POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Requirement Analysis and Specification Document

Best Bike Paths

Authors: **Matteo Caldognetto – Claudia Salone**

Student IDs: 11097854 – 10827565
Academic Year: 2025-26

Contents

1 | Introduction

A. Purpose

The increase in bicycle use as a sustainable means of transport requires effective solutions to encourage more users to choose cycling, while ensuring usability and safety during bike path journeys. Best Bike Paths is designed to support these goals by offering a platform that allows users to record their trips, view bike routes, and share valuable information about the condition of paths and potential obstacles.

The application aims to create an integrated environment where cyclists, both professional and recreational, can actively contribute to the mapping and management of routes through manual entries or recording and collecting automatically their itineraries, integrating the ability to view detailed statistics for each outing, such as distance traveled and average speed.

BBP further incorporates a standardized system to suggest the best bike routes from an origin to a destination, based on reviews and scores assigned to the paths, facilitating the choice of safer and more comfortable routes. The app can also provide updated weather information, enriching the cyclist's experience.

The primary goal of BBP is therefore to promote the use of bicycles as a daily transport option, encouraging the sharing and access to updated and reliable information on bike path conditions and the discovery of new paths. In doing so, BBP aims to enhance the quality of sustainable mobility in cities, offering a useful service both to experienced cyclists and newcomers to urban cycling.

A.1. Goals

G1 : Unregistered cyclists can register into the system

G2 : Registered cyclists can log into the system

G3 : Registered cyclists can manually record and insert information about their bike paths, including their status and the presence of relevant obstacles, to keep track of their cycling activities.

G4 : Registered and unregistered cyclists can visualize on a map the bike paths between a point of origin and a point of destination

G5 : Registered cyclists can automatically record their bike paths, including their status

and the presence o relevant obstacles, to keep track of their cycling activities.

G6 : Registered cyclists can confirm, correct or delete the automatically provided feedback about a cycling trip they've just traveled.

G7 : Registered cyclists can consult their own inventory of recorded bike trips.

G8 : Registered cyclists can delete data they have already recorded in their itinerary on a specific route.

G9 : Registered cyclists can delete information they have already shared with the community on a specific route.

G10 : Registered cyclists can submit a comprehensive report on an existing bike path created by another user.

G11 : Registered cyclists can publish bike paths that were previously recorded privately.

B. Scope

Best Bike Paths (BBP) is a digital platform aimed at supporting cyclists, both professional and amateur, in recording, sharing, and consulting useful information. The main actors are the cyclists, divided into registered and unregistered users.

Registered users can use the system to record their bike itineraries and save them to monitor their cycling activities. The system provides various detailed statistics for each trip, offering concrete support for the evaluation and planning of journeys. Registered users can also input data on the condition of bike paths. These data can be entered manually, such as reports of obstacles or changes in the pathway, or collected automatically through sensors integrated into the users' mobile devices. Before the automatically collected information is published and made available to the community, it must be confirmed or corrected by the users themselves, ensuring the reliability of the reports.

All users, whether registered or not, can use a map visualization feature to identify bike routes from a starting point to a final destination. The routes are ordered and recommended according to how many obstacles they contain, when applicable, and how close they are to the two locations.

The scope of the project is limited to functionalities related to data collection, verification, publication, and consultation within the BBP platform, with the aim of improving cycling experiences, promoting mobility by bike, and enhancing road safety and comfort through active participation by the user community.

B.1. World Phenomena

WP1 : The sensors of registered cyclists' devices detect speed, route, travel time, and any sudden or irregular movements during the bike ride.

WP2 : External services provide real-time information about weather conditions of the trips.

WP3 : Registered cyclist decides to start using the bike.

WP4 : External services provide route mapping information.

WP5 : Roads may contain physical obstacles or irregularities.

B.2. Shared Phenomena

SP1 : Registered cyclists enter information about bike path conditions manually.

SP2 : Registered cyclists confirm, correct or refuse automatically acquired data about their last bike trip.

SP3 : Registered cyclists make the entered information publishable, making it accessible to the community.

SP4 : Registered and unregistered cyclists request the display on a map of bike routes between an origin and a final destination.

SP5 : The system receives information about bike paths from external sensors automatically.

SP6 : Registered cyclists record a new bike itinerary and stores it in the system to keep track of it.

SP7 : The system processes and provides detailed statistics to the user regarding their recorded bike trips.

SP8 : The device's sensors, by detecting the movement speed, update the system that a registered cyclist has started or finished using bike.

SP9 : Registered cyclists remove information already recorded on their personal itinerary.

SP10 : Registered cyclists remove data concerning a route previously shared with the community.

SP11 : The system receives mapping information from external services.

SP12 : Registered cyclists review a bike path previously created by another user.

SP13 : Registered cyclists publish bike paths that were previously recorded privately.

C. Definitions, Acronyms, Abbreviations

C.1. Definitions

Street/Path segment: intended to be one where a proper bike track exists or where cars are rare and speed limits are compatible with the average speed of a bike, that corresponds to 14.3 km/h, so we can assume a proper range could be 5-25km/h [6]. It's a constitutive element of the bike path. It has its own status (e.g., optimal, fair, sufficient, needs maintenance) indicating its condition.

Bike Path: route made up of multiple streets.

Mobile device: a portable device that has an embedded system architecture, processing capability, on-board memory, and may have telephony capabilities (for example, cell phones, tablets, and smartphones) [**mobile_device**].

Manual Mode: bikers insert the data manually, specifying the name of the streets in the path and their status.

Automated Mode: bikers let BBP acquire data from their mobile devices while they bike. BBP should guess the user is biking given their speed; it should collect GPS information to reconstruct the followed path, and, at the same time, it should acquire data from the mobile device's accelerometer and gyroscope to keep track of any significant movement of the device itself that suggests the presence of potholes or other problems. Since there is the possibility of having false positives (e.g., non-existent potholes), the user will have to confirm or correct the information acquired by BBP before this is made available to the community.

Trip: A cycling session recorded by a user, consisting of a sequence of GPS points (route) and associated data like time and speed.

Obstacle: Any significant hazard or impediment on a path, such as a pothole, road construction, fallen debris, or dangerous intersection. Users can report obstacles to warn others.

Accelerometer-Gyroscope: Smartphone sensors that measure acceleration and orientation. BBP uses these to detect significant bumps or motions of the device that may indicate road problems (potholes, rough surface).

C.2. Acronyms

BBP: Best Bike Paths – the name of the platform.

GPS: Global Positioning System – is one of the global navigation satellite systems (GNSS) that provide geolocation and time information to a GPS receiver anywhere on or near the Earth where signal quality permits [7].

GDPR: General Data Protection Regulation – is EU regulation for personal data and privacy [1].

C.3. Abbreviations

G: Goal.

WP: World Phenomena.

SP: Shared Phenomena.

D. Revision history

- Version 1.0: 23/12/2025

E. Reference Documents

[1] GDPR.EU. *What is GDPR, the EU's General Data Protection Regulation?* Accessed November 4, 2025. n.d. URL: https://gdpr.eu/what-is-gdpr/.

[2] M Rossi M Camilli E Di Nitto. *Alloy*. Presentazione in slide. Slide della lezione, Corso di Software Engineering II, Politecnico di Milano.

[3] M Rossi M Camilli E Di Nitto. *Introduction to Requirements Engineering (RE)*. Presentazione in slide. Slide della lezione, Corso di Software Engineering II, Politecnico di Milano.

[4] M Rossi M Camilli E Di Nitto. *Structure of a RASD document*. Presentazione in slide. Slide della lezione, Corso di Software Engineering II, Politecnico di Milano.

[5] M Rossi M Camilli E Di Nitto. *UML and Requirements Engineering*. Presentazione in slide. Slide della lezione, Corso di Software Engineering II, Politecnico di Milano.

[6] D. C. Thompson et al. "Bike speed measurements in a recreational population: validity of self reported speed". In: *Injury Prevention* 3.1 (Mar. 1997), pp. 43–45. DOI: 10.1136/ip.3.1.43.

[7] Wikipedia contributors. *Global Positioning System — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=1318018964. [Online; accessed 4-November-2025]. 2025.

F. Document Structure

Section 1: Introduction

This section offers a brief introduction of the purposes and scope of the platform. It also contains the list of definitions, acronyms and abbreviations that could be found in this document. Finally, there are changelog of the document, containing the revisions list and their content, and document structure, which describes the main purposes of the sections of this document.

Section 2: Overall Description

This section offers a summary description about the overall organization of the system, providing some scenarios and a description of the main features offered by the application, and of the actors who use it. It also contains the considered Assumptions.

Section 3: Specific Requirements

This section contains a description of functional requirements through use cases and diagrams, along with the Hardware and Software constraints and the interfaces needed to get it work.

Section 4: Formal Analysis through Alloy

This section contains a formal analysis of the model presented in the previous sections.

2 | Overall Description

A. Product perspective

A.1. Scenarios

Registration and authentication Marco is an enthusiastic cyclist who wants to efficiently record his bike rides and discover new cycling routes to make the most of his time on the bike.

To do so, he decides to register on Best Bike Paths, gaining access to all features available to registered users. For the registration process, Marco provides his email address, his name and a password, after which the system confirms that the registration was successful.

In this phase, he also grants automatically the BBP app the necessary permissions to access device sensors (GPS, accelerometer, gyroscope) and to collect background data. These permissions are essential for enabling automatic route detection and detailed ride tracking, while ensuring full compliance with privacy regulations and data protection laws.

Once his account is created, he can log in using his credentials to enter the app whenever he wants and start using its tools to record and review his cycling routes.

Automated route recording The registered cyclist Marco launches the BBP app, which then continues running in the background as he starts riding.

The BBP system automatically detects Marco's bicycle usage through the device's speed and motion sensors and begins collecting GPS data to track the entire route taken.

Simultaneously, the BBP system detects any obstacles or issues (e.g., potholes, bumps, etc.) through movements recorded by the cyclist's device accelerometer and gyroscope, using which it assigns a condition rating to the roads traveled (e.g., optimal, fair, sufficient, needs maintenance). BBP also calculates in real time the statistics maintained by Marco during the route, such as total distance covered and average speed. Additionally, during the trip BBP queries external services to provide up-to-date weather data (weather conditions, temperature, wind, etc.) for the area of travel.

The system then automatically detects the end of the trip when Marco stops pedaling. Once the trip is finished, the cyclist can view each detected street, performance statistics, and weather conditions. Marco can confirm or reject the detected streets. At that point, he views all the automatically recorded information related to the path: the entire route

with the status of each street, the overall rating, statistics, weather, and detected obstacles. The cyclist can decide whether to keep or modify the collected data or delete the path entirely.

Once this process is completed, Marco gives a name to the path and can choose either to keep the itinerary private (stored in "My Paths") or to publish and share it with the BBP community. Finally, the system confirms to Marco that the recording or publication was successful.

Manual route recording The registered cyclist Marco decides to record a route he has already completed using the manual mode.

Marco accesses the dedicated function within the BBP app and specifies the names of the streets composing the itinerary, along with an overall rating from 1 to 5. Once these are entered, he can evaluate their conditions (e.g., optimal, fair, sufficient, needs maintenance) and report any obstacles or issues on the path via the map. The cyclist add data calculated by himself, such as the travel time, through which the system, which calculates and provides the total distance based on the entered route, can supply statistics like Marco's average speed.

Once the data entry is confirmed, Marco gives a name to the path and he can choose whether to save the itinerary only for himself (stored in "My paths") or to publish and share it with the BBP community. In the latter case, the BBP system updates the overall path score of the route.

Finally, the system confirms to Marco that the recording or publication was successfully completed.

Path visualization between two points The cyclist Marco wants to know how to get to work by bike.

He opens BBP and in the app's search interface, he specifies his home address as the starting point and his workplace address as the destination. The system then verifies the validity of the addresses and the availability of at least one cycling route between the two points.

Each paths is assigned an overall path score aggregated from the condition of the streets composing it (e.g., optimal, fair, sufficient, needs maintenance) and effectiveness of the itinerary in connecting the origin and destination points is also evaluated.

If multiple routes connect the two points, the BBP system orders them by path score (from highest to lowest) and displays up to the top five in a ranked list next to the map. By clicking on any available route, it is displayed on the map along with the identified obstacles and relevant information. Once Marco finds his preferred itinerary, he can close the consultation and go to work by bike.

