Requirement Analysis and Specification Document

Best Bike Paths

Giuliano Crescimbeni

Luca De Nicola

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

In the world of cycling, it is often useful to record information about trips and track personal performance, as well as to share these experiences with others. Having access to updated data about bike pathssuch as their conditions, safety, and suitabilitycan greatly enhance both the enjoyment and safety of cyclists.

Best Bike Paths (BBP) was conceived in this context, with the goal of creating a digital platform where cyclists can explore, record, and share information about cycling paths. The system promotes collaboration among users, encouraging the community to contribute and maintain reliable data on the status of bike paths.

1.1.1 Goals

G1 - User Registration and Authentication: The system shall allow users to register and authenticate themselves to access the full range of functionalities provided by the application.

G2 - Trip Recording and Manual Path Registration: The system shall enable registered users to record their cycling trips, store them for activity tracking purposes, and manually insert bike path information by specifying street names and their current status.

G3 - Meteorological Data Integration: The system shall enrich recorded trip data with meteorological information (including weather conditions, temperature, and wind speed) retrieved from external services, when available.

G4 - Automated Mobile Data Acquisition: The system shall acquire data automatically from users' mobile devices during cycling activities, including GPS coordinates for path reconstruction and accelerometer/gyroscope data for obstacle detection (e.g., potholes).

G5 - User Validation of Automated Data: The system shall require users to confirm or correct automatically acquired information before making it available to the community, ensuring data accuracy and minimizing false positives.

G6 - Community-Driven Path Information Publishing: The system shall allow registered users to insert bike path information and make it publishable to the community, contributing to the shared knowledge base.

G7 - Universal Path Search and Map Visualization: The system shall allow any user (registered or non-registered) to specify an origin and destination, and visualize the available bike path(s) between these two points on an interactive map.

G8 - Path Scoring: The system shall compute a score for each bike path based on its current status and effectiveness in connecting origin to destination, and visualize paths to users ordered by their computed score.

G9 - Multi-Source Data Consolidation: The system shall merge bike path information collected from multiple users by considering data freshness and the number of confirming reports to determine the most accurate status assessment.

1.2 Scope

Best Bike Paths (BBP) is an application designed to support cyclists in discovering, recording, and sharing information about bike paths. Through BBP, registered users can record their rides, visualize the paths on a map, and obtain performance statistics such as distance, speed, and duration. When available, the system automatically integrates meteorological informationincluding temperature, wind speed, and weather conditions.

Users can insert path information either manually, by specifying the streets and their conditions, or automatically, by allowing the application to collect GPS, accelerometer, and gyroscope data during a trip. This data helps identify irregularities such as potholes or rough road segments. Before being published, automatically detected issues must be validated by the user to ensure reliability.

The platform provides map-based search and visualization tools allowing any user, registered or not, to explore available paths between a chosen origin and destination.

1.2.1 World Phenomena

WP1 - A cyclist rides along a bike path

WP2 - A cyclist seeks to discover a new path

WP3 - A person recommends a starting point, destination, or place to another cyclist

WP4 - Changes in weather conditions

WP5 - Physical deterioration of a bike path

WP6 - Appearance of an obstacle along a bike path

WP7 - Irregularities or bumps on the road surface cause vibrations on the cyclists device

WP8 - A user enables localization services

1.2.2 World-controlled Shared Phenomena

SP1 - A user registers or logs into the system by entering their credentials

SP2 - A cyclist records a bike trip by manually entering path information (e.g. street names, status, obstacles ecc.)

SP3 - A cyclist publish (or make private) a previously saved trips

SP4 - A cyclist starts or stops the automatic tracking mode of a ride

SP5 - A cyclist confirms or corrects the information acquired by the automatic tracking system

SP6 - A cyclist decides to publish a personal recorded trip

SP7 - A user searches for available paths between two locations

SP8 - A user views the information of a specific bike path

1.2.3 Machine-controlled Shared Phenomena

SP9 - The system adds supplementary data after a path is submitted (such as total distance covered, average speed, and other performance metrics)

SP10 - The system contacts an external weather service

SP11 - The system enriches a path by adding weather data retrieved from the external weather service

SP12 - The system collects data from the cyclists device sensors and creates a new trip

SP13 - The system detects irregularities along the path and presents them to the user at the end of the trip

SP14 - The system assigns a score to each path based on its condition and its effectiveness in connecting the selected origin and destination

SP15 - The system displays the available paths on a map between two points based on their score

SP16 - The system merges data acquired from multiple users for the same path, based on data freshness and the number of consistent confirmations

1.3 Definitions

- **Automatic Tracking Mode**: A functional mode of the mobile application where the system continuously collects data from the devices sensors (GPS, accelerometer, gyroscope) to reconstruct the cyclist's path and detect road irregularities without requiring manual input during the ride.
- **Data Freshness**: A parameter used by the Merging Engine representing the age of a submitted report. The system prioritizes more recent data (high freshness) over older data when calculating the consolidated status of a road segment.
- **Draft**: A temporary state of a recorded path (specifically from Automatic Creation) that has been saved locally or on the server but has not yet been validated or finalized by the user. Drafts are not visible in the personal history until confirmed.
- **Merging Engine**: The logical component of the system responsible for consolidating overlapping path segments from different user submissions into a single, consistent global view. It applies the majority consensus algorithm and data freshness logic.
- **Obstacle**: A verified physical impediment or hazard located along a bike path that affects the ride quality or safety.
- **Path Score**: A numerical value computed by the system for each path. Provide a single indicator of the path's "goodness" for ranking search results.
- **Path Segment**: The atomic unit of a path. A path is composed of an ordered sequence of segments.
- **Path Status**: A categorical value assigned to a segment or path indicating its condition. The valid values are: Optimal, Good, Sufficient, Poor, Unsuitable.

- **Path/Trip**: A defined path consisting of a sequence of segments connecting an origin to a destination. A path can be created manually or derived from a recorded trip and can be stored as Private (visible only to the creator) or Public (visible to the community).
- **Post-Ride Validation** : The process where a Registered Cyclist reviews the data collected automatically to confirm real issues and remove false positives before the data is permanently saved.
- **Segment Snapshot**: A specific instance of a road section recorded by a user at a specific point in time. The system aggregates multiple snapshots of the same location to derive the global status.

1.4 Acronyms

- **BBP**: Best Bike Paths
- **UML**: Unified Modeling Language
- **API**: Application Programming Interface
- **GDPR**: General Data Protection Regulation
- **GPS**: Global Positioning System
- **UI**: User Interface

1.5 Abbreviations

- **Gn**: Goal number n
- **Rn**: Requirement number n
- **Dn**: Domain assumption number n
- **WPn**: World phenomena number n
- **SPn**: Shared phenomena number n
- **UC**: Use case

1.6 Revision History

- **Version 1.0**: 17/12/25

1.7 Reference Documents

- Specification Document: Assignment RDD AY 2025-2026
- Course slides

1.8 Document Structure

- **Introduction**: Presents the purpose and goals of the document, the context in which the system operates, and a brief overview of the software to be developed.
- **Overall Description**: Provides a high-level view of the system, describing its main functions, the type of users involved, and the constraints under which it operates.
- **Specified Requirements**: Details the functional and non-functional requirements of the system. This section includes use cases, scenarios, and detailed interactions between users and the system, as well as performance, reliability, and usability requirements.
- **Formal Analysis**: Describes the formal modeling of the systems behavior through conceptual tools such as UML diagrams, Alloy models, or other formal methods. This section ensures the consistency and correctness of the requirements.
- **Effort Spent**:Summarizes the amount of work dedicated to each phase of the project, specifying the contribution of each team member and the approximate time spent on the various activities.
- **Software Used**: Software used to develop the document.

2 Overall Description

2.1 Product Perspective

2.1.1 Scenarios

Cyclist registers on BPP: The cyclist Bob accesses BPP platform for the first time and decides to create an account. He opens the registration form and enters the required information, including his email address, password, and general personal details (such as name, surname, and date of birth). Once the registration is completed successfully, Bob gains access to the main dashboard of the platform, where he can start exploring available paths or contribute by creating his own.

Cyclist manually uploads a trip: After completing a cycling trip from street A to street B, Bob logs into BPP and clicks on the Create New Trip button on the homepage. Manually enters the details of the path - including street names, and possible obstacles - and add a short descriptions to help other cyclists. The system compute statistics like average speed, total distance and other metrics and enrich them with meteorological information. Before saving, Bob set path status to "Optimal" and decide to make the trip public (visible to all users), enabling the "Publish" option. Once submitted, the trip is stored in the system and appears in Bobs personal list of recorded paths. A score is given to the path based on the uploaded information.

Cyclist uploads a trip using automatic mode: After enabling the location services on her device, Alice goes to her BPP profile settings and activate the automatic tracking mode. Some time later she goes for a bike ride while carrying her mobile device. During the ride, BPP automatically records the trip, collecting data such as path segments, total distance, average speed, and detected obstacles. The system enriches with meteorological information. Once the trip is completed, Alice accesses the platform to review the automatically generated path. She confirms the details of the trip, confirms the accuracy of the detected obstacles, and decides to keep the trip private. Finally, she clicks the Save Trip button to store the finalized version in her personal collection. A score is given to the path based on the uploaded information.

Unregistered user searches for paths: Eva, a user who is not registered on the platform, navigates to the Search section of BPP, which consists of an interactive map and a search panel. She enters Street A as the starting point and Street B as the destination. After clicking the Search button, several available paths instantly appear on the map connecting the two locations. Among them, the highest-rated path is

the one previously uploaded by Bob. Eva clicks on this path to view its detailed information, including distance, path conditions, and reported obstacles.

Merging of path information: Alice and her friend, both registered users of BPP, manually record the same path from Street A to Street B. Each of them enters the path details and sets the path status to Sufficient. Since these submissions contain more recent data than the previously stored version of the path, the system automatically merges their information with the existing record. The path is updated with the status of "sufficient" determined by majority consensus, and the newly provided details are incorporated to keep the path information up to date and more reliable.

Publication of a previously saved trip: Alice, who had previously recorded a private trip, decides to make it public. She navigates to the My Trip section, selects the specific trip, and changes its visibility status to Public. Once confirmed, the trip becomes available to all users and is automatically merged with existing records of the same path submitted by other cyclists. The system performs the merge based on the **upload date** of the trip rather than its **publication date**, ensuring data consistency and chronological accuracy in the shared path database.

