9/22/2024

WeDeliverTECH

Reception Management Web Application

SP1

Marcelo de Oliveira Barcelos Jr
NOROFF

Reflection Report: WeDeliverTECH

Introduction

The objective of the WeDeliverTECH™ Reception Management System was to develop a web application that efficiently manages employee attendance and delivery schedules. The goal was to create a system that monitors work hours, schedules, and sends out notifications when there are delays. The application was developed using HTML, Bootstrap, CSS, and JavaScript using an object-oriented approach.

The project lasted for four weeks and was organized into three sprints using Jira for task management:

1. **SP1 - Front End/User Interface (2 weeks)**
2. **SP1 - Back-End/Integration (2 weeks)**
3. **SP1 – Review (Continuously for 4 weeks)**

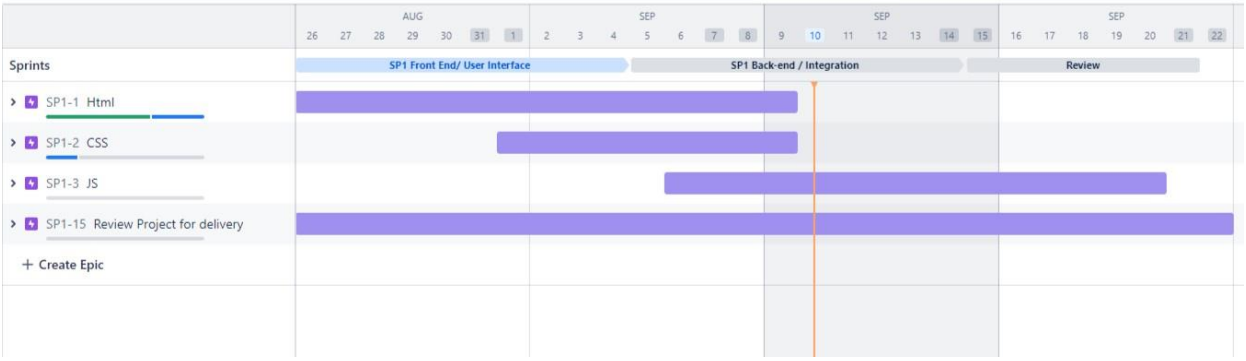

Figure-1: WedeliveryTECH's Jira Timeline.

Sprints:

1. SP1: Front End/User Interface (Figure 2 and 3)

In this sprint, the application's front-end was built using HTML and CSS. The Epics and constituents' tasks were created in Jira as follow:

1.1. Epic: HTML.

Tasks:

- **Analyse Customer Requirements:** Understanding customer needs early was key to building an efficient application.
- **Build a Strategy:** Focused on creating a clear website structure. This helped prevent any rework down the line by ensuring the design was well-planned.
- **Start Coding:** developing a clean, semantically correct HTML framework, which served as the backbone for further styling and functionality.

1.2. Epic: CSS.

Tasks:

- **Analyse HTML/Mock-up:** Firstly, the HTML structure was checked against design mock-up to make sure everything aligned.
- **Start Stylesheet.css:** Following **WeDeliverTECH™** guidelines for colours, fonts, and layout, the CSS stylesheet was developed ensuring site responsiveness across all devices/browsers.
- **Review and Ensure Requirements Are met:** After styling, the design was reviewed to make sure it not only looked good but also met all the user needs.

Bugs

- **Issue:** The delivery button was misaligned initially.
- **Solution:** the buttons were placed inside a to align them properly across various screen sizes.