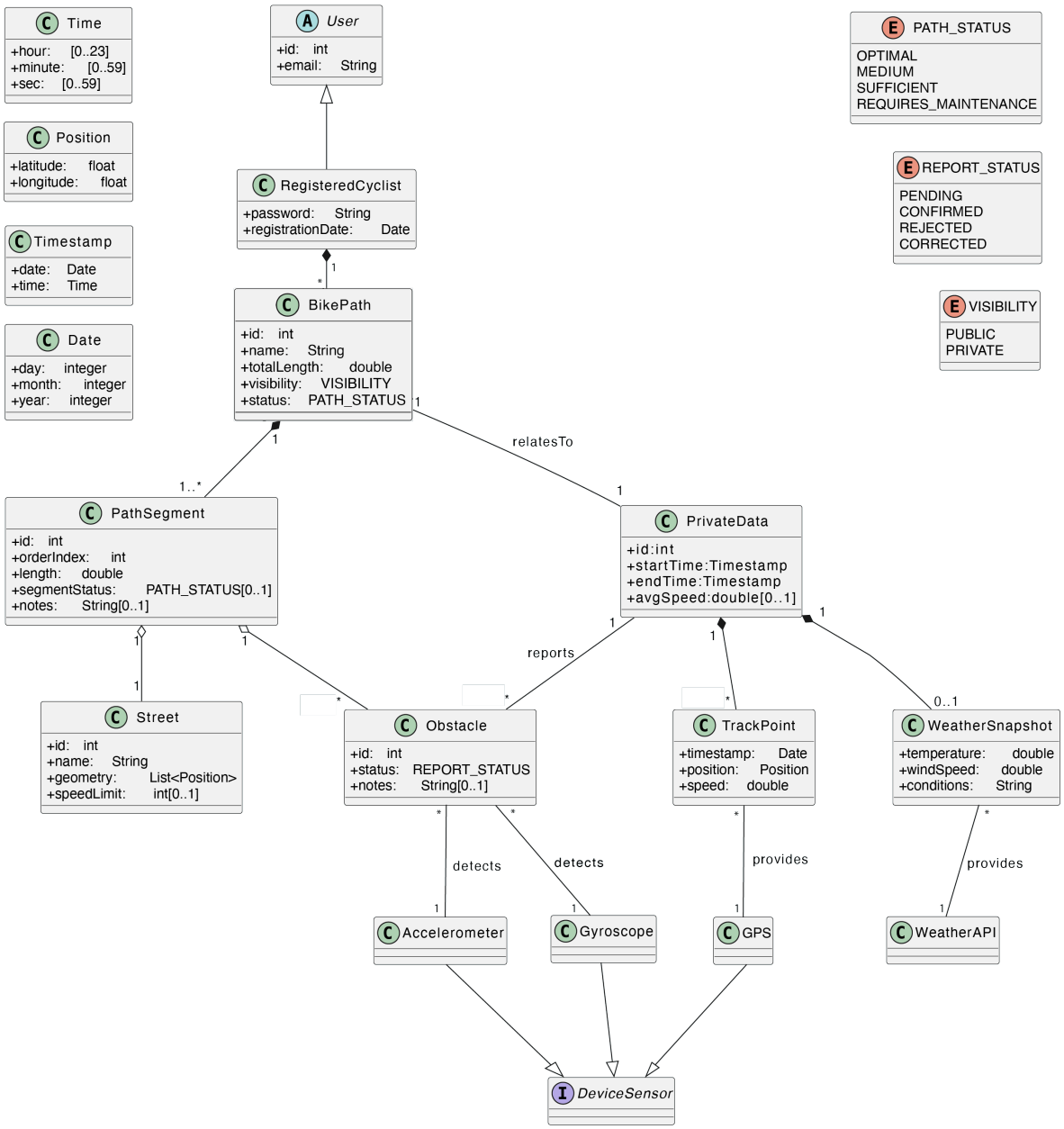
Display of the registered paths in the personal inventory The registered cyclist Marco wants to access the "My Paths" section of BBP to view his recorded routes and check his progress.

The BBP system then retrieves and displays a list of all routes recorded by the cyclist, showing for each the name, registration date, and time. If no routes are recorded, the system prompts the cyclist to record one. Marco can select a route and view it on the map along with any obstacles. In addition to the map, the recorded information about the itinerary is grouped and displayed: performance statistics, reviews (path conditions, specific notes), and any weather conditions. Marco can browse the route details and choose to share or delete a path.

Finally, Marco closes the consultation and returns to the main screen.

Reporting another path Reporting another path Registered cyclist Marco did not particularly like the path he had found to go to work because he encountered additional obstacles along the route that perhaps were not there before. He therefore decides to go to the "Report Path" section of the app, search for the path by name, and review it again. In this section, the cyclist can update the status of the streets, the presence of obstacles, and the overall rating. Once the report is completed, the path score and the overall status of the reported path are updated on the map.

A.2. Domain models



The diagram above shows the Domain Class model, highlighting the entities included in our system and the relationships between them. Additional explanations are necessary to achieve a complete understanding.

The abstract class **User** means to be useful in case of a potential extension, like the support also for runner, for instance.

A **BikePath** represents the trip done by the cyclist, which could be public accessable or available only for that specific cyclist. The Path status here is computed based on the status of the respective the PathSegments.

In case of automated recording, for each trip BBP keeps track of the data for the sensors and the GPS. These data are always private.

If the signals coming from the **SensorSample** are above a certain threshold, the **Obstacle** is created.

When the **TrackPoint**'s speed lows under a specific value, the trip is stopped and the system waits a confirmation about the tracks and the effectiveness of the found obstacles.

In case of manual mode, the cyclist inserts the necessary information, as the street names and their status, in substitution to sensors' data.

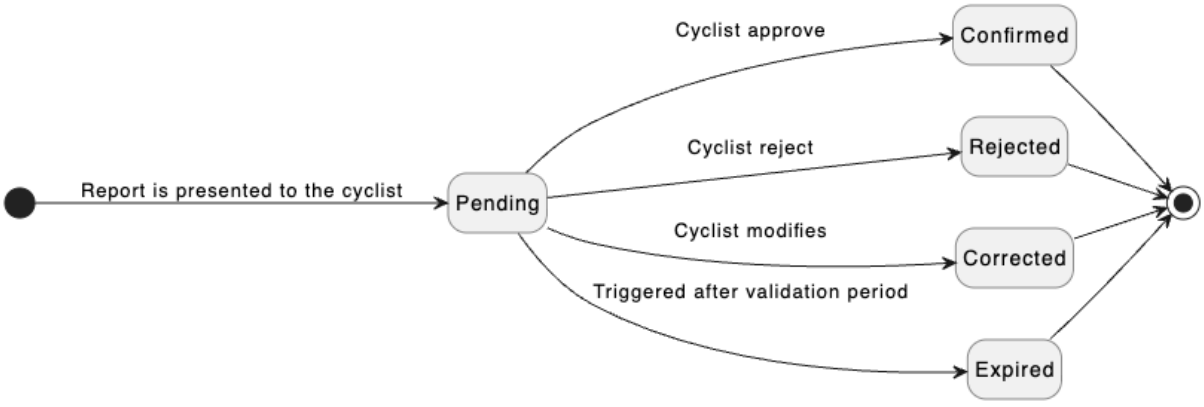
In any case, BBP has the sufficient information to save correctly the path.

From the **TrackPoint**'s position the system derives the **PathSegments** that composed a **BikePath**. A **PathSegment** contains the potential obstacles in its path. The segment refers to a real street.

A **Street** contains the basic data, like the set of coordinates it is made, its name and its speed limit. If there are obstacles across its path, the status is updated accordingly. A street is relevant only if the speed limit is under the threshold we imposed for the ciclist safety.

A.3. State Diagrams

Obstacle lifecycle The Obstacle lifecycle models the validation flow of obstacles detected along a path. In case of automated mode each obstacle starts in PENDING, awaiting confirmation or rejection by the user who performed the ride. A CONFIRMED report indicates that the obstacle truly exists; this event triggers a re-evaluation of the affected segment. A REJECTED report marks the detection as false, excluding it from future computations. Optionally, if a report remains PENDING beyond a configurable time limit, it transitions to EXPIRED, prompting revalidation or removal. This lifecycle ensures both automatic detection and human confirmation, maintaining data quality and credibility across the platform. In case of manual insertion, the obstacle is set automatically in CONFIRMED.



B. Product functions

User Registration and Authentication

This feature is optional and available to all users. It allows them to register and log in using an email and password, while also granting the necessary permissions to access the user's device sensors.

Automatic Route Detection

This function enables the app to detect when a registered user is cycling (start and end of a route) by using the device's motion and speed sensors, as well as GPS data to track the ride in real time. If the system detects that the cyclist is traveling on a road not classified as a cycling route, it pauses recording until a new route starts. This feature operates even when the app is running in the background.

Obstacle and Route Condition Detection

The app collects data from the device's accelerometer and gyroscope to analyze and identify potholes or other potential obstacles on the road. At the end of the route, the system uses this data to assign an overall condition rating (e.g., optimal, fair, sufficient, needs maintenance). This feature is available to registered users while cycling, and continues to work in the background.

Travel Statistics Calculation

The system calculates in real time various performance statistics for registered users while they are cycling. These include average speed and total distance traveled, based on data collected from the device's sensors, even when the app runs in the background.

Manual Route Logging

This feature allows a registered user to manually enter the details of a previously completed route. The cyclist provides the names of the streets that make up the route, together with their condition ratings (e.g., optimal, fair, sufficient, needs maintenance), and reports any obstacles or issues. The user must also input a score and travel time so that the system can calculate performance statistics. It is not possible to add streets that are not part of the recognized cycling path network.

Route Recording and Publishing

Once a route has been defined (either automatically or manually), registered users can assign it a name and choose whether to keep it private, stored in a personal list visible only to them, or publish it to share with the BBP community. If published, the system updates the list of shared routes and refreshes the status of both the paths and the streets involved.

Route Ratings and Reviews Management

This feature allows registered users to review the routes they have recorded assessing road conditions (e.g., optimal, fair, sufficient, needs maintenance), and noting the presence of potholes or other obstacles. In manual mode, the user provides these evaluations directly.

Route Visualization on Map

All users, whether registered or not, can view cycling routes on a map. The feature lets the cyclist enter a starting and destination address, verifies that both are valid, and checks for at least one available cycling path between them. If no route exists, the system informs the user about it. Alongside the map, the app displays a list of up to five routes with the highest path scores. By selecting a route, users can view it on the map along with detailed information in panels, including estimated travel time, distance, reported obstacles, route name, and user reviews (ratings and condition data).

The order is based on the score computed by this formula:

$$\text{Path Score} = \alpha \cdot \overline{S} + \beta \cdot (1 - \tilde{L}) - \gamma \cdot O_{\text{last}}$$

Where:

$\overline{S}$ = is the mean road-condition rating along the path, computed as the average of segment status scores on a 1–5 scale (5 = optimal, 1 = needs maintenance).

$\tilde{L}$ = is the normalized deviation of the path from the straight-line origin–destination segment (range 0–1), where 0 indicates a very close path and 1 indicates a highly distant path.

$O_{\textbf{last}}$ = is the number of obstacles reported since the last inspection/review of the path.

α, β, γ = non-negative weights that control the relative importance of road condition, directness, and obstacle penalties (often chosen so that $\alpha + \beta + \gamma = 1$).

Personal Routes Visualization and Management

In the “My Paths” section, registered users can view all their recorded routes along with their statistics and personal reviews. The system displays a complete list of recorded itineraries, including name, date, and recording time. If no routes are available, the app prompts the cyclist to record one. When a route is selected, it is shown on the map with grouped data displayed in descriptive panels or pop-ups, including performance statistics (total distance, travel time, and average speed), reviews (path conditions, obstacles, ratings, and notes), and any relevant weather details. The user can browse the route details and choose to share or delete a personal path.

External Route Reporting and Updating

This feature allows registered users to provide feedback on shared routes originally created and published by other members of the community. If a user identifies changes in road conditions or encounters new obstacles on an existing path, they can access the "Report Path" section to search for the route and submit an updated review. The user can modify the condition ratings of the specific streets, report new or resolved obstacles, and provide a new overall rating from 1 to 5. Once the report is submitted, the system re-calculates the overall path score and refreshes the status of the route on the map, ensuring that the information available to the community remains accurate and up-to-date.

C. User characteristics

BBP is targeted at cycling enthusiasts and casual riders who stand to benefit from discovering safer or better routes and tracking their rides. The main user classes include:

Registered Cyclist (Contributor)

An individual who creates a BBP account. These users actively use the app to record their cycling trips and contribute information on bike paths. They are typically cyclists who ride frequently enough to want to log their rides or improve the route database. They should have basic familiarity with smartphone apps and maps. Within this group, there may be power-users (for example, members of the cyclist club) who contribute a lot of data and perhaps act as local experts. Contributors are motivated by tracking their performance and by the community benefit of sharing route info.

Guest Cyclist (Viewer)

An individual (possibly a cyclist planning a ride or someone new in town looking for bike routes) who has not signed in. Guests primarily use BBP to query routes between two places and to view the community-contributed path information on the map. They cannot record new trips or add data. This class of users benefits from the system without directly adding to it. The app for them is purely a route-finding and informational tool. They may eventually sign up if they find value in contributing or saving their own data.

One special consideration: cyclists using the app while riding will have limited attention for the device (safety comes first). Therefore, one could consider the "active rider" as a user state – when in this state, the UI needs to be very glanceable. This is more of a use context than a separate user class, but it affects design for all cyclists when they are on the move.

D. Assumptions, dependencies and constraints

The development and operation of BBP are subject to the following constraints and assumptions:

D1: Accuracy of Sensor-based Detection It is assumed that the combination of GPS and motion sensors can provide sufficiently accurate data to identify cycling activity and road anomalies. The system will use a threshold (speeds between 5 km/h and 30 km/h indicate bicycling) to guess when the user is biking. We acknowledge this may not be perfect (e.g., a very slow bike ride might drop below the threshold, or a car stuck in traffic could mimic cycling speeds), but we assume these edge cases are relatively rare or can be handled via user confirmation. The accelerometer/gyroscope-based pothole detection is also an approximation; we assume typical pothole hits produce a distinct sensor signature that our algorithms can detect reliably. Fine-tuning will be needed, but we proceed with the assumption that modern smartphones are capable of this task.

D2: User Confirmation Mitigates False Positives We assume that users will take the time to review and confirm automatically detected obstacles after a ride. This is a crucial step to ensure data quality. It's a constraint that automated data is not published without

user intervention. Therefore, the system's usefulness relies on active user participation. We assume that contributors are motivated to provide accurate info (perhaps out of community spirit or personal benefit of having a better database) and that this extra step is acceptable in the user experience.

