not constitute a "main risk for the success of the project" and that they wouldn't qualify for "something that would keep them up at night" while others thought that the UI element of this project was such a key part that it should be included. We resolved this by having a group discussion to explain our stance on the matter and compared reasonings on each side of the argument. In the end, it was decided that because a well designed UI was such an important factor to the stakeholders of the project, if the UI were not extremely user friendly it could cause problems for the project down the line. The proof of concept demonstration plan now includes a UI element to reflect this.

Appendix — Team Charter

Borrows from: University of Portland Team Charter

External Goals

Our team's external goals are to get a good mark in the course. Our team considers a 85% or above a good mark. We also want to have a well made and well documented project to show on resumes and discuss in interviews.

Attendance

Expectations

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

An acceptable excuse for missing a meeting would be illness, family emergency. If the excuse does not 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

In case of emergency, we ask team members communicate how much work or how many meetings they will be missing, and make a plan to make up missed 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

Our expectations are that every member come to meetings ready to show what they have 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

Team members should strive to be respectful of each other and their ideas, though critical review should not be discouraged due to this. Members should strive to appreciate each other's contributions as well as being patient when mistakes are made. Each team member will be respectful to the supervisor and any outside party directly or indirectly involved with the project. This means in demeanor as well as respecting their time.

Stay on Track

The primary method our team will use to keep on track will be scrum meetings. Our scrum master will ask each member what they have worked on since the last meeting, which will inform the group of each other's progress. This will let the group know if we are falling behind on the project, and will let any team member know if they are less productive than the others.

We will use attendance metrics to make sure team members are attending enough meetings. If a member misses three meetings in the past two weeks, the team will do a check in and make sure the member is on track. We will also loosely monitor commit metrics on GitHub to make sure members are on track.

Team Building

The team will participate in a group crosswords and in sports days.

Decision Making

All disagreements should be discussed with the whole team, or at the very least, relevant team members. Disagreements regarding a major aspect of the project should require a unanimous agreement. More minor disagreements may be agreed upon by majority. Upon ties, a relevant third-party should assist in the decision making.