CS7CS3 Advanced Software Engineering Group Project

Functional and Technical Architecture

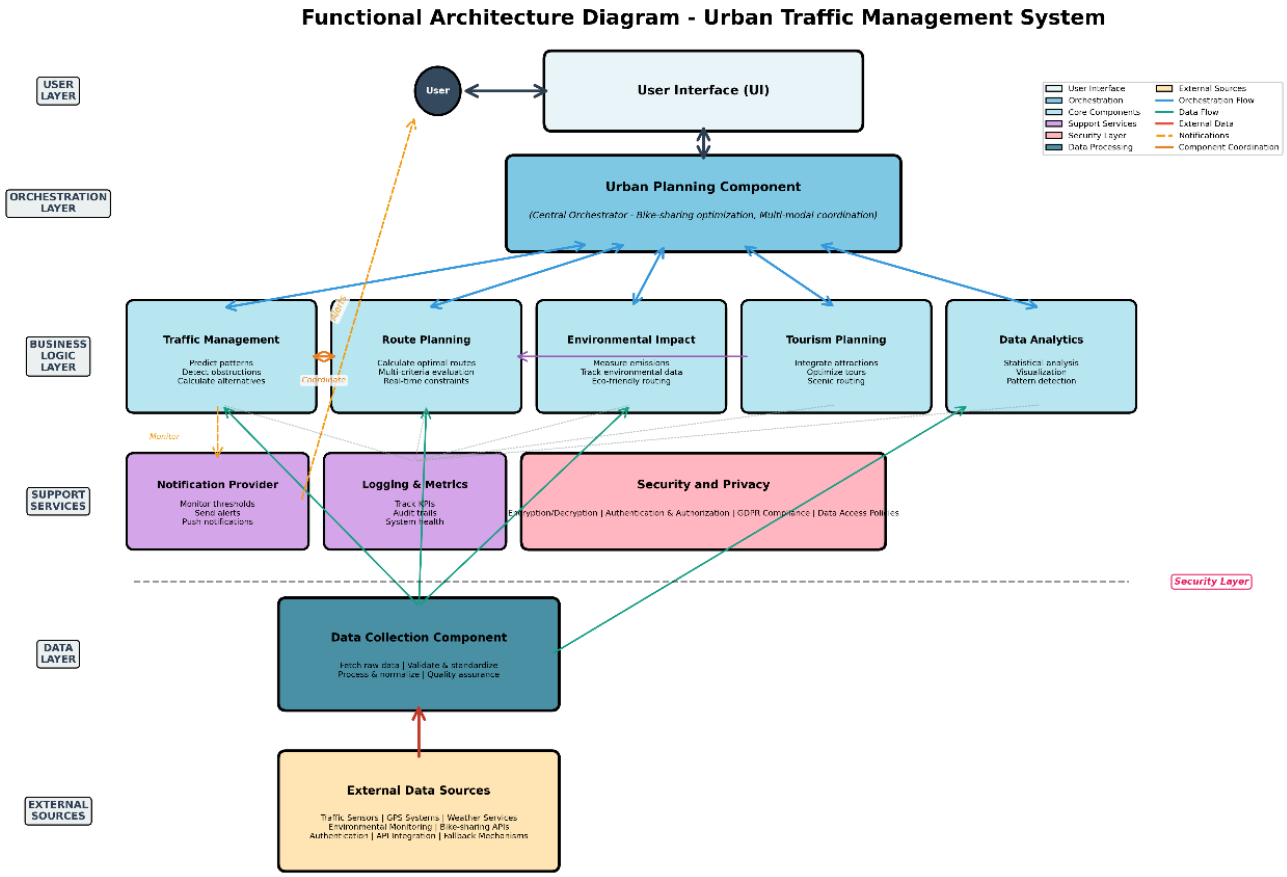
Project Name: Sustainable City Management

Group: 9
Aoife Walshe
Oisin Power
Parker Kavanagh
Darragh Hardiman Smyth
Charlie O'Malley
Diana Pylova

Contents

1. Functional Architecture

1.1. Diagram



1.2. Component Descriptions

Traffic Management
This component predicts traffic patterns and identifies road obstructions in real-time to optimize urban flow. It analyses historical and current traffic data to forecast congestion before it occurs, calculates alternative routes when obstructions are detected, and recommends flow adjustments to reduce delays. By coordinating with the Route Planning Component, it ensures that routing decisions account for dynamic traffic conditions.

Route Planning
The Route Planning Component calculates optimal routes based on multiple criteria including time, distance, environmental impact, and traffic conditions. It evaluates various pathways and selects routes that maximize efficiency while considering real-time constraints from the Traffic Management Component.

Environmental Impact
This component measures and tracks emissions generated by various transportation modes and urban activities. It calculates the environmental impact of different routing choices and provides data to support eco-friendly decision-making across the system.

Tourism Planning
The Tourism Component enhances user experience by integrating local attractions and points of interest into route planning. It provides information about notable locations along suggested routes and optimizes tour experiences by balancing scenic value with travel efficiency. This component helps make transportation recommendations more engaging while promoting local tourism opportunities.

Data Analytics
This component performs comprehensive data analysis and generates statistical insights from collected datasets. It creates visualizations including diagrams and charts to represent data patterns and trends effectively.

Urban Planning
The Urban Planning Component orchestrates and coordinates all other system components to achieve city-wide transportation goals, including bike-sharing infrastructure optimization. It manages bike availability calculation, identifies bike rack locations, and predicts demand at various stations to ensure balanced distribution across the city. This central hub integrates outputs from traffic management, routing, environmental, and analytics components while optimizing bike-sharing placement and accessibility to support multimodal transportation strategies and broader urban planning objectives.

Security and Privacy
Responsible for ensuring all data and transactions are transmitted and stored securely. Implements encryption and decryption mechanisms, manages authentication and authorization, and enforces privacy controls in line with GDPR, in addition to handling data access policies.

Notification Provider
This component will make periodic API calls and compare the results with user defined thresholds sending a push notification/ email to alert the user if the threshold is breached. e.g. User sets alert, choosing city centre traffic and setting the threshold as traffic speed not falling below 10km/hr.
This component is responsible for monitoring specific data values, and sending notifications