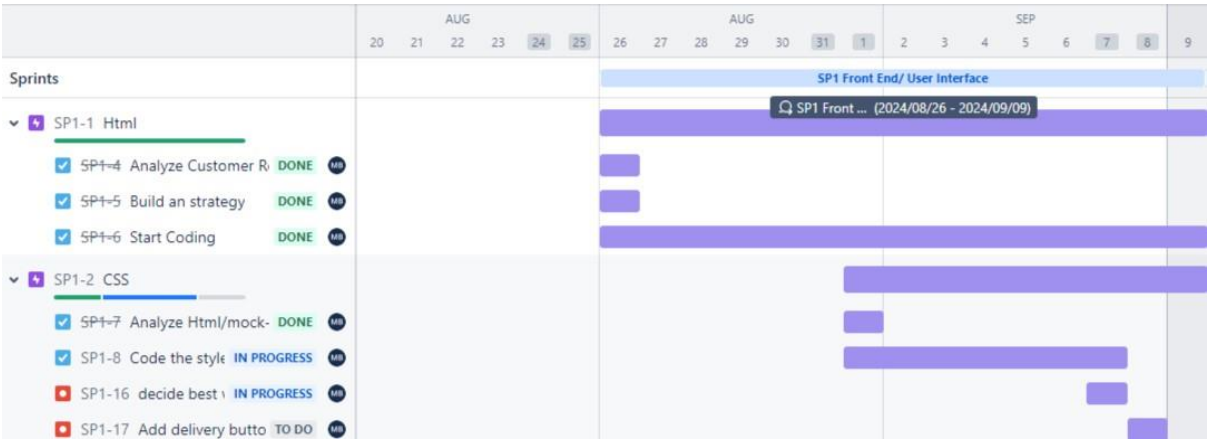

*Figure 2. Timeline of **SP1: Front End/User Interface.***

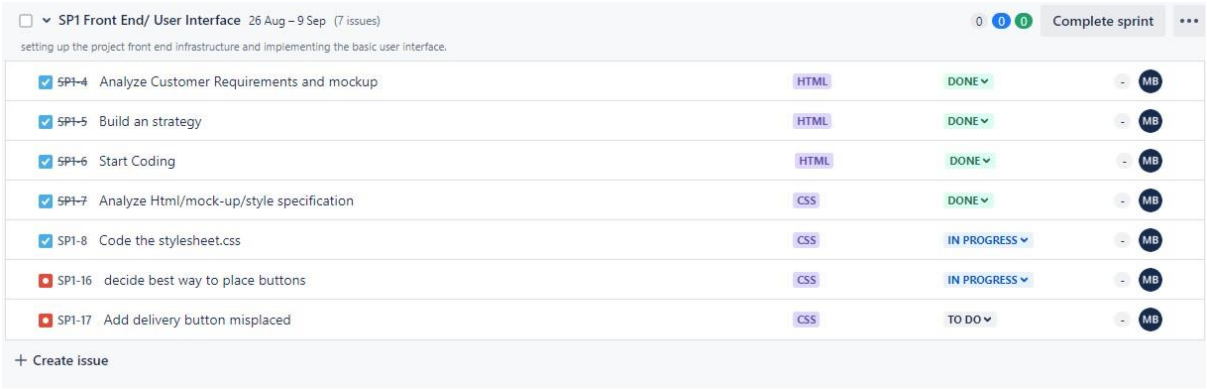

*Figure 3. Backlog of **SP1: Front End/User Interface.***

---

2. SP1: Back-End/Integration

This sprint focused on adding functionality to the application using the script from the file **Wdt_app.js**.

2.1. Epic: JS

Tasks:

- **Analyse Customer Requirements:** Ensure how the backend would match client expectations.
- **API Call (staffUserGet Function):** This task involved pulling and managing staff data from an external API.
- **StaffOut Function:** This feature allowed recording when staff members were 'Out', logging the time and their expected return.
- **StaffIn Function:** This updated staff status to ‘In’ and cleared any absence records.
- **StaffMemberIsLate Function:** Check if staff were running late and send notifications accordingly.
- **Toast Notifications:** Created a visual alert for staff or delivery delays to keep operations running smoothly.

This 2-week sprint made sure the application's features were well integrated and met the company's needs. (Figure 4 and 5).

