Reflection Report for OCT SP-1

Introduction

The goal of this project was to develop a staff management dashboard that would allow a company to efficiently track their employees' out-of-office time and manage their delivery drivers. To meet the client's needs in terms of user experience and content, JavaScript, jQuery, Bootstrap, and other relevant libraries, along with a Random User API were used to protect sensitive information.

To effectively manage this project, we were to employ Jira as our project management tool and I chose to use a Scrum Board to plan out a 4-week sprint. Tasks were divided into three epics:

- Reception Management Dashboard
- User Experience and Branding
- Project Planning

Throughout this project, I was able to create my own sprints, locate issues and tasks effectively, and monitor my progress using Jira. In this reflection report, I will discuss some of the challenges I encountered, the solutions I implemented, and my overall experience working on this project.

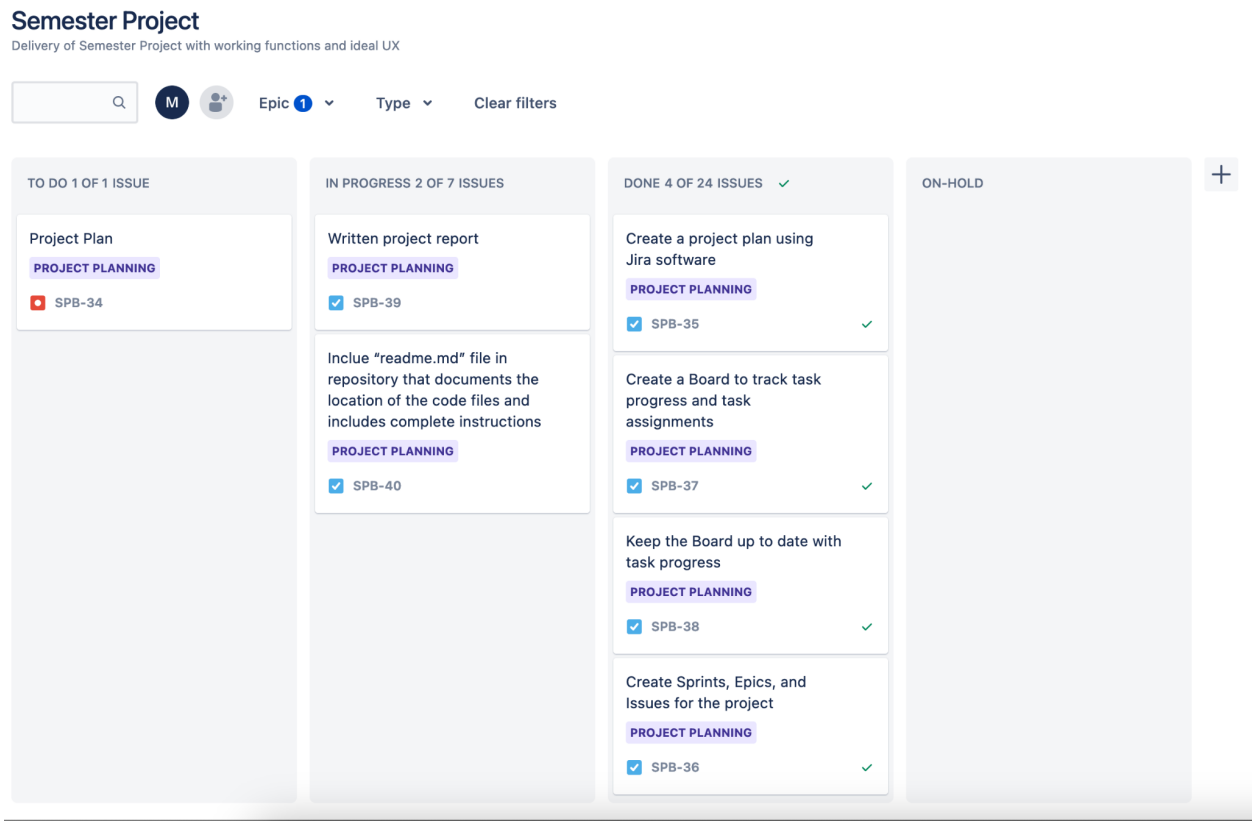
Project Management

To effectively manage my project, I broke down the project assignment into smaller tasks and created my own board to keep track of my progress. The Roadmap shows that I divided the tasks into three epics - Reception Management Dashboard, User Experience and Branding, and Project Planning - which were all part of the 4-week sprint.

