Travlendar+ project by Kaifei Xu, Shinuo Yan, Yanglin Hu

POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Kaifei Xu, Shinuo Yan, Yanglin Hu
Version:	2.0
Date:	30-December-2025
Download page:	[https://github.com/YanglinHu97/HuXuYan.git]
Copyright:	Copyright © 2025, Kaifei Xu, Shinuo Yan, Yanglin Hu – All rights reserved

目录

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements for the Best Bike Paths， BBP system. BBP is a collaborative platform designed to assist cyclists in navigating urban environments by recommending routes based on road surface quality. The system relies on a crowdsourcing model that aggregates data from two sources: automatic detections， using smartphone sensors and manual user reports. A critical goal of the system is to ensure data reliability; therefore, specific validation protocols—such as user confirmation for automatic detections—are enforced before information is shared with the community.

1.1.1 Goals

1. Cyclists are assisted in finding the smoothest and safest routes based on aggregated road condition data.
2. The system automatically detects road anomalies， e.g., potholes using smartphone sensors to minimize user distraction.
3. Users can manually report or validate obstacles to ensure the reliability of the map data.
4. The system aggregates multi-user data to filter out false positives and maintain an up-to-date representation of the road network.

1.2 Scope

The system acts as a mediator between physical road conditions and the digital map used by cyclists.

1.2.1 World Phenomena

1、The road network consists of interconnected segments joined by nodes.
2、Road segments possess a physical status
3、Maintenance regarding their surface quality.
4、Cyclists traverse sequences of segments， trips to reach a destination.
5、Potholes and obstacles generate physical vibrations on the bicycle

1.2.2 Shared Phenomena

1、The smartphone sensors， IMU detect vibration magnitude during a trip on a specific segment.
2、The User provides a confirmation regarding a detected anomaly.
3、The User manually inputs the status of a road segment.
4、The System displays a valid path composed of adjacent segments to the user.

5、The System visualizes the aggregated status of road segments based on community feedback.

1.3 Definitions, Acronyms, Abbreviations

Segment: An atomic portion of the road network connecting two distinct nodes，Start Node and End Node.
Path: A non-empty ordered sequence of segments where the end node of one segment is the start node of the next.
Detection: An automatic identification of a potential hazard，e.g., pothole triggered by sensor data.
Confirmation: An explicit action by the user to validate a detection they triggered.
Publishable Information: Data，either confirmed detections or manual reports that is deemed valid enough to contribute to the global map status.
Aggregation: The logic of determining a segment's final status by "voting," utilizing the majority of publishable reports.

1.4 Revision History

Version	Date	Description	Author
1.0	2025-12-23	Initial release of the RASD document	[kaifei xu,yanglin hu ，shinuo yan]
2.0	2025-12-31	Second vision f the RASD document	[kaifei xu,yanglin hu ，shinuo yan]

2. Overall Description

2.1 Product Perspective

The BBP system integrates user-generated data to build a dynamic map of road quality. Unlike static maps, BBP's road network model is built upon Segments whose attributes， specifically Status evolve based on user input.

The system logic distinguishes between Raw Data， unverified sensor spikes and Publishable Data， verified information. To maintain system integrity, the application strictly couples automatic detections with user feedback: the system assumes sensor data is potentially noisy and requires human-in-the-loop validation.

2.1.1 Scenarios

To clarify the user interaction and the system's role, the following real-world scenarios are described.

Scenario 1: The Silent Guardian， Automatic Detection Alice is commuting to work. She mounts her phone on the handlebars and starts a trip on BBP. She keeps her eyes on the road. As she rides over a deep pothole, the phone vibrates violently. She does not interact with the screen to avoid danger. The BBP app， Machine detects the sharp Z-axis acceleration spike， SP1 and logs the GPS coordinates， SP2. The system tags this as a "Suspected Anomaly" without Alice's active intervention.

Scenario 2: The Safety-First Planner Bob acts as a "Guest User" to plan a Sunday ride for his family. He selects "Safety Mode" in the app. The system， Machine analyzes the aggregated data from thousands of previous trips. It calculates a route that avoids a shorter but pothole-ridden street, offering instead a slightly longer but smoother bike lane. Bob sees the route on the map， SP4 and starts navigation.

Scenario 3: Community Validation Charlie, an experienced user, rides past a location where Alice previously detected a pothole. The system alerts him: "Hazard ahead." Charlie sees the road has been repaired. He taps "Fixed" on the app screen， SP3. The system updates the database, removing the hazard marker for future users.

2.2 Product Functions

The high-level functions are:

1. Trip Recording: Tracking the user's movement across a sequence of road segments.
2. Automatic Anomaly Detection: Monitoring sensor data to create "Detections" linked to specific segments.

3. Data Validation: Prompting the user to confirm or discard their own automatic detections.
4. Manual Reporting: Allowing users to manually set the status of a segment， publishable by default.
5. Information Aggregation: Calculating the effective status of a segment by aggregating all publishable detections and manual reports using a voting mechanism.
6. Path Computation: Generating valid, connected paths that optimize for the best segment status.

2.3 User Characteristics

This section describes the various types of users who interact with the system and their specific needs.

2.3.1 Guest Users，Unregistered:

1. Description: Casual cyclists or first-time users who have not created an account.

2. Goal: To quickly find a route or view the map without commitment.

3. Privileges: They have read-only access. They can view the map, search for destinations, and use navigation.

4. Limitations: They cannot record trips, report potholes, or validate others' reports. This restriction is necessary to prevent spam and maintain data integrity， as enforced by the Alloy model: only registered users can create Detection or Confirmation.

