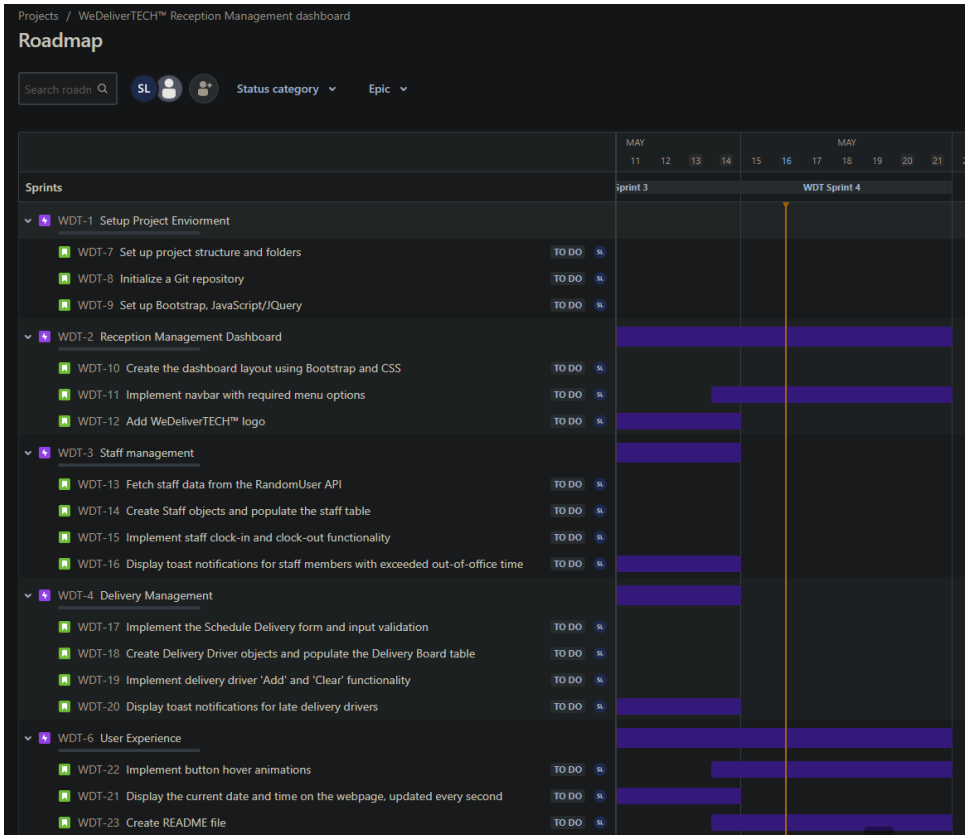
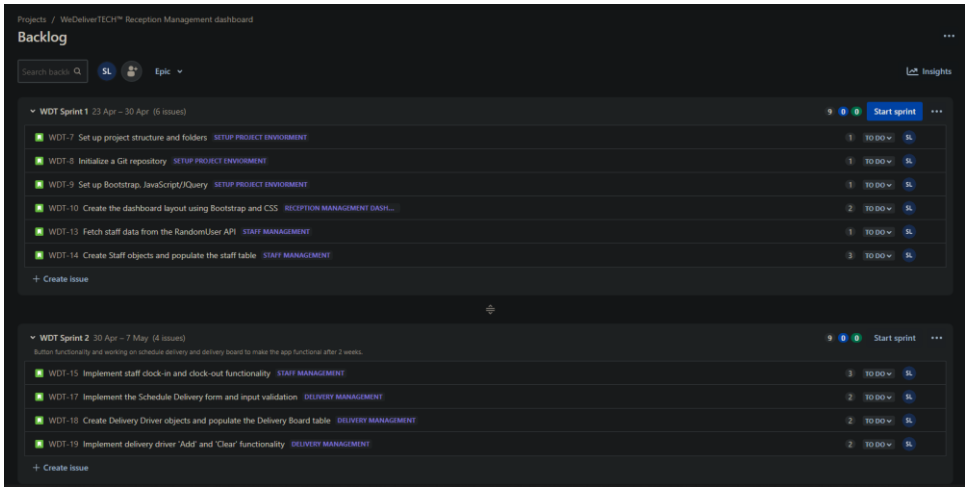
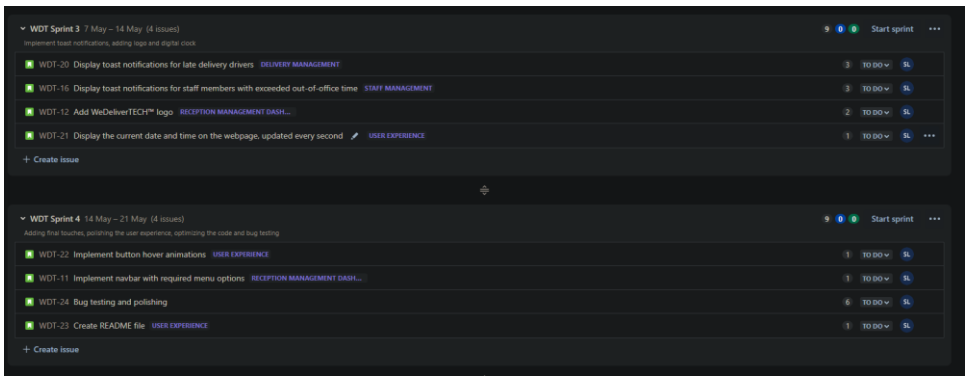