D3: External Services Availability The system depends on third-party services for weather (and possibly routing). We assume these services are generally available and reliable. A constraint is that BBP must abide by those services' usage policies. If an external API is down or unreachable, BBP functionality may be degraded (for instance, missing weather data or map imagery), but core functions like trip recording should remain unaffected. We design BBP to handle such failures gracefully.

D4: Data Privacy and Regulations It is assumed that collecting and storing user location data (trip logs) is permissible as long as we obtain user consent and protect the data. We also assume that the cyclist association has the right to use and distribute the user-contributed data (perhaps via user agreement in the app's terms of service).

D5: Maintaining Updated Path Information It is assumed that once the system is in use, the community will continually update and expand the path database. However, a constraint is that path condition can change over time (a path marked "Optimal" could degrade after a season or be under construction later). We assume that users will provide new reports if things change. We assume that the community input will be ongoing enough to keep data reasonably fresh.

D6: All users have an active email address.

3 | Specific Requirements

A. External interface requirements

A.1. User interfaces

The user interface design for Best Bike Paths aims to ensure simple, intuitive, and functional use of the application for both registered and unregistered cyclists. The key interfaces are:

Homepage: The homepage centers on the "Find Route" section, featuring an interactive map that displays only roads recognized by the system as cycling paths. The interface includes a search bar with two text fields for quickly entering a starting and destination address. Following data entry and validation, the path search is initiated. The map supports standard zoom and pan operations and features icons highlighting points of interest. The interface is visible to both registered and unregistered users, with limitations on other areas for the latter.

Registration/Login: The registration and login interface presents a simple and clear form with fields for email and password, along with specific error messages for invalid inputs (e.g., incorrect email format). Registration requires implicit user consent for access to device sensors and for personal data and privacy processing. The process provides instant feedback to inform the user of success or any registration issues.

Find Route: This interface is viewable by both registered and unregistered cyclists and displays on the map the possible cycling paths between the specified origin and destination. If no cycling paths are available between the points, the system shows an explanatory message. The routes, along with relevant information, are listed by name in a list on the right limited to five items, ordered by path score. By selecting a path, it is displayed on the map with a line tracing the path in the color corresponding to its status, along with reported obstacles shown on the map.

Manual Path Entry: The manual path entry interface allows a registered user to manually enter a path into the system with all relevant information. The user can add each of the path's streets through a search bar where available street names can be typed or selected. Streets not recognized by the system as cycling paths are not accepted. For each street, the user can indicate its condition by choosing from a list of basic options (e.g., optimal, fair, sufficient, needs maintenance). Obstacles can then be reported directly on the map along with a brief description. The interface includes fields for entering the path name, an optional description, travel time, and a star rating from 1 to 5. A final button

allows saving the entered data and viewing a path summary with relevant automatically calculated statistics (such as distance traveled and average speed). At that point, the user can choose to save the path in "My Paths" or publish it.

Automatic Path Entry: The automatic route entry interface activates automatically when a registered user begins cycling, thanks to the device's sensors. At the end of the route, performance statistics and a list of detected streets to be confirmed or rejected are shown. Following confirmation, a summary is displayed featuring a map highlighting the traced route with obstacles automatically detected by the sensors, street names with their respective status, star ratings, statistics, and a box with weather conditions. The user can confirm, delete, or modify the data. They can then add a description and must provide a name for the path. At that point, the user can choose to save the path in "My Paths" or publish it.

Report Path: Report Path: This interface allows for the re-review of an existing path previously created and published by another user. A search bar allows the user to find the existing path by name. Once the itinerary is selected, the interface displays its current details and enables the user to submit a new report. The cyclist can update the status of individual streets (e.g., optimal, fair, sufficient, needs maintenance), manage obstacles, add an optional description, and provide a new star score. Upon completion, the system confirms the success of the operation and automatically recalculates the path score and the overall path status, updating the information on the map for the entire community. Alternatively, the user can discard the report while in progress.

My Paths: The "My Paths" interface allows users to view their saved routes in chronological order. Each entry shows information such as name, date, statistics, and any weather conditions. Simple buttons are available to delete, view, or publish private paths. By choosing to view a route, it is displayed on the map along with detected obstacles. From there, the user can return to the path list. Deletions require confirmation to prevent accidental removal. This section is available only to registered cyclists.

A.2. Hardware interfaces

Best Bike Paths is designed to be used on mobile devices such as smartphones and smartwatches, which must be equipped with essential built-in sensors including GPS, accelerometer, and gyroscope. These sensors are necessary to accurately detect cycling activity, track routes, and identify any road anomalies. The app must also operate efficiently under limited resource conditions, considering the typical battery and performance constraints of mobile devices. For proper operation, the device must support continuous data recording in the background, requiring the appropriate permissions from the user. No additional external hardware is expected.

A.3. Software interfaces

BBP uses various internal and external software components to deliver its functionalities. It interfaces with location service APIs to acquire GPS data and with external weather services to obtain real-time environmental updates. For interaction with device sensors

and management of push notifications, BBP communicates with operating system features on smartphones and smartwatches. Collected data is synchronized with a cloud backend via RESTful APIs, which handle storage, updates, and sharing of routes and ratings. For interactive map visualization, BBP integrates specific libraries and SDKs to ensure a smooth and responsive user experience.

A.4. Communication interfaces

The application requires a stable internet connection for communication with the backend server and external services. Communications use secure protocols such as HTTPS with TLS to protect privacy and data integrity, including personal data, route tracking, and user feedback. Synchronization, route condition updates, and weather data retrieval occur in real time to keep information current. The system also supports push notifications through dedicated services that inform users about registration confirmations or important events related to detected routes.

B. Functional requirements

This section enumerates the detailed functional requirements of the Best Bike Paths system.

R1 The system shall allow a new user to create an account by providing necessary information.

R2 The system shall allow a registered user to log in by providing their credentials.

R3 The system shall permit users who are not logged in (guests) to use core viewing functions.

R4 The system shall provide an interface for a logged-in user to view and edit their profile information.

R5 The system shall allow a user to log out of their account on a device.

R6 The system should attempt to detect when the user is cycling and start recording.

R7 While recording, the system shall log the sequence of GPS coordinates that represent the user's path.

R8 The system shall automatically compute summary statistics for the trip.

R9 After stopping the recording and computing stats, the system shall show the user a summary of their ride.

R10 The system shall save the recorded trip data to the user's account.

R11 When a trip is recorded, the system shall attempt to retrieve weather data for the ride.

R12 If weather data was successfully obtained for a trip, the system shall display it along with the other stats on the trip summary view (R11).

R13 The system shall provide a feature for users to add a bike path entry manually.

R14 The system shall allow the user to mark any known obstacles or hazards along the path.

R15 The system shall prompt the user whether to publish this information to the community or keep it private.

R17 The system shall support continuing recording even if the user switches apps or turns off the screen, once a recording is started.

R18 While a trip recording is active (started by the user via R7), the system shall collect accelerometer and gyroscope data continuously in the background.

R19 The system shall analyze the sensor data in real-time (or in short batches) to detect possible road anomalies.

R20 For each detected potential obstacle or anomaly during the ride, the system shall log it internally in the context of the current recording.

R21 When the trip is finished, if the system has any automatically detected events (R20), it shall prompt the user to review them.

R22 The system shall present the detected events in a user-friendly way.

R23 The user shall be able to confirm, reject or modifies the events.

R24 After the user has finished reviewing and confirming the data from an automated collection ride, the system shall save the verified information.

R25 The system shall maintain a central repository of all published bike path entries.

R26 The system shall allow updates to existing path entries.

R27 Whenever new path information is published, the system shall integrate it into the dataset used for route queries.

R28 The system shall ensure that published path data does not include personal information about cyclists.

R29 The system shall provide an interface for the user to specify a start and end location for their desired trip (route query).

R30 The system shall compute a score or rating for each route found, to communicate its quality to the user.

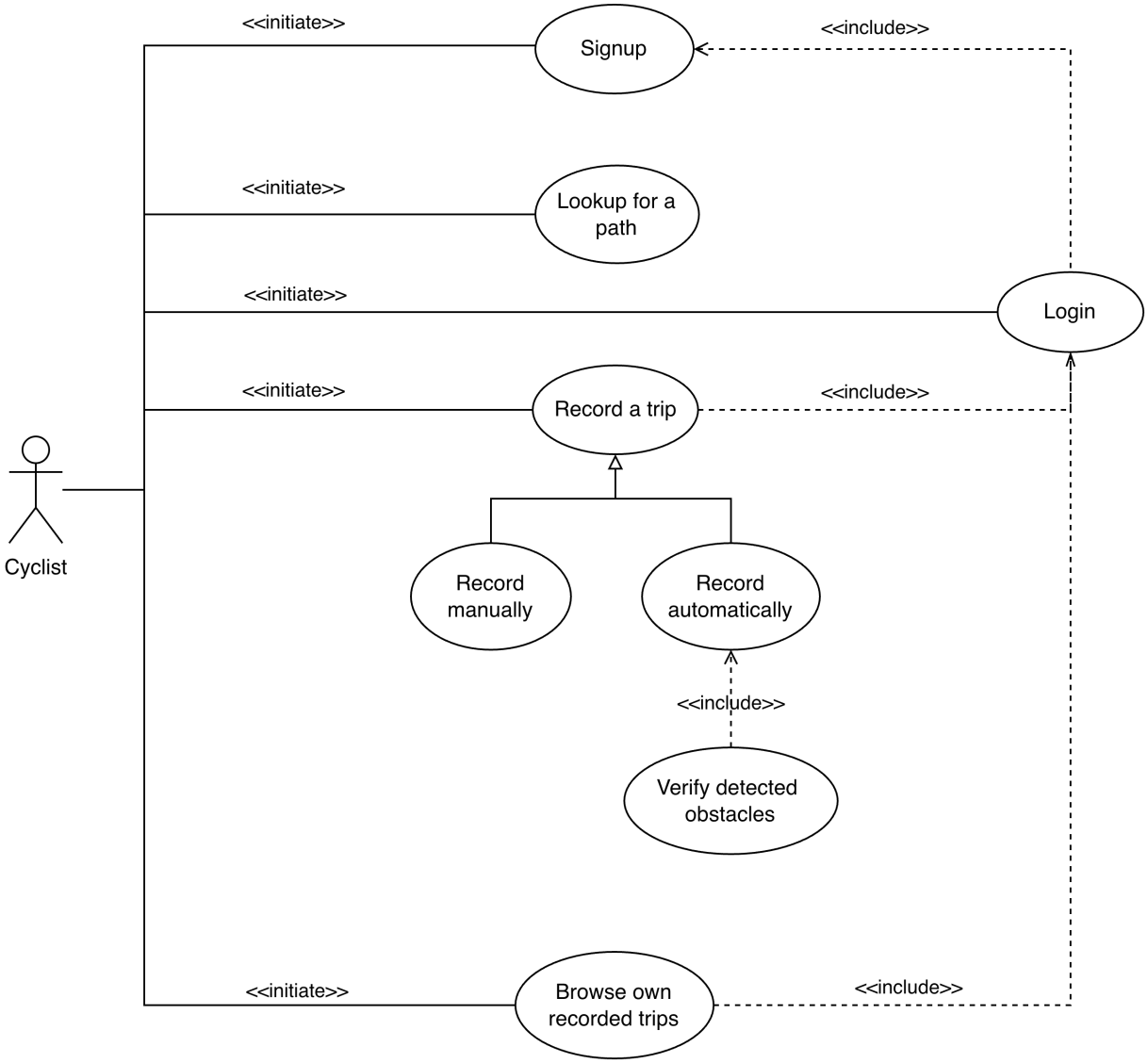
R31 The system shall display the calculated route(s) on the map UI for the user.

R32 For each route, the system shall provide a summary of key details.

R33 The system shall provide a section where a logged-in user can view their past recorded trips.

