POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering II

Acceptance Testing Document

Project: Best Bike Paths

Authors: Ianosel Bianca Roberta, Gholami Vajihe, Errigo Simone
Version: 1.0
Date: 08.02.2026
Project Link: Errigo-Gholami-Ianosel (github.com)

Contents

1 | Introduction

1.1. Tested Project

The analyzed project is the **Best Bike Paths (BBP)** System, a web-based crowdsourcing application aimed at supporting cyclists in discovering safer and more comfortable routes, developed by **Huang Shijie, Li Yuqing, and Ma Zeyao**(github.com).
The delivered material includes both the frontend and backend components of the system, as well as a deployable JAR package for the backend.
For the purposes of this Acceptance Test Document, the following reference documents provided by the authors were considered:

- **RASD**: Requirements Analysis and Specification Document
- **DD**: Design Document
- **ITD**: Implementation and Test Document

1.2. Definitions, Acronyms

This section provides definitions and explanations of the terms and acronyms used through out the document, making it easier for readers to understand and reference them.

1.2.1. Definitions

- **Path:** A route created by users (manual drawing or GPS-based creation). A path consists of one or more Path Segments.
- **Path Segment:** A portion of a Path defined by its polyline geometry and linked to adjacent segments.
- **Path Status:** The overall condition of a Path or Path Segment, computed from user reports.
- **Path Score/Ranking:** The value used to order suggested paths, derived from status and distance when searching routes.

- **Trip:** A cycling activity tracked through the BBP application. If the user is logged in, the trip is stored and becomes part of the trip history, including temporal and spatial data (e.g., duration, distance, route).
- **Report:** A submission of path information by a logged-in user. Reports describe obstacles and path condition for a specific segment.
- **Freshness:** A metric used when aggregating reports; newer reports carry more weight than older ones when determining Path Status.
- **Obstacle:** An element on a path that negatively impacts cycling conditions, such as potholes or flooding, as identified by users.
- **Manual Creation Mode:** The creation mode in which a logged-in user defines a new path by drawing it on the map.
- **Automatic Creation Mode:** The creation mode in which a logged-in user defines a new path by cycling along it, allowing the system to reconstruct the path using GPS data.
- **Manual Report:** The functionality where a logged-in user manually creates a report by selecting the path condition and obstacle through the application interface.

1.2.2. Acronyms

- **BBP:** Best Bike Paths.
- **API:** Application Programming Interface.
- **CLI:** Command Line Interface.
- **CRUD:** Create, Read, Update, Delete.
- **DBMS:** Database Management System.
- **DD:** Design Document.
- **GPS:** Global Positioning System.
- **HTTP:** HyperText Transfer Protocol.
- **ITD:** Implementation and Test Document.
- **JSON:** JavaScript Object Notation.

- **JWT:** JSON Web Token.
- **ORM:** Object-Relational Mapping.
- **OSRM:** Open Source Routing Machine.
- **RASD:** Requirements Analysis and Specification Document.
- **REST:** Representational State Transfer.
- **SDK:** Software Development Kit.
- **TLS:** Transport Layer Security.
- **UI:** User Interface.
- **URL:** Uniform Resource Locator.
- **UX:** User Experience.

1.3. Revision History

- Version 1.0 (08 February 2026);

1.4. Document Structure

This document is divided into six chapters, which are organized as follows:

1. **Introduction:** provides an overview of the document, including the project under evaluation, the goals of the analysis, and how the document is organized.
2. **Installation:** describes the installation process, including the steps followed to install and run the system, as well as any issues encountered.
3. **Testing:** details the acceptance test cases executed on the system and their outcomes.
4. **Conclusions:** summarizes the results of the installation and testing phases, highlighting significant findings or issues.
5. **References:** lists the references and resources used in the creation of the document and the project.
6. **Effort Spent:** details the distribution of work and time spent by each team member throughout the project.

2 | Installation

The BBP platform can be used in different configurations, depending on the intended use case and on whether the user aims to simply run the application or to actively develop and test it.

2.1. Prerequisites

Here we should say if they gave us proper prerequisites in the ITD, and ehat they were

2.2. Backend Installation

Here we should say how the installation proceded, if everything was explained okay, and what steps we followed.

2.3. Frontend Installation

Here we should say how the installation proceded, if everything was explained okay, and what steps we followed.

2.4. Issues Encountered

Any issue encountered during the process

3 | Testing

Explain what we tested and how.

3.1. Signup and Login

3.1.1. Steps

How we tested it, step by step.

3.1.2. Expected Result

What we expected to happen.

3.1.3. Actual Result

What actually happened.

3.1.4. Edge Cases

Edge Cases (duplicate email, wrong password, etc.)

3.2. Path Creation

3.3. Reporting Obstacles

3.4. Path Searching

3.5. Other Functionalities

4 | Conclusions

How smooth everything was, the errros, if it is a good product, things like this.

