POLITECNICO DI MILANO

1863

Computer Science and Engineering

Software Engineering II

2025 – 2026

DD

Design Document

Best Bike Paths(BBP)

By

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

Deliverable:	DD
Title:	Design Document
Authors:	Jayasurya Marasani, Arunkumar Murugesan and Sneharajalakshmi Palanisamy
Version:	1.0
Date:	07 January 2026
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 have multiple bicycle routes from between same origin and destination. But the "best" route depends on more than just distance. Surface quality, potholes, hazardous crossings, temporary construction and exposure to motor traffic all influence safety, comfort and duration. Research also provides structured ways to quantify bike path quality. For example, [1] evaluates cycling infrastructure using the QualiCiclo index. It scores bicycle paths across four categories: infrastructure, signalization, environment and safety. The method uses an ordinal 0 to 3 scale from insufficient to excellent. BBP's notion of segment status and obstacle based quality follows the same idea of making quality explicit and comparable. It can be combined with effectiveness measures during route rankings. At the same time, cyclists increasingly want an easy way to record trips and review quantified history such as distance, speed, recurring commutes and weekly totals.
Best Bike Paths (BBP) is a mobile application system that enables users to:

1. Record bicycle trips and store them in a personal trip log with computed statistics.
2. Contribute information about bike paths, including segment condition and obstacles, through manual reports and confirmed automatic detection.
3. Query and compare routes between two points, ranking candidate paths based on both route effectiveness (e.g., distance, elevation, turns) and path quality indicators derived from consolidated reports.
4. Use the system as guest users to obtain a fast and easy route from point A to point B without requiring registration.

1.2 Scope

BBP is a mobile first application that interacts with the user's device capabilities (GPS, accelerometer, gyroscope and network connectivity) and integrates external services (maps/routing and weather) to support both individual trip logging and community-maintained bike path information.

The system scope includes:

1. **User management**
   - User registration and login.
   - Role-based access for administrative features.
2. **Trip recording and personal log**
   - Real-time trip tracking using GPS.
   - Automatic computation and storage of trip statistics (e.g., distance, duration, average speed).
   - Trip saving and review through a personal trip history.
3. **Optional Trip Enrichment**
   - Retrieval and attachment of weather information to trips when the weather service is reachable.
4. **Manual Bike Path Contributions**
   - Manual input of bike path information, including streets/segments, segment condition/status, and obstacles.

- Ability to edit or remove user-submitted reports as allowed by permissions.

5. **Automatic Aquistion of Bike Path Data**
   - Collection of GPS traces and sensor events (accelerometer/gyroscope) during trip recording when automatic mode is enabled.
   - Detection of candidate issues (e.g., potholes/roughness indicators) based on sensor patterns.
6. **User Confirmation Workflow**
   - Presentation of automatically detected candidate issues after a trip.
   - User confirmation, rejection, or correction of detected issues before they are stored as publishable path reports.
7. **Path Search and visualization for all Users**
   - Route search between origin and destination for guest and registered users.
   - Ranking of candidate routes by a path score that combines route effectiveness (e.g., distance/elevation/turns) and path quality (segment status and obstacles).
   - Map-based visualization of selected routes with overlays for segment condition and reported obstacles.
8. **Report merging and consolidated status**
   - Periodic merging of multiple user's reports into consolidated segment status information, applying freshness and majority/weighted-majority logic.
9. **Administration and data quality controls**
   - Administrative ability to block users who repeatedly submit false data.
   - Administrative ability to remove or hide problematic reports/obstacles and keep the consolidated data consistent.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1. **Flutter**: A cross-platform UI framework used to build native mobile apps from a single codebase (Android and iOS). In BBP it is used to implement UI elements and access to device sensors through plugins.
2. **Firebase/FlutterFire**: Firebase is Google's backend platform. FlutterFire is the official set of Flutter plugins to use Firebase services. In BBP it provides authentication, database, file storage, server-side logic and (optionally) push notifications.
3. **MapService(RoutesAPI)**: An external mapping service providing geocoding (address $\rightarrow$ coordinates) and routing (candidate routes between two points). In BBP it is used during route search (UC6) to generate candidate routes that the system then scores using internal path-quality data.
4. **Weather Service**: An external API used to retrieve weather conditions for a given location and time. In BBP it enriches recorded trips with contextual information when the service is reachable (UC3).
5. **GPS Location Provider**: A recorded ride by a registered user including GPS track and computed statistics.The device capability that provides geographic coordinates over time. In BBP it enables real-time trip tracking and supports associating sensor events with positions.

1.3.2 Acronyms

1. **BBP (Best Bike Paths)**: The name of the system/application that records bike trips, collects path quality information and provides ranked route suggestions.
2. **Auth (Authentication):** It is an authentication service provided by Firebase. It manages user identities and provides secure login sessions used to authorize access to data and features.

1.3.3 Abbreviations

1. **Firestore:** In BBP it stores users, trips, segment reports, obstacles and consolidated segment statuses. Short for Cloud Firestore, NoSQL document database.
2. **Storage:** Stores larger files such as recorded trip traces and optionally photos attached to obstacle reports. Short for Firebase Cloud Storage.
3. **Scheduler:** Short for Cloud Scheduler/Scheduled functions. Periodically triggers the merge process that consolidates multiple PathSegmentReports into a single consolidated status per segment.

1.4 Revision History

Version	Date	Description
1.0	07/01/2026	Full Document of DD.

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 DD “Students & Companies”

1.6 Document Structure

This document is organized as follows:

1. **Introduction:** Presents the purpose and scope of BBP, introduces key definitions and terminology and provides the context needed to understand the rest of the document.
2. **Architectural Design:** Describes the overall architecture of the system, including the main components of the Flutter application and Firebase backend, their responsibilities, deployment choices and how key workflows (e.g., trip recording, route search, merging) are supported.
3. **User Interface Design:** Summarizes the main user interfaces and screens of the application, showing how users interact with BBP for trip recording, reporting and route search and how the UI supports the main use cases.
4. **Requirements Traceability:** Maps each functional requirement defined in the RASD to the corresponding architectural components, data structures and modules, ensuring every requirement is covered by the design.

5. **Implementation, Integration and Test Plan:** Defines the planned implementation order, integration strategy (including dependencies between modules and services) and the test approach (unit, integration, system and non-functional testing) to validate the system.
6. **Effort Spent:** Reports the effort distribution among team members and activities, summarizing how work was allocated across design tasks.
7. **References:** Lists the documents, standards and external resources used to produce this document.

2 Architectural Design

2.1 Architecutural Overview

Best Bike Paths (BBP) is designed as a mobile first system composed of a Flutter application and a serverless backend built on Firebase. The architecture follows a layered approach that separates presentation, application logic and data management responsibilities, while keeping the client end lightweight and delegating sensitive or compute intensive tasks to protected server side logic. The backend is implemented using managed services to reduce operational overhead and improve scalability.

BBP relies on Firebase Authentication to manage identities, Cloud Firestore as the primary database, Cloud Storage for larger binary assets and Cloud Functions as the application tier that encapsulates business rules, data aggregation and integrations with third party APIs. External services provide geocoding, routing candidates and weather enrichment. This composition supports the core system goals of trip recording, community contributions, route search and ranking, data merging and privacy controlled publishing.

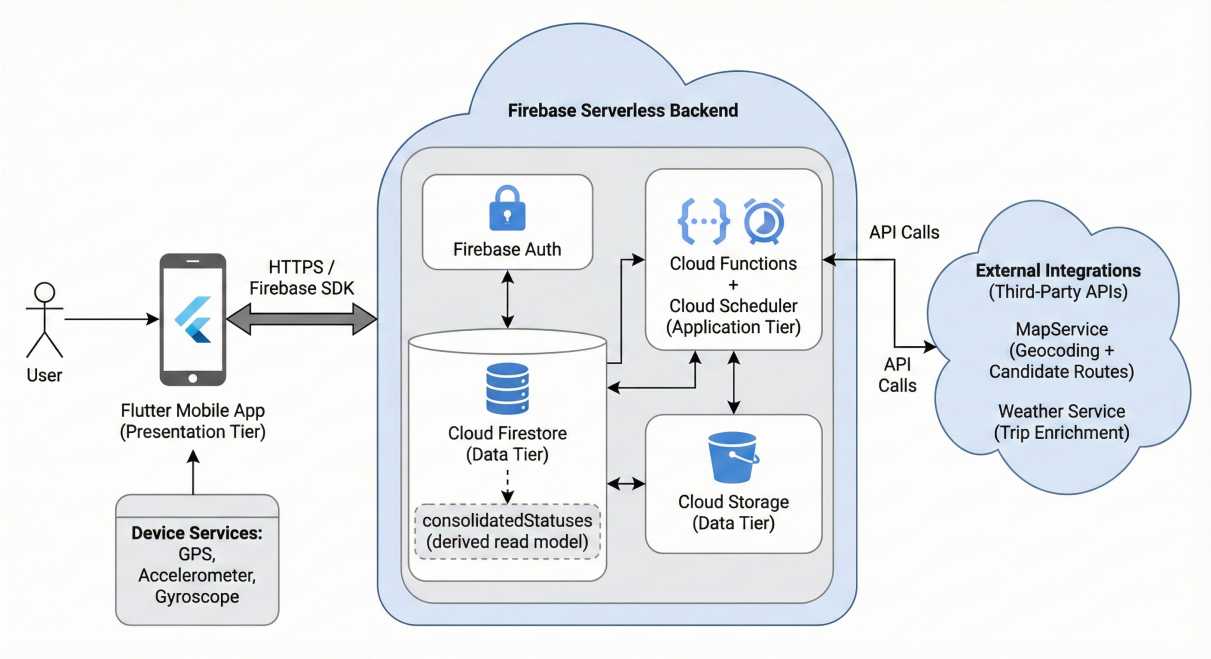

Figure 1: Architectural Overview Diagram

2.2 Component View

The system is organized into the following main components.

1. **Flutter Mobile App (Presentation Tier)**: The Flutter client provides the user interface for registration and login, trip recording, contribution submission and route search. It interacts with Firebase services through the Firebase SDK over HTTPS. It also accesses device services such as GPS, accelerometer and gyroscope to collect trip traces and sensor signals that can support automatic detection workflows.

2. **Firebase Authentication:** Firebase Authentication manages user registration, login and session handling. It issues identity tokens that are used by clients to access protected backend resources according to Firestore Security Rules and Cloud Functions authorization checks.

3. **Cloud Firestore (Data Tier):** Firestore stores the structured data required by BBP, including users, trips metadata, trip summaries, segment reports, obstacles, publishable flags and derived read models. Firestore is also used to support reactive UI updates where appropriate through real time listeners.

4. **Cloud Storage (Data Tier):** Cloud Storage stores larger objects, compressed location sequences and optional media attachments related to reports. Firestore stores references and metadata for these objects.

5. **Cloud Functions and Cloud Scheduler (Application Tier):** Cloud Functions implement protected operations such as route scoring, report merging and generation of consolidated segment status. Functions also act as a controlled gateway to external services to prevent exposing API keys and to ensure consistent validation and rate limiting. Cloud Scheduler is used to trigger periodic maintenance and aggregation tasks such as recomputing consolidated statuses or cleaning up temporary records.

6. **External Integrations (Third Party APIs):** BBP integrates with a map service for geocoding and route candidate generation between origin and destination. It also integrates with a weather service to enrich recorded trips with contextual conditions such as temperature, wind and precipitation.

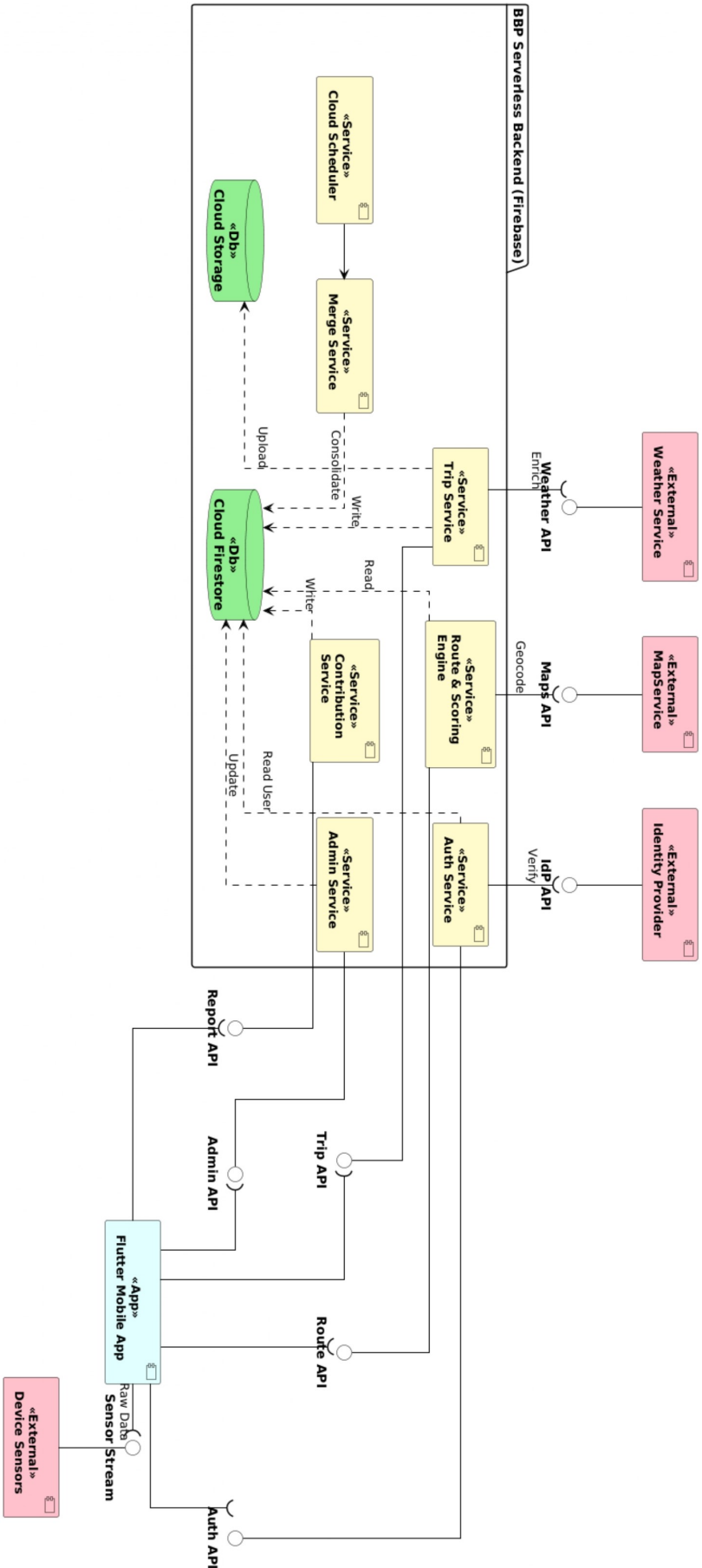

Figure 2: Domain Class Diagram.

