POLITECNICO DI MILANO

1863

Computer Science and Engineering

Software Engineering II

2025 – 2026

RASD

Requirement Analysis and Specification Document

Best Bike Paths(BBP)

By

Jayasurya Marasani (11023924)
Arunkumar Murugesan (11051547
Sneharajalakshmi Palanisamy (11132155)

Deliverable:	RASD
Title:	Requirement Analysis and Specification Document
Authors:	Jayasurya Marasani, Arunkumar Murugesan and Sneharajalakshmi Palanisamy
Version:	1.0
Date:	23-December-2025
Download page:	https://github.com/BBPpolimi/MurugesanPalanisamyMarasani
Copyright:	Copyright © 2025, Jayasurya Marasani, Arunkumar Murugesan and Sneharajalakshmi Palanisamy – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

Cities and suburbs often offer multiple ways to reach a destination by bicycle, but the best route depends on more than just distance. Surface quality, potholes, dangerous crossings, temporary construction and car traffic exposure can all influence safety and comfort. At the same time, cyclists increasingly want quantified trip histories (distance, speed, recurring commutes, weekly totals) that are easy to record and review.
Best Bike Paths (BBP) is a software system that lets users:

1. Record their bike trips and store them as part of a personal log.
2. Contribute information about bike paths, including surface quality and obstacles.
3. Query the system for "best" bike paths between two points, based on both quality and effectiveness of the route.
4. Be a guest user and make use of the fastest and easiest route from point A to point B.

1.1.1 Goals

G1 – Personal trip tracking: Registered users can record, store, and review trips with key statistics and map visualization.

G2 – Manual path information: Registered users can insert or edit path segment statuses and obstacles and decide whether each contribution is publishable.

G3 – Automated path information: Registered users can enable sensor-assisted acquisition while biking; BBP detects candidate issues that must be confirmed or corrected by the user before publication.

G4 – Route search and visualization: Any user (registered or guest) can request bike paths between an origin and a destination and visualize one or more route options.

G5 – Path scoring: BBP orders multiple route options by a path score combining segment quality and route effectiveness.

G6 – Merging: BBP merges publishable information from multiple users about the same segments, considering freshness and confirmations or contradictions.

G7 – Privacy and compliance: Personal and location data are protected; publication is always under user control.

G8 – Weather and context enrichment: When possible, BBP enriches trip data with meteorological information such as temperature, wind, and weather conditions retrieved from an external service.

1.2 Scope

BBP is a mobile-first web application (and/or native app) that interacts with user's devices (GPS, accelerometer, gyroscope, network) and with external services (maps and weather). The core domain is community-maintained bike path information plus individual trip logs. The system covers:

1. Trip recording (real-time tracking, saving, reviewing).
2. Manual input of bike path data (streets, segments, status, obstacles).
3. Automatic acquisition of bike path data (GPS tracks and sensor events).

4. User confirmation or correction of automatically detected issues before publishing.
5. Path search between origin and destination for any user, ranked by path score.
6. Merging of multiple user's reports into consolidated path status information.

1.2.1 World Phenomena

These are facts/events that happen in the real world, possibly observed by the system but not controlled by it:

WP1 – Cyclists ride in the real world, following some route along streets and bike tracks.

WP2 – Cyclists use their mobile devices while riding; the device's GPS position and motion sensors change over time.

WP3 – The physical road surface and infrastructure change quality over time (e.g., a new pothole appears, a segment is repaired).

WP4 – Weather conditions (temperature, wind, rain, etc.) evolve over time.

WP5 – Users judge the quality and safety of paths (e.g., "optimal", "requires maintenance").

WP6 – Different users may have conflicting judgments on the same path.

WP7 – Map providers maintain base map data (streets, bike lanes).

WP8 – Weather providers expose meteorological data via an external service.

1.2.2 Shared Phenomena

These are phenomena that involve both the world and the BBP system:

SP1 – A user registers and logs in to BBP.

SP2 – A registered user starts/stops trip recording; BBP stores GPS tracks and timing.

SP3 – BBP queries the device's GPS and motion sensors while a trip is ongoing.

SP4 – BBP stores, displays and updates statistics about a user's trip.

SP5 – A user manually inserts or edits path information (streets, segments, status, obstacles).

SP6 – BBP interprets raw sensor data as possible obstacles or rough patches.

SP7 – BBP asks the user to confirm or correct automatically detected issues.

SP8 – BBP marks path information as "publishable" or "private" according to user choice.

SP9 – Any user specifies an origin and destination; BBP computes one or more bike paths.

SP10 – BBP computes a score for each candidate path and returns an ordered list with map visualization.

SP11 – Different users submit information about the same path; BBP merges it into a consolidated status.

SP12 – BBP contacts an external weather service to enrich recorded trips with weather data.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1. **Bike path**: A route where either a dedicated bike track exists or car traffic is light and speed limits are compatible with typical cycling speeds.
2. **Path segment**: A contiguous portion of a route, usually corresponding to a street segment in a map provider dataset.
3. **Path status**: A qualitative label describing segment condition (Optimal, Medium, Sufficient, Requires Maintenance).
4. **Obstacle**: A localized issue on a segment (pothole, debris, bump, dangerous crossing, etc.) with a type, severity, and location.
5. **Trip**: A recorded ride by a registered user including GPS track and computed statistics.
6. **Publishable information**: Path-related data the owner chooses to share with the community.
7. **Candidate obstacle**: An issue detected automatically from sensor data that is not publishable until user confirmation.
8. **Consolidated status**: The system's current best estimate for a segment status obtained by merging multiple reports.
9. **Trip statistics**: Distance, duration, average speed, maximum speed, elevation gain, etc.

1.3.2 Acronyms/Abbreviations

1. **BBP**: Best Bike Paths
2. **GPS**: Global Positioning System
3. **API**: Application Programming Interface
4. **NFR**: Non-functional requirement
5. **UC**: Use case
6. **OS**: Operating system

1.3.3 Abbreviations

1. **G**: Goal
2. **WP**: World Phenomenon
3. **SP**: Shared Phenomena
4. **R**: Requirement

1.4 Revision History

Version	Date	Description
1.0	23/12/2025	Full Document of RASD.

Table 1: Document Version History

1.5 Reference Documents

The assignment for this document and all the information included refer to the following documentation:

1. The specification for the 2025/26 Requirement Engineering and Design Project for the Software Engineering II course.
2. The slides on the webeep page of the Software Engineering II course.
3. Sample RASD “Students & Companies”
4. Alloy Analyzer [1] for Formal Analysis of Alloy.
5. Figma [2] for UI/UX design.
6. ChatGpt [3] for Paraprashing the text.
7. Overleaf [5] for RASD document preparation.
8. Planttext [4] for UML Diagrams Creation.

1.6 Document Structure

The entire document is structured as follows:

1. **Introduction:** a brief description of the project
2. **Overall Description:** Description of the system including scenarios and domain model.
3. **Specific Requirements:** Detailed requirements: interfaces, functional requirements, diagrams, performance requirements, constraints and software attributes.
4. **Formal Analysis Using Alloy:** Formalizes part of the domain using Alloy programming.
5. **Effort Spent:** Report on efforts spent by each member of the team.
6. **References:** References for the project.

2 Overall Description

2.1 Product Perspective

BBP is a new stand-alone system that:

1. Runs as a mobile-first web application and/or native mobile app.
2. Uses the device's GPS, accelerometer and gyroscope to collect data while a user is biking.
3. Integrates external map data (for routing and visualization) and an external weather API.
4. Maintains a central database of trips, path segments, segment reports and merged path status.

2.1.1 Scenarios

Scenario 1: Comprehensive Trip Recording (Online, Offline and Edge Cases)
User A, a registered user commutes to work. Before starting, they open the BBP app and tap *Start Trip*. The app requests necessary location permissions; if granted, it begins sampling GPS points.

- **Live Tracking:** As User A rides, the application displays a live map, elapsed time, approximate distance, and current speed.
- **Handling Signal Loss:** During the ride, User A passes through a tunnel. GPS accuracy drops and location updates pause. The application continues the trip timer but displays a *Poor GPS* warning. When the signal returns, the application resumes sampling, marking the interpolated segment as *low confidence*.
- **Pausing:** User A stops for coffee and manually pauses the trip. The system records this interval separately to calculate moving speed versus total duration.
- **Completion:** Upon reaching the destination, User A taps *Stop Trip*. BBP computes final statistics (distance, average and maximum speed, and duration) and saves the trip to the user's history. If the device is online, weather data for the corresponding time and location is retrieved and appended to the trip log.

Scenario 2: Automated Obstacle Detection and Sensor Management
User B has enabled *Automatic Path Contribution.*

- **Data Acquisition:** When they start a trip, BBP begins background sampling of accelerometer and gyroscope data.
- **Detection:** During the ride, the sensors detect repeated vertical spikes (indicating a rough surface) and a strong jolts (indicating a pothole). BBP flags these locations as *candidate obstacles* with a specific confidence score.
- **Permission Revocation:** Midway through the ride, User B decides to revoke sensor permissions for privacy. BBP immediately stops sensor sampling and marks the contribution data as incomplete, though GPS recording continues for their personal log.
- **User Confirmation:** After the trip, User B is presented with a review screen of the candidate obstacles. They confirm the true positive (the pothole) and reject a false positive (caused by hopping a curb). Only the confirmed data is uploaded as a publishable contribution.

Scenario 3: Manual Path Contribution, Publishing and Maintenance
User C is familiar with a specific riverside track.

- **Creation:** They access *Contribute Paths* to map a route that is not yet in the system. They draw the track on the map (or select existing segments) and assign a status to each section-marking the majority as *Optimal* and a damaged section as *Requires Maintenance*.
- **Detailing:** User C places a specific obstacle marker on the damaged section with a description (e.g., *Deep pothole*).
- **Publishing:** They toggle the contribution to "Publishable," allowing the community to see this data.
- **Editing:** Weeks later, after the city repairs the track, User C returns to their previous contribution history, edits the segment status from "Requires Maintenance" to "Optimal," and submits the update. This new report is treated as fresh evidence by the system.

Scenario 4: Smart Route Search and Personalization
User D (a guest or registered user) wants to find a safe route to the city park.

- **Search & Scoring:** They enter an origin and destination. BBP queries the database and calculates scores for potential routes based on **Path Quality** (surface status, obstacles, data freshness) and **Efficiency** (distance, number of turns).
- **Preferences:** User D sets a specific preference to *Prioritize Smoothness*, indicating a willingness to cycle up to 20% longer to avoid rough terrain.
- **Results:** The system presents three options. Route A is short but rough; Route B is longer but *Optimal.* BBP ranks Route B higher based on User D's preferences, displaying a profile for each (e.g., *Route B: +0.12 quality score, -0.05 effectiveness*). User D selects Route B to view the specific map details.

Scenario 5: Data Aggregation and Privacy Controls
User Eand other community members submit data that the system must evaluate and merge.

- **Conflict Resolution:** User E reports a segment as *Requires Maintenance*, while another user reports it as *Optimal*. BBP considers report timestamps, user confirmations, and overall data freshness to resolve the conflict.
- **Privacy-First Sharing:** User E records a trip for personal statistics but does not want to share their exact commute route. They choose the *Publish Confirmed Obstacles Only* option. BBP extracts the confirmed pothole data and segment status for the community map but discards the full trip polyline (the specific path taken) from the public database to protect User E's privacy.

2.1.2 Domain Class Diagram

The Domain Class Diagram as shown in Fig. 1 for the BBP project models the ecosystem of cyclists, their activities, and the physical road infrastructure they navigate. It visually represents how Users interact with the system to record Trips and generate crowd-sourced data about Path Segments and Obstacles.

1. **User Management**
    - **User (Abstract Class):** The base entity representing any person interacting with the system, containing shared attributes like name, email, and credentials.
        - **RegisteredUser:** An authenticated user who can record trips, contribute reports, manage preferences, and access personal history.

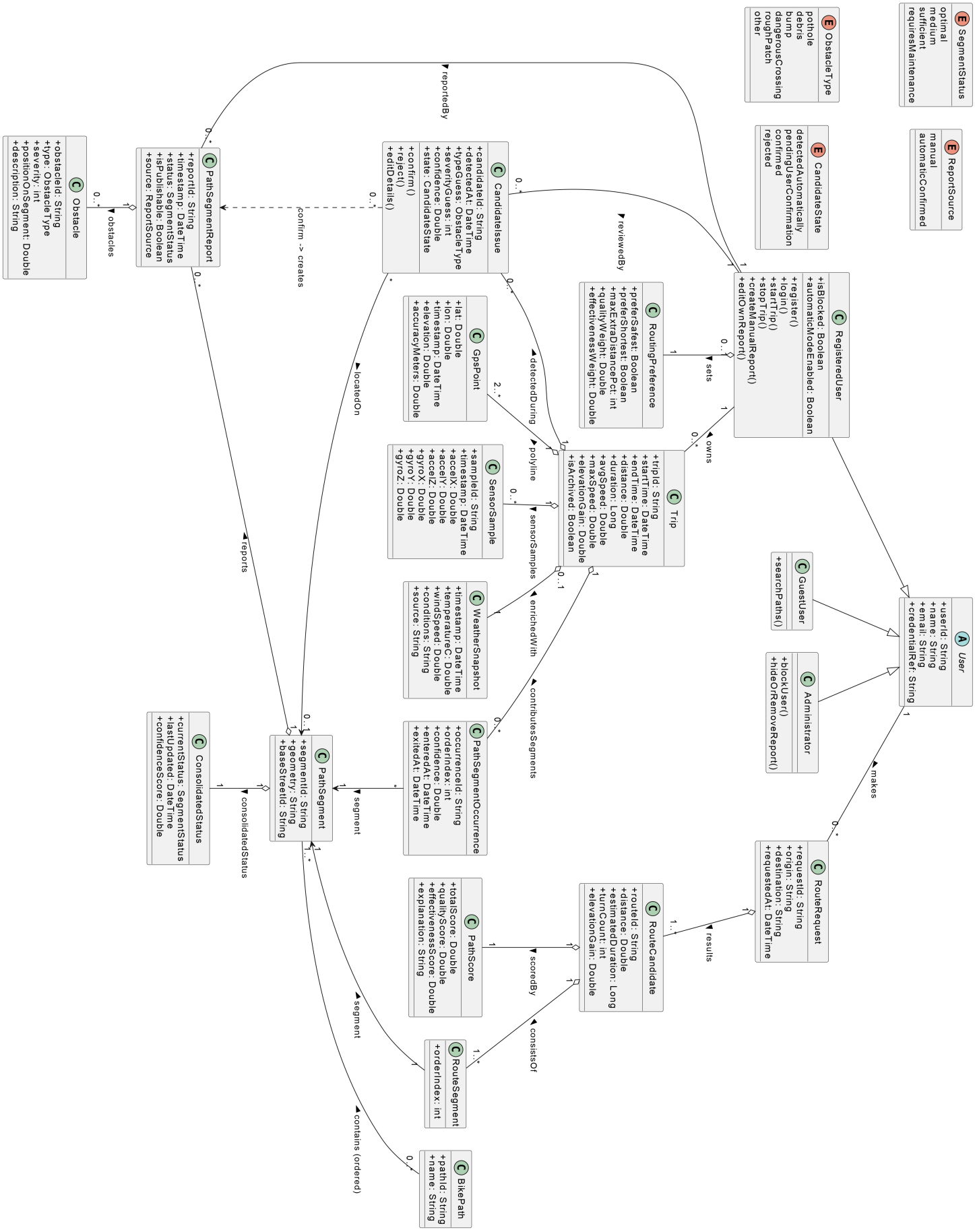

Figure 1: Domain Class Diagram.

