2022

Design Documentation

PHASE ONE – FRONT END

RESONATE

Table of Contents

Overview of Project

As the services listed on in-station train message boards typically only display the origin and destination, it is sometimes difficult to work out which train a customer may need to board. Having a web application that can provide personalized information depending on a customer's interaction may aid them in their planning and ease the boarding process.

National Rail[1] currently has a public information engine called Darwin[2] which provides real time arrivals and departure predictions, and it is the only system in the United Kingdom that takes feeds directly from every TOC[3] customer information system (CIS), combining it with train location data provided by the railway infrastructure manager, National Rail. Unfortunately, apart from Darwin's 'Historic Service Performance' data feed, all others use SOAP rather than REST[4], making the initial development slightly more complex. Luckily, Huxley2 Community Edition[5] provides us with a CORS[6] enabled proxy for Darwin, allowing for ease-of-use development for client-side browser-based applications[7].

The main aim of this project is to retrieve relevant information from the Huxley2 API and display it, in a helpful way, to the user.

[1] National Rail - an unincorporated association whose membership consists of the passenger train operating companies (TOCs) of England, Scotland, and Wales.

[2] Darwin - the GB rail industry's official train running information engine, providing real-time arrival and departure predictions, platform numbers, delay estimates, schedule changes and cancellations. It is the only system in the UK to take feeds directly from every TOC customer information system (CIS), combining it with train location data provided by the railway infrastructure manager, Network Rail.

[3] TOC - Train operating companies (TOCs) run passenger services, leasing and managing stations from Network Rail.

[4] SOAP vs REST - what is the difference between SOAP and REST APIs

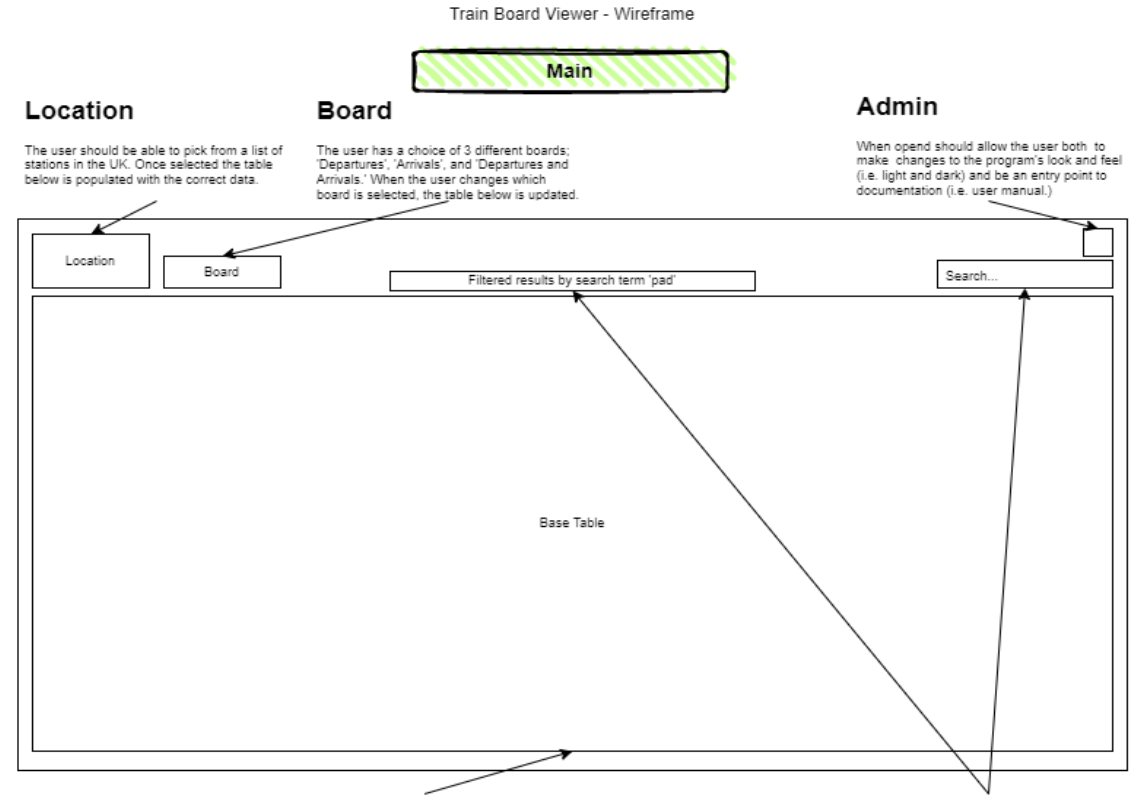
[5] Huxley2 Community Edition - a CORS enabled cross-platform JSON ReST proxy for the GB NRE LDB WCF SOAP XML API (called Darwin). It supports both the Public Version (PV) and the Staff Version (SV). It's built with ASP.NET Core LTS (.NET 6.0), C# 10

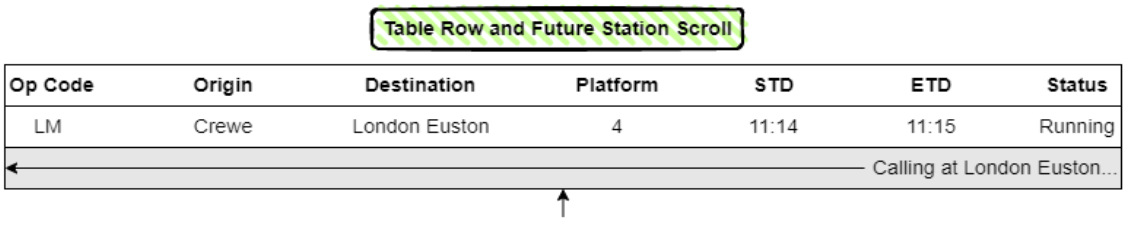
[6] CORS - a HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.