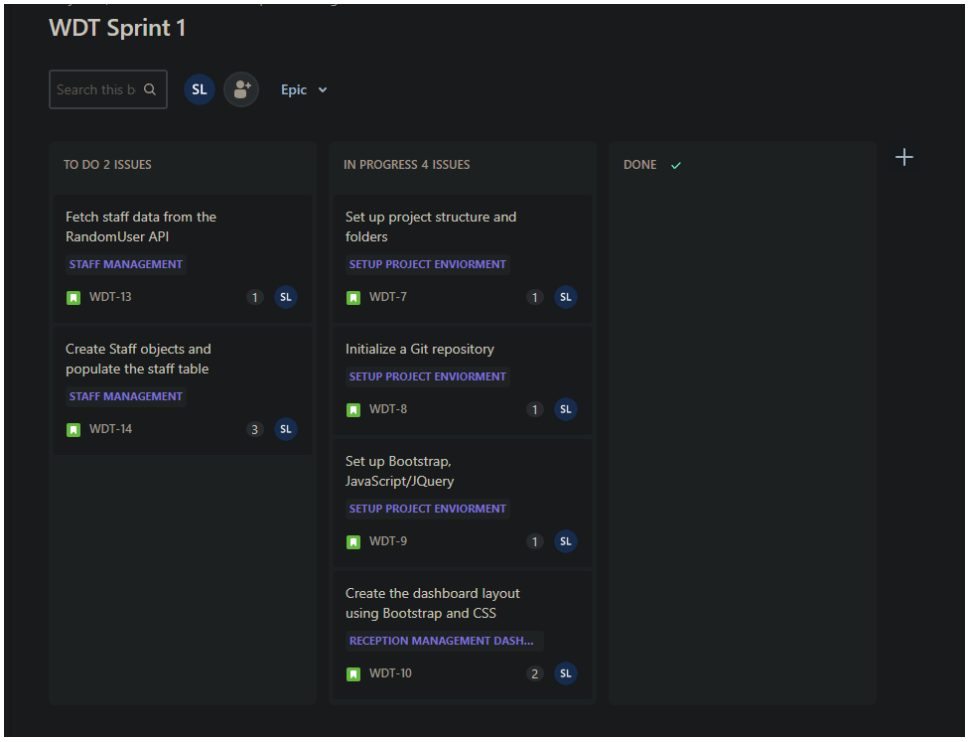
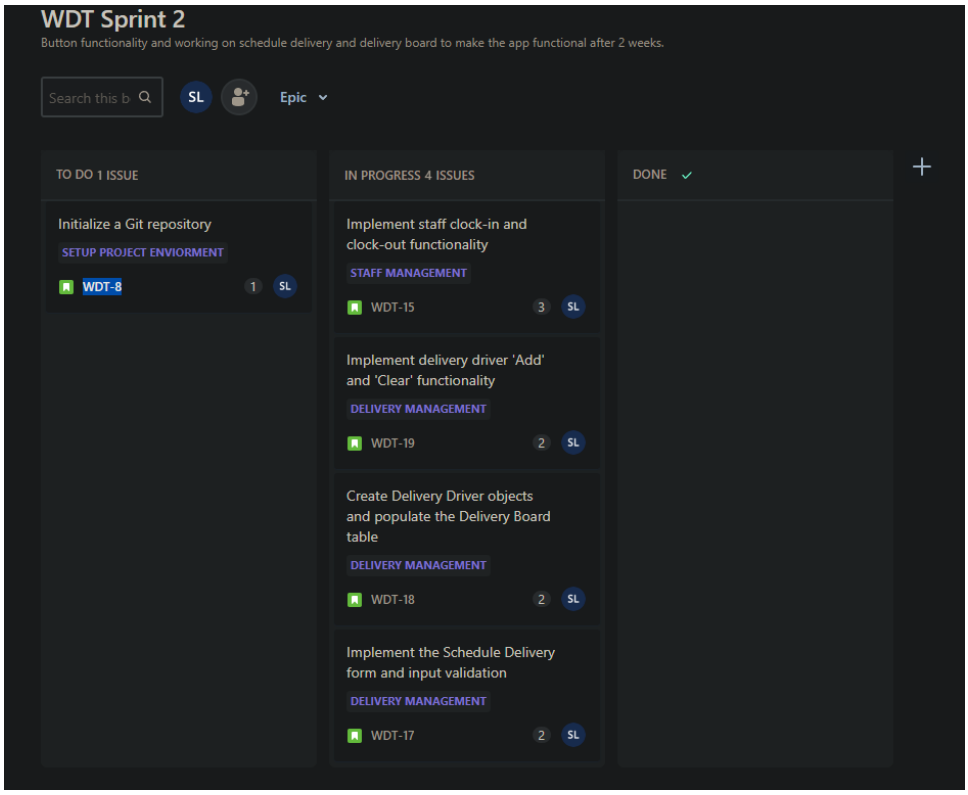