2.3.2 Registered Users:

1. Description: Regular cyclists who want to contribute to the community.

2. Goal: To track their rides, avoid hazards, and help improve map accuracy.

3. Privileges: They can record trips， generating automatic detections, manually report hazards, and vote/confirm existing reports.

4. Role in System: They are the core data source. The system relies on their "Confirmations" to make a detection "Publishable".

2.3.3 Experienced vs. Inexperienced Cyclists:

2.3.3.1 Inexperienced Cyclists:

1.Needs: Prioritize safety and comfort above all else. They rely heavily on the

2.system's "Optimal" path recommendations and avoid "Requires Maintenance" segments.

3.Technical Proficiency: May need a simpler, turn-by-turn interface.

2.3.3.2 Experienced Cyclists:

1.Needs: May prioritize speed or distance. They might be willing to traverse "Medium" or "Sufficient" quality roads if it saves time.

2.Role: They are more likely to provide accurate manual reports on specific road conditions.

2.4 Assumptions, Dependencies, Constraints

2.4.1 Domain Assumptions，D Assumptions about the real world that must hold true for the system to function correctly.

Device Placement:

It is assumed that users mount their smartphones rigidly on the bicycle handlebars or frame during a trip.，Rationale: If the phone is in a pocket or backpack, the accelerometer data will be dampened, making it impossible to distinguish potholes from body movements.

GPS Availability:

It is assumed that GPS signals are available with sufficient accuracy，< 10 meters in the target urban environments to correctly map a user's location to a specific road Segment.

User Honesty:
It is assumed that the majority of Registered Users provide truthful feedback when confirming or denying a pothole. This validates the "Voting/Aggregation" logic used in the system.

Verification Capability:

It is assumed that cyclists can safely verify a notification，e.g., "Is there a pothole here?" either via a quick glance, voice command, or by stopping, without compromising their safety.

2.4.2 Technical Constraints Hardware Limitations:

The application must run efficiently on mid-range smartphones without draining the battery excessively, even while continuously sampling sensors，Accelerometer/Gyroscope and GPS in the background.

Offline Functionality:

The system must support data buffering. If the internet connection is lost during a ride, the system must locally store the trip data and sensor readings, uploading them once connectivity is restored.

Sensor Heterogeneity:

The anomaly detection algorithms must account for differences in sensor sensitivity across different phone manufacturers，e.g., iPhone vs. Samsung.

2.4.3 **Legal and Regulatory Constraints GDPR Compliance，Privacy:**

Since the system tracks user location，personal data, it must strictly adhere to GDPR，General Data Protection Regulation.

1. Explicit consent is required for location tracking.
2. Trip data stored on the server must be anonymized.
3. Start and End points of trips，which often correspond to home or workplace must be obfuscated in public datasets.

Liability Disclaimer:

The application must include a clear legal disclaimer stating that route recommendations are for informational purposes only. The cyclist remains solely responsible for their safety and adherence to traffic laws.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The Best Bike Paths，BBP system is primarily accessed through a mobile application，smartphone. The UI shall be designed to support two main contexts of use:

Stationary interaction，planning / reporting: when the user is not actively cycling，e.g., at home or after stopping.

On-bike interaction，recording / minimal confirmation: when the user is cycling, where safety and low distraction are critical.

Main UI views and features:

Landing / Map Home: Display the map centered on the user's current location，if permission granted. Provide search fields for Origin and Destination for route planning，Guest and Registered users.

Route Results View: Show one or multiple candidate bike paths between origin and destination. Each path shall include a score based on segment status and route effectiveness，e.g., distance/time efficiency. Provide route details， distance, estimated time, and warning summary such as number of "Requires Maintenance" segments.

Trip Recording View， Registered users: Start / Pause / Stop trip recording. Show live trip metrics， distance, elapsed time, average speed. Show a "Trip Recording Active" indicator while background tracking is running.

Detection Confirmation View， Registered users: When an anomaly is detected automatically, the system shall present a confirmation workflow that can be completed quickly， e.g., one-tap "Confirm", "Reject", "Not sure", since automatic detections can contain false positives and must be validated before becoming publishable. The UI should support a "review later" option to allow safe validation after the ride.

Manual Reporting View， Registered users: Allow manual insertion of segment/path information by specifying streets/segments and selecting a status， Optimal/Medium/Sufficient/Requires Maintenance and obstacles， e.g., potholes.

Publish Management View， Registered users: Allow users to set or change whether their contributed information is "publishable"， shared with the community.

Profile / History， Registered users: List recorded trips and show statistics per trip， distance, speed, etc., potentially enriched with weather data when available.

Usability and safety UI constraints:

While cycling, the UI shall minimize required interaction， large buttons, limited steps, optional audio/vibration cues.

Any confirmation prompts shall be deferrable to avoid forcing interaction during unsafe moments.

3.1.2 Hardware Interfaces

BBP relies on standard smartphone hardware components.

GPS receiver: Used to sample the user's location during trip recording and to reconstruct traveled paths/segments.

Inertial Measurement Unit， IMU sensors: Accelerometer and Gyroscope are used to detect significant vibrations/movements that may indicate potholes or obstacles during cycling.

Device storage: Used to buffer trip/sensor data during intermittent connectivity.

Assumption-related note: correct detection quality depends on the user's smartphone placement and sensor quality variations across devices.