B.1. Use Cases and Use Case Diagrams

Cyclist Case Diagram



UC1. Login Cyclist

Name	Login Cyclist
Actors	Cyclist
Entry condition	Cyclist already registered
Events	a) Cyclist inserts credential in the proper input fields(email, password) b) The system checks the validity of such information c) The system redirects the cyclist to his/her personal dashboard
Exit condition	Cyclist is logged in
Exceptions	Invalid credentials

UC2. Sign-up Cyclist

Name	Sign-up Cyclist
Actors	Cyclist
Entry condition	Cyclist is not registered
Events	a) The cyclist inserts his/her email in the proper input field. b) The system checks if the email belongs to the whitelist of email domains and if the email has never been used. c) The cyclist inserts his/her password in the proper input field. d) The system checks if the password respects the requirements. e) The system redirects the cyclist to a confirmation page.
Exit condition	The cyclist is registered
Exceptions	• The user's email is already registered • The password doesn't satisfy the requirements

UC3. Record Manually

Name	Record Manually
Actors	Cyclist
Entry condition	The cyclist has correctly logged in
Events	a) The user enters the name of the streets in the path b) The system validates the streets inserted c) The cyclist inserts the status of each of the streets provided d) The system creates the new bike path
Exit condition	The user has correctly created manually his/her trip
Exceptions	• The user inserts an invalid street • The user doesn't provide the status of the streets

UC4. Record Automatically

Name	Record Automatically
Actors	Cyclist, WeatherAPI, DeviceSensor
Entry condition	Cyclist already registered
Events	a) The cyclist starts his/her trip bringing with him/her his/her mobile device b) The system triggers the start of the recording process when the cyclist reaches a certain speed c) The system keeps track of the path and any significant movement of the device d) The user stops the trip e) «*include*» "Verify detected obstacles" f) The system creates the new bike path
Exit condition	A bike path has been correctly created in automated mode
Exceptions	The device dies during the trip

UC5. Verify detected obstacles

Name	Verify detected obstacles
Actors	Cyclist
Entry condition	• The cyclist has correctly logged in • The cyclist has finished his/her trip
Events	a) The system displays a report of the path and retrieves a list of detected potential obstacles b) The cyclist checks every proposed obstacles and submits the report
Exit condition	The obstacles has been validated
Alternative 1	The user modifies the position of an obstacle
Exceptions	The user doesn't submit the report

UC6. Paths Lookup

Name	Path Lookup
Actors	Cyclist
Entry condition	Cyclist opens the BBP platform
Events	a) The cyclist inserts the origin and the destination point b) The system checks if there exists any bike paths that runs along those two points c) The system show the results bike paths
Exit condition	The cyclist has explored every bike path there is

UC7. Paths Browsing

Name	Trips Browsing
Actors	Cyclist
Entry condition	The cyclist has correctly logged in
Events	a) The user navigates to his personal page b) The system displays all his/her saved bike paths
Exit condition	The user has seen all his/her recorded bike paths
Alternative 1	The user decides to delete one of his/her trips
Alternative 2	The user decides to modify one of his/her trips
Alternative 3	The user decides to make public one of his/her trips
Exceptions	The cyclist actually has no previous bike paths