2.1.2 UML class diagram

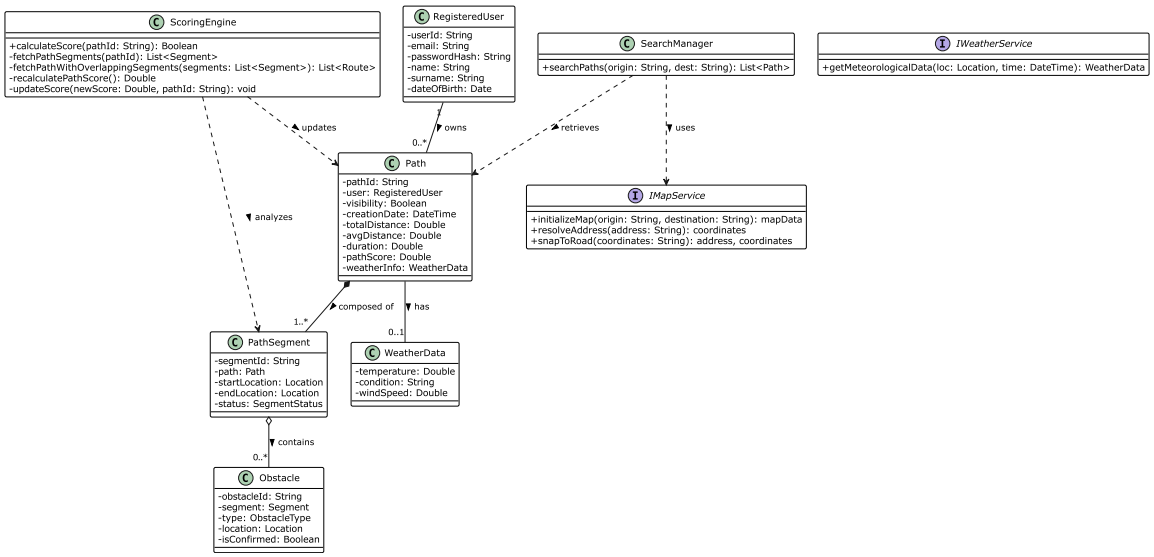

Figure 1: Class diagram of the BBP system

This diagram illustrates how the system manages cycling data. The **Path** entity is composed of multiple **Path Segment** objects. A distinct Status is identified for both entities: the segment status is explicitly defined by the user, while the overall path status is calculated based on the statuses of its composing segments. Additionally,

users can identify and insert specific **Obstacle objects** associated with a segment.

Segment states never get overwritten: each PathSegment is the same physical stretch observed at a **different point in time**. When computing the score of a given Path, for every stretch that compose it the system gathers all matching segments for that same stretch coming from other paths, then derives the score from the majority status, weighted by how fresh the observations are.

2.1.3 State Diagrams

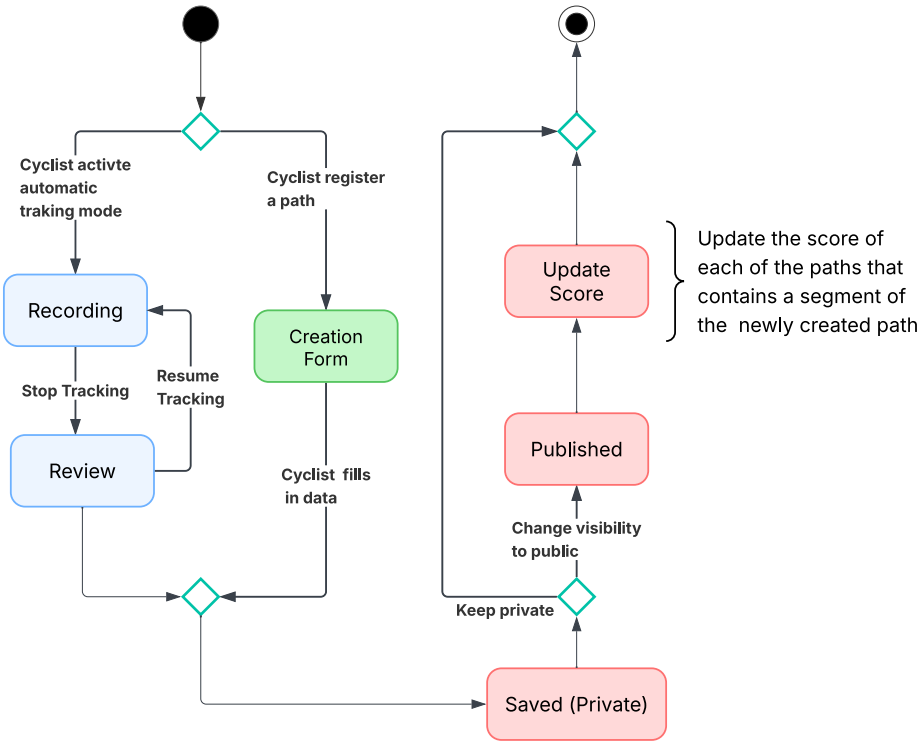

Figure 2: State diagram of the Path class

In the state diagram shown in **Figure 2**, the lifecycle of a single **Path** during its creation is described. It is important to note that once a new Path is created, the score of every existing Path that includes any of the path segments contained in the new one is recomputed and updated.

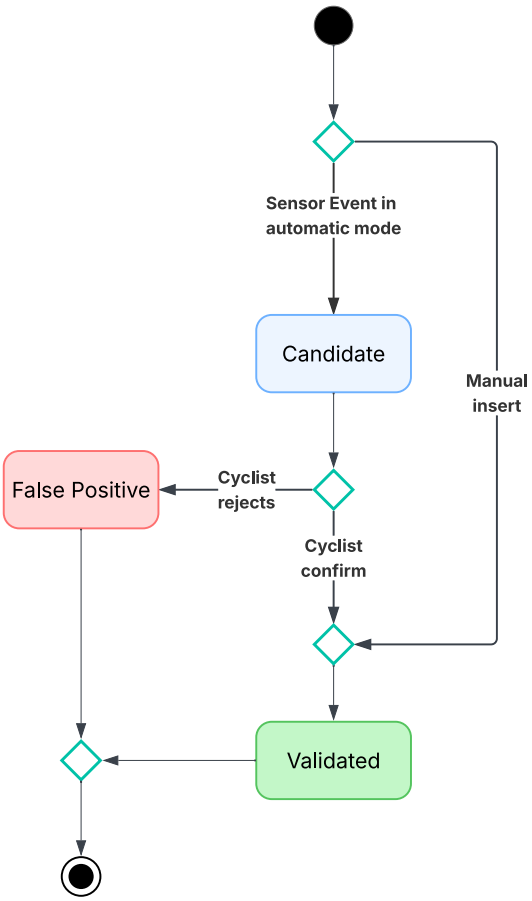

Figure 3: State diagram of the Obstacle class

In the state diagram shown in **Figure 3**, the evolution of a single **Obstacle** is illustrated, covering both manual insertion and automatic validation. An obstacle created through the automatic mode, of course, does not require a second verification step.

2.2 Product Functions

Registration and Login: This functionality allows cyclists to create a personal account on the platform to access advanced features. Registered users can manage their profile, maintain a personal history of recorded tracks, and contribute to the community by publishing their trips.

Trip Recording and Data Acquisition: The system offers two distinct modes for recording cycling trips. Users can manually insert path information by specifying street names, current status, and obstacles. Alternatively, an automatic tracking mode utilizes the device's sensors (GPS, accelerometer, and gyroscope) to reconstruct the path and automatically detect irregularities or road issues. To enhance the quality of the information, the system automatically enriches recorded trips with meteorolog-

ical data retrieved from external services. Furthermore, to minimize false positives from automatic detection, the system requires users to review and validate detected obstacles and path details before the data is stored or published.

Dynamic Scoring and Data Merging: The system automatically merges path information collected from multiple users, prioritizing data freshness and majority consensus to determine the most accurate status of a path segment . Based on this consolidated data, the system computes a specific score for each bike path, reflecting its safety and effectiveness.

Path Search and Visualization: The platform provides an interactive map interface that allows any user (registered or unregistered) to search for bike paths between a specific origin and destination. The search results visualize available paths ranked by their calculated score, allowing cyclists to easily identify the best and safest options for their ride.

2.3 User Characteristics

The BBP platform is designed to serve a diverse community of cyclists, from casual to more athletic. Users are categorized according to their interaction with the system: Registered cyclists and Unregistered cyclists.

2.3.1 Registered Cyclist

The registered cyclist is an active contributor willing to share personal ride data to support the community; therefore, he needs a system that enables him to easily publish trips and report road obstacles. Beyond contribution, he benefits from the community's information, requiring the capability to discover new paths and assess their conditions and safety scores derived from other users' data before starting. Since he interacts with the application primarily while riding,the automated data acquisition method could be require to minimize manual input and ensure safety during the activity. Furthermore, he plays a crucial role in maintaining overall data quality by reviewing and validating automatically detected anomalies before they are shared.

2.3.2 Unregistered Cyclist

The unregistered user is an individual who accesses the platform primarily for information retrieval without the need for performance tracking. His main goal is to find the best available bike path between two locations; therefore, he needs an immediate

and accessible search tool that does not require authentication. Furthermore, he relies heavily on the systems scoring mechanism to select paths that are not only fast but also safe and in good condition.

2.3.3 Common Characteristics

Both user categories share a fundamental need for reliability. Whether recording a trip or searching for one, users rely on the accuracy of the Path Score and the map visualization to make decisions about their trip. Both groups interact with map-based interfaces and require clear visual indicators of path status to plan their rides effectively.

2.4 Assumpion, dependencies and constraints

2.4.1 Domain Assumption

The following assumptions are properties of the real world that the system takes for granted. They must hold true for the software to function correctly and provide meaningful data.

D1 - Users possess a mobile device equipped with functioning GPS, accelerometer, and gyroscope sensors.

D2 - During the Automatic Tracking Mode, the mobile device is securely mounted to the bicycle, to ensure that sensor data correctly reflects road obstacles.

D3 - The GPS signal is available and has sufficient accuracy in the areas where the cyclist rides to allow for correct path reconstruction.

D4 - Users act in good faith when uploading a path in manual mode and when validating automatically detected obstacles, confirming only real issues and discarding false positives.

D5 - Users have access to an internet connection for map visualization, path searching, and data synchronization.

D6 - The external services integrated into the system provide accurate, truthful, and up-to-date information.

2.4.2 Dependencies

The system relies on external services to provide specific functionalities that are outside the scope of BBP development.

- Map Service Provider: The system depends on third-party mapping services to render the interactive map.
- Weather Service API: The system depends on an external meteorological service provider to retrieve real-time weather data.