  - **GuestUser** An unauthenticated user limited to searching for paths without storing history or contributing data.
  - **Administrator:** An unauthenticated user limited to searching for paths without storing history or contributing data.

2. **Trip & Data Acquisition**

   - **Trip:** Represents a recorded journey, storing aggregate statistics (distance, duration, speed) and linking to the specific data collected during the ride.
   - **RoutingPeference:** Stores a registered user's specific settings for route calculation, such as prioritizing safety or limiting extra distance.
   - **GpsPoint:** A single geographical data point recorded during a trip, containing coordinates, elevation, and timestamp.
   - **SenorSample:** Raw data captured from device sensors (accelerometer, gyroscope) during a trip to detect surface irregularities.
   - **WeatherSnapshot:** A record of environmental conditions (temperature, wind, etc.) fetched for a specific trip's time and location.

3. **Path & Routing**

   - **RouteRequest:** Represents a user's query for a path, defining the origin, destination, and the time the request was made.
   - **RouteCandidate:** A potential path calculated by the system in response to a request, containing estimated metrics like duration and turn count.
   - **RouteSegment:** An ordered portion of a route candidate that links to a specific physical path segment.
   - **PathScore:** A calculated rating for a specific route candidate, breaking down the quality and effectiveness scores with an explanation.
   - **BikePath:** A logical grouping of path segments representing a named track or street (e.g., "Riverside Track").
   - **PathSegment:**The fundamental unit of the road network (a specific stretch of physical geometry) that holds the consolidated status.
   - **PathSegmentOccurence:** Links a recorded Trip to the specific PathSegments it traversed, tracking entry and exit times for that segment.

4. **Reporting & Consolidation**

   - **PathSegmentReport:** A user-submitted report on a specific segment's condition, which acts as input for the system's consolidation logic.
   - **ConsolidatedStatus:** The system-calculated final status of a path segment, derived from aggregating multiple user reports and confidence scores.
   - **CandidateIssue:** A potential problem (like a pothole) detected automatically by sensors, waiting for user review and confirmation.
   - **Obstacle:** A confirmed physical obstruction (e.g., pothole, debris) located on a path segment, created manually or via confirmed candidate issues.

5. **Enumerations (Value Types)**

   - **SegmentStatus:** Defines the condition of a path (e.g., optimal, medium, sufficient, requires-Maintenance).

- **ReportSource:** Indicates whether a report was created manually or confirmed from automatic detection.
- **ObstacleType:** Categories of obstacles (e.g., pothole, debris, bump, dangerousCrossing).
- **CandidateState:** Tracks the lifecycle of an automated detection (e.g., detectedAutomatically, pendingUserConfirmation, confirmed, rejected).

2.1.3 State Diagrams

Trip Recording State Diagram
The diagram as shown in Fig. 2. defines the user and system states during a Trip Recording session. It traces the flow from *Idle* to *Recording* which includes sub-states to handle GPS signal quality (*High Accuracy* vs. *Low Signal*) and *Paused* states. Upon stopping the trip, the system enters a "Processing" composite state to compute statistics and fetch weather data (depending on connectivity) before finally transitioning to "Saved to History".

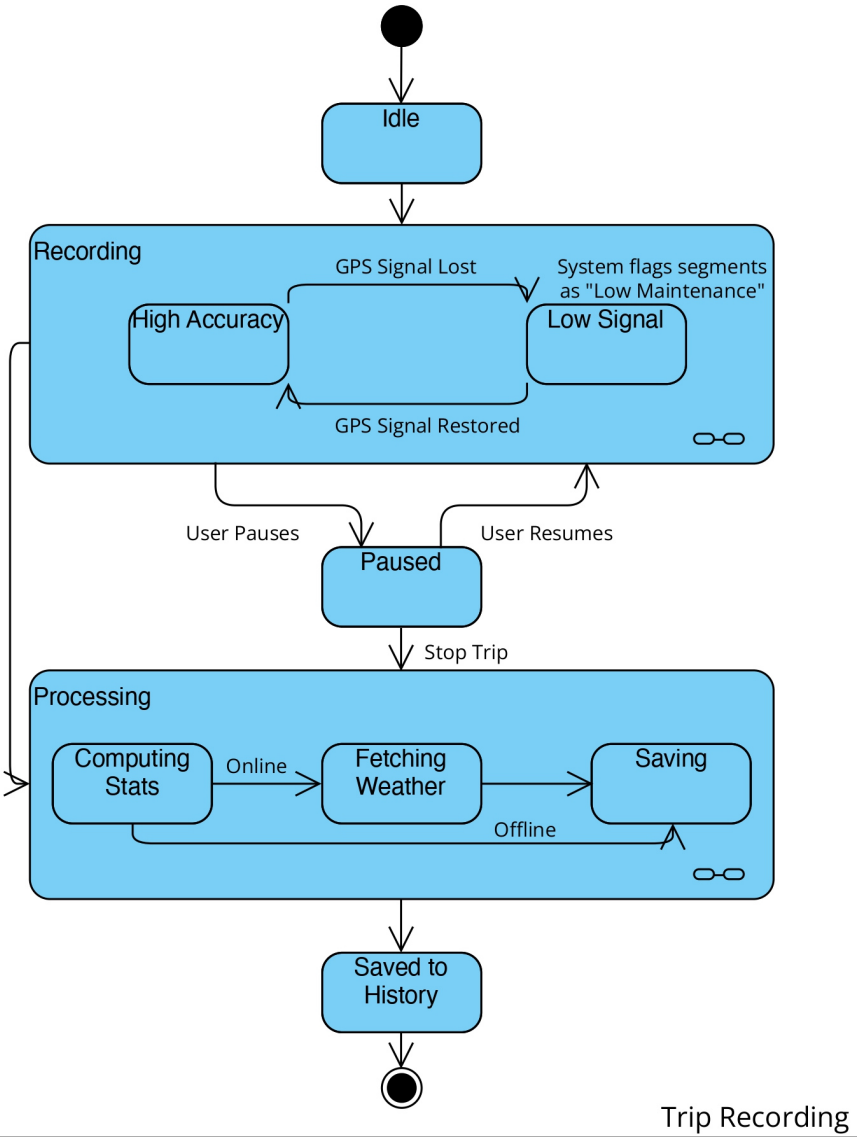

Figure 2: State Diagram for Trip Recording.

Candidate Issue Lifecycle

The diagram as shown in Fig. 3 outlines the workflow for Automated Path Information, specifically tracking a potential obstacle detected by sensors. The process begins with a *Sensor Spike Detected* which remains *Hidden From User* during the *Internal Detection* phase while the ride is ongoing. Once the trip ends, the state shifts to *Awaiting User Action,* requiring the user to Confirm, Edit or Reject the candidate to determine if it becomes *Publishable, Private,* or is *Discarded* entirely.

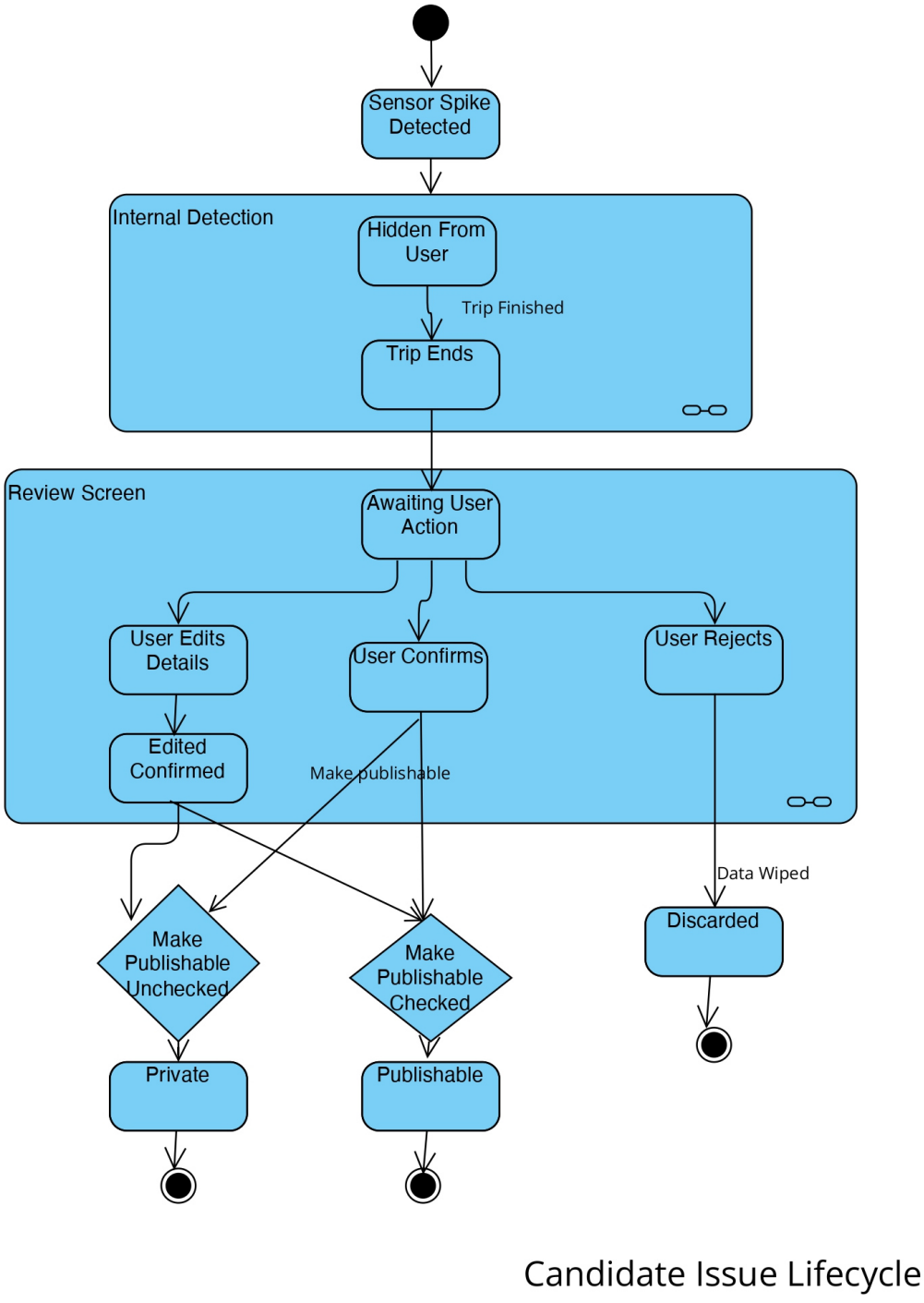

Figure 3: State Diagram for Candidate Issue Lifecycle.

Path Segment Consenses

The diagram as shown in Fig. 4 models the lifecycle of a Consolidated Status for a specific road segment. It illustrates how the system transitions a segment from an *Unknown Status* to specific classifications like *Optimal, Medium, or Requires Maintenance* based on incoming user reports. It demonstrates the merging logic where *fresh* positive or negative reports trigger state changes, reflecting the community consensus over time.

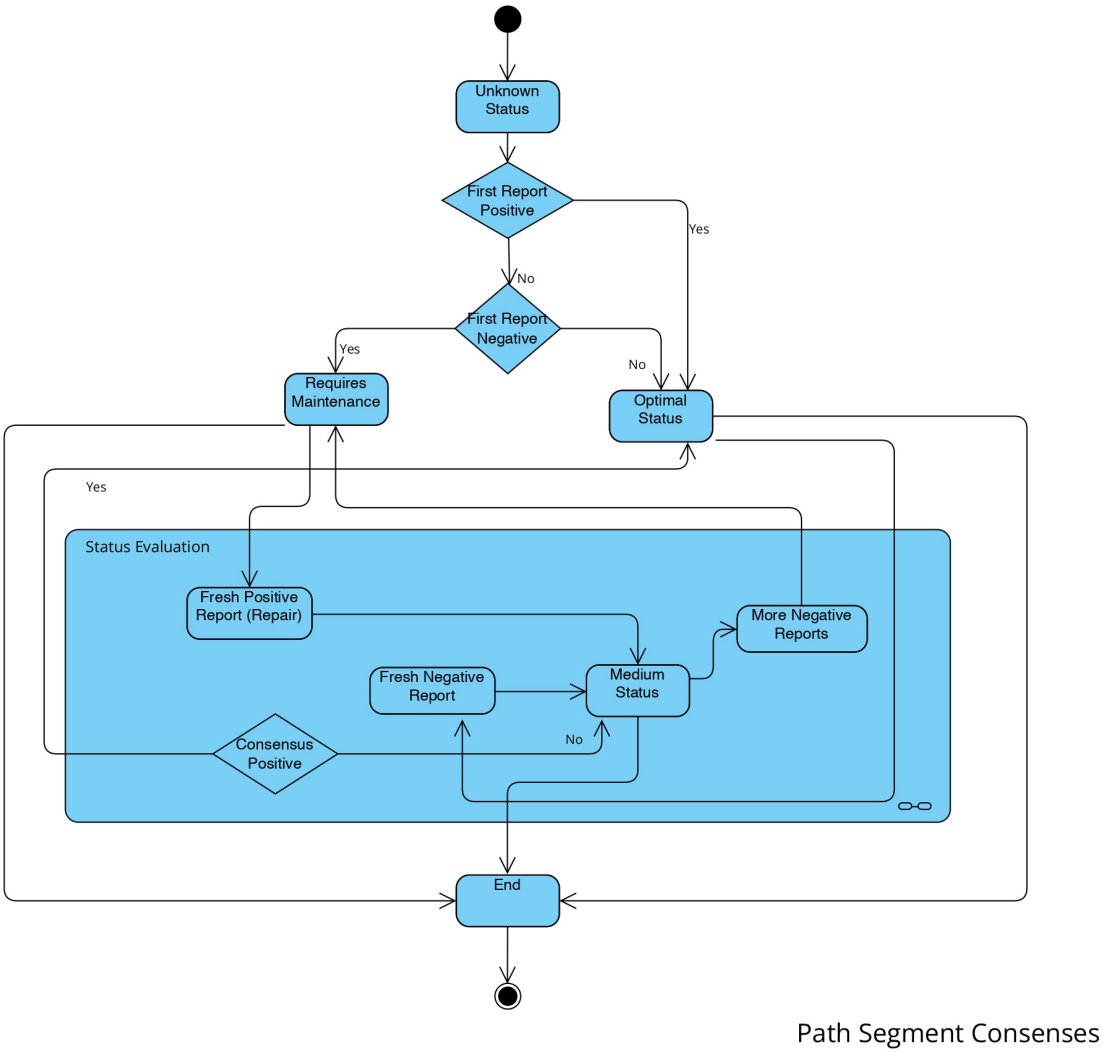

Figure 4: State Diagram for the Path Segment Consensus Process.