B.2. Activity and Sequence diagrams

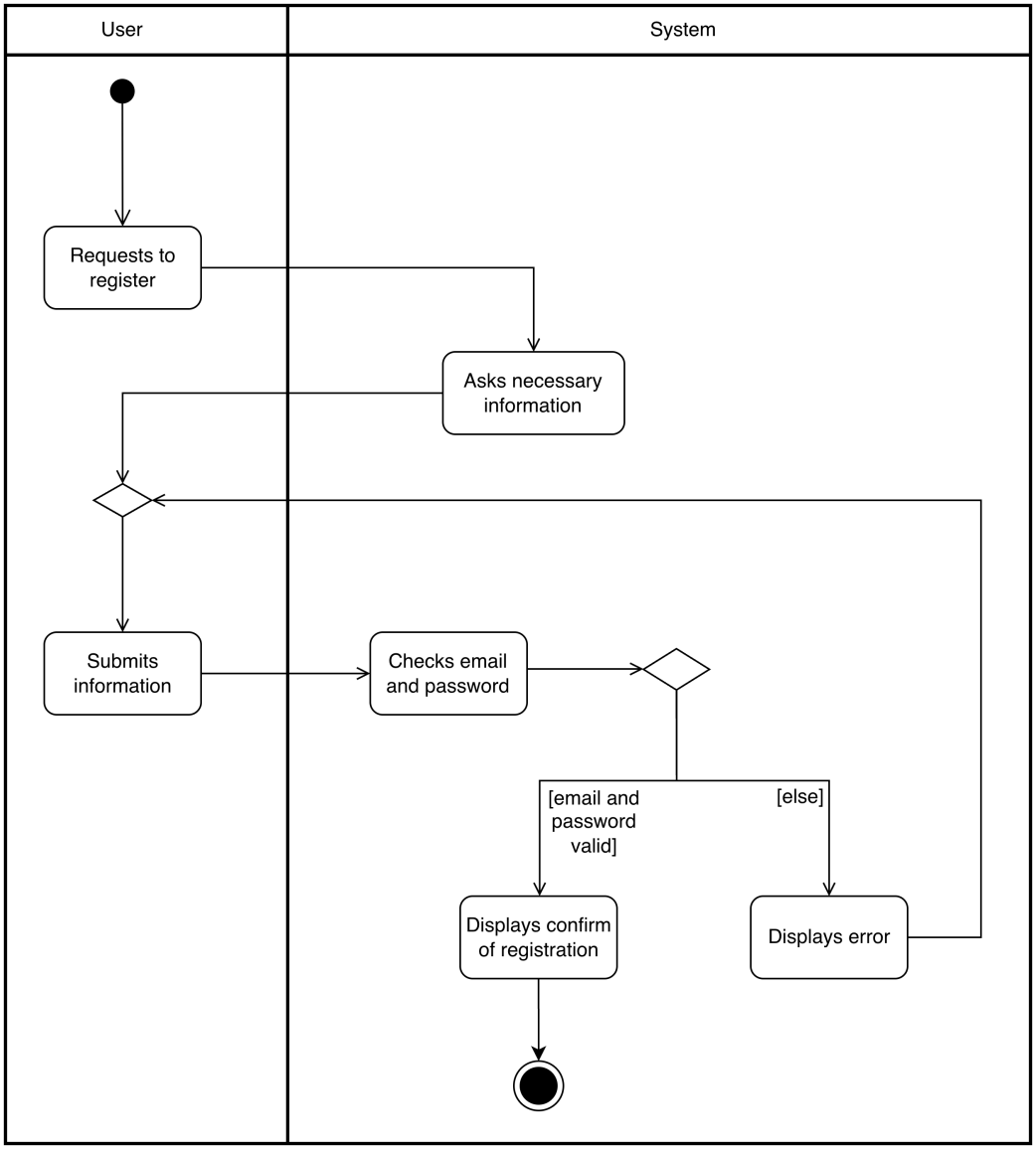


Figure 3.1: Signup

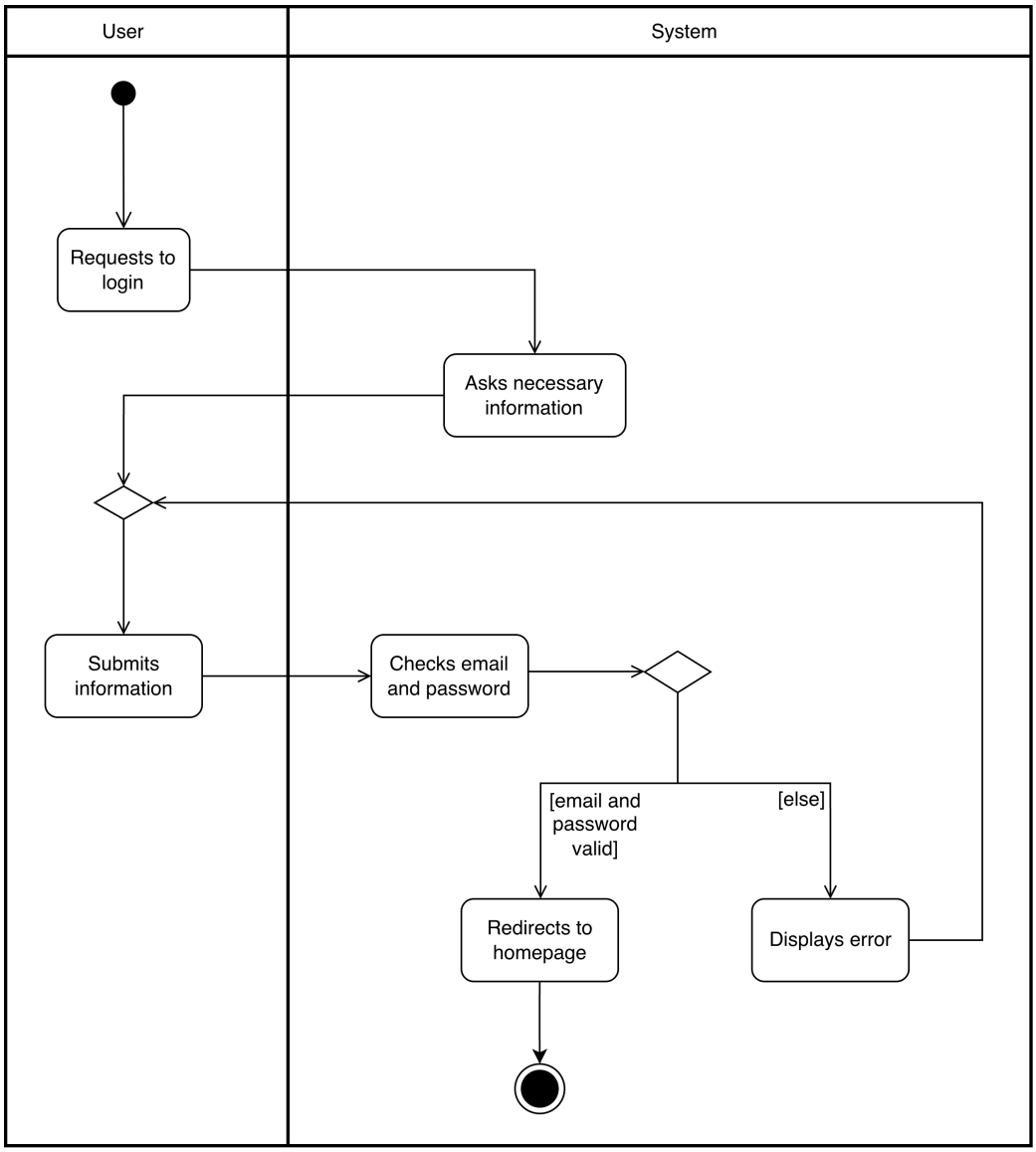


Figure 3.2: Login

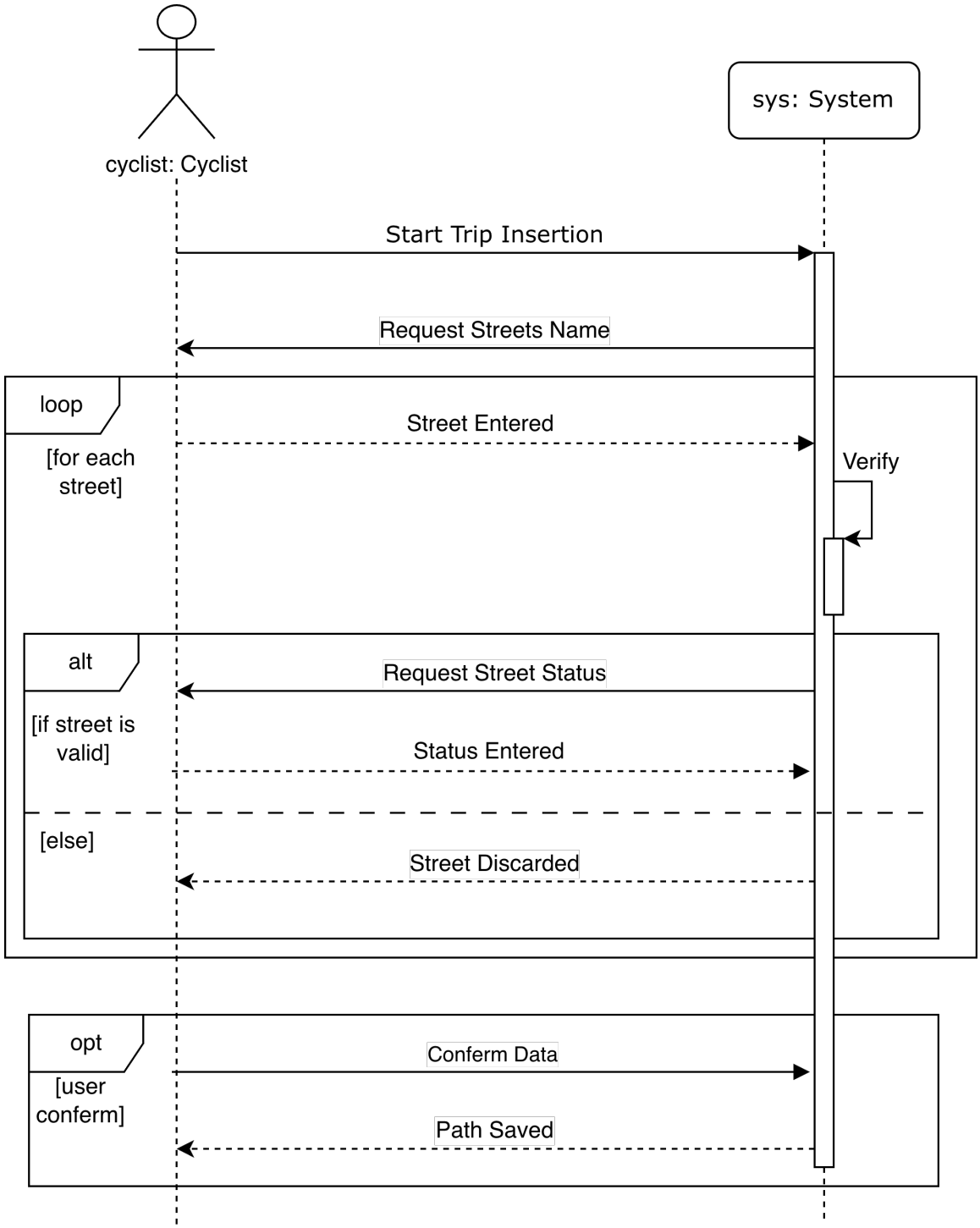


Figure 3.3: Manual Recording

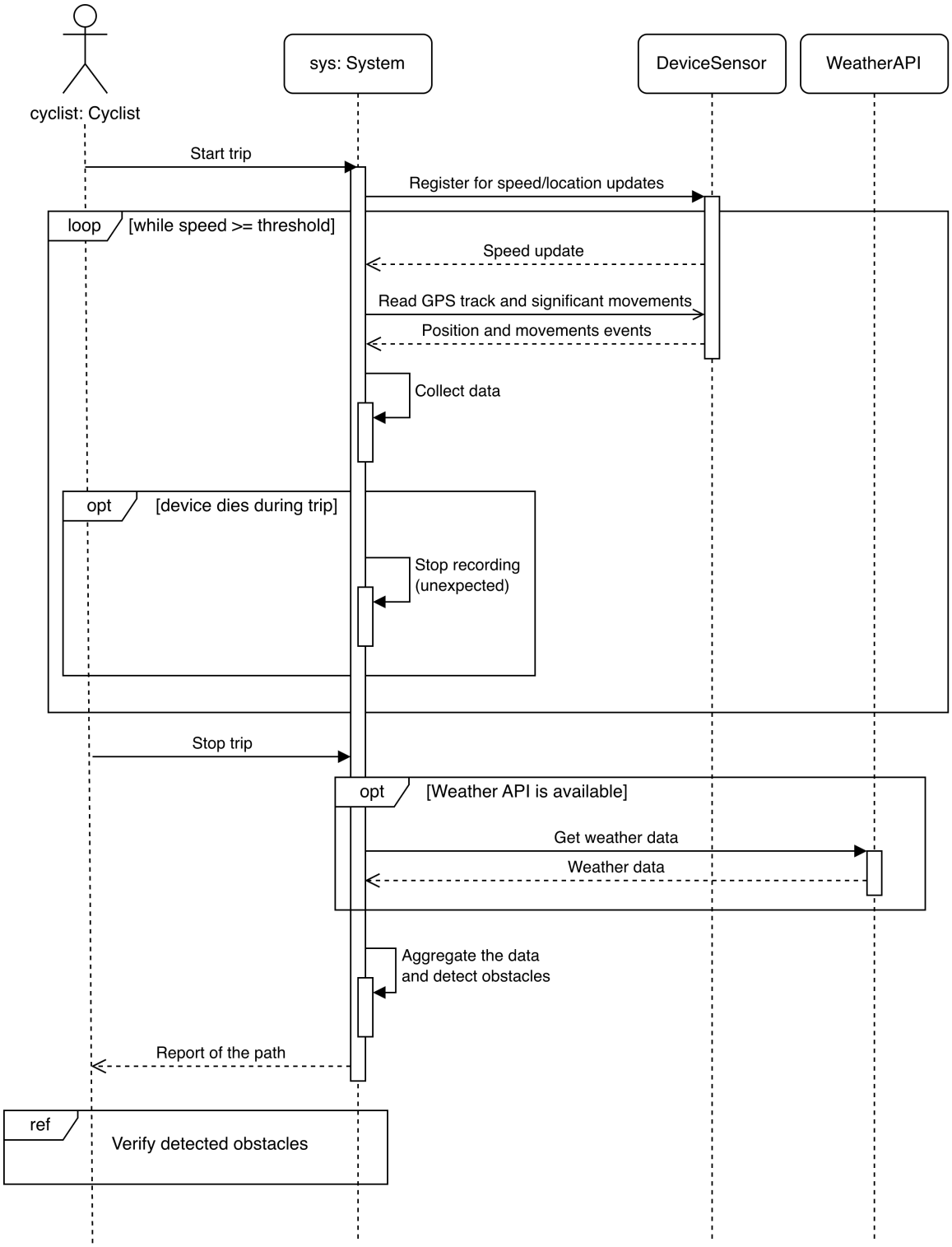


Figure 3.4: Automatically Recording

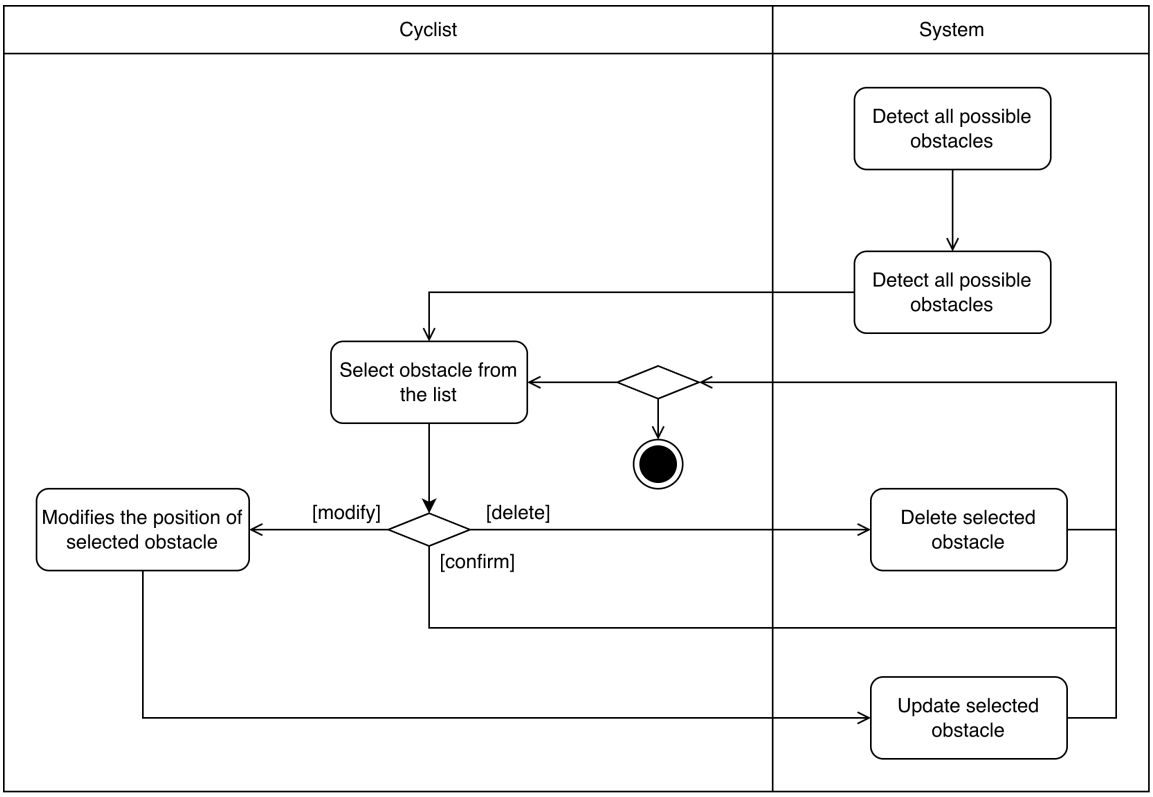


Figure 3.5: Verify Detected Obstacles

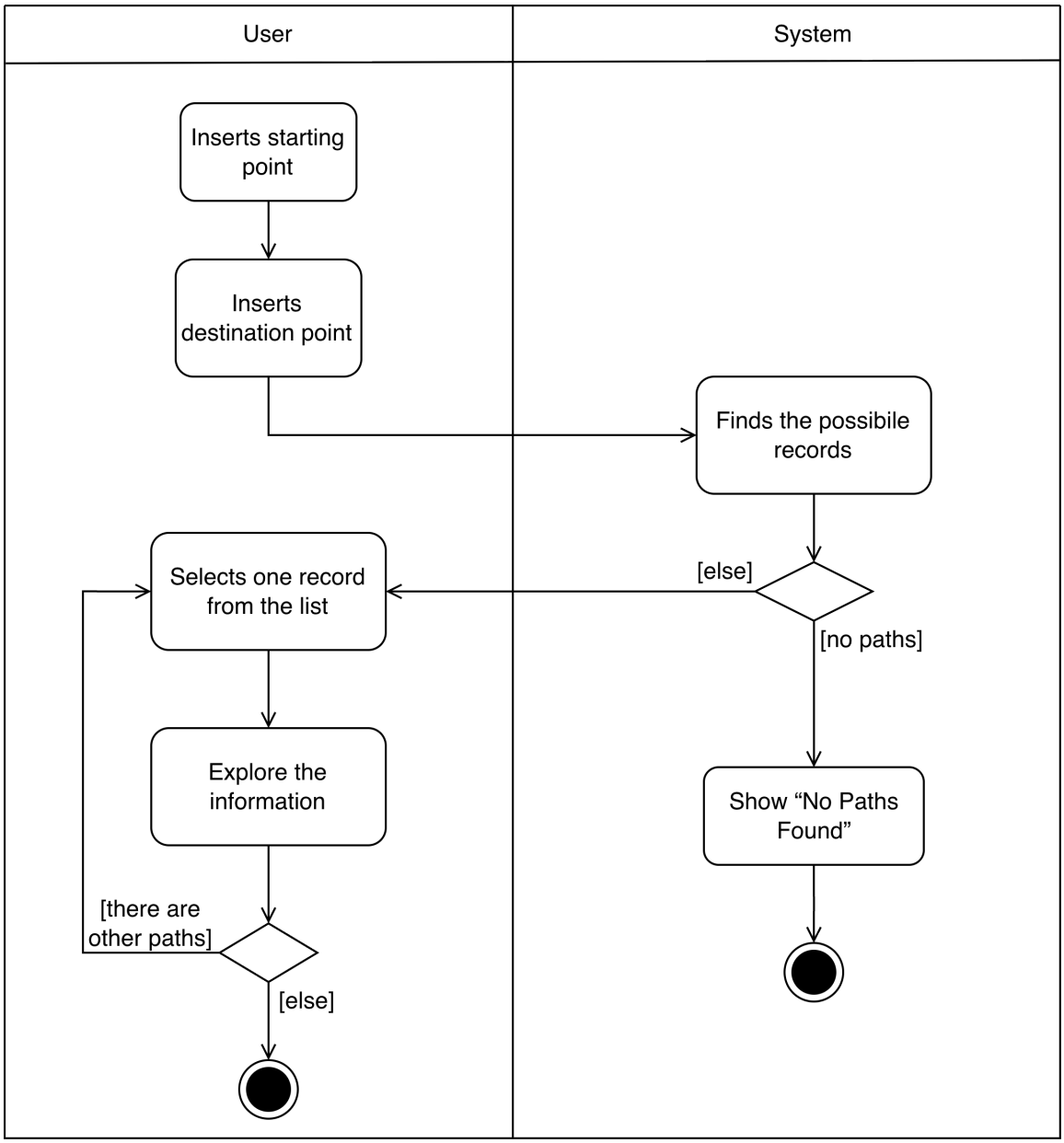


Figure 3.6: Path Lookup

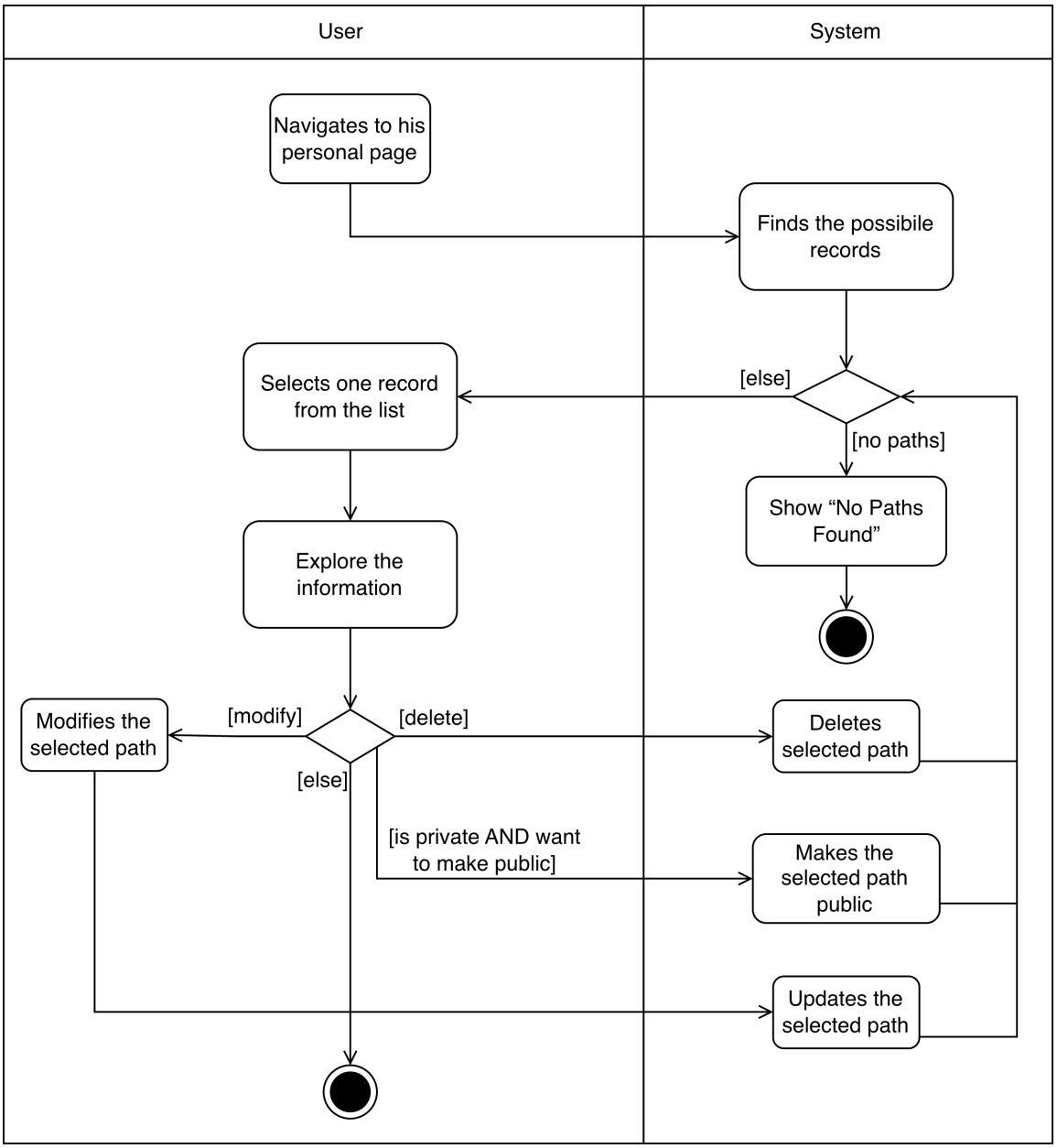


Figure 3.7: Path Browsing

B.3. Requirements Mapping

G1 : Unregistered cyclists can register into the system	
R1 The system shall allow a new user to create an account by providing necessary information.	D6 All users have an active email address.