[7] Single page application - is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages. The goal is faster transitions that make the website feel more like a native app.

Wireframe

Please see below for an initial wire mock annotated with brief explanations of the project. Use this to guide you as you programmatically design your front end. Please note that the diagrams have been created using Draw.io[8] and originals can be provided upon request.





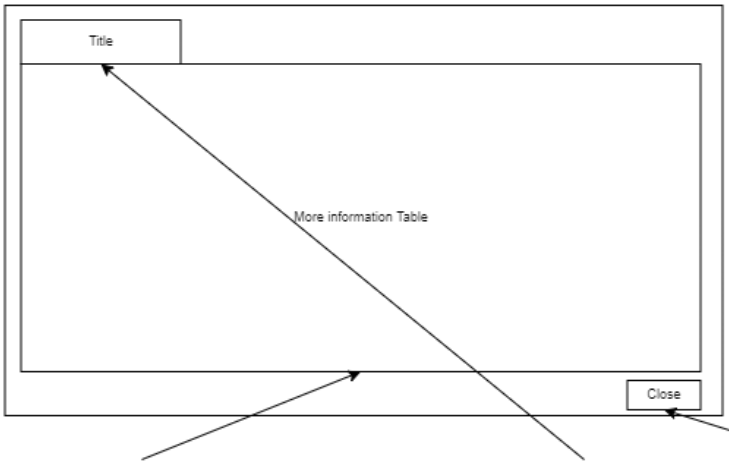
Displayed Service

When the user hovers over a specific service on the base table, scrolling text should apear below that service giving the user more information. This should inclide where the service hs called at / is calling at, and the amount of carrages that form the train. Double clicking on a service should mke a More Information modal apear.

[8] Draw.io - free diagram creation software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams.

Train Board Viewer - Wireframe

More Information Modal



More information Table

Provides more information about a specific service. The datapoints are as follows: 'Generated at', 'Service type', 'Location name', 'CRS', 'Operator', 'RSID', 'Is cancelled', 'Cancel reason', 'Delay reason', 'Overdue message', 'Length', 'Detached front', 'Is reverse formation', 'Platform', 'STA', 'ETA', 'ATA', 'STD', 'ETD', and 'STD'

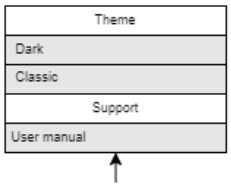
Title

Displaies identifier showing the user which services more inforation modal is opened.

Close Button

When clicked, closes the modal. Please note that if the user clickes anywhere else on the screen (outside of the modal) the modal should also close.

Admin Menu



Admin menu cog

When the user clicks the admin menu, a drop down should apear givin them options to change to look and feel through dark and light mode, or see the applications user manual in a new browser tab.

Epic, Feature, User Story, and Tasks

Epic 10001

10001: Train Board Viewer - 2023

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Objectives

- To aid National Rail customers in planning their journey by providing detailed information about the state of the railway.

Problem Statement

When using NR train services, it is sometime difficult to understand the various boards in different stations, and as most services displayed only reveal the destination of a schedule, finding the data points that are valuable is sometimes confusing and time consuming.

Benefit Hypothesis

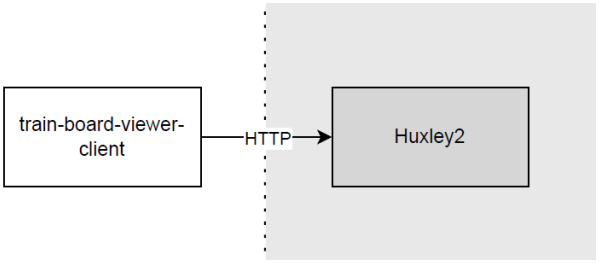
- To show an understanding of difficulties our customers customer may have when viewing in-station boards and obtaining relevant information to them with regards to the current live state of the railway.
- To empower our customers customer by allowing them to view every station message board remotely.

Scope

- Much of the scope (in terms of what is included) has been provided within the wireframe mocks. This includes:
  - Displaying 3 different boards depending on the user's choice; Departures, Arrivals, and Departures and Arrivals
  - Display Schedule information relating to any train station in the UK
  - Find additional information relating to a particular service
  - Filter the displayed services based on a search term
  - Light and dark mode
  - User manual
- With regards to what is out of scope:
  - We are only retrieving and displaying information provided by the Huxley2 API. This data is not checked, manipulated, or changed. Should there be an issue with their data, that does not fall under the applications responsibility.

Technical Design

- Below provides a diagram exspaining the overall archetecture of the application.