2.4.3 Constraints

- The system must comply with local data protection regulations (e.g., GDPR), particularly regarding the storage and sharing of location history, which is considered sensitive personal data.
- Since the application uses power-intensive components (GPS and sensors continuously) during rides, the software must be optimized to minimize battery drain, ensuring it can track long-distance trips without depleting the user's device.
- The path scoring algorithm is constrained by the "freshness" of the data; the system must implement a decay logic where older reports have significantly less weight than recent ones.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The application interface is organized into the following areas:

- **Public Section** (Accessible to Unregistered and Registered Users):
  - **Landing and Search Section**: The entry point of the application, featuring an interactive map and a search interface where users can specify origin and destination to find bike paths. This section displays paths filtered by their calculated score and status.
  - **Path Detail Section**: A view dedicated to displaying specific information about a selected path, including total distance, average time, path status and a list of reported obstacles.
  - **Login and Registration Section**: The interface allowing new users to sign up and existing users to authenticate to access the private area.
- **Private Section** (Accessible only to Registered Users):
  - **User Profile Section**: Displays user details and aggregate statistics, such as total km traveled.
  - **My Trips Section**: A personal space where the user can view their history of recorded trips, distinguishing between private and public trips.
  - **Manual Upload Section**: A form-based interface allowing users to manually insert trip details, defining the start/end points, street names, and path status.
  - **Active Tracking Interface**: A dedicated screen for the "Automatic Mode". It displays real-time metrics (speed, duration) and provides controls to Start, Stop, and Pause the recording. It is designed for minimal interaction to ensure safety while riding.
  - **Post-Ride Validation Section**: A crucial interface presented at the end of an automatically tracked trip. It lists the obstacles and anomalies detected by the sensors, requiring the user to confirm or discard them before saving the trip.

3.1.2 Hardware Interfaces

The system does not require any dedicated hardware. Basic platform functionalities can be accessed from any device with an Internet connection and a compatible web browser or mobile application. Advanced features, such as automated tracking of trips, are available when using a mobile device equipped with Internet connectivity and built-in sensors, including GPS, gyroscope and accelerometer. No additional external devices or proprietary hardware interfaces are needed.

3.1.3 Software Interfaces

The platform relies on external software services to provide core functionality. In particular, it must integrate a map provider to render the map and display bike paths. In addition, it must acquire weather data from an external API to retrieve meteorological information associated with specific locations and paths, so that such data can be shown within the user interface.

3.1.4 Communication Interfaces

Client applications must have access to the Internet, either through a fixed network connection or a mobile data network. Communication between client and server for all data exchange is carried out over HTTPS, ensuring confidentiality and integrity of the transmitted information. Additional application-level communication (such as third-party map and weather services) also relies on secure HTTP-based APIs.

3.2 Functional Requirements

3.2.1 Requirements List

Requirements for Goal 1

[**R1.1**] The system allows unregistered users to sign up to the platform by providing a valid email address, a password, and their personal details (name, surname).

[**R1.2**] When a user registers, the system verifies that the provided email address is not already associated with an existing account.

[**R1.3**] The system allows registered users to log in by providing their registration email and password.

[**R1.4**] The system allows registered users to modify their profile information (name, surname) and change their password.

*To satisfy **G1**, these domain assumptions must hold:*

[**D5**] Users have access to an internet connection for map visualization, path searching, and data synchronization.

Requirements for Goal 2

[**R2.1**] The system allows registered users to manually create a new trip by specifying the sequence of street names, the starting point, and the destination.

[**R2.2**] The system allows registered users to assign a specific status (e.g., Optimal, Good, Sufficient, Unsuitable) to a manually inserted path or segment.

[**R2.3**] The system allows registered users to add obstacles information to a manually created trip.

[**R2.4**] The system allows registered users to save a new trip into their personal trip history.

[**R2.5**] The system allows registered users to delete a previously saved trip from their personal collection.

*To satisfy **G2**, these domain assumptions must hold:*

[**D4**] Users act in good faith when uploading a path in manual mode and when validating automatically detected obstacles, confirming only real issues and discarding false positives.

[**D5**] Users have access to an internet connection for map visualization, path searching, and data synchronization.

Requirements for Goal 3

[**R3.1**] The system automatically queries the external Weather Service API upon saving of a trip.

[**R3.2**] The system retrieves the following meteorological data based on the trip's location and timestamp.

[**R3.3**] The system associates the retrieved weather data with the specific trip, making it visible in the trip details.

[**R3.4**] If the external weather service is unavailable or unreachable, the system saves the trip without meteorological data without interrupting the saving process.

[**R3.5**] The system does not allow users to manually modify the automatically retrieved weather data, ensuring the objective consistency of the environmental conditions recorded.

*To satisfy **G3**, these domain assumptions must hold:*

[**D6**] The external services integrated into the system provide accurate, truthful, and up-to-date information.

Requirements for Goal 4

[**R4.1**] The system shall sample the user's position at regular intervals during the Automated Tracking Mode to reconstruct the path.

[**R4.2**] The system shall collect continuous data streams from the mobile device's accelerometer and gyroscope sensors while tracking is active.

[**R4.3**] The system shall analyze the sensor data to identify road obstacles.

[**R4.4**] When a potential obstacle is detected via sensor analysis, the system shall automatically save it for a user review, tagging it with the current GPS coordinates and timestamp.

[**R4.5**] The system shall execute data acquisition processes in the background, ensuring tracking continues even if the device screen is locked or the application is minimized.

[**R4.6**] The system shall calculate instantaneous metrics based on the distance between consecutive GPS points and the time elapsed.

*To satisfy **G4**, these domain assumptions must hold:*

[**D1**] Users possess a mobile device equipped with functioning GPS, accelerometer, and gyroscope sensors.

[**D2**] During the Automatic Tracking Mode, the mobile device is securely mounted to the bicycle (e.g., on the handlebars) to ensure sensor data reflects road conditions rather than body movements.

[**D3**] The GPS signal is available and has sufficient accuracy in the areas where the cyclist rides to allow for correct path reconstruction.

Requirements for Goal 5

[**R5.1**] The system displays a "Post-Ride Review" summary to the registered user immediately after stopping the automatic tracking, listing all the obstacles detected during the trip.

[**R5.2**] The system allows the user to confirm a detected obstacle, permanently associating it with the recorded trip.

[**R5.3**] The system allows the user to discard a detected obstacle as a "false positive", removing it from the final trip data.

[**R5.4**] The system allows registered users to modify the type of a detected obstacle if the automatic classification is incorrect.

[**R5.5**] The system allows registered users to visualize the location of each detected obstacle on the map during the review phase to aid in the verification process.

*To satisfy **G5**, these domain assumptions must hold:*

[**D4**] Users act in good faith when uploading a path in manual mode and when validating automatically detected obstacles, confirming only real issues and discarding false positives.

[**D5**] Users have access to an internet connection for map visualization, path searching, and data synchronization.

Requirements for Goal 6

[**R6.1**] The system allows registered users to change the visibility status of a saved trip from "Private" to "Public".

[**R6.2**] The system allows registered users to change the visibility status of a previously published trip from "Public" back to "Private", removing it from the global map and search results.

[**R6.3**] Upon the publication of a trip, the system automatically triggers the Scoring Engine to incorporate path segments into the global dataset.

*To satisfy **G6**, these domain assumptions must hold:*

[**D5**] Users have access to an internet connection for map visualization, path searching, and data synchronization.

Requirements for Goal 7

[**R7.1**] The system allows any user (registered or unregistered) to search for bike paths by specifying an origin and a destination via text input (addresses) or by selecting points directly on the map.

[**R7.2**] The system displays the available paths connecting the specified origin and destination on an interactive map.

[**R7.3**] The system visualizes the Path Score and Status of the retrieved paths.

[**R7.4**] The system allows users to select a specific path from the search results to view its detailed information.

[**R7.5**] The system orders the search results by default based on their Path Score, presenting the best-rated path first.

*To satisfy **G7**, these domain assumptions must hold:*

[**D5**] Users have access to an internet connection for map visualization, path searching, and data synchronization.

[**D6**] The external services integrated into the system provide accurate, truthful, and up-to-date information.

Requirements for Goal 8

[**R8.1**] The system shall automatically compute a numerical Path Score for every published trip stored in the database.

[**R8.2**] The scoring algorithm shall calculate the score as a weighted function of the Path Status of the composing segments and the presence of obstacles.

[**R8.3**] The system shall automatically trigger a recalculation of the Path Score for a path whenever the status of its underlying segments is modified or updated, ensuring the score always reflects latest informations.

[**R8.4**] The system shall utilize the computed Path Score to rank the results of a path search, displaying the highest-scoring paths at the top of the list.

*To satisfy **G8**, these domain assumptions must hold:*

[**D6**] The external services integrated into the system provide accurate, truthful, and up-to-date information.

Requirements for Goal 9

[**R9.1**] The system shall identify geographically overlapping path segments from different published trips by analyzing their GPS coordinates to determine which user submissions refer to the same physical road section.

[**R9.2**] The Scoring Engine shall calculate the consolidated Path Status for a specific road section by applying a majority consensus algorithm to the overlapping segments submitted by different users.

[**R9.3**] The merging algorithm shall assign a weight to each user submission based on its Data Freshness. Recent reports must have a significantly higher influence on the than older reports.

[**R9.4**] The system shall retain the history of merged segments to allow for the re-evaluation of the status if new data arrives.

*To satisfy **G9**, these domain assumptions must hold:*

[**D4**] Users act in good faith when uploading a path in manual mode and when validating automatically detected obstacles, confirming only real issues and discarding false positives.

3.3 Use Case Diagram

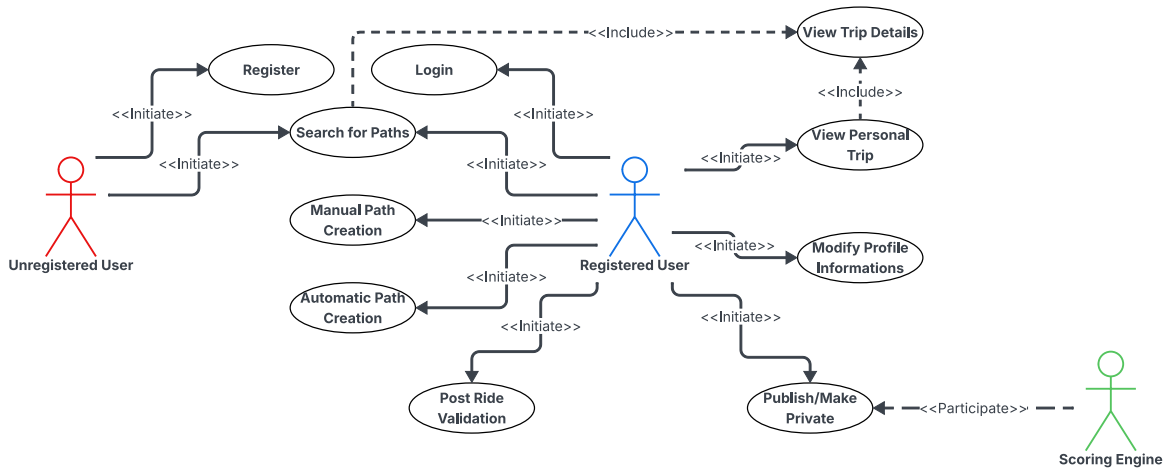

Figure 4: Use case diagram

3.4 Use-Cases

3.4.1 UC1 - User Registration

Name	**User Registration**
Actors	Unregistered User
Entry Conditions	The user has accessed the BBP application and is currently on the landing page or login screen.
Flow of Events	1. The User clicks on the "Sign Up" button. 2. The System displays the registration form requesting personal details and credentials. 3. The User fills in the required information. 4. The User submits the form. 5. The System verifies that the email address is not already associated with an existing account. 6. The System confirms the registration and logs the user in automatically.
Exit Conditions	The User is successfully registered as a "Registered Cyclist" and gets redirected to the homepage.
Exceptions	1. **Email already in use:** If the email provided is already in the database, the System displays an error message and asks the User to use a different email or log in.