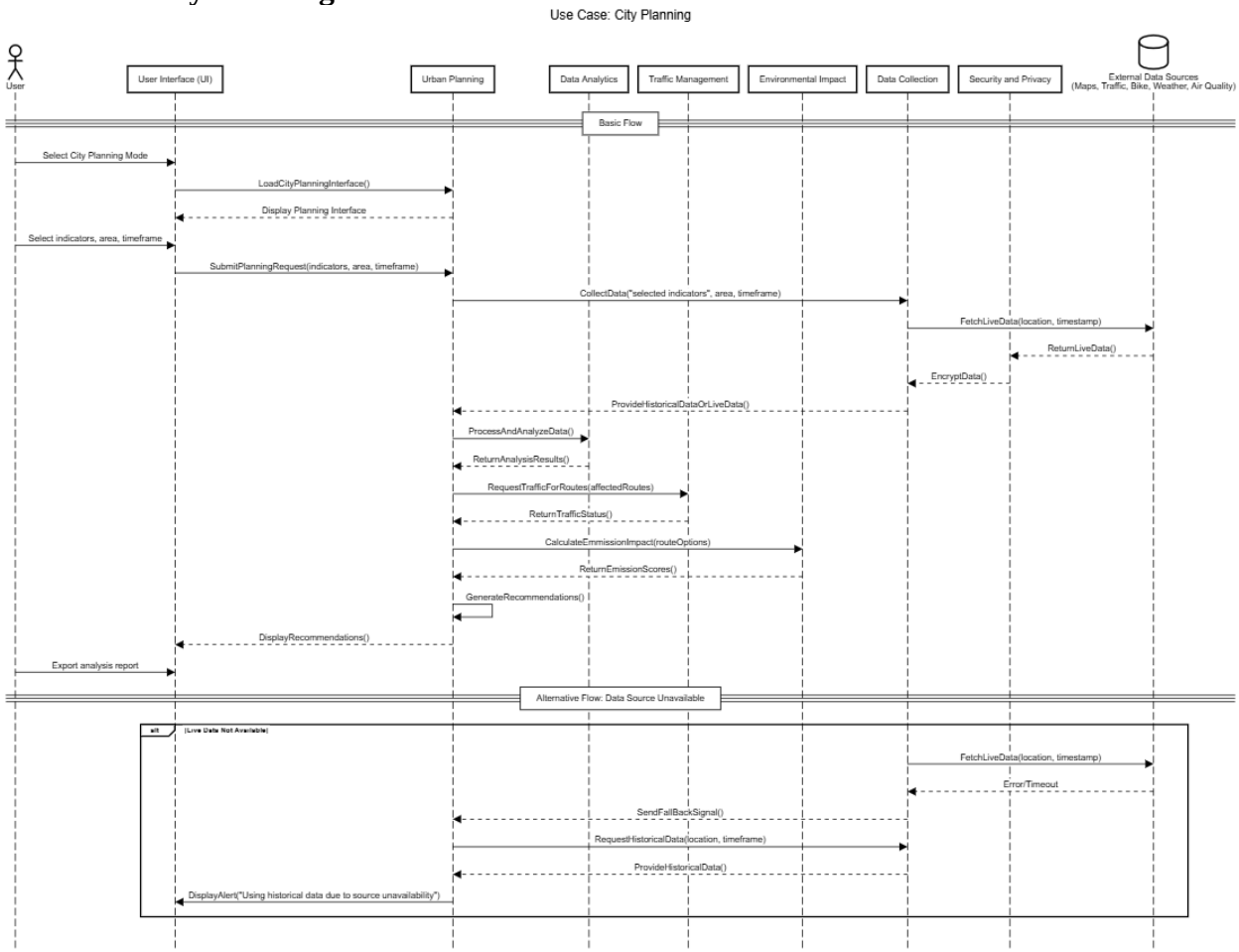
User interface (UI)
This is the component through which all user interactions will flow. It receives processed data from other components and users and presents it in an intuitive and accessible format. The module manages interactive elements, real-time updates, and responsive design across different devices. It ensures consistent user experience while displaying routing, traffic, and analytical information.

Logging and Metrics Management
This component collects, stores, and manages comprehensive logging and metrics data across all system components. It tracks key performance indicators, system events, and operational metrics to ensure visibility into system health and performance. The component maintains audit trails for security and compliance purposes.

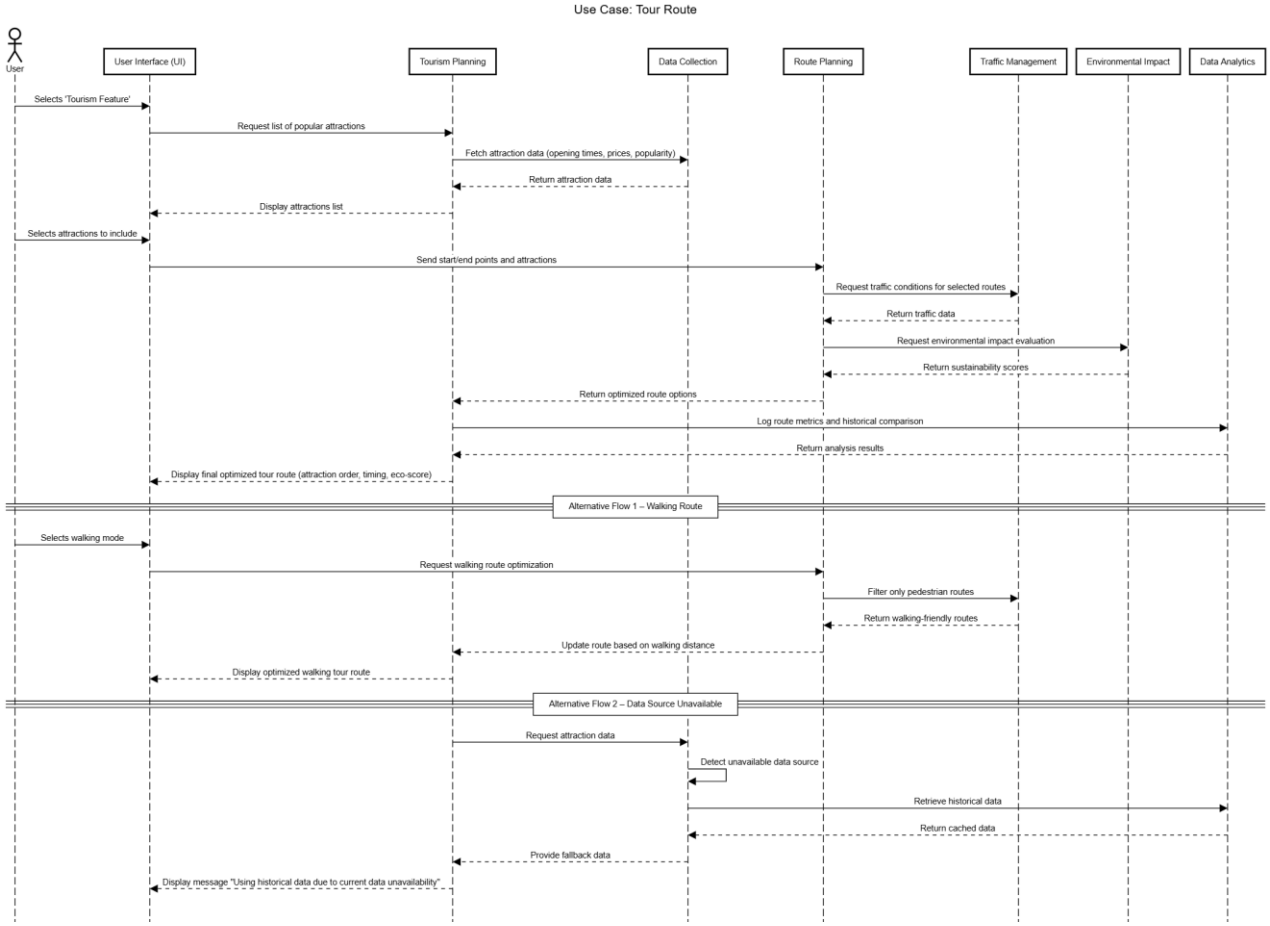
Data Collection
The Open-Source Data Collector serves as the primary interface for gathering raw data from various city sources including traffic sensors, GPS data, environmental monitors, and bike-sharing systems. It standardizes disparate data formats into a unified structure and feeds this information to downstream components. This component acts as the foundation for all decision-making in the system by ensuring data quality, consistency, and availability across the platform.