Feature 2001

20001: Home page to retrieve and display multiple services data

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Problem Statement

When using National Rail train services, currently we are not able to see the various train boards located at different stations remotely. Having this ability may aid a user in making decisions when planning their journey.

Benefit Hypothesis

- Receive service schedule information based on a chosen location.
- Decide what service schedule information table I am receiving (for e.g., Departures or Arrivals.)

Approach

- Build a front end only application that mocks responses from the Huxley2 API and displays service information to the user.

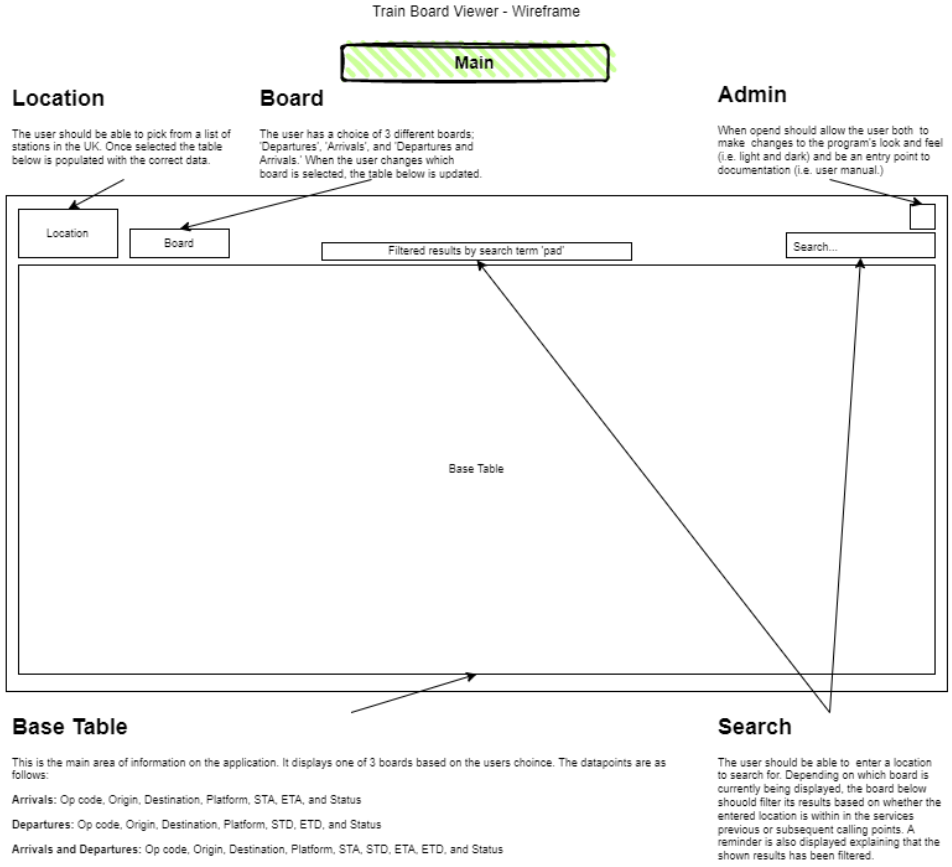
Time Sheet Code

- LD3847583 – New Starters (Training Project)

Risks

Technical Design

- Below provides a wireframe mock of the main application.



User Story 30001

30001: Set up application

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want the ability to view the most up to date version of the message board viewer application.

Acceptance Criteria

- Have a message board viewer client application built using the Angular client[9].
- Allow Resonate to keep track of the applications progress by setting up sensible branches / policies and monitor progress using Git.[10]

TASK 40001

40001: Create a new repository in Azure DevOps [PLACEHOLDER]

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create repository in Azure DevOps[11]
- In Azure DevOps, create sensible branches and set branch policies.

Hint

- If confused by some of the branch policies titles, look to the repository LuminateClient for inspiration.
- It would also be a good idea at this point to clone the LuminateClient repository to your local workspace. This will be referenced throughout the project.

TASK 40002

40002: Create MessageBoardViewerClient Application

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Using the Angular client, set up a new project titled MessageBoardViewerClient in your local repository.
- Open in your IDE and test the application by running.

Hints

- When asked by Angular's interactive prompt, answer 'yes' to the question 'Would you like to add Angular routing?' and 'SCSS' as the stylesheet format used. This will make things easier later during development.
- Note: It is socially accepted that following completion of a task, a pull request is raised and completed from the feature branch to develop. Tasks (on the board) should be moved from 'active' to 'in review' then finally to 'closed' and the time spent of the task is recorded correctly. This helps Resonate plan for future work.

TASK 40003

40003: Run Luminate's Client Locally

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Build and run the Luminate Client application locally on you PC.

Hints

[9] Angular Cli - a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.

[10] Git - distributed version control system designed to handle everything from small to very large projects with speed and efficiency

[11] Azure DevOps (Luminate) – Microsoft web application aimed to help Plan smarter, collaborate better, and ship faster.

- A CI's IP address can be found on teams under 'Luminate Develop Public – Environment Reports.' Further information about how to set up a client can be found at the confluence page 'Setting up Luminate Client' and usernames and passwords can be found by asking your mentor.