3.1.3 Software Interfaces

BBP interfaces with external software services to provide complete functionality:

Map Service API: Map visualization， tiles, geocoding/search, and route rendering. Optional: map-matching service to associate raw GPS traces with known road segments.

Weather Service API: When available, BBP enriches trip data with meteorological information， weather conditions, temperature, wind speed based on time and location.

The BBP system shall treat external services as unreliable， temporary unavailability, rate limits. The system shall degrade gracefully， e.g., route display without weather enrichment when weather API is unavailable.

3.1.4 Communication Interfaces

BBP client-server communication shall use HTTPS.

Data exchange format shall be JSON for REST-style endpoints.

External API calls， Map/Weather shall also use HTTPS and their documented interfaces.

3.2 Functional Requirements

3.2.1 Use Case Diagram

Actors:

Guest User， Unregistered: can view map and request best bike paths， read-only.

Registered User: can record trips, contribute data， manual/automatic, confirm detections, and manage publishability.

Map Service， external system.

Weather Service， external system.

Use Cases:

UC1 Record Trip

UC2 Automatic Obstacle Detection

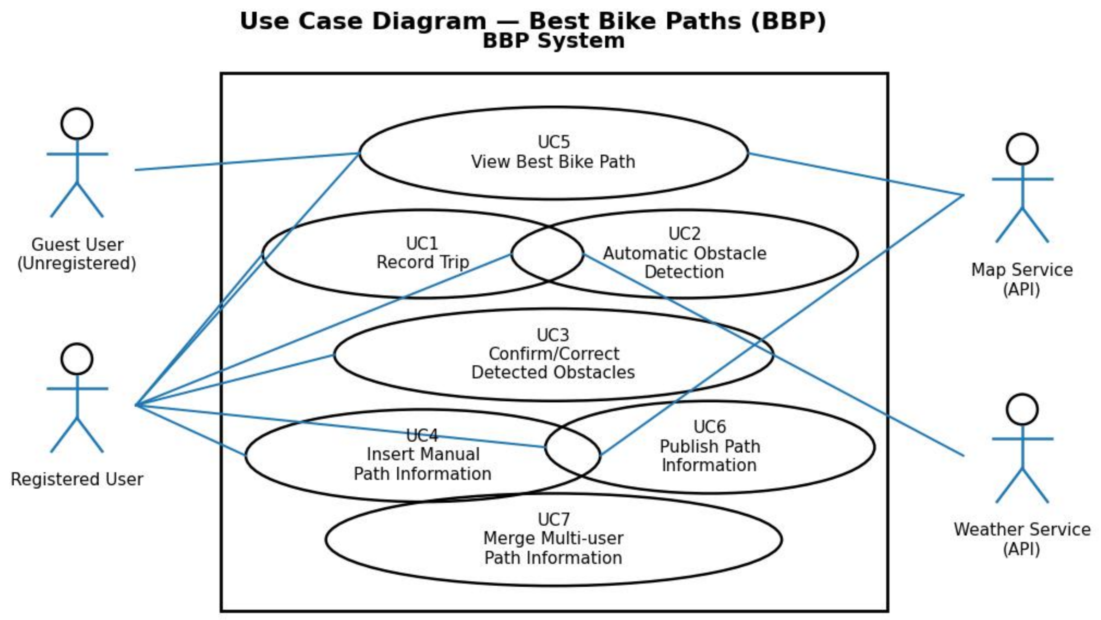
UC3 Confirm/Correct Detected Obstacles

UC4 Insert Manual Path Information

UC5 View Best Bike Path

UC6 Publish Path Information

UC7 Merge Multi-user Path Information



3.2.2 Use Case Descriptions

Notation: "BBP App" indicates the mobile client; "BBP Backend" indicates server-side components that store and aggregate publishable information.

UC1 – Record Trip

Primary Actor: Registered User

Goal: Track a cycling trip, storing the traveled path and trip statistics.

Preconditions: User is authenticated as Registered User; Location permission granted， or user explicitly starts in a limited mode.

Trigger: User taps "Start Trip".

Main Flow:

BBP App starts tracking GPS positions periodically.

BBP App computes live metrics， distance, elapsed time, average speed.

BBP App reconstructs the traveled path as a sequence of segments， map-matching if available.

BBP App stores trip locally during recording.

User taps "Stop Trip".

BBP App uploads the trip to BBP Backend， if online, otherwise queues it for later upload.

BBP Backend stores the trip and statistics for the user.

Postconditions: Trip data is stored for the user; trip may later be enriched with weather data when available.

Alternative Flows:

Offline recording: trip remains buffered locally until connectivity is restored.

Permission denied: BBP App informs user and prevents trip recording until permission is granted.

UC2 – Automatic Obstacle Detection

Primary Actor: Registered User

Goal: Detect potential potholes/obstacles using smartphone sensors during a trip.

Preconditions: An active trip is being recorded，UC1; Accelerometer/Gyroscope are available and enabled.

Trigger: Sensor data indicates a significant vibration/movement consistent with an obstacle.

Main Flow:

BBP App continuously samples IMU signals during the trip.

BBP App detects an anomaly event and associates it with the current GPS location/segment.

BBP App creates a Detection entry marked as "Unverified".

BBP App schedules a confirmation request，immediate if safe, or deferred.

Postconditions: A new unverified detection exists and awaits user validation.

Rationale: False positives are possible; therefore detections require confirmation before becoming publishable.

