CS7CS3 Advanced Software Engineering Group Project

Functional and Technical Architecture

Project Name: Sustainable City Management

Group 5

Group Members:

Yatheendra Pravan K M 20305901	Lokesh Selvakumar 21331969	Karina Salem 20323363	Kevin Reynolds 20308229
Kalaivani Kandasamy 21332195	Deeksha Vyas 21334447	Ayush Kalra 20327429	Shubham Maurya 20310455

Contents

1. Functional Architecture

1.1. Diagram

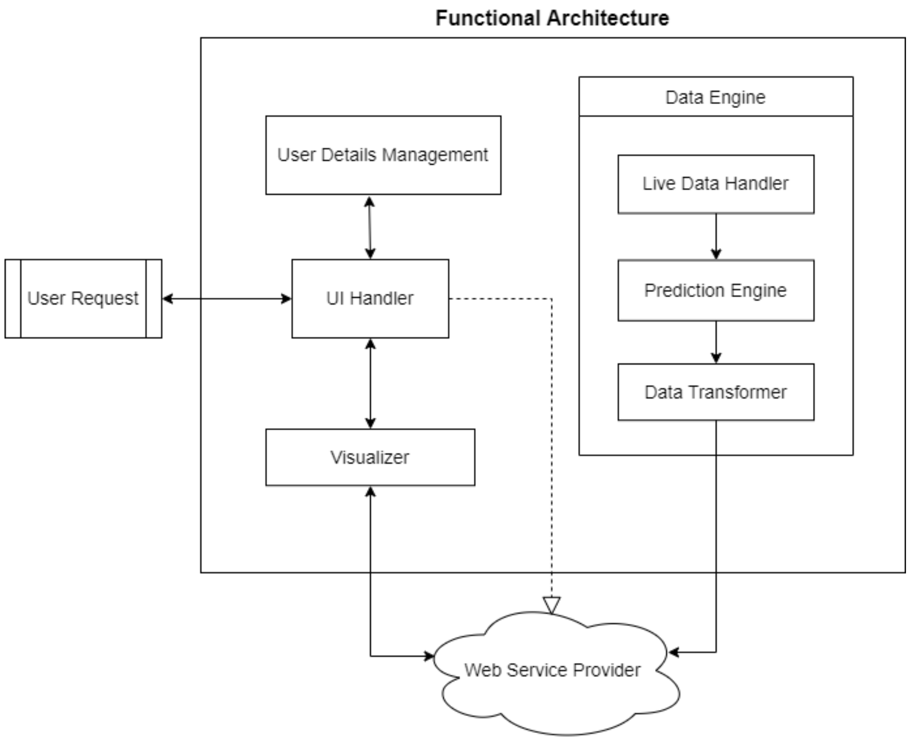

Figure 1: Functional Architecture Diagram

1.2. Component Descriptions

User Details Management
This component will be responsible for user login, as well as managing the roles for the different users' types using a *django* server.

User Interface (UI) Handler
This component handles the user interactions with the interface. It will also be responsible for the display and the user experience (UX). Moreover, it subscribes to the Firebase server and listens for data changes.

Data Engine
Every five minutes, the engine will fetch live data and perform the functionality specified in the following three sub-components:

1. **Live Data Handler**: it will dynamically select the API end-points, fetch live data from those sources, and pass it to the Prediction Engine.
2. **Prediction Engine**: It makes predictions from the live data and passes the predicted data to the data transformer.
3. **Data Transformer**: It receives data from the Predication Engine and performs the following:
   a. Converts the received raw data to a data format compatible with the visualization components such as heat maps, bar charts, etc.
   b. Estimations for:

1. The Luas energy usage.
2. The bicycle station usage, station capacity report and station swap suggestions.
3. Bus locations, bus emissions alternative route suggestions based on CO2 emissions and congestion factors.
4. Event crowd sizes.

c. The Data Transformer reads the aggregated and historical data from the Web Service Provider and then re-writes the newly computed data into it.

Then, the **Data Transformer** sends the aggregated and transformed data to the **Web Service Provider**.

Web Service Provider
A Firebase real-time database is used to implement a publisher-subscriber model; whenever it receives new data from the **Data Engine**, it will pass it to every subscribed device.

Visualizer
It is responsible for displaying a different set of visualizations for each data indicator in our system - Bikes, Buses, Luas, and Events/Incidents. It will retrieve the relevant data from **Firebase** based on the system time.

1.3. Component interactions

1.3.1 Data Pipeline

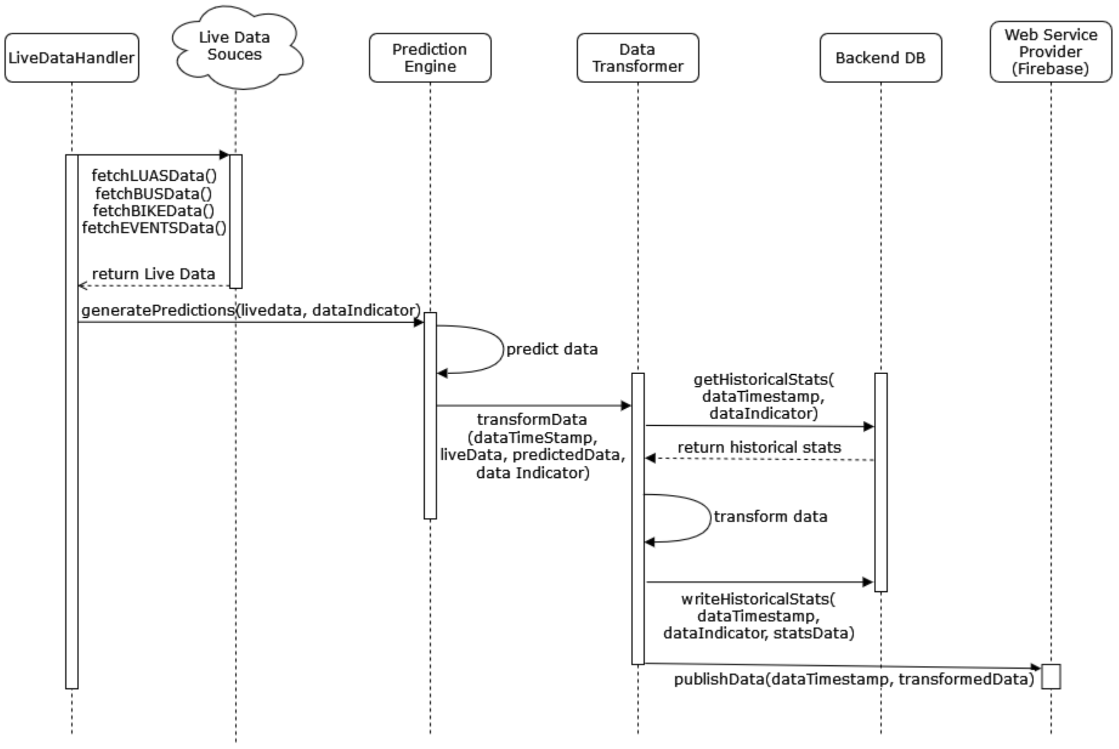
LiveDataHandler
Live Data Souces
Prediction Engine
Data Transformer
Backend DB
Web Service Provider (Firebase)
fetchLUASData()
fetchBUSData()
fetchBIKEData()
fetchEVENTSData()
return Live Data
generatePredictions(livedata, dataIndicator)
predict data
transformData (dataTimeStamp, liveData, predictedData, data Indicator)
getHistoricalStats(dataTimestamp, dataIndicator)
return historical stats
transform data
writeHistoricalStats(dataTimestamp, dataIndicator, statsData)
publishData(dataTimestamp, transformedData)

