Semester Project 1 – Reflection Report

In this project, I have developed a JavaScript-based management dashboard system for staff and delivery drivers utilizing Object-Oriented Programming principles. The application manages staff and driver data, as well as tracking their "**in** or **out**" status, and notifies when a member is late based on their return time. In this report I will reflect on the design choices, technical challenges, and my personal learnings from this project.

From the very beginning I designed the application with the requirements from WeDeliverTECH in mind;

Backlog (19 of 20 issues visible)

KAN-39	The Company wants a notification system for staff and delivery drivers.
KAN-44	Enhance Reception Dashboard Functionality
KAN-45	Allow receptionists to easily mark staff as 'in' or 'out' and display this status instantly on the dashboard.
KAN-47	A toast should appear when a staff member and delivery driver is late.
KAN-49	The toast must stay on the screen until the receptionist closes the notification.
KAN-50	the 'In' button will clear their Out Time, Duration, and Expected Return Time cells and update their status
KAN-53	Manual input information about the delivery driver and their current delivery.
KAN-54	The driver's type of vehicle is (Motorcycle or Car).
KAN-55	Drivers info: name, surname, telephone number, delivery address and the return time need to be captured.
KAN-56	Clicking the 'Add' button will add the delivery information to the Delivery Board table
KAN-57	There must be vehicle icons used in the Delivery Board.
KAN-59	Implement a dynamic clock that updates every second.
KAN-61	The company brand must be reflected in this system (logo and styling)
KAN-62	The IT department requested Bootstrap with JavaScript/jQuery and OOP.
KAN-63	The IT department does not know NodeJS
KAN-65	Ensure all buttons have hover effects to improve user experience
KAN-67	The company has requested that you use the API (Application Programming Interface) https://randomuser.me/
KAN-68	The input data should be used to create the relevant Object, and the Delivery Driver table should then be filled with this object's data.
KAN-69	The Staff table should be filled with the newly created objects (created according to the diagram)

+ Create issue

I avoided the usage of **Node.js** and employed the **OOP** approach to create a clean code and promote reusability. The Parent class **Employee** serves as the base for its children **StaffMember** and **DeliveryDriver** classes. The classes serve a major role as their information is used in nearly every function.

The Company also specifically asked for the use of **jQuery** and **Bootstrap**. For dynamic content management and to enhance the user interaction, jQuery has been extensively used in the application. For its aesthetic purposes I decided to replace the alerts with **SweetAlert,** which also has very functional advantages delivering responsive feedback through modals and alerts.

To avoid bugs and errors, I implemented functions to regularly check the staff and drives delay, by using JavaScript's **setInterval** method. This ensures real-time monitoring and immediate feedback to the user interface.

Time calculations and time functions were also important for the application to work. One of the challenges was accurately calculating and comparing time data to determine if an employee or driver was late. Using the **Date** object for operation, ensured that all time data was consistently handled in a uniform format.

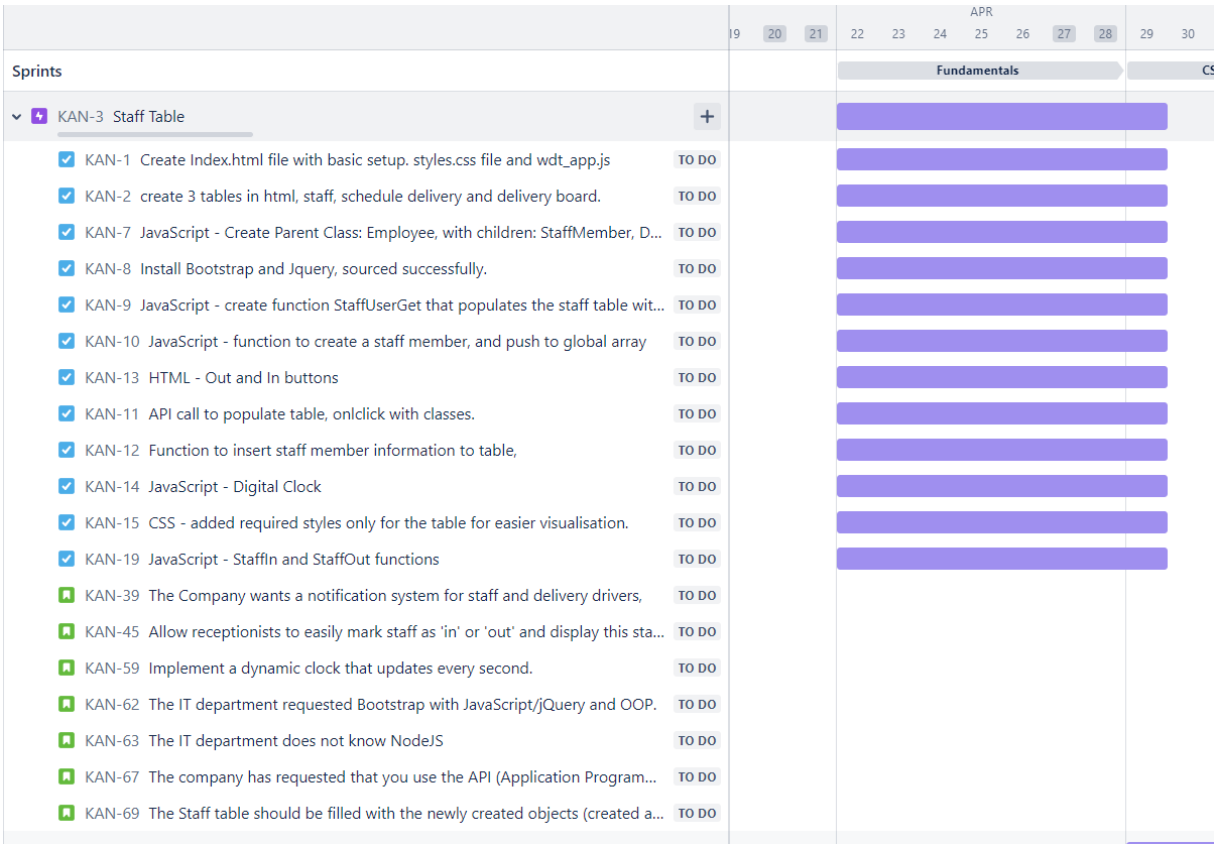
As a little fun bonus function, I added duplication handling, to prevent data redundancy. This was achieved using array manipulation methods like **.some()** to check for existing entries, and then prompting the user with SweetAlert to confirm their action.

