Reflection Report SPCA

1. Digital Clock Functionality: The `digitalClock` function is responsible for displaying the current date and time in a specific format. It gets the current date and time using the `Date` object, formats it, and then displays it in an HTML element with the id 'clock'. This function is called every second using the `setInterval` function.

2. Employee, StaffMember, and DeliveryDriver Classes: These classes are used to create objects that represent employees, staff members, and delivery drivers. The `Employee` class is the base class and includes properties for the name and surname of the employee. The `StaffMember` and `DeliveryDriver` classes extend the `Employee` class and add additional properties specific to staff members and delivery drivers.

3. StaffUserGet Function: The code is wrapped in a `$(document).ready()` function to ensure that the HTML document is fully loaded before the JavaScript code runs. An AJAX request is made to the Random User API to fetch data for 5 users. The `dataType` is set to 'json' to specify that the data should be returned as JSON. The success function of the AJAX request handles the response data. It gets a reference to the 'staffTable' element and then loops through the results from the API. Creating StaffMember Objects and Table Rows: For each result from the API, a new `StaffMember` object is created and a new row is added to the 'staffTable'. The `StaffMember` object is created with the first name, last name, thumbnail picture, and email from the result. The new row is created with cells for the picture, name, surname, email, status, out time, duration, and expected return time of the `StaffMember`. The `StaffMember` object is also attached to the row using jQuery's `.data()` method. Adding StaffMember Objects to Array: The `StaffMember` object is added to the `staffMembers` array. This array is used to store all the `StaffMember` objects.

4. Row Click Event: An event listener is added to the rows of the 'staffTable'. When a row is clicked, it checks if the row has the 'selected' class. If it does, it removes the 'selected' class and sets `selectedStaffMember` to null. If it doesn't, it removes the 'selected' class from all rows, adds the 'selected' class to the clicked row, and sets `selectedStaffMember` to the `StaffMember` object attached to the clicked row.

5. StaffOut Function: The code is wrapped in a `$(document).ready()` function to ensure that the HTML document is fully loaded before the JavaScript code runs. An event listener is added to the 'confirmButton' element that triggers when the button is clicked. The value of the 'durationInput' element is retrieved and converted to an integer. If the value is a valid number, the code continues to execute. The current time is retrieved, and the expected return time is calculated by adding the duration to the current time. The current time is also formatted to a string that only includes the hours and minutes. The status, out time, duration, and expected return time of the selected staff member are updated. The duration is formatted to a string that

includes the number of hours and minutes. The expected return time is formatted to a string that only includes the hours and minutes. The staff members array is updated with the new data of the selected staff member. The index of the selected staff member in the array is found and if the staff member is in the array, their data is updated.

6. StaffMemberIsLate Function: This function checks if any staff member is late. It gets the current time and then loops through the `staffMembers` array. For each staff member, it checks if their status is 'Out'. If it is, it gets the expected return time of the staff member and converts it to a `Date` object. It then checks if the current time is greater than the expected return time plus 1 second. If it is, it calculates how many minutes the staff member is late and sets the `staffMemberIsLate` property of the staff member to this value. If a toast has not already been created for the staff member, it creates and shows a toast with the staff member's picture, a title that says the staff member is late, and a body that says how many minutes the staff member is late. The toast is appended to the 'lateToastContainer' element, set to not autohide, and shown. The `toastCreated` property of the staff member is set to true. If a toast has already been created for the staff member, it updates the toast with the staff member's picture, a title that says the staff member is late, and a body that says the staff member is 'Minutes Out of Office'. The `checkIfStaffMemberIsLate` function is set to run every 1 second using `setInterval`.