Table 1: Use Case: User Registration

3.4.2 UC2 - User Login

Name	**User Login**
Actors	Registered Cyclist
Entry Conditions	The user has accessed the BBP application and is currently on the landing page or login screen.
Flow of Events	1. The User clicks on the "Login" button. 2. The System displays the login form requesting the registration email and password. 3. The User enters their credentials. 4. The User clicks the "Login" button to submit the form. 5. The System validates the credentials.
Exit Conditions	The User is authenticated and is redirected to the homepage.
Exceptions	1. **Invalid Credentials:** If the email or password does not match any record in the database, the System displays an error message ("Invalid email or password") and prompts the User to retry.

Table 2: Use Case: User Login

3.4.3 UC3 - View Personal Trips

Name	**View Personal Trips**
Actors	Registered Cyclist
Entry Conditions	The User is logged into the platform and is currently on the main dashboard.
Flow of Events	1. The User navigates to the "My Trips" section. 2. The System retrieves the list of trips associated with the user's account from the database. 3. The System displays the list of recorded trips, showing summary information for each (e.g., Date, Duration, Distance, Status).
Exit Conditions	The User successfully visualizes all his personal trips.
Exceptions	1. **No Trips Found:** If the user has not recorded any trips yet, the System displays a "No trip found" message.

Table 3: Use Case: View Personal Trip History

3.4.4 UC4 - View Trip Details

Name	**View Trip Details**
Actors	Registered Cyclist (or Unregistered User for public trips)
Entry Conditions	The User is viewing a list of trips and selects a specific trip to inspect.
Flow of Events	1. The User clicks on the desired trip. 2. The System retrieves all data associated with the requested trip, including performance metrics, path coordinates, meteorological data and obstacles. 3. The System renders the "Trip Detail" view, displaying the interactive map with the path overlay, the obstacles markers, and the statistics panel.
Exit Conditions	The User is presented with detailed view of the selected trip.
Exceptions	1. **Loading Error:** If the system fails to retrieve track data, an error message is displayed, and the user remains on the list view.

Table 4: Use Case: View Trip Details

3.4.5 UC5 - Modify Profile Informations

Name	**Modify Profile Informations**
Actors	Registered Cyclist
Entry Conditions	The User is logged into the platform and is currently on the main dashboard.
Flow of Events	1. The User selects the option to edit the profile details. 2. The System displays the current information in editable fields. 3. The User updates informations. 4. The User clicks the “Save Changes” button. 5. The System validates the new data and displays a success message.
Exit Conditions	The User’s profile data is successfully updated in the system.
Exceptions	1. **Validation Failed:** If the input data is invalid, the System displays an error message and prompts the User to correct it.

Table 5: Use Case: Modify Profile Informations

3.4.6 UC6 - Search for Paths

Name	**Search for Paths**
Actors	Unregistered/Registered Cyclist, Map Service Provider
Entry Conditions	The User has accessed the application and is on the Landing Page.
Flow of Events	1. The User specifies the origin and destination by entering addresses or selecting points on the map and click the "Search" button. 2. The System interacts with the Map Service Provider to identify the locations. 3. The System retrieves available bike paths connecting the two points and their scores. 4. The System displays the results on the interactive map.
Exit Conditions	The User visualizes the available paths on the map.
Exceptions	1. **Map Service Provider Offline:** If the necessary map data cannot be retrieved from the external provider, the System displays an error message indicating that the search service is temporarily unavailable. 2. **Locations Not Found:** If the entered addresses cannot be identified, the System displays an error message requesting valid locations. 3. **No Paths Found:** If no bike paths connect the origin and destination, the System informs the User that no paths are available.

Table 6: Use Case: Search for Paths

3.4.7 UC7 - Manual Trip Creation

Name	**Manual Trip Creation**
Actors	Registered Cyclist, Weather Service
Entry Conditions	The User is logged into the BBP platform and navigates to the "Create New Trip" section.
Flow of Events	1. The System displays a map interface with a form for path details. 2. The User specifies the path by entering the sequence of street names or by drawing the path segments on the interactive map. 3. The User assigns a specific status to the trip 4. *(Optional)* The User modify the status of a single path segment. 5. *(Optional)* The User inserts specific obstacles at precise locations along the path. 6. *(Optional)* The User sets the trip visibility to 'Public'. 7. The User clicks the "Save Trip" button. 8. The System validates path data. 9. The System queries the external *Weather Service* to retrieve the meteorological conditions for the specified location and time. 10. The System saves the tirp into the User's personal collection ("My Trip"). 11. The user is redirected to the main dashboard
Exit Conditions	The trip is successfully saved in the user's history.
Exceptions	1. **Unrecognized Location:** If the User types a street name that the map service cannot resolve, the System suggests valid alternatives. 2. **Weather Service Offline:** If the weather data cannot be retrieved, the trip is saved without meteorological information.

Table 7: Use Case: Manual Trip Creation

3.4.8 UC8 - Automatic Trip Creation

Name	**Automatic Trip Creation**
Actors	Registered Cyclist, Weather Service
Entry Conditions	The User is logged into the BBP platform, has enabled location services on their device, and is ready to start a ride.
Flow of Events	1. The User taps the "Start Tracking" button. 2. The System begins recording the GPS coordinates to reconstruct the path in real-time. 3. The System continuously samples data from the device's sensors. 4. The System monitors the travel speed to verify consistency with cycling activity. 5. The User completes the ride and taps the "Stop Tracking" button. 6. The System queries the external *Weather Service* to retrieve the meteorological conditions for the current location and time. 7. The System saves the raw trip data, including the path, detected anomalies, and weather info, as a "Draft" trip. 8. The System prompts the User to proceed to the Validation phase.
Exit Conditions	A raw trip is successfully recorded and stored locally or on the server, ready to be validated by the user.
Exceptions	1. **Sensor Failure:** If the System cannot access the necessary sensors, it displays an error message and prevents the tracking from starting. 2. **Activity Mismatch:** If the System detects a speed consistent with driving or walking rather than cycling for the entire duration, the activity is paused automatically. 3. **Weather Service Offline:** If the weather data cannot be retrieved, the trip is saved without meteorological information.

Table 8: Use Case: Automatic Trip Creation

3.4.9 UC9 - Post Ride Validation

Name	**Post Ride Validation**
Actors	Registered Cyclist
Entry Conditions	The User has just stopped an Automatic Tracking session.
Flow of Events	1. The System displays the "Post-Ride Review" screen, showing the trip summary and the list of detected potential obstacles. 2. The User selects an obstacle to visualize its specific location on the map. 3. At this point the User can confirm the obstacle as real, modify it or discard due to incorrect measurement. 4. The User repeats the validation for the remaining detected obstacles. 5. The User assigns a specific status to the trip 6. *(Optional)* The User modify the status of a single path segment. 7. *(Optional)* The User sets the trip visibility to 'Public'. 8. The User clicks the "Save Trip" button. 9. The System saves the trip.
Exit Conditions	The trip is successfully saved in the user's history with validated data enriched by the user.
Exceptions	1. **Discard Entire Trip:** The User decides not to save the trip at all. The System requests confirmation and then permanently deletes all temporary data recorded during the session.

Table 9: Use Case: Post Ride Validation

3.4.10 UC10 - Change Trip Visibility

Name	**Change Trip Visibility**
Actors	Registered Cyclist, Scoring Engine
Entry Conditions	The User has selected a specific private or public trip. This can occur in two scenarios: • **Instant:** Immediately after saving a new trip (Manual or Automatic). • **Late:** By navigating to the details of an existing trip in the "My Trip" section.
Flow of Events	1. **Navigation (Late Scenario Only):** • The User navigates to the list of saved trips. • The System displays the trips with their current status (Private/Public). • The User locates and selects the specific trip. 2. The User toggles the visibility switch. 3. **Case A: Publishing (Private → Public)** • The Scoring Engine identifies overlapping segments with the global dataset. • The Scoring Engine merges the user's data with existing paths, updating the reliability weight. • The Scoring Engine recalculates the Path Score based on the new input. • The System sets the trip status to "Public". 4. **Case B: Unpublishing (Public → Private)** • The Scoring Engine removes the path's contribution from the global map aggregation. • The Scoring Engine triggers a recalculation of the Path Score for the affected segments. • The System sets the trip status back to "Private". 5. The System updates the interface to reflect the new visibility status.
Exit Conditions	The visibility status is updated, and the global path data is adjusted accordingly.
Exceptions	1. **System Busy:** If the Scoring Engine is overloaded, the System queues the update and notifies the User that the trip will appear public shortly.