Figure 2: Data Pipeline Sequence Diagram

1.3.2 Login and Firebase subscription

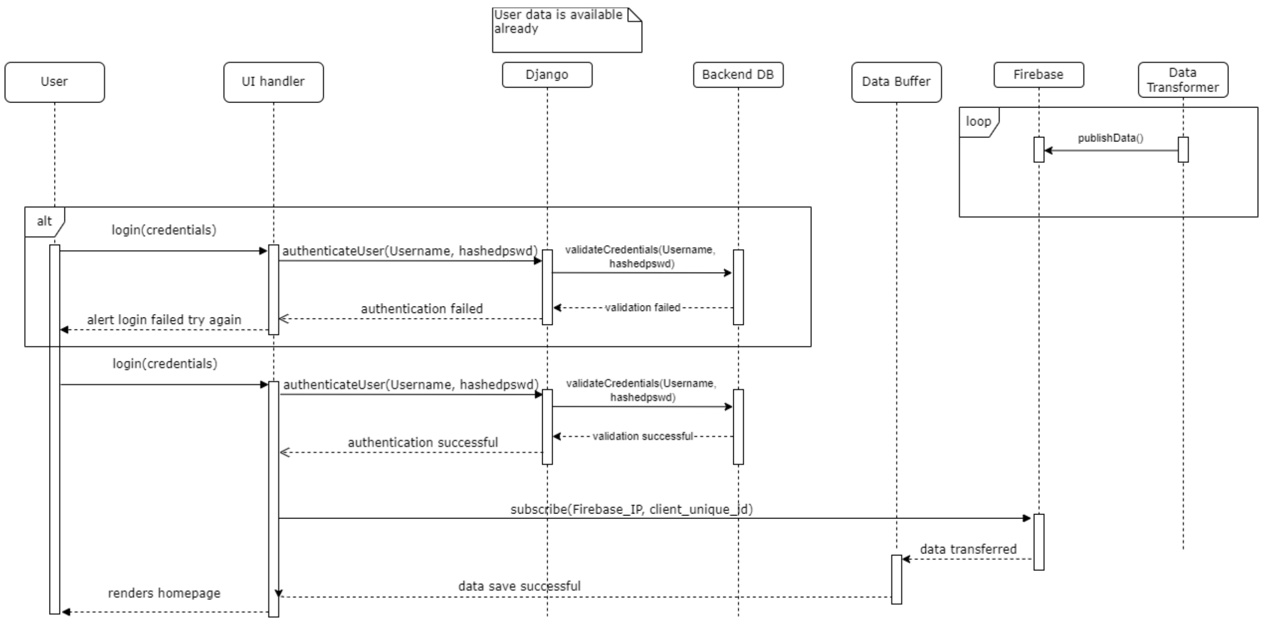
User data is available already
User
UI handler
Django
Backend DB
Data Buffer
Firebase
Data Transformer
loop
publishData()
alt
login(credentials)
authenticateUser(Username, hashedpswd)
validateCredentials(Username, hashedpswd)
validation failed
authentication failed
alert login failed try again
login(credentials)
authenticateUser(Username, hashedpswd)
validateCredentials(Username, hashedpswd)
validation successful
authentication successful
subscribe(Firebase_IP, client_unique_id)
data transferred
data save successful
renders homepage