UC3 – Confirm/Correct Detected Obstacles

Primary Actor: Registered User

Goal: Validate，confirm/reject/correct an automatically detected obstacle so it can become publishable.

Preconditions: At least one unverified Detection exists for the user.

Trigger: User opens "Detections to Review" or receives a prompt.

Main Flow:

BBP App shows the detection details，location/segment, timestamp, confidence.

User selects one action: Confirm, Reject, or Correct，e.g., change type/severity.

BBP App sends a Confirmation to BBP Backend.

BBP Backend updates the Detection state: If confirmed, mark as Publishable Information; if rejected, discard or keep as non-publishable internal data; if corrected, update details and mark publishability accordingly.

Postconditions: The detection becomes publishable only after explicit user confirmation.

Safety Alternative Flow: User selects “Review later”; the detection remains unverified until confirmed.

UC4 – Insert Manual Path Information

Primary Actor: Registered User

Goal: Manually report status/obstacles on a road segment or path by specifying streets/segments.

Preconditions: User is authenticated; map is available.

Trigger: User selects “Report” / “Add information”.

Main Flow:

User selects a segment/path on the map or enters street names.

User selects a Status， Optimal/Medium/Sufficient/Requires Maintenance and optionally adds obstacle notes.

BBP App submits the report to BBP Backend as user-provided information.

BBP Backend stores the manual report as publishable by default， as deliberate user input.

Postconditions: Manual information is stored and can be used for aggregation.

UC5 – View Best Bike Path

Primary Actor: Guest User or Registered User

Goal: Visualize one or more bike paths between an origin and a destination, ranked by score.

Preconditions: Map service reachable， or cached map.

Trigger: User submits Origin/Destination query.

Main Flow:

User inputs， or selects Origin and Destination.

BBP Backend， or app computes candidate paths using the road graph.

For each candidate path, BBP computes a Path Score using aggregated segment status， community information and effectiveness in reaching destination from origin.

System displays ranked paths on the map.

Postconditions: User can view and choose a recommended route.

UC6 – Publish Path Information

Primary Actor: Registered User

Goal: Control whether contributed information， manual or confirmed detection is shared with the community.

Preconditions: User has at least one contribution， manual report or confirmed detection.

Trigger: User toggles publish settings or confirms a detection.

Main Flow:

User selects a contribution entry.

User sets it as publishable， or retracts it, if allowed by system policy.

BBP Backend updates the publishable dataset.

Postconditions: Information becomes available for aggregation and for route scoring for all users.

UC7 – Merge Multi-user Path Information

Primary Actor: BBP Backend， system-initiated

Goal: Merge multiple publishable reports for the same segment/path considering freshness and confirming data.

Preconditions: At least two publishable items exist for the same segment/path from different users.

Trigger: New publishable information arrives or periodic recomputation.

Main Flow:

Backend groups publishable reports by segment/path.

Backend computes an aggregated status using number of confirming data， support count/votes and freshness， more recent reports weigh more.

Backend stores the current aggregated status for route scoring and map visualization.

Example Rule: If two recent reports label a segment “Optimal” and one labels it “Requires Maintenance” in the same timeframe, the aggregated result is “Optimal”, unless newer evidence arrives.

Postconditions: A single aggregated status is available for each segment/path.

3.2.3 Sequence Diagrams

The following sequences should be represented in UML sequence diagrams:

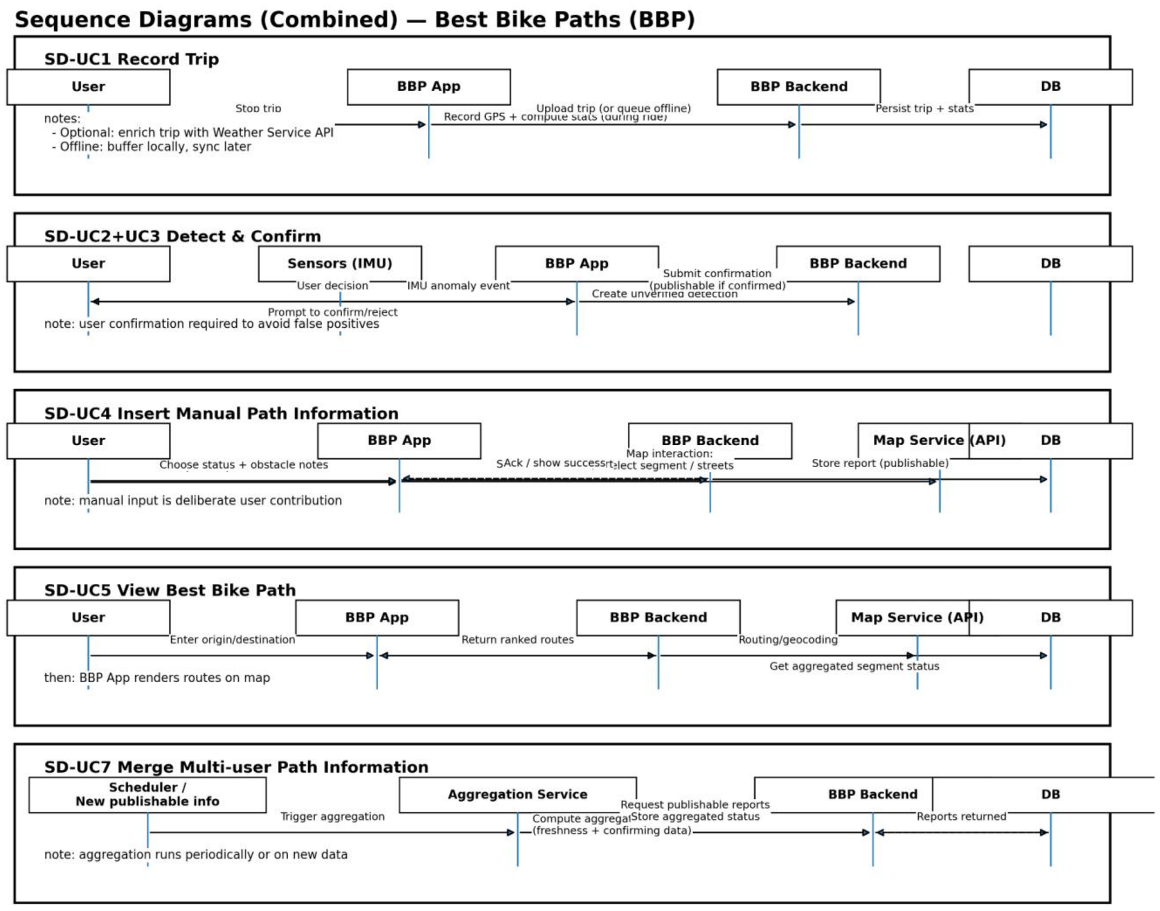
SD-UC1 Record Trip: User → BBP App →， optional Map Service → BBP Backend → DB.