I also added a autoComplete function, that auto fills the names generated from the API call, into the schedule delivery table automatically to make it easier for the receptionist to fill in the data. It is still possible to type manually.

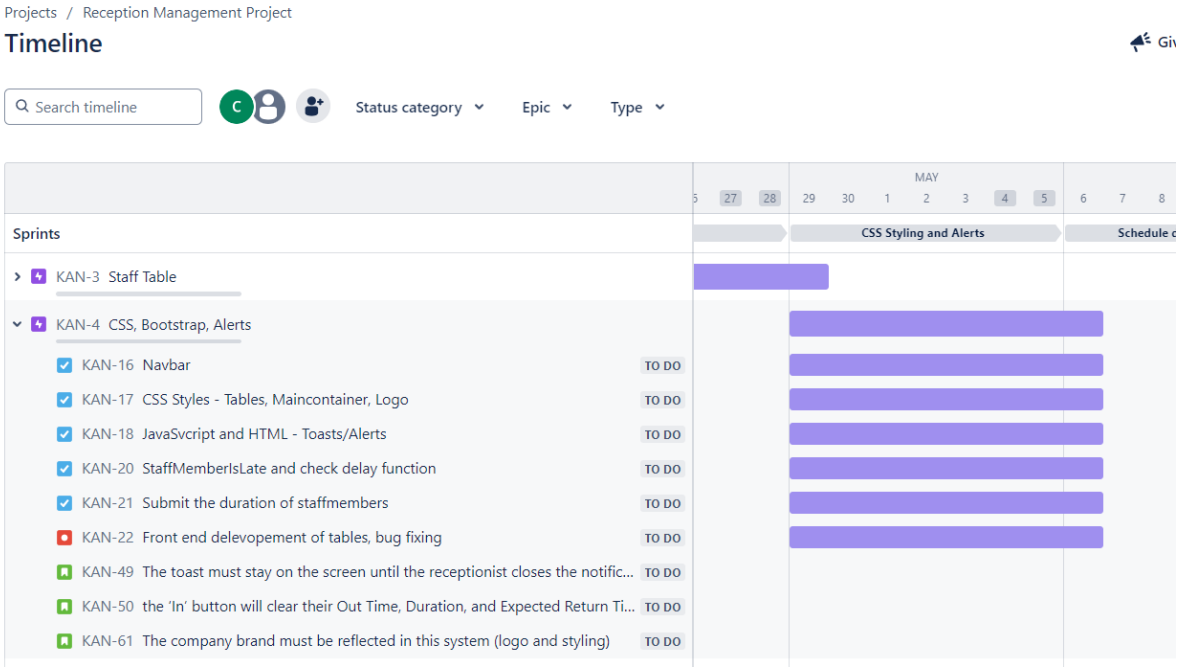
Ensuring the UI is responsive and accurately reflects the data state was challenging, especially with frequent updates. Using jQuery, I managed DOM manipulations effectively, such as adding or removing table rows dynamically based on the application state.