Projects / WeDeliverTECH™ Reception Management dashboard
Roadmap
Search roadn
SL
Status category
Epic
MAY
11 12 13 14 15 16 17 18 19 20 21
Sprints
Sprint 3
WDT Sprint 4
WDT-1 Setup Project Enviorment
WDT-7 Set up project structure and folders TO DO SL
WDT-8 Initialize a Git repository TO DO SL
WDT-9 Set up Bootstrap, JavaScript/JQuery TO DO SL
WDT-2 Reception Management Dashboard
WDT-10 Create the dashboard layout using Bootstrap and CSS TO DO SL
WDT-11 Implement navbar with required menu options TO DO SL
WDT-12 Add WeDeliverTECH™ logo TO DO SL
WDT-3 Staff management
WDT-13 Fetch staff data from the RandomUser API TO DO SL
WDT-14 Create Staff objects and populate the staff table TO DO SL
WDT-15 Implement staff clock-in and clock-out functionality TO DO SL
WDT-16 Display toast notifications for staff members with exceeded out-of-office time TO DO SL
WDT-4 Delivery Management
WDT-17 Implement the Schedule Delivery form and input validation TO DO SL
WDT-18 Create Delivery Driver objects and populate the Delivery Board table TO DO SL
WDT-19 Implement delivery driver 'Add' and 'Clear' functionality TO DO SL
WDT-20 Display toast notifications for late delivery drivers TO DO SL
WDT-6 User Experience
WDT-22 Implement button hover animations TO DO SL
WDT-21 Display the current date and time on the webpage, updated every second TO DO SL
WDT-23 Create README file TO DO SL