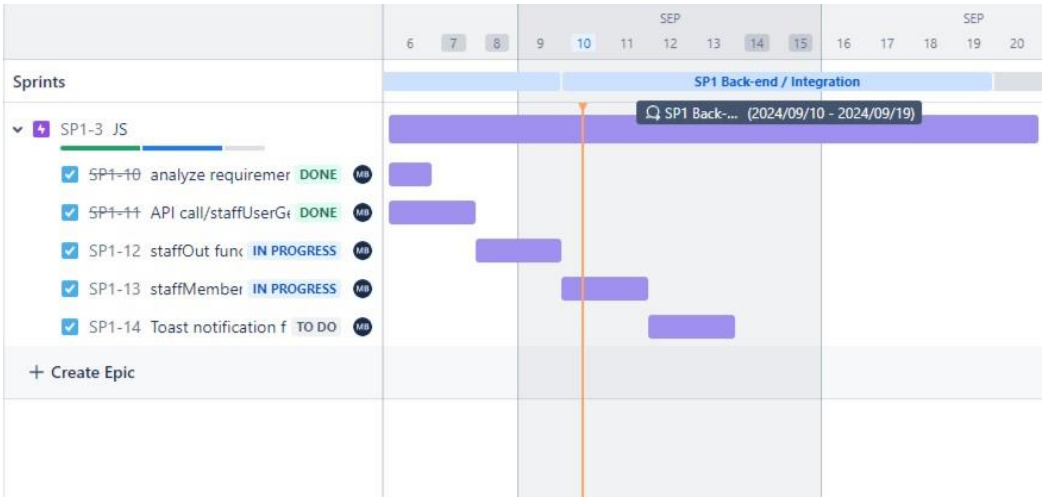

*Figure 4. Timeline of **SP1: Back-End/Integration.***

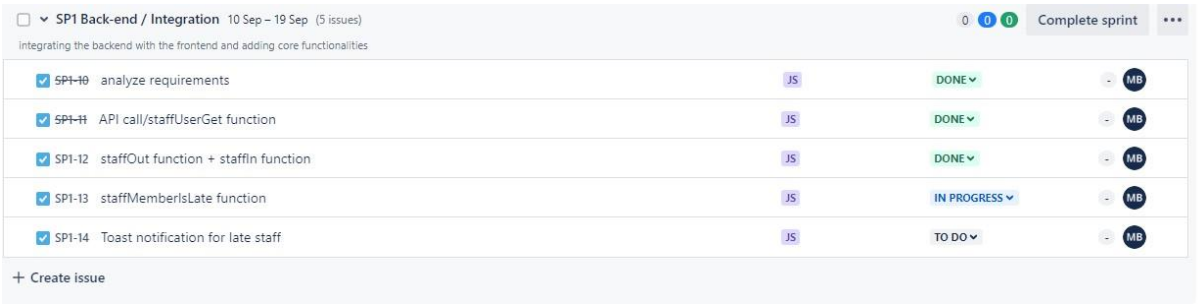

*Figure 5. Backlog of **SP1: Back-End/Integration.***

3.SP1: Review

Why Continuous Review? (Figure 6)

- Feedback Loop: Ongoing testing and evaluations give quick feedback, which enables one to fix problems early and make sure the final product meets expectations.
- Incremental Improvements: Following an Agile approach, testing the Application in stages ensured that each requirement was met.

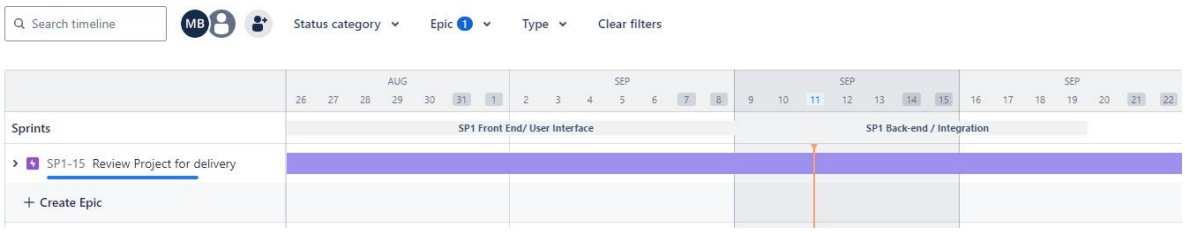

Figure 6. Reviewing the project throughout the whole development cycle.