Figure 3: Login and Firebase subscription Sequence Diagram

1.3.3 Dublin Bikes - Station Visualization

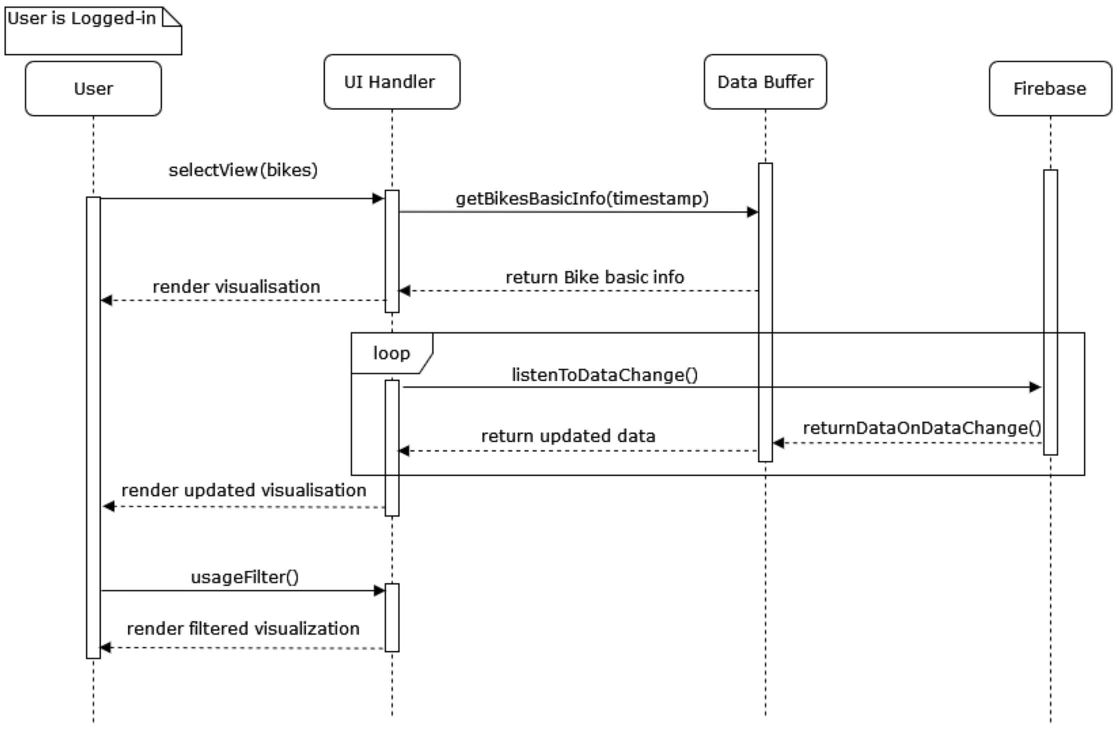

Figure 4: Dublin Bikes - Station Visualization Sequence Diagram

1.3.4 Dublin Bikes - Station Usage

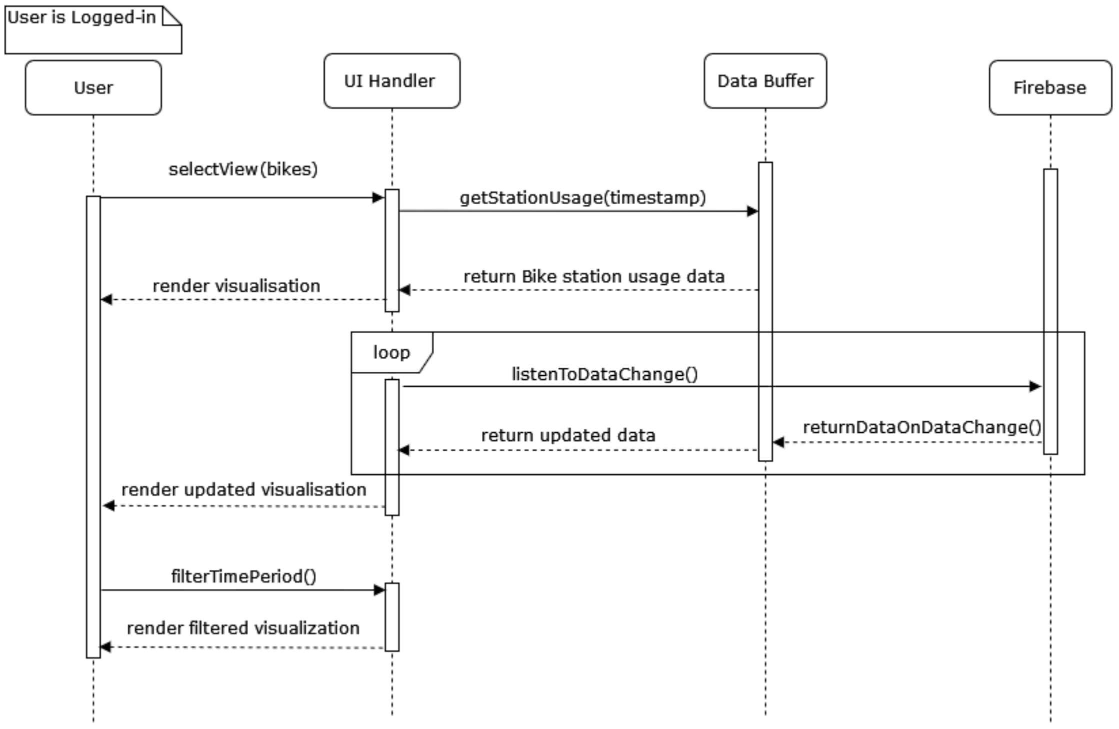

Figure 5: Dublin Bikes - Station Usage Sequence Diagram

1.3.5 Dublin Bikes – Heat-map

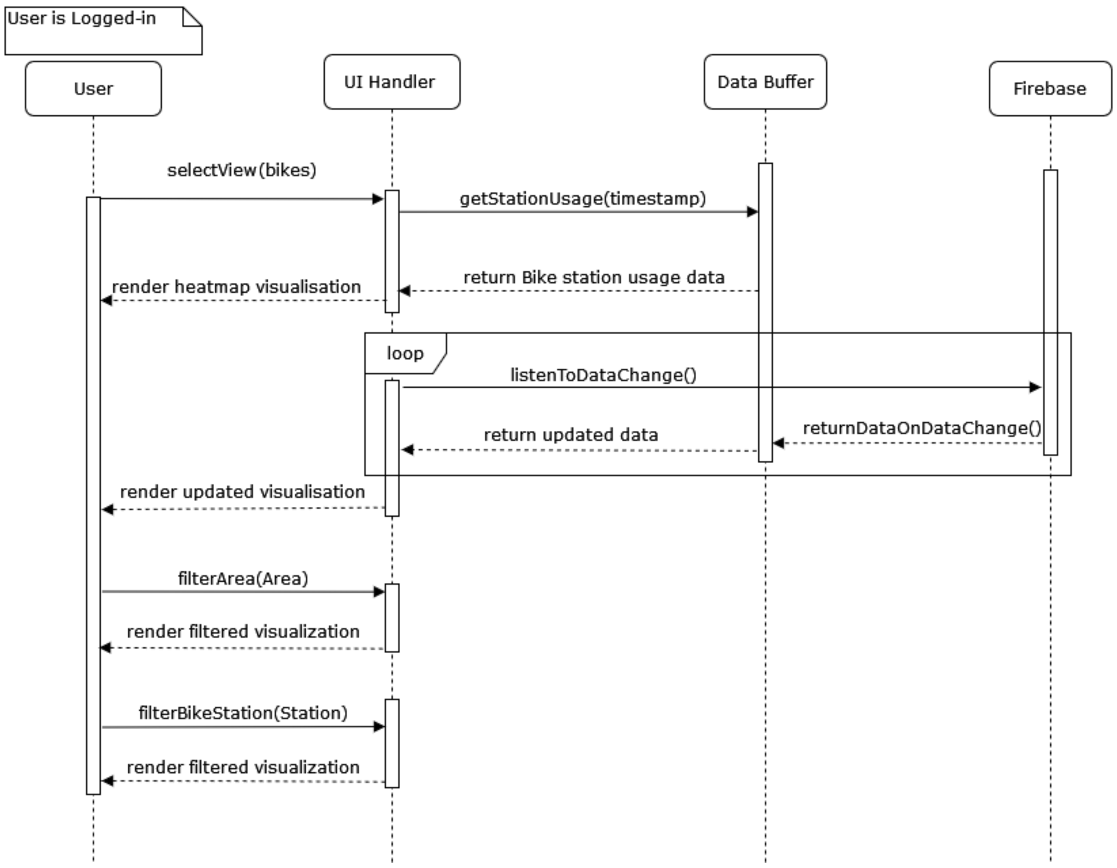
User is Logged-in
User
UI Handler
Data Buffer
Firebase
selectView(bikes)
getStationUsage(timestamp)
return Bike station usage data
render heatmap visualisation
loop
listenToDataChange()
returnDataOnDataChange()
return updated data
render updated visualisation
filterArea(Area)
render filtered visualization
filterBikeStation(Station)
render filtered visualization