G2 : Registered cyclists can log into the system	
R2 The system shall allow a registered user to log in by providing their credentials.	

G3 : Registered cyclists can manually record and insert information about their bike paths, including their status and the presence o relevant obstacles, to keep track of their cycling activities	
R13 The system shall provide a feature for users to add a bike path entry manually. R14 The system shall allow the user to mark any known obstacles or hazards along the path. R8 The system shall automatically compute summary statistics for the trip. R10 The system shall save the recorded trip data to the user's account. R11 When a trip is recorded, the system shall attempt to retrieve weather data for the ride. R12 If weather data was successfully obtained for a trip, the system shall display it along with the other stats on the trip summary view (R11). R15 The system shall prompt the user whether to publish this information to the community or keep it private.	D3 External Services Availability D4 Data Privacy and Regulations

G5 : Registered cyclists can automatically record their bike paths, including their status and the presence o relevant obstacles, to keep track of their cycling activities, confirming, correcting or deleting the provided feedback about the cycling trip they've just traveled	
R16 The system should attempt to detect when the user is cycling even if they haven't explicitly started recording. R7 While recording, the system shall log the sequence of GPS coordinates that represent the user's path. R8 The system shall automatically compute summary statistics for the trip. R9 After stopping the recording and computing stats, the system shall show the user a summary of their ride. R10 The system shall save the recorded trip data to the user's account. R11 When a trip is recorded, the system shall attempt to retrieve weather data for the ride. R12 If weather data was successfully obtained for a trip, the system shall display it along with the other stats on the trip summary view (R11). R17 The system shall support continuing recording even if the user switches apps or turns off the screen, once a recording is started. R18 While a trip recording is active (started by the user via R7), the system shall collect accelerometer and gyroscope data continuously in the background. R19 The system shall analyze the sensor data in real-time (or in short batches) to detect possible road anomalies. [R20] For each detected potential obstacle or anomaly during the ride, the system shall log it internally in the context of the current recording.	D1 Accuracy of Sensor-based Detection. D2 User Confirmation Mitigates False Positives. D3 External Services Availability. D4 Data Privacy and Regulations. D5 Maintaining Updated Path Information.

R21 When the trip is finished, if the system has any automatically detected events (R20), it shall prompt the user to review them. R22 The system shall present the detected events in a user-friendly way. R23 The user shall be able to confirm, reject or modifies the events. R14 The system shall allow the user to mark any known obstacles or hazards along the path. R24 After the user has finished reviewing and confirming the data from an automated collection ride, the system shall save the verified information.	

G6 : Registered and unregistered cyclists can visualize on a map the bike paths between a point of origin and a point of destination	
R3 The system shall permit users who are not logged in (guests) to use core viewing functions. R15 The system shall prompt the user whether to publish this information to the community or keep it private. R26 The system shall maintain a central repository of all published bike path entries. R27 The system shall allow updates to existing path entries. R28 Whenever new path information is published, the system shall integrate it into the dataset used for route queries. R29 The system shall provide an interface for the user to specify a start and end location for their desired trip (route query). R30 The system shall compute a score or rating for each route found, to communicate its quality to the user. R31 The system shall display the calculated route(s) on the map UI for the user. R32 For each route, the system shall provide a summary of key details.	D5 Maintaining Updated Path Information

G7 : Registered cyclists can consult their own inventory of recorded bike trips.	
R10 The system shall save the recorded trip data to the user's account. R26 The system shall maintain a central repository of all published bike path entries. R27 The system shall allow updates to existing path entries. R33 The system shall provide a section where a logged-in user can view their past recorded trips.	

G8 : Registered cyclists can delete data they have already recorded in their itinerary on a specific route.	
R27 The system shall allow updates to existing path entries.	

G9 : Registered cyclists can delete information they have already shared with the community on a specific route.	
R27 The system shall allow updates to existing path entries.	

C. Performance requirements

The application's GPS logging should be efficient and accurate. It shall obtain location updates at least once every few seconds (configurable based on desired detail). The position error should ideally be within 5-10 meters (typical for GPS) when signal is good. The system must handle up to, say, 10 hours of continuous tracking without crashing or exhausting memory (ensuring route data is streamed to storage incrementally, not all kept in RAM).

D. Design constraints

D.1. Standards compliance

- Best Bike Paths complies with the General Data Protection Regulation (GDPR) to ensure user privacy protection and proper management of personal data.
- Communications between client and server occur via secure protocols, such as HTTPS and TLS, to guarantee the confidentiality and integrity of transmitted information.
- The app adheres to usage policies of external service APIs, including request limits and specific provider conditions.
- The system uses the standard UTC time format to ensure correct synchronization of events and data across various devices and servers.

D.2. Hardware limitations

- BBP is designed for mobile devices equipped with GPS, accelerometer, and gyroscope sensors necessary for detecting cycling activity and route conditions.
- It must be compatible with a wide range of smartphones and smartwatches, ensuring efficient use of computing resources and battery life.
- The app supports background recording with optimized sensor sampling to limit energy consumption, while respecting mobile operating system restrictions.
- Explicit user authorization is required for continuous access to location sensors during recording.

D.3. Any other constraint

- BBP must manage degradation or interruptions of external services (weather and routing), ensuring core functions remain operational in such cases.

E. Software systems attributes

E.1. Reliability

The system must operate without interruption for extended periods. To achieve fault tolerance, its back-end deployment should utilize replication and redundancy. Additionally, the system should maintain offline backups of data storage for use in disaster recovery in the event of data loss.

E.2. Availability

The BBP system (especially server-side components for route queries and data sync) should have high availability. Target uptime could be 99.9%; this implies that the Mean Time To Recovery (MTTR), or the average time required to restore service after a fault occurs, should be limited to approximately 0.365 days per year. Maintenance should be scheduled in off-peak times and, if possible, done without taking the service completely offline (rolling updates).

E.3. Security

The BBP platform does not store sensitive personal information about users; only email and password are collected during the sign-up process.

All communication between the mobile app and the backend server shall be encrypted using HTTPS. This protects sensitive data like login credentials and location info from eavesdropping. Similarly, calls to third-party APIs should use HTTPS endpoints. There should be no transmission of personal data over unencrypted channels.

On the backend, user data (account info, ride logs) shall be stored securely. Passwords must be hashed with a strong algorithm – so even if the database is compromised, raw passwords are not exposed. Personal information linking a user to their contributions should be protected – for example, published path data might not reveal the user's identity publicly, but in the database it's linked; that database should have proper access controls and not be exposed. Regular security measures like firewalling the database, using prepared statements (to avoid SQL injection) are required. These might be considered design-level, but they stem from the requirement of safeguarding data integrity and confidentiality.

E.4. Maintainability

The system must uphold a strong standard for maintainability. This involves implementing appropriate design patterns and maintaining high-quality standards. The code needs to be well-documented, and the use of hard coding should be kept to a minimum. Furthermore, a thorough testing routine should be established, which covers at least 75% of the entire codebase, excluding interface code.

E.5. Portability

The mobile app shall support a range of devices like Android and iOS. While the app should follow platform UI conventions, core functionality and data should be consistent between Android and iOS versions. A user switching platforms should find BBP has the same features and their account data accessible. This means the backend and API serve both. Any platform-specific limitations should be abstracted (e.g., differences in sensor API are handled within each app, but both deliver similar results).

4 | FORMAL ANALYSIS USING ALLOY

In this chapter, we present a report of the most significant results of the formal analysis of the system performed through the specification language Alloy 6. The primary objective is to validate the consistency of the domain model and the correctness of the system's dynamic behaviors.

Specifically, we focused on the lifecycle of the Bike Path and its associated Obstacle Reports. Since the application relies on both manual input and automated sensor detection (which may produce false positives), it is critical to ensure that unverified data is strictly isolated from the public domain. To achieve this, we defined a dynamic model that captures how the state of these entities evolves over time, from the moment a trip is recorded to its eventual publication to the community. This formal model specifically focuses on reports generated via automated sensor detection.

A. Signatures

Here is the list of all the signatures acting in the model, representing the main entities involved in the tracking and reporting process. We utilized the var keyword to indicate relations or attributes that can mutate over time, such as the growing inventory of a user or the changing status of a report during the validation process.

```
// STATUS ENUMERATIONS
enum PathCondition { Optimal, Medium, Sufficient, Maintenance }
enum ReportStatus { Pending, Confirmed, Rejected, Corrected }
enum Visibility { Private, Public }

// CORE ENTITIES

// The physical street infrastructure.
sig Street {}

// The registered user of the BBP platform.
sig RegisteredCyclist {
    var inventory: set BikePath
}

```

```
// The digital representation of a trip recorded by a user.
sig BikePath {
    owner: one RegisteredCyclist,
    segments: set PathSegment,
    var visibility: one Visibility,
    var score: one Int
}

// A segment of a trip that maps to a specific physical street.
sig PathSegment {
    refersTo: one Street,
    var reports: set ObstacleReport,
    var status: one PathCondition
}

// A report regarding an obstacle or road condition.
// (Maps to the 'Obstacle' entity in the Domain Model,
// but in this case we're talking about the entire
// lifecycle of an obstacle)
var sig ObstacleReport {
    var status: one ReportStatus,
    var condition: one PathCondition
}
```

B. Facts

With the fact keyword, Alloy allows to define constraints and properties that must be true in all the possible instances of the model. Since we are modeling a dynamic system, we distinguish between structural constraints (which define valid relationships) and evolution constraints (which define valid state transitions over time).

```
// STRUCTURAL CONSTRAINTS

// Every obstacle report must belong to exactly one path segment.
fact ReportHierarchy {
    always (all r: ObstacleReport | one s: PathSegment | r in s.reports)
}

// Reports associated with a segment are bound to the context of that segment.
fact ConsistentReporting {
    always (all s: PathSegment, r: s.reports | r in ObstacleReport)
}

// EVOLUTION AND LIFECYCLE

// Initialization: All newly created reports must start in the 'Pending' state.
fact ReportInitialization {
```

```
    always (all r: ObstacleReport | once (r.status = Pending))
}

// Evolution of Report Status
fact ReportStatusEvolution {
    always (all r: ObstacleReport |
        (r.status = Pending implies r.status' in (Pending + Confirmed + Rejected + Correc
        (r.status in (Confirmed + Rejected + Corrected) implies r.status' = r.status)
    )
}

// Evolution of Visibility
fact VisibilityEvolution {
    always (all b: BikePath |
        (b.visibility = Private implies b.visibility' in (Private + Public)) and
        (b.visibility = Public implies b.visibility' = Public)
    )
}

// SAFETY CRITICAL CONSTRAINTS

// Safe Publication Rule
// Ensures that a Public path NEVER contains Pending reports.
// This covers both the transition (becoming public) and the steady state (staying public
fact SafePublication {
    always (all b: BikePath |
        b.visibility = Public implies
        no r: b.segments.reports | r.status = Pending
    )
}

// Frame Condition
fact FrameConditions {
    always (all b: BikePath |
        b.visibility' != b.visibility implies (b.visibility = Private and b.visibility' =
    )
}

// Consistency: A segment with confirmed severe obstacles cannot be Optimal
fact SafetyConsistency {
    always (all s: PathSegment |
        (some r: s.reports | r.status = Confirmed and r.condition = Maintenance)
        implies s.status != Optimal
    )
}
```

C. Scenarios

In this section, we utilize specific predicates to simulate the most critical behaviors of the Best Bike Paths system. The generated instances allow us to visualize the evolution of the data over time steps, verifying that the lifecycle of reports and paths adheres to the specified requirements.