2.2 Product Functions (High Level)

The system is designed around the following core functional pillars:

1. **User Account & Session Administration** handles the full lifecycle of user identity. This includes secure registration and authentication (login/logout), encrypted storage of user credentials, and self-service password recovery mechanisms to ensure account security and accessibility.

2. **Permission & Consent Handling** Manages the request and revocation of runtime permissions necessary for core functionality. This specifically governs access to sensitive device capabilities, including fine-grained location services (GPS) and motion sensors (accelerometer/gyroscope), ensuring user consent is explicitly granted.

3. **Real-Time Trip Acquisition** Performs active data collection during a ride. This involves continuous GPS sampling to plot routes and optional background sensor sampling to detect surface irregularities. The system computes real-time statistics (speed, duration, distance) and ensures data is persisted locally on the device to prevent loss during connectivity drops.
4. **Post-Trip Review & Management** Provides a comprehensive interface for users to access their personal data. Features include a history list filtered by date, a detailed view of specific trips with map visualizations, and data management options such as deleting records or exporting trip data for external use.
5. **Manual Contribution Interface** Enables users to actively map and rate infrastructure. Users can select existing paths or draw new segments on the map, assign specific condition statuses (e.g., *Optimal* vs. **Damaged**), and drop markers for specific obstacles. This workflow includes capabilities to publish, unpublish, or edit their previous reports.
6. **Automated Contribution Workflow** Utilizes sensor data to passively crowdsource road conditions. The system detects candidate issues (anomalies) via motion algorithms, presents them to the user in a review UI for confirmation or rejection, and processes verified data for publication.
7. **Intelligent Route Search & Scoring** A search engine that calculates optimal biking paths between an origin and destination. It retrieves path data, applies a weighted scoring algorithm (balancing surface quality, obstacles, and distance efficiency), ranks the results, and visualizes the routes with clear explanations for the calculated scores.
8. **Data Aggregation & Consensus Engine** The backend logic responsible for reconciling conflicting or overlapping user reports. This function periodically recomputes segment statuses by applying logic such as *freshness weighting* (prioritizing recent data), majority consensus, and confidence scoring to derive a single, reliable status for every road segment.
9. **System Moderation & Safety** Administrative tools designed to maintain platform integrity. This includes the ability to block abusive users, hide or remove erroneous reports, and maintain an audit log of administrative actions.

2.3 User Characteristics

1. **Registered Cyclist**
   - Has a basic familiarity with smartphones and maps.
   - Expects simple interactions like starting a ride, tapping on a map and filling small forms.
   - Motivated to log personal activity and contribute to community data.
2. **Guest User**
   - May have no account; just wants to see safe paths between two points.
   - Needs a very simple UI: origin, destination and path list/map.
3. **System Administrator**
   - Manages users (e.g., blocking abusive ones), monitors data quality, may adjust merging parameters.
4. **Non-human actors**
   - Map service (e.g., routing and tiles).
   - Weather service (meteorological data provider)

2.4 Assumptions, Dependencies and Constraints

2.4.1 Regulatory Policies

1. BBP processes personal data (location history of trips, user accounts), so it must comply with applicable data protection laws (e.g., GDPR in the EU). Identifiable trip data must not be published without user consent and sensitive data must be encrypted at rest and in transit.
2. Map and weather services must be used according to their licenses/terms of service.
3. BBP only provides advisory information; it does not replace local traffic regulations or official safety advisories.

2.4.2 Domain Assumptions

- **D1:** GPS accuracy is sufficient (e.g., within 10-20 meters) to associate positions with map segments.
- **D2:** Device sensors (accelerometer, gyroscope) are calibrated and reliable enough to detect strong pothole-like events.
- **D3:** Users generally act in good faith and report path status honestly.
- **D4:** External map and weather services are available and return correct data.
- **D5:** User's devices have an active network connection at least intermittently (for trip upload and weather fetching).
- **D6:** A bike path is defined as in the assignment: dedicated bike track or low-traffic, bike compatible road.
- **D7:** When multiple users provide conflicting information, a meaningful consolidated view can be derived using time and majority rules.
- **D8:** Users explicitly enable automatic sensor-based data collection before BBP accesses motion sensors for pothole detection.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

1. **Home/Dashboard (registered user):**
    - Shows quick buttons: *Start Trip*, *My Trips*, *Contribute Paths*, *Find Path*. Shows recent trips and short stats.

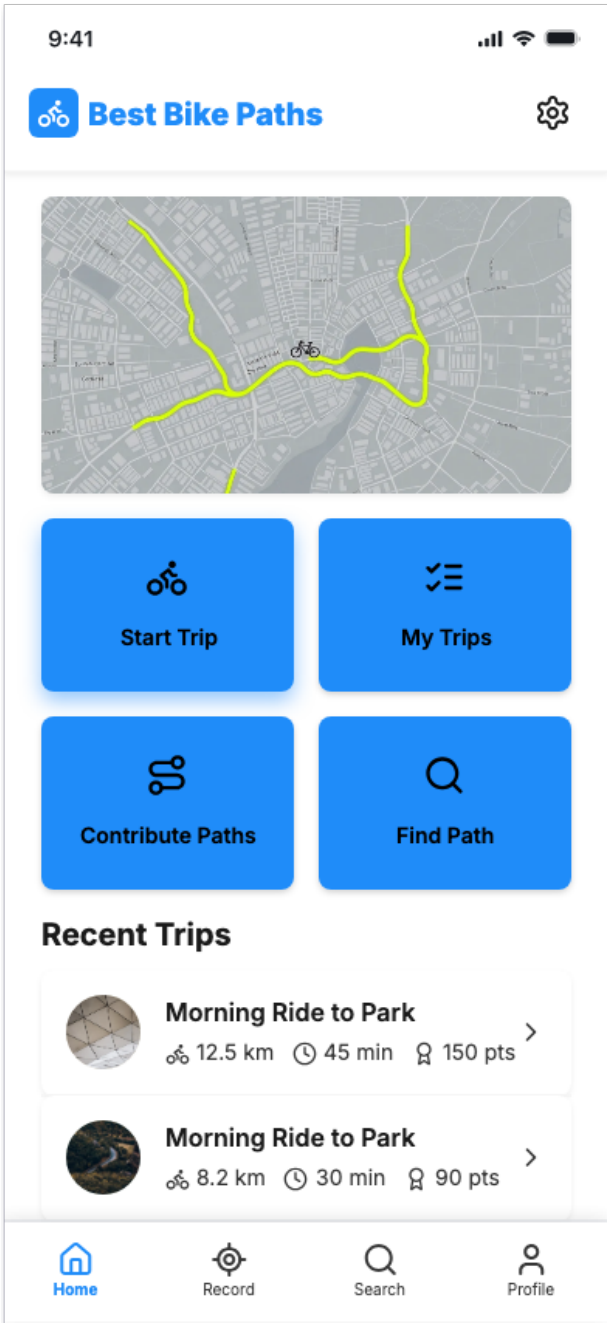

Figure 5: Dashboard

2. **Trip Recording Screen:**
    - Displays elapsed time, distance and current speed.
    - Shows the user's position on a map.
    - Contains buttons: Start, Pause/Resume, Stop.

Figure 6: Trip Recording Screen

3. **Trip Details Screen:**
    - Map with route polyline.
    - Trip statistics.
    - Weather summary.

- Optional toggle “use this trip to update path information”.

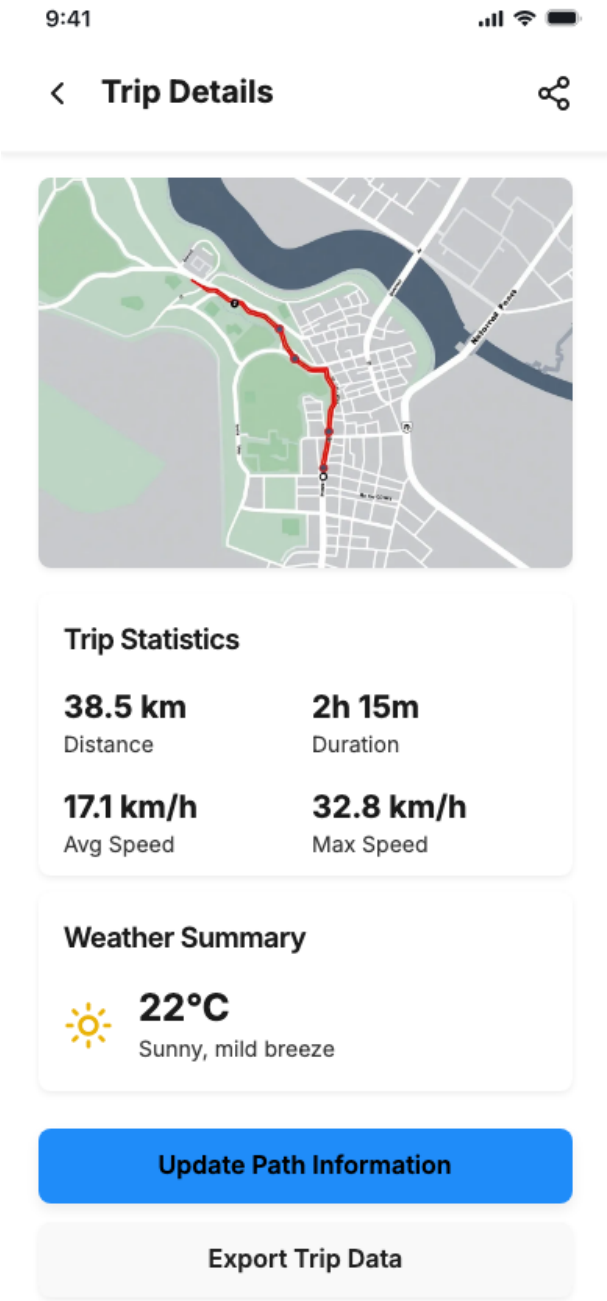

Figure 7: Trip Details Screen

4. **Contribute Paths (Manual) Screen**
    - Map where the user can draw/select a path or list of streets.
    - Form for assigning status to each segment and adding obstacles.
    - Checkbox or switch *Make this contribution publishable*.

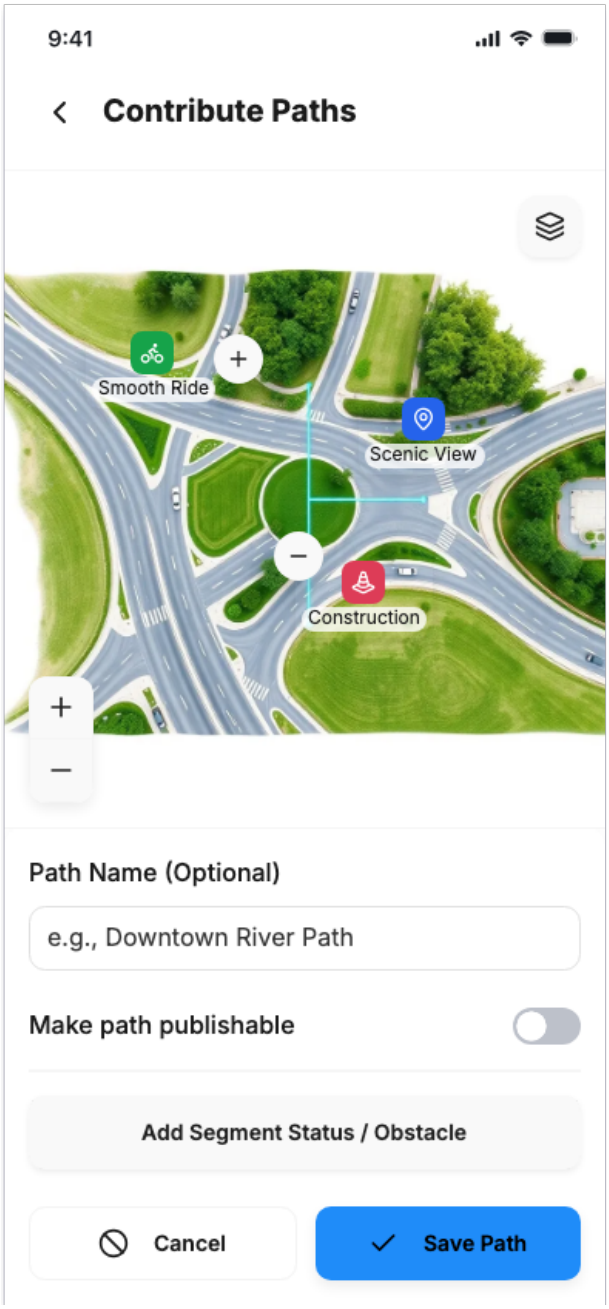

Figure 8: Contribute Path (Manual) Screen

5. **Automatic Detection Review Screen**
    - List or map of suspected obstacles detected during recent trips.
    - For each candidate: buttons *Confirm, Reject, Edit details.*

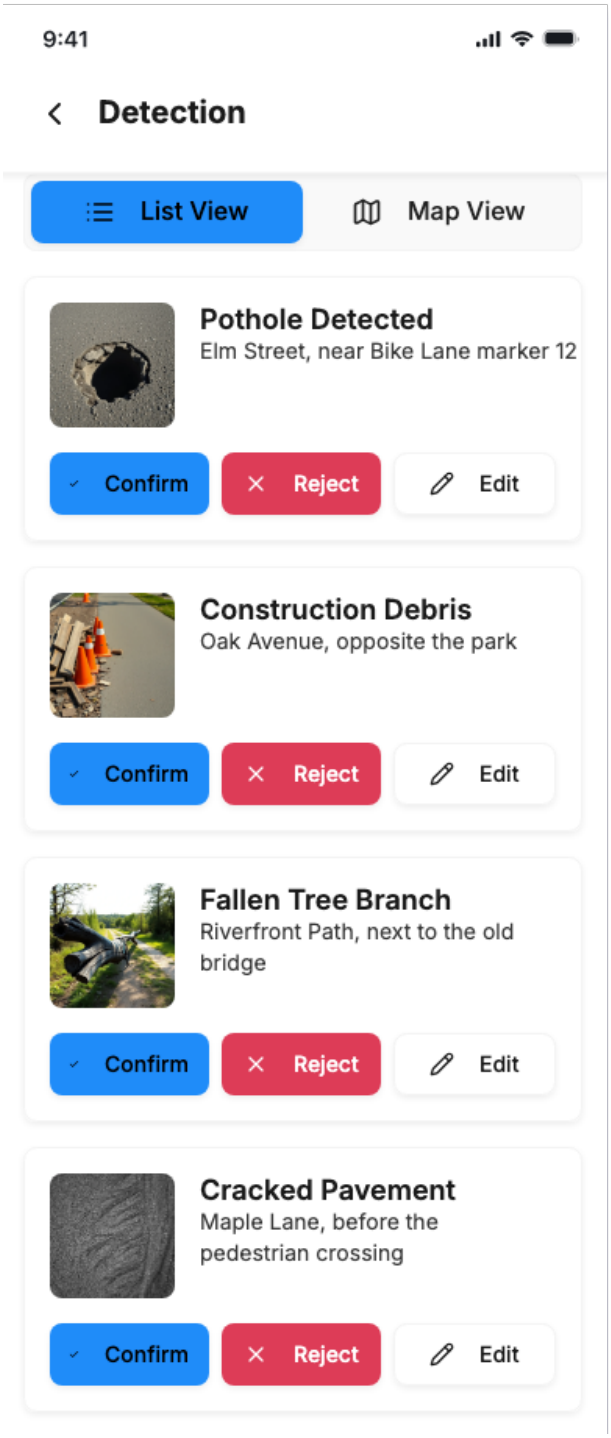

Figure 9: Automatic Detection Review

6. **Path Search Screen (Public)**
    - Input fields for origin and destination (address, point on map).
    - Optional filters (e.g., *prefer safest, shortest path, avoid unlit segments*).
    - List of candidate paths with score and summary (distance, expected quality).
    - Map visualization of selected path.