Figure 6: Dublin Bikes – Heat-map Sequence Diagram

1.3.6 Dublin Bikes – Swap Suggestions

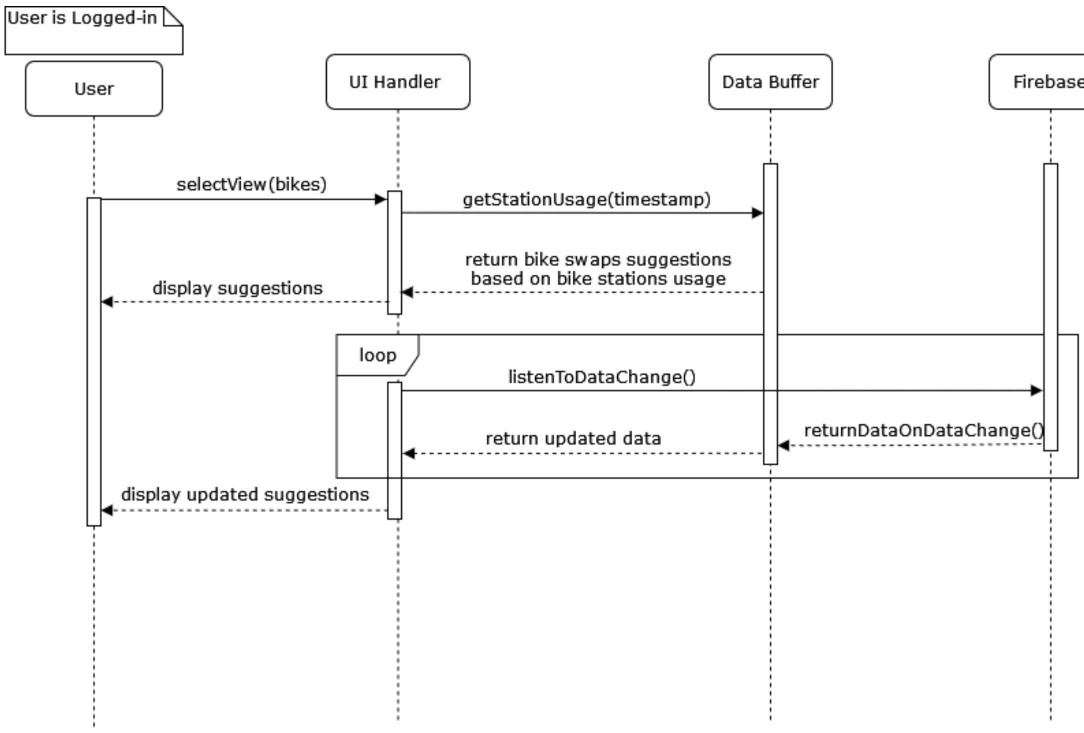

Figure 7: Dublin Bikes – Swap Suggestions Sequence Diagram

1.3.7 Buses - Map & Heat-map of Bus/Bus Stop Locations

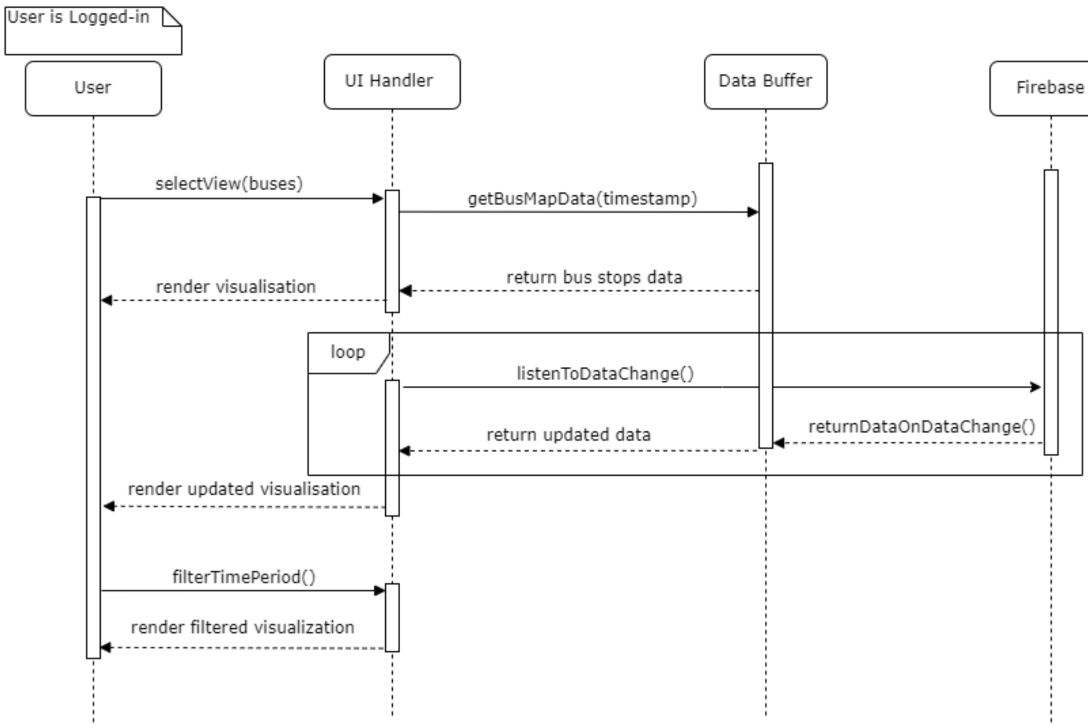

Figure 8: Buses - Map & Heat-map of Bus/Bus Stop Locations Sequence Diagram

1.3.8 Buses - Environmental Impact

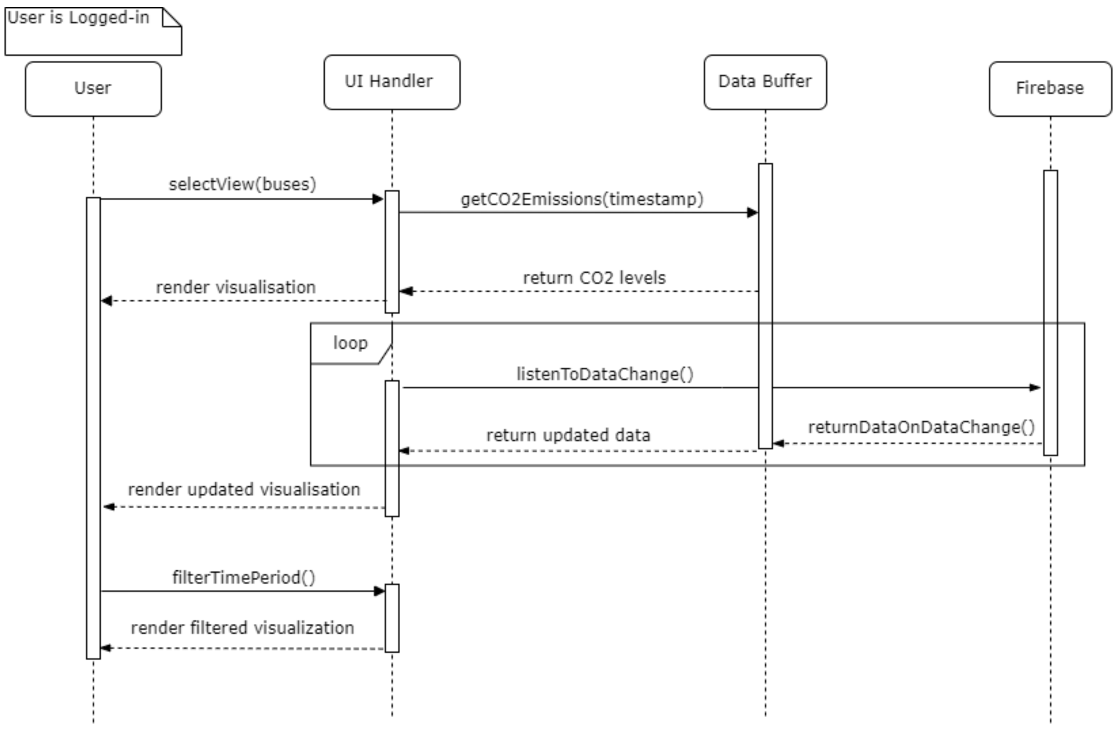

Figure 9: Buses - Environmental Impact Sequence Diagram

1.3.9 Buses - Predict CO_2 emissions

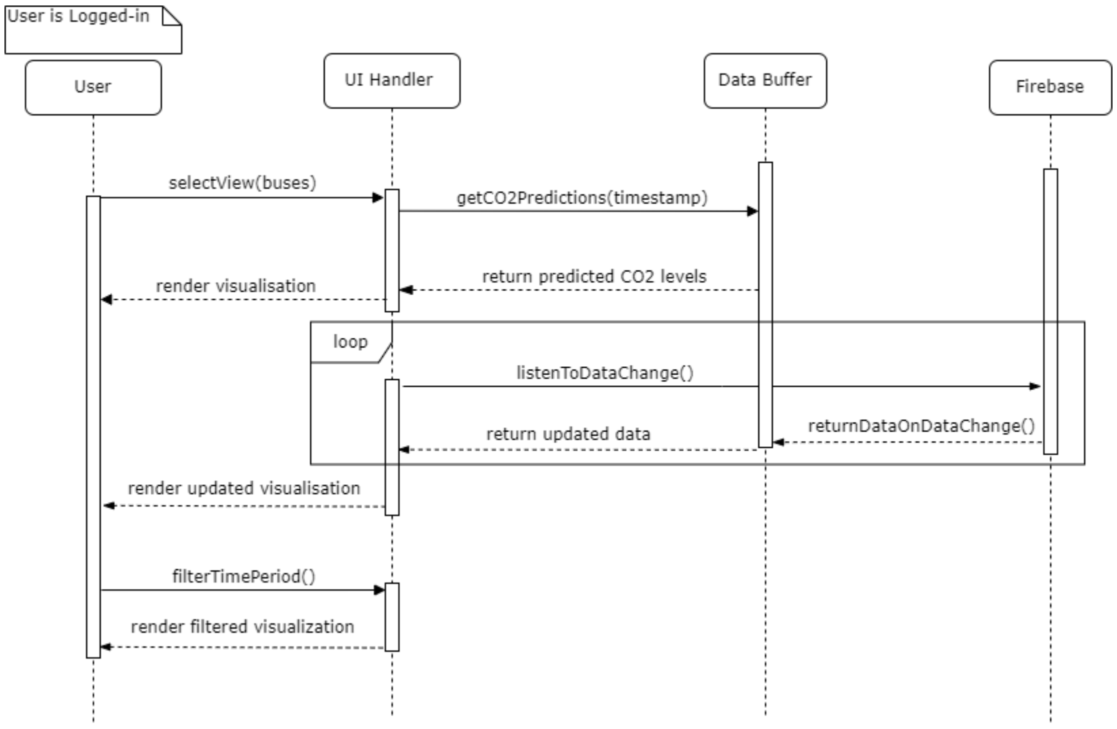

Figure 10: Buses - Predict CO2 emissions Sequence Diagram

1.3.10 Buses - Rerouting Suggestions

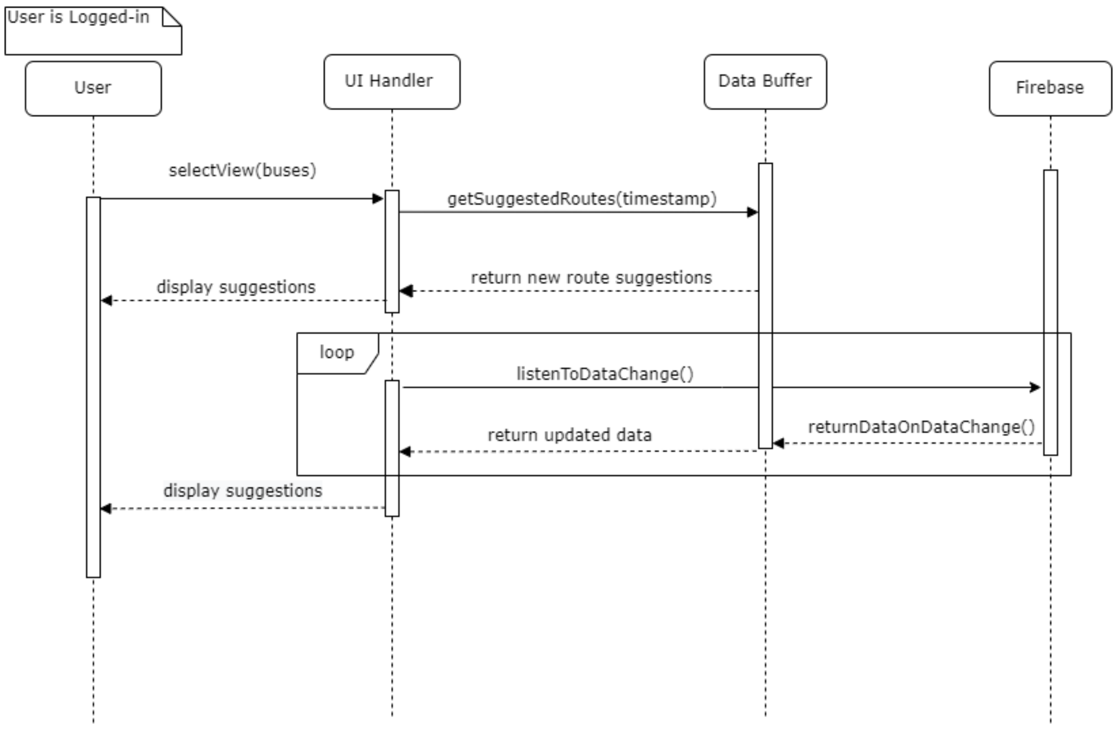

Figure 11: Buses - Rerouting Suggestions Sequence Diagram

1.3.11 Luas - Map Visualization

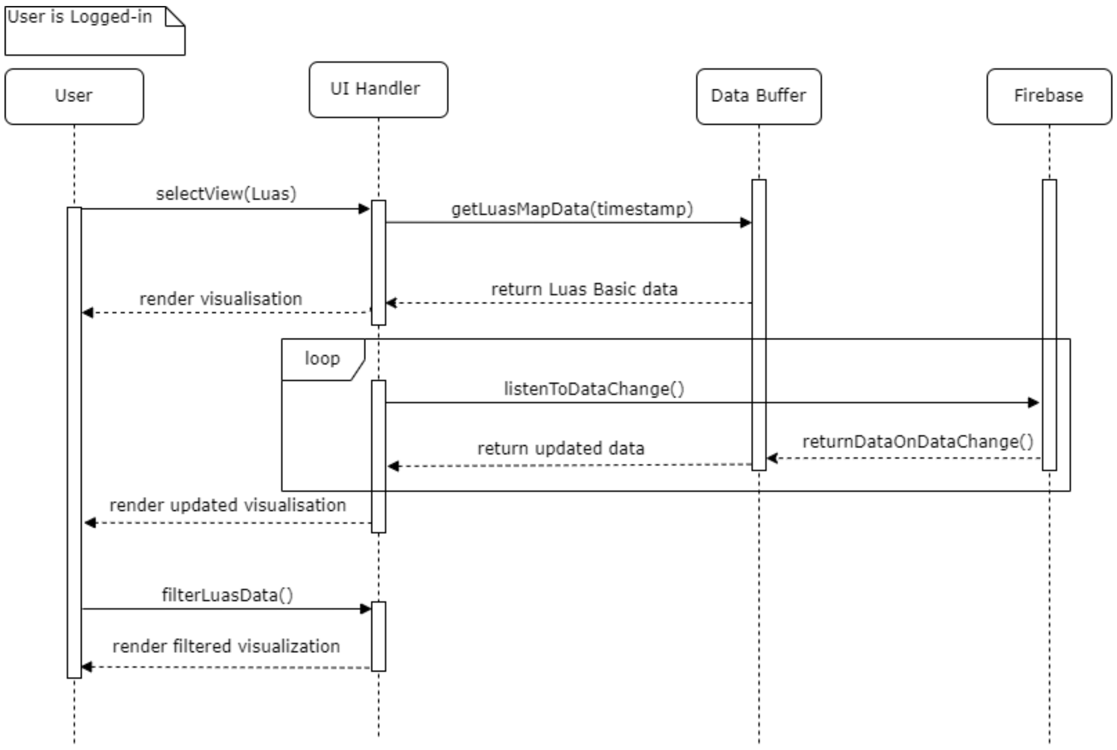

Figure 12: Luas - Map Visualization Sequence Diagram

1.3.12 Luas - Electricity Usage Estimates

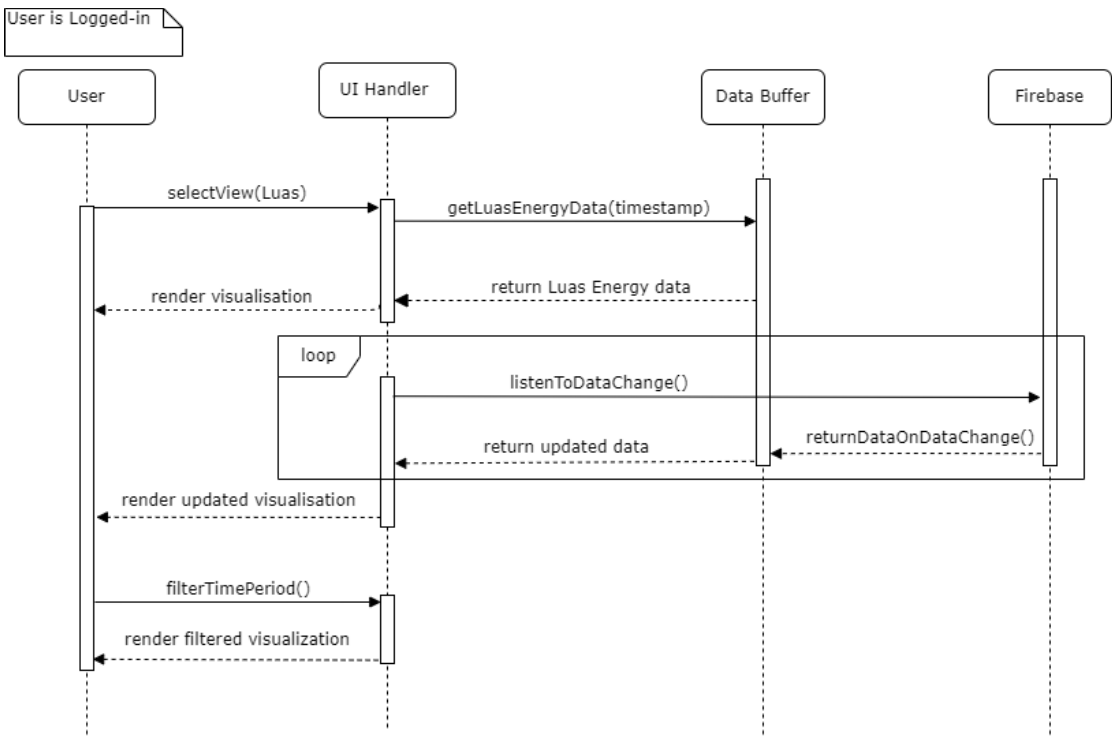

Figure 13: Luas - Electricity Usage Estimates Sequence Diagram

1.3.13 Event Locations and Estimated Crowd Size

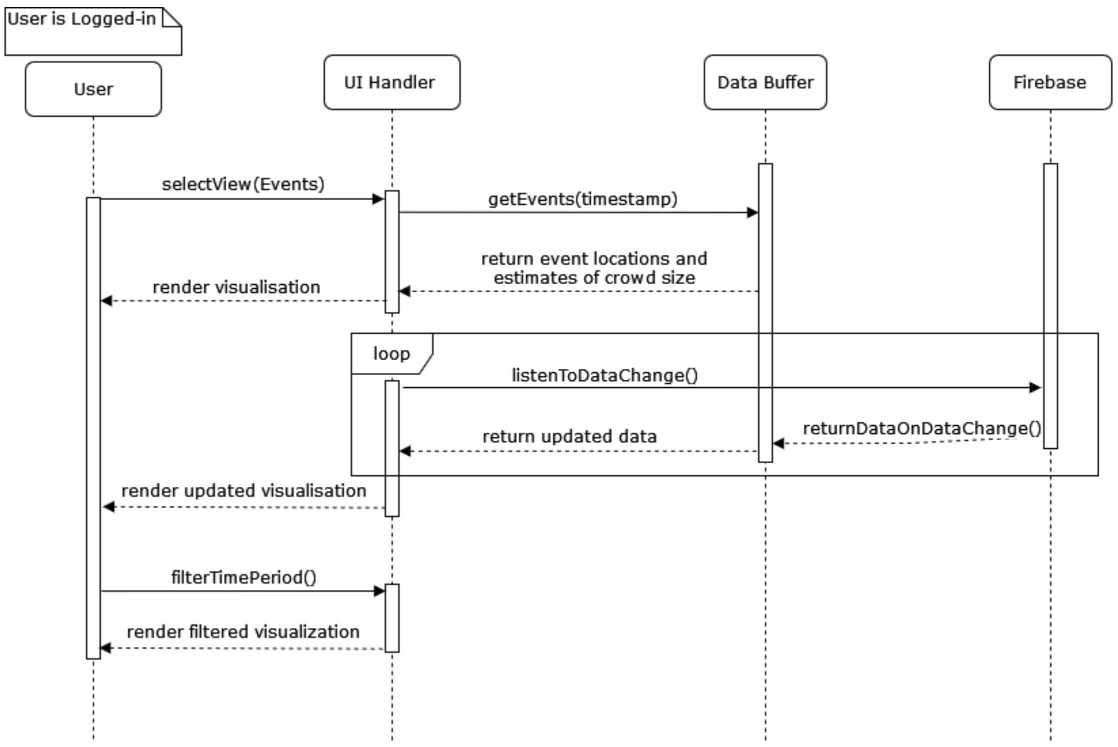

Figure 14: Event Locations and Estimated Crowd Size Sequence Diagram