Timeline Screenshots from Jira,

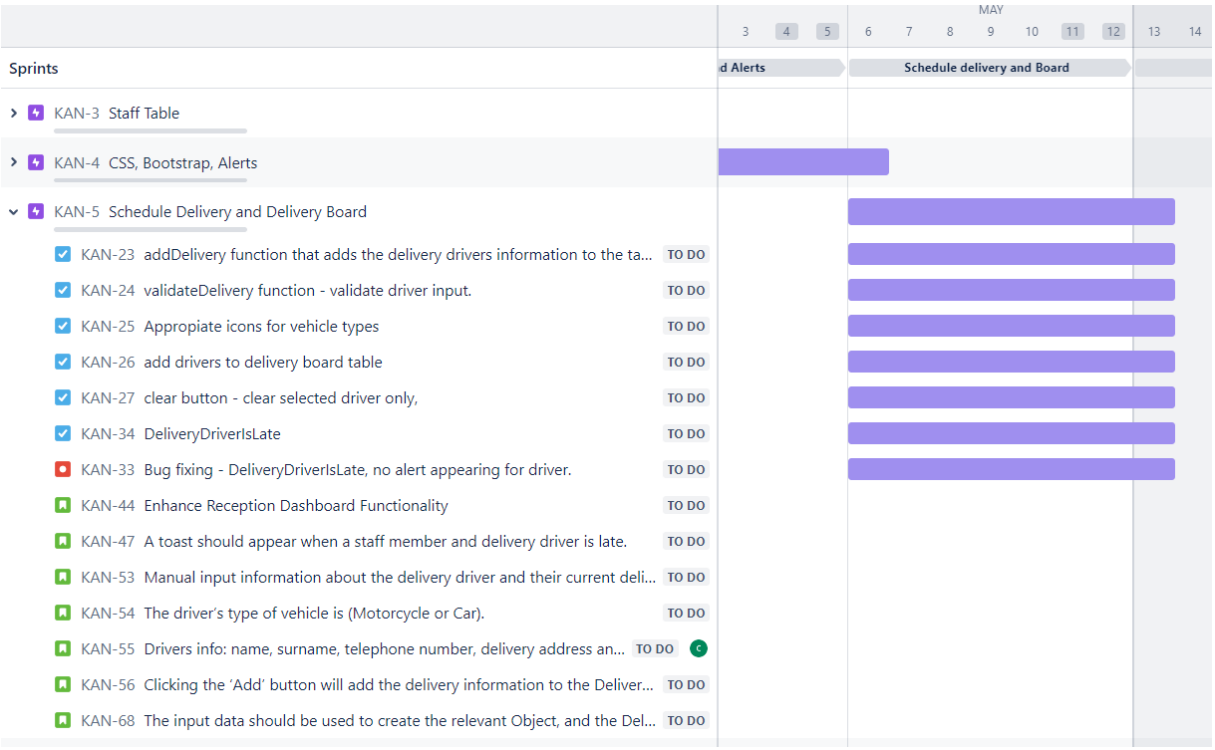
Week 1:


APR
19 20 21 22 23 24 25 26 27 28 29 30
Sprints
Fundamentals
KAN-3 Staff Table
KAN-1 Create Index.html file with basic setup. styles.css file and wdt_app.js TO DO
KAN-2 create 3 tables in html, staff, schedule delivery and delivery board. TO DO
KAN-7 JavaScript - Create Parent Class: Employee, with children: StaffMember, D... TO DO
KAN-8 Install Bootstrap and Jquery, sourced successfully. TO DO
KAN-9 JavaScript - create function StaffUserGet that populates the staff table wit... TO DO
KAN-10 JavaScript - function to create a staff member, and push to global array TO DO
KAN-13 HTML - Out and In buttons TO DO
KAN-11 API call to populate table, onlclick with classes. TO DO
KAN-12 Function to insert staff member information to table, TO DO
KAN-14 JavaScript - Digital Clock TO DO
KAN-15 CSS - added required styles only for the table for easier visualisation. TO DO
KAN-19 JavaScript - StaffIn and StaffOut functions TO DO
KAN-39 The Company wants a notification system for staff and delivery drivers, TO DO
KAN-45 Allow receptionists to easily mark staff as 'in' or 'out' and display this sta... TO DO
KAN-59 Implement a dynamic clock that updates every second. TO DO
KAN-62 The IT department requested Bootstrap with JavaScript/jQuery and OOP. TO DO
KAN-63 The IT department does not know NodeJS TO DO
KAN-67 The company has requested that you use the API (Application Program... TO DO
KAN-69 The Staff table should be filled with the newly created objects (created a... TO DO

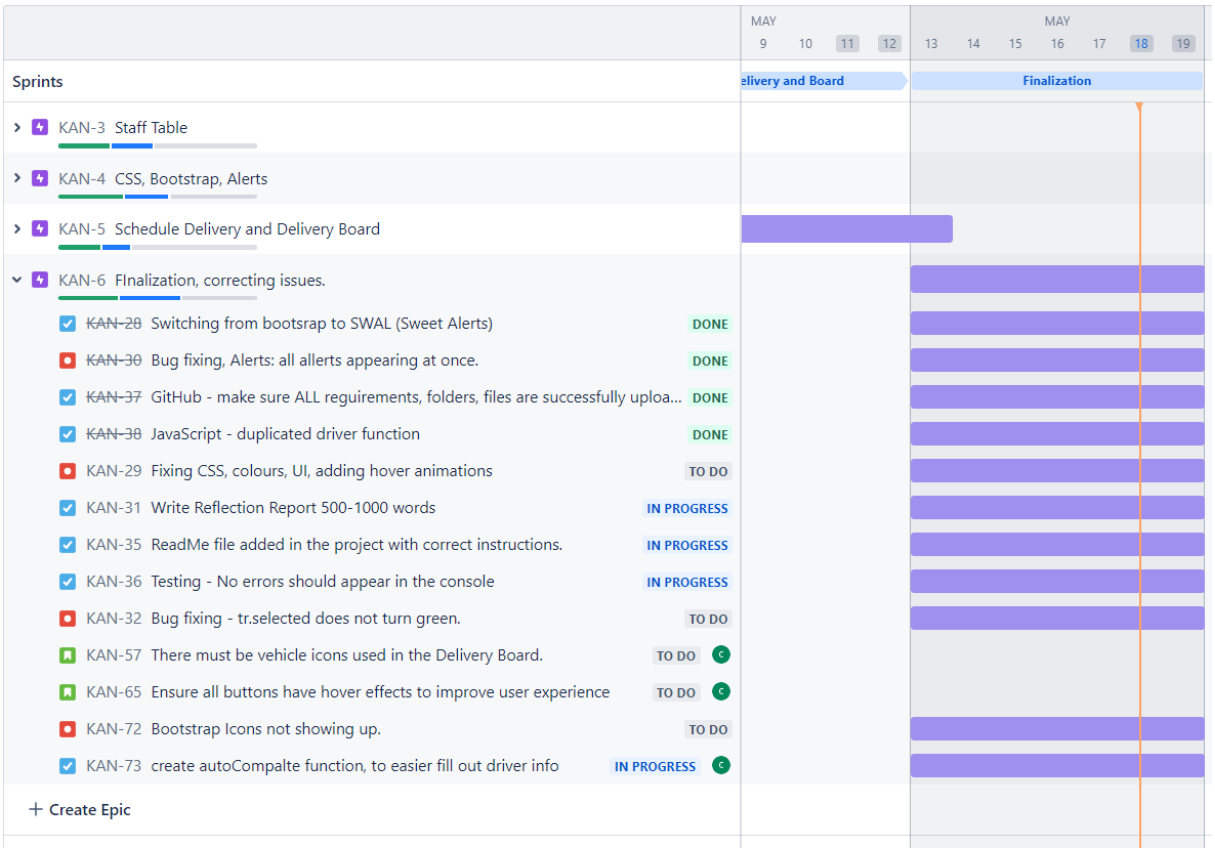
Week 2:


Projects / Reception Management Project
Timeline
Search timeline
Status category
Epic
Type
MAY
5 27 28 29 30 1 2 3 4 5 6 7 8
Sprints
CSS Styling and Alerts
KAN-3 Staff Table
KAN-4 CSS, Bootstrap, Alerts
KAN-16 Navbar TO DO
KAN-17 CSS Styles - Tables, Maincontainer, Logo TO DO
KAN-18 JavaSvcript and HTML - Toasts/Alerts TO DO
KAN-20 StaffMemberIsLate and check delay function TO DO
KAN-21 Submit the duration of staffmembers TO DO
KAN-22 Front end delevopement of tables, bug fixing TO DO
KAN-49 The toast must stay on the screen until the receptionist closes the notific... TO DO
KAN-50 the 'In' button will clear their Out Time, Duration, and Expected Return Ti... TO DO
KAN-61 The company brand must be reflected in this system (logo and styling) TO DO

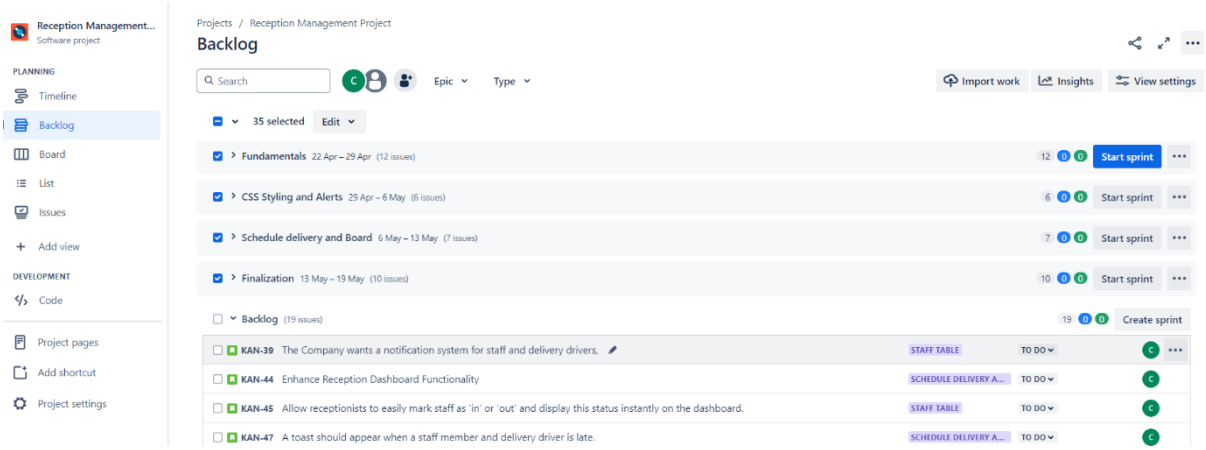
Week 3:


MAY
3 4 5 6 7 8 9 10 11 12 13 14
Sprints
d Alerts
Schedule delivery and Board
KAN-3 Staff Table
KAN-4 CSS, Bootstrap, Alerts
KAN-5 Schedule Delivery and Delivery Board
KAN-23 addDelivery function that adds the delivery drivers information to the ta... TO DO
KAN-24 validateDelivery function - validate driver input. TO DO
KAN-25 Appropiate icons for vehicle types TO DO
KAN-26 add drivers to delivery board table TO DO
KAN-27 clear button - clear selected driver only, TO DO
KAN-34 DeliveryDriverIsLate TO DO
KAN-33 Bug fixing - DeliveryDriverIsLate, no alert appearing for driver. TO DO
KAN-44 Enhance Reception Dashboard Functionality TO DO
KAN-47 A toast should appear when a staff member and delivery driver is late. TO DO
KAN-53 Manual input information about the delivery driver and their current deli... TO DO
KAN-54 The driver's type of vehicle is (Motorcycle or Car). TO DO
KAN-55 Drivers info: name, surname, telephone number, delivery address an... TO DO
KAN-56 Clicking the 'Add' button will add the delivery information to the Deliver... TO DO
KAN-68 The input data should be used to create the relevant Object, and the Del... TO DO