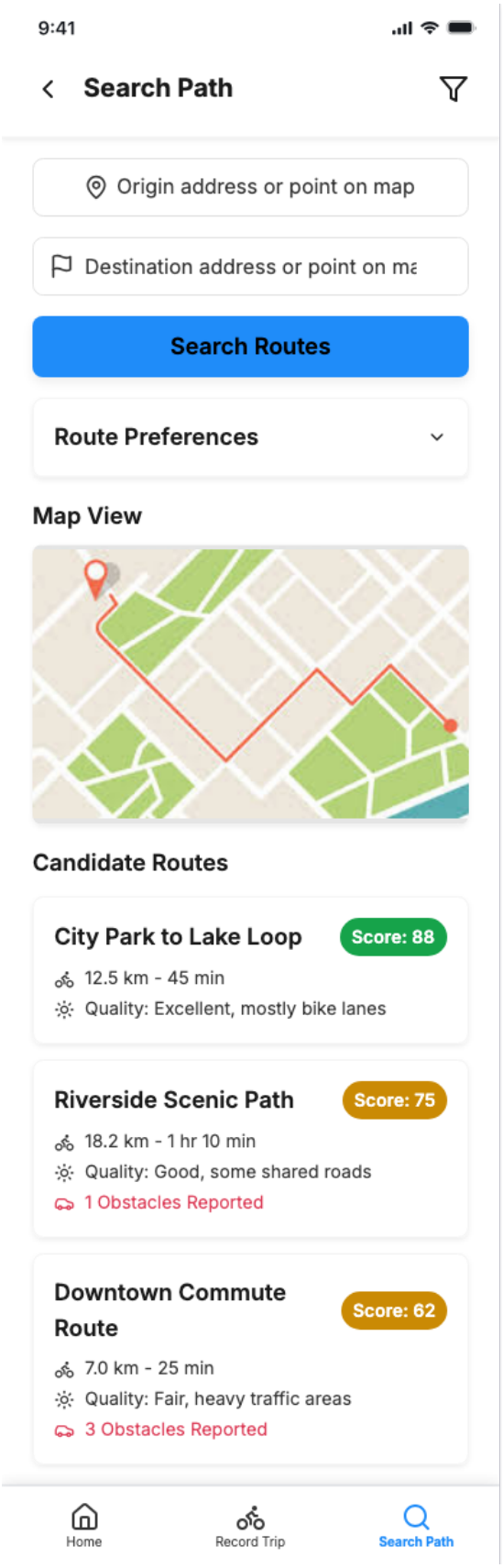

Figure 10: Path Search Screen

3.1.2 Hardware Interfaces

1. Device with:
    - GPS Receiver
    - Accelerometer and gyroscope (for automatic mode).
    - Network connectivity.
2. BBP accesses these sensors through the OS APIs (e.g.android/iOS location & sensor APIs).

3.1.3 Software Interfaces

1. **Map/Routing API:** For map tiles, geocoding (address <==> coordinates)and route computation.
2. **Weather API:** BBP calls the external weather service with time and location to enrich trips with weather snapshots.
3. **Persistent Storage (DB):** For users, trips, segments, reports, merged statuses.

3.1.4 Communication Interfaces

1. All communication between client and server uses HTTPS.
2. Calls to external APIs (maps, weather) also use secure HTTP endpoints.
3. Where live updates are needed (e.g., showing trip progress on web), WebSockets or equivalent may be used.

3.2 Functional Requirements

Trip Recording & Statistics

R1 - User registration The system shall allow a user to create an account using an email (or federated login) and password.

R2 - User login The system shall allow a registered user to authenticate and access their personal data.

R3 - Start trip recording The system shall allow a logged-in user to start recording a new trip, initializing GPS logging and timing.

R4 - Stop trip recording The system shall allow the user to stop an ongoing trip and save it as a completed trip.

R5 - Trip statistics After a trip is saved, the system shall compute and store at least distance, duration, and average speed.

R6 - Trip history The system shall allow a logged-in user to view the list of all recorded trips and open detailed views.

R7 - Trip weather enrichment For each trip, if the weather service is reachable, the system shall fetch and attach weather information based on time and location.

Manual Path Information

R8 - Manual path creation The system shall allow a logged-in user to select or draw a bike path using the map or by specifying street names.

R9 - Segment status assignment The system shall allow the user to set a status (optimal, medium, sufficient, requires maintenance, etc.) for each segment of a path.

R10 - Obstacle creation The system shall allow the user to create obstacles associated with a segment, specifying the obstacle type and a description.

R11 - Publishability The system shall allow the user to mark manual contributions as publishable or private.

R12 - Edit manual contributions The system shall allow users to edit or delete their previously submitted path reports.

Automatic path information

R13 - Enable/disable automatic mode The system shall allow a user to enable or disable automatic collection of sensor data for path quality detection.

R14 - Sensor data logging When automatic mode is enabled and a trip is being recorded, the system shall periodically sample accelerometer and gyroscope data and associate them with geographic positions.

R15 - Detection of candidate obstacles The system shall process sensor data and detect significant events (e.g., strong jolts) as candidate potholes or rough path segments.

R16 - Candidate list presentation After a trip ends, the system shall present the user with a list or map of candidate issues detected during that trip.

R17 - User confirmation/correction For each candidate, the user shall be able to confirm it, reject it, or edit its details (type, severity, or position) before it becomes a report.

R18 - Conversion into reports Confirmed candidates shall be stored as *PathSegmentReport* objects associated with path segments. Rejected candidates shall not be stored as publishable data.

Path search and visualization

R19 - Public path search The system shall allow any user, whether registered or not, to specify an origin and destination and request bike paths.

R20 - Route computation The system shall compute one or more route candidates using external map and routing services in combination with the internal path database.

R21 - Path scoring The system shall compute a score for each candidate path based on:

- Consolidated statuses of included segments.
- Severity and number of obstacles.
- Route effectiveness (e.g., distance, elevation, number of turns).

R22 - Ordered path list The system shall present candidate paths ordered by decreasing score.

R23 - Path visualization The system shall visualize a selected path on a map with overlays representing segment status and reported obstacles.

Data Merging

R24 - Segment report storage The system shall store all PathSegmentReports with timestamps and user identifiers.

R25 - Periodic merging The system shall periodically recompute the consolidated status of each segment based on all available reports.

R26 - Freshness handling The merging process shall weigh newer reports more heavily than older ones.

R27 - Majority handling If multiple reports with similar freshness disagree, the consolidated status shall follow the majority assessment (e.g., by count or weighted count).

Administration and data quality

R28 - User Blocking The system shall allow administrators to block users who repeatedly submit obviously false data.

R29 - Data Removal Administrators shall be able to remove or hide problematic reports.

3.2.1 Use Case Diagrams

The diagrams that from Fig. 11 to Fig. 14 are the use case diagrams shown below:

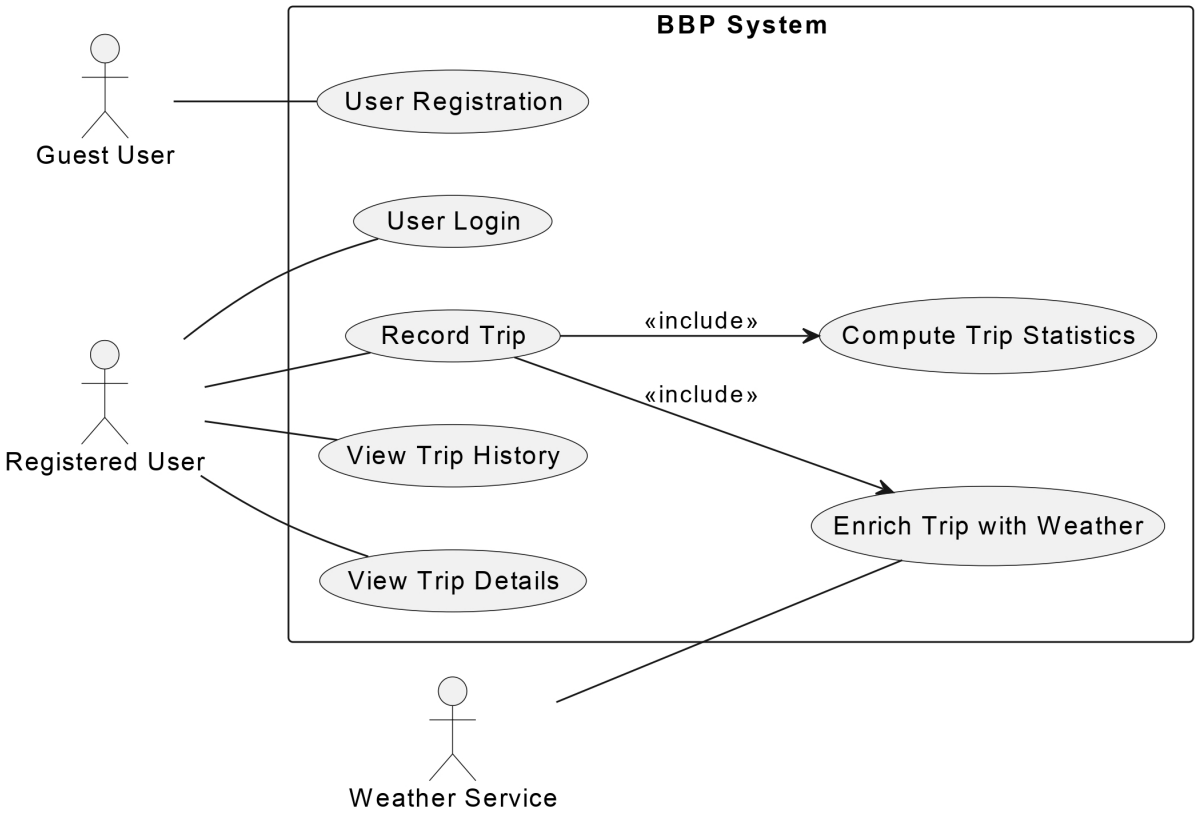

Figure 11: Use Case Diagram for Account Management.

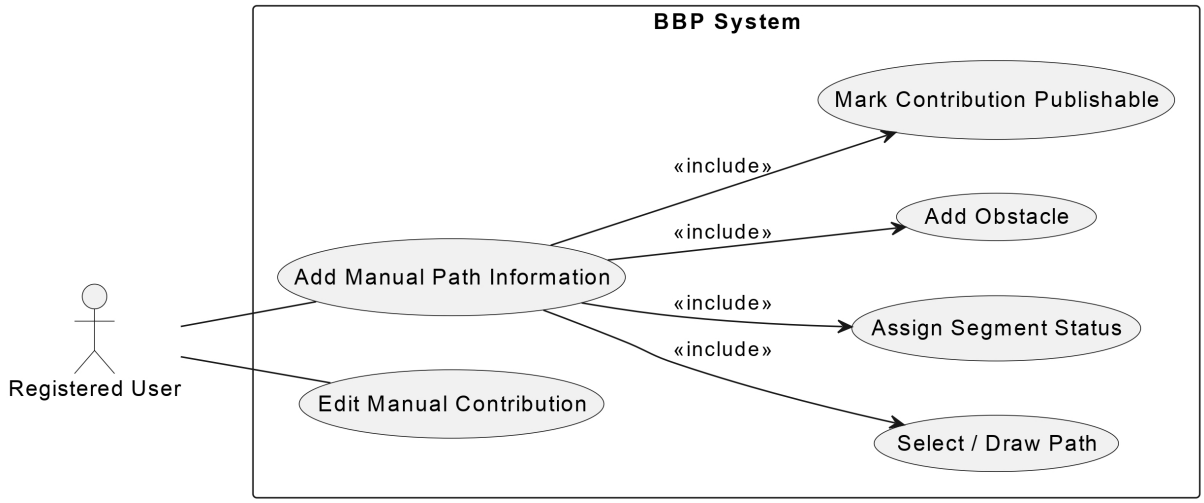

Figure 12: Use Case Diagram for Manual path Contribution.

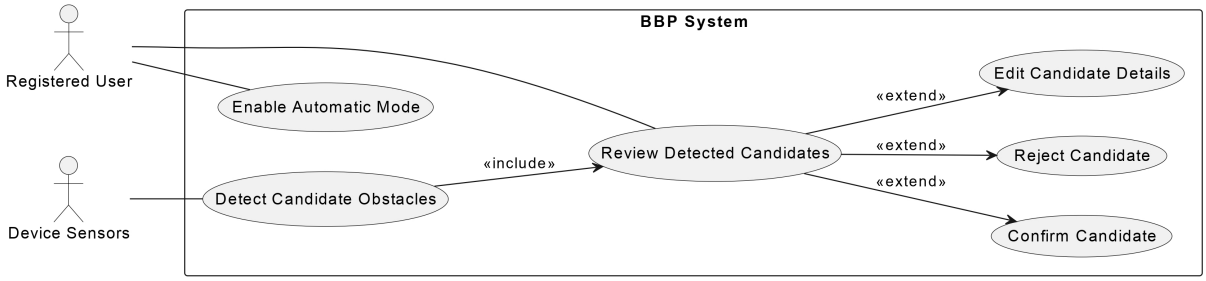

Figure 13: Use Case Diagram for Automatci Path Information and Confirmation.