1.3. Component interactions

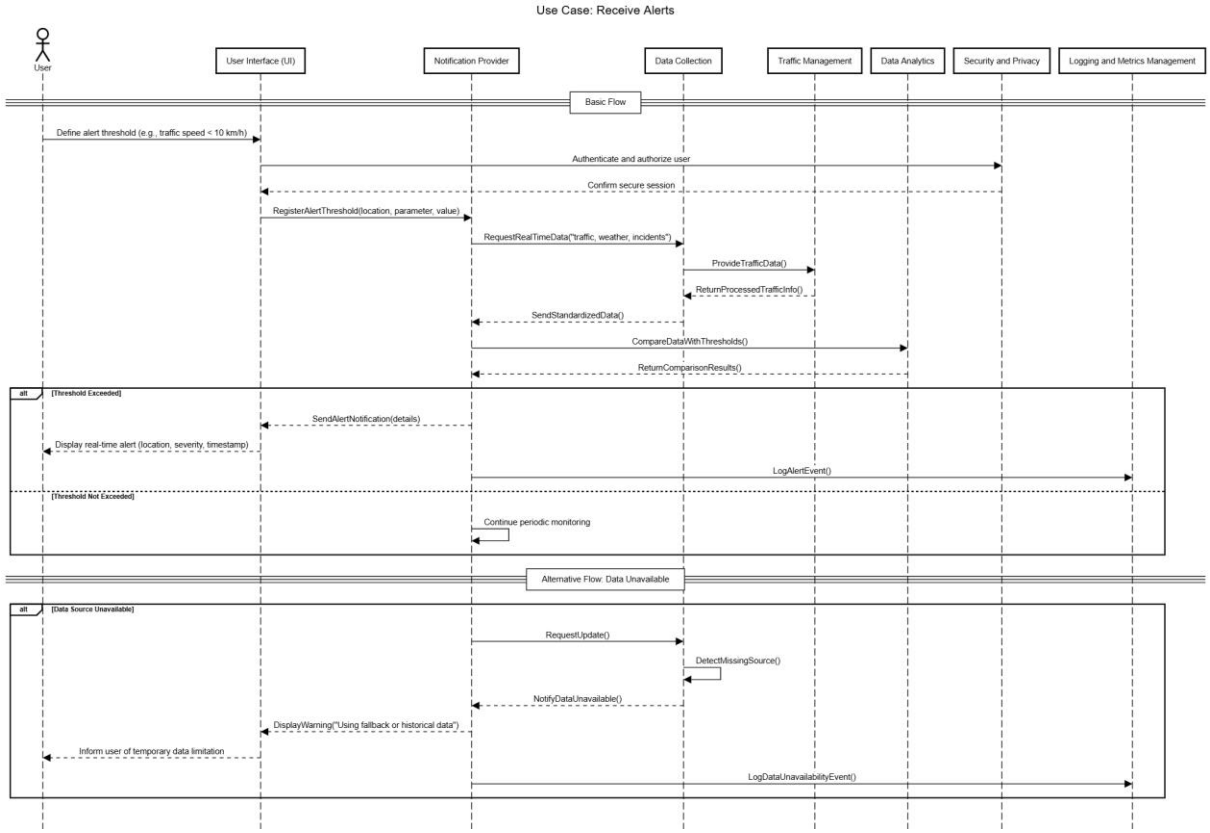
Use case 1 City Planning



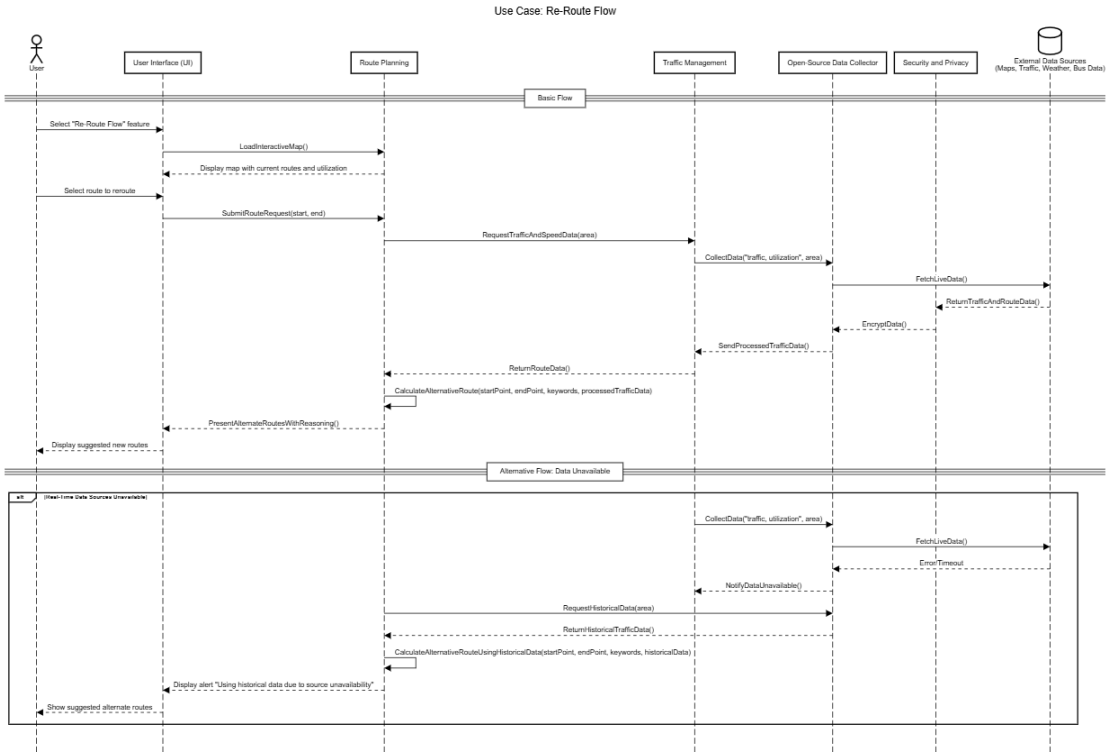
Use case 2 Tour Route



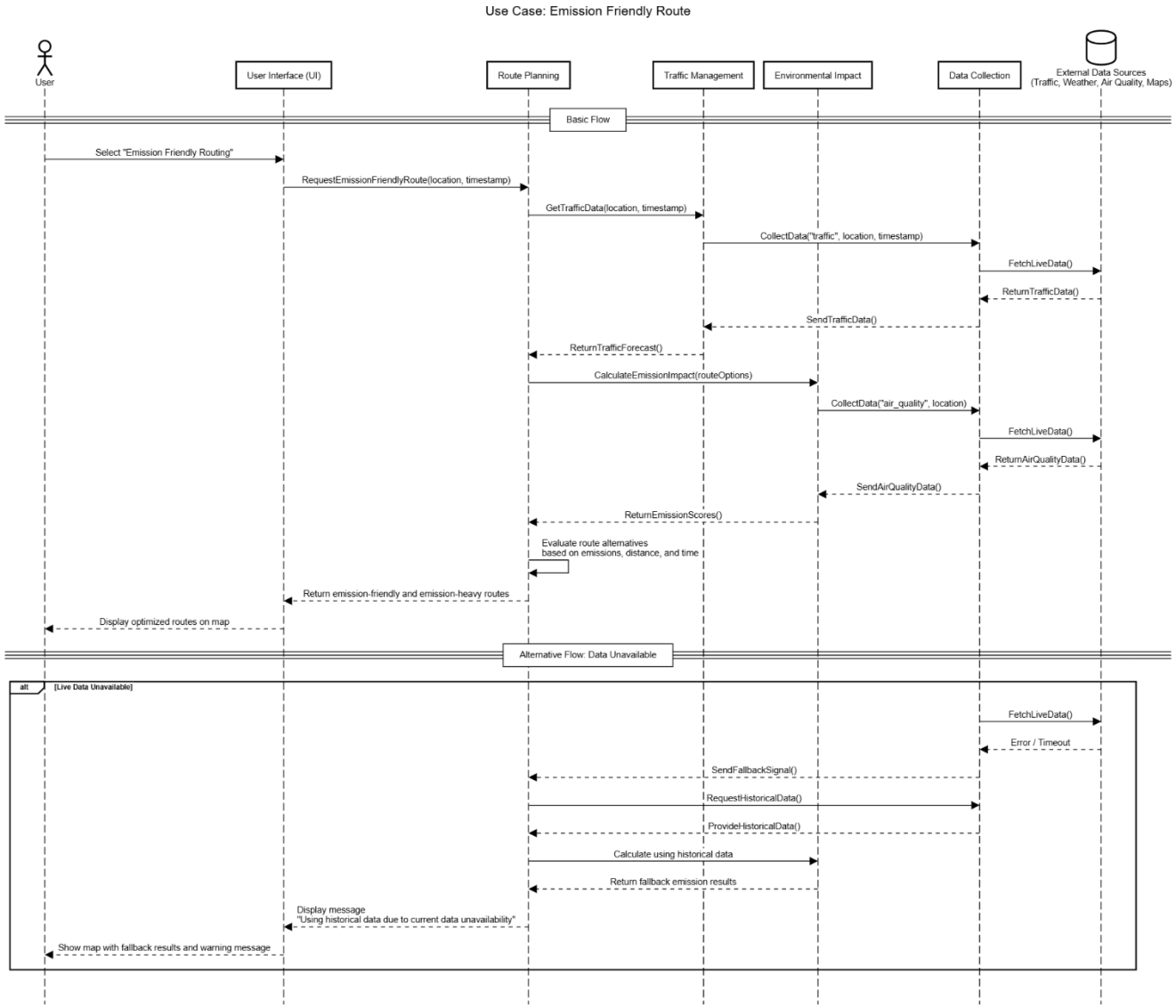
Use case 3 Receive Alerts



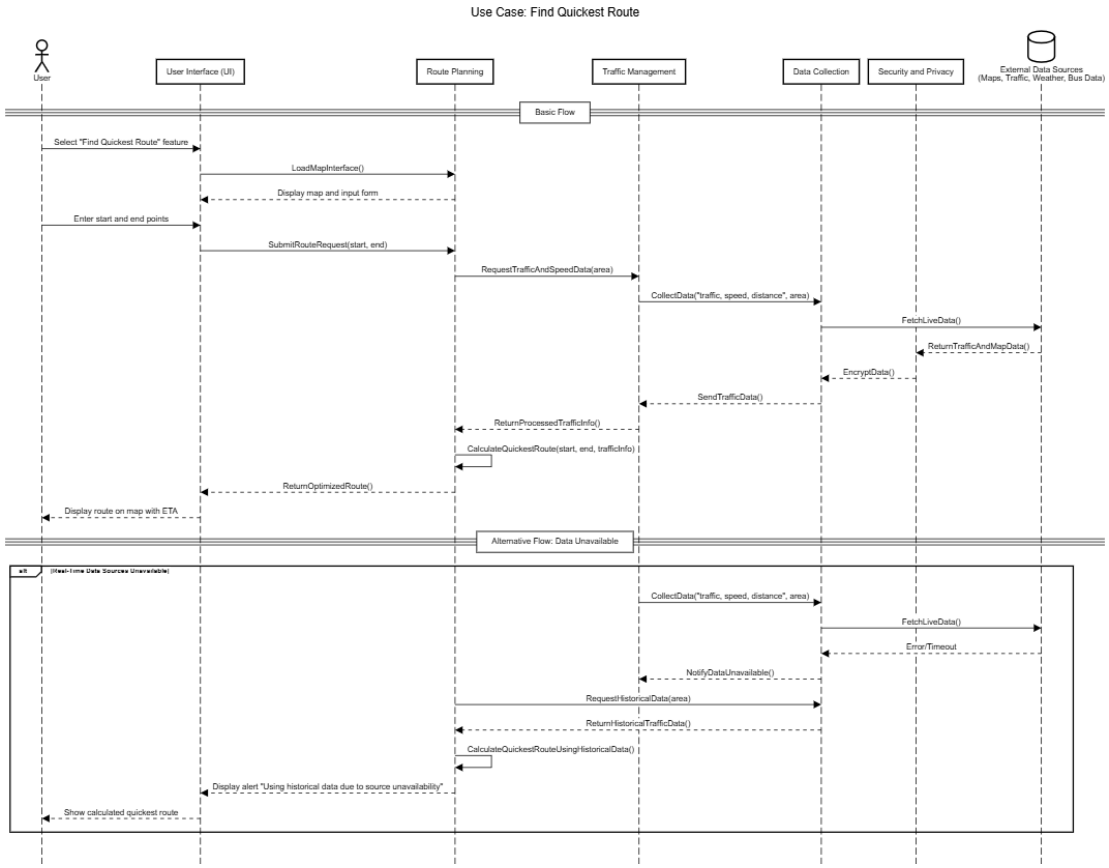
Use case 4 Re-Route Flow



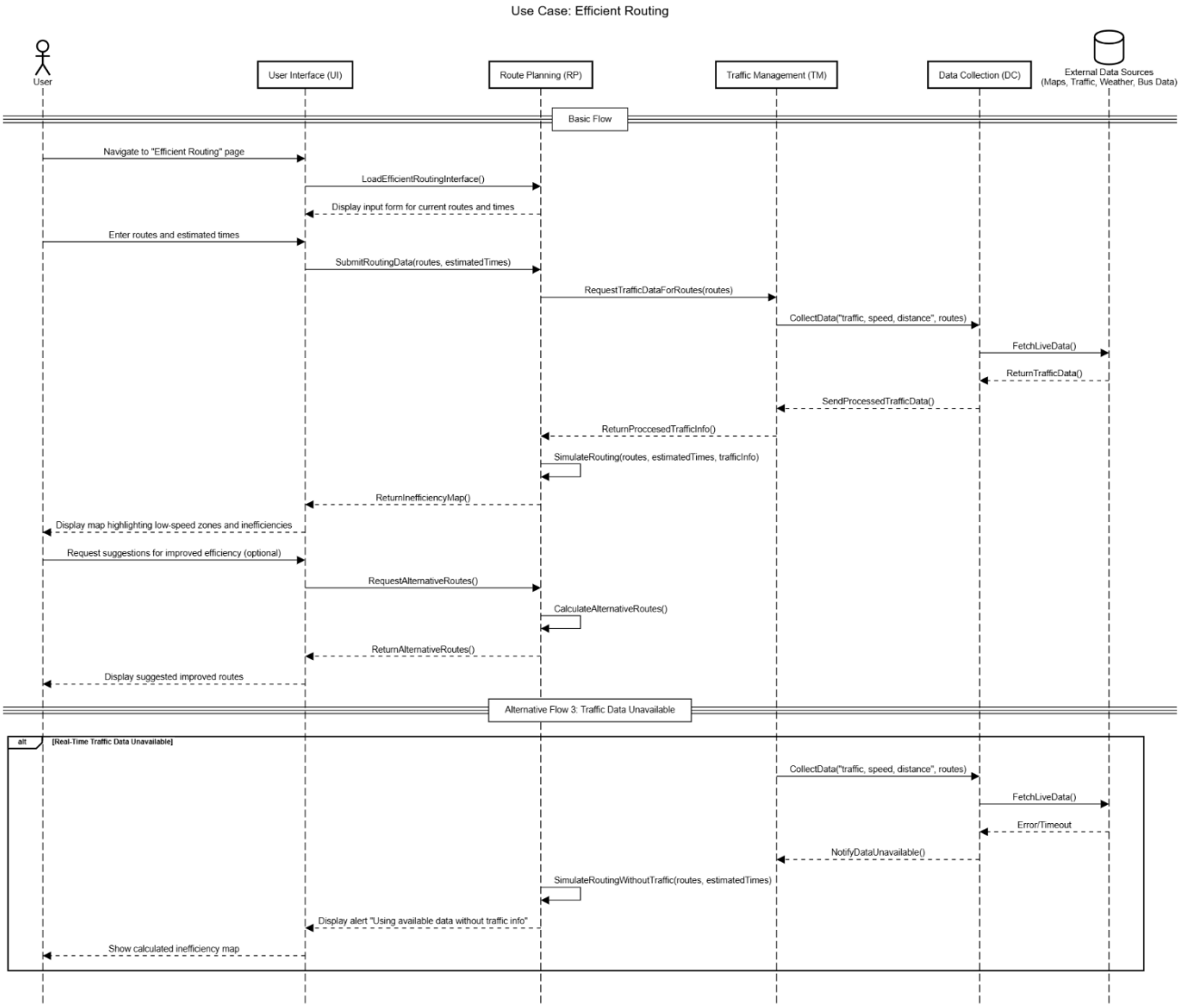
Use Case 5 Emission Friendly Route



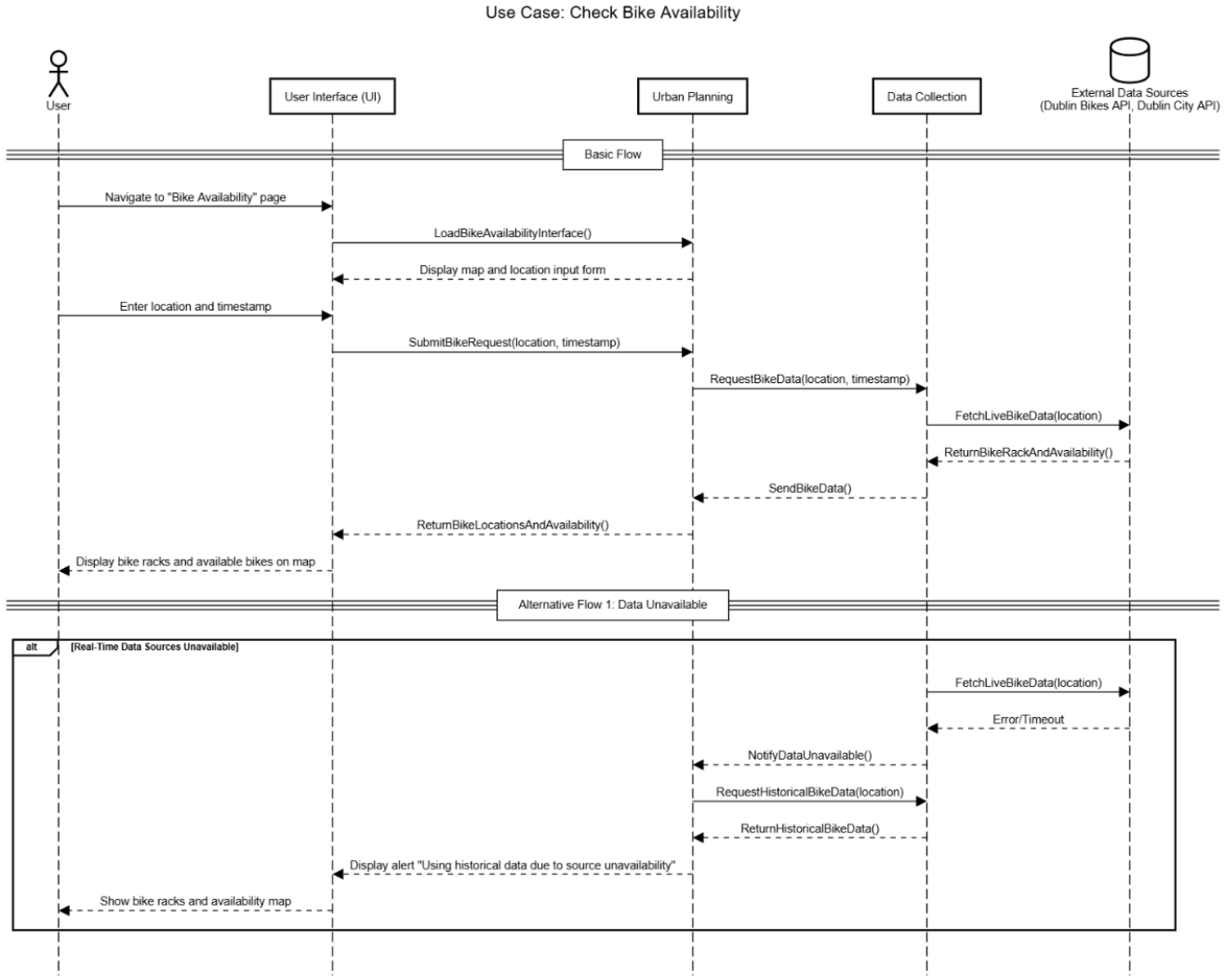
Use Case 6 Find Quickest Route



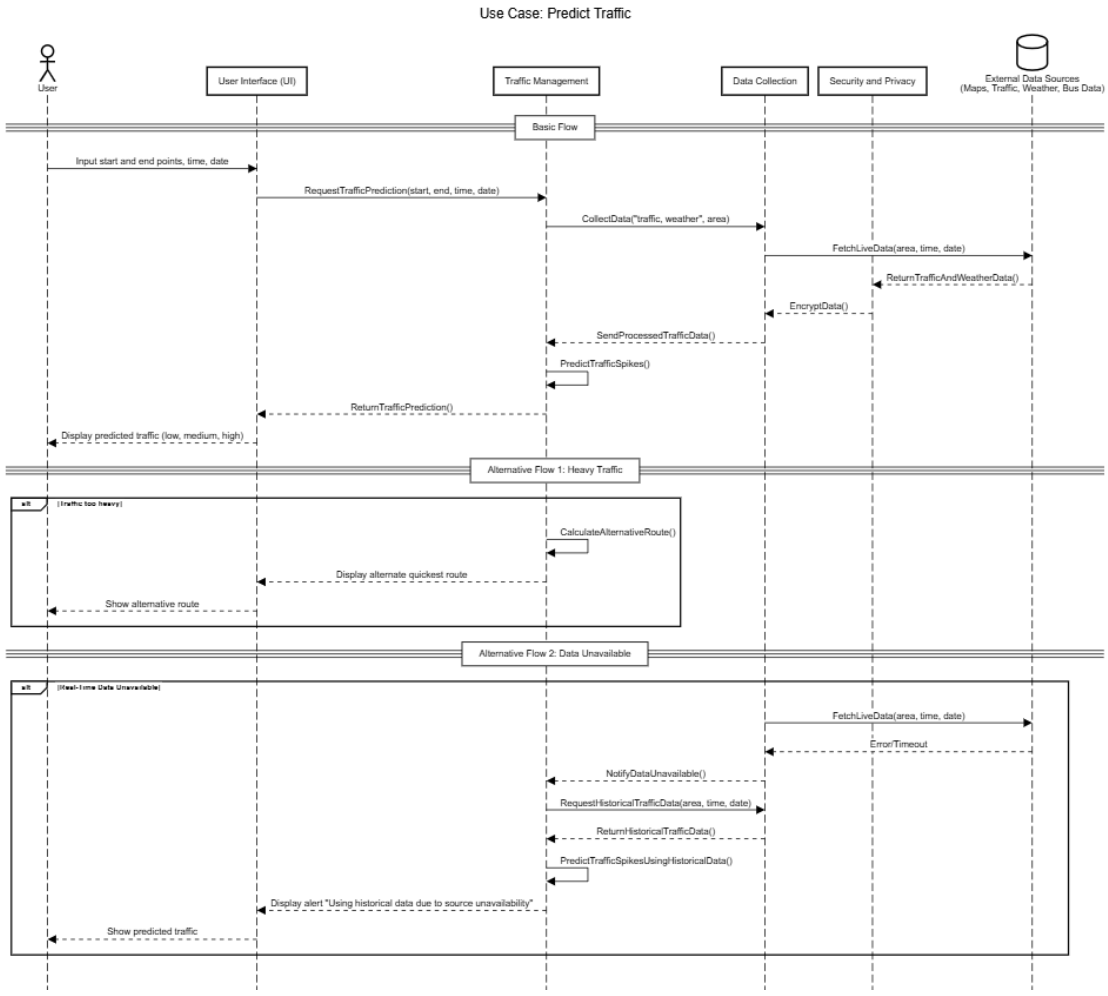
Use Case 7 Efficient Routing



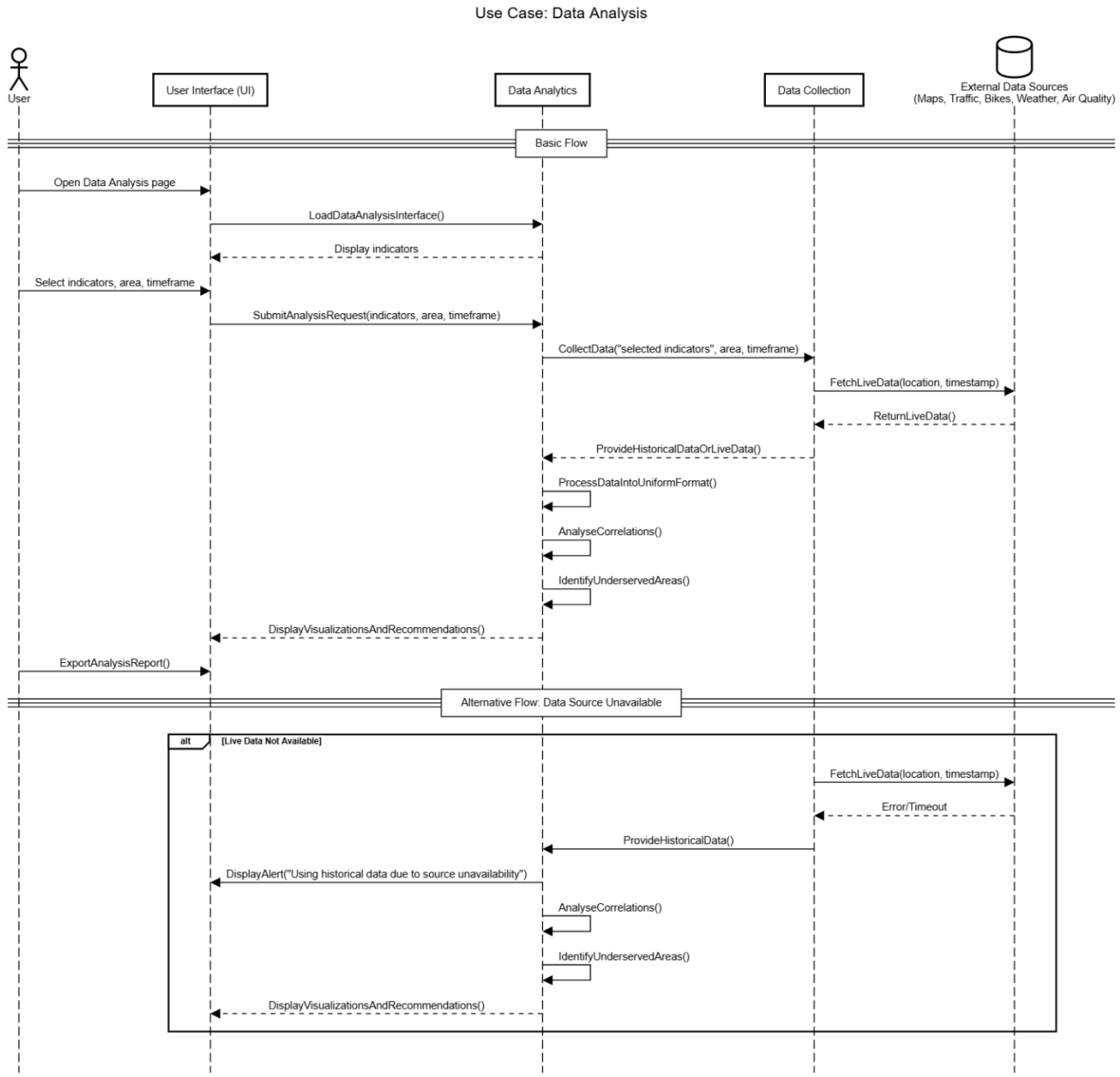
Use Case 8 Check Bike Availability



Use Case 9 Predict Traffic



Use case 10 Data Analysis



1.4. Component APIs

Component 1: Urban Planning

- ***SubmitPlanningRequest(indicators, area, timeframe)***
  Creates Request for data to be retrieved for the indicators in the area and timeframe specified
- ***LoadBikeAvailabilityInterface()***
  Returns bike availability maps and data to be displayed to the user.
- ***SubmitBikeRequest(location, timestamp)***
  Returns the number of available bikes at a given location and time.
- ***LoadCityPlanningInterface()***
  Fetches required data and returns it to be displayed to the user.
- ***generateRecommendations(indicators, area, time)***
  Generate and return recommendations based on given indicators, location and time. Should use all available data for the given timeframe and location.

Component 2: Traffic Management:

- ***RequestTrafficForRoutes(routes)***
  Accepts a list of routes and retrieves the current traffic conditions for each. The responsibilities include querying live traffic data for specific routes, aggregating and standardizing the data obtained & lastly, providing up-to-date traffic status for routing and navigation.
- ***GetTrafficData(location, timestamp)***
  Fetches traffic data for a specific location and timestamp. Responsible for retrieving and returning real-time traffic information while supporting location-based queries for time specific analysis.
- ***RequestTrafficPrediction(start, end, time, date)***