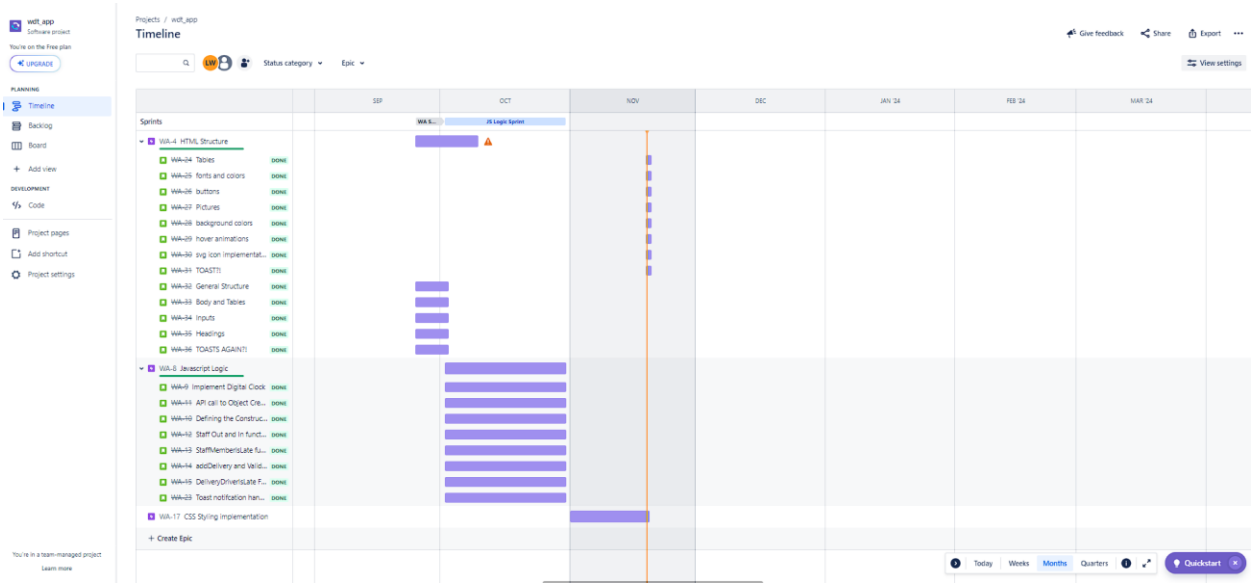
7. addDelivery and ValidateDelivery function: An event listener is added to the 'addButton' element that triggers when the button is clicked. The values of the input fields are retrieved and validated. If any of the fields are empty or if the telephone number is less than 7 digits, an error message is added to the errors array. If there are any errors, a toast is created with the error messages and shown. The toast is appended to the 'toastContainerOne' element, set to not autohide, and shown. **: If there are no errors, a new DeliveryDriver object is created with the input values and added to the deliveryDrivers array. A new row is added to the 'deliveryBoardTable' with the new delivery driver's data. The input fields and the selected vehicle and its icon are reset.

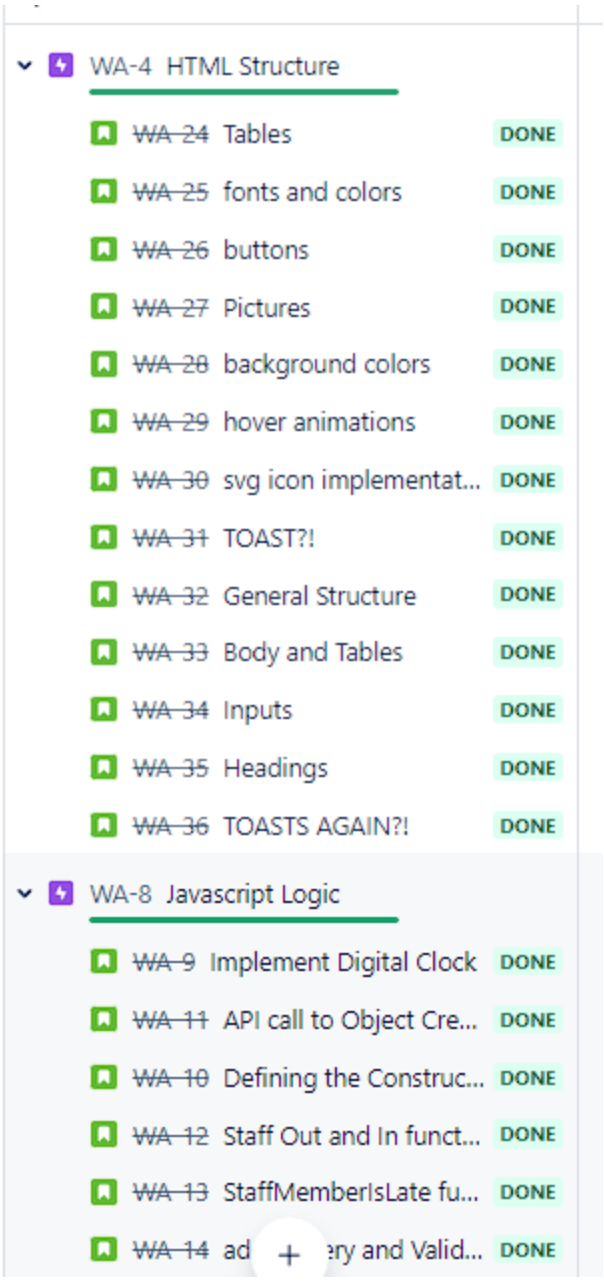
8. deliveryDriverIsLate Function: This function checks if any delivery driver is late. It gets the current time and then loops through the `deliveryDrivers` array. For each delivery driver, it gets the expected return time of the delivery driver and converts it to a `Date` object. It then checks if the current time is greater than the expected return time. If it is, and a toast has not already been created for the delivery driver, it calculates how many minutes the delivery driver is late. It creates and shows a toast with the delivery driver's details. The toast is appended to the 'deliveryToastContainer' element, set to autohide after a delay, and shown. The `toastCreated` property of the delivery driver is set to true. The `deliveryDriverIsLate` function is set to run every 1 second using `setInterval`.