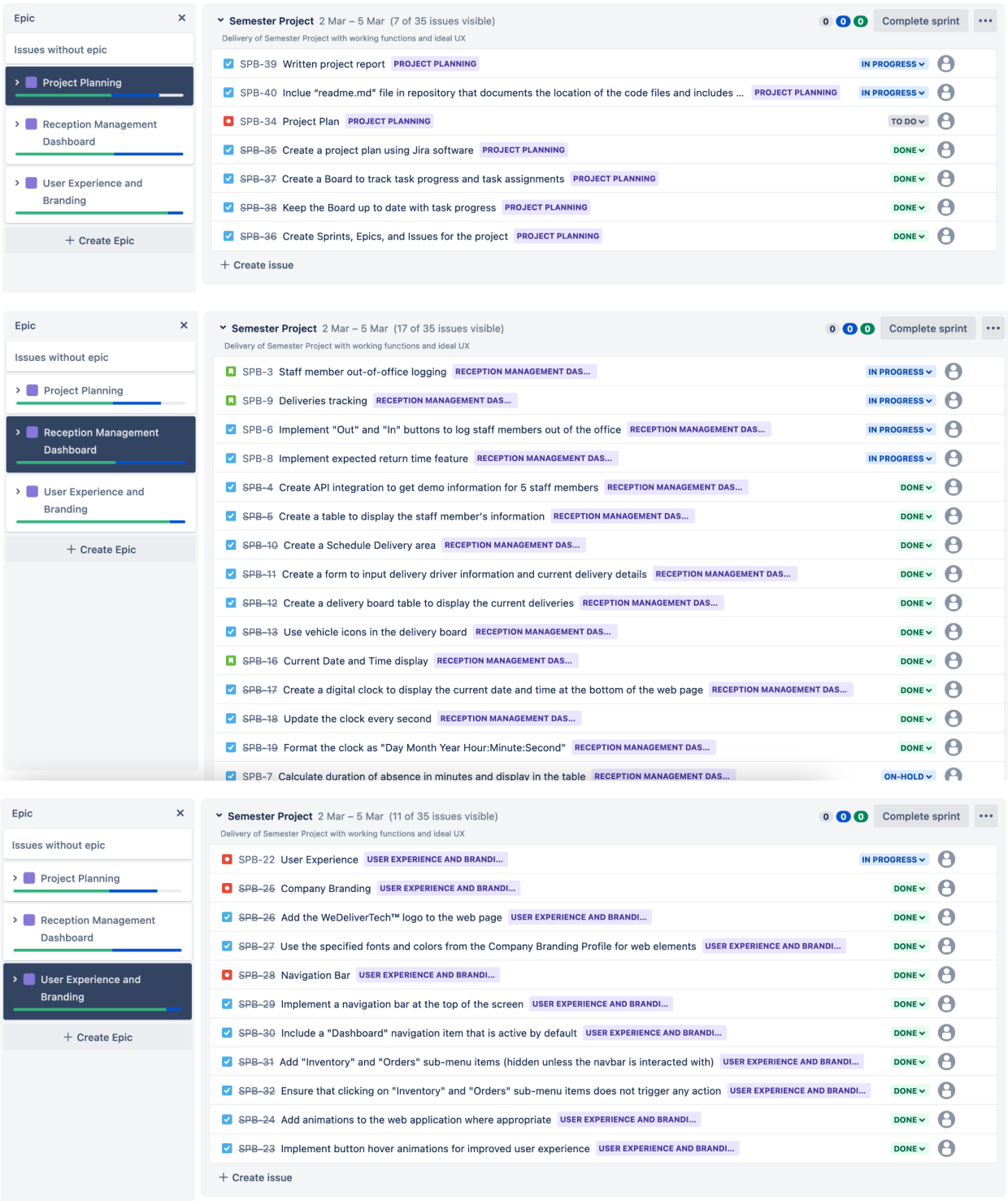
My board had four columns: "To do", "In progress", "Done", and "On Hold". I added the "On Hold" column so I could move tasks that I got stuck on or encountered errors or bugs. This allowed me to easily pick up where I left off and make notes for future reference and link to related topics and websites that might help me solve the issues.

