POLITECNICO DI MILANO

1863

Computer Science and Engineering

Software Engineering II

2025 – 2026

ATD

Acceptance Testing Document

Best Bike Paths(BBP)

By

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

Deliverable:	ITD
Title:	Acceptance Testing Document for Best Bike Paths (BBP)
Authors:	Jayasurya Marasani, Arunkumar Murugesan and Sneharajalakshmi Palanisamy
Version:	1.0
Date:	07 February 2026
Download page:	https://github.com/BBPpolimi/MurugesanPalanisamyMarasani
Copyright:	Copyright © 2026, Jayasurya Marasani, Arunkumar Murugesan and Sneharajalakshmi Palanisamy – All rights reserved

Contents

List of Tables

1 Product Identifitication

1.1 Reviewd Team

1. Puthil Sanjay Rao - **11124419**
2. Pinto Emmanuel Neil - **11115108**
3. Jiang Jiaxun - **11021337**

1.2 Github Repo Link

Link: https://github.com/imsanjayrao/JiangPintoPuthli

1.3 Documents Used

1. RASD
2. DD
3. readme.md
4. ITD
5. Code files

2 Installation Setup and Issues

2.1 Operating System

- The prototype was installed and tested on a system running **Arch Linux**.

2.2 Environment

- The installation environment required the following dependencies:
  - Go
  - Docker
  - Docker Compose
- The above dependencies were installed using the **YAY** package manager.

2.3 Installation Process and Observations

- The project documentation does not provide explicit instructions for installing the required prerequisites on the target system.
- After setting up the required environment the **Docker-based deployment procedure** described in the repository was followed step by step.
- The deployment process completed successfully and the application started correctly without any runtime errors.
- No missing steps broken dependencies version incompatibilities or setup inconsistencies were encountered during the installation and execution of the prototype.
- The only minor limitation observed concerns the lack of guidance in the documentation regarding the installation of Go Docker and Docker Compose.

3 Acceptance Testing

3.1 Acceptance Test Cases and Outcomes

1. **Acceptance Test Case 01**
    - **Requirement Being Tested:** R1 – Register to access personalized features
    - **Preconditions:**
        - System is running
        - Evaluator has no existing account for the test email.
    - **Test Steps:**
        (a) Open the application.
        (b) Navigate to Sign Up / Register.
        (c) Enter valid registration data (username/email/password as required).
        (d) Submit the registration form.
        (e) Try logging in with the newly created credentials.
    - **Expected Result:** Account is created and user can log in successfully.
    - **Actual Result:** PASS
    - **Comments:**
        - No email verification and no reset/ forgot password.
2. **Acceptance Test Case 02**
    - **Requirement Being Tested:** R1 – Registration input validation
    - **Preconditions:**
        - System is running
    - **Test Steps:**
        (a) Go to Register.
        (b) Submit form with missing required field(s) (e.g., blank password).
        (c) Submit form with invalid email format (if email exists).
    - **Expected Result:** Registration is rejected; user receives a clear validation error message.
    - **Actual Result:** PASS
    - **Comments:**
        - Validation is correct.
3. **Acceptance Test Case 03**
    - **Requirement Being Tested:** R2 – Log in to account
    - **Preconditions:**
        - A valid account exists.
    - **Test Steps:**
        (a) Navigate to Login.
        (b) Enter correct credentials.
        (c) Submit.
    - **Expected Result:** User is authenticated and redirected to a logged-in area (dashboard/profile).

- **Actual Result:** PASS
- **Comments:**
  - ITD: login partially implemented (no password reset).

4. **Acceptance Test Case 04**

   - **Requirement Being Tested:** R2 – Login failure on wrong credentials
   - **Preconditions:**
     - A valid account exists.
   - **Test Steps:**
     (a) Navigate to Login.
     (b) Enter correct username/email but wrong password.
     (c) Submit.
   - **Expected Result:** Login fails with a clear error; user stays unauthenticated.
   - **Actual Result:** PASS
   - **Comments:**
     - No sensitive information was revealed.
     - Display of message “Invalid credentials”.