SD-UC2+UC3 Detect & Confirm: BBP App， Sensors → Detection module → User confirmation UI → BBP Backend.

SD-UC4 Manual Report: User → BBP App → BBP Backend → DB.

SD-UC5 View Best Path: User → BBP App → BBP Backend →， Map Service → BBP App.

SD-UC7 Merge Multi-user Info: Scheduler/New Event → Aggregation service → DB.


Sequence Diagrams (Combined) — Best Bike Paths (BBP)
SD-UC1 Record Trip
User
BBP App
BBP Backend
DB
Stop trip
Record GPS + compute stats (during ride)
Upload trip (or queue offline)
Persist trip + stats
notes:
- Optional: enrich trip with Weather Service API
- Offline: buffer locally, sync later
SD-UC2+UC3 Detect & Confirm
User
Sensors (IMU)
BBP App
BBP Backend
DB
User decision
IMU anomaly event
Submit confirmation (publishable if confirmed)
Create unverified detection
Prompt to confirm/reject
note: user confirmation required to avoid false positives
SD-UC4 Insert Manual Path Information
User
BBP App
BBP Backend
Map Service (API)
DB
Choose status + obstacle notes
SAck / show success
Map interaction: select segment / streets
Store report (publishable)
note: manual input is deliberate user contribution
SD-UC5 View Best Bike Path
User
BBP App
BBP Backend
Map Service (API)
DB
Enter origin/destination
Return ranked routes
Routing/geocoding
Get aggregated segment status
then: BBP App renders routes on map
SD-UC7 Merge Multi-user Path Information
Scheduler / New publishable info
Aggregation Service
BBP Backend
DB
Trigger aggregation
Compute aggregat (freshness + confirming data)
Request publishable reports
Store aggregated status
Reports returned
note: aggregation runs periodically or on new data

3.3 Performance Requirements

The system shall satisfy the following performance requirements under normal urban usage on mid-range smartphones:

Real-time data sampling: During an active trip, BBP App shall process GPS/IMU samples in real time without blocking the UI .

Detection responsiveness: When an anomaly occurs, BBP App shall register a Detection event and associate it with a location/segment promptly， near real-time to avoid losing context.

Route computation latency: For typical city-scale routing requests, BBP shall return route results within a few seconds, and shall remain usable under temporary external API slowdowns， graceful degradation.

Map rendering latency: Map interactions， pan/zoom shall remain smooth; route overlay rendering shall not noticeably freeze the UI.

Storage and buffering: BBP shall support local buffering of trips and detections when offline, with bounded storage usage and eventual upload once connectivity is restored.

If measurable numeric thresholds are required by the project team, PR2–PR4 can be refined into concrete targets.

3.4 Design Constraints

Sensor precision limits: BBP shall tolerate GPS inaccuracies typical of urban environments and avoid making hard safety-critical guarantees based on single-sample readings.

Mobile device constraints: BBP shall run efficiently on mid-range smartphones and avoid excessive battery drain while sampling GPS and IMU sensors.

Offline-first constraint: BBP shall buffer data locally during connectivity loss and synchronize later.

Privacy/GDPR compliance: Because BBP processes location data, it shall require explicit consent for location tracking, anonymize stored trip data, and obfuscate sensitive start/end points in public datasets.

Standards compliance: The RASD and design artifacts shall align with the adopted standards and notations， e.g., IEEE/ISO requirements and UML.

3.5 Software System Attributes

3.5.1 Reliability

BBP shall prevent unverified automatic detections from becoming publishable without user confirmation, reducing false positives in shared data.

BBP Backend shall preserve data integrity for publishable information and aggregation results， e.g., no partial updates.

3.5.2 Availability

Core read-only functionality， map visualization and viewing best paths shall remain available to all users whenever external services permit.

When an external service， Weather/Map is unavailable, BBP shall continue operating with reduced features， e.g., no weather enrichment.

3.5.3 Security

BBP shall protect user accounts and stored data through authenticated access for registered-only operations， trip recording, reporting, confirmation.