The backlog provided a comprehensive overview of all the tasks assigned to each epic. I found it useful to have a list of tasks and an automatic movement of completed tasks to the bottom.

Although I didn't have any team members for this project, if I did, I would have assigned tasks to each member and allowed time for collaboration and cooperation at the end of the sprint.



Overview of the board for my project, showing my “on-hold” tab as well as my current progress and remaining to-do objects.



An overview of my epics and related issues and tasks.

Challenges faced

During the development of the web application, I encountered several challenges that tested my skills in Bootstrap and CSS styling. To overcome these challenges, I used browser developer tools and selectors to target specific elements inside the Bootstrap-styled tables. The use of Jira as a project management tool was also a challenge as I was accustomed to using Trello for my planning. However, I overcame this challenge by familiarizing myself with the use of Epics and Issues/Tasks in Jira and using a detailed approach to task management to gain a better understanding of the project requirements.

During the development of required functionalities, I encountered an issue when applying a ‘onclick function’ to the Staff table. I had an ongoing challenge with a styling not appearing even though the function was correct. Because of the function being called before the table is filled from the API and thus not registering any table rows when clicked, I had to implement the ‘onclick function’ after I received the data from the API and filled the HTML table. The same solution was done for the delivery table, where I had to use a separate class tag to not mix up the marked staff table and the delivery table.

Another challenge faced was creating a method to check if staffmembers and deliverydrivers were late. Using UNIX time with JavaScript Date() method. I calculated the registered duration of absence in minutes for the staffmembers, and then converted the result to human readable language using toLocalTimeString() and tried to implement this for the delivery drivers as well.

To tackle the more complex aspects of the project, I reserved them for the end and sought help from the module content and online resources. Frequent use of ‘console.log’ helped me debug my code and resolve the issues. By adopting these strategies, I was able to overcome the challenges and deliver a better quality web application for the assignment.