5. **Acceptance Test Case 05**

   - **Requirement Being Tested:** R3 – Record trip in Manual Mode
   - **Preconditions:**
     - User is logged in.
   - **Test Steps:**
     (a) Navigate to Create Trip (Manual mode).
     (b) Enter trip metadata (name/date/time/duration as required).
     (c) Enter street/path info and select a status for the path/segments.
     (d) Save trip.
   - **Expected Result:** Trip is saved; user can view it in their trip history with entered data.
   - **Actual Result:** PASS
   - **Comments:**
     - Unable to edit the entered points/ locations.
     - Destination limits in the dropdown for “End Point”.
     - The dropdown has only 3-4 places listed for "End Point".
     - The start and end points are predetermined. No choice is given for users to search for places.
     - No option to edit/delete the entered path.
     - Unable to edit the manually saved path.

6. **Acceptance Test Case 06**

   - **Requirement Being Tested:** R3 – Manual trip required fields & constraints
   - **Preconditions:**

  - User is logged in.
- **Test Steps:**
  (a) Open Manual trip creation.
  (b) Leave mandatory fields empty and attempt save.
  (c) Enter invalid values (e.g., negative duration) if allowed by UI.
  (d) Attempt save.
- **Expected Result:** Trip is not saved; UI shows validation messages.
- **Actual Result:** PASS
- **Comments:**
  - Multiple paths with the same name are accepted.
  - Users unable to distinguish paths.

7. **Acceptance Test Case 07**

- **Requirement Being Tested:** R7 – Stored trips appear in trip history
- **Preconditions:**
  - User has atleast one saved trip.
- **Test Steps:**
  (a) Go to My Trips / Trip History.
  (b) Locate the created trip.
  (c) Open trip details page.
- **Expected Result:** Trip list contains the trip; details page shows correct metadata and path info.
- **Actual Result:** PASS
- **Comments:**
  - Trip persistence and retrieval is verified for manual paths.

8. **Acceptance Test Case 08**

- **Requirement Being Tested:** R8 – Trip statistics are displayed
- **Preconditions:**
  - At least one trip exists.
- **Test Steps:**
  (a) Open trip details.
  (b) Locate statistics section (distance, duration, avg speed, etc.).
- **Expected Result:** Stats are displayed and consistent with trip data (at minimum: non-empty, plausible values).
- **Actual Result:** PASS
- **Comments:**
  - The displayed stats include: avg speed, duration, weather, wind speed, distance travelled and temperature.

9. **Acceptance Test Case 09**

- **Requirement Being Tested:** R9 – Weather attached to trip (current or historical)

- **Preconditions:**
  - User is logged in
  - System has internet access if weather API is live
  - A trip exists or will be created.
- **Test Steps:**
  (a) Create a trip with a date = today.
  (b) Save and open trip details.
  (c) Verify weather fields exist (temperature, wind, etc.).
  (d) Create another trip with a past date (if UI permits).
  (e) Verify weather is still attached (historical behavior).
- **Expected Result:** Weather info is shown on trip details and matches "current vs historical" logic described in ITD.
- **Actual Result:** PASS
- **Comments:**
  - Weather Information exists
  - Not able to create the manual path with the current time. Validation is mentioning the time should be at least one hour before the current time.

10. **Acceptance Test Case 10**

- **Requirement Being Tested:** R10 – User can define path status (manual)
- **Preconditions:**
  - User is logged in
- **Test Steps:**
  (a) Create a manual trip.
  (b) For at least one path/segment, set status (optimal/medium/sufficient/maintenance or equivalents).
  (c) Save trip and reopen details.
- **Expected Result:** Status is saved and visible; values match allowed set.
- **Actual Result:** PASS
- **Comments:**
  - The selected path statuses for each segment determine the status for the entire trip.
  - No option to delete or edit the created path or path statuses again.

11. **Acceptance Test Case 11**

- **Requirement Being Tested:** R4 – Record trip in Automated Mode (availability + flow)
- **Preconditions:**
  - System is running
  - User is logged in
- **Test Steps:**
  (a) Navigate to trip creation and select Automated Mode.
  (b) Start automated recording (or simulated recording).
  (c) Stop recording.
  (d) Proceed to review/summary step.

- **Expected Result:** Automated mode exists and produces a trip draft for review (even if simulated).
- **Actual Result:** PASS
- **Comments:**
  - The automated trip recording starts with a predetermined location and the list goes on until the trip is ended.