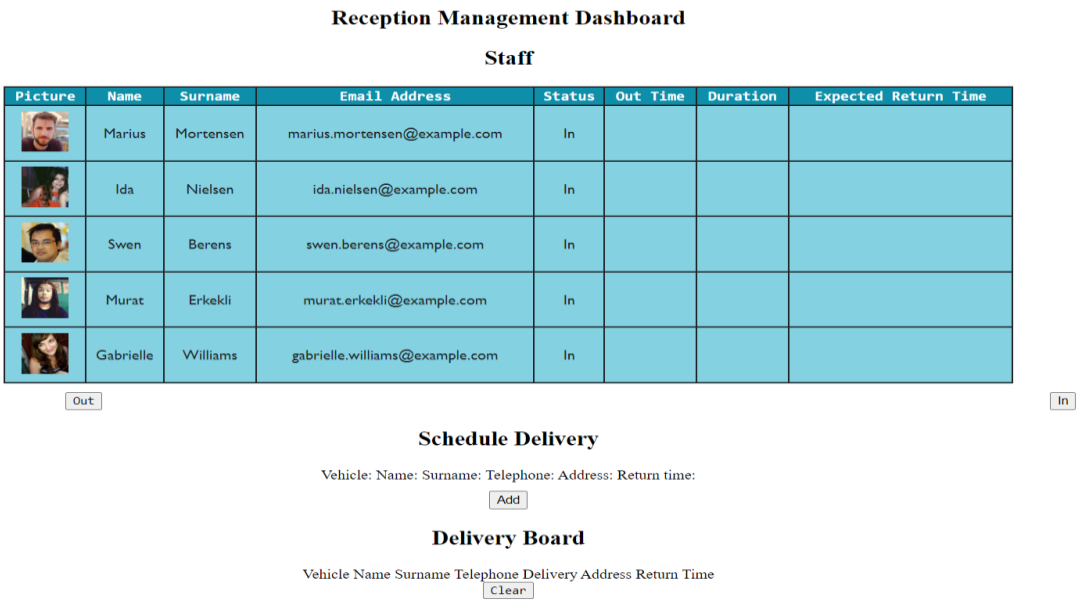
Week 4:


MAY
9 10 11 12 13 14 15 16 17 18 19
Sprints
elivery and Board
Finalization
KAN-3 Staff Table
KAN-4 CSS, Bootstrap, Alerts
KAN-5 Schedule Delivery and Delivery Board
KAN-6 FInalization, correcting issues.
KAN-28 Switching from bootsrap to SWAL (Sweet Alerts) DONE
KAN-30 Bug fixing, Alerts: all allerts appearing at once. DONE
KAN-37 GitHub - make sure ALL requirements, folders, files are successfully uploa... DONE
KAN-38 JavaScript - duplicated driver function DONE
KAN-29 Fixing CSS, colours, UI, adding hover animations TO DO
KAN-31 Write Reflection Report 500-1000 words IN PROGRESS
KAN-35 ReadMe file added in the project with correct instructions. IN PROGRESS
KAN-36 Testing - No errors should appear in the console IN PROGRESS
KAN-32 Bug fixing - tr.selected does not turn green. TO DO
KAN-57 There must be vehicle icons used in the Delivery Board. TO DO
KAN-65 Ensure all buttons have hover effects to improve user experience TO DO
KAN-72 Bootstrap Icons not showing up. TO DO
KAN-73 create autoCompalte function, to easier fill out driver info IN PROGRESS
+ Create Epic

Backlog:



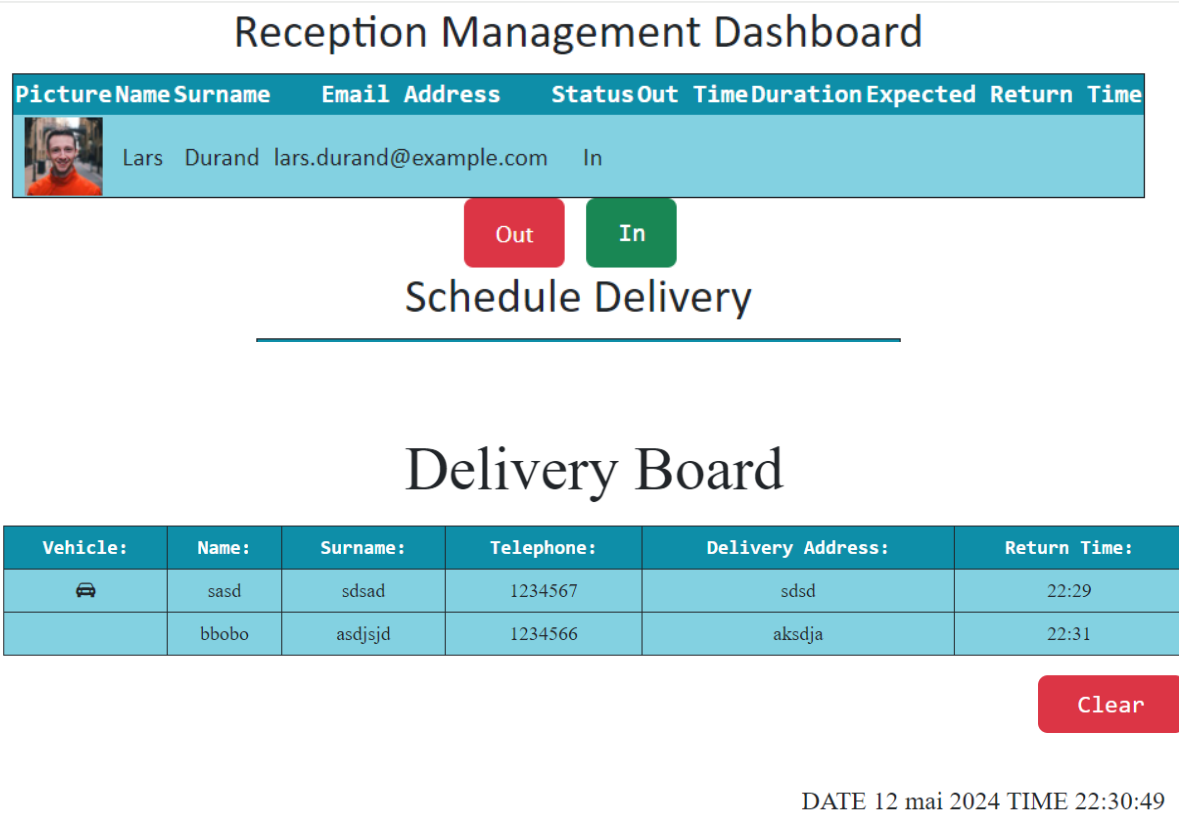
This image shows an early development stage from the first week. This phase laid the foundational elements of the project, integrating basic HTML and CSS with essential JavaScript functionalities. Successfully implementing API calls and filling the staff table with data, was a crucial first step to build more complex functionalities:



Bugs and Solutions.

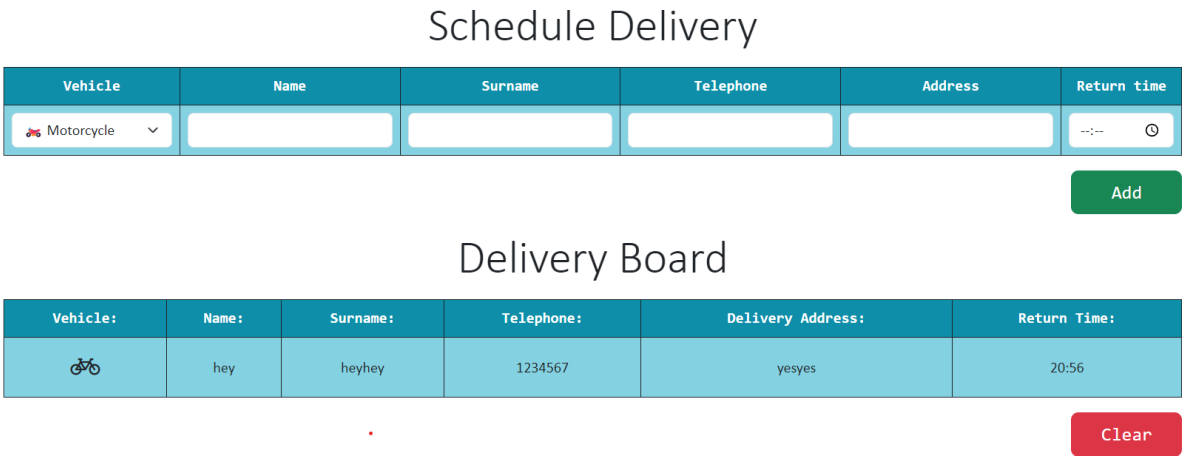
1. API Call Inconsistency

- **Problem:** Initially, API calls to populate staff data were inconsistent, resulting in random or incomplete data retrieval.
- **Solution:** Modified the API request URL to include a specific parameter (**/?results=5**), ensuring a consistent retrieval of exactly five staff entries. This adjustment provided a reliable and customizable method for data generation.



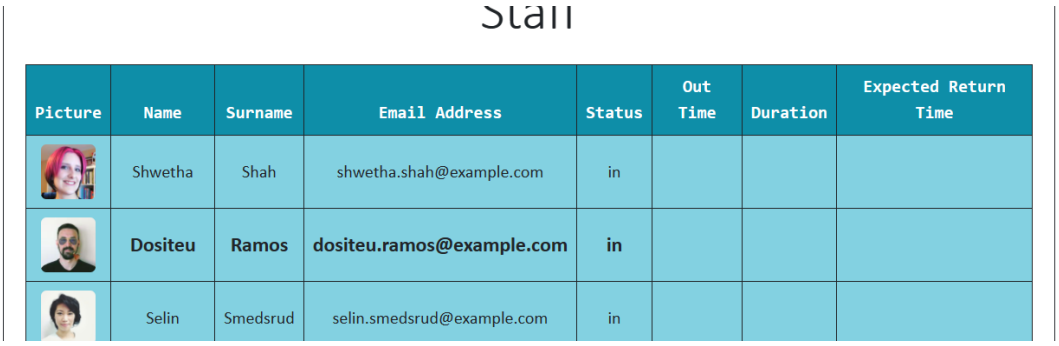
2. Icon Display Issues

- **Problem:** Icons for vehicles, particularly motorcycles, were not displaying correctly, showing as undefined or not appearing at all.
- **Solution:** Due to the absence of a suitable motorcycle icon in Bootstrap, alternative emojies were considered for the html file. The issue was circumvented by selecting compatible icons that closely matched the intended visuals.