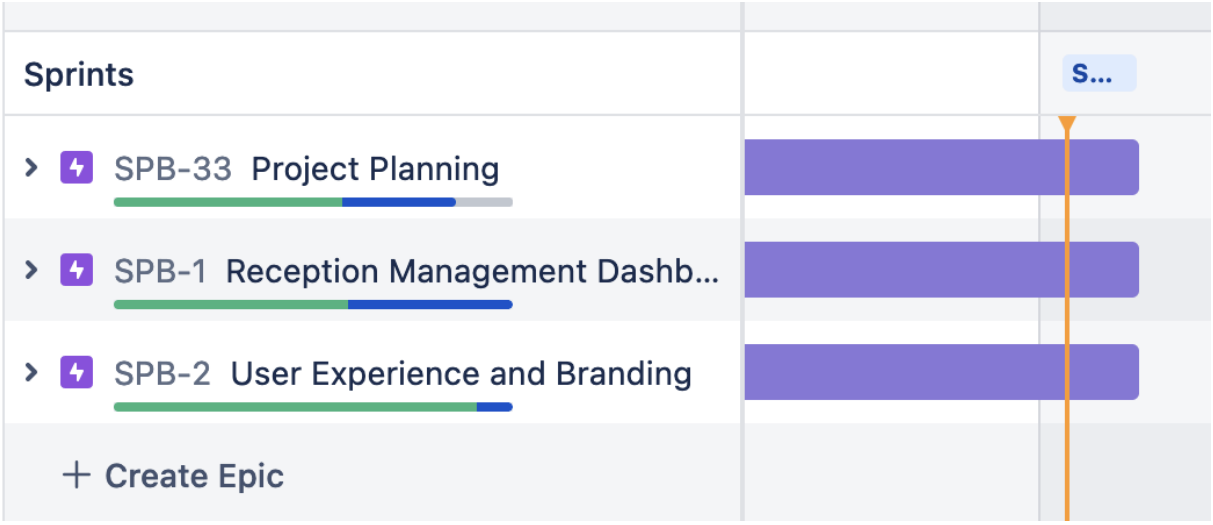
Conclusion

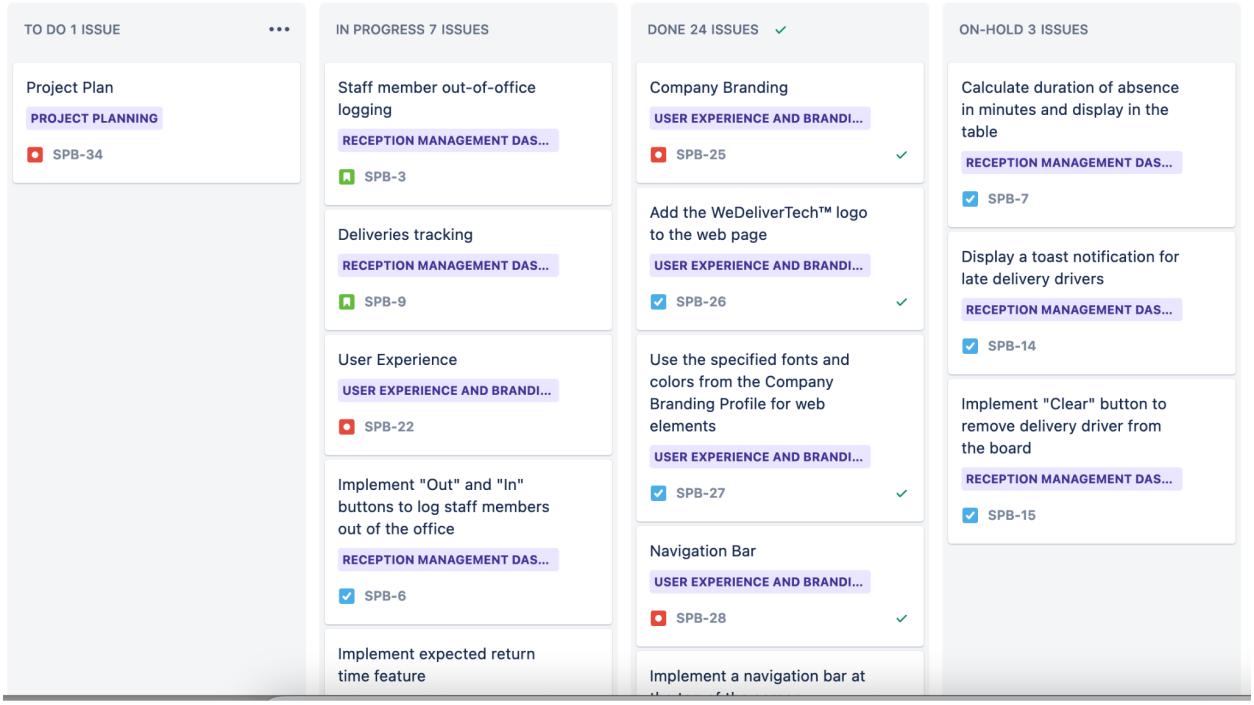
In conclusion, this project has been an insightful experience in terms of project management and web application development. Using Jira for the first time was a challenge, but ultimately allowed for effective organization of tasks and progress tracking. The creation of the Reception Management Dashboard, User Experience and Branding, and Project Planning epics were crucial in dividing and conquering the necessary tasks for the successful completion of the project.

Challenges faced throughout the project, including styling the website using Bootstrap and navigating Jira's interface, were overcome through the use of browser developer tools, online resources, and detailed task planning. By saving the most difficult parts of the project until the end, I was able to apply knowledge gained throughout the project and ultimately deliver a web application that meets the requirements set forth in the project assignment.

Overall, this project has allowed for the development of new skills and knowledge in project management and web application development, which will undoubtedly be useful in future endeavors.