12. **Acceptance Test Case 12**

- **Requirement Being Tested:** R11 – User must confirm/correct/reject auto-acquired data before storing
- **Preconditions:**
  - Automated trip draft exists (AT-11).
- **Test Steps:**
  (a) Reach the review/confirmation page for an automated trip.
  (b) Attempt to save without confirmation.
  (c) Perform an explicit confirm action.
  (d) Save trip.
- **Expected Result:** System enforces review; trip is stored only after confirm/correction/rejection step.
- **Actual Result:** PASS
- **Comments:**
  - The Confirmation is not given.
  - Automated trip recording has Add Path which is not needed.

13. **Acceptance Test Case 13**

- **Requirement Being Tested:**R5 – Automated mode path reconstruction output is shown
- **Preconditions:**
  - Automated trip draft exists.
- **Test Steps:**
  (a) On the review page, locate the reconstructed path (map/segment list).
  (b) Check that origin→destination segments exist (or at least a generated route).
  (c) Save and reopen trip details.
- **Expected Result:** A reconstructed path representation is shown and persisted with the trip.
- **Actual Result:** PASS
- **Comments:**
  - The recorded path is visible

14. **Acceptance Test Case 14**

- **Requirement Being Tested:** R6 – Automated detection of inconsistencies/obstacles is presented
- **Preconditions:**
  - Automated trip draft exists.

- **Test Steps:**
  (a) On automated review, locate detected issues/statuses (e.g., anomalies).
  (b) Confirm there is at least one segment with a non-default status if simulation provides it.
  (c) Save trips and verify issues remain visible.
- **Expected Result:** Detected inconsistencies are displayed to the user and carried into the stored trip after confirmation.
- **Actual Result:** FAIL
- **Comments:**
  – After the Automated Recording draft, there is an option to change the statuses.
  – No option for obstacle detection like potholes, construction etc.

15. **Acceptance Test Case 15**

- **Requirement Being Tested:** R12 – User can publish trip/path info to community
- **Preconditions:**
  – User is logged in
  – Atleast one trip exists
- **Test Steps:**
  (a) Open trip details.
  (b) Select option to Publish (or mark publishable).
  (c) Confirm action.
  (d) Navigate to the community/public section and search for the published content.
- **Expected Result:** Trip/path info becomes visible in the "Community" view and is marked published.
- **Actual Result:** PASS
- **Comments:**
  – The paths get published to the community when the user publishes the recorded paths.

16. **Acceptance Test Case 16**

- **Requirement Being Tested:** R13 – Guest can search paths (no login)
- **Preconditions:**
  – User is not logged in / guest access
  – System is running
- **Test Steps:**
  (a) open application as guest.
  (b) Navigate to path search.
  (c) Choose origin and destination (dropdowns).
  (d) Run search.
- **Expected Result:** Guest users can perform a path search and receive results.
- **Actual Result:** PASS
- **Comments:**
  – But the provided path doesn't show the status of the entire route nor others details.

17. **Acceptance Test Case 17**

- **Requirement Being Tested:** R14 – Map visualization of relevant paths
- **Preconditions:**
  - Path search results exist.
- **Test Steps:**
  (a) Perform path search.
  (b) Locate map visualization (canvas/map component).
  (c) Select/highlight a returned path.
  (d) Use "Open in Google Maps" (if present).
- **Expected Result:** Paths are visualized on map; selecting a path highlights it; external map link opens correct route.
- **Actual Result:** PASS
- **Comments:**
  - The visualization is a graph of the predetermined points.
  - The button "Open in Google Maps" redirects to the map visualization of the entire route on Google Maps.

18. **Acceptance Test Case 18**

- **Requirement Being Tested:** R13 – Registered user can search paths
- **Preconditions:**
  - User is logged in
- **Test Steps:**
  (a) Navigate to path search.
  (b) Choose origin + destination.
  (c) Run search.
- **Expected Result:** Path search works for logged-in users as well; results displayed.
- **Actual Result:** PASS
- **Comments:**
  - Once the origin is selected, the same point is visible in the destination.