3. CSS Styling Conflicts

- **Problem:** The most challenging issue was CSS conflicts with Bootstrap, particularly with the **.selected** class on table rows, which was not displaying as intended.
- **Solution:** After experimenting with various Bootstrap versions and stylesheet arrangements, a visual cue was implemented by enhancing the size and boldness of selected rows, ensuring clear visibility of selection and interaction, despite the inability to resolve the color conflict directly.

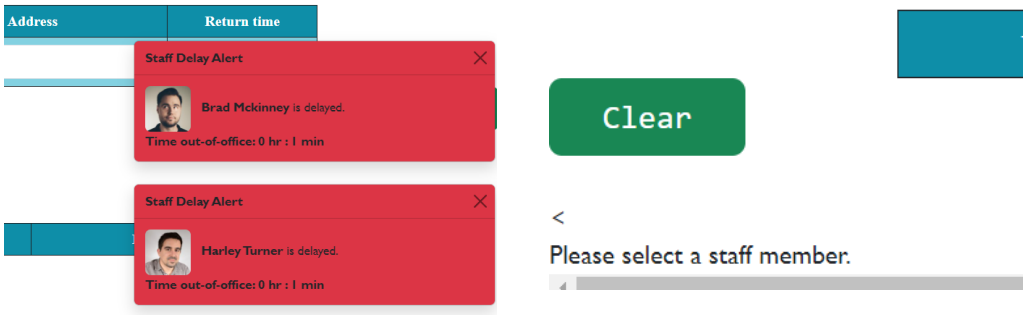



```
#staffTableBody tr.selected {
  background-color: #198754 !important;
  font-weight: bolder;
  font-size: 19px;
}

#deliveryTable tr.selected {
  background-color: #198754 !important;
  font-weight: bold;
  font-size: 19px;
}
```

4. Alert Overload and Functionality Bugs

- **Problem:** Encountered issues with simultaneous alert pop-ups and JavaScript functions related to delivery delays not triggering as expected.
- **Solution:** Implemented controlled sequencing for alerts and refined the JavaScript logic to ensure timely and appropriate function execution. All of the alerts appeared at the bottom of the page, and they would sometimes stack or appear simultaneously.



5. Interface Responsiveness

- **Problem:** Adding new entries sometimes resulted in empty rows or undefined data in the delivery board.
- **Solution:** Debugged and adjusted the event handling script associated with the 'Add' button to validate data before insertion, preventing the creation of undefined entries.

Delivery Board

Vehicle:	Name:	Surname:	Telephone:	Delivery Address:	Return Time:
Car	undefined	undefined	1234567		
Motorcycle	undefined	undefined	7654321		

Delivery Board

Vehicle:	Name:	Surname:	Telephone:	Delivery Address:	Return Time:
Car			1234567		
Motorcycle			1234567		

Jira Boards:

Projects / Reception Management Project

Fundamentals

Starting the project by creating user stories, checking requirements from the client, creating required files: index.html, styles.css and wdt_app.js. The client asked for Jquery, bootst

Search | Epic

TO DO 3

- Function to insert staff member information to table, — STAFF TABLE — KAN-12
- JavaScript - Digital Clock — STAFF TABLE — KAN-14
- CSS - added required styles only for the table for easier visualisation. — STAFF TABLE — KAN-15

+ Create issue

IN PROGRESS 4

- JavaScript - create function StaffUserGet that populates the staff table with API. — STAFF TABLE — KAN-9
- JavaScript - function to create a staff member, and push to global array — STAFF TABLE — KAN-10
- API call to populate table, onlclick with classes. — STAFF TABLE — KAN-11
- JavaScript - StaffIn and StaffOut functions — STAFF TABLE — KAN-19

+ Create issue

DONE 5

- Create Index.html file with basic setup. styles.css file and wdt_app.js — STAFF TABLE — ~~KAN-1~~
- create 3 tables in html, staff, schedule delivery and delivery board. — STAFF TABLE — ~~KAN-2~~
- JavaScript - Create Parent Class: Employee, with children: StaffMember, DeliveryDriver. — STAFF TABLE — ~~KAN-7~~
- HTML - Out and In buttons — STAFF TABLE — ~~KAN-13~~
- Install Bootstrap and Jquery, sourced successfully. — STAFF TABLE — ~~KAN-8~~

Projects / Reception Management Project

CSS Styling and Alerts

Adding more visuals for UI like Navbar, Logo, styling of the tables, headers, the Maincontainer of the page. I also added and fixed the toasts/Alerts to notify the receptionist for s