2.3 Deployment View

BBP is deployed as a client serverless system.

1. The Flutter application runs on end user mobile devices and communicates with Firebase over HTTPS using the Firebase SDK.
2. The backend is deployed entirely on Firebase and Google Cloud managed infrastructure. Firestore and Cloud Storage store data, while Cloud Functions provide execution environments for application logic.
3. External services are accessed by Cloud Functions, not directly by the mobile client, for operations that require secrets or controlled execution.

This deployment minimizes backend administration, enables elastic scaling and supports a pay per use model.

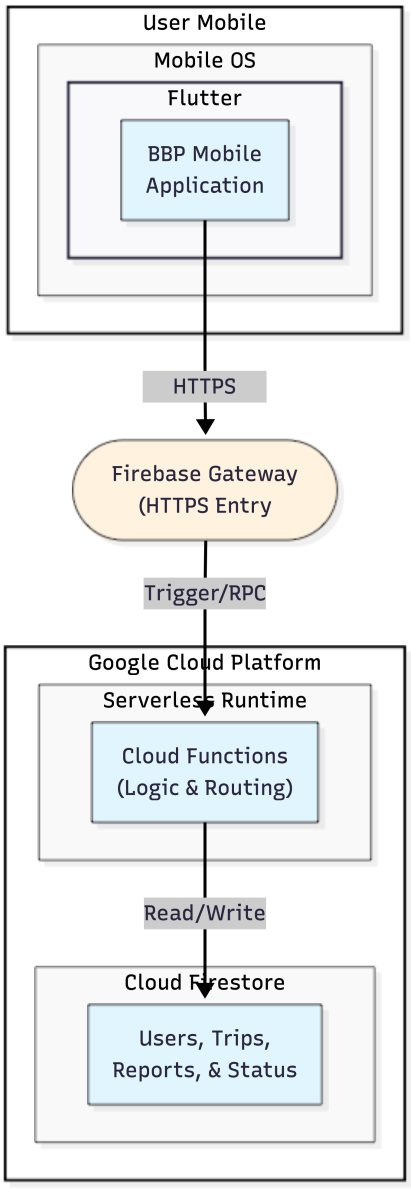

Figure 3: Deployment Diagram

2.4 Runtime View

The runtime behavior of BBP can be described by considering how components collaborate to realize the core use cases.(UC1 to UC2)

1. **User authentication and session establishment(UC1, UC2)**

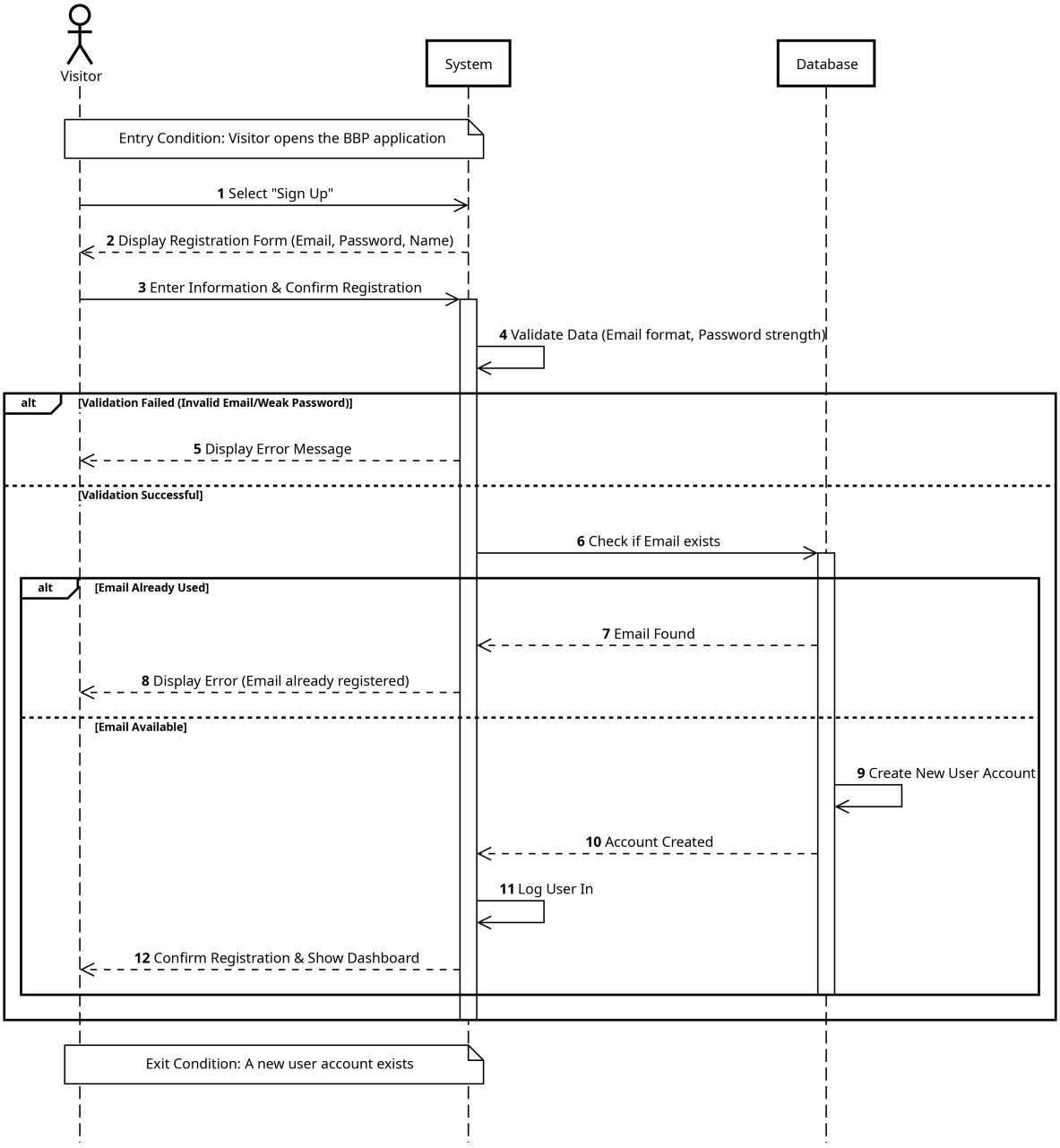

Figure 4: User Authentication Sequence Diagram

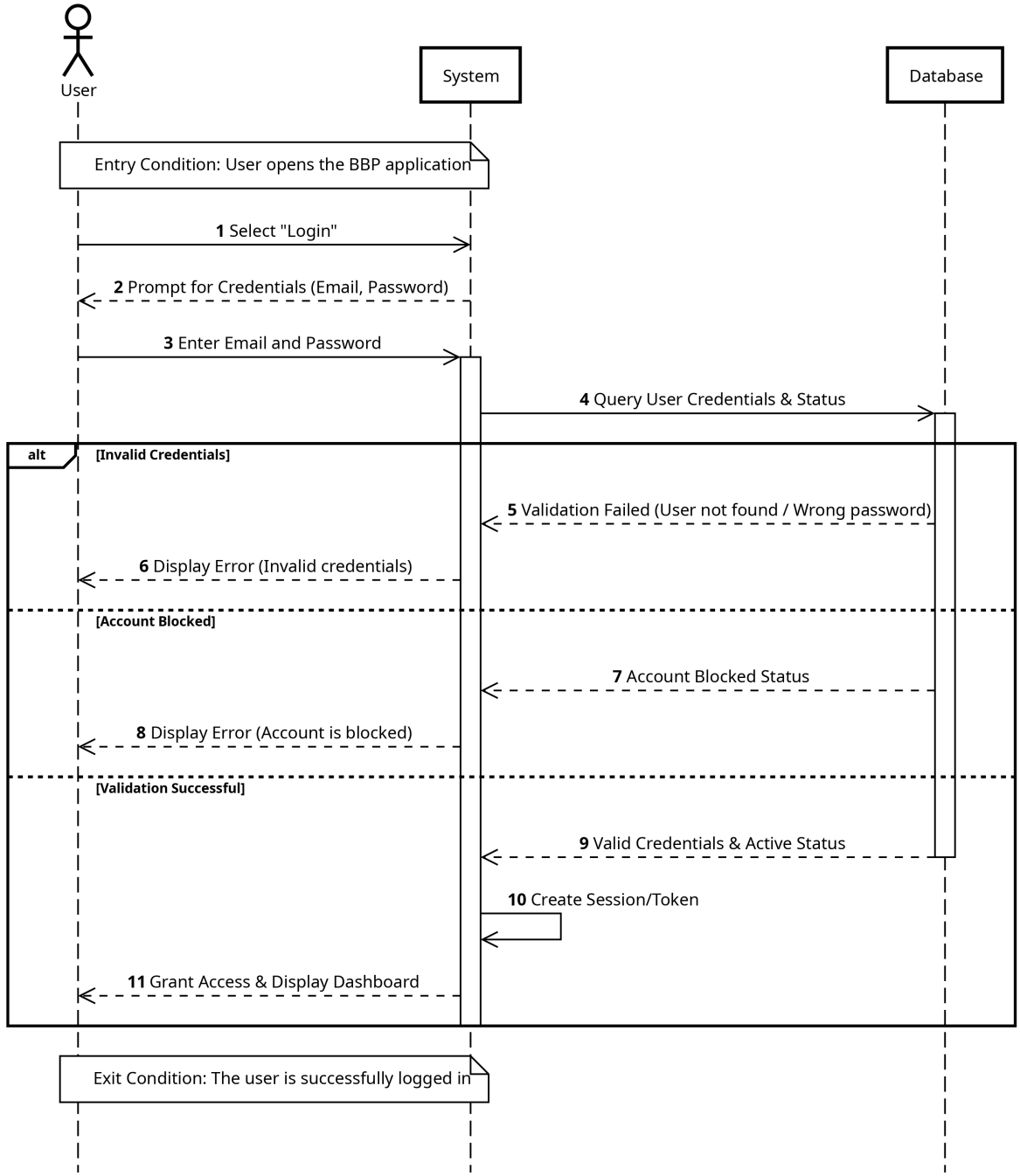

Figure 5: User Session Establishment Sequence Diagram

- The user registers or logs in through the Flutter application.
- Firebase Authentication verifies credentials and returns an identity token.
- The client uses the token to access Firestore documents and callable or HTTPS Cloud Functions.

2. **Trip recording and storage(UC3)**

- The user starts trip recording in the Flutter application.
- The client reads GPS and sensor data at configured intervals.
- During recording, the client may compute lightweight statistics locally and stores intermediate data in memory.

- At the end of the trip, the client uploads trip metadata to Firestore and uploads the heavier trace payload to Cloud Storage.
- A Cloud Function may be triggered to compute additional derived statistics or to normalize trip structures for later visualization.