1.3.14 Bus Route Frequency Suggestions

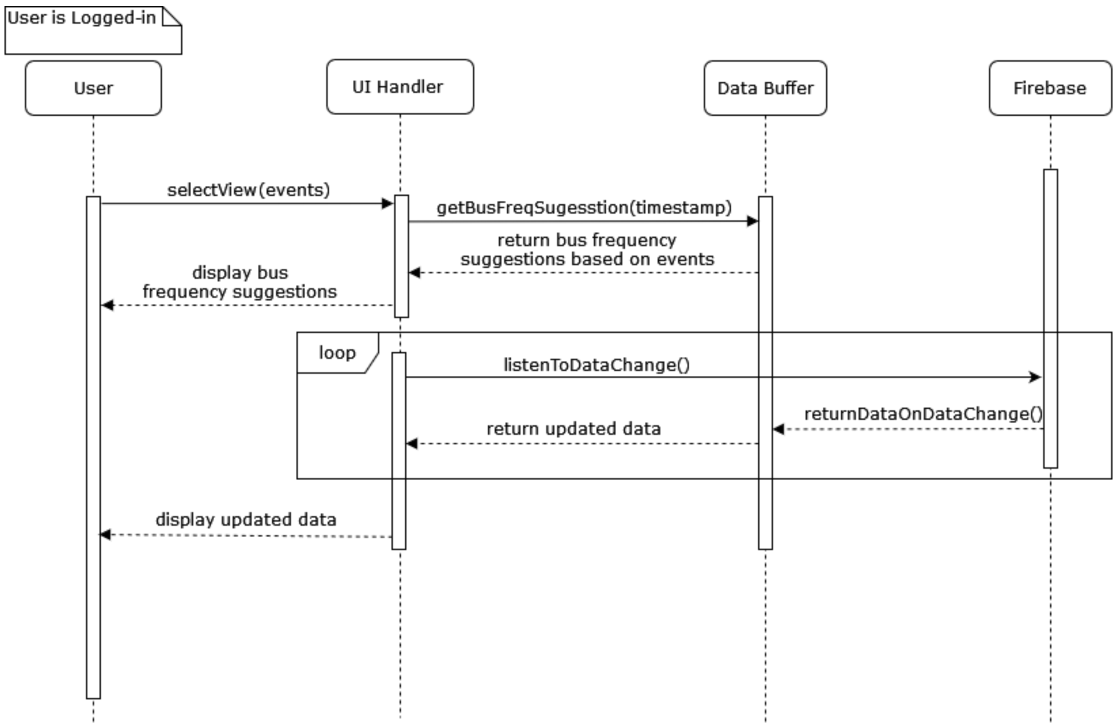

Figure 15: Bus Route Frequency Suggestions Sequence Diagram

1.3.15 Maintenance Events

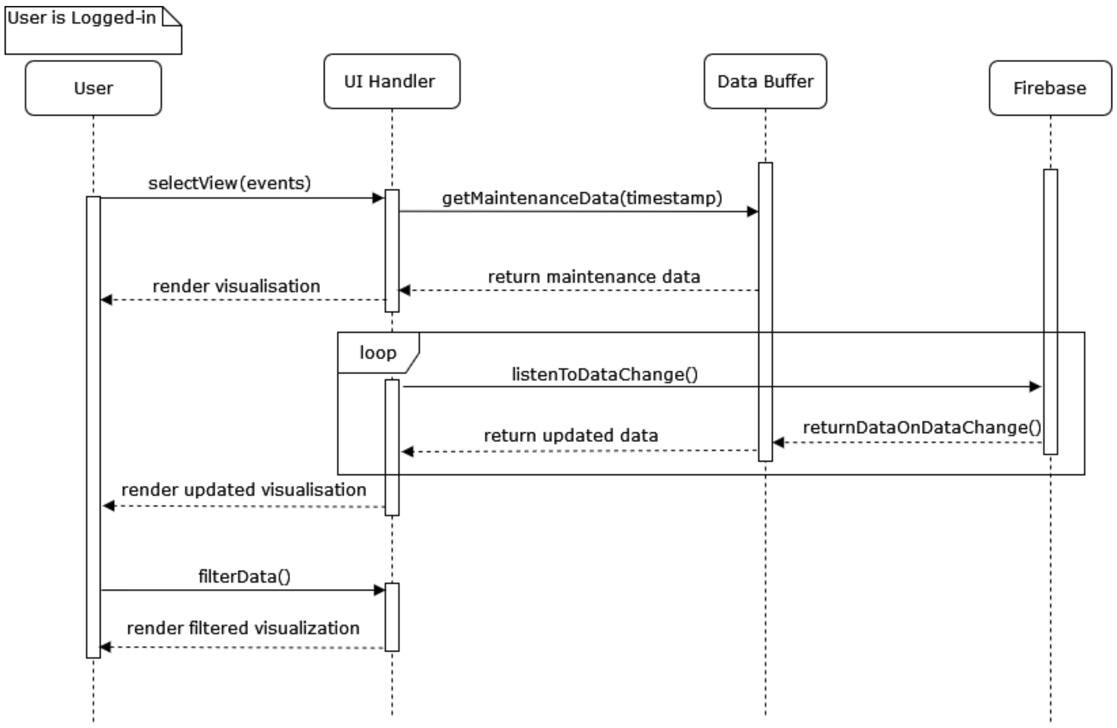

Figure 16: Maintenance Events Sequence Diagram

1.3.16 Weather Conditions

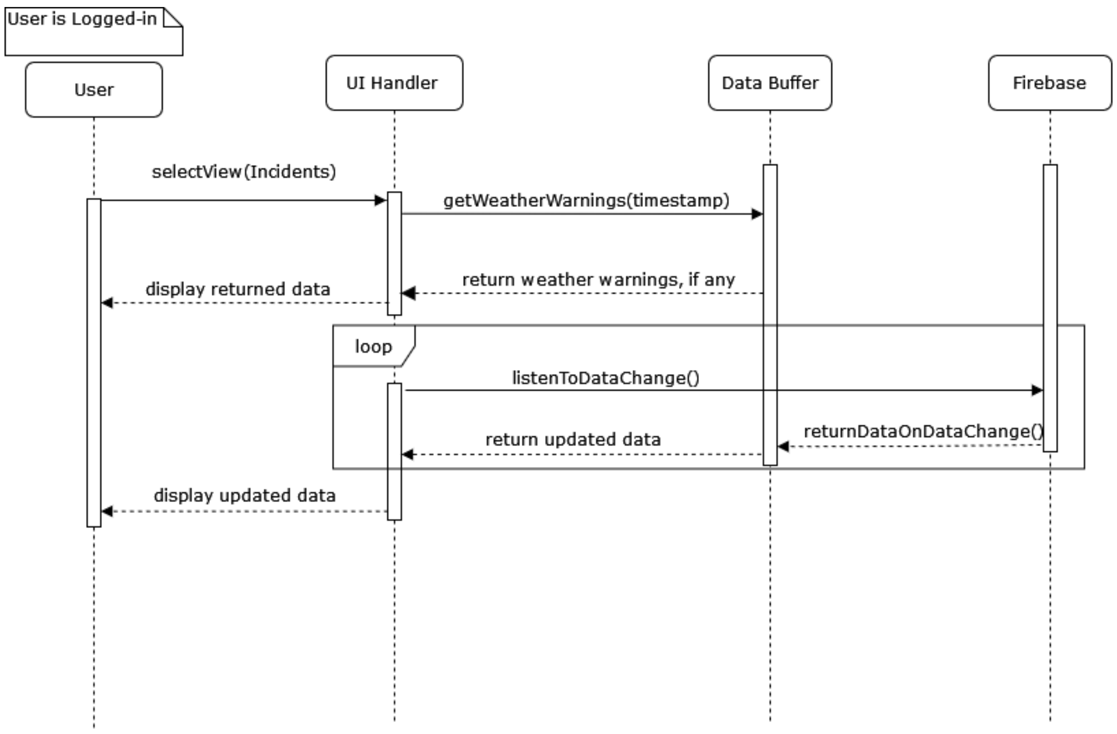

Figure 17: Weather Conditions Sequence Diagram

1.4. Component APIs

Live Data Handler

- *LiveLuas Data - fetchLUASData(timestamp)*
  This API is responsible for retrieving the live Luas data.

- *LiveBikeData - fetchBIKESData(timestamp)*
  This API is responsible for retrieving the live Dublin Bikes data.

- *LiveBusData - fetchBUSData(timestamp)*
  This API is responsible for retrieving the live Dublin Bus data.

- *LiveEventsData - fetchEVENTSData(timestamp)*
  This API is responsible for retrieving the live events data.

- *No return parameter - generatePredictions(dataTimestamp, liveData,dataIndicator)*
  This API sends the retrieved live data for each data indicator (identified by the *dataIndicator*) to the **Prediction Engine**, to generate predicted data for the next 24 hours.

Prediction Engine

- *No return parameter -transformData(dataTimestamp, liveData predictedData, dataIndicator)*