Mentors Notes

I am expecting some teething problems with regards to how their development environment has been set up. The above tasks provide us with an opportunity to check that everything is working correctly on their machines and that they have everything they need to complete the project. Some known issues / solutions may include:

- They are unable to create a new repository within Luminates repository. This is most likely because they do not have permission to do so. They can contact IT asking for permission to create their online repository or have someone with permission to do it for them. Note, should they get permission make sure this is turned off again following the principle of least privilege.
- Angular client does not ask any questions when creating a new project. This is likely because interactive mode is false by default. Typing 'ng new <whatever> --interactive' should solve the issue but if it persists, you can add the flags '—routing' and '—style scss' to get the same result. Please see https://angular.io/cli/new
- Issues with credentials when pushing. This shouldn't be an issue. Debug the issue with the mentee (unfortunately I can't see this issue at the minute as my environment is set up correctly. I have heard of others having a problem though.
- When running the Luminate client for the first time there may be an issue where NPM pauses for a long amount of time on rxjs. This can be solved by downgrading NPM to an earlier version.

User Story 30002

30002: Board Picker

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to alter the service information displayed based on a chosen board.

Acceptance Criteria

- Add board picker following the design outlined in the provided wireframe
- When the board picker is clicked a drop-down menu should appear populated with the various boards available within the application. This includes Arrivals, Departures, and Arrivals and Departures.
- If the user selects a different board, the drop-down menu is no longer displayed, that board's name is displayed within the board picker, and the URL changes to an appropriate address.
- Once a new board is selected, table below updates to show relevant information.
- When the drop down is opened, and the user clicks anywhere else on the application (i.e., does not select a new board) the drop-down menu is not displayed.
- If the user changes the URL address manually, the board picker and table is updated / remains correct.

TASK 40004

40004: [Expository] Review Basic Organisation of the Luminate Client

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Open the Luminate client in your IDE and review how different parts of application are filed away. Make special reference to the base folder of the components typically displayed to the user.
- Look at what happens to the URL in the browser when you change pages. This is the router outlet at play. Find this in Luminates code and think about sensible URLs for the 3 main board in your application.

Hint

- Run the Luminate client locally, connect it to an appropriate CI environment and log in. Use the developer tools within the browser to view the source of a particular component displayed to you and find these components within Luminate's code.
- The word 'routing' is a big giveaway for finding out how the router-outlet works.

TASK 40005

40005: Create and style departures, arrivals, and departures-and-arrivals components

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As per the design specification, one of 3 tables is going to be displayed to the user, given their choice. Create and style a component for each of these.
- [Optional] To aid with styling feel free to hard code a table into the HTML using dummy data.

Hint

- Keep your project organised. Make sure you create these UI components in an appropriate location within your project. The more organised you are, the easier future work will be. Take inspiration from the Luminate codebase.

TASK 40006

40006: Create board-picker component

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create a board picker component to allow the user to pick which board should be displayed. This should contain 3 options: Arrivals, Departures, and Arrivals and Departures

Hint

- As there is a finite amount of board options, an enumeration[12] type may be useful for later when adding logic to the application.

TASK 40007

40007: Add routing to boards and wire in picker

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Add routing to the application. The table displayed should change depending on the users choice. Should no choice have been made, '/departures-and-arrivals' may be set.
- The user also has the choice (although slightly more unlikely) to change the URL in the bowser manually. The picker should therefore default to whatever page the user is on.

Hint

[12] Enum - allow a developer to define a set of named constants. Using enums can make it easier to document intent, or create a set of distinct cases.

- There are many ways to find out how to do the above. Looking at the Luminate code is a good starting point, but also further information can be found using the Tour of Heroes walkthrough[13]. See navigation.

Mentors Notes

This may be the first major part of functionality a mentee has added to a program. Some known issues / solutions:

- My IDE never recognises the HTML tag router-outlet. The program should still work fine regardless.
- The JavaScript string method replace only replaces the first character it finds; therefore, you have to give it a regex expression (i.e. / /g for replacing all spaces within the string.) Super silly but confused me for half an hour.

User Story 30003

30003: Station Picker

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to alter the service information displayed based on a chosen location.

Acceptance Criteria

- Add location picker and table following the design outlined in the provided wireframe
- When the location picker is clicked a drop-down menu should appear populated with human readable names of the available stations within the application.
- If the user selects a new location, the drop-down menu is no longer displayed, and that location is displayed within the location picker.
- Once a new location is selected, data populating the table below updates to information relating to the chosen location.
- When the drop down is opened, and the user clicks anywhere else on the application (i.e., does not select a new location) the drop-down menu is not displayed.
- If the user refreshes the page, their previous chosen location is retained.

TASK 40008

40008: [Exploratory] Huxley2 API endpoints and returned objects

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Take some time to read the documentation on the Huxley2[14] API and Darwin[15].
- Using postman[16] hit a few of the endpoints and review what is returned.
- Review wireframe and think about what API calls and what returned data is needed

Hint

- The more you understand the data that is being returned, the easier the next step is going to be. Spend a little time with the JSON[17], understand what the properties mean and try to imagine how the objects are constructed.