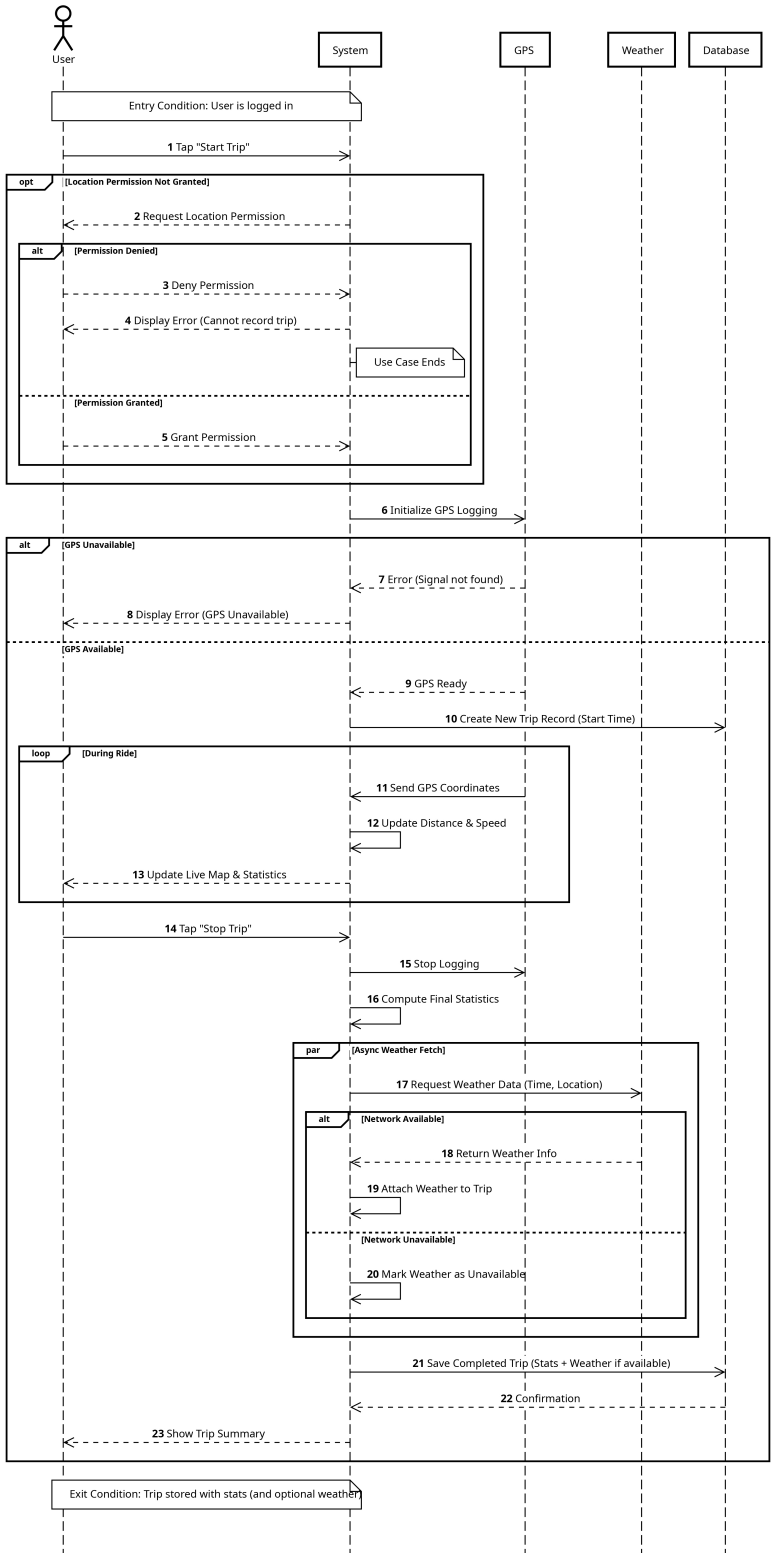

Figure 6: Trip Recording Sequence Diagram

3. **Manual contribution workflow(UC4)**

   - A registered user submits a path segment status or obstacle through the UI.
   - The client writes the report into Firestore with a publishable flag selected by the user.
   - A backend function validates the submission and may trigger aggregation updates for the affected segment.

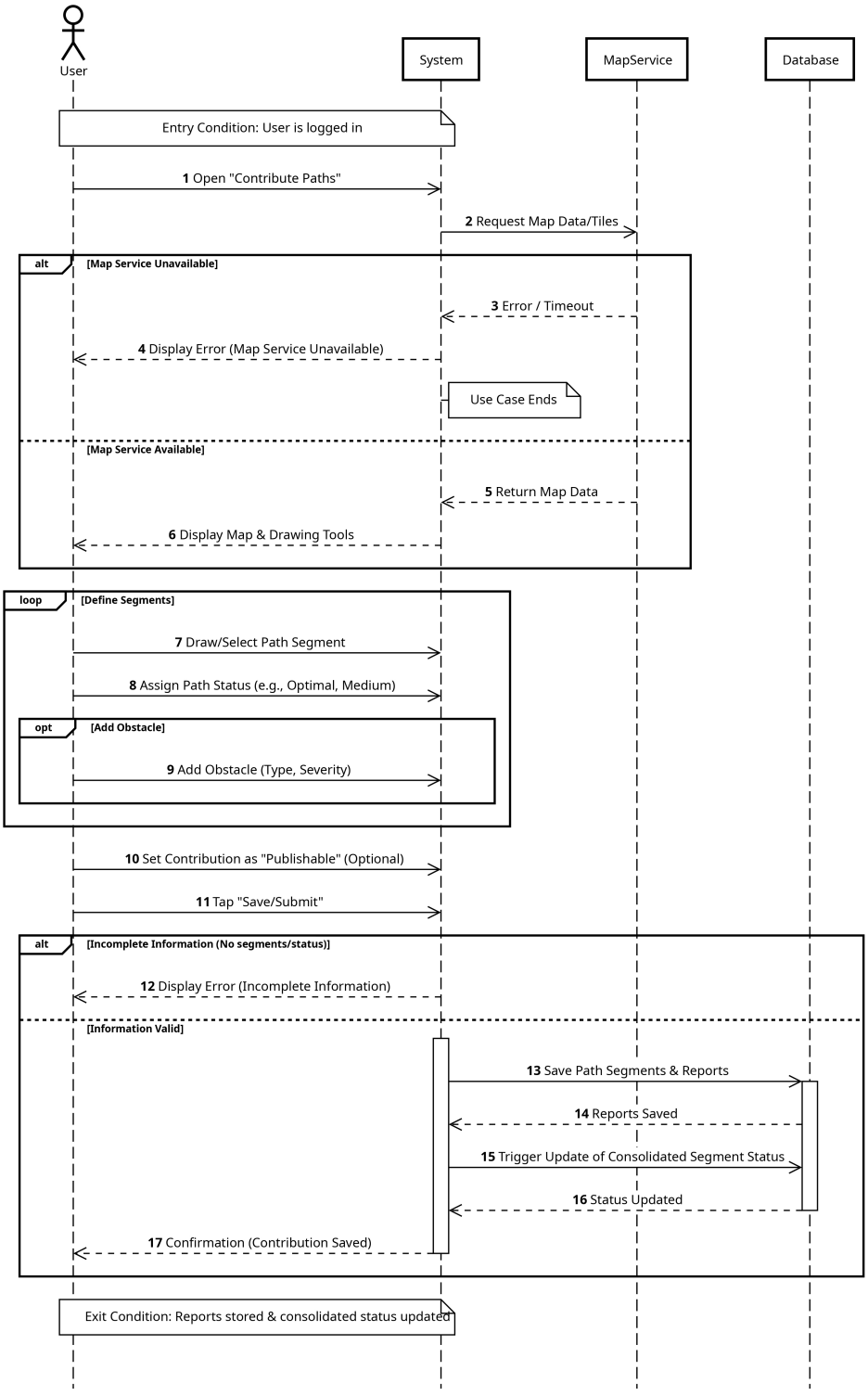

Figure 7: Manual Contribution Workflow Sequence Diagram

4. **Sensor assisted detection confirmation(UC5)**

   - While recording, the client can detect candidate issues based on sensor signals and location context.
   - Candidate issues are stored as private draft items for the user.
   - The application prompts the user to confirm, correct or discard each candidate issue.
   - Only confirmed and publishable items are written to the community visible collections.

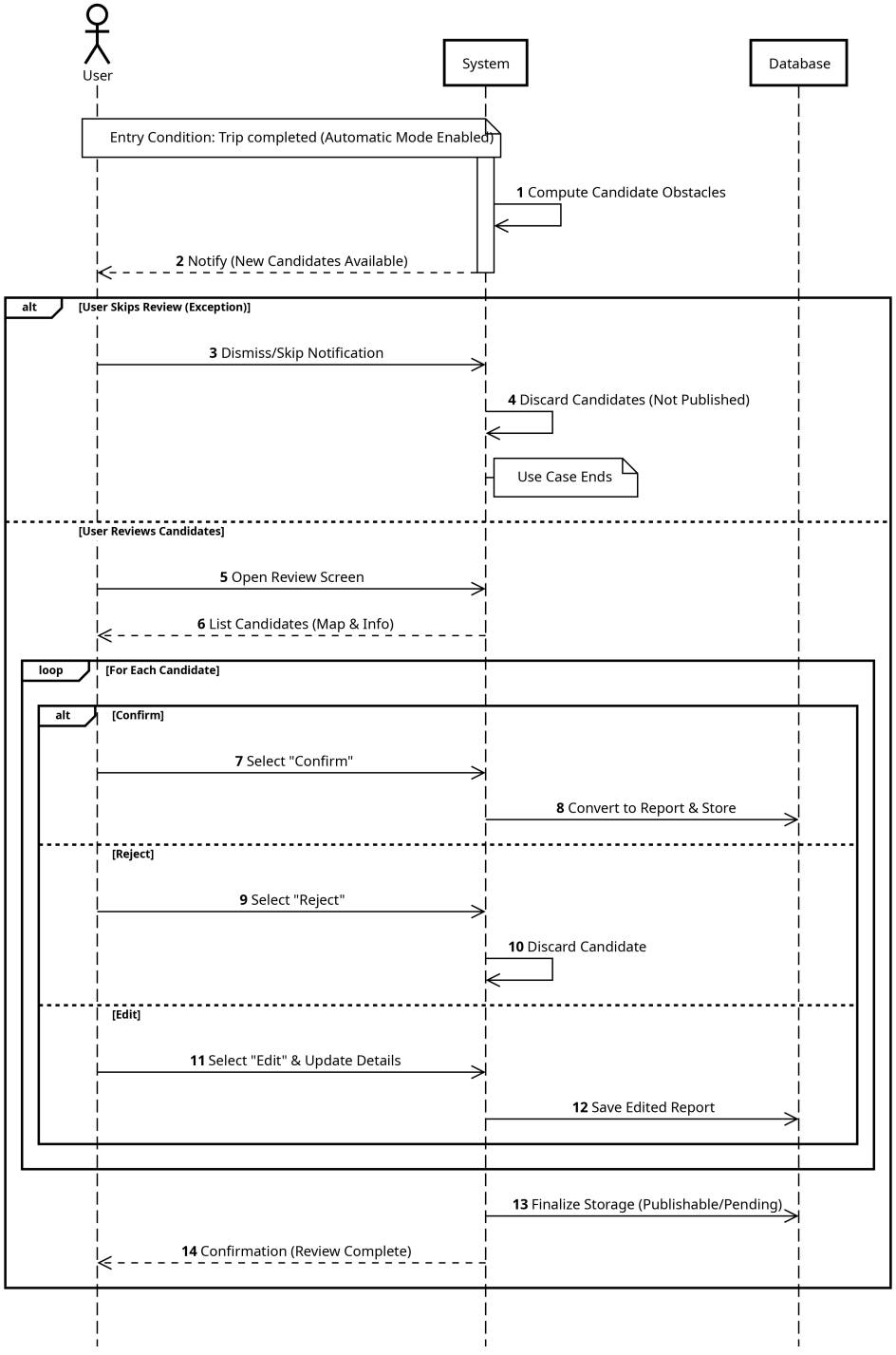

Figure 8: Sensor Assisted Detection Confirmation Sequence Diagram

5. **Route search, scoring and visualization(UC6)**

   - The user specifies origin and destination in the Flutter application.
   - The client requests candidate routes via a protected Cloud Function.
   - The Cloud Function calls the external map service to obtain candidate routes and associated polyline or segment representations.
   - The backend retrieves consolidated segment information from Firestore and computes a route score that combines quality and effectiveness factors.
   - The ranked routes are returned to the client for visualization and selection.
   - Generally, this behaviour reflects the formal cycling route research where candidate paths are ranked. Usually by a combined utility or attractiveness objective and sometimes under constraints. BBP implements a practical real time version of this idea for end users [2].

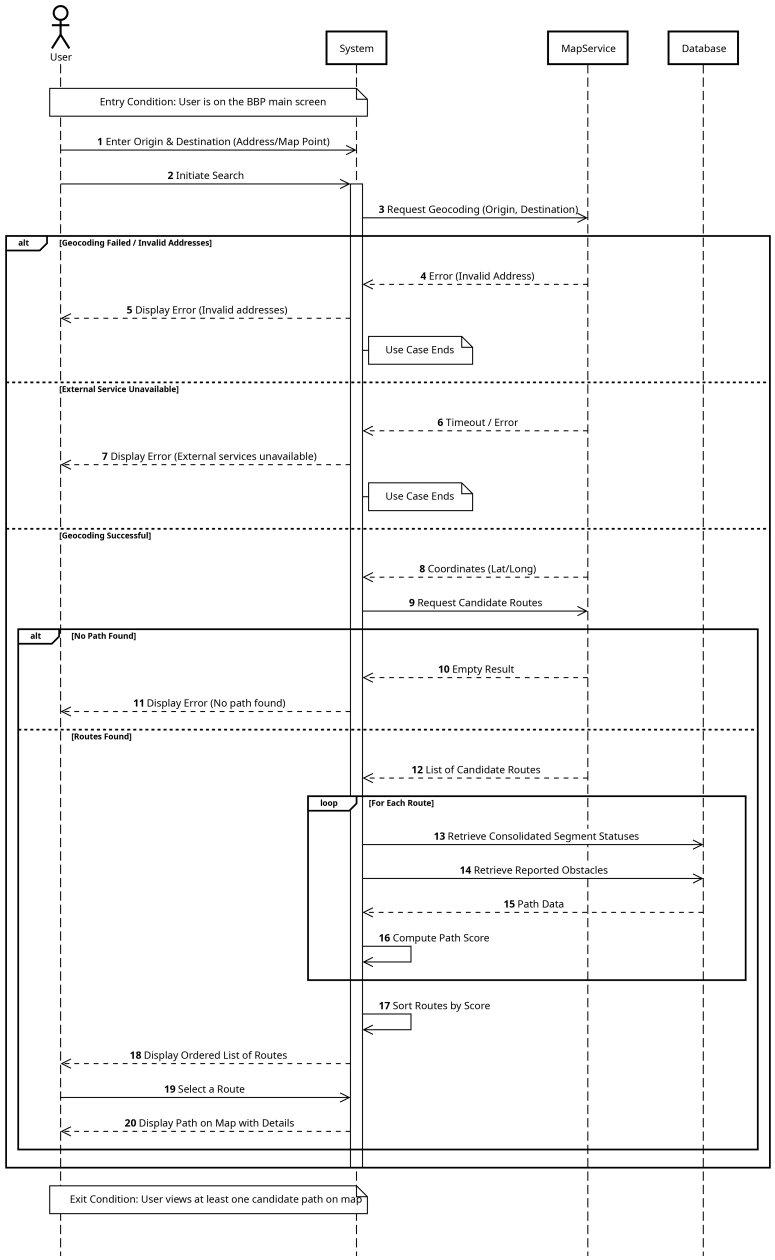

Figure 9: Route Search and Scoring Sequence Diagram

6. **Merging and consolidated status updates(UC7)**

    - When multiple reports affect the same segment, a backend function computes a consolidated status by applying freshness and confirmation logic.
    - The consolidated read model is stored in a dedicated collection for efficient querying during route scoring and map rendering.
    - Periodic recomputation may be scheduled via Cloud Scheduler to ensure consistency and to incorporate aging policies.