19. **Acceptance Test Case 19**

- **Requirement Being Tested:** R15 – Path score ranking is applied to results
- **Preconditions:**
  - There are multiple candidate paths/segments with different statuses (from published reports).
- **Test Steps:**
  (a) Ensure at least two published trips report different statuses for different segments/paths.
  (b) Run path search for a route that includes those segments.
  (c) Observe ordering / ranking indicators.
- **Expected Result:** Results are ranked with higher-quality statuses preferred (optimal > medium > sufficient > maintenance), as described in ITD.
- **Actual Result:** PASS
- **Comments:**

  - No score is displayed
  - The overall rank of that path is mentioned as status.
  - Paths are Not ordered based on Ranking.

20. **Acceptance Test Case 20**

- **Requirement Being Tested:** R16 – Multi-user merge: majority wins
- **Preconditions:**
  - Two (or more) different users exist
  - Publishing is available
  - Both can report on the same segment/path.
- **Test Steps:**
  (a) User A creates a trip on segment S and sets status = optimal; publish.
  (b) User B creates a trip on the same segment S and sets status = maintenance; publish.
  (c) User C (or A/B) creates a third published report on the same segment S = optimal.
  (d) Run a search that includes segment S or view aggregated segment status.
- **Expected Result:** Aggregated/merged status for S becomes "optimal" (majority).
- **Actual Result:** Not Executed
- **Comments:**
  - Not able to verify the merge whether it is working or not.

21. **Acceptance Test Case 21**

- **Requirement Being Tested:** R16 – Merge tie-breaker uses most recent timestamp
- **Preconditions:**
  - Two users are needed.
  - Equal number of reports for two different statuses on same segment/path.
- **Test Steps:**
  (a) User A publishes status for segment S = medium.
  (b) User B publishes status for segment S = sufficient (same count now).
  (c) Publish a newer report to break tie (e.g., User B publishes again same segment S = sufficient later) OR follow the system's mechanism to update.
  (d) Observe merged outcome for S.
- **Expected Result:** When counts tie, the system selects the most recent report's status.
- **Actual Result:** PASS
- **Comments:**
  - It counts ties, but displaying both the paths and not able to verify which is considered.

22. **Acceptance Test Case 22**

- **Requirement Being Tested:** R7/R12 – Community/public view only shows published items
- **Preconditions:**
  - Logged out or guest
  - Atleast one published trip exists and one non-published trip exists.
- **Test Steps:**

(a) As a logged-in user, ensure Trip A is published and Trip B is not.
(b) Log out and go to the community/public section.
(c) Search/browse public trips list.

- **Expected Result:** Published Trip A is visible; non-published Trip B is not visible to guests.
- **Actual Result:** PASS
- **Comments:**
  - The Community tab only holds the trips published by the users.

23. **Acceptance Test Case 23**

- **Requirement Being Tested:**R3/R7 – User can delete or remove a trip (if UI supports it)
- **Preconditions:**
  - User is logged in
  - Atleast one trip exists.
- **Test Steps:**
  (a) Open trip list.
  (b) Delete a trip (if feature exists).
  (c) Refresh trip list.
- **Expected Result:** Trip is removed from trip history and no longer accessible.
- **Actual Result:** PASS
- **Comments:**
  - The users are able to delete/ unpublish trips when needed.
  - But the manual path recording has no provision to remove segments.

24. **Acceptance Test Case 24**

- **Requirement Being Tested:** R13/R14 – Handling invalid search input (negative acceptance test)
- **Preconditions:**
  - System running; any user (guest or logged-in).
- **Test Steps:**
  (a) Navigate to path search.
  (b) Try searching with missing origin or destination.
  (c) If possible, select the same origin and destination and search.
- **Expected Result:** System prevents invalid search or returns a clear error message; no crash.
- **Actual Result:** PASS
- **Comments:**
  - The origin and the destination are predetermined and the user has no provision to enter points of his/ her choice.
  - But when the result has no origin or destination selected, “Please select both origin and destination” message is displayed.