Screenshots





Implementing the ‘onclick function’ for table rows inside the randomUserGenerator and addDelivery:

```
//POPULATE STAFF
let staffList = [];
const randomUserGenerator = () => {
    fetch('https://randomuser.me/api/?results=5')
    .then((response) => {
        return response.json()
    }).then((data) => {
        const tableBody = document.querySelector("#staffTable tbody");
        data.results.forEach((staffData) => {
            const staff = new Staff(
                staffData.picture.thumbnail,
                staffData.name.first,
                staffData.name.last,
                staffData.email,
                "In", //DEFAULT
                "",
                "",
                ""
            );

            //ADD STAFF TO LIST ARRAY
            staffList.push(staff);
            const row = document.createElement("tr");
            row.innerHTML = `
                <td><img src="${staff.picture}" alt="${staff.name} ${staff.surname}" /></td>
                <td>${staff.name}</td>
                <td>${staff.surname}</td>
                <td>${staff.email}</td>
                <td>${staff.status}</td>
                <td>${staff.outTime}</td>
                <td>${staff.duration}</td>
                <td>${staff.expectedReturnTime}</td>
            `;

            $(row).on('click', function() {
            $(this).siblings('tr.selected').removeClass('selected');
            $(this).addClass("selected");
            });

        tableBody.appendChild(row);

        });
    });
};
```

```
//CREATE NEW DRIVER
//VALIDATION OF DATA USING REGEX
if (vehicle == 'car' ||  vehicle == 'motorcycle') {
    if (name.match(validName) && surname.match(validName) && telephone.match(validPhone) && address.match(validAddress) && returnTime.match(validTimeFormat)) {
        const deliveryBody = document.querySelector("#deliveryTable tbody");
        //NEW DRIVER FROM DELIVERYDRIVER CLASS BASED ON INPUT
        const driver = new DeliveryDriver(
            vehicle,
            name,
            surname,
            telephone,
            address,
            returnTime);

        const newRow = document.createElement("tr");
        newRow.innerHTML = `
        <td><i class="fa fa-${vehicle}"></i></td>
        <td>${driver.name}</td>
        <td>${driver.surname}</td>
        <td>${driver.telephone}</td>
        <td>${driver.address}</td>
        <td>${driver.returnTime}</td>
        `
        deliveryBody.appendChild(newRow);

        //HIGHLIGHT TR ON CLICK
        $("#deliveryTable tbody tr").click(function() {
            $("#deliveryTable tbody tr").removeClass("selectedDriver");
            $(this).addClass("selectedDriver");
        });
        // ADD THE DRIVER TO THE LIST OF DRIVERS
        drivers.push(driver);
    } else {
        alert("Please enter valid input for all fields.");
    }
}
```

Prompting for duration in minutes and using this to calculate outtime and expected return. Activating staffMemberIsLate() if current time has exceeded expected return time:

```
    //PROMPT
    const durationInMinutes = parseInt(prompt("Duration (in minutes):"));
    if (isNaN(durationInMinutes) || durationInMinutes <= 0) {
        alert("Please enter a valid duration in minutes.");
        return;
    }

    //CALCULATING TIME FROM MINUTES
    const outTime = new Date();
    const expectedReturnTime = new Date(outTime.getTime() + durationInMinutes * 60000);

    //OUT TIME AND EXP RETURN TIME IN TABLE
    selectedRow.find("td:nth-child(5)").text("Out");
    selectedRow.find("td:nth-child(6)").text(outTime.toLocaleTimeString("en-US", {hour12: false}));

    //MINUTE TO HH:MM
    selectedRow.find("td:nth-child(7)").text(`${Math.floor(durationInMinutes / 60)}:${durationInMinutes % 60}`);
    selectedRow.find("td:nth-child(8)").text(expectedReturnTime.toLocaleTimeString("en-US", {hour12: false}));

    //UPDATES OBJECT TO USE staffMemberIsLate();
    const staffMember = staffList.find((staff) => staff.name === selectedStaffName);
    staffMember.status = "Out";
    staffMember.outTime = outTime;
    staffMember.duration = `${Math.floor(durationInMinutes / 60)}:${durationInMinutes % 60}`;
    staffMember.expectedReturnTime = expectedReturnTime.toLocaleTimeString("en-US", {hour12: false});

    // Check if staff member is late
    const checkLateStatus = setInterval(() => {
        const currentDateTime = new Date();
        if (currentDateTime > expectedReturnTime) {
            clearInterval(checkLateStatus);
            staffMember.status = "Late";
            staffMember.staffMemberIsLate();
            // Update the table with the new status
            selectedRow.find("td:nth-child(5)").text("Late");
        }
    }, 5000);
};
```

Removing delivery driver on “clear”:

```
//REMOVE DRIVER AND ASK FOR CONFIRMATION
function clearDeliveryDriver() {
    const selectedRow = $("#deliveryTable tbody tr.selectedDriver");
    if (selectedRow.length > 0) {
        const index = selectedRow.index();
        const confirmDelete = confirm("Are you sure you want to clear this delivery driver?");
        if (confirmDelete) {
            selectedRow.remove();
            drivers.splice(index, 1);
        }
    } else {
        alert("Please select a Delivery Driver.")
    }
}
```