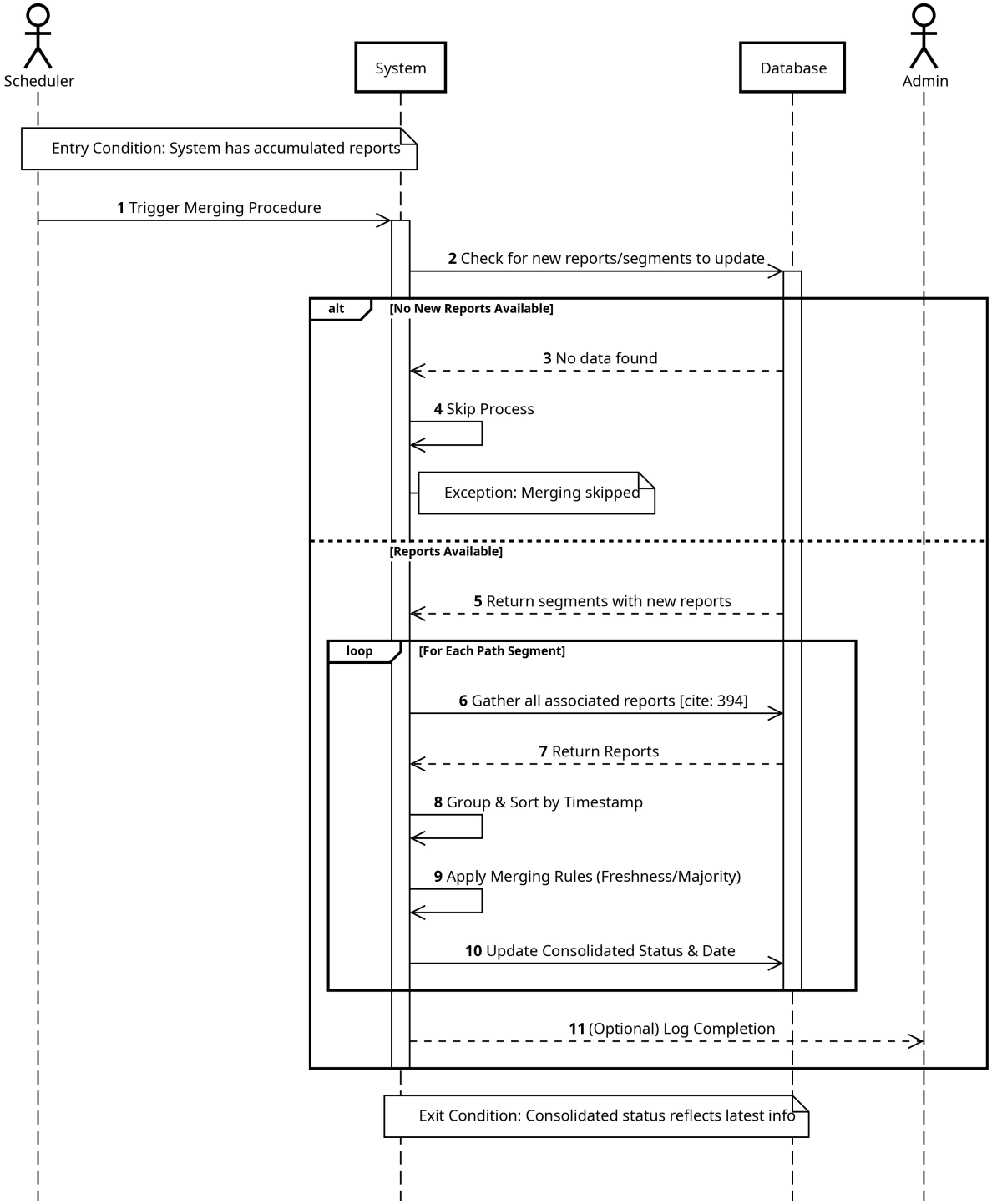

Figure 10: Merging and Consolidated Status Update Sequence Diagram

2.5 Interface View

BBP exposes interfaces between the client and the backend through Firebase SDK operations and Cloud Functions endpoints.

1. **Client to Firebase Authentication:** Registration, login, password reset and token management are handled through the Firebase SDK.
2. **Client to Firestore:** The client reads and writes documents according to Security Rules. Typical reads include user profile, trip list, trip summaries and route results. Typical writes include trip metadata, user preferences and user owned contribution drafts.
3. **Client to Cloud Functions:** Cloud Functions provide APIs for operations requiring server side enforcement or third party integrations such as:
    - Requesting candidate routes and performing route scoring
    - Triggering consolidation and merge processes
    - Performing validation and normalization of submissions
    - Retrieving weather enrichment for trips when needed
4. **Cloud Functions to External APIs:** Cloud Functions call map and weather services using secure stored credentials. Responses are transformed into BBP domain structures before being persisted or returned.

2.6 Architectural Style and Patterns

BBP adopts the following architectural styles and patterns:

1. **Serverless architecture:** The backend relies on managed services and function based computation. This supports scalability and reduces infrastructure management.
2. **Layered architecture:** Presentation logic resides on the Flutter client, application logic resides in Cloud Functions and persistence is handled by Firestore and Cloud Storage.
3. **Event driven and reactive data access:** Firestore enables real time updates for relevant UI elements. Backend functions may be triggered by database events to update derived models.
4. **CQRS inspired read models:** Consolidated segment status is stored in a dedicated collection as a derived read model optimized for frequent route scoring queries. This reduces repeated computation during interactive route searches.

2.7 Data Management

BBP stores and manages data in a way that supports both personal trip logs and community maintained path quality.

1. **Core entities stored in Firestore**
    - Users and preferences
    - Trips metadata and summary statistics
    - Segment reports and obstacle reports with publishable or private designation
    - Consolidated segment statuses used for scoring and map visualization
2. **Large payloads stored in Cloud Storage**

- Full GPS traces and dense samples
- Optional attachments related to reports

3. **Derived read models:** Consolidated segment statuses are maintained in a separate collection to support low latency reads during route scoring. Updates occur when new publishable reports arrive and may also be recomputed periodically.

4. **Consistency model:** Firestore provides strong consistency for document reads, while derived models may be eventually consistent depending on trigger timing. The design prioritizes user facing responsiveness, while ensuring that consolidation logic converges to stable results.

2.8 Security Design

Security is enforced through a combination of authentication, authorization rules and controlled server side execution.

1. **Authentication:** Users authenticate through Firebase Authentication. Identity tokens are required for operations tied to personal data or contributions.

2. **Authorization:** Firestore Security Rules restrict reads and writes based on user identity and ownership. Private trip data and private candidate issues are accessible only to their owners. Community visible publishable data is readable according to the intended visibility rules.

3. **Protected operations:** Cloud Functions encapsulate operations that require secrecy, controlled validations or merging logic. External API keys are not embedded in the client.

4. **Privacy by design:** The system maintains a strict separation between private user data and publishable community data. Publication is always an explicit user choice.

2.9 Design Decisions and Rationale

The chosen architecture reflects the project constraints and the expected usage patterns.

1. Firebase services provide a consistent ecosystem for authentication, real time data access and secure server side logic, reducing integration complexity.

2. Cloud Functions centralize business rules such as merging and scoring, which improves integrity and prevents client side manipulation.

3. The use of a consolidated status read model improves performance during route searches by avoiding expensive aggregation at query time.

4. Cloud Storage is used for large traces to keep Firestore efficient and to reduce document size constraints.

5. External API calls are handled server side to protect secrets, manage quotas and provide a stable contract to the client.

3 User Interface Design

This section maps UI screens to RASD use cases.

3.1 Registration

- Registration form
- Login form
- Blocked account message

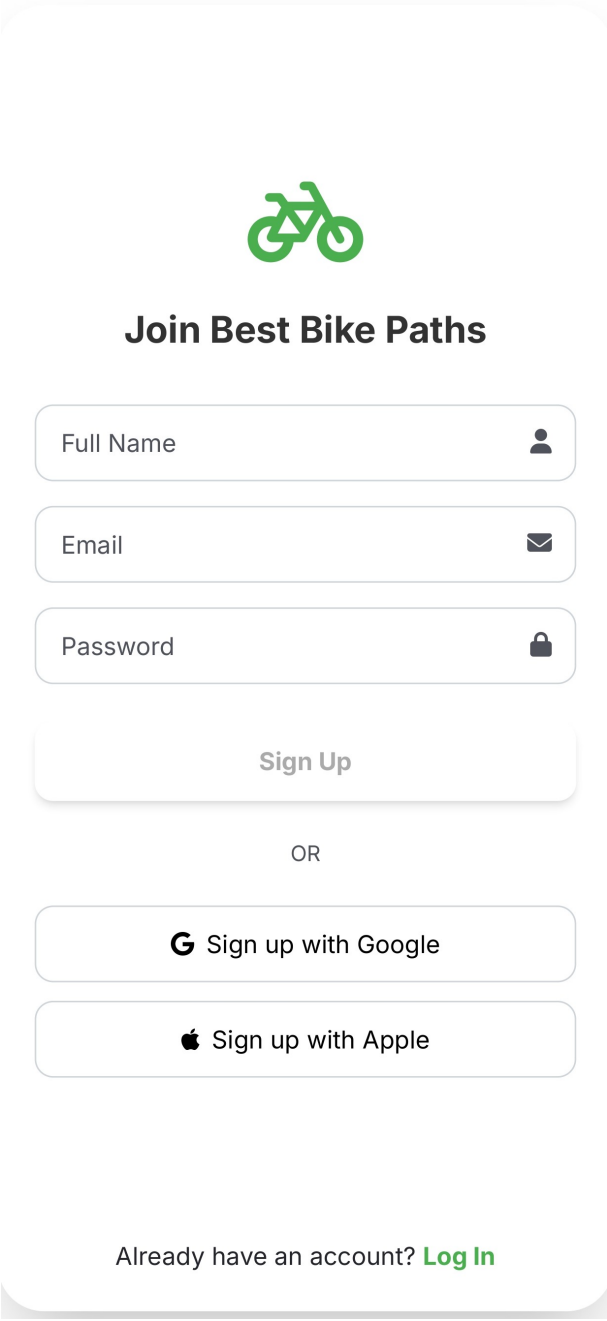

Figure 11: User Registration Screen

3.2 Dashboard

- Start recording
- Quick route search
- Recent trips

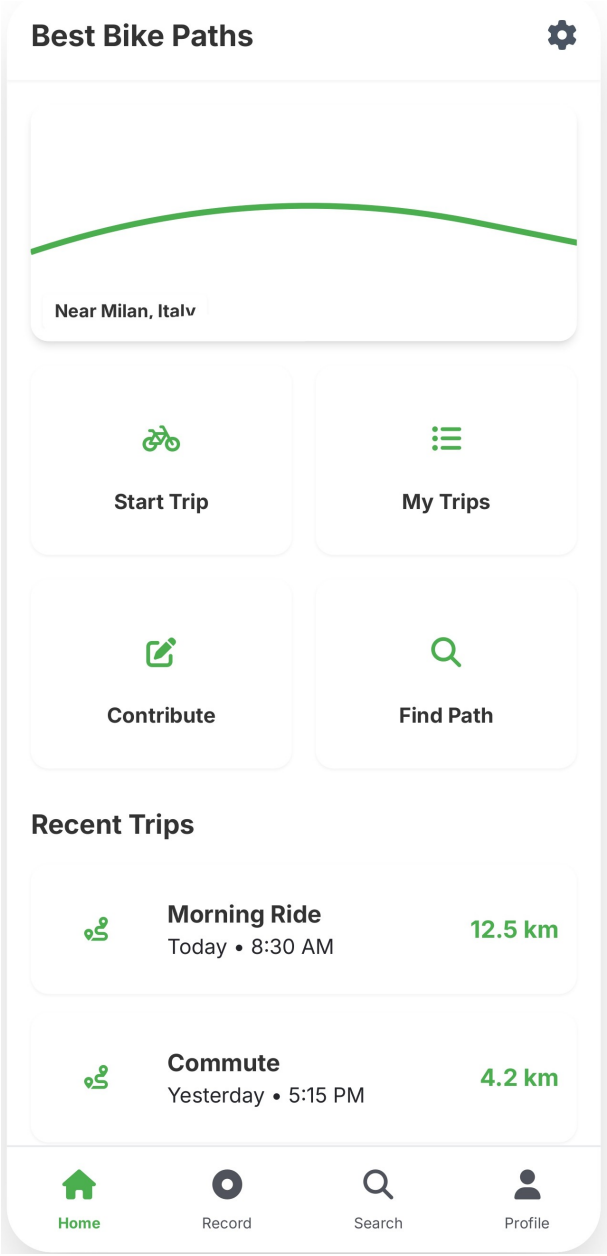

Figure 12: Dashboard Screen

3.3 Trip Recording

- Start/Stop
- Stats (distance, duration, speed)
- Automatic mode indicator (if enabled)