4 Quality Observations

4.1 Code Readability

- Clear package separation: `handlers/`, `models/`, `database/`, `weather/`, `static/`
- Descriptive naming for functions and variables with self explanatory intent
- Consistent use of Go idioms such as explicit `err` handling concise handlers JSON tags and DTO usage
- Authentication utilities are clean and readable including `generateToken`, `hashPassword` and `checkPassword`
- Useful inline comments for key business logic and requirement traceability for example `// (R16)`
- SQL schema is well commented and easy to understand
- Inconsistent error handling as some errors are logged with fallback behavior while others are silently ignored
- Long and complex functions in `handlers/trips.go` reduce clarity due to mixed responsibilities
- Repeated SQL NULL parsing patterns introduce duplication and reduce maintainability
- Magic numbers and hardcoded constants such as session duration and bcrypt cost are present without centralized configuration
- Minor robustness and security related readability issues such as raw SQL `LIMIT` concatenation

4.2 Project Structure

- Clean repository level organization with separate documentation directories such as `RASD/`, `DD/` and `Implementation/`
- Implementation structure is conventional and easy to navigate
- Entry point is simple and contained in `main.go`
- Docker and Docker Compose configuration is provided and the use of a multi stage Docker build is a strong design choice
- Database schema is embedded and managed cleanly with volume persistence properly configured
- External dependencies are minimal and appropriate
- No explicit service layer is present as business logic is embedded directly in HTTP handlers
- No repository or data access abstraction layer is implemented and SQL queries are mixed into handlers
- Middleware pipeline is limited with no centralized logging metrics or recovery mechanism
- No automated tests are provided as `*_test.go` files are missing across packages
- Potential evaluator setup friction exists as the PostgreSQL port mapping may conflict with local usage on port 5432

4.3 Documentation Clarity

- The root README provides clear setup commands and a usage entry point including the application URL
- The implementation README includes a useful endpoint list which supports black box evaluation
- Repository structure overview is clear and evaluator friendly
- A public deployment URL is documented and useful for verification
- SQL schema documentation is detailed with tables constraints and seeded locations clearly explained
- No troubleshooting section is provided for common issues such as port conflicts Docker reset or service startup timing
- Environment variable and configuration documentation is minimal for example database URL and port configuration
- No formal API documentation is provided such as OpenAPI Swagger or request and response examples
- Package level GoDoc comments are missing which reduces long term maintainability

4.4 Coherence with RASD and DD

- Strong coverage of core RASD requirements including R1, R2, R3, R7, R8, R9, R13, R14, R15 and R16
- Weather enrichment functionality aligns with RASD goal G6 and the designed external service integration
- Publishing and community related behavior aligns with aggregation and shared knowledge goals defined in G5
- Search and route visualization features align with browsing and recommendation goals defined in G4
- Database schema is coherent with the DD graph model concept including nodes edges user reports and aggregation
- Implementation maps well to the component responsibilities described in the DD including authentication trips paths routing search database and weather
- Sensor based automatic detection goals related to G3 and requirements R4 to R6 are simulated or limited compared to real device sensing
- The DD layering is conceptually present using a client server model but implemented as a monolithic backend which is acceptable for a prototype
- A minor documented deviation is present where nginx is used instead of ngrok which represents an implementation choice and does not violate requirements

5 Effort Spent

Team Member	Jayasurya Marasani	Arunkumar Murugesan	Sneharajalakshmi Palanisamy	Section Total
Document Analysis	4	4	4	12
Installation	1	1	1	3
Test Case Design	5	5	5	15
Test Case Execution	3	3	3	9
ATD Report Writing	6	4	5	15
Total	**19**	**17**	**18**	**54**

Table 1: Effort spent per member

The table as shown in Table. 1 displays the number of hours each group member spent on document analysis, installation setup, test cases designing, execution and report writing. Please note that the division is only approximate and each task still required the colloboration of all team members.

References

[1] Y. Jiang, L. Pinto, and S. Puthli. *Requirement Analysis and Specification Document (RASD)*. Version 1.00, December 23, 2025.

[2] Y. Jiang, L. Pinto, and S. Puthli. *Design Document (DD)*. Version 1.0, January 6, 2026.

[3] Y. Jiang, L. Pinto, and S. Puthli. *Implementation and Test Deliverable (ITD)*. Version 1, 2026.

[4] Y. Jiang, L. Pinto, and S. Puthli. *README.md*. BestBikePaths project repository, GitHub.

[5] Politecnico di Milano. *Software Engineering 2 – Implementation & Testing Assignment Rules*. Academic Year 2025–2026.

[6] BestBikePaths Project Repository. https://github.com/imsanjayrao/JiangPintoPuthli.

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

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