  Predicts traffic conditions between a start and end location for a given data and time using machine learning models or predictive algorithms with historical data. Estimates expected traffic congestion along a route incorporating weather events, and historical data to improve accuracy.
- ***CalculateAlternativeRoute()***
  Computes an optimal alternative route based on current or predicted traffic data to minimize travel time or distance. Analyses multiple route options with real-time and predicted traffic information to suggest the most efficient alternative route.
- ***PredictTrafficSpikesUsingHistoricalData()***
  Analyses historical traffic data to identify and predict potential traffic spikes. Utilizes the data to model traffic behaviour, identifying recurring congestion patterns and provides insight for proactive traffic management and scheduling.
- ***filterPedestrianOnlyRoute*()**
  Filters and returns routes that are accessible only to pedestrians. Responsible for identifying routes restricted to pedestrian access, supporting pedestrian navigation.

Component 3: Environmental Impact:

- ***CalculateEmmissionImpact(routeOptions)***
  Evaluates the environmental impact of one or more route options by calculating the estimated emissions based on factors such as distance, traffic conditions, and fuel efficiency. Responsible for analysing each routes potential carbon footprint and supporting eco-friendly decision-making route selection.
- ***requestEnvironmentalImpact()***
  Responsible for returning general environmental impact such as CO2, NO2 levels etc.

Component 4: Route Planning:

- ***RequestEmissionFriendlyRoute(location, timestamp)***
  Evaluates the environmental impact of given route options by estimating emissions based on distance, traffic conditions, and fuel efficiency. Responsible for supporting sustainability-focused decision-making.
- ***LoadEfficientRoutingInterface()***
  Loads and displays the user interface for eco-friendly routing, allowing users to view available route options and environmental statistics. Retrieves and initializes the routing interface components, displaying an interactive map and route visualizations. Ensures accessibility and responsiveness of the routing interface.
- ***SubmitRoutingData(routes, estimatedTimes)***