This API sends the live & predicted data to the **Data Transformer**, where it will be converted into data suitable for visualization/display.

Data Transformer

- *historicalStatistics -getHistoricalStats(dataTimestamp, dataIndicator)*

  This API retrieves relevant historical statistics for the *dataIndicator* provided. These statistics will be used by the **Data Transformer** to generate usage statistics over a wide range of time periods. For example, total CO2 emissions for the current week.

- *no return parameter - writeHistoricalStats(dataTimestamp, dataIndicator)*

  After the data has been transformed, and new up-to-date usage statistics have been generated, this API will write the new statistics to the **Web Service Provider** (Firebase).

- *no return parameter - publishData(dataTimestamp, transformedData)*
  After the data has been transformed, and new up-to-date usage statistics have been generated, this API will publish the data to the **Web Service Provider** (Firebase), where it will be disseminated to all registered devices. This data will include both live and predicted data covering the following 24 hour period.

UI Handler

- *authentication status - authenticateUser(Username, hashed password)*
  This API sends an authentication request for a particular user to the ***django*** server, containing a username and hashed password.

- *subscribe status - subscribe(Firebase_IP, client_unique_id)*
  This API is responsible for sending the initial subscribe request to the **Web Service Provider** (Firebase), following a successful login from the user.

The following APIs are responsible for requesting the data required for their respective visualizations/dashboard components. They are all requested by the UI Handler, from ***Firebase.***

Luas

- *getLuasMapData(timestamps)*
- *getLuasEnergyData(timestamps)*

Bus

- *getSuggestedRoutes(timestamp)*
- *getBusMapData(timestamp)*

- *getCO2Emissions(timestamp)*
- *getCO2Predictions(timestamp)*

Events

- *getWeatherWarnings(timestamp)*
- *getMaintenanceData(timestamp)*
- *getEvents(timestamp)*
- *getBusFreqSuggestion(timestamp)*

Bikes

- *getBikesBasicInfo(timestamp)*
- *getStationUsage(timestamp)*

User Details Management

- *validation status - validateCredentials(Username, hashedpswd)*

  This API validates the user credentials sent from the **UI Handler** against the credentials stored in the **Web Service Provider.**