Table 10: Use Case: Change Trip Visibility

3.5 Sequence Diagram

3.5.1 [UC1] - User Registration

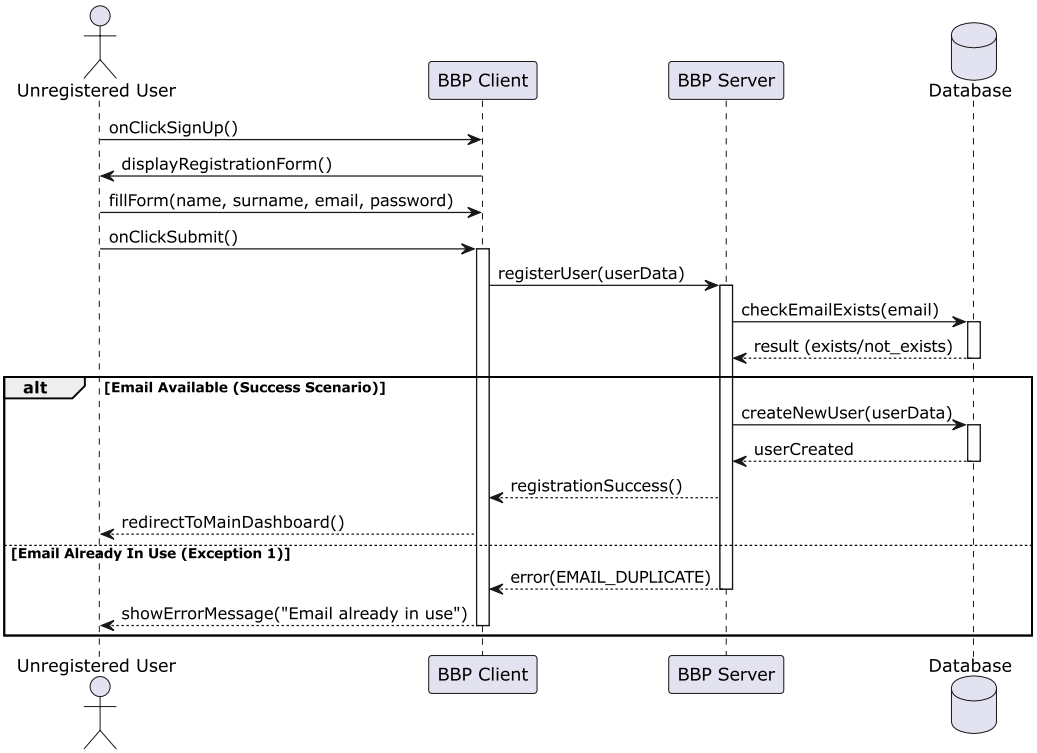

Figure 5: UC1 diagram

3.5.2 [UC2] - User Login

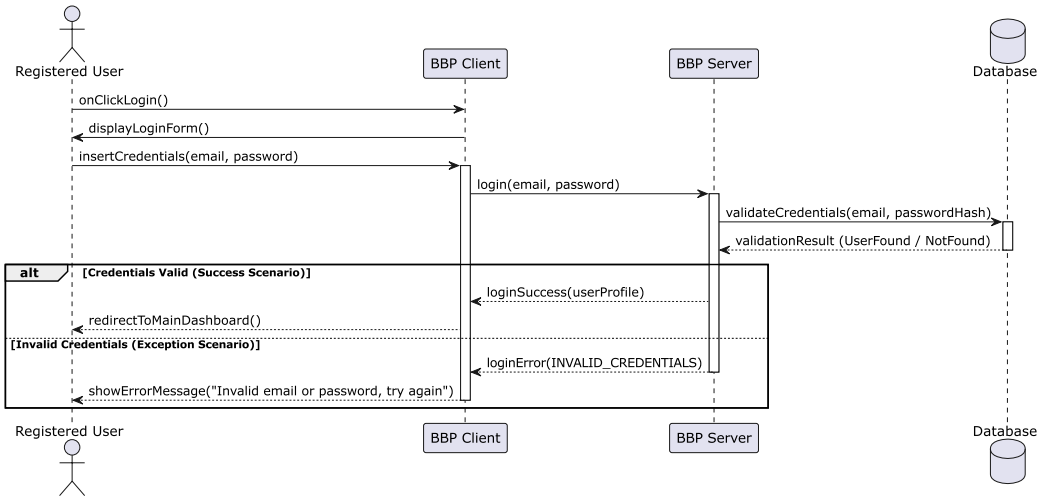

Figure 6: UC2 diagram

3.5.3 [UC3] - View Personal Trips

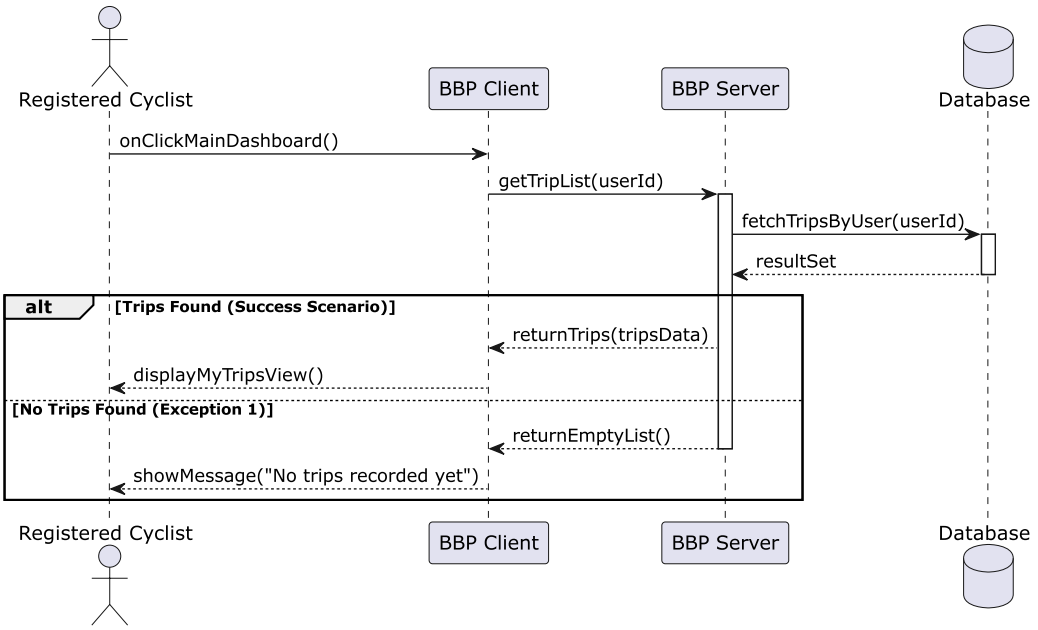

Figure 7: UC3 diagram

3.5.4 [UC4] - View Trip Details

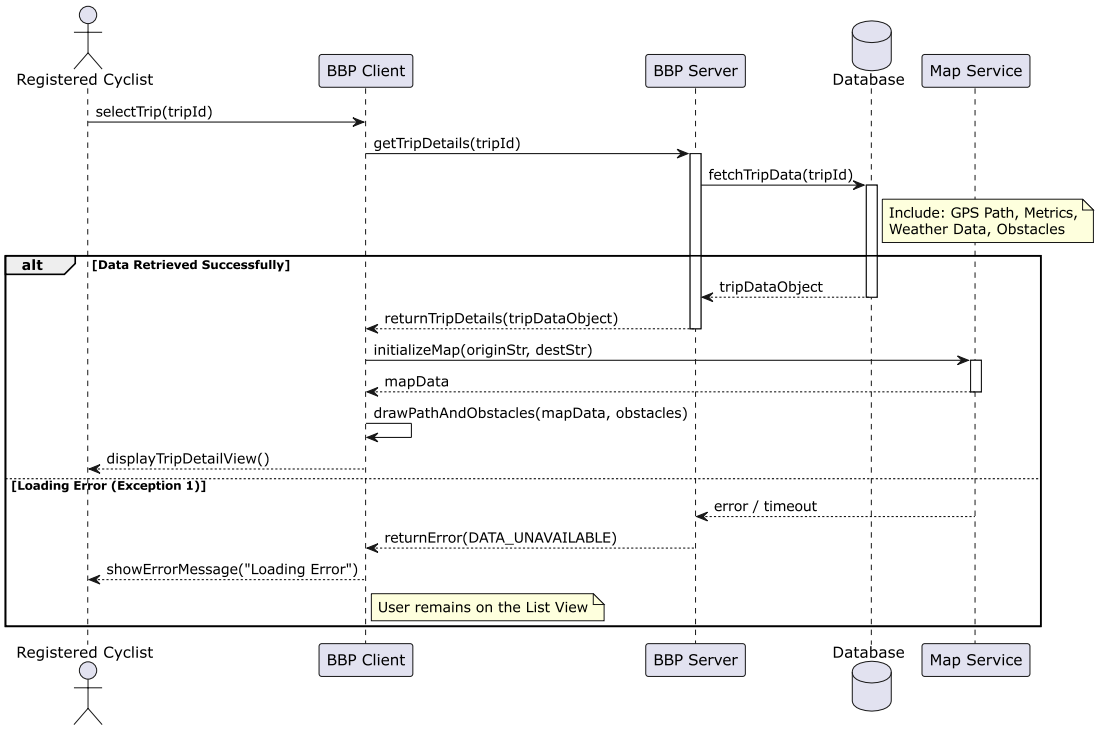

Figure 8: UC4 diagram

3.5.5 [UC5] - Modify Profile Informations

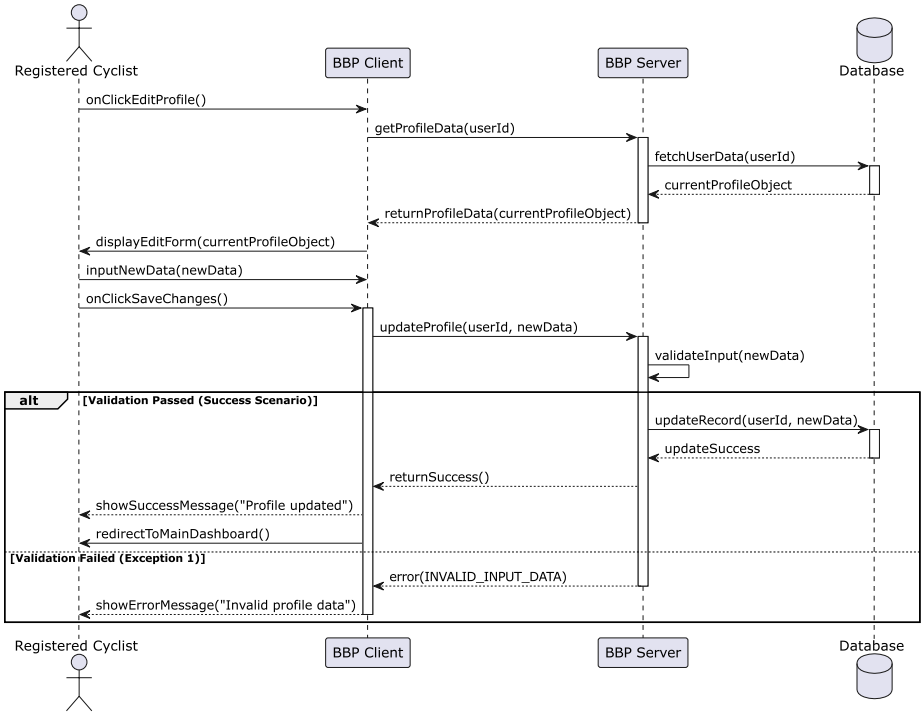

Figure 9: UC5 diagram

3.5.6 [UC6] - Search for Paths

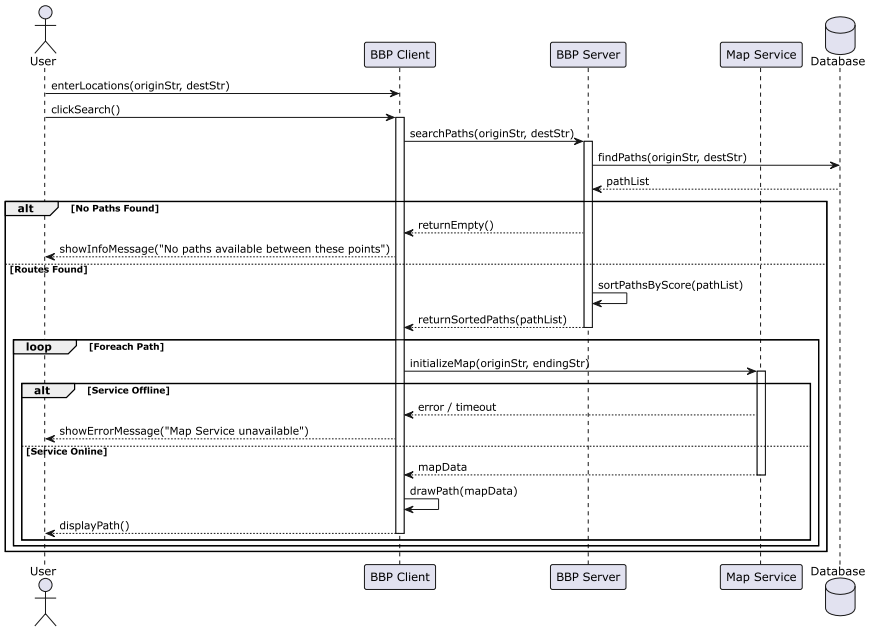

Figure 10: UC6 diagram

3.5.7 [UC7] - Manual Trip Creation

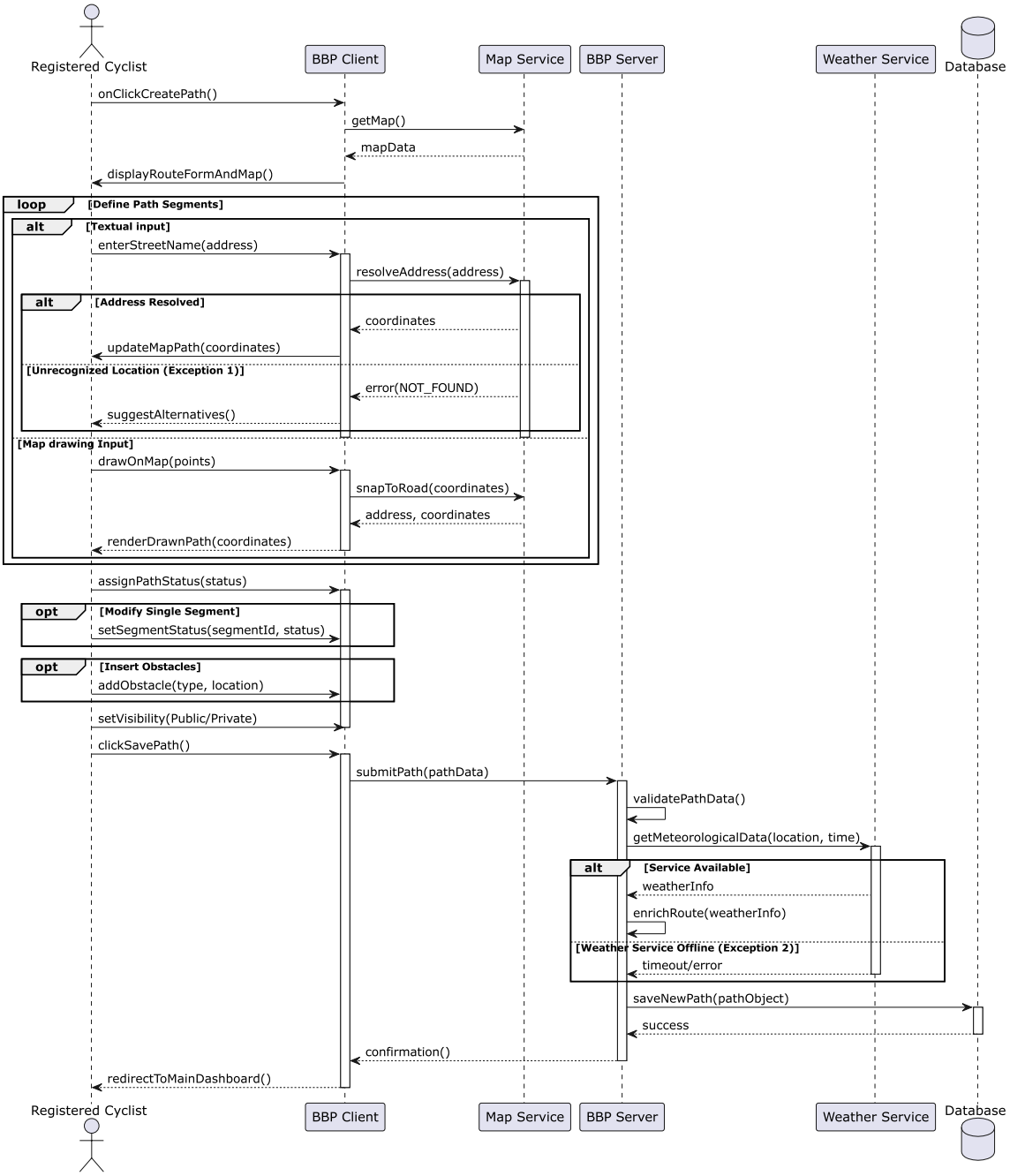

Figure 11: UC7 diagram

3.5.8 [UC8] - Automatic Path Creation

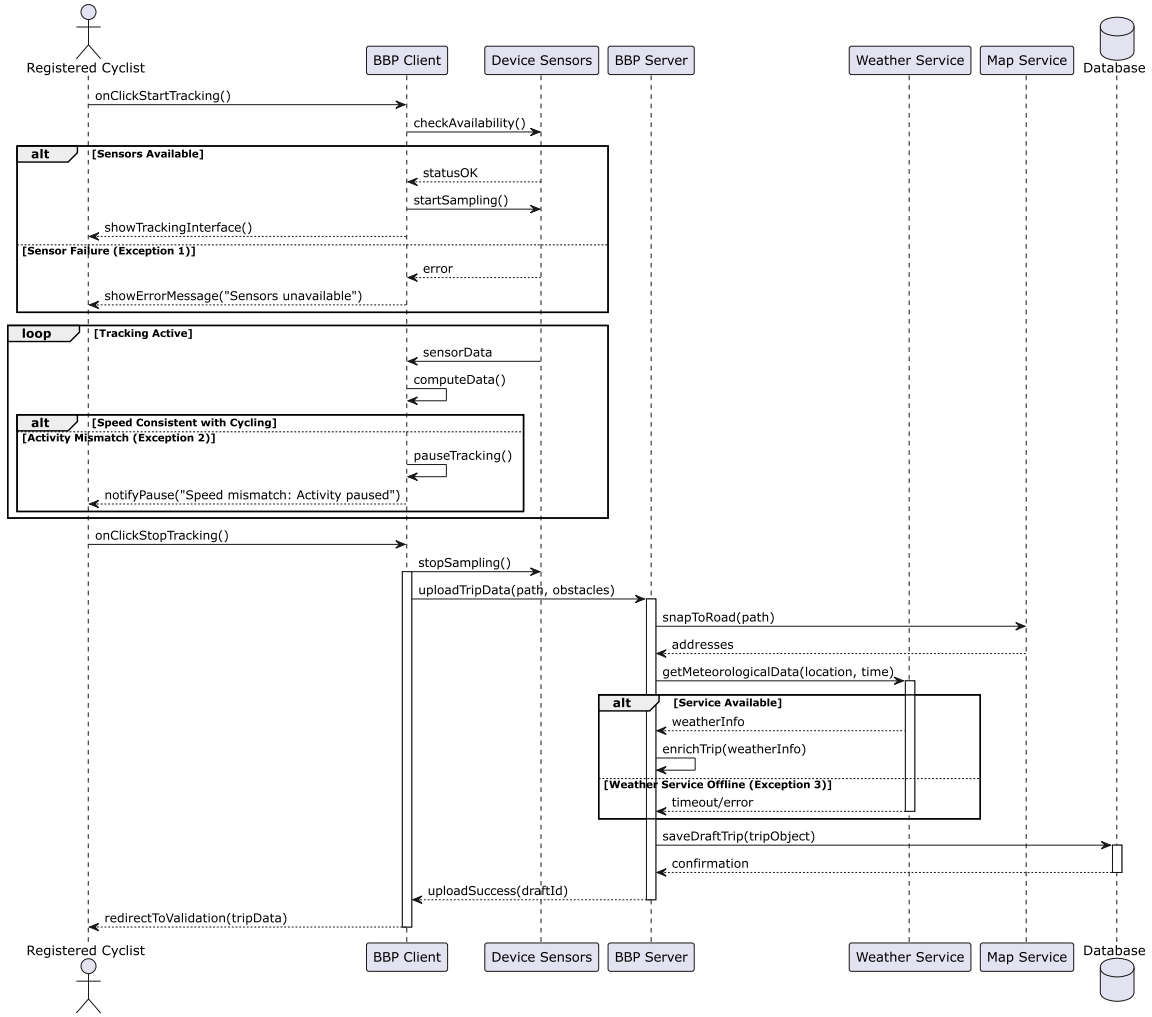

Figure 12: UC8 diagram

3.5.9 [UC9] - Post Ride Validation

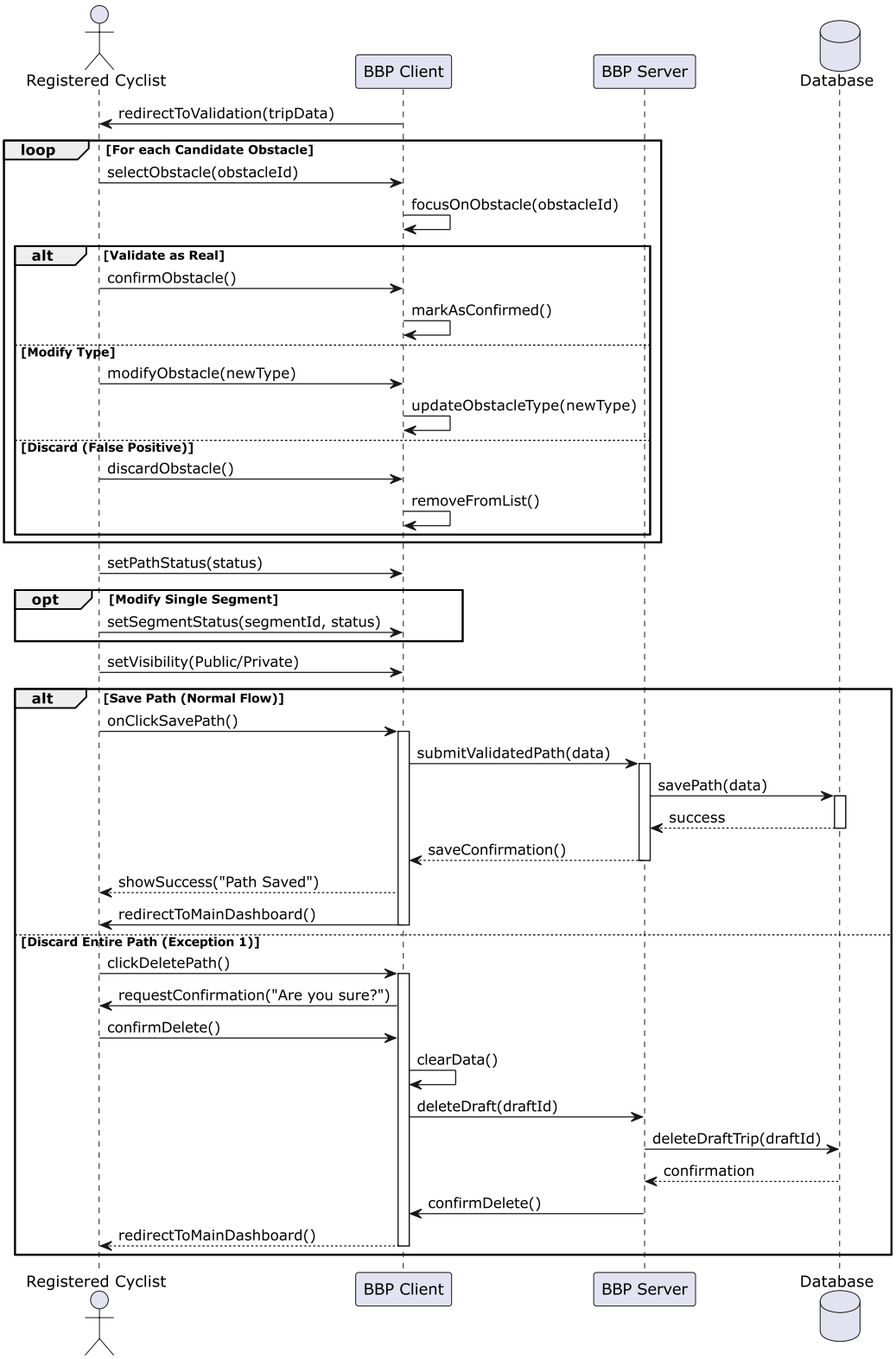

Figure 13: UC9 diagram

3.5.10 [UC10] - Change Path Visibility

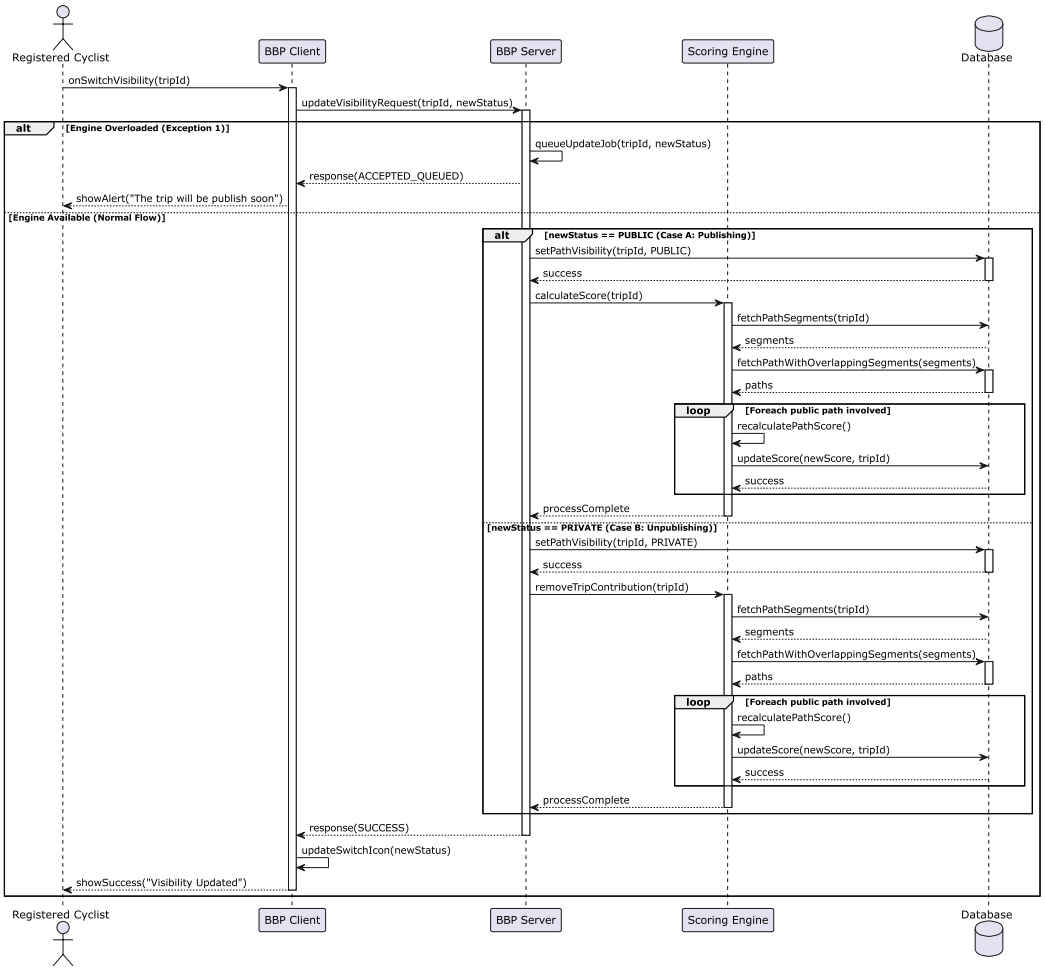

Figure 14: UC10 diagram

3.6 Performance Requirements

The system must meet specific performance metrics to ensure a satisfactory user experience:

Search Response Time: The Search for Paths function *(UC6)* shall return results and render them on the map within 3 seconds.

Sensor Sampling Rate: To effectively detect road irregularities, the mobile application must sample sensors data at a sufficient frequency without degrading the device's UI responsiveness.

Publishing: The Scoring Engine must process the publication of a path *(UC10)* asynchronously. While the user must receive immediate feedback that the request

was queued, the uploaded trip and new scores for all trips involved must be available within 5 minutes of the submission.

Storage Efficiency: The mobile application must optimize the local storage of "Draft" paths. A long trip recording should not exceed 50 MB of local storage space.