BBP shall use HTTPS for all client-server communication and follow secure API practices， token-based auth, least privilege.

Location data shall be treated as sensitive and processed according to privacy constraints.

3.5.4 Maintainability

The system shall separate concerns across modules， Trip management, Detection, Confirmation/Publishability, Aggregation, Routing/Scoring, External API adapters to allow independent updates and testing.

Logging and monitoring hooks， client-side and server-side should support debugging of sensor-based detections and aggregation behavior.

3.5.5 Portability

BBP mobile application should be portable across mainstream smartphone platforms and devices with heterogeneous sensors, accounting for differences in sensor sensitivity.

Backend services should be deployable on commodity cloud infrastructure without dependence on proprietary hardware.

---

4. Formal Analysis Using Alloy

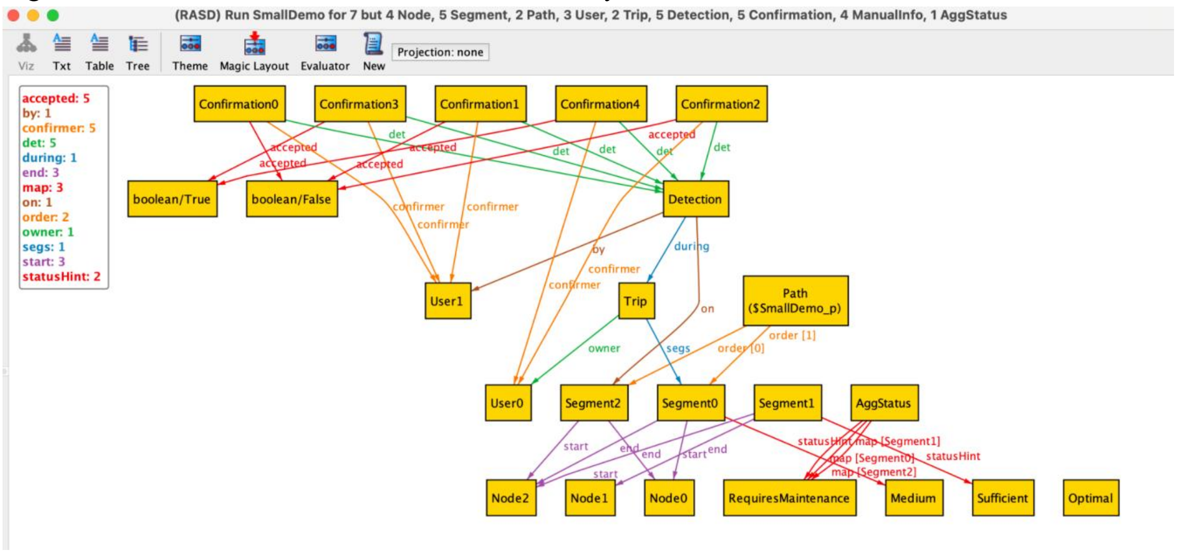
In order to complement the natural-language requirements defined in the previous sections, a formal analysis of the Best Bike Paths system was carried out using Alloy. Natural-language specifications may hide ambiguities or implicit assumptions, especially when multiple users and automated mechanisms interact. For this reason, Alloy is used to precisely define and verify the logical consistency of selected core requirements of the system.

The formal model focuses on three critical aspects of BBP. The first aspect concerns the conditions under which automatically detected potholes can be published. The second aspect addresses the structural validity of bike paths shown to users. The third aspect focuses on the consistency of multi-user information aggregation. These aspects are central to the correctness and reliability of the system.

The road network is modeled using nodes and segments.
Each segment connects two distinct nodes and represents a portion of a bike path. A path is modeled as a non-empty ordered sequence of segments. To ensure structural correctness, the model enforces that consecutive segments in a path are adjacent, meaning that the end node of one segment must coincide with the start node of the next segment.
The corresponding Alloy constraints are shown below.

```
sig Node {}

abstract sig Status {}
one sig Optimal, Medium, Sufficient, RequiresMaintenance extends Status {}

sig Segment {
  start: one Node,
  end:   one Node
} { start != end }

sig Path {
  order: seq Segment
}

fact PathConnectivity {
  all p: Path |
    some p.order.elems and
    all i: p.order.inds |
    ,  i.plus[1] in p.order.inds implies
       p.order[i].end = p.order[i.plus[1]].start
}
```

An Alloy-generated instance illustrating a valid path composed of adjacent segments is shown in Figure 4.1, which demonstrates that the connectivity constraints are satisfiable.



(RASD) Run SmallDemo for 7 but 4 Node, 5 Segment, 2 Path, 3 User, 2 Trip, 5 Detection, 5 Confirmation, 4 ManualInfo, 1 AggStatus

Projection: none

accepted: 5
by: 1
confirmer: 5
det: 5
during: 1
end: 3
map: 3
on: 1
order: 2
owner: 1
segs: 1
start: 3
statusHint: 2