Project Management and Timeline

Jira's Timeline and board were particularly useful for the project as it offered the means of structuring tasks and tracking their progress with the help of phases including To Do, In Progress, and Done. The first one was front-end development; the second one was back-end deployment; moreover, during all these processes, I conducted assessments constantly (Reviewing).

Figures 7 and 8 below demonstrate how Jira features were used throughout the development of the project:

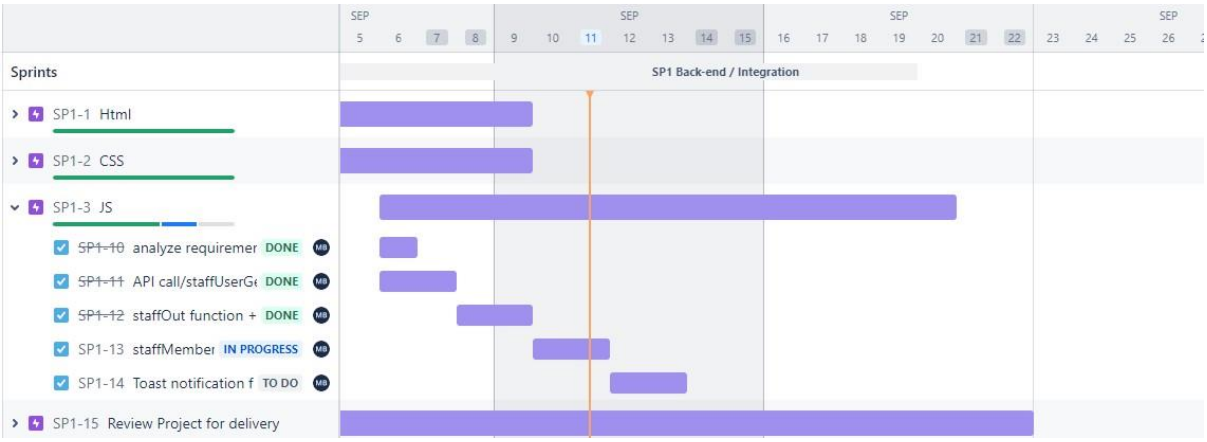

Figure 7. Jira's project timeline .