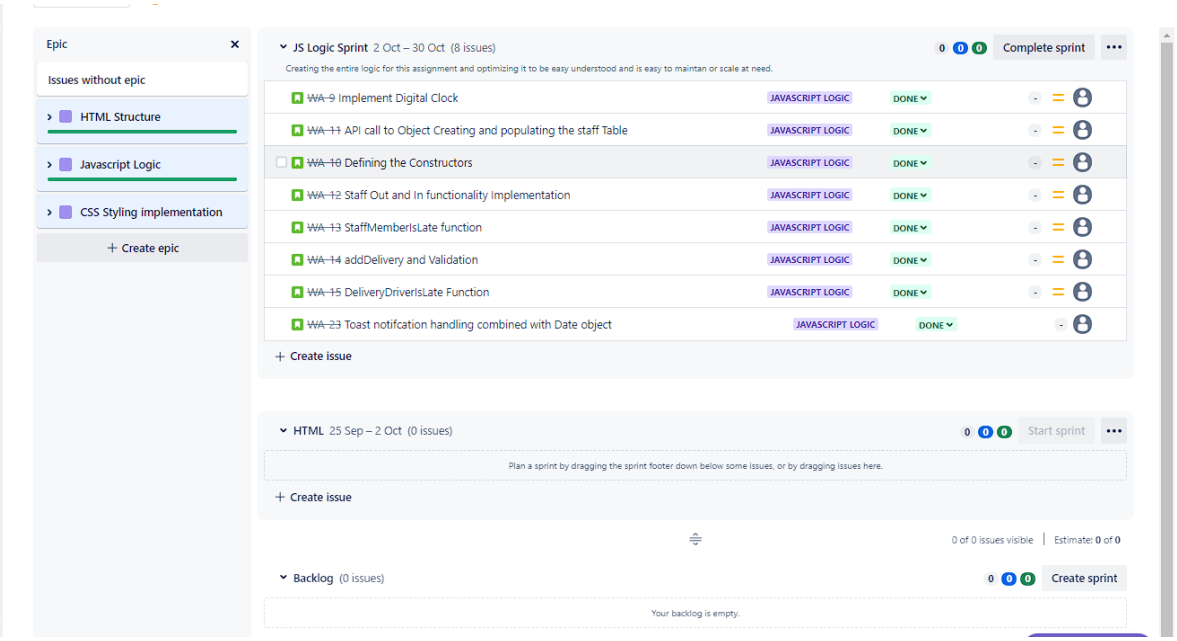
For me personally this project has been difficult, but a lot was learned. The biggest issues in this project for me have been the time formatting of the different objects to make the toast appear at the right times displaying the correct amount of time away.. man 😊tough one. Creating the entire logic for this assignment and optimizing it to be easy understood and is easy to maintain or scale at need was probably the best thing I learned from this assignment, I guess that's how one would develop software in the real world? I really hope we soon will dive more into backend programming tho!

About Jira:

I think I have misunderstood something very badly, I figure you guys wanted us to start the Jira project and not start it when everything is already done. I will add some screenshots to show what I tried to understand atleast.




WA-4 HTML Structure
WA-24 Tables DONE
WA-25 fonts and colors DONE
WA-26 buttons DONE
WA-27 Pictures DONE
WA-28 background colors DONE
WA-29 hover animations DONE
WA-30 svg icon implementat... DONE
WA-31 TOAST?! DONE
WA-32 General Structure DONE
WA-33 Body and Tables DONE
WA-34 Inputs DONE
WA-35 Headings DONE
WA-36 TOASTS AGAIN?! DONE
WA-8 Javascript Logic
WA-9 Implement Digital Clock DONE
WA-11 API call to Object Cre... DONE
WA-10 Defining the Construc... DONE
WA-12 Staff Out and In funct... DONE
WA-13 StaffMemberIsLate fu... DONE
WA-14 ad ery and Valid... DONE



Yeah Idk. This has been super wonky! This webpage feels very glitchy, there are sprints that I can’t, see?? Child issues of epics I can’t find, I think I got the general sense of it Tho.