Projects / WeDeliverTECH™ Reception Management dashboard
Backlog
Search backlo
SL
Epic
Insights
WDT Sprint 1 23 Apr – 30 Apr (6 issues)
Start sprint
WDT-7 Set up project structure and folders SETUP PROJECT ENVIORMENT
WDT-8 Initialize a Git repository SETUP PROJECT ENVIORMENT
WDT-9 Set up Bootstrap, JavaScript/JQuery SETUP PROJECT ENVIORMENT
WDT-10 Create the dashboard layout using Bootstrap and CSS RECEPTION MANAGEMENT DASH...
WDT-13 Fetch staff data from the RandomUser API STAFF MANAGEMENT
WDT-14 Create Staff objects and populate the staff table STAFF MANAGEMENT
Create issue
WDT Sprint 2 30 Apr – 7 May (4 issues)
Button functionality and working on schedule delivery and delivery board to make the app functional after 2 weeks.
Start sprint
WDT-15 Implement staff clock-in and clock-out functionality STAFF MANAGEMENT
WDT-17 Implement the Schedule Delivery form and input validation DELIVERY MANAGEMENT
WDT-18 Create Delivery Driver objects and populate the Delivery Board table DELIVERY MANAGEMENT
WDT-19 Implement delivery driver 'Add' and 'Clear' functionality DELIVERY MANAGEMENT
Create issue