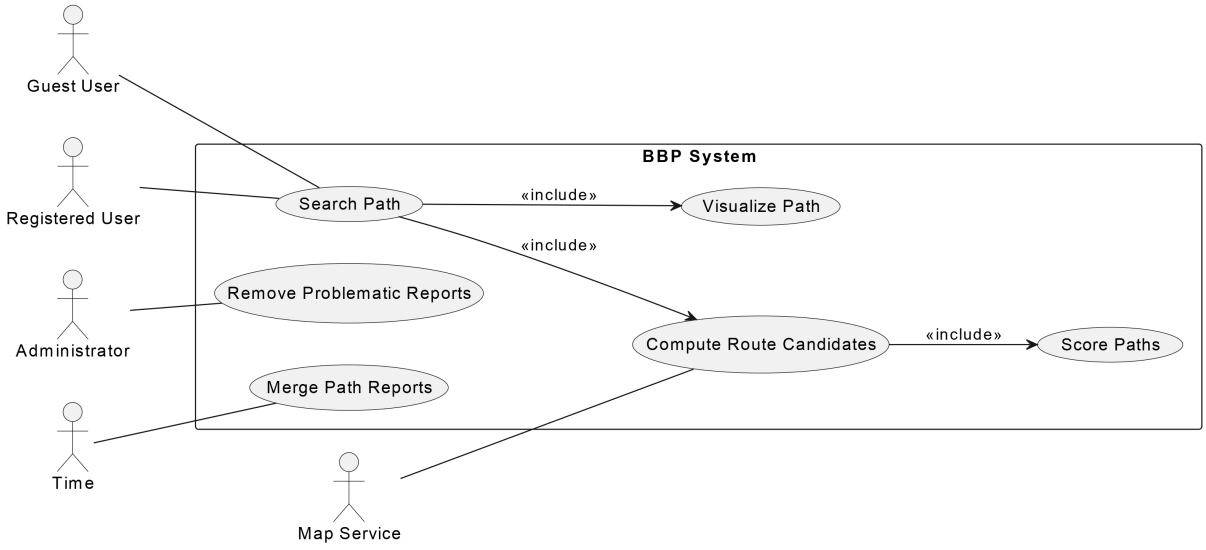

Figure 14: Use Case Diagram for Path Search, Scoring and Merging.

3.2.2 Use Cases

The use cases of Best Bike Path (BBP) are shown as Table. 2 to Table. 8

UC1: User Registration

Name	**User Registration**
Actors	Visitor
Entry Condition	Visitor opens the BBP application.
Event Flow	(a) The visitor selects *Sign Up*. (b) The system displays the registration form requesting email, password, and optional name. (c) The visitor enters the required information and confirms the registration. (d) The system validates the provided data and creates a new user account. (e) The system confirms successful registration and may automatically log the user in.
Exit Condition	A new user account exists.
Exceptions	Email already used; invalid email format; weak password.

Table 2: Use Case 1: User Registration

UC2: User Login

Name	**User Login**
Actors	Registered User
Entry Condition	User opens the BBP application.
Event Flow	(a) The user selects *Login*. (b) The system prompts the user to enter credentials. (c) The user enters their email and password. (d) The system validates the provided credentials. (e) The system grants access and displays the user dashboard.
Exit Condition	The user is successfully logged in.
Exceptions	Invalid credentials; account blocked.

Table 3: Use Case 2: User Login

UC3: Record Trip

Name	**Record Trip**
Actors	Registered User
Entry Condition	User is logged in.
Event Flow	(a) The user taps *Start Trip*. (b) The system requests permission to access location data, if not already granted. (c) The system begins logging GPS points and the trip start time. (d) The user rides while the system continuously updates position and trip statistics. (e) The user taps *Stop Trip*. (f) The system stops logging, computes trip statistics, and saves the completed trip. (g) The system requests weather data and attaches it to the trip, if available.
Exit Condition	The completed trip is stored with statistics and, if available, weather information.
Exceptions	Location permission denied; GPS unavailable; network unavailable for weather data (trip is stored without weather information).

Table 4: Use Case 3: Record Trip

UC4: Add Manual Path Information

Name	**Add Manual Path Information**
Actors	Registered User
Entry Condition	User is logged in.
Event Flow	(a) The user opens *Contribute Paths*. (b) The system displays a map and tools for selecting streets or drawing a path. (c) The user defines one or more path segments. (d) For each segment, the user selects a path status. (e) The user optionally adds obstacles to segments, specifying type and severity. (f) The user chooses whether the contributed information is publishable. (g) The system saves the path segments and associated reports.
Exit Condition	Path segment reports are created and stored; consolidated segment status may be updated.
Exceptions	Incomplete information (e.g., no segments selected); map service unavailable.

Table 5: Use Case 4: Add Manual Path Information

UC5: Automatic Detection Confirmation

Name	**Automatic Detection Confirmation**
Actors	Registered User
Entry Condition	User has completed a trip with automatic mode enabled.
Event Flow	(a) After the trip, the system computes candidate obstacles. (b) The system notifies the user that new candidates are available for review. (c) The user opens the review screen. (d) The system lists detected candidates with map positions and basic information. (e) For each candidate, the user selects *Confirm*, *Reject*, or *Edit*. (f) The system converts confirmed items into reports, discards rejected candidates, and saves any edits.
Exit Condition	Confirmed obstacles are stored as publishable or pending publication according to the user's choice.
Exceptions	User skips the review; in this case, candidate obstacles are not published.

Table 6: Use Case 5: Automatic Detection Confirmation

UC6: Search Path Between Origin and Destination

Name	**Search Path Between Origin and Destination**
Actors	Guest User, Registered User
Entry Condition	User is on the BBP main screen.
Event Flow	(a) The user enters an origin and destination as addresses or map points. (b) The user initiates the search. (c) The system geocodes the origin and destination. (d) The system computes one or more candidate routes. (e) For each route, the system retrieves consolidated segment statuses and reported obstacles. (f) The system computes scores for each route and sorts them accordingly. (g) The system displays a list of candidate routes and allows the user to select one. (h) The system displays the selected path on the map with detailed information.
Exit Condition	The user views at least one candidate path on the map.
Exceptions	Invalid addresses; no path found; external services unavailable.

Table 7: Use Case 6: Search Path Between Origin and Destination

UC7: Merging Path Reports

Name	**Merging Path Reports**
Actors	Time (scheduled job), Administrator (indirect observer)
Entry Condition	The system has accumulated multiple reports for path segments.
Event Flow	(a) At scheduled intervals, the merging procedure is triggered. (b) The system gathers all reports associated with each path segment. (c) The system groups reports by segment and sorts them by timestamp. (d) The system applies merging rules based on report freshness and majority. (e) The system updates each segment's consolidated status and last-updated date.
Exit Condition	Consolidated segment status reflects the latest known information.
Exceptions	None specific; the merging process may be skipped if no new reports are available.

Table 8: Use Case 7: Merging Path Reports

3.2.3 Sequence Diagrams

The diagrams from Fig. 15 to Fig. 21 represents use cases from UC1 to UC7

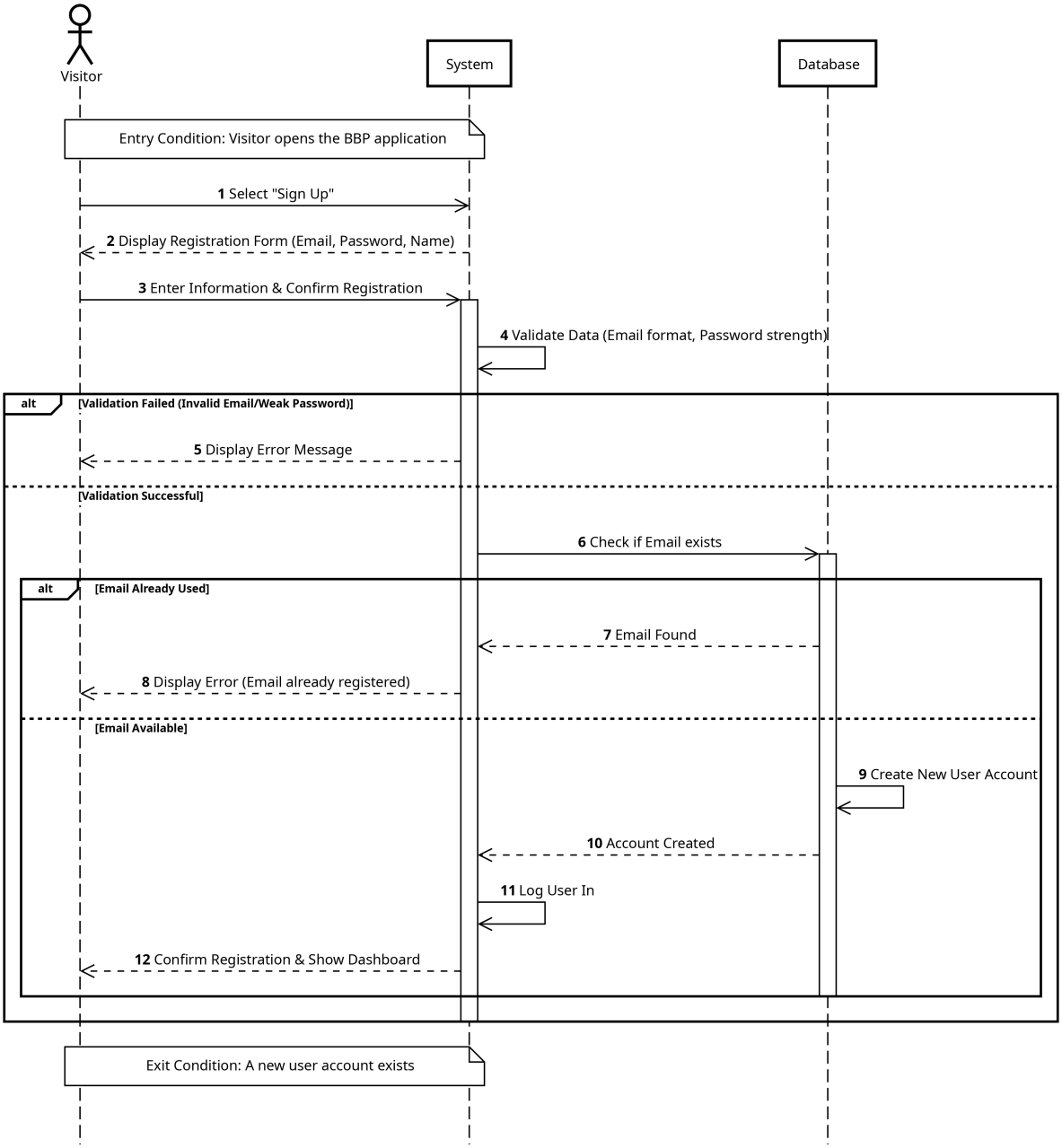

Figure 15: Sequence Diagram for UC1-User Registration.

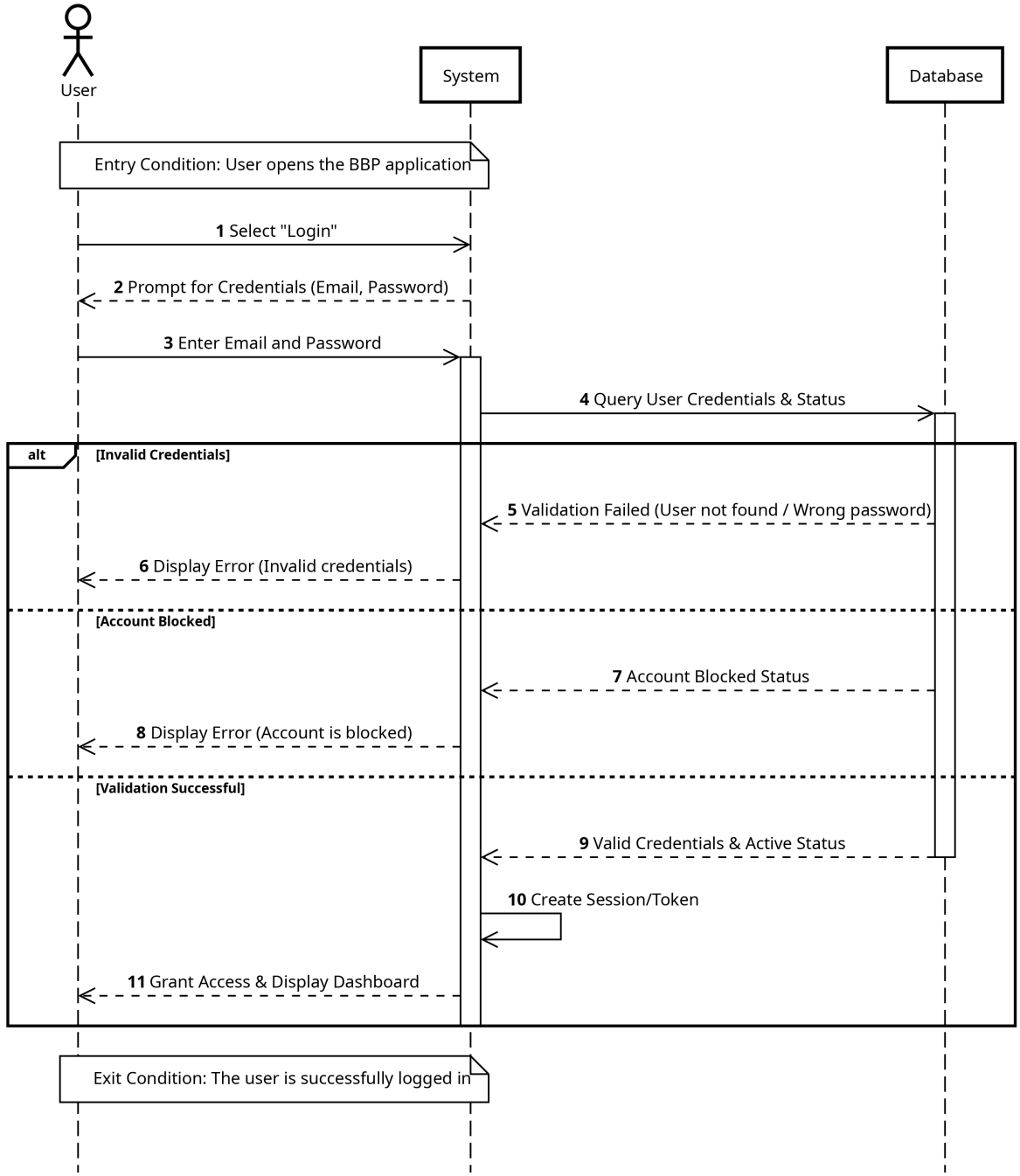

Figure 16: Sequence Diagram for UC2-User Login.