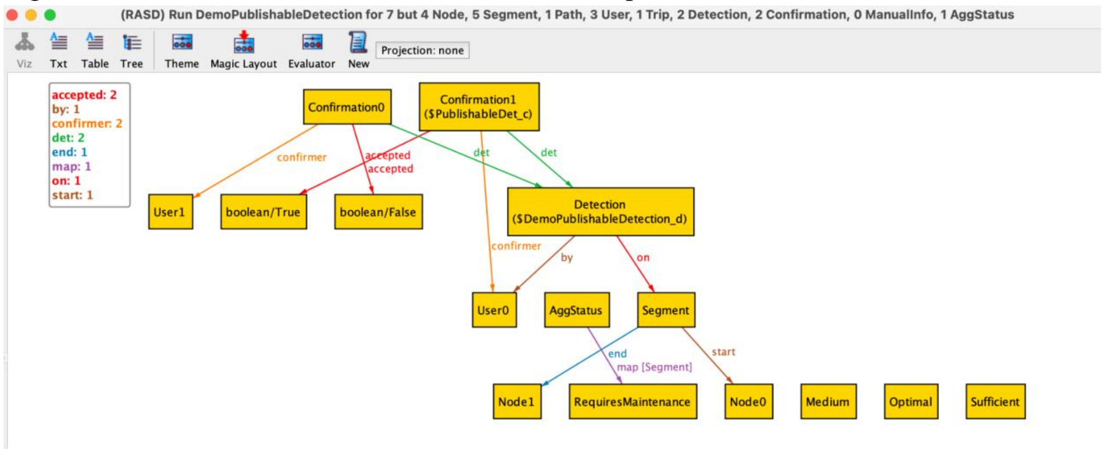
Figure 4.1 – Alloy-generated instance showing a valid bike path composed of adjacent segments.

User interactions with the system are modeled through trips, detections, confirmations, and manual information. A trip represents a bicycle ride recorded by a user and consists of one or more segments. During or after a trip, the system may automatically detect potential potholes on specific segments. However, such detections are not immediately considered reliable and must be validated.

To prevent the dissemination of incorrect information, the model enforces a publishability rule for automatically detected potholes. A detection can be published only if it is explicitly confirmed by the same user who triggered the detection. This rule ensures that unverified or false-positive detections are not shared with the community.
The Alloy predicates defining this behavior are shown below.

```
sig User {}

sig Detection {
  by: one User,
  on: one Segment
}

sig Confirmation {
  det: one Detection,
  confirmer: one User,
  accepted: one Bool
}

pred PublishableDet[d: Detection] {
  some c: Confirmation |
    c.det = d and
    c.confirmer = d.by and
    c.accepted = True
}
```

An Alloy instance showing a publishable detection supported by a valid confirmation is reported in Figure 4.2, where the confirmation is marked as accepted.


(RASD) Run DemoPublishableDetection for 7 but 4 Node, 5 Segment, 1 Path, 3 User, 1 Trip, 2 Detection, 2 Confirmation, 0 ManualInfo, 1 AggStatus

Projection: none

accepted: 2
by: 1
confirmer: 2
det: 2
end: 1
map: 1
on: 1
start: 1

Figure 4.2 – Alloy-generated instance illustrating an automatically detected pothole that becomes publishable after user confirmation.

In addition to automatic detections, users may manually provide information about the condition of a path segment. Manual information is considered publishable by default, as it represents deliberate user input. Both confirmed detections and manual information contribute to the evaluation of a segment's condition when multiple pieces of information are available.

When several users provide feedback for the same segment, the system aggregates this information into a single status. The aggregation mechanism is modeled as a voting process, where each publishable piece of information contributes a vote to the corresponding segment. The aggregated status is selected among those with the highest number of votes, ensuring that the final result reflects the dominant evidence.
An example of multi-user information aggregation is shown in Figure 4.3, where multiple votes are associated with the same segment and the aggregated status is determined accordingly.

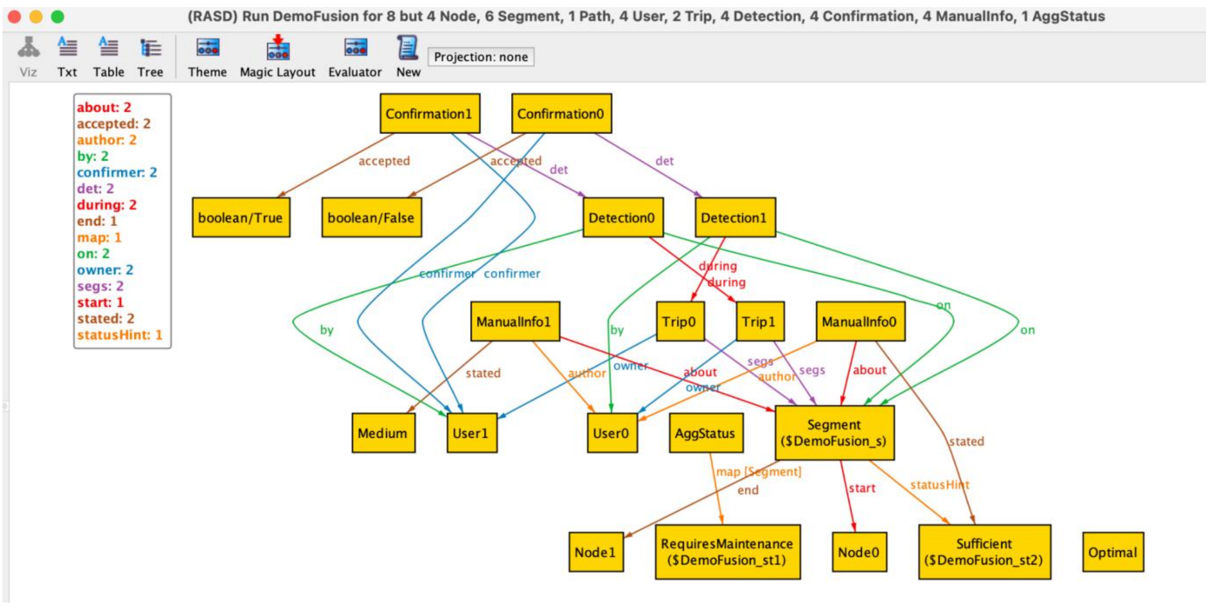
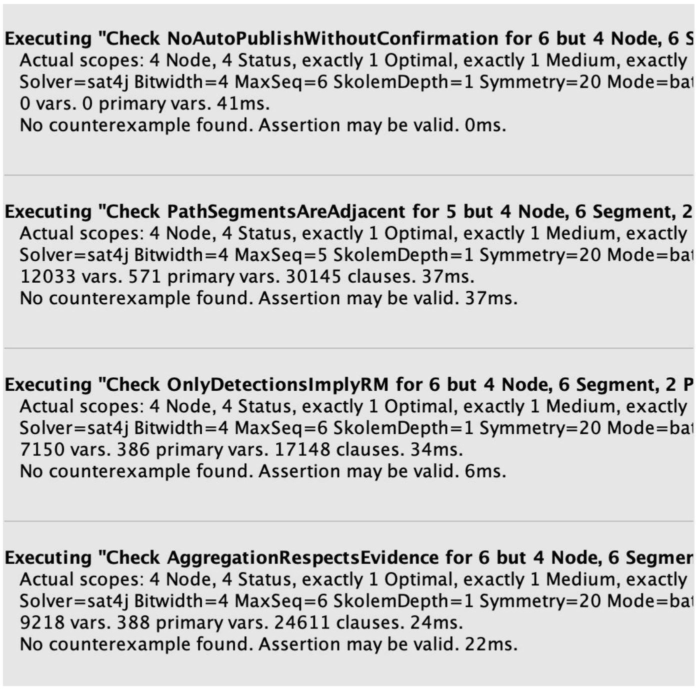