3.7 Design Constraints

3.7.1 Standards Compliance

GDPR Compliance (Regulation EU 2016/679): BBP collects and stores sensitive personal data, specifically the Users' location history and movement patterns; Because of this the system must strictly adhere to the General Data Protection Regulation. This implies:

- Users must explicitly consent to GPS data collection.
- Users have the right to request the complete deletion of their account and associated data.
- Personal data must be stored securely and anonymized where possible.

3.7.2 Hardware limitations

Internet Connectivity: An active internet connection is a mandatory constraint for path searching, path uploading, Publishing/Unpublishing, login and registration as they require real-time communication with the BBP Server or external Map/Weather Providers.

Offline buffering: The system must be designed to handle intermittent network connectivity. The Recording and Validation phases must be fully functional offline, buffering data locally until a connection is re-established for the upload.

Battery Consumption: The application tracking algorithm must be optimized for energy efficiency.

Sensor Availability: The Automatic Path Creation feature is constrained by the physical presence of hardware sensors. This functionality cannot function on devices lacking a GPS receiver, accelerometer, or gyroscope.

3.8 Software System Attributes

3.8.1 Reliability

In the event of software crash or battery failure during a ride, the system must automatically recover the tracking data up to the last saved checkpoint upon the next restart, preventing the loss of an entire trip. Also tracking and Path rendering should be as accurate as possible.

3.8.2 Availability