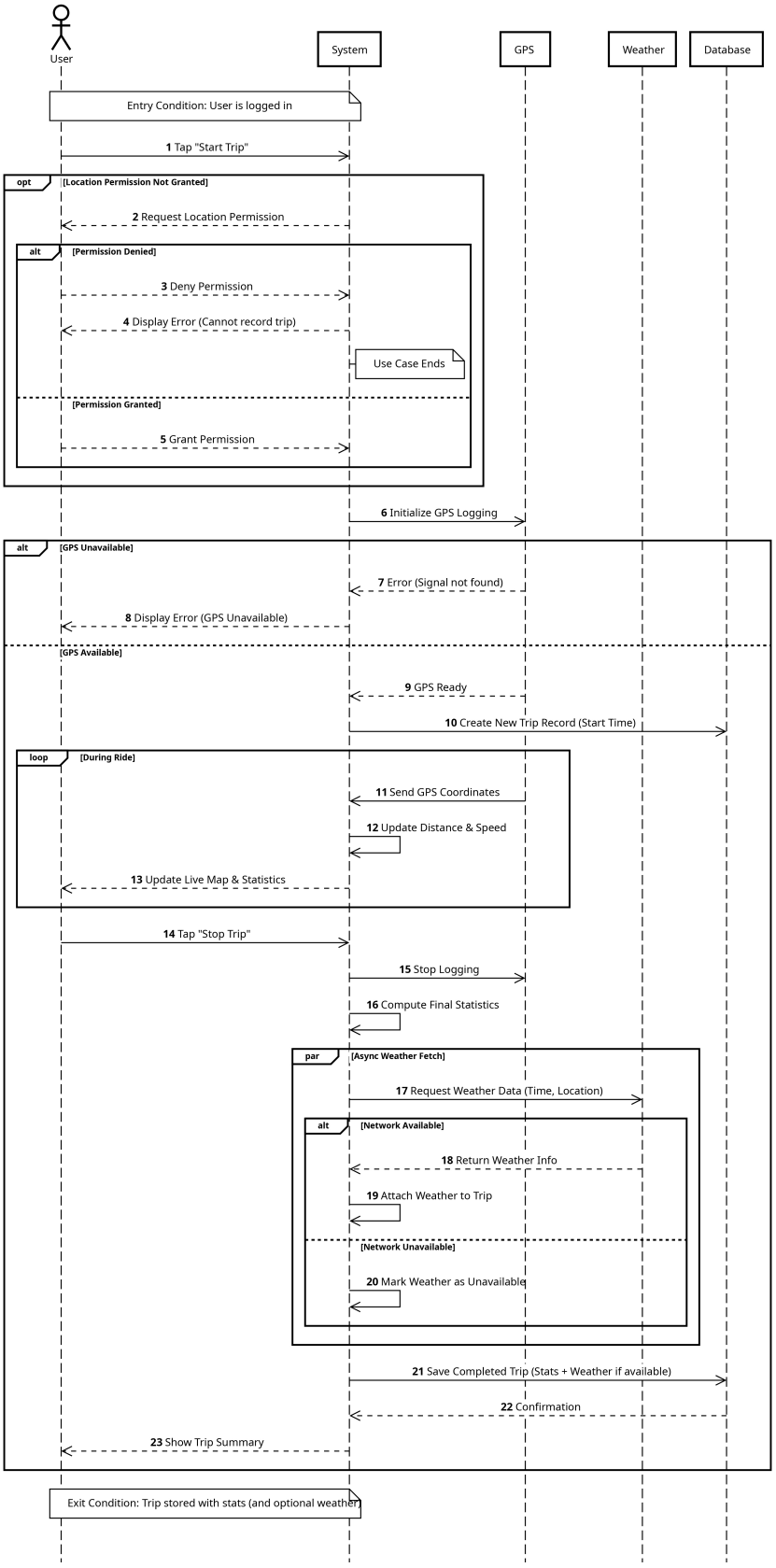

Figure 17: Sequence Diagram for UC3- Record Trip

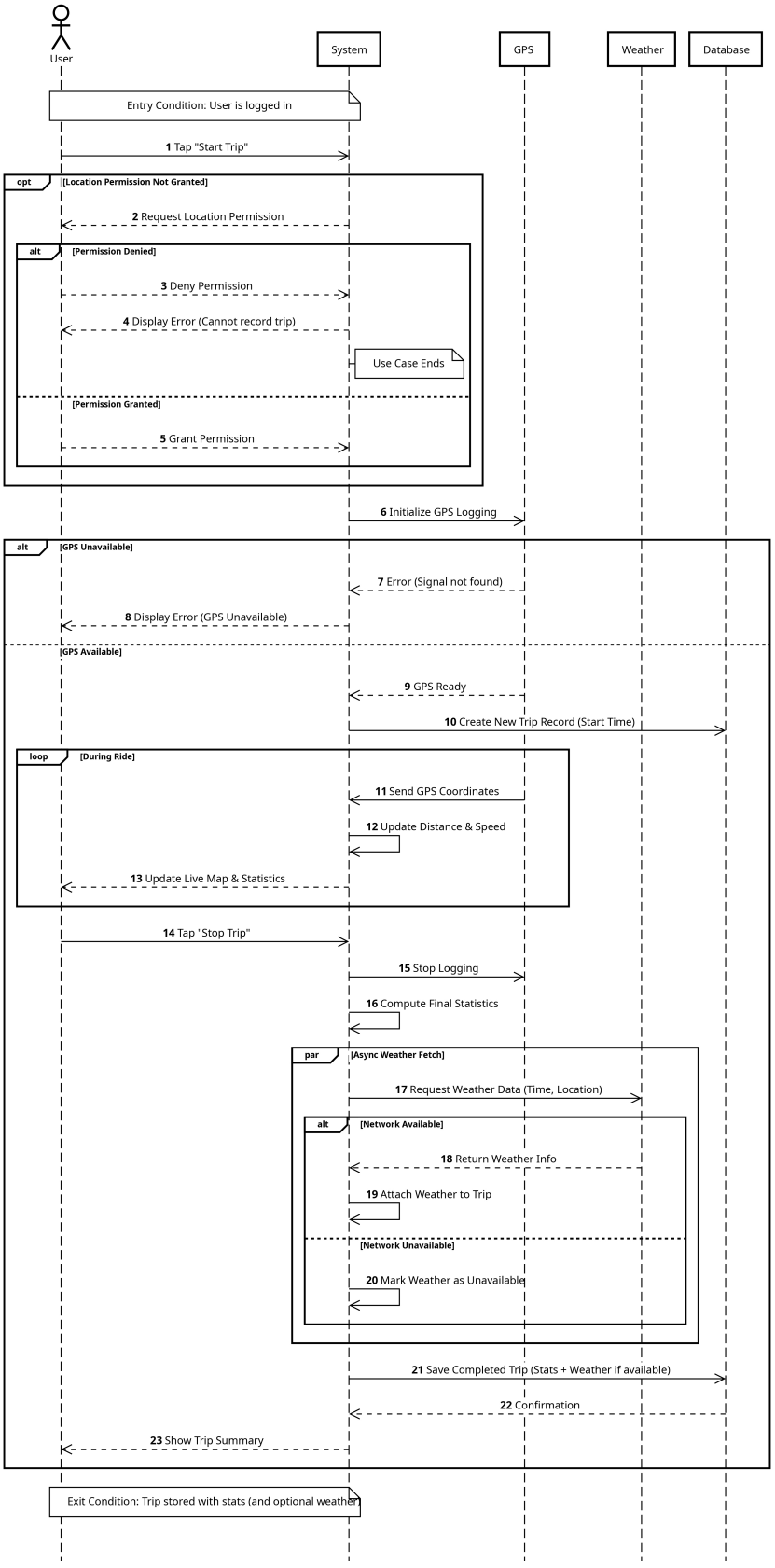

Figure 18: Sequence Diagram for UC4- Add Manual Path Information

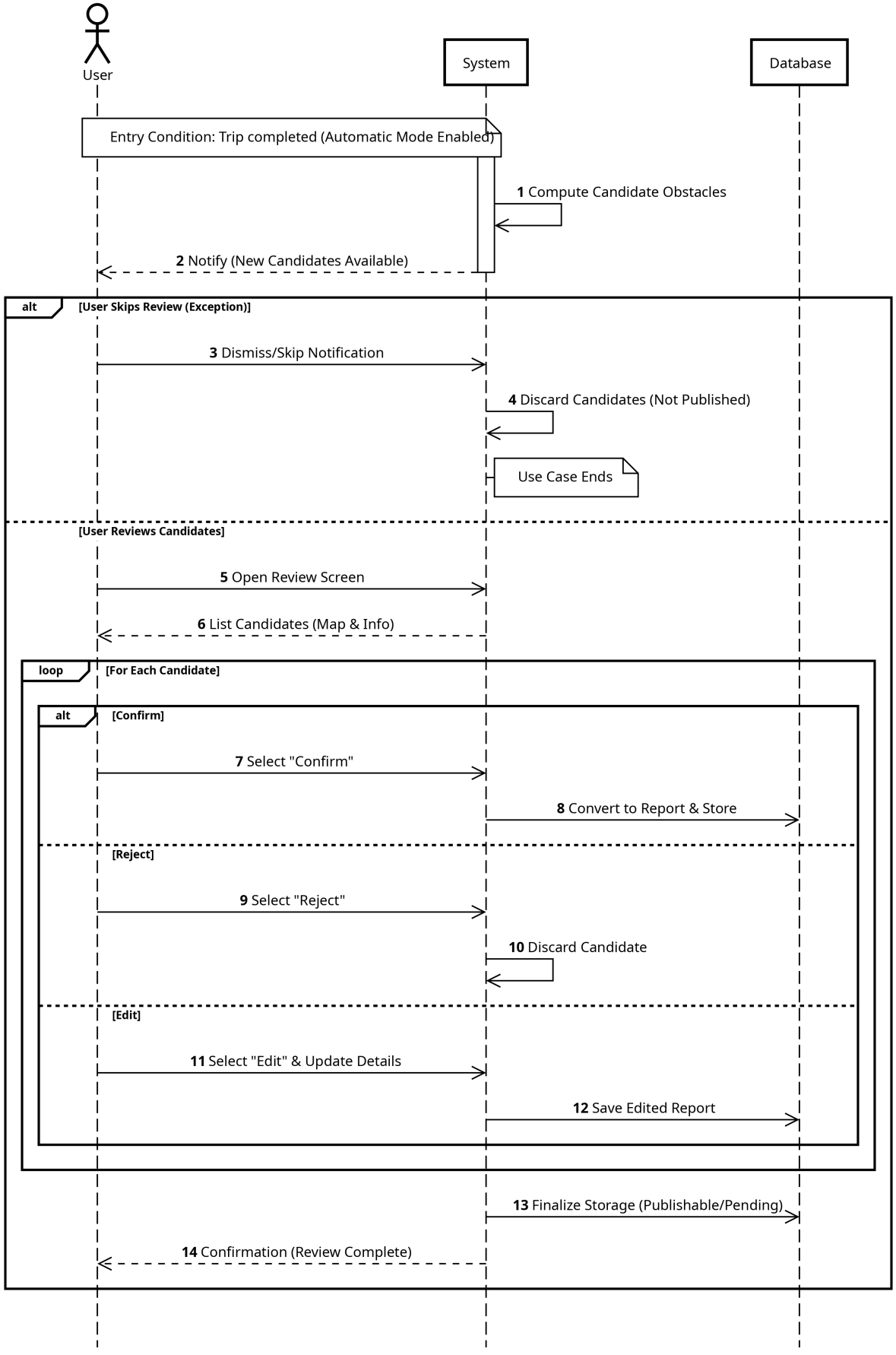

Figure 19: Sequence Diagram for UC5- Automatic Dectection Confirmation

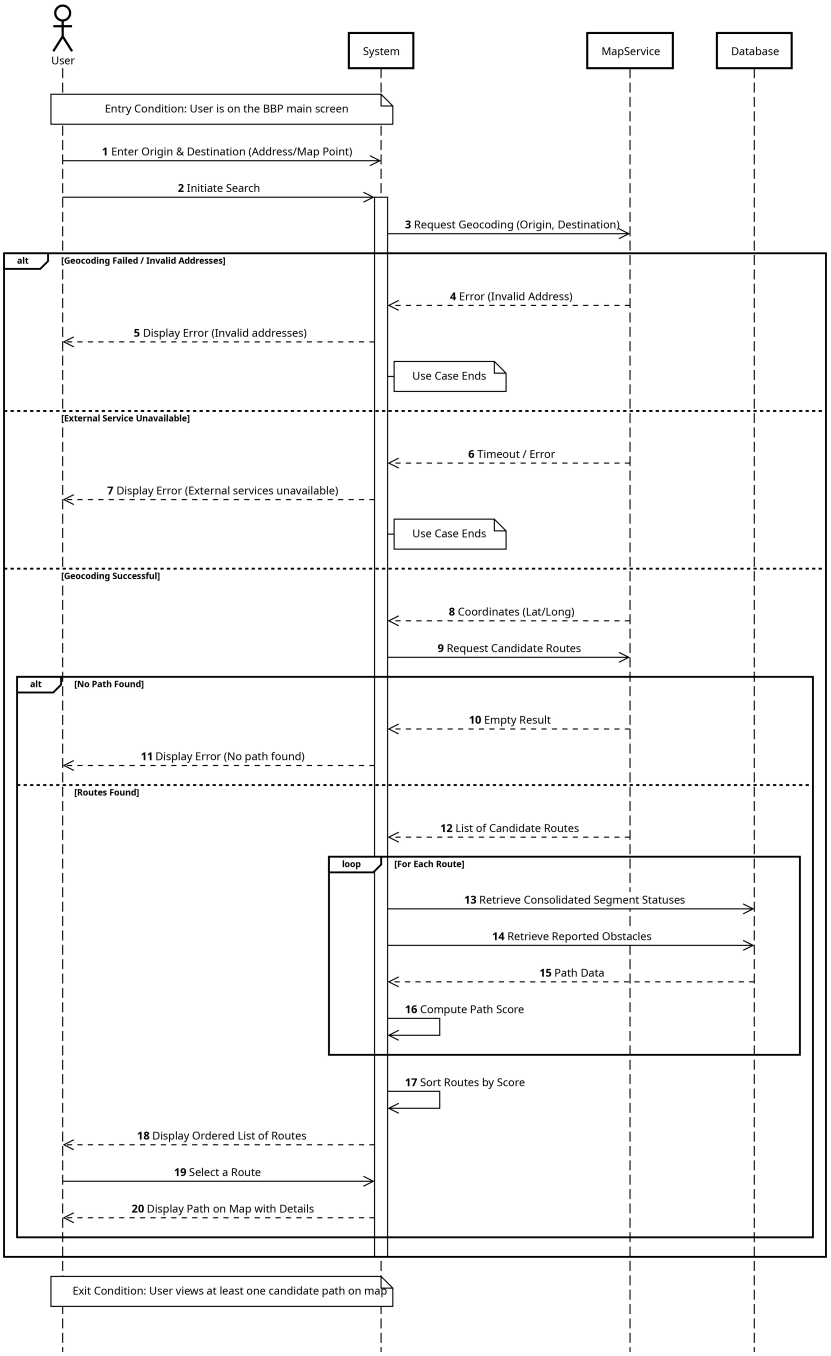

Figure 20: Sequence Diagram for UC6- Search Path Between Origin and Destination

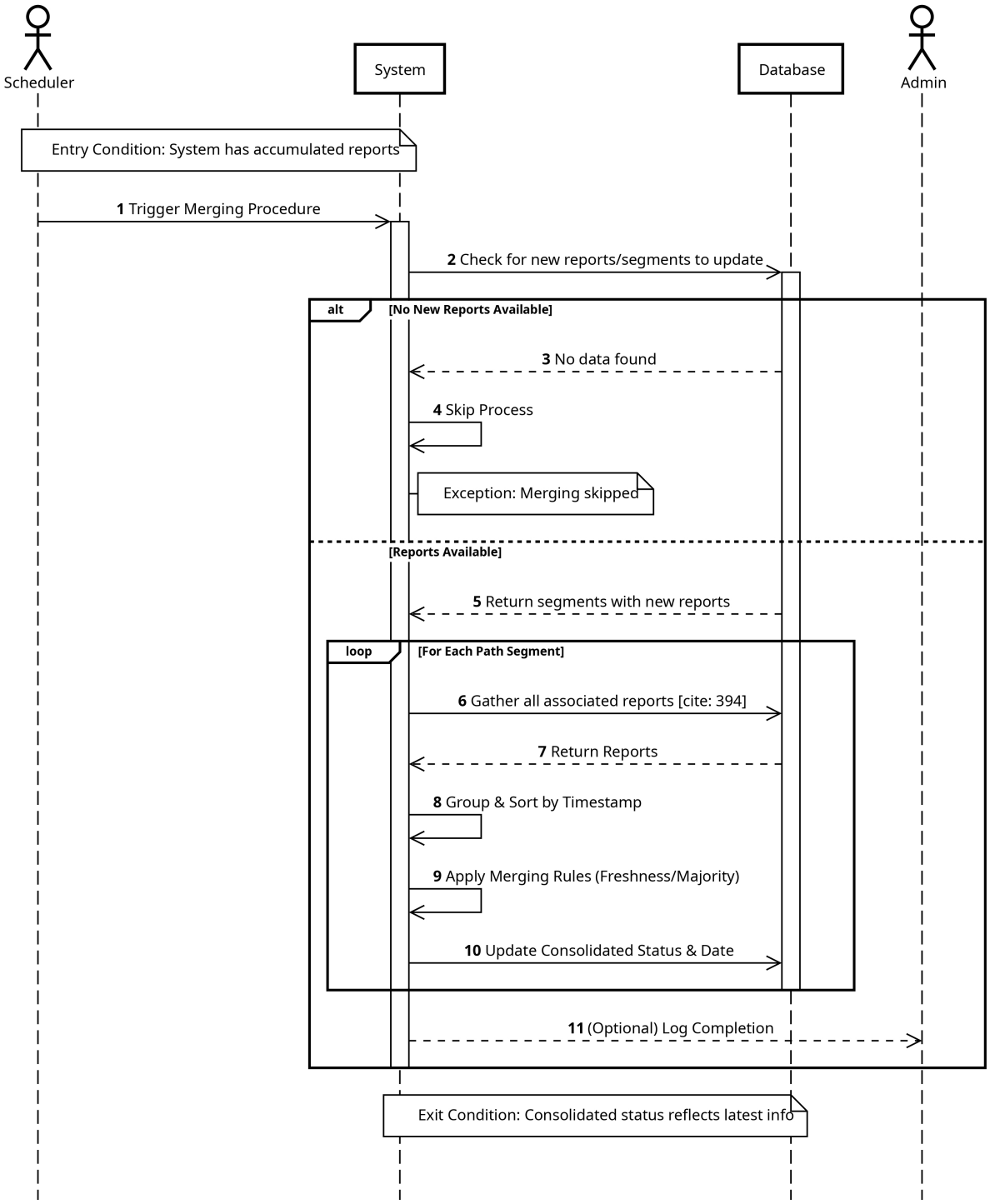

Figure 21: Sequence Diagram for UC7- Mergin Path Reports

3.2.4 Requirement Mapping

G1: Personal trip tracking: Registered users can record, store, and review trips with key statistics and map visualization.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R1 – User registration R2 – User login R3 – Start trip recording R4 – Stop trip recording R5 – Trip statistics computation R6 – Trip history visualization R7 – Trip weather enrichment	D1 – GPS accuracy is sufficient to associate positions with map segments D4 – External map and weather services return correct data D5 – Users' devices have intermittent or active network connectivity D10 – Users have a compatible device with GPS and network access

Table 9: Goal G1 Mapping to Functional Requirements and Domain Assumptions

G2: Manual path information: Registered users can insert or edit path segment statuses and obstacles and decide whether each contribution is publishable.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R8 – Manual path creation R9 – Segment status assignment R10 – Obstacle creation R11 – Mark contribution as publishable or private R12 – Edit manual contributions	D3 – Users generally act in good faith and report path status honestly D6 – A bike path is defined as a dedicated bike track or low-traffic road D7 – Conflicting reports can be meaningfully merged D10 – Users have a reliable internet connection

Table 10: Goal G2 Mapping to Functional Requirements and Domain Assumptions

G3: Automated path information: Registered users can enable sensor-assisted acquisition while biking; BBP detects candidate issues that must be confirmed or corrected by the user before publication.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R13 – Enable/disable automatic mode R14 – Sensor data logging R15 – Detection of candidate obstacles R16 – Candidate list presentation R17 – User confirmation, rejection, or correction R18 – Conversion of confirmed candidates into reports	D2 – Device sensors are calibrated and reliable enough to detect pothole-like events D5 – Devices have sufficient computational capability and battery D8 – Users explicitly enable automatic sensor-based data collection D10 – Users have compatible devices and OS support

Table 11: Goal G3 Mapping to Functional Requirements and Domain Assumptions

G4: Route search and visualization: Any user (registered or guest) can request bike paths between origin and destination and visualize one or more route options.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R19 – Public path search R20 – Route computation R22 – Ordered path list R23 – Path visualization	D4 – External map and routing services are available D6 – Bike paths are correctly represented in map provider data D10 – Users have a working internet connection

Table 12: Goal G4 Mapping to Functional Requirements and Domain Assumptions

G5: Path scoring: BBP orders multiple route options by a path score combining segment quality and route effectiveness.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R21 – Path scoring	D7 – A meaningful consolidated status can be derived from reports D4 – Map data and segment metadata are accurate D10 – External services respond within acceptable time

Table 13: Goal G5 Mapping to Functional Requirements and Domain Assumptions

G6: Merging: BBP merges publishable information from multiple users about the same segments considering freshness and confirmations or contradictions.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R24 – Segment report storage R25 – Periodic merging R26 – Freshness handling R27 – Majority handling	D3 – Users act in good faith when submitting reports D7 – Time- and majority-based rules produce reliable consolidated status D10 – Backend infrastructure supports scheduled jobs

Table 14: Goal G6 Mapping to Functional Requirements and Domain Assumptions

G7: Privacy and compliance: Personal and location data are protected and publication is always under user control.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R11 – Publishability control R17 – User confirmation before publication R29 – Data removal by administrators	Regulatory constraint – GDPR or equivalent data protection regulations apply D8 – Users explicitly consent to sensor usage D10 – Secure communication channels are available

Table 15: Goal G7 Mapping to Functional Requirements and Domain Assumptions

G8: Weather & context enrichment: BBP enriches trip data with meteorological information retrieved from an external service.	
Mapped Functional Requirements	**Mapped Domain Assumptions / Dependencies**
R7 – Trip weather enrichment	D4 – Weather service is available and returns correct data D5 – Network connectivity exists at least intermittently D10 – External APIs respond within acceptable time

Table 16: Goal G8 Mapping to Functional Requirements and Domain Assumptions

3.3 Performance Requirements

1. The system should return a list of candidate paths between origin and destination within 10 seconds for typical urban distances (up to 15 km) under normal load.
2. Trip recording must sample GPS at least every 5 seconds when in motion.
3. Sensor sampling should be frequent enough (e.g., $\geq$ 10 Hz) to detect pothole-like events, but not so frequent as to drain the battery excessively.
4. Under normal conditions, the system should support concurrent use by several thousand users without noticeable degradation of response time for basic operations (trip list, path search).

3.4 Design Constraints

3.4.1 Standards Compliance

1. **Data protection:** GDPR or analogous regulations in target regions.
2. **Security:** Recommended best practices (e.g., OWASP for web/mobile applications).
3. **Map licensing:** Comply with provider's license (e.g., attribution, usage limits).

3.4.2 Hardware Limitations