Figure 4.3 – Alloy-generated instance illustrating the aggregation of multiple user inputs for a single path segment.

All critical properties of the model are formally verified using Alloy assertions. These assertions check the correctness of the publishability rules, the adjacency constraints of paths, and the soundness of the aggregation mechanism. The Alloy Analyzer reports that no counterexamples exist for the checked properties.The assertions used in the analysis are reported below.

```
assert NoAutoPublishWithoutConfirmation {
 all d: Detection |
   PublishableDet[d] iff
     some c: Confirmation |
       c.det = d and
       c.confirmer = d.by and
       c.accepted = True
}

assert PathSegmentsAreAdjacent {
 all p: Path |
   all i: p.order.inds |
    ，  i.plus[1] in p.order.inds implies
       p.order[i].end = p.order[i.plus[1]].start
}
```

The results of the assertion checks are shown in Figure 4.4, where each assertion returns the outcome “No counterexample found”.



```
Executing "Check NoAutoPublishWithoutConfirmation for 6 but 4 Node, 6 S
   Actual scopes: 4 Node, 4 Status, exactly 1 Optimal, exactly 1 Medium, exactly
   Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=ba
   0 vars. 0 primary vars. 41ms.
   No counterexample found. Assertion may be valid. 0ms.

Executing "Check PathSegmentsAreAdjacent for 5 but 4 Node, 6 Segment, 2
   Actual scopes: 4 Node, 4 Status, exactly 1 Optimal, exactly 1 Medium, exactly
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20 Mode=ba
   12033 vars. 571 primary vars. 30145 clauses. 37ms.
   No counterexample found. Assertion may be valid. 37ms.

Executing "Check OnlyDetectionsImplyRM for 6 but 4 Node, 6 Segment, 2 P
   Actual scopes: 4 Node, 4 Status, exactly 1 Optimal, exactly 1 Medium, exactly
   Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=ba
   7150 vars. 386 primary vars. 17148 clauses. 34ms.
   No counterexample found. Assertion may be valid. 6ms.

Executing "Check AggregationRespectsEvidence for 6 but 4 Node, 6 Segmer
   Actual scopes: 4 Node, 4 Status, exactly 1 Optimal, exactly 1 Medium, exactly
   Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20 Mode=ba
   9218 vars. 388 primary vars. 24611 clauses. 24ms.
   No counterexample found. Assertion may be valid. 22ms.
```

Figure 4.4 – Results of Alloy assertion checking showing that no counterexample was found.

The formal model intentionally abstracts away several aspects of the real system. Numerical GPS coordinates, sensor signal processing, and temporal behavior are not represented. Path scoring and ranking are simplified, and aggregation relies on a basic majority rule. Despite these simplifications, the model effectively validates the logical correctness of the core BBP requirements and supports the consistency of the overall system design.

4.Effort Spent

The following table shows the time allocation of each group member across different stages of the project:

Group Member	Chapter 1	Chapter 2	Chapter 3	Chapter 4	Total Hours
Kaifei Xu	7	12	6	4	29
Shinuo Yan	3	3	15	9	29
Yanglin Hu	1	5	8	15	29

6. References

[R1] Project Assignment Description: "Best Bike Paths", Software Engineering II, Academic Year 2024-2025, Politecnico di Milano.

[R2] ISO/IEC/IEEE 29148:2018 - Systems and software engineering -- Life cycle processes -- Requirements engineering.

[R3] IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

[R4] UML 2.5 Specification, Object Management Group，OMG.

[R5] Alloy Language Reference, Daniel Jackson, MIT.