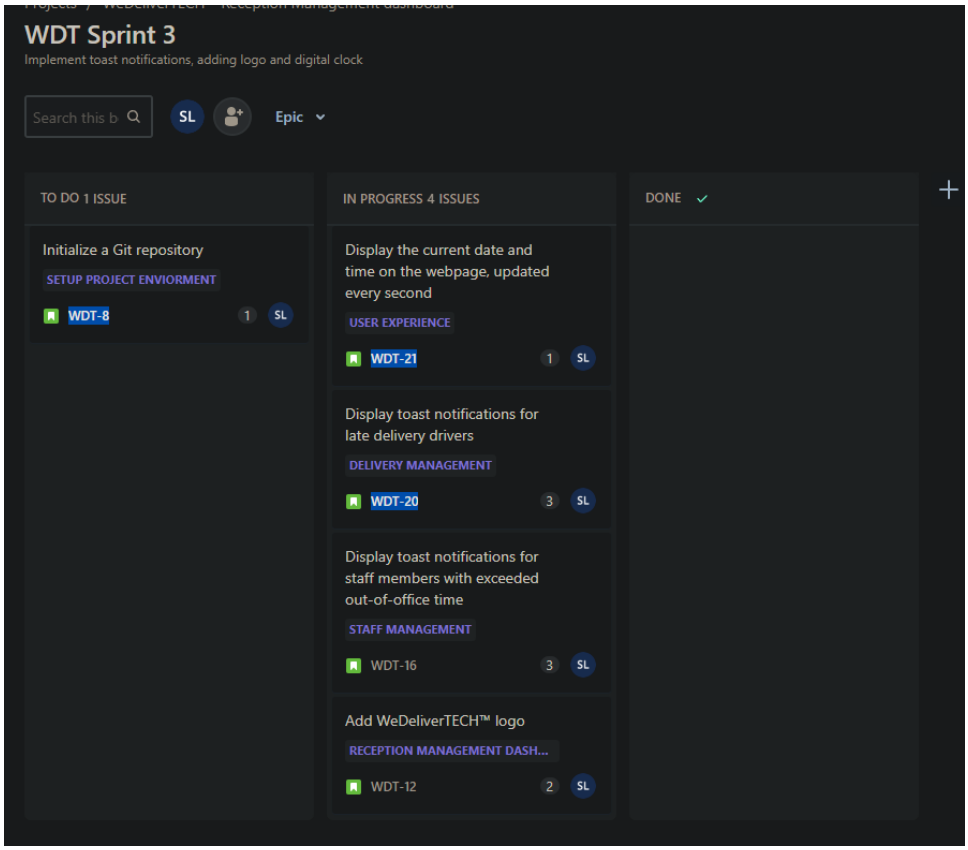

WDT Sprint 3 7 May – 14 May (4 issues)
Implement toast notifications, adding logo and digital clock
Start sprint
WDT-20 Display toast notifications for late delivery drivers DELIVERY MANAGEMENT
WDT-16 Display toast notifications for staff members with exceeded out-of-office time STAFF MANAGEMENT
WDT-12 Add WeDeliverTECH™ logo RECEPTION MANAGEMENT DASH...
WDT-21 Display the current date and time on the webpage, updated every second USER EXPERIENCE
Create issue
WDT Sprint 4 14 May – 21 May (4 issues)
Adding final touches, polishing the user experience, optimizing the code and bug testing
Start sprint
WDT-22 Implement button hover animations USER EXPERIENCE
WDT-11 Implement navbar with required menu options RECEPTION MANAGEMENT DASH...
WDT-24 Bug testing and polishing
WDT-23 Create README file USER EXPERIENCE
Create issue



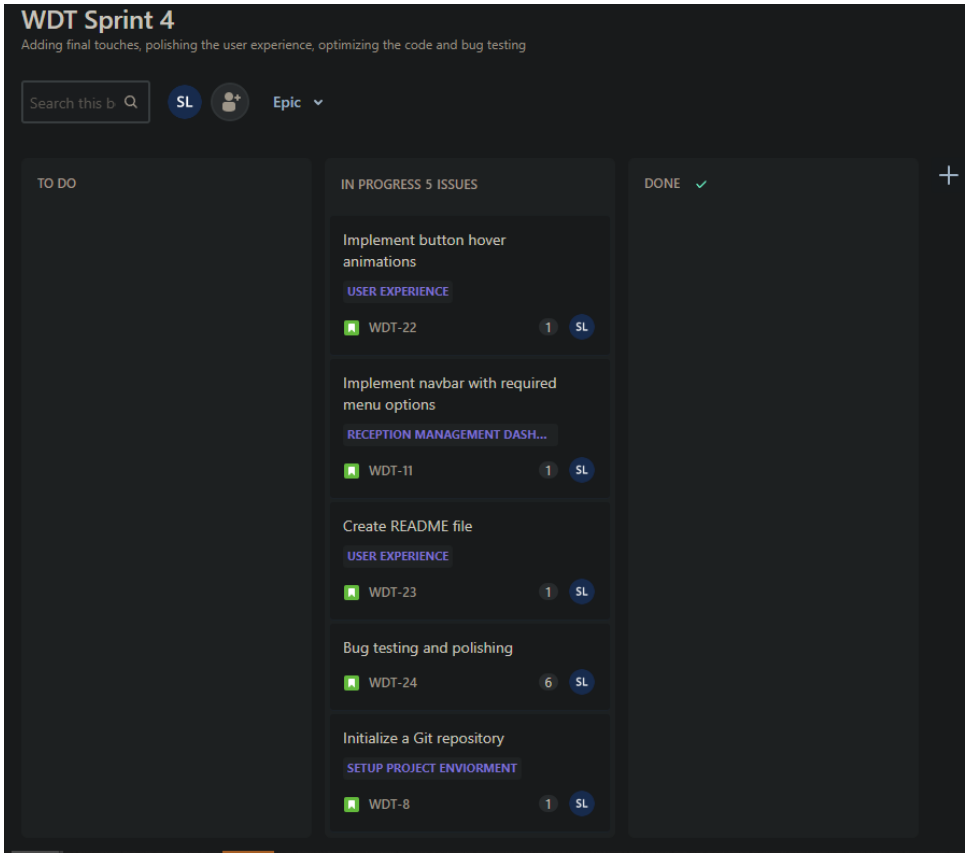
Started week 1 with setting up my work environment, creating the branding profile, and creating the staff table with bootstrap. Decided against initializing the git repository as it was a solo project, but in hindsight it probably would have been good practise to do it first eighter way. Went on the last two issues, and it seemed to match my estimated time that I set for them.