TASK 40009

40009: Mock API Responses from Huxley2 and design TrainService and CRS model

[13] Tour of Heroes – Angular tutorial where you build your own Angular application from start to finish.
[14] Huxley2 - a CORS enabled cross-platform JSON ReST proxy for the GB NRE LDB WCF SOAP XML API (called Darwin)
[15] Darwin - the GB rail industry's official train running information engine, providing real-time arrival and departure predictions, platform numbers, delay estimates, schedule changes and cancellations.
[16] Postman - an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.
[17] JSON - is a standard text-based format for representing structured data based on JavaScript object syntax.

Unassigned

State	New	Area	Illuminati\Luminate
Reason	New	Iteration	illuminati

Description

- Using postman, call the needed endpoints and copy the responses into a mocked service. The service's method is likely to return a different instantiation of the interface depending on a parameter.
- From the responses design the model interfaces that are needed. This is centred around two main objects. TrainService and CRS.

Hint

- If you copy in the responses first, then start building your object definitions, you can use the IDE to your benefit. Slowly work through each of the objects and get rid of the red lines.
- Note: There may be some primitive types you are unsure of as not data fills them in your mocked responses. Make your best guess, and if there is an issue, this can be solved later when the app is connected to the internet.

Mentors Notes

Deserializing a complex object is often quite difficult especially if the dataset provided by their mocks do not give them enough information about the primitive data types used in the interface's definition. Making mistakes at this stage doesn't matter and all that is needed is the interface definition to work with their specific mocks. These definitions are likely to be re-written when connecting the application to the web, and hopefully with a larger data-set (every service in the UK) will solve any queries. Undefined and null values may also cause issues. Fix this any way you can, there is a solution again when connecting to the web.

TASK 40010

40010: Create Station Picker component

Unassigned

State	New	Area	Illuminati\Luminate
Reason	New	Iteration	illuminati

Description

- Create and style station picker component. The options should be populated with the returned CRS objects from the mocked service.
- The selected CRS should be stored somewhere to make it persist over a bowser refresh.

Hint

- You may find some knowledge of RXJS useful here. Watch some videos and learn some theory.
- You can find similar lines to this within Luminate. Open a Luminate client and click on the admin menu cog. The text size, theme, and extend timeframe share data between components using RXJS and save data for persistence. See if you can follow these lines.

TASK 40011

40011: Retrieve and display data in Arrivals, Departures, and Departures and Arrivals

Unassigned

State	New	Area	Illuminati\Luminate
Reason	New	Iteration	illuminati

Description

- Now that the correct CRS is stored, we can retrieve the correct stations data from our mock service.
- Create and style a table to display the data returned by the service.

Hint

- You will need to use Angular's For directive[18] to achieve this.
- You may notice yourself copying a substantial amount of code between the 3 table components. This may lead you to the conclusion that you need a base-table component that the 3 more specific inherit from. This should stop code duplication.
- A similar comparison can be made between the two pickers (board-picker and station-picker.)

Mentors Notes

RXJS is super confusing when encountered for the first time. This is made more difficult when looking into how it is used within Luminate. My issues when learning it largely revolved around passing a call-back function as a parameter. Both research into RXJS and call-back functions are needed.

Feature 2002

20002: More service information

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Problem Statement

When displaying a board within the train board viewer application the user may want to find some more detailed service information about a particular service. This may include subsequent / previous calling points, the services formation, reasons for delay or cancelations, who is the operator etc.

Benefit Hypothesis

- To be able to find more detailed information about a particular service.

Approach

- Allow the user to see some additional information about a service on hover. This can include its previous / future calling points, and its formation (i.e. how many carriages form a service.) This should be based on mocked Huxley2 API responses.
- Allow the user to see detailed information modal about a service when double clicked. This can include reasons for delay or cancelations, who is the operator etc.

Time Sheet Code

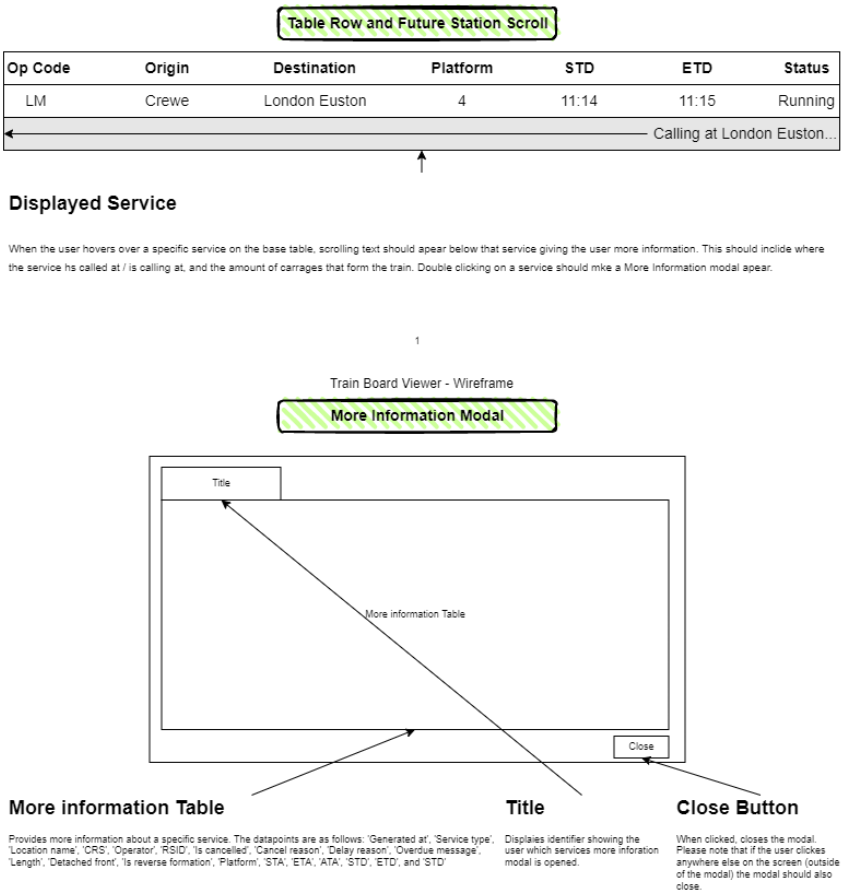
- LD3847583 – New Starters (Training Project)