5 | References

5.1. Reference Documents

The preparation of this document was supported by the following reference materials:

- Assignment specification for the ITD of the Software Engineering II course, held by professors Matteo Rossi, Elisabetta Di Nitto, and Matteo Camilli at the Politecnico di Milano, Academic Year 2025/2026 [4];
- Slides of the Software Engineering II course available on WeBeep [5].

5.2. Software Used

The following software tools have been used to support the development of this project:

- **Visual Studio Code**: editing of source code and documentation (LaTeX), with project-wide search and formatting support [3].
- **LaTeX**: typesetting system used to produce the final RASD document in a consistent format [2].
- **Git**: version control used to track changes and support collaborative development [6].
- **GitHub**: remote repository hosting and collaboration platform used for versioning, reviews, and issue tracking [1].

5.3. Use of AI Tools

AI tools were used during the project as supporting tools, in the same way as other software adopted in the development process. Their role was not to automatically generate content, but to help improve the clarity, structure, and overall quality of the documentation.

Their use was mainly limited to the writing and revision phases. They were helpful in rephrasing sentences, simplifying long or unclear passages, and checking whether explanations could be misunderstood. In some cases, interacting with an AI assistant also helped clarify ideas before writing the final version of the text.

5.3.1. Tools Used

The AI tools employed during the project were:

- Gemini
- ChatGPT
- Copilot

5.3.2. Typical Prompts

AI tools were queried using prompts such as:

- "Rephrase this description to make it clearer."
- "Does this explanation of the backend deployment process contain any ambiguities?"
- "Suggest alternative wording for this technical paragraph to improve flow."
- "Identify any terms in this section that may be inconsistent with the definitions provided earlier in the document."
- "Format this design description or table using LaTeX".
- "What's the difference between interface and type in TypeScript, and in what specific scenarios is one recommended over the other? How do you achieve type safety?"
- "What is the config.ts file and how do you configure it in the best way?"
- "What's the ideal structure for organizing a Node.js/Express project to clearly separate routes, controllers, and business logic?"
- "What are the best practices for structuring a React Native project?"
- "Explain how mocking works in Jest. What are the best practices?"
- "How does a Dockerfile work, and what are the best practices for writing one for a Node.js application?"

- "How does a Docker Compose file work, and what are the best practices to set up Docker Compose for a multi-service Node.js application?"
- "How to monitor logs of Docker containers effectively?"
- "Is Redis persistent, or is data lost on reboot?"
- "What's the difference between Redis and Memcached?"
- "How do TTL, EXPIRE and setex work?"
- "How should cache invalidation be handled correctly?"
- "How should I test the Redis cache service with Jest?"
- "How can I see if Redis is working correctly and how can I access the service with redis-cli?"

5.3.3. Input Provided

The input given to AI tools consisted mainly of:

- Early drafts of paragraphs.
- Short text fragments requiring clarity checks.
- Sections with repeated structure where consistent wording was needed.
- Technical descriptions needing LaTeX formatting.
- Code snippets or configuration files requiring explanations or best practices.

5.3.4. Constraints Applied

When using AI tools, the following constraints were strictly enforced:

- Preserve the intended meaning of the original text.
- Avoid introducing new design decisions or assumptions.
- Maintain terminology aligned with the definitions provided in this document.

5.3.5. Outputs Obtained

The interaction with AI tools resulted in:

- Clearer or more concise formulations of existing statements.
- Identification of potentially ambiguous sentences.
- Terminology suggestions to improve internal coherence.
- LaTeX formatting assistance for tables and code snippets.
- Explanations of technical concepts and best practices.

5.3.6. Refinement Process

All AI-generated outputs were subject to a manual refinement process that included:

- Critical review of all suggestions.
- Verification against the original intent to avoid unintended changes.
- Manual integration to ensure consistency with the overall writing style.
- Alignment checks with established terminology and definitions.

6 | Effort Spent

This section provides a breakdown of the number of hours each group member dedicated to completing this document. The work distribution is tracked per section and task.

Section	Ianosel Bianca	Simone Errigo	Vajihe Gholami
Documents Inspection	2 hours	2 hours	2 hours
Installation Procedure	1 hours	1 hours	1 hours
Test Case Execution	2 hours	2 hours	2 hours
Document Preparation	2 hours	2 hours	2 hours

Table 6.1: Time spent

Bibliography

[1] GitHub Inc. Github. Online platform, 2025. https://github.com/.

[2] LaTeX Project Team. Latex: A document preparation system. Document preparation system, 2025. https://www.latex-project.org/.

[3] Microsoft. Visual studio code. Source code editor, 2025. https://code.visualstudio.com/.

[4] M. Rossi, E. Di Nitto, and M. Camilli. Software engineering 2 itd assignment specification, Academic Year 2025/2026.

[5] M. Rossi, E. Di Nitto, and M. Camilli. Slides of the software engineering 2 course. WeBeep platform, Academic Year 2025/2026.

[6] Software Freedom Conservancy. Git. Version control system, 2025. https://git-scm.com/.

List of Tables