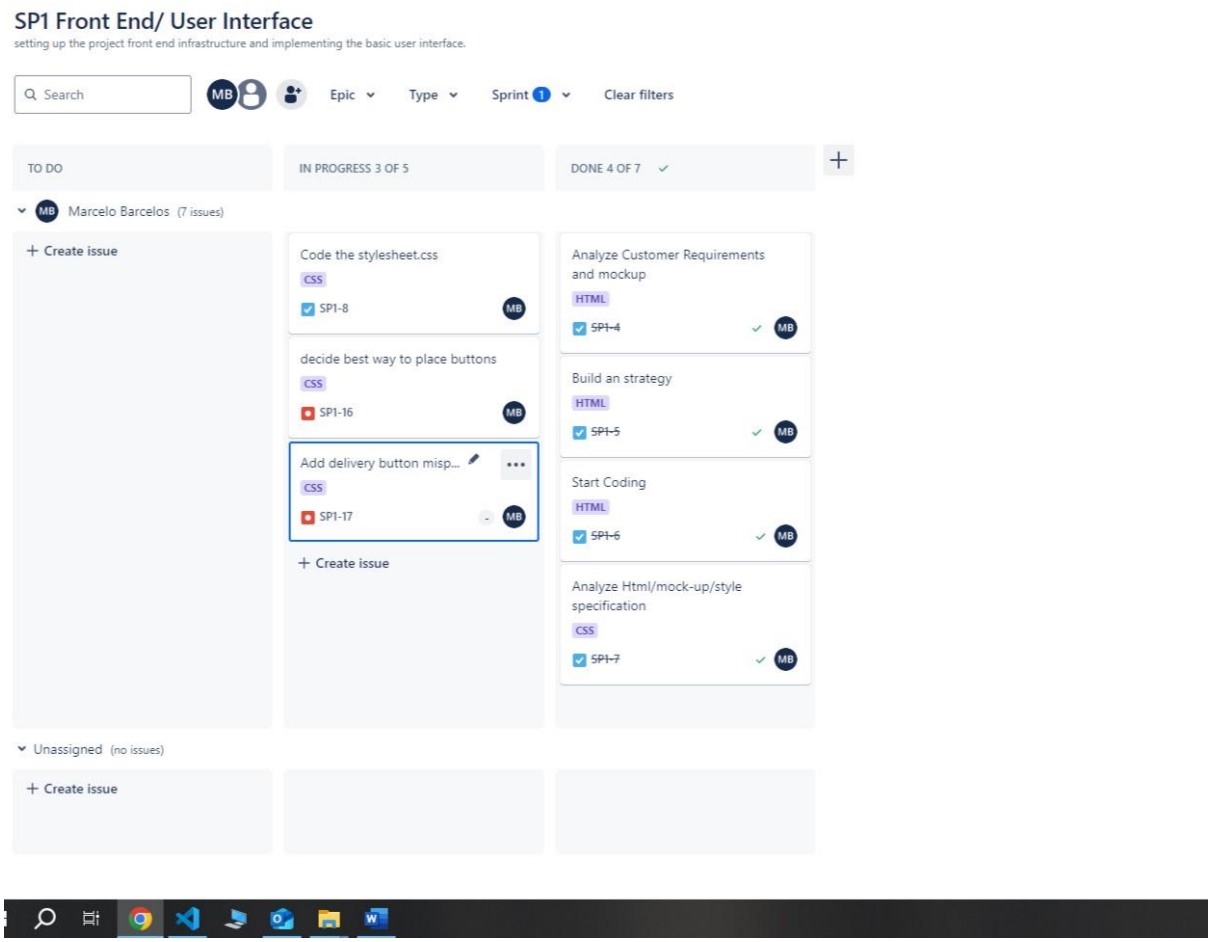

Figure 8. Demonstration of Jira's project board.

Main Challenges

1. Dynamic API Integration:

Converting randomuser.me API data into Staff objects and filling the table was tricky. This was achieved by using Object -Oriented Programming (OOP) principles with the creation of two classes.

Person Class, a base class with common properties and Staff class which by inheritance was derived from the Person class. Each Staff object included details like user's picture, first name, last name, email, and default values for status, out time, duration, and expected return time.

This was followed by the creation of the staffUserGet function to fetch data from the API, through a JSON response the data was transformed into staff objects. This was accomplished by using the map function.

The populateStaffTable was further created to iterate over the “staffMembers” array and create the rows for each “Staff “object.

2. Unified Toast Notification System:

The creation of a function that worked for both delayed cases (Staff and drivers) was achieved by implementing the ShowToast Function that accesses common properties of both classes (Staff and DeliveryDriver) which inherit from the Person class.

By leveraging Inheritance, the function can display the necessary information on the notification for both classes.

3. Standardization of date and time :

The formatDateTime(date) function was developed to standardize the formatting of date and time values across the web application. By converting JavaScript Date objects into a consistent, human-readable string format (DD/MM/YYYY HH:MM: SS).

This uniformity really improves the user experience, making it simple for users to understand the date and time details. For example, the function formats the "Out Time" and "Expected Return Time" in the Staff Table, shows expected return times on the Delivery Board, updates the Digital Clock every second, and accurately displays delay times in Notifications.

Conclusion

It was really fulfilling to work the WeDeliverTECH™ Reception Management System. This project allowed me to gain good experience in all of the aspects of Object-Oriented Programming as well as JavaScript, HTML and CSS. There were several challenges throughout the development of the application, from integrating dynamic data to efficiently managing numerous tasks and resolving complex difficulties, such as creating a single notification system. It also allowed me to efficiently organize and prioritize my work while utilizing Jira as a project management tool. Overall, this project helped me improve as a developer and prepared me for more complex future projects.