2. Technical Architecture

2.1. Diagram

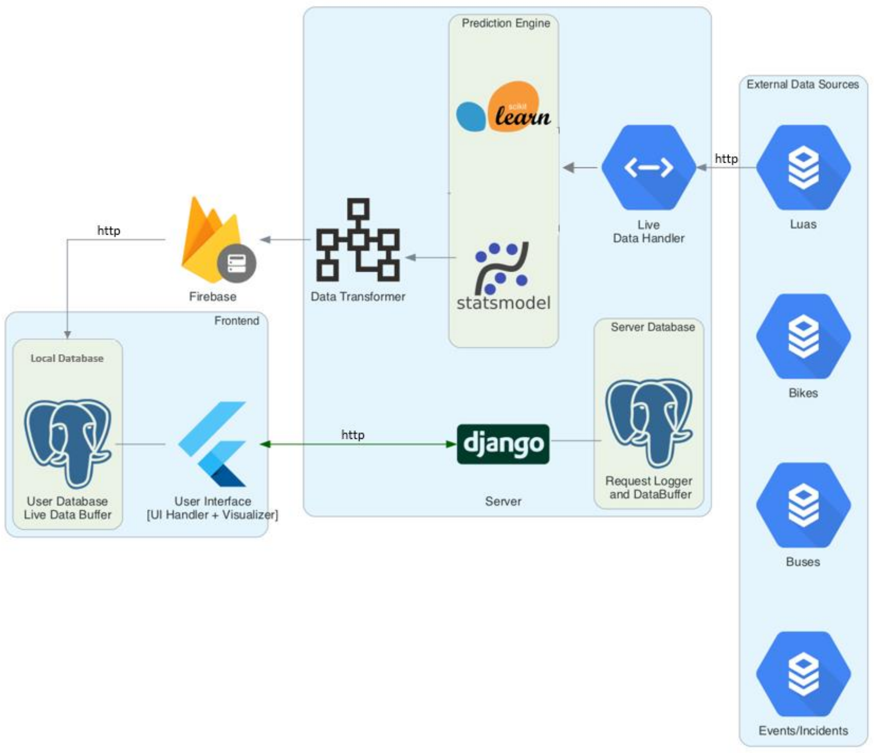

Figure 18: Technical Architecture Diagram

2.2. Quality of Service Technical Requirements

Security Requirements

Security Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Encryption - Do transactions need to be encrypted? - Level of encryption? (e.g., 40-bit encryption in US)	All the transaction data involved in the system are from the public domain and are not sensitive. The user data that has been collected will be encrypted using AES-256 encryption both **in transit** and **at rest**.
User Identification - uid/pw, cookies, certificates, application-level? - Existing customer database that should be used to identify online visitors?	At the application level, user identification will be achieved using a username and password combination. No existing customer databases are adopted and reused.
Access to data - Do you need to restrict access to parts of the site? - What privacy rules should be applied to information provided by users	Yes, the end-users will not have access to the site where predictions are generated. The end-users will only have **read-access** to the centralized database where all the predictions are published.
What are the legal requirements and policies for auditing content, changes, and transactions?	All the data that is used in the data pipeline are from external public domain data sources. No additional legal requirements are required to use this data, and Policies enforced by GDPR will be applicable for audits, changes, and transactions.
Do you plan to use a secure demilitarized zone in which your project server code could be placed?	Yes. The project uses a publisher-subscriber model. The server with open access to end-users that allows them to subscribe to the centralized database will be in the DMZ and the entire data pipeline that generates prediction will not be exposed to the end-users.

System Management

System Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***

Do you have access to the infrastructure required to install and run your own server?	Yes, the infrastructure requirement is minimal and we have the required access.
What are the response time targets?	Response time for rendering visualizations will be capped at 500ms.
Availability: - What hours should the service be available? - Is it acceptable to have any scheduled downtime for maintenance? - How important is it that the service be never interrupted, even for unscheduled component failures? - If interruptions do occur, what should be the target time for resuming service?	The service will work on the real-time data of the data indicators during their respective operating hours. During off-hours, the aggregated analytics will be displayed. Scheduled maintenance and downtime are accepted for this project since the number of end-users are limited and there is no need for 24*7 availability of the service. Fault-tolerant mechanisms will be implemented to provide uninterrupted service during operating hours. Dynamic API rerouting will be implemented so that the system is able to fetch data without interruptions, but in case of service interruption or component failure, there will be a target time of 12 hours for the service to resume.
How should partial or total service failures be monitored and handled?	Master-slave architecture pattern will be implemented in every component so that partial failures are tackled ensuring high availability of system service.
Do you need a recovery plan, or will it be covered by existing processes?	Since Master-slave architecture is implemented, the exact action plan for recovery will be covered by this design.
Tracking/Documenting: How should the architecture support the process of problem reporting, tracking and fixing? What statistics do you need to keep about the site, and how will they be analysed? What instrumentation should be included in the design to measure performance, response times and availability?	The system will allow the users to report the bug. When the user reports the bug the local copy of the logs will be sent to the server for tracking and fixing. The log file will include all the stats about user interaction with the system, and a separate log file will be maintained for the data pipeline. No separate components are added in the architecture to measure the performance and latency. The average response time of all the requests will be monitored and logged for audit. The individual logs will be maintained at the user end and will be sent only when there is an issue. All the system logs will be maintained in a separate

Should the architecture include a repository for statistical data?	repository in the system, and a separate repository will be maintained for storing user logs.

Client-side Management

Client-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Who is the customer? (Internet or Intranet) – affects browser choice	The core component of the product which provides prediction will be open only to certain users in an organization. The general visualizations will be open to the public.
What is the level of the user's skill?	There are no specific skill requirements to use the product, but familiarity with the city and different modes of transport is essential.
What languages should the site support?	Initially, the product is targeted only to city managers of Dublin. Only English will be supported, and support for other languages will be added as required.
What are the user's usage patterns? (search or browse)	The user usage pattern is expected to be 'browse' for the presented visualisations on various data indicators in our platform.
How will the application maintain its state?	The state of the application will be managed using session management and the state will be preserved across sessions using a local database.
Is there a need to distribute the application code, and if so, how will it be done?	There is no need to distribute the application code.
How will the choice of a client affect end-to-end response? (HTML, JavaScript, AJAX, JQuery, VBScript?)	The application will be served as a mobile app and a web application. The system design for both remains the same, with no change in the technical architecture.
What are the different user interfaces needed?	There will be a single consistent user interface across both the mediums where the visualizations and recommendations will be displayed to the user.

Network Management

Network Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Will the solution involve the internet?	For both publishing data to firebase by servers and retrieving data from firebase by subscribers will involve HTTP/2.

What protocols will be used? (*HTTP? HTTPS? FTP? RMI? Messaging? Etc)*	
What about data, object, and application placement? *projected transaction volumes, amount of data, interaction?*	4 API Calls every 5 minutes, and every API call will have approximately 100 KB of the data payload. Data payload to publish into firebase will be capped at 50KB, which will be pushed into all subscribed devices.
What security functions are required/provided by chosen protocol? *The level of encryption will affect this, and also performance!*	IPSec and TLS V.2 will be utilized for every network interaction.
How does the network affect end-to-end response time?	The system is designed to work with offline capabilities, the latency of the network will impact only the data pipeline and a little impact from the end-user perspective.

Server-side Management

Server-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Single server or multiple servers? Peer-to-Peer? Sensors?	Publisher-subscriber model, multiple servers for High Availability.
Geographic location for servers?	The location of the servers will be in the Europe region, inside Ireland.
End-user client to server, or server to server required also?	End-user client to server is required for Firebase subscription. Server-to-server is required for publishing data from the Data Transformer to Firebase.
What security functions are required on the server?	Security functions are responsible for enforcing secure connections, encrypting passwords and user logs data will be required.
How can the impact of server on end-to-end response time be estimated, and catered for in the architecture?	To minimise the impact of the server on the system, a publisher-subscriber model is adopted and offline capabilities are enabled for all the devices.

Application Logic

Server-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Will the site use client-side executables? What are their connectivity requirements?	Yes, .apk and .ipa will be required by the end-users to use the app. They should be able to connect to the internet to subscribe to a centralized database to receive updates.
How will the application be split between client-side and server-side logic? (*affects communications for validation etc/performance?*)	The system involves two separate high-level components that interact with each other through a centralized database. Mobile application + web application will be client-side, and the server-side logic will be isolated where the core business logic will be implemented. They will interact through the centralized database.
Additional access security required?	There is no need for additional access security

Connectors

Server-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
What external systems, applications, and (sensor) data do your project need to access?	The entirety of the project depends on the live data from the external data sources for the different modes of transport around Dublin.
How should data be transferred between different systems?	The aggregated data along with the predictions will be published to a centralized NoSQL database (firebase) and the end-users will subscribe to the database. Whenever there is a change in data, the difference payload will be published to all the subscribers.
How current does the information have to be? *Use caches?*	The recency of the data is capped at 5 minutes. This limit is consistent throughout the architecture, and a copy of the recent data along with the predictions for the next 24 hours is stored in the local database to enable offline capabilities.
Is synchronous or asynchronous access required? *Off-line OK?*	All the requests from client-side applications to the centralized database will be asynchronous. Each instance of the application will also maintain a separate local database to enable offline capabilities.
Is access to different operating systems, network	N/A

protocols, application environments required? *which connector? CICS? MQSeries? RPC?*	We are implementing a cross-platform solution that is compatible with the most common operating systems (Windows, Mac, Android, iOS etc.)
Are additional security policies required?	Additional Security policies are not required, as all data utilised for the visualizations are based on publicly available data. All user credentials will be encrypted.
Can scalability and performance requirements be predicted, and how will the project address these?	The current system design has the capacity to scale to a large number of users. The server capacity can be increased as required and redundant databases can be added to make the service highly available.