Risks

Technical Design

- Below provides a wireframe mock of the future station scroll and the more service information modal:

[18] For directive - A structural directive that renders a template for each item in a collection.



User Story 30004

30004: Future Station Scroll

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to quickly view a service's calling points and formation

Acceptance Criteria

- When the user hovers over a particular service displayed in the table, additional information should appear below relating to the services formation and calling points.
- The information should move across the page from right to left.
- When the user moves their mouse off the service, the additional information scroll should dis-appear.

TASK 40012

40012: Create and style Future Station Scroll Component

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create and style the future station scroll component and show when the user hovers over a service.

Hint

- You will need to start sharing data between parent and child components most likely by making use of Angular's Input decorator[19].
- With regards to having a component shown / not be shown Angular's If directive[20] may be of use.

User Story 30005

30005: More Service Information Modal

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to see detailed information relating to a particular service.

Acceptance Criteria

- When the user double clicks on a particular service, a more information modal should appear containing the following information: 'Generated at', 'Service type', 'Location name', 'CRS', 'Operator', 'RSID', 'Is cancelled', 'Cancel reason', 'Delay reason', 'Overdue message', 'Length', 'Detached front', 'Is reverse formation', 'Platform', 'STA', 'ETA', 'ATA', 'STD', 'ETD', and 'STD'.
- When the modal is active, the background should be darkened.
- If the user clicks the modals close button or anywhere else (outside of the modal) the modal closes, and the user is returned to the main page.

TASK 40013

40013: Create and style More Service Information modal

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create and style the more service information modal component and populate with data retrieved from mock service.

Hint

- Modals typically consist of two parts; the modal itself and a div that sits on top of the background greying it out and awaiting a click event to close.

Feature 2003

20003: Search

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Problem Statement

When displaying a board within the train board viewer application the user may want to filter the returned services based on a provided search term. This may aid them in planning a journey.

Benefit Hypothesis

- To be able to find all trains relating to two different locations. For example, find me all trains departing from London Euston that will arrive at some point in Derby.

[19] Input Decorator - give a child component a way to communicate with its parent component.

[20] If Directive - A structural directive that conditionally includes a template based on the value of an expression coerced to Boolean.

Approach

- Allow the user to input a search term, and filter the services displayed on the board.