  Accepts and stores routing and estimated time data provided by users for analysis or simulation. Responsible for recording user-submitted data, updating the system database for future predictions and simulations, in addition to validating and processing the data before storing it.
- ***RequestAlternativeRoutes()***
  Generates multiple route options between 2 locations. It identifies and ranks alternative paths based on factors like distance, travel time and traffic conditions, providing users with optimal routes.
- ***DisplayAlert()***
  Display alerts or notifications to inform users about traffic changes, route updates, or environmental warnings. Notifies the user of system or route-related events, providing real-time feedback within the routing interface.
- ***SimulateRouting(routes, estimatedTimes, trafficInfo)***
  Simulates the routing process based on the user-provided routes, estimated times, and real-time traffic information to evaluate travel outcomes.
- ***CalculateAlternativeRoutes()***
  Computes an optimal alternative route based on current or predicted traffic data to minimize travel time or distance. Analyses multiple route options with real-time and predicted traffic information to suggest the most efficient alternative route.
- ***SimulateRoutingWithoutTraffic(routes, estimatedTimes)***
  Responsible for simulating routing scenarios without incorporating real-time traffic data, providing a benchmark for evaluating traffic effects, as well as supporting planning and analytical evaluations.
- ***walkingRouteOptimised()***
  Identifies and provides pedestrian-only routes, evaluating the routes based on distance, terrain, and accessibility.

Component 5: UI

- ***DispalyReturnedData()***
  Displays returned data in the correct format so that the user can easily understand the information given.
- ***DisplayIndicators()***
  Displays all available indicators to be picked for data analysis
- ***DisplayVisualizationsAndRecommendations()***
  Displays the visualizations and recommendations generated by the Data Analysis component

Component 6: Data Collection

- ***CollectData(key, location, timestamp)***