Week 2 was spent finishing the most important functionalities so that the company could in theory start using the “beta” version already.



Week 3 was spent implementing the toasts, logo and the clock function.



Week 4 was adding the hover animation, navbar, readme file along with initializing the git repository. This gave good room for bug testing and polishing.

Spent alot of time understanding if I were to add borders and border corners only around the edges of the table. Got it working with alot of experimenting and searching (https://stackoverflow.com/questions/4932181/rounded-table-corners-css-only) but it ended up looking nothing like the mock picture so I went with adding it to the table header and cells instead. After 2 days of attempting to get rounded corners on my table I eventually added overflow: hidden to my css table that made the rounded class work.

The API sometimes made duplicate staff members so I read up on cache-control (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control) and from my understanding the 'nocache' parameter with the 'new Date().getTime()' value is used in the API URL to ensure that each request bypasses any cached responses and fetches fresh data from the API. Edit: Kept seeing duplicates, attempting with `&timestamp=${new Date().getTime()}`. Not sure if the API even allows me to do cache control but atleast we tried.

Wanted to do a ternary operator on the delivery board code but it made it harder then an 'if else' statement when adding more vehicles. It would have been selfish towards the company; ${deliveryDriver.vehicle?.toLowerCase() === 'motorcycle' ? '' : ''}.

Was interesting that I could throw the entire script inside a function (IIFE) which doesn't serve a practical function nor solve any particular problems in the script, but could help with encapsulation, code isolation, readability and maintainability, minification and optimization and name pollution prevention, and it seemed to not have any negative effects, so I kept it. Edit: Upon further reflection I removed this as it's something the company can very easily add if they need it and I did not want to add things we don't need.

```
staffMemberIsLate() {
  const now = new Date();
  const expectedReturnDate = new Date(this.expectedReturn);
  if (now > expectedReturnDate) {
    this.timeElapsed = Math.floor((now - expectedReturnDate) / 60000);
    const timeElapsedMilliseconds = this.timeElapsed * 60000;
    const message = `${this.name} ${this.surname} has been out-of-office for ${timeElapsedMilliseconds} m
    const toast = `
      <div class="toast" role="alert">
        <div class="toast-header">
          <img src="${this.picture}" class="rounded me-2" alt="${this.name} ${this.surname}'s picture">
          <strong class="me-auto">${this.name} ${this.surname}</strong>
          <small>${this.outTime.toLocaleTimeString()}</small>
          <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
        </div>
        <div class="toast-body">${message}</div>
      </div>`;

    // Append the new toast
    $('#toast-container').append(toast);

    // Show the toast
    $('.toast').toast('show');
  } else {
    this.timeElapsed = 0;
  }
}
}
```

In my implementation, I tried first to directly display the toast notifications in the 'staffMemberIsLate' and 'deliveryDriverIsLate' functions. I think I was close but was unable to accurately determine the expected return time for each staff member and delivery driver.

I went for a new approach where I introduced two Maps, 'timeoutIds' and 'intervalIds' as global variables to store the timeout and interval IDs. These allowed me to manage and clear the timeouts and intervals created for each staff member and delivery driver.

By using 'setTimeout' I schedule a single toast notification for each staff member when they are expected to be late. The 'setTimeout' function is called once for each individual, with a delay calculated based on their expected return time. It then displays a toast notification if I have not yet clocked them in.

On the delivery driver I instead went with a 'setInterval' which checks if the delivery driver is late every second. The reason for choosing interval is that the company can very easily integrate a GPS API to each driver, so they get their location up on the dashboard or in the toast that updates with the late checker.