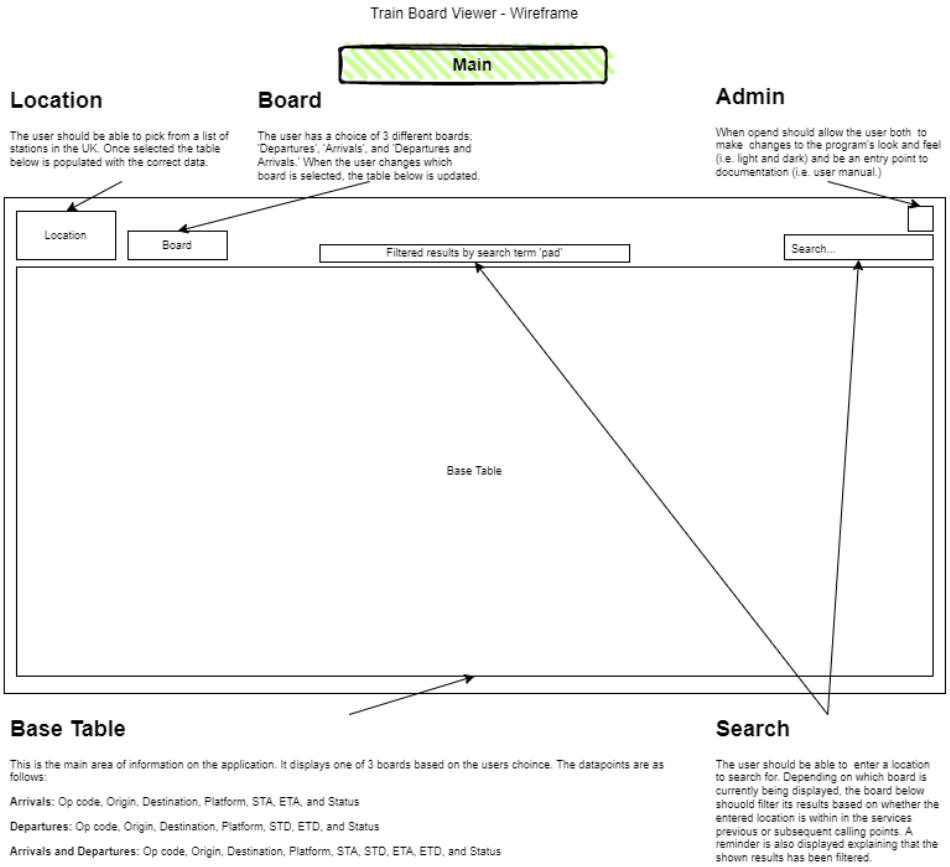
Time Sheet Code

- LD3847583 – New Starters (Training Project)

Risks

Technical Design

- Below provides a wireframe mock of the main application:



User Story 30006

30006: Add Search Bar to Filter Table

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to filter the displayed services based on a search term.

Acceptance Criteria

- Search bar added following the design outlined in the provided wireframe
- When the user starts typing a location into the search bar, the table below should dynamically filter its services based on that term.
- The comparison should be different depending on which table is being displayed. If arrivals the comparison should be between the search term and each services previous calling points. If departures, the comparison should be between the search term and each services subsequent calling points. And if departures and arrivals, the comparison should be between the search term and both the subsequent and previous calling points.
- A reminder should be displayed showing that the tables services has been filtered.
- The reminder should disappear should the table not currently be filtered.

TASK 40014

40014: Create and style Search component

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create and style search component as per the wireframe provided.
- Make reminder appear in the centre consisting of a phrase and the search term (sharing data will need to be used here.)

TASK 40015

40015: Return filtered services in table.

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Make call to service to find out previous / subsequent calling points of services for comparison.
- Filter the displayed list depending on search term.

Mentors Notes

This is a little confusing and may take a few attempts to get right. The issue is the information needed for the comparison is not included within the objects on the table. They will therefore need to get more information about each service before making the comparison, however the filtered objects in the list remain of the original type. Some knowledge of the order in which the lines are run may be helpful.

Feature 2004

20004: Admin menu

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Problem Statement

Both the look and feel of the application may not appeal to the user, and it may be difficult to understand how to use the application.

Benefit Hypothesis

- To be able to provide the user with more information about how to use the train board viewer application.
- To allow the user some choice in how the application looks and feels.

Approach

- Add an admin menu cog that both:
    - allows the user to change the look and feel of the application (i.e., light, and dark mode.)
    - links them to additional information about the web app (i.e., user manual.)

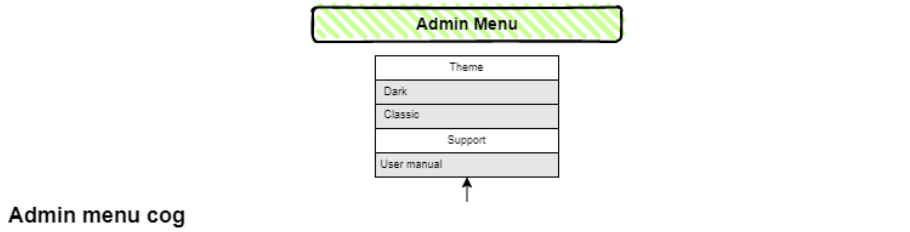
Time Sheet Code

- LD3847583 – New Starters (Training Project)

Risks

Technical Design

- Below provides a wireframe mock of the admin menu:
-



-

User Story 30007

30007: Admin menu cog

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to have access to look and feel options using an admin menu cog.

Acceptance Criteria

- An admin menu cog added following the design provided in the wireframe mock.
- If the user clicks on the cog, a drop down with options should appear.
- If the user clicks an option or anywhere outside of the cogs drop down, the dropdown should disappear.

TASK 40016

40016: Create and style Admin Menu component

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Create and style admin menu component as per the wireframe mock provided. You may wish to look at how Luminate achieves this.

User Story 30008

30008: Light and Dark Mode

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to change the look and feel of the application through light and dark mode.

Acceptance Criteria

- The admin menu drop down should have the option to change the look and feel of the application through light and dark mode.
- When the user clicks on a mode, the look of the application should dynamically change.

TASK 40017

40017: [Exploratory] Luminate's light and dark mode vs internets

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Take some time googling how others have achieved light and dark mode in an Angular application. You may need to study SCSS a little as it may require the use of functions, mixins, variables etc. It may also be worth experimenting with these new concepts a little.
- Look at how Luminate has accomplished the same thing. Are there any differences? Why has this been done?
- What are the important lines through the application that you are going to need?

TASK 40018

40018: Refactor components style sheets

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Look through the style sheets of each of your components and see if you can make better use of SCSS. One way may be to have global mixings, and import them to style each component.
- As colour is the main thing that will be changing depending on the users theme choice, it may be best to start storing these are variables, and using the variables in the mixins. This will allow you to change the colour by changing the variable used.

TASK 40019

40019: Add theme button to admin menu and store current theme

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Similar to your work building a station picker, there is a requirement to have the users theme persist over a refresh. You can use the same model to store this information.

