Software Development Schedule

CS 3300-002 Spring 2025

Introduction

This document outlines the Software Development Schedule for the John Doe Portfolio project. It includes our initial planning, how the schedule was maintained throughout the document, and a reflection on how our Software Development process evolved over time. Snapshots of our tracking document are provided to better show our progress and approach.

Initial Plan

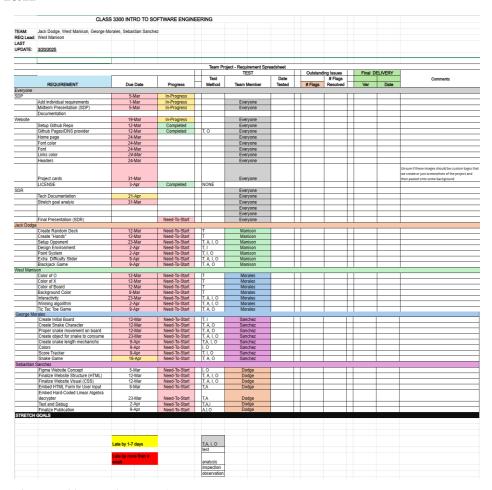


Figure 1 Initial Requirements Spreadsheet

Above is a screen capture of our initial requirements spreadsheet. Initially, we created an extensive list of possible tests and requirements need to complete the project. To help manage the workload and ensure clear expectations, we split up our requirements by team members. While

individual responsibilities were separated by team member, we collaborated to create a schedule that worked for the entire team, aiming to sync our submissions whenever possible.

Since we are developing with Test-Driven Development (TDD) as our development model, it made sense to combine our software **and** testing requirements into a single document. This allowed easy visibility of what tests needed to be done for each software component and allowed us to quickly observe what each team member needed to work on (whether that was developing code or writing/refining tests).

Continuously Maintained

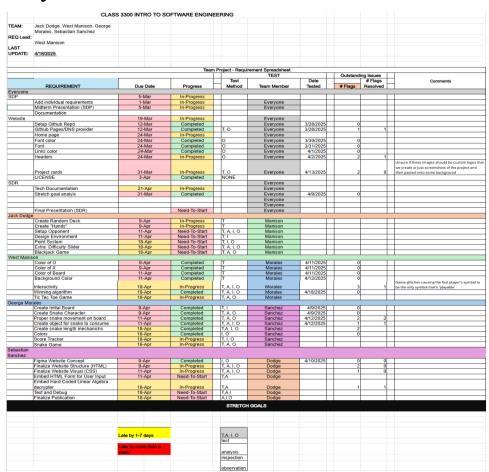


Figure 2 Maintained Requirements Spreadsheet (halfway through the project development)

Above is a screen capture of our requirements spreadsheet two weeks after spring break.

Maintaining this document proved to be one of the harder aspects of the project. It was challenging to ensure everyone consistently updated the spreadsheet as they developed and tested their code. However, our regular team check-ins and open communication (on Discord) helped keep the team aligned and moving forward a completed project.

As It Actually Happened

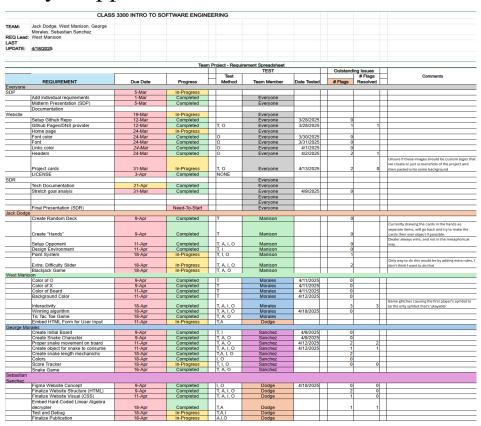


Figure 3 Finalized Software Requirements Spreadsheet

Above is a screen capture of the final requirements spreadsheet. Although not every item was updated consistently, the document overall served as a helpful tool for tracking team progress.

Over time, we shifted slightly away from a strict TDD model and moved toward a more agile development approach. We began working in small 'mini sprints', often writing code first and requesting testing after. This unspoken change made our combined requirements/testing spreadsheet slightly less effective, highlighting a potential area for process improvement.

Conclusion

In the end, our initial planning, schedule creation, and effort to maintain a shared tracking document were critical to the project's success. Although we faced challenges with consistent updates and experienced a natural evolution in our development model, we were able to work through all of it as a team due to our open and honest communication. Allowing team members to discuss when things weren't working and where they needed support, created a deeply collaborative environment that persisted through the schedule changes.