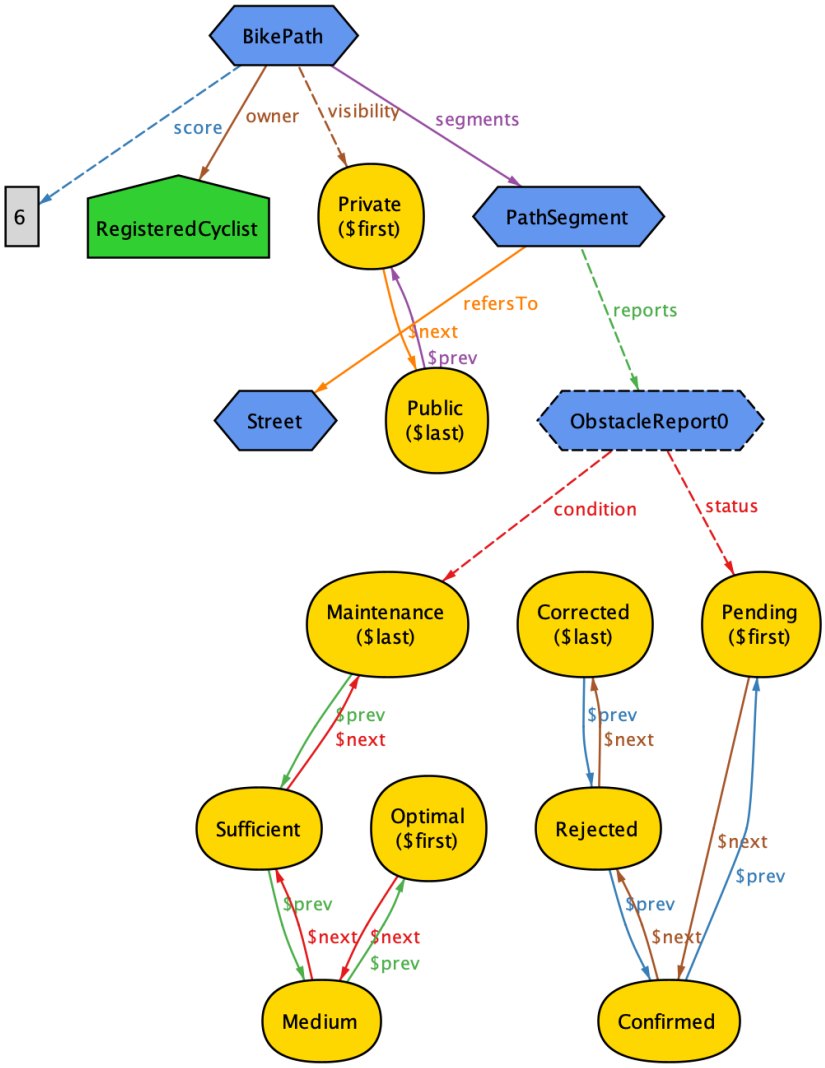
C.1. Model Instantiation

Before analyzing the dynamic evolution of the system, we present a static snapshot of a populated system instance. This visualization highlights the hierarchical relationships between the main entities (Cyclist, Bike Path, Segment, Report) and the environment (Street) as defined in the structural facts.

```
// Visualization of the Domain Structure
// Shows a populated instance with users, paths, segments, and streets
pred WorldInstance[] {
    #RegisteredCyclist = 1
    #BikePath = 1
    #PathSegment > 0
    #ObstacleReport > 0

    // Constraint to force visual connections in the diagram
    all b: BikePath | #b.segments > 0
    all s: PathSegment | #s.reports > 0
}
run WorldInstance for 3
```

The diagram below depicts a consistent state of the system where a RegisteredCyclist possesses a BikePath. The path is composed of PathSegments, which map to physical Streets and contain ObstacleReports. This structure validates the hierarchy enforced by the fact ReportHierarchy.

Note on Visual Projection: To enhance the readability of the dynamic traces presented in the following sections (4.3.2 to 4.3.5), a visual projection has been applied to the generated diagrams. While the underlying model retains full structural consistency (including users, streets, and segments), the specific scenario diagrams explicitly display only the entities involved in the state transitions. Static structural elements that remain unchanged during the execution are hidden to strictly focus on the evolution of the data lifecycle.



C.2. The "Happy Path" (Validation and Publication)

This scenario illustrates the standard workflow for a registered cyclist. We simulate a sequence of events where a user records a trip containing a sensor-detected obstacle. Initially, the information is uncertain (Pending) and private. The user then reviews the data, confirms the obstacle, and finally publishes the path to the community.

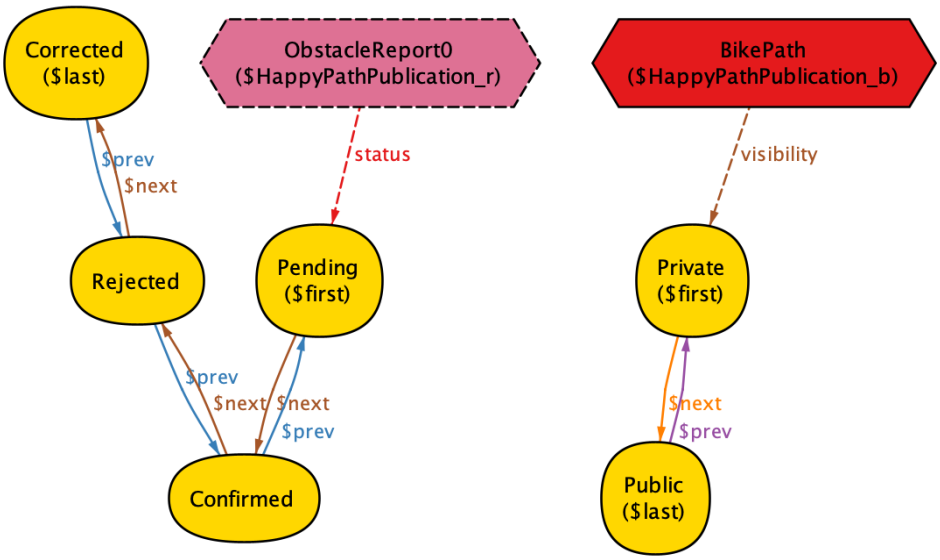
```
pred HappyPathPublication[] {
    some b: BikePath, r: ObstacleReport | {
        // Pre-condition
        b.visibility = Private
        r in b.segments.reports
        r.status = Pending

        // Step A: User confirms
        eventually (r.status = Confirmed)
```

```
10
11          // Step B: Publish
12          eventually (b.visibility = Public)
13      }
14  }
15  run HappyPathPublication for 4 but exactly 1 RegisteredCyclist, 1 BikePath, 1 ObstacleReport
```

The following sequence of diagrams depicts the temporal evolution of the instance generated by the predicate HappyPathPublication. The arrows indicate the current state of the entities at each significant time step.

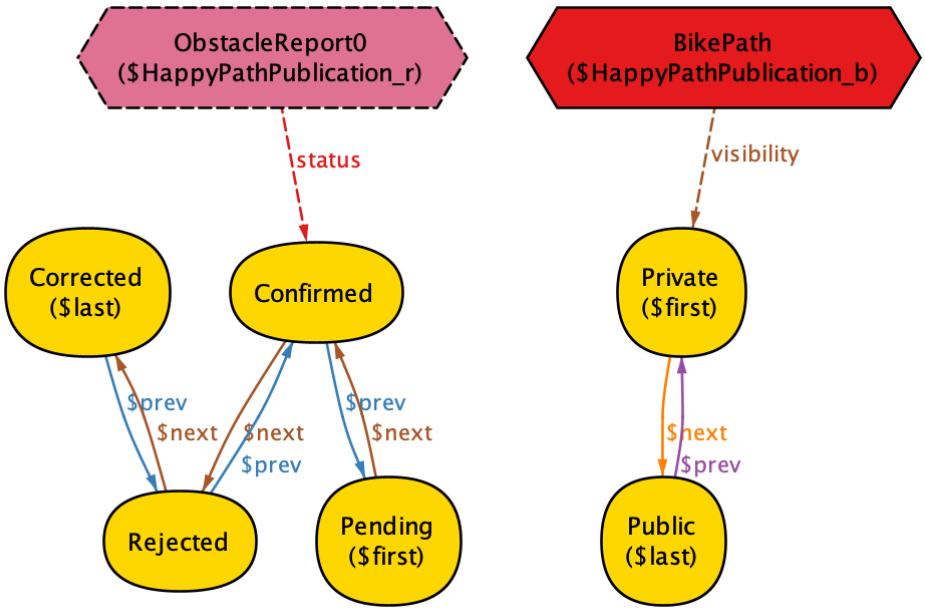
Step 0: Initial Recording

At the beginning of the trace, the system has instantiated a BikePath containing one ObstacleReport generated by the sensors. As defined in the structural facts, the report status is initialized to Pending (indicating raw, uncertain sensor data), and the path visibility is Private.



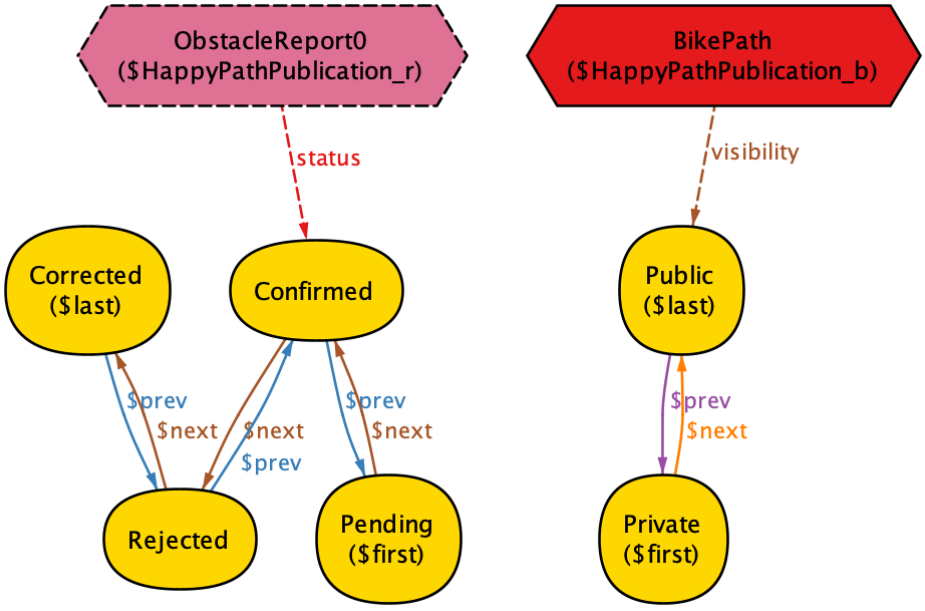
Step 1: User Confirmation

In the subsequent state, the registered cyclist intervenes to review the collected data. The ObstacleReport status transitions to Confirmed. Crucially, the path visibility remains Private at this stage; the system enforces that the path cannot become public immediately, as the confirmation is a prerequisite for publication.



Step 2: Publication

Finally, since the data integrity constraint is satisfied (no Pending reports exist within the path), the system allows the BikePath visibility to transition to Public. The validated information is now shared with the community.



C.3. Handling False Positives (Safety Mechanism)

This scenario demonstrates the system's ability to filter out incorrect data (e.g., a bump detected as a pothole), ensuring the quality of the information database. We simulate a

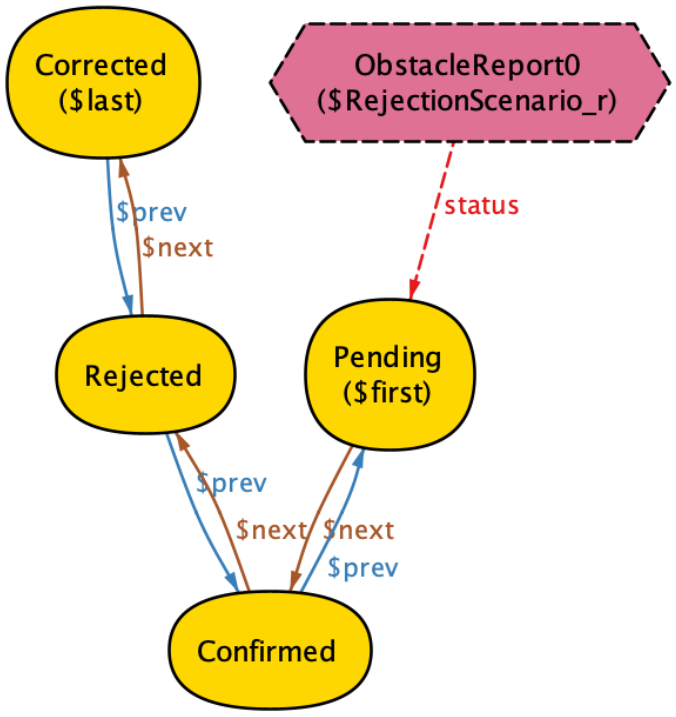
case where the sensors detect an anomaly, but the user identifies it as a false positive.

```
pred RejectionScenario[] {
    some r: ObstacleReport | {
        // Pre-condition: Sensor detects an anomaly (Pending)
        r.status = Pending

        // Step A: User rejects the report
        eventually (r.status = Rejected)

        // Post-condition: The report stays rejected and never reverts to Pending
        // This ensures the false positive is permanently excluded.
        always (r.status = Rejected implies after r.status = Rejected)
    }
}
run RejectionScenario for 3
```

The following diagrams illustrate the temporal evolution of the instance generated by the predicate RejectionScenario. This trace highlights the safety mechanism preventing incorrect sensor data from polluting the system.

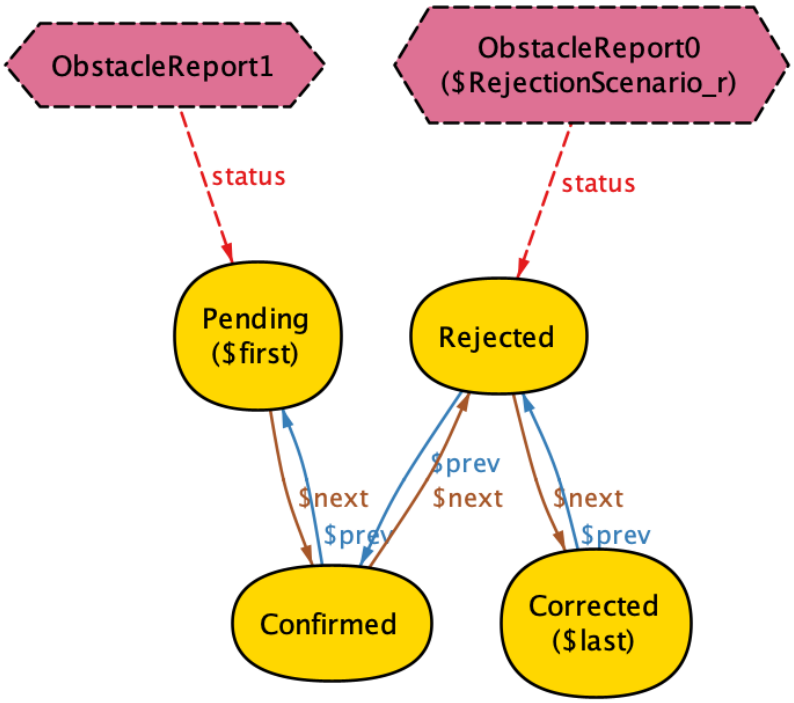
Step 0: Detection (Potential False Positive)

At the beginning of the trace, an ObstacleReport is instantiated by the system based on sensor readings (e.g., accelerometer spikes). Its status is initialized to Pending, indicating that an anomaly has been detected but not yet validated by the user.