  Retrieves specific data points from external data sources based on the provided parameters. The key parameter determines the data type being requested, while location and timestamp specify where and when the data should be retrieved from, enabling targeted data collection from various external APIs.
- ***SendTrafficData()***
  Transmits retrieved traffic data to the requesting user or client application. Formats and packages traffic information obtained from external APIs into an appropriate response structure for downstream consumption.
- ***SendAirQuality()***
  Delivers air quality metrics to the requesting user or client application. Formats and transmits environmental data including pollutant levels and air quality indices after successful retrieval from external monitoring APIs.
- ***RequestHistoricalData()***
  Retrieves historical data records as a fallback when live data is unavailable or for comparative analysis. Fetches past data matching current request parameters to ensure service continuity, providing trend-based information and improving accuracy of real-time predictions.
- ***RequestBikeData(location, timestamp)***
  Retrieves bike-sharing system data filtered by specific location and time parameters. Optimizes data retrieval by requesting only relevant subsets of bike station information, including availability and dock status for the specified geographic area and time range.
- ***RequestHistoricalBikeData()***
  Fetches historical bike-sharing data using the same parameters as the original live data request. Maintains consistency with initial query parameters to provide comparable historical datasets when real-time data is unavailable.
- ***SendFallBackSignal()***
  Broadcasts a system notification indicating live data is currently unavailable. Triggers alternative data retrieval pathways and initiates historical data fetching to ensure graceful service degradation without request failure.
- ***ProvideHistoricalData()***
  Delivers retrieved historical datasets to the requesting function or user. Formats and transmits historical data in a structure compatible with live data responses to ensure seamless integration into existing workflows.