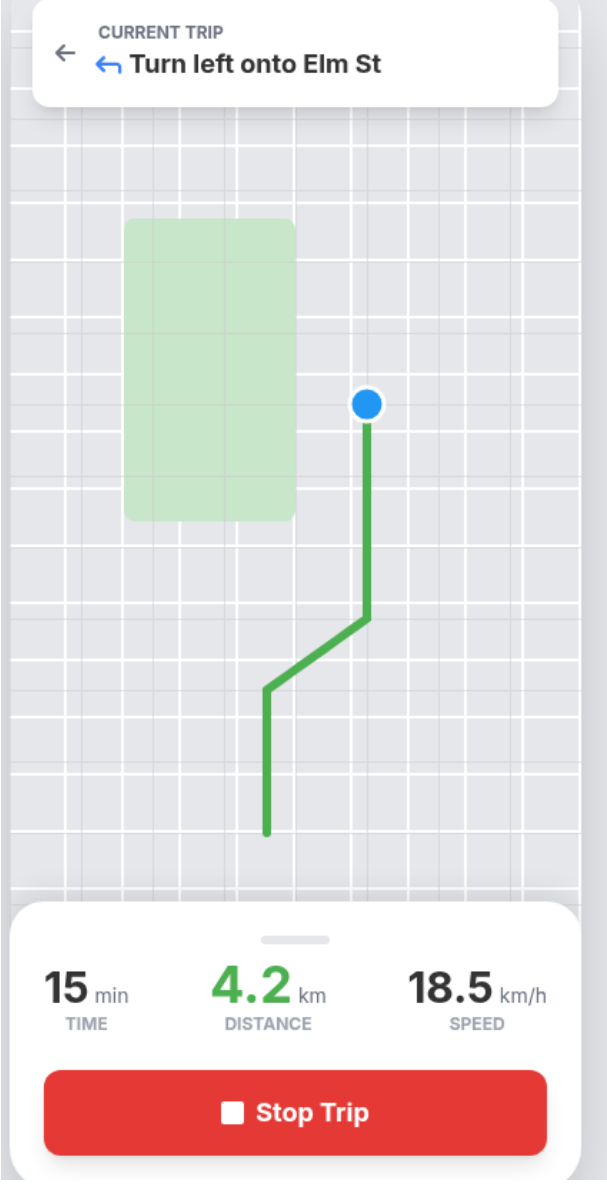

Figure 13: Trip Recording Screen

3.4 Trip History & Details

- List trips
- Details view: map, stats, weather snapshot

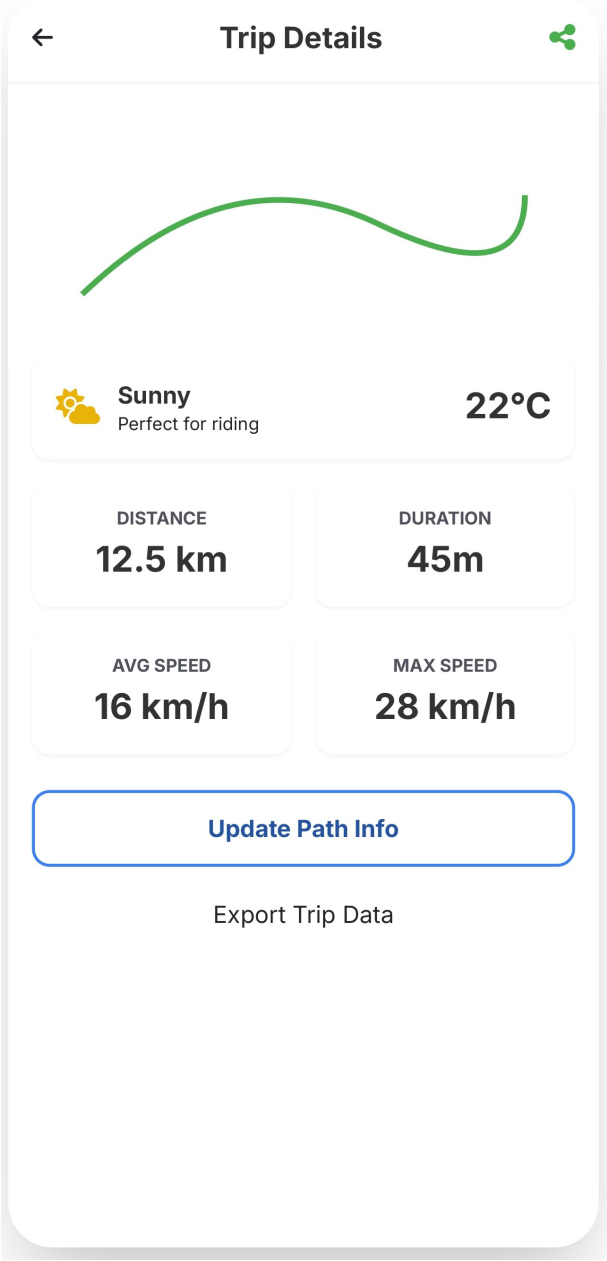

Figure 14: Trip History and Details Screen

3.5 Manual Contribution

- Select/draw segment(s)
- Assign status
- Add obstacles
- Publishable/private toggle
- Edit/delete previous reports

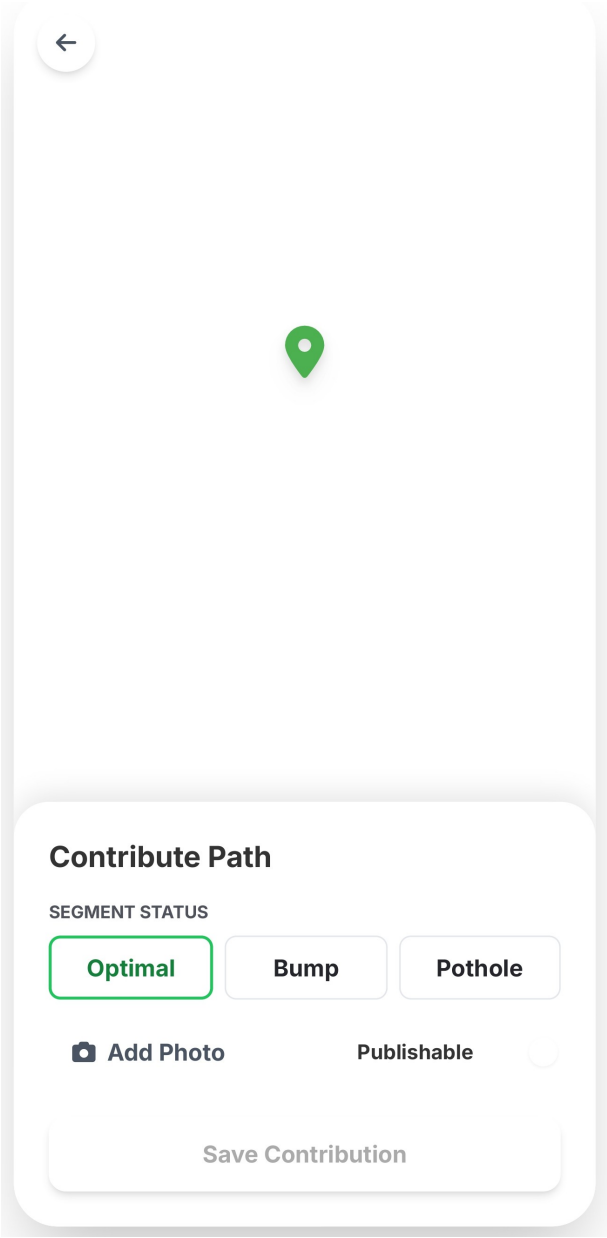

Figure 15: Manual Contribution Screen

3.6 Automatic Review

- Candidate list/map
- Confirm/reject/edit candidate
- Finalize

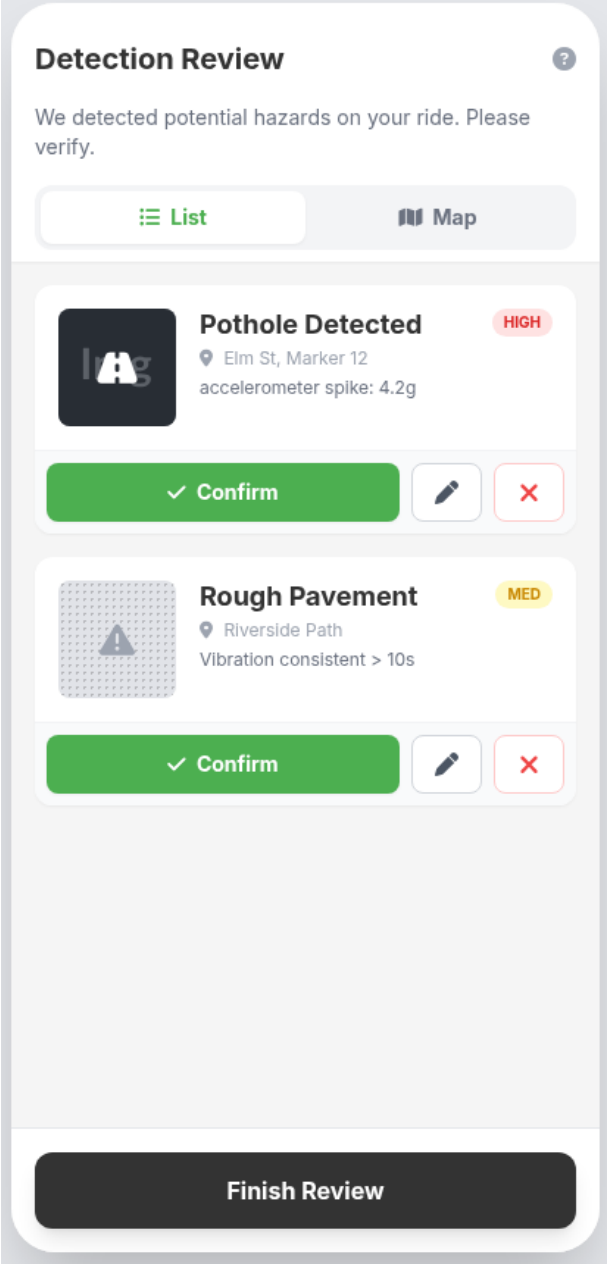

Figure 16: Automatic Detection Review Screen

3.7 Route Search & Results

- Enter origin/destination
- Candidate list ordered by score
- Detail explanation
- Map overlay visualization (segment status + obstacles)

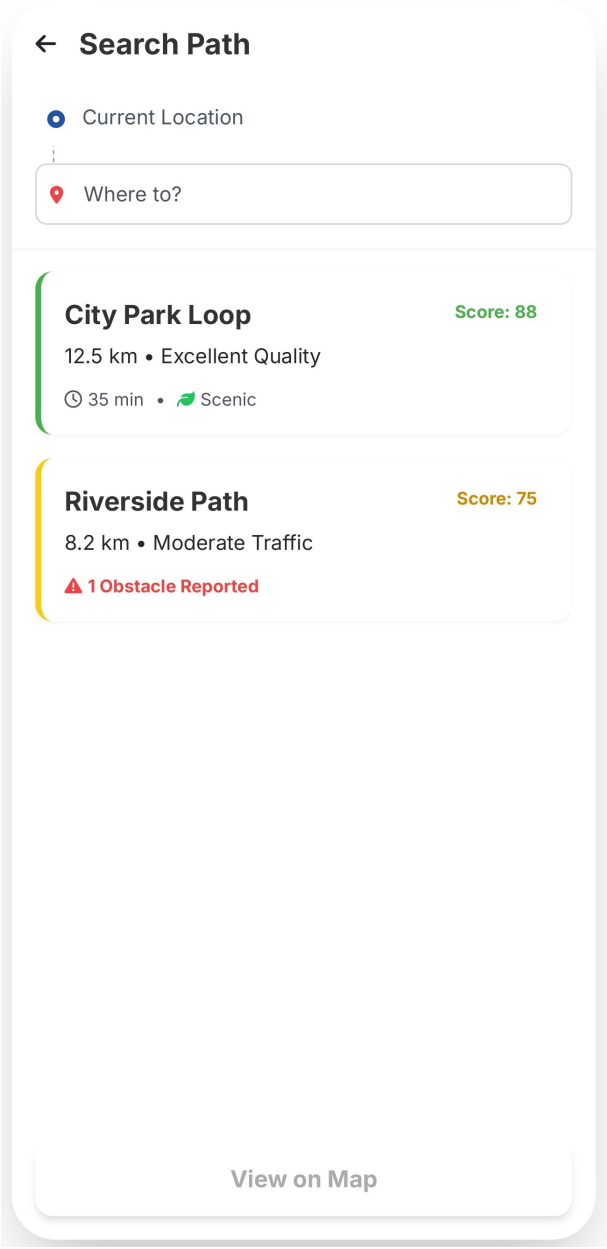

Figure 17: Route Search and Results Screen

3.8 Admin

- Block user
- Hide/remove report or obstacle

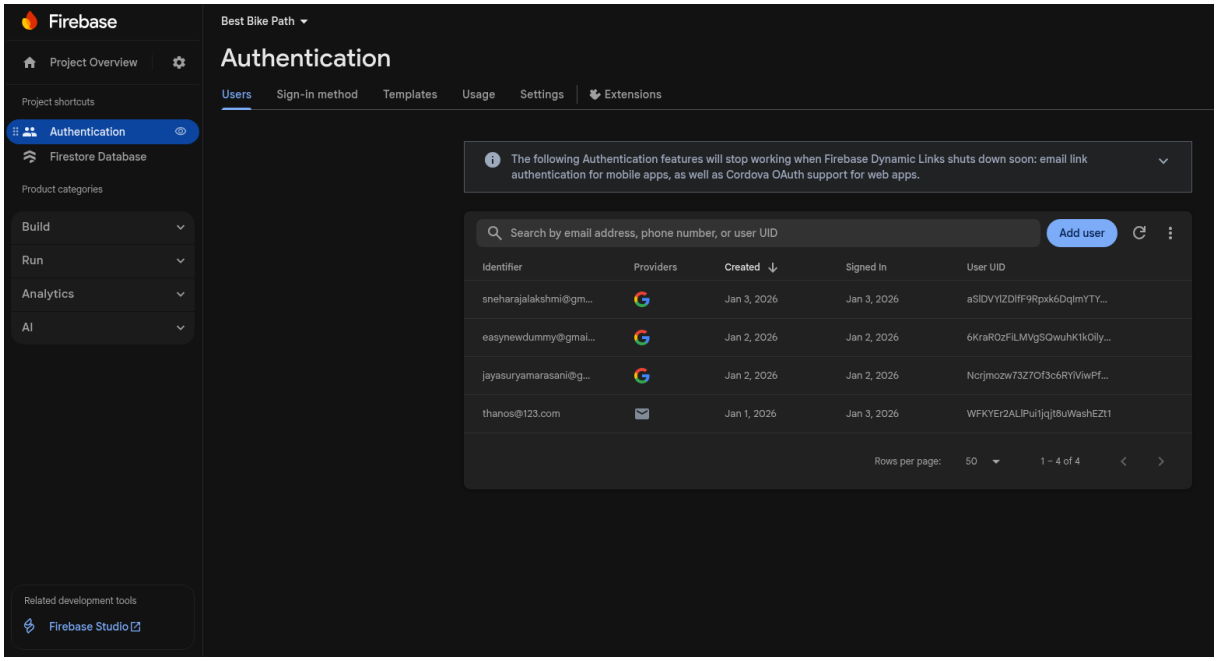

Figure 18: Admin Screen

4 Requirements Traceability

4.1 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.