Search | Epic | Type | Sprint 1 | Clear filters

TO DO 1 OF 4

- Front end delevopement of tables, bug fixing — CSS, BOOTSTRAP, ALERTS — KAN-22

+ Create issue

IN PROGRESS 2 OF 6

- JavaSvcript and HTML - Toasts/Alerts — CSS, BOOTSTRAP, ALERTS — KAN-18
- StaffMemberIsLate and check delay function — CSS, BOOTSTRAP, ALERTS — KAN-20

DONE 3 OF 8

- Navbar — CSS, BOOTSTRAP, ALERTS — ~~KAN-16~~
- CSS Styles - Tables, Maincontainer, Logo — CSS, BOOTSTRAP, ALERTS — ~~KAN-17~~
- Submit the duration of staffmembers — CSS, BOOTSTRAP, ALERTS — ~~KAN-21~~

Projects / Reception Management Project

Schedule delivery and Board

The third week is all about the delivery driver functions and updating the delivery board with their manually inputed data. The client asked for these functions specifically: addDeliver

Search | Epic | Type | Sprint 1 | Clear filters

TO DO 2 OF 6

- Appropiate icons for vehicle types
  SCHEDULE DELIVERY AND DELIVERY BOA...
  KAN-25
- Bug fixing - DeliveryDriverIsLate, no alert appearing for driver.
  SCHEDULE DELIVERY AND DELIVERY BOA...
  KAN-33

+ Create issue

IN PROGRESS 2 OF 8

- validateDelivery function - validate driver input.
  SCHEDULE DELIVERY AND DELIVERY BOA...
  KAN-24
- DeliveryDriverIsLate
  SCHEDULE DELIVERY AND DELIVERY BOA...
  KAN-34

DONE 3 OF 11

- addDelivery function that adds the delivery drivers information to the table.
  SCHEDULE DELIVERY AND DELIVERY BOA...
  ~~KAN-23~~
- add drivers to delivery board table
  SCHEDULE DELIVERY AND DELIVERY BOA...
  ~~KAN-26~~
- clear button - clear selected driver only,
  SCHEDULE DELIVERY AND DELIVERY BOA...
  ~~KAN-27~~

Projects / Reception Management Project

Finalization

For the last week all the requirements from the clients should have already been added to the webpage. This week should focus on bug fixing, UI, final touches and s

Search | Epic 1 | Type | Sprint 1 | Clear filters

TO DO 3 OF 9

- Fixing CSS, colours, UI, adding hover animations
  FINALIZATION, CORRECTING ISSUES.
  KAN-29
- Bug fixing - tr.selected does not turn green.
  FINALIZATION, CORRECTING ISSUES.
  KAN-32
- Bootstrap Icons not showing up.
  FINALIZATION, CORRECTING ISSUES.
  KAN-72

+ Create issue

IN PROGRESS 4 OF 12

- Write Reflection Report 500-1000 words
  FINALIZATION, CORRECTING ISSUES.
  KAN-31
- ReadMe file added in the project with correct instructions.
  FINALIZATION, CORRECTING ISSUES.
  KAN-35
- Testing - No errors should appear in the console
  FINALIZATION, CORRECTING ISSUES.
  KAN-36
- create autoComplete function, to easier fill out driver info
  FINALIZATION, CORRECTING ISSUES.
  KAN-73

DONE 4 OF 15

- Switching from bootsrap to SWAL (Sweet Alerts)
  FINALIZATION, CORRECTING ISSUES.
  ~~KAN-28~~
- Bug fixing, Alerts: all allerts appearing at once.
  FINALIZATION, CORRECTING ISSUES.
  ~~KAN-30~~
- GitHub - make sure ALL reguirements, folders, files are successfully uploaded.
  FINALIZATION, CORRECTING ISSUES.
  ~~KAN-37~~
- JavaScript - duplicated driver function
  FINALIZATION, CORRECTING ISSUES.
  ~~KAN-38~~

Lessons learned.

Despite encountering numerous technical challenges, I believe the solutions were effectively implemented for each issue, ensuring the project met WeDeliverTECH's requirements. Some solutions, like handling the **.selected** class and **tr** element styling, weren't ideal but were practical under the circumstances of **Bootstrap** conflicts. The final product successfully mirrored the initial mock-up, containing all critical functionalities asked in the brief.

This project was a big learning experience, marking my first experience in building an application entirely from scratch. It created a trust in my coding abilities, free from reliance on project resources and my teachers assistance. The use of Object-Oriented Programming (OOP) principles, particularly classes and inheritance in JavaScript, provided a steep learning-curve. Managing highly difficult operations with jQuery highlighted the critical skill of precise timing and response handling, which was crucial to the project's dynamic functionality.

Furthermore, integrating user feedback mechanisms such as alerts and notifications emphasized the importance of interactive elements in maintaining user engagement. The development process underscored the significance of methodical problem-solving and design considerations, which were crucial in addressing the complexities of the project. This approach not only enhanced my technical skills but also boosted my confidence, preparing me for future challenges.

Conclusion.

The development of this application was not only a comprehensive technical challenge, but also a significant personal development milestone. It reinforced the importance of detailed planning, user-centric design, and repeated testing. This project has fundamentally shifted my approach to software development by building a disciplined yet creative mindset that I look forward to bringing to my future projects.