Component 7: External Data Source:

- ***FetchLiveData(location, timestamp)***
  Attempts to retrieve real-time data from live sources for the specified location and timestamp. Serves as the primary data retrieval method and triggers fallback mechanisms if live sources are unavailable or unresponsive.

Component 8: Notification Provider

- **RegisterAlertThreshold(location, parameter, value)**
  Creates alert threshold to be checked for specific locations and parameters. Responsible for configuring what to monitor and at what limits alerts should trigger.
- ***ContinuePeriodicMonitoring()***
  Performs continuous periodic monitoring of registered thresholds. Sends requests to retrieve updated data at defined intervals, compares current values against thresholds, and triggers alerts if values are breached.

Component 9: Logging and metric Management

- ***logAlertEvent()***
  Responsible for making a log in the database with details about the Alert created by the user including the parameters and the time of creation.
- ***LogDataUnavaiableEvent()***
  Makes a log in the database with details of the time when real-time data was not available and what data sources were not available.

Component 10: Security and privacy

- ***authenticateUser(username,password)***
  Responsible for verifying user credentials and grants or denies system access depending on the input. Handles role-based or permission-based access once implemented, ensuring the user can only access data or functionality relevant to their privileges.
- **EncryptData()**
  Responsible for ensuring that sensitive data is encrypted, protecting it from unauthorized access. Strengthens data confidentiality in storage and transmission and supports compliance with security best practices and privacy regulations. Encrypts data leaving the external data source ensuring confidentiality.

Component 11: Tourism Planning

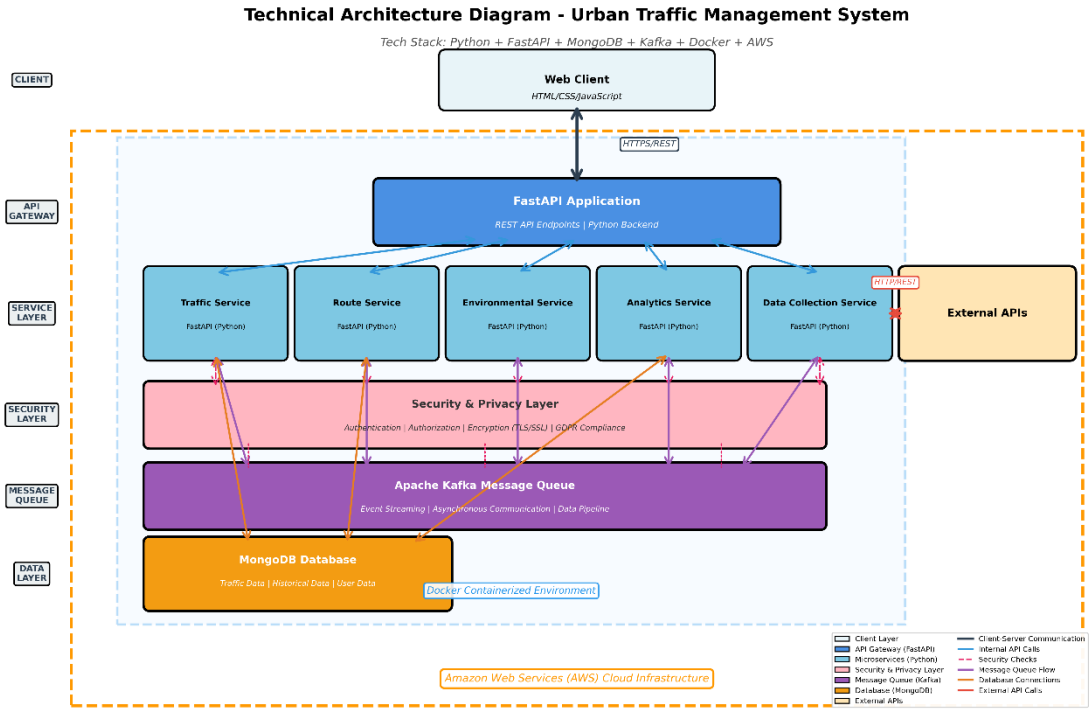
- ***requestPopularAttractions(location)***
  Retrieves list of tourist attractions with opening times and prices, sorted by number of visitors per year.

Component 12: Data Analysis

- ***logRoutesAndHistoricalComparison()***
  Records route information and compares it against historical data patterns. Logs current route selections and traffic conditions, performs comparative analysis with historical records to identify trends, anomalies, and deviations from expected patterns for performance evaluation and system improvement.
- ***retrieveHistoricalData()***
  Fetches historical records from the data storage system based on specified query parameters. Retrieves past data for analysis, comparison, or fallback purposes, supporting queries by location, time range, and data type to enable trend analysis and predictive modelling.
- ***ProcessAndAnalyseData()***
  Processes raw data and performs analytical operations to extract meaningful insights. Applies data transformation, cleaning, and statistical analysis techniques to identify patterns, trends, and anomalies, preparing actionable information for decision-making and system optimization
- ***CompareDataWithThreshold()***
  Checks collected or processed data against thresholds registered in RegisterAlertThreshold and determines whether an alert should be triggered.
- ***LoadDataAnalysisInterface()***
  Updates the UI to include necessary elements for data analysis providing users with interactive controls and visualizations for insights.
- ***SubmitAnalysisRequest(indicators, area, timeframe)***
  Submits a request to perform data analysis for specific indicators, areas, and timeframes, triggering data collection, normalization and correlation analysis and identification of underserved areas.
- ***ProcessDataIntoUniformFormat()***
  Normalizes heterogeneous data sources into a singular uniform format, ensuring data consistency for further analysis.
- ***AnalyseCorrelations()***
  Performs statistical analysis on the normalized data identifying relationships, correlations, and trends among indicators.
- ***IdentifyUnderservedAreas()***
  Reviews analysed data to find areas where demand exceeds supply. Responsible for highlighting regions where infrastructure or resources are insufficient.

2. Technical Architecture

2.1. Diagram



2.2. Quality of Service Technical Requirements

For mobility service providers, the platform ensures high availability and performance, allowing real-time delivery of routing, traffic predictions, city planning, and alerts to users. Features like efficient routing and traffic prediction are optimized are optimized using local caching and efficient data processing to minimize response times. While the system runs locally on a laptop, the microservices architecture allows individual features to operate independently, which helps handle multiple user requests simultaneously during high-demand periods, such as rush hours. Data accuracy is maintained through the use of historical datasets alongside sample real-time feeds, and secure local APIs ensure privacy of user inputs. The microservices design also makes it easy to update or add new routing features for testing or demonstration purposes.

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)	The transactions don't need to be encrypted. The dashboard will be served over HTTPS ensuring that the data is protected in transit and ensures integrity of the dashboard content. Content stored on the server for historical reference and preferences will be encrypted using AES-256. Cloud or on-premises data stores will use encrypted volumes and managed encryption keys, for instance AWS KMS.
User Identification - uid/pw, cookies, certificates, application-level? - Existing customer database that should be used to identify online visitors?	Access will be restricted to authenticated and authorized city users, said authorization will leverage Single Sign-On (SSO) integrated with the city's existing identity management system. We will use cookies to give customers better recommendations. This will be implemented using their JWT token.
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	Some access restrictions are needed given that the dashboard has a series of authentication levels depending on the user. Certain users will have access to certain data depending on role. User and user preference data will be encrypted using AES-256 and handled in line with GDPR regulations. Raw data will not be encrypted as we are just using open-source data.
What are the legal requirements and policies for auditing content, changes and transactions?	Transaction logs will record recommendations made along with data sources that contributed to decisions. Secure records of user's history will also be kept. Audit retention will comply with GDPR, ISO 27001, and local public sector information management policies.
Do you plan to use a secure demilitarised zone into which your project server code could be placed?	For production deployment, the system will be deployed within a secure network architecture, following defence-in-depth principles. The web front-end will reside in a DMZ accessible via HTTPS. The application logic layer and data integration layer will reside on internal subnets, not directly expose to the internet. Firewalls and API gateways will manage traffic between zones, and intrusion detection systems and web application firewalls will protect against common web attacks. By following a structure like so, we will be able to isolate critical systems from public exposure.