4.2 Requirements Traceability Table

Table 2: Requirements traceability table

Req ID	Requirement (short)	Flutter Module(s)	Firebase Services	Main Data/Artifacts
R1	User registration	A1	F1, F2	`users/{uid}`
R2	User login	A1	F1, F2	`users/{uid}`
R3	Start trip recording	A2	(F2 optional)	local GPS buffer
R4	Stop trip recording	A2, A3	F2, F3, (F4 optional)	`trips/{tripId}` + trace file
R5	Trip statistics	A2, A3	F2, (F4 optional)	fields in `trips/{tripId}`
R6	Trip history	A3	F2	query trips by uid
R7	Trip weather enrichment	A3	F4, F2	`trips.weatherSnapshot`
R8	Manual path creation	A4	F2	pathSegments, pathSegmentReports
R9	Segment status assignment	A4	F2	pathSegmentReports.segmentStatus
R10	Obstacle creation	A4	F2, (F3 optional)	obstacles (optional photo in Storage)
R11	Publishability	A4	F2	publishable flag on reports and obstacles
R12	Edit manual contributions	A4	F2, (F4 validation optional)	update or delete own pathSegmentReports
R13	Enable/disable automatic mode	A1, A2, A5	F2	`users.automaticModeEnabled`
R14	Sensor data logging	A2	(F2 optional), (F3 optional)	sensor samples + GPS tags
R15	Detect candidate obstacles	A5	(F4 optional)	candidate events (local or stored)
R16	Candidate list presentation	A5	none required	candidate UI list or map
R17	User confirmation or correction	A5	F2	confirmed or edited candidate data
R18	Conversion into reports	A5	F2	create pathSegmentReports and obstacles
R19	Public path search	A6	F2, F4	read consolidated data + compute route
R20	Route computation	A6	F4 + external routing	computeRoutesAndScores() + MapService
R21	Path scoring	A6	F4, F2	uses consolidatedStatuses + obstacles
R22	Ordered path list	A6	F4	sorted candidates returned
R23	Path visualization	A6	F2	overlays from consolidated statuses + obstacles

Req ID	Requirement (short)	Flutter Module(s)	Firebase Services	Main Data/Artifacts
R24	Segment report storage	A4, A5	F2	pathSegmentReports with uid + timestamps
R25	Periodic merging	none	F5, F4, F2	refresh consolidatedStatuses
R26	Freshness handling	none	F4, F2	timestamp weighting in merge
R27	Majority handling	none	F4, F2	majority or weighted-majority merge rule
R28	User blocking	A7	F1 (role), F2, F4	users.blocked + admin enforcement
R29	Data removal or hide	A7	F2, F4	hide or delete report + recompute merge

5 Implementation, Integration and Test Plan

5.1 Overview and adopted strategies

The implementation plan defines the order in which BBP is built, while the integration and test plan defines how BBP is assembled and validated step by step. The two plans must be consistent, because integration testing becomes effective only when it follows the same sequence used to combine components. BBP does not use a big bang integration approach, where all modules are integrated only at the end. Big bang provides late feedback and makes fault localization expensive. Instead, BBP follows an incremental and iterative integration process where components are integrated and tested as soon as they are available.

The BBP adopts the following approach as our integration strategies:

1. **Thread Based Integration:** BBP integrates the system by delivering complete user visible features, each requiring cooperation among multiple modules. This provides early working functionality and reduces the need for artificial drivers compared to purely structural integration.

2. **Bottom Up Integration inside each thread:** Within each feature, BBP integrates starting from the lower level elements such as data models, repositories, persistence and backend functions, then connects state management and UI. Bottom up integration reduces the need for stubs, but often requires drivers to exercise low level units, which BBP provides through repository and function level test harnesses.

3. **Critical Modules First:** BBP prioritizes high risk modules early specifically GPS tracking, sensor sampling and candidate detection feasibility and MapService integration, so feasibility issues are discovered early.

Testing in isolation often requires scaffolding because modules depend on other modules. BBP uses stubs and drivers when needed to simulate missing units during incremental integration.

5.2 Implementation Plan

BBP is divided into features to implement the thread based strategy. The RASD use cases and requirements are the critical components to implement. A feature is intended as a coherent functionality that becomes usable to the user and typically requires integration across several subcomponents.

Each feature has a significant outcome and provides a stable base for further development. The following are the identified features and the order of implementation feature-wise.

1. **F1 - Sign Up and Sign In (UC1, UC2, R1, R2):** Implementation of registration and login through Auth, create user profiles in Firestore and enforce blocked user handling at login.

2. **F2 - Trip Recording Core (UC3 partial, R3, R5 computation base):** Implementation of the live trip tracking loop using GPS, including permission handling, start and stop controls, local buffering of GPS points and real time computation of core statistics while recording.

3. **F3 - Trip Persistence, History and Details (UC3 completion and R4, R5, R6):** Persist completed trips to Firestore, upload traces to Cloud Storage and implement trip list and trip details views. This feature makes recorded trips durable and reviewable.

4. **F4 - Weather Enrichment (UC3 extension, R7):** Add optional weather enrichment after trip persistence is stable, storing weather snapshots in trip records when the service is reachable.

5. **F5 - Manual Path Contributions (UC4, R8 to R12, R24 partially):** Implement map based segment selection or creation, segment status assignment, obstacle reporting, publishable or private selection and editing or deletion of own contributions.

6. **F6 - Automatic Detection Review (UC5, R13 to R18):** Implement automatic mode settings, sensor sampling during trips, candidate detection, candidate review UI and conversion of confirmed candidates into reports and obstacles.
7. **F7 - Route Search (UC6 part 1, R19, R20):** Implementation of origin and destination input, geocoding and retrieval of candidate routes from MapService. This feature focuses on obtaining feasible route candidates.
8. **F8 - Path Scoring and Visualization (UC6 part 2, R21 to R23):** Implementation of segment mapping, retrieval of consolidated statuses and obstacles, scoring of candidates, ordered results list and map visualization with overlays.
9. **F9 - Report Merging into Consolidated Status (UC7, R24 to R27):** Implement periodic merging with freshness weighting and majority resolution and maintain consolidatedStatuses as the derived read model used for fast scoring.
10. **F10 - Administration and Moderation (R28, R29):** Implement admin operations to block users and hide or remove problematic reports and obstacles, enforced through roles, security rules and backend logic.

5.3 Component Integration and Testing

Integration testing focuses on the interaction between components and the correctness of their interfaces because many failures arise from inconsistent assumptions, misinterpreted parameters and unintended side effects across modules.

BBP integrates by feature threads and tests after each integration step rather than deferring testing to the end. It uses Firebase Emulator Suite for Auth, Firestore, Functions and Storage to run repeatable integration tests, validate security rules and avoid dependence on a remote environment during continuous development. External services, MapService and Weather service are outside the system boundary. BBP therefore does not re-test their internal correctness, but it validates the integration points through stubs, controlled inputs and failure mode testing such as timeouts.

5.3.1 Integration steps by feature thread

1. **F1 - Sign Up and Sign In:**
    - **Integrated components:** Auth UI, Firebase Authentication, Firestore user profiles, security rules for profile data.
    - **Integration tests:** register, login, profile creation, blocked user access denial.

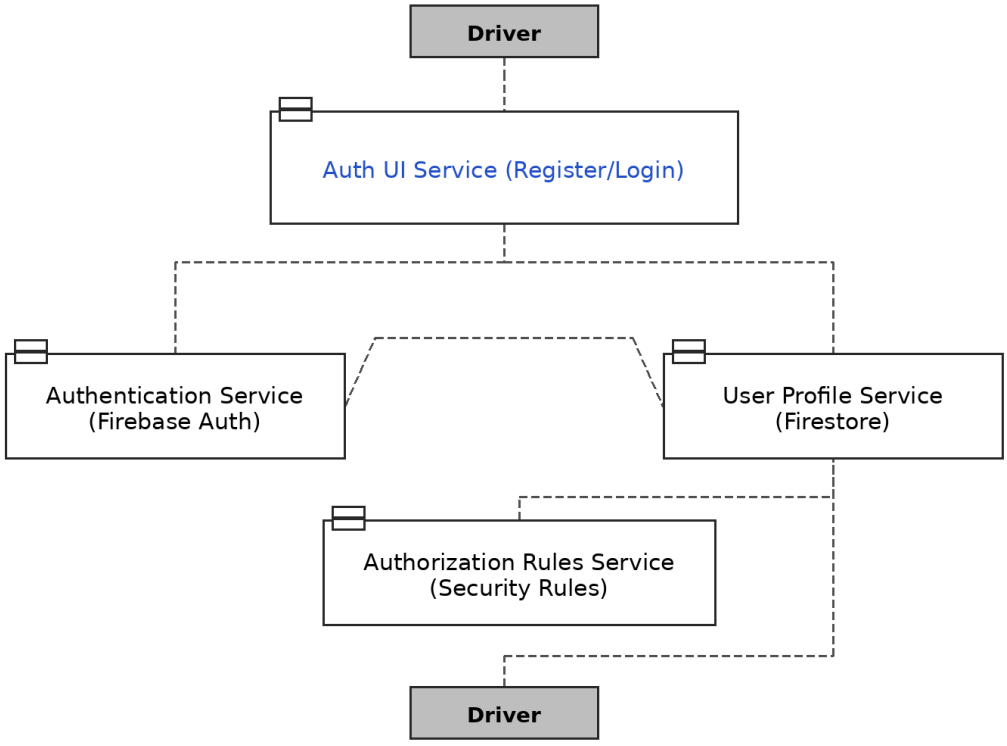

Figure 19: Integration testing diagram for Feature Thread F1 (Sign Up & Sign In)

2. **F2 - Trip Recording Core:**
    - **Integrated components:** GPS permission flow, GPS streaming, local buffer, real time statistics computation.
    - **Integration tests:** start trip, confirm GPS points are received, confirm stats update, stop trip produces a completed in memory trip object even before persistence.
    - **Drivers:** Simulated GPS streams for deterministic tests, plus at least one real device test for feasibility.

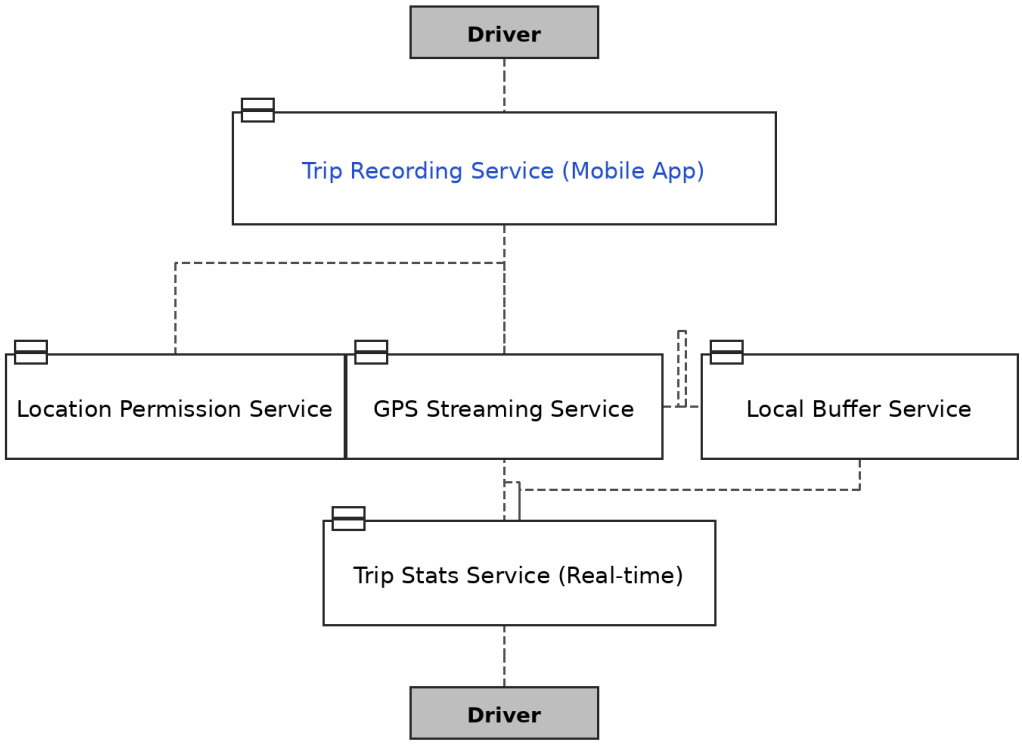

Figure 20: Integration testing diagram for Feature Thread F2 (Trip Recording Core)

3. **F3 - Trip Persistence, History and Details:**
    - **Integrated components:** Firestore trip writes, Storage trace upload, trip list query by uid, trip details rendering.
    - **Integration tests:** Stopping a trip creates trips/tripId and uploads a trace, trip history displays correct items, opening a trip shows stored stats and trace link.