1. BBP is primarily used on smartphones; performance and UI must accommodate mid-range devices (limited CPU, battery and memory).
2. GPS and sensors may be temporarily unavailable or degraded (e.g., tunnels, urban canyons).

3.4.3 Any Other Constraint

1. Application requires a reasonably recent browser/OS with support for HTTPS and JavaScript.
2. Offline usage: full trip recording should work with intermittent network; upload and path search may require connectivity.

3.5 Software System Attributes

3.5.1 Reliiability

1. Trips and path reports should be stored durably; server uses replicated storage and regular backups.
2. Failure during recording (e.g., app crash) should not lose all data; partial tracks should be salvaged when possible.

3.5.2 Availablity

1. BBP should target at least 99% uptime.
2. Core services (trip storage and path search) should be deployed with redundancy.

3.5.3 Security

1. All communications are over HTTPS.
2. Passwords stored securely or external identity provider used.
3. Access control: only the owner (and admins) can view private trips; only publishable info is visible to others.
4. Rate limiting and request validation to reduce abuse.

3.5.4 Maintainability

1. Modular design separating front-end, back-end APIs and integration with external services.
2. Well-documented public APIs and clear separation between trip logging, path search and merging modules.

3.5.5 Portability

1. Front-end should run on major mobile OSes and modern browsers.
2. Back-end deployable on common cloud infrastructures using containerization.

4 Formal Analysis Using Alloy

In this section, we use Alloy to check that the BBP reporting and consolidation logic behaves as intended. We first describe the main Alloy model and its key components, and then walk through several scenarios that demonstrate the expected behavior of the system.

4.1 Main Alloy Model

Listing 1: Main Alloy model: reporting and consolidation

```
module bbp
open util/integer

sig Email {}
abstract sig Bool {}
one sig True, False extends Bool {}

abstract sig SegmentStatus {}
one sig Unknown, Optimal, Medium, Sufficient, RequiresMaintenance extends
    SegmentStatus {}


sig RegisteredUser {
  email : one Email,
  blocked : one Bool
}

sig Trip {
  owner : one RegisteredUser,
  start : one Int,
  end : one Int
}

sig PathSegment {
  consolidated : one ConsolidatedStatus
}

sig ConsolidatedStatus {
  segment : one PathSegment,
  status : one SegmentStatus,
  lastUpdated : one Int
}

sig PathSegmentReport {
  author : one RegisteredUser,
  segment : one PathSegment,
  status : one SegmentStatus,
  ts : one Int,
  publishable : one Bool
}

sig CandidateIssue {
  trip : one Trip,
  reporter : one RegisteredUser,
  detectedAt : one Int,
  seg : one PathSegment,
  publishIntent : one Bool,
  producedReport : one PathSegmentReport
```



```
}


// Helpers

fun pubReports[s: PathSegment] : set PathSegmentReport {
  { r: PathSegmentReport | r.segment = s and r.publishable = True }
}

fun latestPubReportsByTime[s: PathSegment] : set PathSegmentReport {
  { r: pubReports[s] | no r2: pubReports[s] | r.ts < r2.ts }
}

fun maxPubTime[s: PathSegment] : lone Int {
  { ti: Int |
      some r: pubReports[s] | r.ts = ti
      and no r2: pubReports[s] | r2.ts > ti
  }
}

fun countStatus[rs: set PathSegmentReport, st: SegmentStatus] : Int {
  #( { r: rs | r.status = st } )
}

fun majorityStatusAtLatestTime[s: PathSegment] : set SegmentStatus {
  let L = latestPubReportsByTime[s] |
    { st: SegmentStatus - Unknown |
        all st2: SegmentStatus - Unknown |
          countStatus[L, st] >= countStatus[L, st2]
    }
}


// Facts (invariants)

fact UniqueEmail {
  all disj u1, u2: RegisteredUser | u1.email != u2.email
}

fact TripWellFormed {
  all tr: Trip | tr.start <= tr.end
}

fact CandidateOwnership {
  all c: CandidateIssue | c.trip.owner = c.reporter
}

fact CandidateCreatesReport {
  all c: CandidateIssue |
    c.producedReport.author = c.reporter
    and c.producedReport.segment = c.seg
    and c.producedReport.ts = c.detectedAt
    and c.producedReport.publishable = c.publishIntent
    and c.producedReport.status != Unknown
}

fact BlockedUsersCannotReport {
```



```
  no r: PathSegmentReport | r.author.blocked = True
}

fact ConsolidatedStatusBijection {
  all s: PathSegment | s.consolidated.segment = s
  all cs: ConsolidatedStatus | cs = cs.segment.consolidated
}

fact ConsolidationFromPublishableReports {
  all s: PathSegment |
    (no pubReports[s]) implies (
      s.consolidated.status = Unknown
      and s.consolidated.lastUpdated = 0
    )

  all s: PathSegment |
    (some pubReports[s]) implies (
      s.consolidated.lastUpdated = maxPubTime[s]
      and s.consolidated.status in majorityStatusAtLatestTime[s]
    )
}
```

4.2 Scenario S1: Publishable confirmed issue updates the segment

A user confirms an automatically detected issue and chooses to publish it. The report becomes public, so the segment consolidated status is updated using that report. Here, the report time is $t = 5$, so the consolidated time becomes 5 as shown in Fig. 22 and the status becomes `RequiresMaintenance`.

Listing 2: Scenario S1: confirmed publishable candidate updates consolidation

```
pred Scenario_S1_ConfirmedCandidateUpdatesConsolidation {
  some u: RegisteredUser, tripVar: Trip, s: PathSegment, c: CandidateIssue, r:
      PathSegmentReport | {
    u.blocked = False

    tripVar.owner = u
    tripVar.start = 0
    tripVar.end = 10

    c.trip = tripVar
    c.reporter = u
    c.seg = s
    c.detectedAt = 5
    c.publishIntent = True

    c.producedReport = r
    r.author = u
    r.segment = s
    r.ts = 5
    r.publishable = True
    r.status = RequiresMaintenance

    s.consolidated.lastUpdated = 5
    s.consolidated.status = RequiresMaintenance
  }
}

```



```
run Scenario_S1_ConfirmedCandidateUpdatesConsolidation for
  1 RegisteredUser, 1 Trip, 1 PathSegment, 1 ConsolidatedStatus, 1 CandidateIssue, 1
      PathSegmentReport,
  1 Email, 8 Int
```

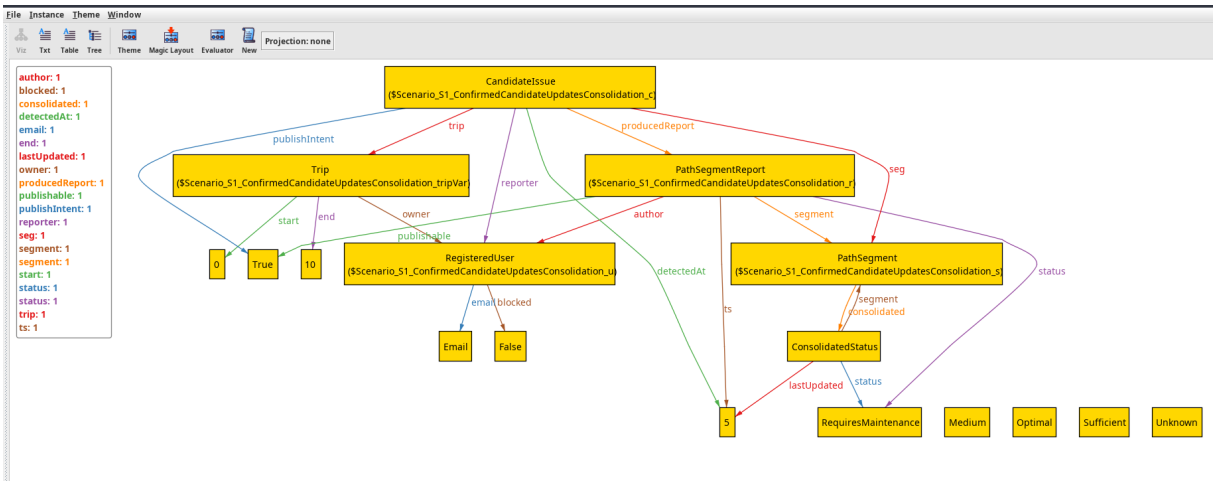

Figure 22: Scenario 1

4.3 Scenario S2: Private confirmed issue does not affect the public consolidated status

A user confirms an issue but marks it private (`publishable=False`). Because only publishable reports are used for consolidation, the segment stays `Unknown` and `lastUpdated` stays 0 as shown in Fig. 23.