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 you own server?	Yes, the application could be deployed on a single sever as a single server process as high user traffic is not anticipated, in the case of unexpected high traffic causing response delays, some kind of load balancing could be employed to manage multiple server processes.
What are the response time targets?	We would be aiming for page loading of 2 seconds with real time updates every 30-60 seconds. Commands and executions would be 5-10 seconds
***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?	- 24/7 as traffic patterns and events don't follow 9-5 schedules - Yes during low impact hours but give advance notice to users - High availability is desired and the system could work at a reduced rate when something goes wrong - 15-30 minutes would be ideal in a production development standpoint
How should partial or total service failures be monitored and handled?	Basic logging will track server and API errors, and failures will trigger a visual indicator on the dashboard, total failures can be handled by hard resetting the server and reviewing the logs. In case of server failure, we could implement a log exporting strategy to retain some traceability.
Do you need a recovery plan, or will it be covered by existing processes?	If we are unable to access real time data, historical data will be used for analysis and predictions. Using persistent session data and a process manager could minimize the perceived effect of disruptions to the server process, anything more serious or unrecoverable may need to be resolved manually, in which case the user will be notified of the estimated down time.
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?	- A logger/system performance monitoring component will be introduced to collect and process all logs, metrics and traces from each other component and possibly provide an admin dashboard or equivalent to manually monitor the system status. - Administrators will be notified when critical issues occur. There will be a log for errors,

What instrumentation should be included in the design to measure performance, response times and availability? Should the architecture include a repository for statistical data?	warnings and system events that will be in severity categories. - Dashboard access patterns and API call volumes and times will be monitored. Response times for different dashboard views and prediction computation time will also be tracked. To analyse statistics accuracy could be compared, performance benchmarking over time and analysis on deteriorating data sources. - To measure performance, an admin dashboard could be used to show system health metrics and performance markers in code to track expensive operations. - It will include a repository for statistical data so as to store audit logs, performance metrics, historical data and model performance.

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	As the users will likely be based across different organizations and locations, the server will need to be accessible via browser on multiple networks, an internet connected server, and application would satisfy this constraint.
What is the level of the user's skill?	Basic computer literacy is expected, the intended audience aren't tech professionals. The application needs to be accessible for individuals with various technical levels.
What languages should the site support?	English is sufficient for this project given that it is intended to be used in Irish offices where the official workplace language is English.
What are the user's usage patterns? (search or browse)	Users will be specifically searching for information to inform decision making rather than casually browsing the dashboard
How will the application maintain state?	The application will use an SQL database on the backend to maintain any persistent data pertaining to a user or session.
Is there a need to distribute application code, and if so, how will it be done?	Access to this application will be via a website on a browser therefore code distribution is not necessary
How will the choice of client affect end-to-end response? (HTML, JavaScript, AJAX, JQuery, VBScript?)	The dashboard is served as HTML with JavaScript, most likely using AJAX to update the page without the need for a full-page refresh

What are the different user interfaces needed?	Single interface: the dashboard, in which the user will navigate to the intended tool, accessible via browser

Network Management

Network Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Will the solution involve the internet? What protocols will be used? *(HTTP? HTTPS? FTP? RMI? Messaging? Etc)*	- For the client browser to server communication, we will likely make use of HTTPS requests etc. - For notifications and alerts a pub/sub system could be set up with an MQTT broker. - For admin and maintenance access ssh will likely be used.
What about data, object and application placement? *projected transaction volumes, amount of data, interaction?*	Requested data (i.e. bus stop locations) will be ingested through secure APIs, retrieved via the appropriate open source and cached on the host server with a time to live depending on the type of data (i.e. bus stops unlikely to change but weather and traffic will change more frequently). Components could be cloud hosted ensuring scalability, reliability and high availability. Session and user data should be stored in a secure server-side SQL database on the server
What security functions are required/provided by chosen protocol? *Ievel of encryption will affect this, and also performance!*	HTTPS provides TLS encryption, data integrity and confidentiality. Data integrity is the most crucial for the application to ensure recommendations are based on correct information
How does the network affect end-to-end response time?	The network the server is in will be the main bottleneck and users' distance from this network will have the biggest impact on response time.

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?	The production system will initially run on a single server following a client-server model. However, through development cycles and expansions, the system could adopt multi-server, distributed architecture.
Geographic location for servers?	Initially the program will be run on a single local server. As the service expands more servers will be added to reduce latency and packet loss.
End-user client to server, or server to server required also?	Initially it will just be end-user to server as we will run off a singular server. As we expand there will be server to server communication to ensure consistency across all servers. However, the end-user will still only communicate with a single server.
What security functions are required on the server?	Server-side security is a critical part of the architecture with authentication and authorization being enforced through role-based access control. All data in transit will be encrypted in TLS 1.3 and at rest in AES-256. Integrity protection will be enforced using digital signatures, checksums and secure hashing, while system hardening will be address through regular patching, vulnerability scanning, and network segmentation. Lastly, backup and disaster recovery will be handled with automated daily backups with tested recovery procedures.
How can impact of server on end-to-end response time be estimated, and catered for in the architecture?	The server's performance will depend on its hardware capabilities and the response time of external data sources. Given the nature of the project, it is designed for low-latency and high-throughput performance, minor delays are acceptable. The system will use efficient data retrieval and caching where possible to improve responsiveness. In future iterations of the application where multi-server networking is a feature, load balancing will be used to distribute traffic evenly.

Application Logic

Server-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
Will site use client-side executables? What are their connectivity requirements?	The production dashboard will be delivered as a web-based application accessible via modern browsers on desktop. Said dashboard will communicate with backend services over secure HTTPS/TLS 1.3 connections to ensure data confidentiality and integrity.
How will application be split between client-side and server-side logic? (*affects communications for validation etc/performance?*)	Client-side logic will involve handling user interactions, input and visualization of results. The server-side will handle data retrieval and processing, returning the processes results to the client.
Additional access security required?	Admin will have full access to server configuration, databases, system logs and deployment management. Admin authentication will require multi-factor authentication and SSO integration with the city's identity management systems. Users will only have access to the dashboard and specific sections of said dashboard on a role-to-role basis.

Connectors

Server-side Management Requirement	***Impact on Project and how the Technical Architecture will address the impact***
What external systems, applications and (sensor) data does your project need to access?	Public Data APIs: Traffic, Weather, Air Quality, Bike availability, Bus Data, Tourist attractions
How should data be transferred between different systems?	The HTTPS protocol will be used for communication between the server and APIs and between the client side and the server
How current does the information have to be? *Use caches?*	Real-time is preferred however for many of the tools it is not essential. Historical data will be used when real time is not needed or available. Therefore, there will be a cache for the historical data
Is synchronous or asynchronous access required? *Off-line OK?*	The application requires asynchronous access so it can get data from different sources and update in real time without freezing. Offline access is not ideal as the application heavily relies on real time data, but we will have some features available offline depending on their process and the systems deployment.

Is access to different operating systems, network protocols, application environments required? *which connector? CICS? MQSeries? RPC?*	The application environment will be on the cloud so it can be accessed across networks, the server is likely to be run on Linux and can be accessed from any operating system with browser support
Are additional security policies required?	Api keys need to not be hard coded into the front-end application, instead being securely stored in the server backend. Input sanitization also needs to be implemented in order to make sure the website is not compromised by a malformed input.
Can scalability and performance requirements be predicted, and how will the project address these?	The application will perform expensive computations such as machine learning predictions for the whole region, so that it does not need to be repeated constantly, assisting scalability