Step 1: User Rejection

In the subsequent step, the cyclist reviews the notification and identifies the detection as a false positive (e.g., a rough road patch mistaken for a pothole). The status transitions to Rejected. As specified in the model's logic, this report is now permanently flagged as invalid and will be excluded from the public map and path scoring algorithms.



C.4. Data Correction

This scenario illustrates the "Modify" workflow within the Obstacle Report lifecycle. It addresses the situation where a registered cyclist acknowledges a sensor detection but finds the automatically inferred details (such as the specific type of obstacle or its severity) to be inaccurate. This modification may include updating the report's condition/severity based on the user's perception. Instead of rejecting the report, the user updates the information, causing the report to transition from Pending to Corrected. In the system's logic, this state is treated as a valid verification, allowing the data to be preserved and used in future path scoring calculations.

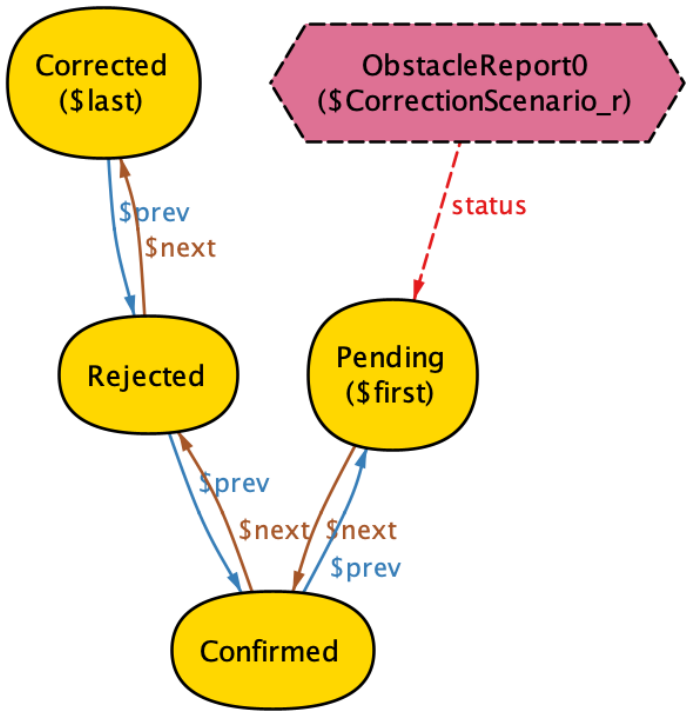
```
pred CorrectionScenario[] {
    some r: ObstacleReport | {
        // Pre-condition: Sensor detects an anomaly (Pending)
        r.status = Pending

        // Step A: User modifies the details (e.g., obstacle type) and submits
        // The status transitions to Corrected instead of Confirmed
        eventually (r.status = Corrected)

```

```
        // Post-condition: The report remains in the Corrected state
        // This ensures the user's manual input is finalized and preserved.
        always (r.status = Corrected implies after r.status = Corrected)
    }
}
run CorrectionScenario for 3
```

The following diagrams visualize the temporal evolution of the instance generated by the predicate CorrectionScenario. These snapshots demonstrate how the system handles user modifications to automated reports.

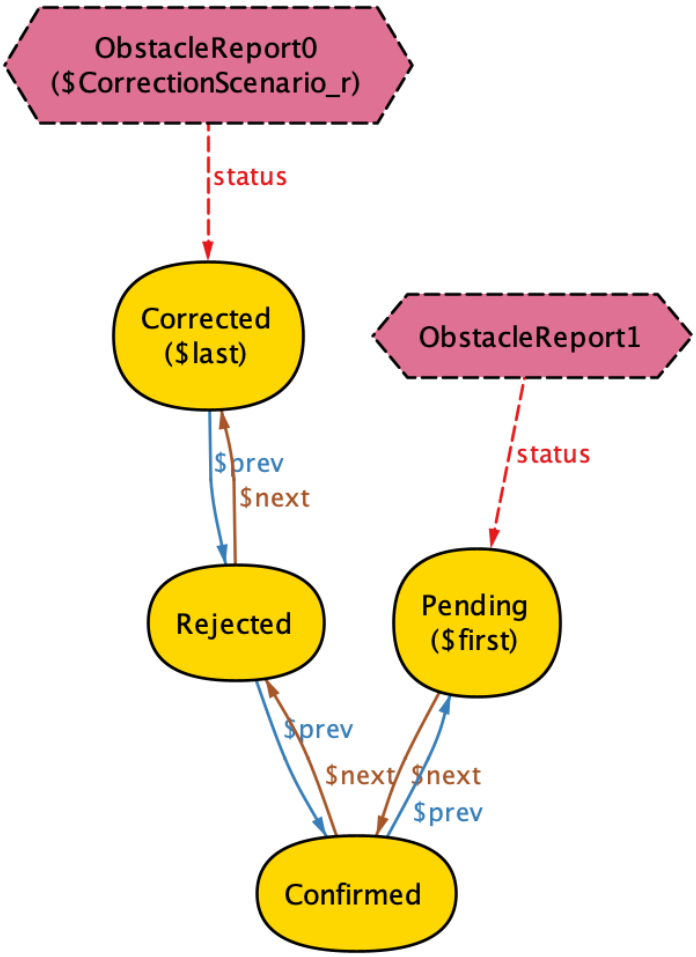
Step 0: Initial Detection

At the beginning of the trace, the sensors instantiate an ObstacleReport with the status set to Pending. This indicates that the system has detected a potential anomaly, but the specific details (such as obstacle type or severity) are provisional and await user review.



Step 1: User Correction

In the next step, the registered cyclist chooses to modify the report details instead of simply confirming or rejecting it. The status transitions to Corrected. This state validates the report's existence while updating its attributes based on the user's manual input, making it ready for inclusion in the path's statistics.



C.5. Path Consistency and Degradation

This scenario focuses on the logical consistency between the reported data and the segment status. It addresses the requirement that the system must react to validated hazards by updating the path's condition properties. Specifically, we simulate a case where a segment is initially classified as Optimal. Subsequently, a sensor-detected report flagging a severe obstacle (Maintenance) is validated and Confirmed by the user. The scenario demonstrates that upon confirmation, the system enforces a consistency rule: the segment status automatically degrades, ensuring it is no longer marked as Optimal. This behavior proves that validated risk factors correctly influence the path's representation.

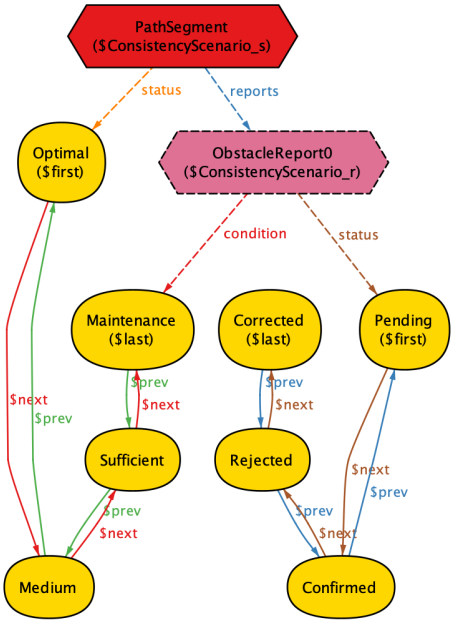
```
pred ConsistencyScenario {
    some s: PathSegment, r: ObstacleReport | {
        // Pre-condition:
        // The segment is initially Optimal
        s.status = Optimal
        // The report exists on this segment, is severe (Maintenance), but still Pending
        r in s.reports
        r.condition = Maintenance
        r.status = Pending
```

```
10
11          // Step A: The user confirms the severe obstacle
12          eventually (r.status = Confirmed)
13
14          // Post-condition:
15          // Once the severe report is confirmed, the segment status MUST change.
16          // It serves as a proof that the segment is no longer considered 'Optimal'.
17          always (r.status = Confirmed implies s.status != Optimal)
18      }
19  }
20  run ConsistencyScenario for 3
```

The following diagrams visualize the temporal evolution of the instance generated by the predicate ConsistencyScenario. These snapshots demonstrate the enforcement of the data integrity rule, highlighting how the validation of a severe hazard forces an immediate update to the path's condition status.

Step 0: Initial Inconsistency (Transient State)

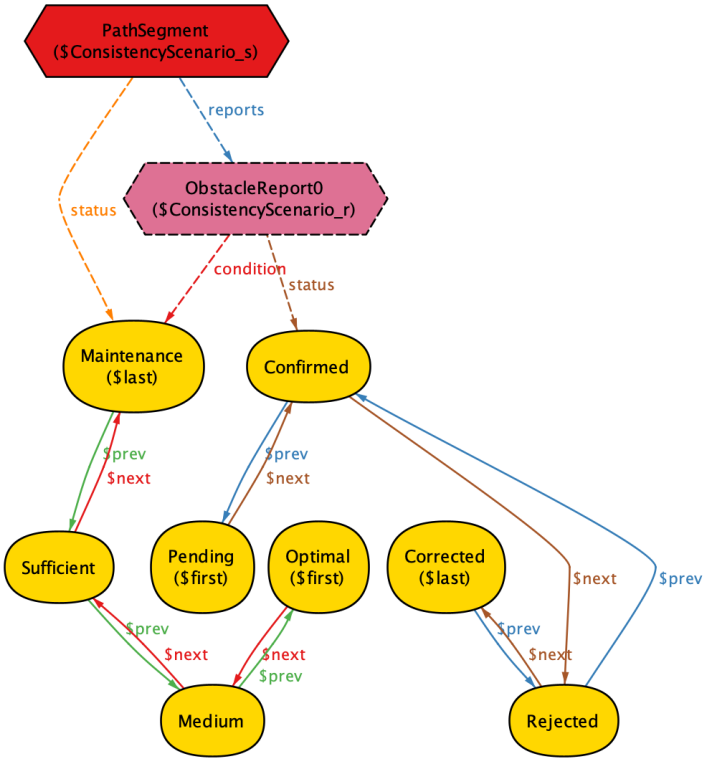
At the beginning of the trace, the PathSegment is currently rated as Optimal (first). Simultaneously, the system holds an ObstacleReport flagged with a severe condition (Maintenance), but its validation status is still Pending. This represents the transient state where a potential hazard has been detected by sensors but has not yet been validated by a user.



Step 1: Consistency Enforcement

In the subsequent state, the user validates the report, transitioning its status to Confirmed. As visualized in the diagram, the consistency rule triggers an immediate update: the PathSegment status shifts away from Optimal, settling into Maintenance (last). This

proves that the system automatically enforces data integrity, ensuring that a path with a confirmed severe obstacle cannot remain classified as optimal.



D. Assertions

In addition to simulating specific execution traces through predicates, which serve primarily to demonstrate that certain behaviors are possible (as seen in the "Happy Path" and "Rejection" scenarios), the formal analysis requires a stronger verification method to prove system correctness. To achieve this, we employ assertions. Unlike predicates, which look for a single instance satisfying a condition, assertions command the Alloy analyzer to search the entire state space for any possible violation of a specified property. If no violation is found, it provides a high degree of confidence that the property holds true in all possible execution scenarios.

For the Best Bike Paths (BBP) system, the most critical invariant concerns Data Integrity and Trust. Since the platform relies heavily on automated sensor detection, there is an inherent risk of generating "false positives" (e.g., detecting a pothole where there is none). If such unverified data were to be exposed to the public community, the reliability of the platform would be compromised. Therefore, we must formally prove that the system's logic strictly prevents the publication of any path that has not completed the validation workflow.

The following assertion, PublicSafety, formalizes this requirement. It imposes a constraint that must hold valid at every instant of time (always) and for every bike path instance in the system.

```
// Safety Check: Public Data Integrity
// We assert that it is IMPOSSIBLE for a BikePath to be 'Public'
// while containing any 'Pending' (unverified) reports.
assert PublicSafety {
    always (
        all b: BikePath |
        b.visibility = Public implies
        no r: b.segments.reports | r.status = Pending
    )
}
check PublicSafety for 5
```

The assertion logic can be broken down as follows:

Temporal Scope: The always keyword ensures the check is performed continuously across all state transitions, not just at the beginning or end of a trace.

Universal Quantification: We iterate over all b: BikePath to ensure no path escapes the check.

Implication: The condition is triggered only when a path becomes Public (b.visibility = Public implies...).

Forbidden State: The right-hand side of the implication (no r... | r.status = Pending) asserts that the set of pending reports within that public path must be empty.

To verify the results, executing the check command on the model produced the result: No counterexample found. This outcome is significant as it confirms that the structural constraints and evolution rules defined in Section 4.2 are robust. Specifically, the model mathematically demonstrates that there is no sequence of operations (user actions, sensor updates, or system transitions) that could trick the system into publishing a path while it still contains unverified sensor data. The requirement for manual user confirmation before publication is thus formally guaranteed.

5 | EFFORT SPENT

	Chapter 1	Chapter 2	Chapter 3	Chapter 4	Revision
Caldognetto	2	10	18	2	3
Salone	2	9	4	12	4