Listing 3: Scenario S2: private report is ignored by consolidation

```
pred Scenario_S2_PrivateCandidateDoesNotAffectConsolidation {
  some u: RegisteredUser, tr: Trip, s: PathSegment, c: CandidateIssue, r:
      PathSegmentReport | {
    u.blocked = False

    tr.owner = u
    tr.start = 0
    tr.end = 10

    c.trip = tr
    c.reporter = u
    c.seg = s
    c.detectedAt = 5
    c.publishIntent = False

    c.producedReport = r
    r.author = u
    r.segment = s
    r.ts = 5
    r.publishable = False
    r.status = RequiresMaintenance

    no pubReports[s]
    s.consolidated.status = Unknown
    s.consolidated.lastUpdated = 0
  }
}
```



```
27
28 run Scenario_S2_PrivateCandidateDoesNotAffectConsolidation for
29   1 RegisteredUser, 1 Trip, 1 PathSegment, 1 ConsolidatedStatus, 1 CandidateIssue, 1
         PathSegmentReport,
30   1 Email, 8 Int
```

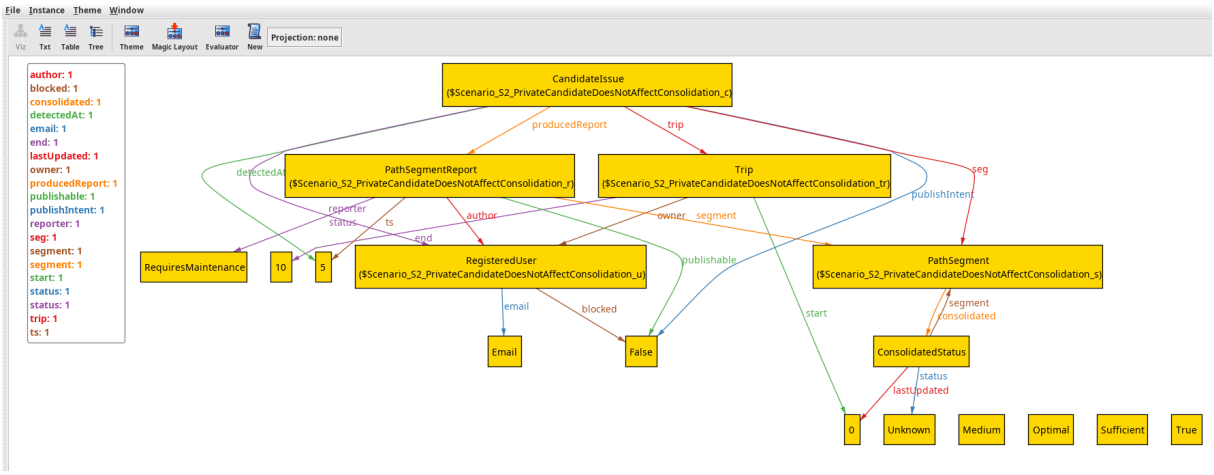

Figure 23: Scenario 2

4.4 Scenario S3: Blocked user cannot produce a report (UNSAT)

A blocked user tries to produce a publishable report. This must be impossible because the model forbids reports authored by blocked users. When running this scenario, Alloy should return *no instance found* as shown in Fig. 24

Listing 4: Scenario S3: blocked user producing a report is unsatisfiable

```
1  pred Scenario_S3_BlockedUserCannotProduceReport {
2    some u: RegisteredUser, tr: Trip, s: PathSegment, c: CandidateIssue, r:
         PathSegmentReport | {
3      u.blocked = True
4
5      tr.owner = u
6      tr.start = 0
7      tr.end = 10
8
9      c.trip = tr
10     c.reporter = u
11     c.seg = s
12     c.detectedAt = 5
13     c.publishIntent = True
14
15     c.producedReport = r
16     r.author = u
17     r.segment = s
18     r.ts = 5
19     r.publishable = True
20     r.status = RequiresMaintenance
21   }
22 }
23
24 run Scenario_S3_BlockedUserCannotProduceReport for
```



```
  1 RegisteredUser, 1 Trip, 1 PathSegment, 1 ConsolidatedStatus, 1 CandidateIssue, 1
      PathSegmentReport,
  1 Email, 8 Int
```

```
Executing "Run Scenario_S3_BlockedUserCannotProduceReport for 8 int, 1 RegisteredUser, 1 Trip, 1 PathSegment, 1 ConsolidatedStatus, 1 CandidateIssue, 
   Actual scopes: 1 Email, 2 Bool, exactly 1 True, exactly 1 False, 5 SegmentStatus, exactly 1 Unknown, exactly 1 Optimal, exactly 1 Medium, exactly 1
   Solver=sat4j Bitwidth=8 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
   24344 vars. 1318 primary vars. 145891 clauses. 330ms.
   No instance found. Predicate may be inconsistent. 1ms.
```

Figure 24: Scenario 3

4.5 Scenario S4: Two publishable reports, newest one is used

Two publishable reports exist for the same segment at different times ($t = 3$ and $t = 7$). The consolidated status must use the newest publishable timestamp, so it updates using the $t = 7$ report as shown in Fig. 25.

Listing 5: Scenario S4: freshness rule (latest report wins)

```
pred Scenario_S4_TwoPublishableReportsLatestWins {
  some u: RegisteredUser, tr: Trip, s: PathSegment, c1, c2: CandidateIssue, r1, r2:
      PathSegmentReport | {
    u.blocked = False

    tr.owner = u
    tr.start = 0
    tr.end = 10

    c1.trip = tr
    c1.reporter = u
    c1.seg = s
    c1.detectedAt = 3
    c1.publishIntent = True
    c1.producedReport = r1
    r1.author = u
    r1.segment = s
    r1.ts = 3
    r1.publishable = True
    r1.status = RequiresMaintenance

    c2.trip = tr
    c2.reporter = u
    c2.seg = s
    c2.detectedAt = 7
    c2.publishIntent = True
    c2.producedReport = r2
    r2.author = u
    r2.segment = s
    r2.ts = 7
    r2.publishable = True
    r2.status = RequiresMaintenance

    s.consolidated.lastUpdated = 7
    s.consolidated.status = RequiresMaintenance
  }
}

run Scenario_S4_TwoPublishableReportsLatestWins for
```



```
  exactly 1 RegisteredUser, exactly 1 Trip, exactly 1 PathSegment, exactly 1
      ConsolidatedStatus,
  exactly 2 CandidateIssue, exactly 2 PathSegmentReport, exactly 1 Email,
  5 Int
```

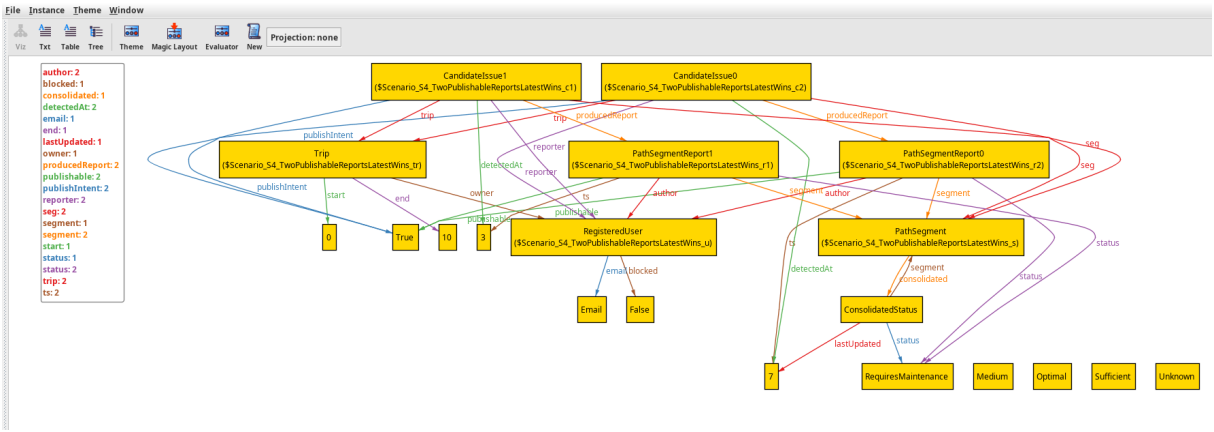

Figure 25: Scenario 4

4.6 Scenario S5: No reports means Unknown

There are no reports at all. Since consolidation only uses publishable reports and there are none, the segment consolidated status remains `Unknown` and `lastUpdated` remains 0 as shown in Fig. 26.

Listing 6: Scenario S5: baseline (no reports)

```
pred Scenario_S5_NoReportsMeansUnknown {
  some s: PathSegment | {
    no PathSegmentReport
    s.consolidated.status = Unknown
    s.consolidated.lastUpdated = 0
  }
}

run Scenario_S5_NoReportsMeansUnknown for
  exactly 1 PathSegment, exactly 1 ConsolidatedStatus,
  exactly 0 PathSegmentReport, exactly 0 CandidateIssue, exactly 0 Trip,
  exactly 0 RegisteredUser, exactly 0 Email,
  5 Int
```

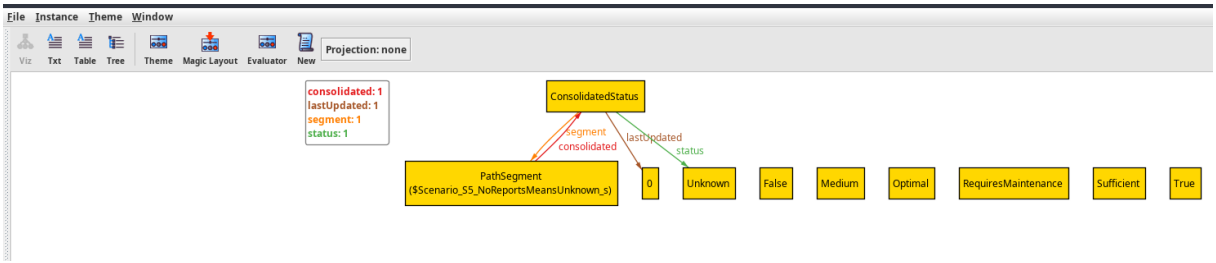

Figure 26: Scenario 5

4.7 Scenario S6: Same latest time, majority vote wins

Three publishable reports exist at the same newest timestamp ($t = 10$). Two reports say `RequiresMaintenance` and one says `Optimal`. The consolidated status follows the majority at the latest time, so it becomes `RequiresMaintenance` with `lastUpdated=10` as shown in Fig. 27.

Listing 7: Scenario S6: majority at latest timestamp

```
pred Scenario_S6_MajorityAtLatestTimestampWins {
  some u1, u2, u3: RegisteredUser, s: PathSegment, r1, r2, r3: PathSegmentReport | {
    u1.blocked = False and u2.blocked = False and u3.blocked = False

    r1.author = u1
    r1.segment = s
    r1.ts = 10
    r1.publishable = True
    r1.status = Optimal

    r2.author = u2
    r2.segment = s
    r2.ts = 10
    r2.publishable = True
    r2.status = RequiresMaintenance

    r3.author = u3
    r3.segment = s
    r3.ts = 10
    r3.publishable = True
    r3.status = RequiresMaintenance

    s.consolidated.lastUpdated = 10
    s.consolidated.status = RequiresMaintenance
  }
}

run Scenario_S6_MajorityAtLatestTimestampWins for
  exactly 3 RegisteredUser, exactly 3 Email,
  exactly 1 PathSegment, exactly 1 ConsolidatedStatus,
  exactly 3 PathSegmentReport,
  exactly 0 CandidateIssue, exactly 0 Trip,
  5 Int
```

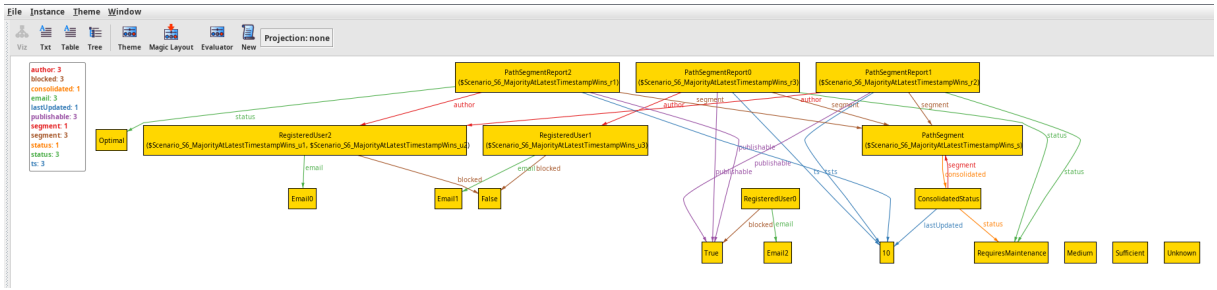

Figure 27: Scenario 6

When the code is executed for all scenarios, the output is shown in Fig. 28

Figure 28: Alloy Analyzer console output for scenarios S1 to S6

5 Effort Spent

Table 17: Effort spent per member

Team Member	Introduction	Overall	Specific Req.	Alloy	Total
Jayasurya Marasani	3	12	22	5	42
Arunkumar Murugesan	4	10	23	4	41
Sneharajalakshmi Palanisamy	3	15	20	3	41
Section Totals	10	37	65	12	124

The table above displays the number of hours each group member spent on the various sections of the document. Please note that the division is only approximate and each section still required the colloboration of all team members.

References

[1] Alloy Analyzer. *Alloy Analyzer Documentation (Analyzer Tooling)*. https://alloy.readthedocs.io/en/latest/tooling/analyzer.html.

[2] Figma, Inc. *Figma: Collaborative Interface Design Tool*. https://www.figma.com.

[3] OpenAI. *ChatGPT (used for paraphrasing)*. https://chatgpt.com.

[4] PlantText. *PlantText: Online Text Editor for Writing and Notes*. https://www.planttext.com.

[5] Overleaf. *Overleaf: Online LaTeX Editor*. https://www.overleaf.com/.