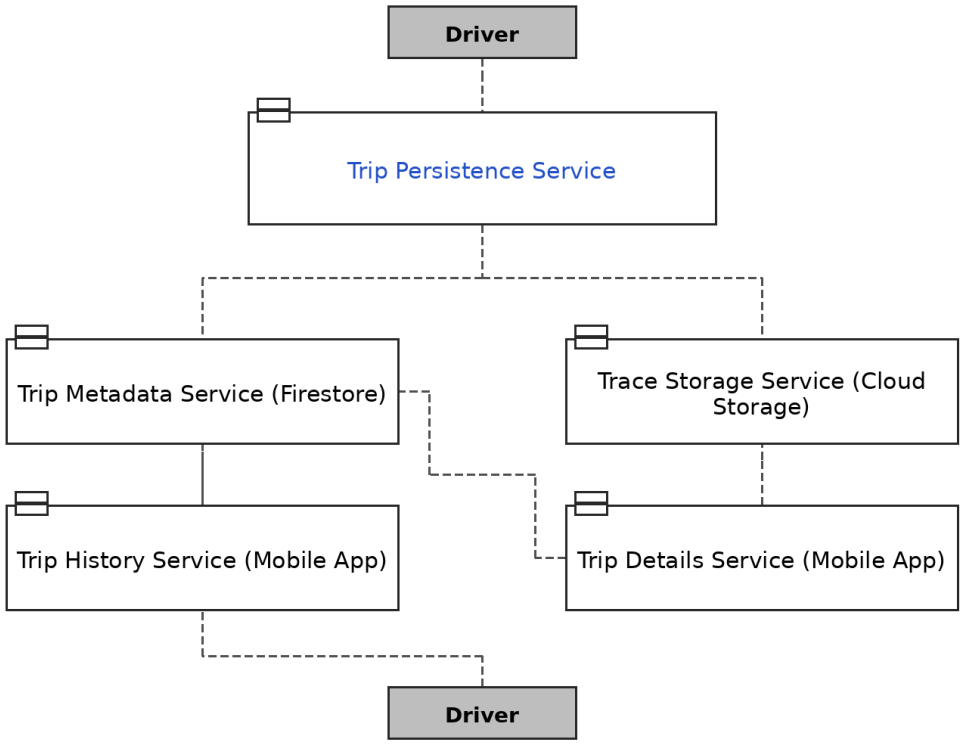

Figure 21: Integration testing diagram for Feature Thread F3 (Trip Persistence, History and Details)

4. **F4 - Weather Enrichment:**
    - **Integrated components:** weather client wrapper and trip update pipeline.
    - **Integration tests:** reachable weather provider stores snapshot, unreachable provider results in trip stored without weather, with a clear fallback state.

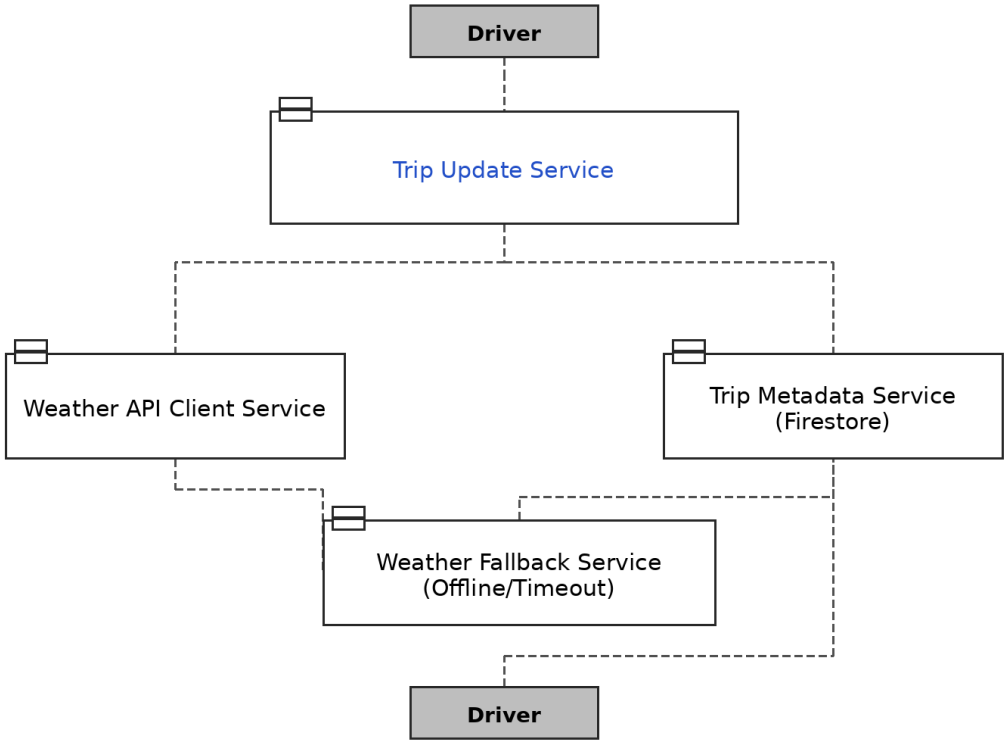

Figure 22: Integration testing diagram for Feature Thread F4 (Weather Enrichment)

5. **F5 - Manual Path Contributions:**
    - **Integrated components:** segment creation or selection, report creation, obstacle creation, publishable or private logic, edit and delete own reports.
    - **Integration tests:** create publishable reports and verify guest visibility rules, create private reports and verify only owner access, edit and delete restrictions validated by rules tests.

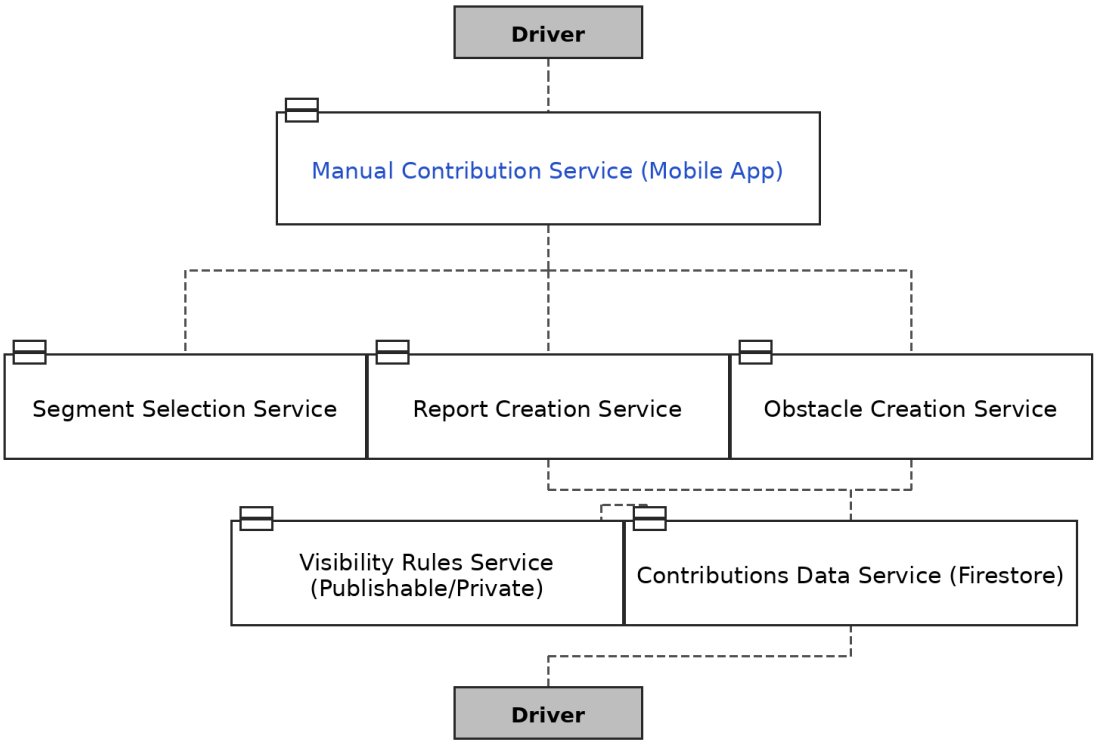

Figure 23: Integration testing diagram for Feature Thread F5 (Manual Path Contributions)

6. **F6 - Automatic Detection Review:**
    - **Integrated components:** ensor sampling with GPS tagging, candidate detector, candidate review UI, conversion into reports and obstacles.
    - **Integration tests:** candidate list produced deterministically using prerecorded sensor traces, confirm creates stored artifacts, reject creates none, edits modify stored artifacts.

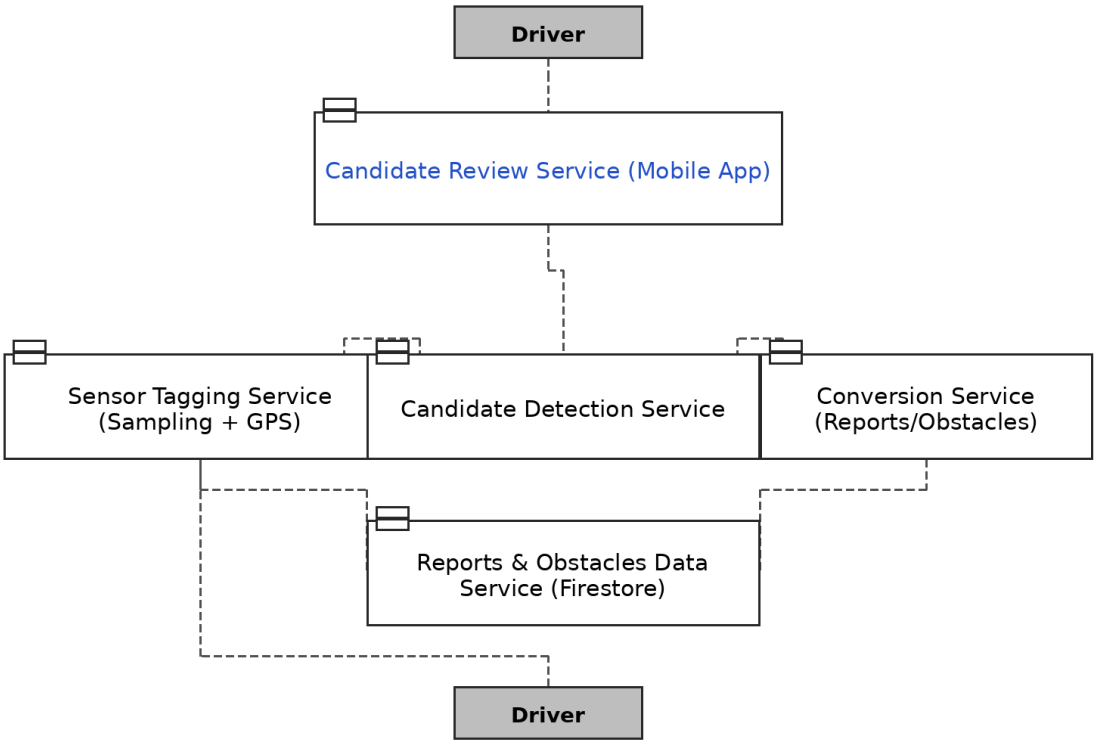

Figure 24: Integration testing diagram for Feature Thread F6 (Automatic Detection Review)

7. **F7 - Route Search:**

   - **Integrated components:**origin and destination UI, geocoding, candidate route retrieval from MapService.
   - **Integration tests:** valid locations return candidates, invalid address fails gracefully, timeouts handled with fallback messages.
   - **Stubs:** MapService stub used for deterministic automated tests, replaced by real MapService for system tests.

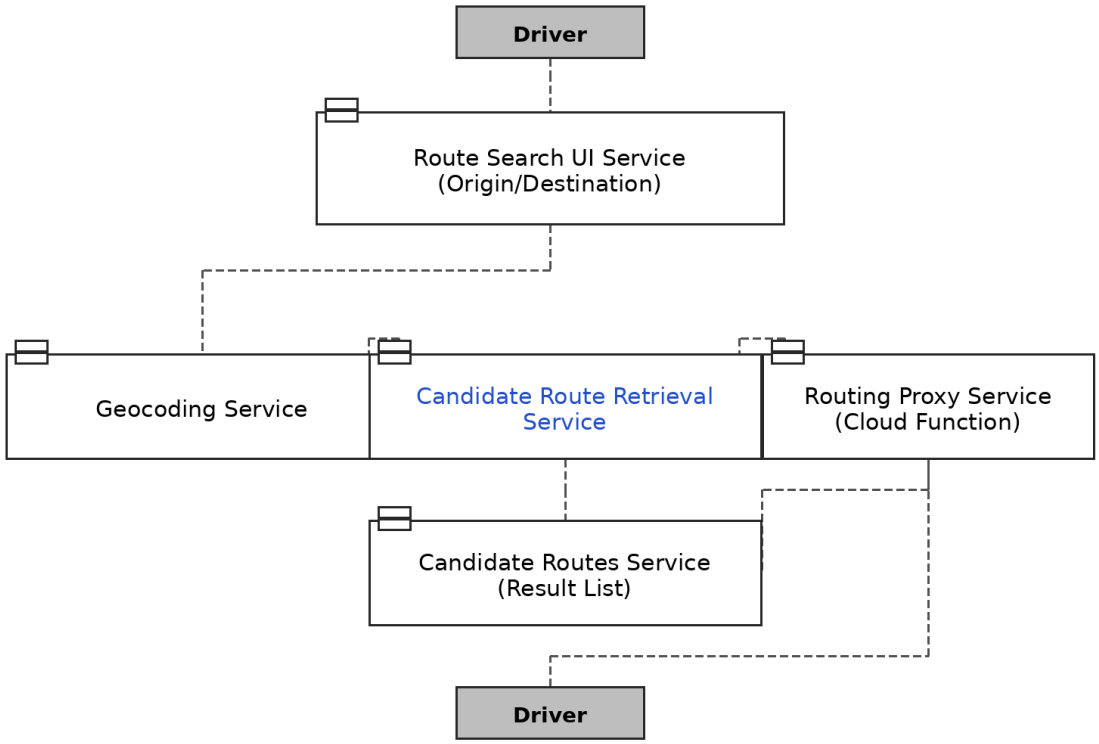

Figure 25: Integration testing diagram for Feature Thread F7 (Route Search)

8. **F8 - Path Scoring and Visualization:**
    - **Integrated components:**segment mapping, retrieval of consolidatedStatuses and obstacles, scoring logic, sorted list, map overlays.
    - **Integration tests:** seeded Firestore dataset produces predictable scores, verify ordering by score, verify overlays match consolidated statuses and obstacles, verify guest sees only publishable and non hidden data.