The BBP Server and Database shall be available 99.9% of the time during operating hours (24/7), excluding scheduled maintenance windows. If external services (Map or Weather providers) are unavailable, the system must continue to operate with limited functionalities rather than crashing or blocking the user.

3.8.3 Security

Communication Security: All data in transit between the Client and Server must be encrypted via HTTPS.

Password Storage: User passwords must never be stored in plain text. They must be hashed and salted using industry-standard algorithms.

Private Trip Privacy: Trips saved as "Private" must be accessible strictly by the creator.

3.8.4 Maintainability

The BBP platform shall be designed using a modular architecture. Core functionalities must be implemented as distinct, loosely coupled modules. This compartmentalization ensures that each component can be developed, tested, and maintained independently. Consequently, updates or bug fixes in one specific module will not negatively impact the stability or functionality of other parts of the system.

3.8.5 Portability

The system is designed to be deployed as both a Web Application and a Mobile App to ensure broad accessibility. However, hardware dependent features like Automated Tracking are exclusively available on the mobile platform due to the strict requirement for onboard sensors.

4 Formal Analysis using Alloy

4.1 Scoring and Aggregation Analysis

This section details the algorithmic logic used to evaluate bike paths (*Trips*).

The system persists user data using a "Snapshot Pattern": for every trip, the system stores individual instances of the traversed segments. To determine the quality of a physical road, the system aggregates all the stored snapshots referring to that location.

In this model, time is excluded from the consensus function used for data merging. Its inclusion would introduce unnecessary complexity without impacting the formal verification results.