TASK 40020

40020: Add class in app component to apply theme

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Change CSS class in app component on next, when subscribed to theme.

Mentors Notes

The light / dark mode feature is one of the most fun in the application. You suddenly start to understand the power of SCSS and see almost all CSS in the components disappear. Super useful video I found when writing this is Creating a Dark & Light Toggle Mode on your UI Designs and useful article is [add]. With regards to learning about SCSS (or any CSS topic) this guy is pretty good.

User Story 30009

30009: User manual

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to be able to view a user manual of the application in a new browser tab.

Acceptance Criteria

- The admin menu drop down should have the option to view a user manual of the application in a new tab.
- When the user clicks on the user manual, a new tab opens displaying a pdf of the user manual.

TASK 40021

40021: Add user manual link to admin menu

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Find the user manual in Luminate and add any document in a sensible location to your project.
- Add link to admin menu to open this document in a new tab.

Feature 2005

20005: Return live responses

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

Problem Statement

Currently the application returns mocked responses from the Huxley2 API. This means that the information provided to the user is most likely incorrect.

Benefit Hypothesis

- To allow to user to see live information about train services throughout the United Kingdom.

Approach

  - Send API GET requests to Huxley2 API endpoints and use to responses as the data within the train board viewer application.

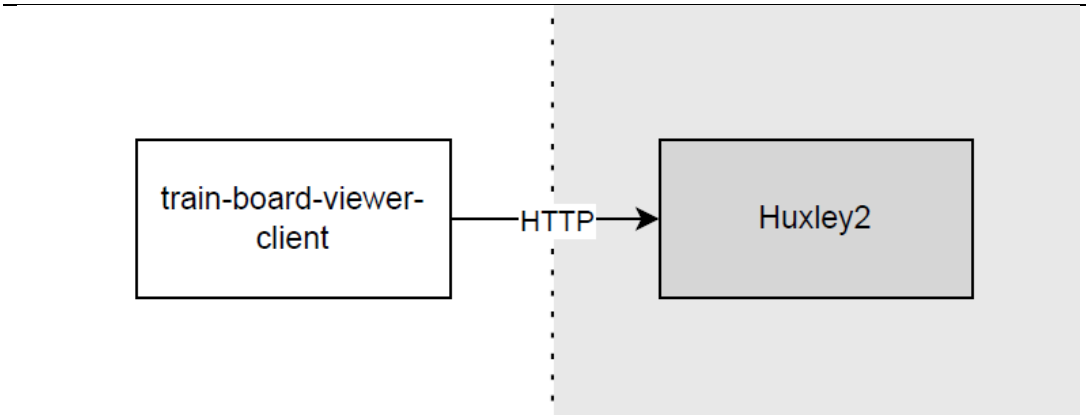
Time Sheet Code

- LD3847583 – New Starters (Training Project)

Risks

Technical Design

- Below provides a diagram exspaining the overall archetecture of the application.



User Story 30010

30010: Connect Train Board Viewer to the Web

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- As a user I want to receive live information about services within the UK

Acceptance Criteria

- The information displayed to the user is both Live and correct.

TASK 40022

40022: [Exploratory] Luminate client to Live Scenario Gateway

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Open a Luminate client in your bowser and start to explore the connection between it and the Live scenario gateway. Make some network recordings using the developer tools, and see what endpoints are being hit and what data is being returned.
- Open the Luminate client in your IDE and find the files responsible for communication to the gateway. You may notice that the file names state that they have been generated.
- How are the IP addresses listed in the eviroments.ts file pulled into the classes responsible for communication?

TASK 40023

40023: Connect to CRS, Departures and Arrivals, and Services API endpoints

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Unfortunately, as the Huxley2 API does not come with a specification there is no way to auto generate the API files in your application. Therefore, use the generated API files of Luminate as a model, and write appropriate files for CRS, Departures and Arrivals, and Services.
- Make sure your base URL is stored in the environments.ts file, and inject where needed.

Mentors Notes

Note making these API files by hand is super tedious. I would ask to do this for CRS (as it's the smallest object) then give the others from my example work.

TASK 40024

40024: Connect to Service to API

Unassigned

State	New	**Area**	Illuminati\Luminate
Reason	New	**Iteration**	illuminati

Description

- Connect the service that is currently mocking the responses to the newly created API files.
- Remove redundant code.

Next Steps

Although this application does have most of the things one would expect from a front end, there are some very large wholes of knowledge. Shallow integration testing is the greatest, and this has only been excluded as Luminate, due to its size, has very little tests. The project should really be tested.

Other small feature ideas one may add are:

- Ordering to the tables
- Moveable and resizable modals
- Polling Huxley2 and different colours based on the service being at the platform (or something like that.)

As we have started to investigate how a user may use the application to plan a journey (with the search component), the major next step in my view is adding our own back end to this application that will allow us to return results should the user have to make a change or several changes to different services. We can then revisit the front end to work out how to best display our generated results. This will get us into generating the API files, give us the opportunity to use real CIFs returned from Darwin, and implement both SOAP and REST APIs in Java / Spring. Spinning up the application on AWS would give a good general overview into how an application may be built.