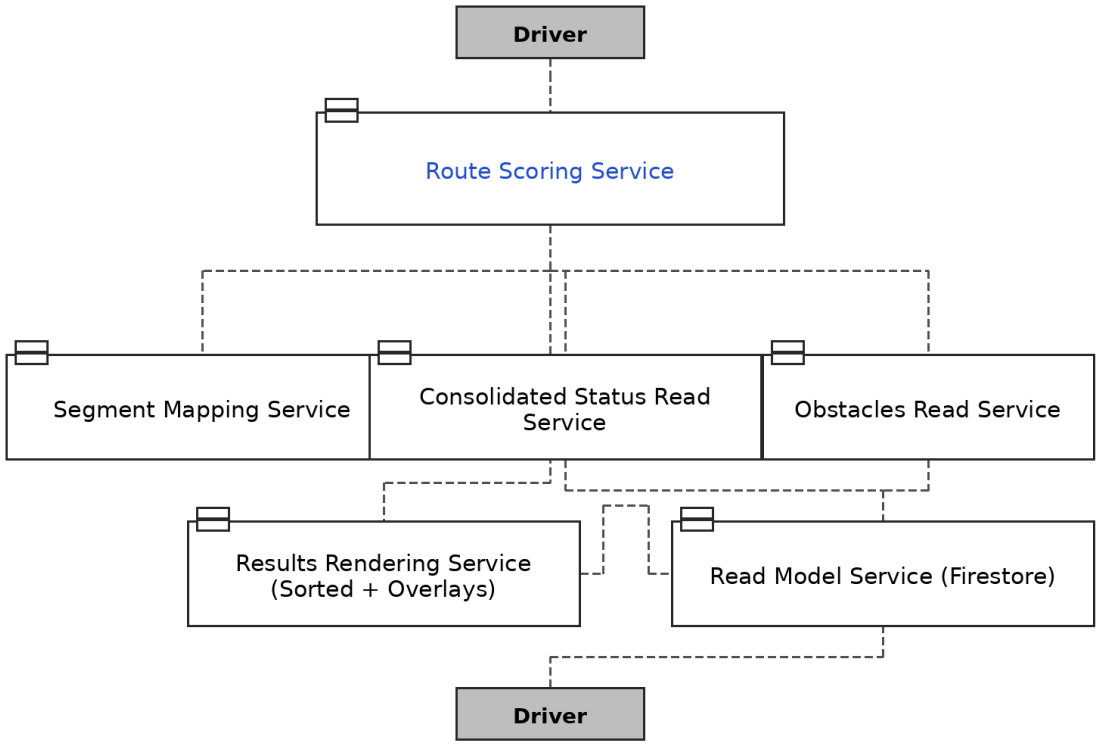

Figure 26: Integration testing diagram for Feature Thread F8 (Path Scoring and Visualization)

9. **F9 - Report Merging into Consolidated Status:**
    - **Integrated components:** scheduled merge function, freshness weighting, majority resolution, consolidatedStatuses updates.
    - **Integration tests:** seed reports with known timestamps and statuses, execute merge manually in emulator, verify consolidated result, verify hidden and private items excluded.

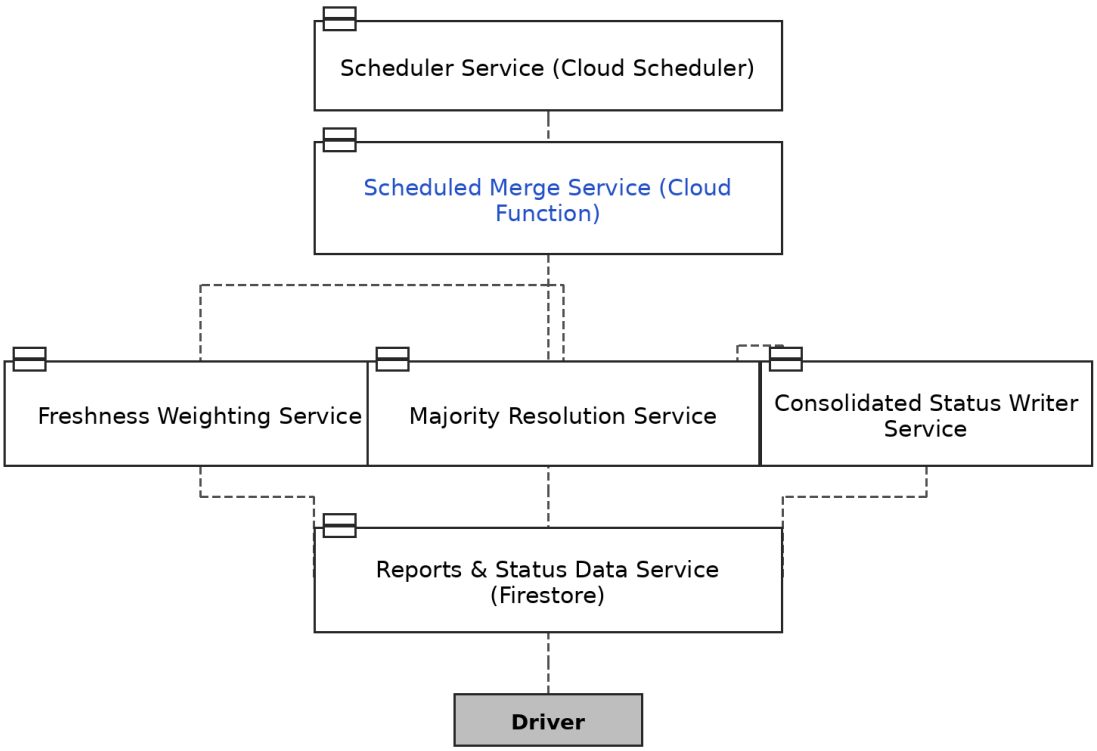

Figure 27: Integration testing diagram for Feature Thread F9 (Report Merging into Consolidated Status)

10. **F10 - Administration and Moderation:**
    - **Integrated components:** admin restricted UI, admin Cloud Functions, rule enforcement, optional audit logging.
    - **Integration tests:** only admin can block users, blocked users cannot contribute, admin hide and remove excludes data from consolidated results after merge.

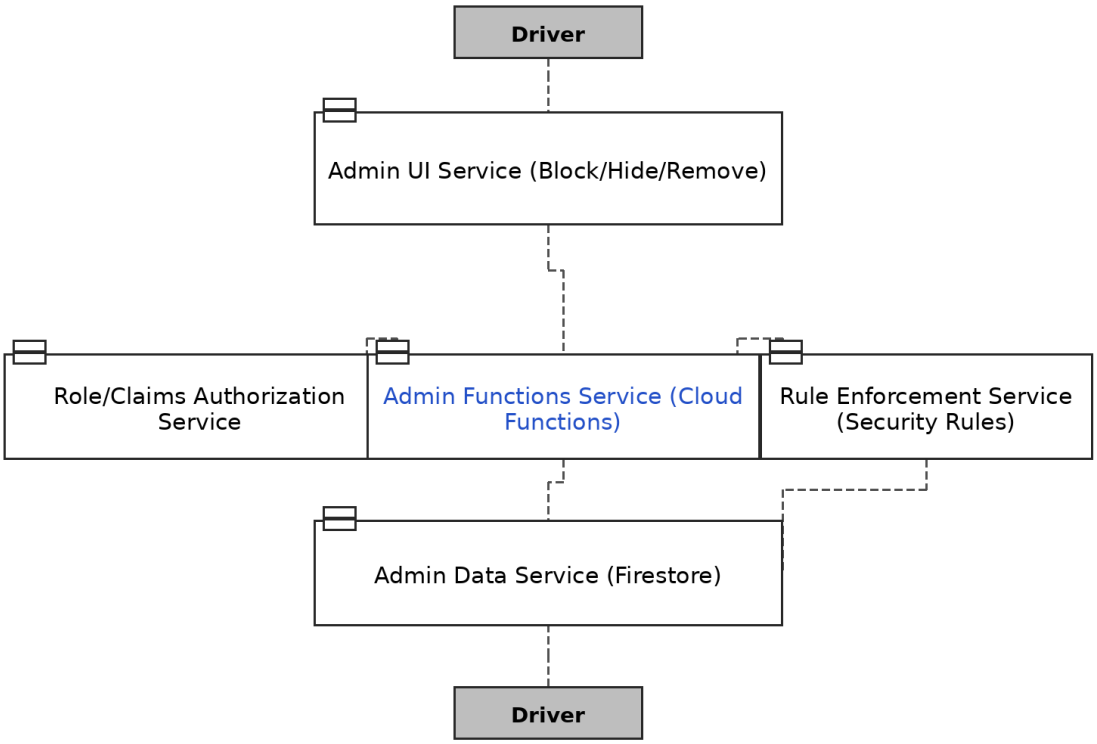

Figure 28: Integration testing diagram for Feature Thread F10 (Administration and Moderation)

5.4 System Testing

System testing validates BBP as a whole, using the RASD use cases as scenarios. The test environment is kept as close as possible to production, especially for GPS, sensors and external services.

System tests are executed after each major feature thread becomes stable and repeated as regression tests as the system grows.

5.4.1 Functional System tests by use case

1. **UC1 - Registration**
    - Valid registration creates Auth user and Firestore profile
    - Invalid email or password rejected
    - Duplicate email rejected

2. **UC2 - Login**
    - Correct credentials allow access
    - Wrong credentials rejected
    - Blocked user denied after profile check
3. **UC3 - Record Trip**
    - Permission denied prevents recording
    - GPS unavailable handled safely
    - Normal trip produces saved trip and trace when persistence is enabled
    - Trip history shows new trip and details
    - Weather reachable adds weather snapshot, weather unreachable still stores trip
4. **UC4 - Manual Path Contribution**
    - Create segment reports for each status
    - Add obstacles and verify correct storage and visualization
    - Publishable contributions visible to guests, private contributions visible only to owner
    - Edit and delete own contributions, not others' contributions
5. **UC5 - Automatic Detection Review**
    - Candidate detection produces list
    - Confirm creates PathSegmentReport and obstacle when applicable
    - Reject creates nothing
    - Edit modifies stored data before publishing
6. **UC6 - Search Route**
    - Route search returns candidate routes for valid origin and destination
    - Invalid address fails geocoding
    - MapService timeout handled gracefully
    - Scoring returns ordered list by decreasing score
    - Selected route shows overlays matching consolidated statuses and obstacles
7. **UC7 - Merge**
    - Merge updates consolidatedStatuses after new reports
    - Freshness weighting influences result
    - Majority rule followed
    - Hidden and private reports excluded

5.4.2 Non Functional System testing types

BBP includes performance, load and stress testing at system level.

1. Performance testing targets route scoring latency and overlay loading times.
2. Load testing targets repeated route searches and repeated reads of consolidated status data.
3. Stress testing targets failure and degraded conditions such as network loss or external service failure, ensuring BBP degrades gracefully and recovers.

5.5 Security tests and regression policy

Security and access control testing is treated as continuous integration work rather than a one time activity:

1. Guests cannot write any Firestore data
2. Users can only modify their own trips and contributions
3. Only admins can block users and hide or remove content
4. Blocked users cannot submit publishable contributions

Every defect found during integration or system testing is converted into a regression test, so the same fault does not reappear in the further tests. This supports incremental development and keeps integration stable as new features are added.

6 Effort Spent

Team Member	Jayasurya Marasani	Arunkumar Murugesan	Sneharajalakshmi Palanisamy	Section Total
Introduction	3	3	3	9
Architecture	10	10	10	30
UI Design	7	7	8	22
Requirements Traceability	8	8	8	24
Implementation, Integration and Test Plan	13	12	11	36
Total	**41**	**40**	**40**	**121**

Table 3: Effort spent per member

The table as shown in Table. 3 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] Valenzuela, A., Lopes, A., Rescarolli, M., Pazin, J., and Rech, C. Analysis of the quality of bicycle paths and their relationship with bicycle use in Florianópolis. *Journal of Physical Education*, vol. 34, article 13, 2023. doi:10.4025/jphyseduc.v34i1.3428.

[2] Černá, A., Černý, J., Malucelli, F., Nonato, M., Polena, L., and Giovannini, A. Designing optimal routes for cycle-tourists. *Transportation Research Procedia*, vol. 3, pp. 856–865, 2014. doi:10.1016/j.trpro.2014.10.064.

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

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

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

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