4.1.1 Alloy Specification

These are the signatures for the model. *RoadSection* corresponds to the physical location of the segment, it functions as an anchor, it does not contain any data. SegmentSnapshot represents an instance of a single road section generated when a user uploads a new trip. Trip signature is a collection of segments that form a path.

```
module BBP_Snapshot_Merging
open util/integer

-- [SIGNATURES]

sig RoadSection {}

enum Quality { Unsuitable, Poor, Sufficient, Good, Optimal }

sig SegmentSnapshot {
    location: one RoadSection,
    quality: one Quality
}

sig Trip {
    recordedSegments: some SegmentSnapshot,
    currentScore: Int
}

-- [HELPER FUNCTIONS]
```

```
-- Maps the Enum to a calculable Integer (0 to 4)
fun qualityToInt[q: Quality]: Int {
    q = Unsuitable implies 0 else
    q = Poor       implies 1 else
    q = Sufficient implies 2 else
    q = Good       implies 3 else
    4
}
```

The following facts define the model constraints:

```
-- [FACTS] Structural Constraints

-- A snapshot must belongs to exactly one trip
fact SnapshotsBelongToTrips {
    all s: SegmentSnapshot | one t: Trip | s in t.
       recordedSegments
}

-- A trip does not contain two snapshot of the same road
   section
fact UniqueRoadPerTrip {
    all t: Trip | no disj s1, s2: t.recordedSegments | s1.
       location = s2.location
}

fun getMergedRoadStatus[r: RoadSection]: Int {
    let snapshots = location.r |

    #snapshots = 0 implies 0 else {
        let consensusQualities = { q: Quality |
            -- Take the resulting quality based on the
               consensus
            all other: Quality |
            #{s: snapshots | s.quality = q} >= #{s: snapshots |
                s.quality = other}
        } |
        -- Since time is excluded, the max function is used to
           break ties. In a practical scenario, this would be
           handled via data freshness.
        max[{ val: Int | some q: consensusQualities | val =
```

```
            qualityToInt[q] }]
    }
}

fact DefineTripScore {
    all t: Trip |
    let segmentsCount = #t.recordedSegments |
    -- Retrieves the consensus scores from all segments within
       the trip and sums them to calculate the total
    let sumOfSegments = sum s: t.recordedSegments |
       getMergedRoadStatus[s.location] |

    -- Avoid division by zero
    segmentsCount > 0 implies
        -- The total trip score is calculated as the sum of the
            segment scores divided by the number of segments
        t.currentScore = div[sumOfSegments, segmentsCount]
    else
        t.currentScore = 0
}
```

4.1.2 Scenario 1

The following simulation, `QualityDegradationScenario`, is designed to stress-test the data merging logic under conflicting conditions. It postulates a specific instance where a single Optimal report is countered by two subsequent Unsuitable reports on the same road section. The goal is to verify if the consensus mechanism correctly prioritizes the majority, effectively downgrading the merged road status to Unsuitable

```
-- [ASSERTIONS & SIMULATION]

pred QualityDegradationScenario {
    some disj tGood, tBad1, tBad2: Trip | some road:
       RoadSection | {

        -- All trips share the same road
        -- Initially there is an optimal score
        some sGood: tGood.recordedSegments | {
            sGood.location = road
```

```
            sGood.quality = Optimal
        }

        -- tGood is composed of only one segment, so it's
           initial score is optimal
        #tGood.recordedSegments = 1

        -- Two unsuitable score are assigned later
        some sBad1: tBad1.recordedSegments | {
            sBad1.location = road
            sBad1.quality = Unsuitable
        }
        some sBad2: tBad2.recordedSegments | {
            sBad2.location = road
            sBad2.quality = Unsuitable
        }

        -- The new score given by the merging of these 3
           snapshot should be unsuitable
        getMergedRoadStatus[road] = qualityToInt[Unsuitable]

        -- The new score of tGood has to be degradated by the
           new data
        tGood.currentScore < qualityToInt[Optimal]
    }
}

run QualityDegradationScenario for 5
```

4.1.3 Analysis Results

Executing the predicate using the Alloy solver yielded a positive result, confirming that the described scenario is logically consistent with the model's structural constraints. No formal contradictions were detected.

Figure 15 illustrates the graphical representation of the generated instance, highlighting the relationships between the three trips and the contested road section.

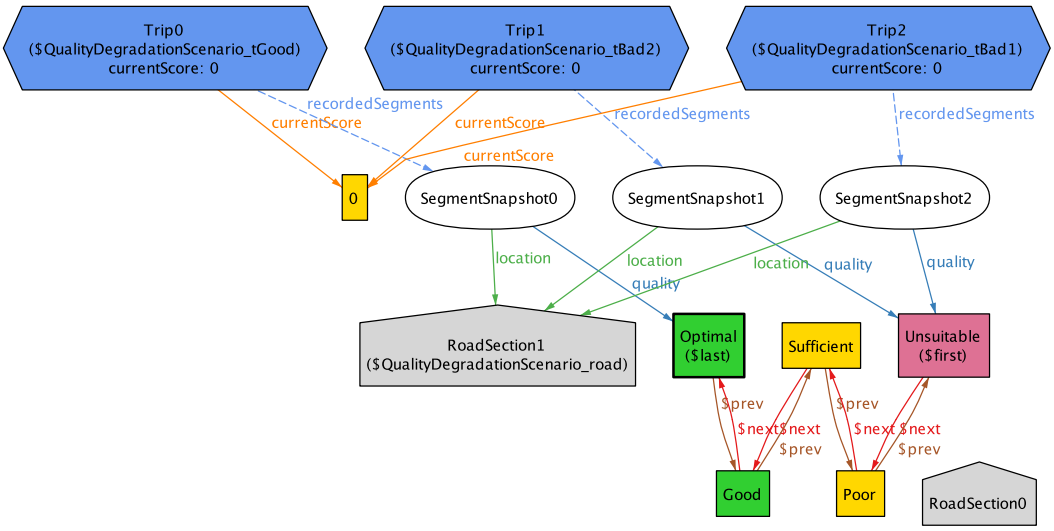

Figure 15: Scenario 1 Run

4.1.4 Scenario 2

The following simulation, `MajorityOverrideScenario`, evaluates the robustness of the trip scoring algorithm in a multi-segment context. It postulates a scenario where a user's positive subjective experience (a trip consisting of three "Good" segments) is partially contradicted by a negative majority consensus on specific road sections. The objective is to verify that the individual's final trip score is derived from the merged community data rather than their raw local input, demonstrating the system's ability to prioritize collective knowledge over individual outliers.

```
pred MajorityOverrideScenario {
    -- 3 distinct segments and 3 trips (1 focus + 2 others)
    some disj r1, r2, r3: RoadSection, disj tFocus, tOther1,
       tOther2: Trip | {

        -- tFocus trip
        -- This user traverses 3 segments, rating all of them "
           Good" (3).
        -- In isolation, this trip's score would be 3.
        #tFocus.recordedSegments = 3
        some s1, s2, s3: tFocus.recordedSegments | {
            s1 != s2 and s1 != s3 and s2 != s3

            s1.location = r1 and s1.quality = Good
            s2.location = r2 and s2.quality = Good
```

```
    s3.location = r3 and s3.quality = Good
}

-- tOther trip
-- On R1 and R3, two other users report "Unsuitable"
   (0).
-- Result for R1 and R3: 1 "Good" vs 2 "Unsuitable".
-- Majority rule applies: R1 and R3 become "Unsuitable
   ".

some o1_r1, o1_r3: tOther1.recordedSegments | {
    o1_r1.location = r1 and o1_r1.quality = Unsuitable
    o1_r3.location = r3 and o1_r3.quality = Unsuitable
}

some o2_r1, o2_r3: tOther2.recordedSegments | {
    o2_r1.location = r1 and o2_r1.quality = Unsuitable
    o2_r3.location = r3 and o2_r3.quality = Unsuitable
}

-- On R2, no one votes against, so tFocus's "Good"
   rating remains valid

-- Merged Status Verification
-- Verify that the system has correctly adopted the
   majority consensus
getMergedRoadStatus[r1] = qualityToInt[Unsuitable] --
   (0)
getMergedRoadStatus[r2] = qualityToInt[Good]       --
   (3)
getMergedRoadStatus[r3] = qualityToInt[Unsuitable] --
   (0)

-- Final score calculation for tFocus
-- Sum of merged scores = 0 (r1) + 3 (r2) + 0 (r3) = 3
-- Number of segments = 3
-- Average = 3 / 3 = 1
-- 1 corresponds to 'Poor'
```

```
        -- The system assigns 'Poor' to the trip, even though
           the user input was entirely 'Good'.
        tFocus.currentScore = qualityToInt[Poor]
    }
}

run MajorityOverrideScenario for 5 but 10 SegmentSnapshot, 6
    int
```

4.1.5 Analysis Results

Executing the predicate using the Alloy solver yielded a positive result, confirming that the multi-segment consensus logic is consistent. The generated instance validates that the system correctly overrides the user's optimistic input (Good) when the community consensus indicates a degradation (Unsuitable) on the majority of the traversed path.Specifically, the solver confirmed that the final score calculation correctly averaged the merged road statuses $(0, 3, 0)$, resulting in a final score of 1 (Poor), rather than the 3 (Good) that would have resulted from the user's isolated perception.Figure 16 illustrates the graphical representation of the generated instance, highlighting how the "Other" trips influence the status of roads $r1$ and $r3$, subsequently impacting the score of the $tFocus$ trip.

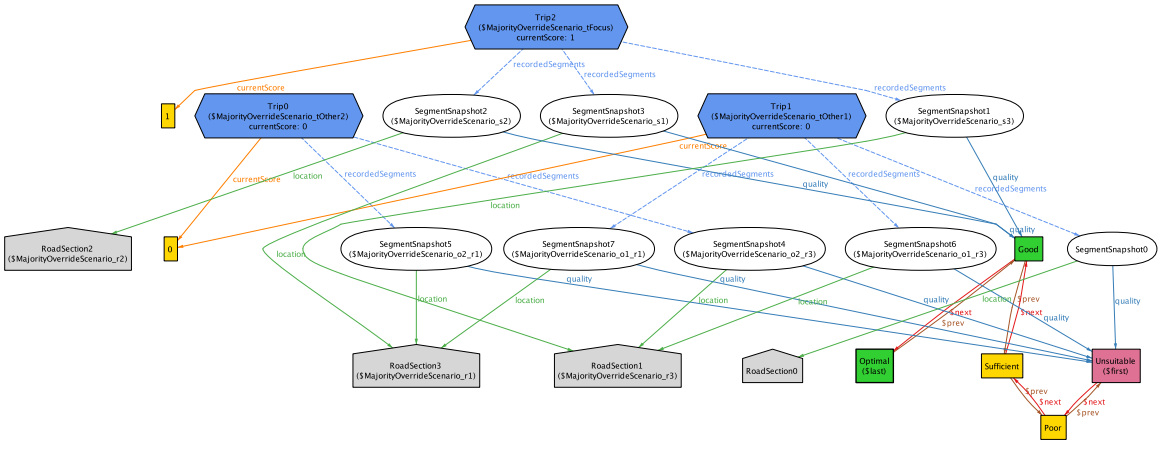

Figure 16: Scenario 2 Run

5 Effort Spent

Document Section	Giuliano Crescimbeni (h:m)	Luca De Nicola (h:m)
Introduction	3:15	3:15
Overall Description	8:30	8:30
Specific Requirements	14:45	14:45
Formal Analysis	6	6
Total Effort	**32:30**	**32:30**

Table 11: Effort spent by group members on each section of the document.

6 Software Used

Software Tool	Category	Usage in Project
GitHub	Version Control	Source code management and versioning.
Overleaf	Documentation	Collaborative LaTeX editing and compilation.
Lucidchart	Modeling	Designing State and Use Case diagrams.
PlantText	Modeling	Designing Class and UML Sequence diagrams.
Alloy Analyzer	Formal Methods	Formal specification and model verification.

Table 12: List of software tools used during development.