POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Software Engineering II

Implementation Document

Project: Best Bike Paths

Authors: Ianosel Bianca Roberta, Gholami Vajihe, Errigo Simone
Version: 1.0
Date: 01.02.2026
Project Link: Errigo-Gholami-Ianosel (github.com)

Contents

1 | Introduction

1.1. Purpose

1.2. Scope

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

1.3.2. Acronyms

- **API:** Application Programming Interface.
- **APK:** Android Package
- **DBMS:** DataBase Management System.
- **DD:** Design Document.
- **DOM:** Document Object Model.
- **DTO:** Data Transfer Object, represents a link between the user input and a Java Object.
- **HTTP:** HyperText Transfer Protocol.
- **IPA:** iOS App Store Package.
- **JPA:** Java Persistence API.
- **JS:** JavaScript.
- **QR Code:** Quick Response Code.
- **REST:** REpresentational State Transfer (see DD).
- **RASD:** Requirements Analysis and Specification Document.

- **S2B:** Software To Be.
- **UI:** User Interface.
- **URL:** Uniform Resource Locator.
- **UX:** User eXperience.
- **ORM:** Object-Relational Mapping.

1.3.3. Abbreviations

- **something:**

1.4. Revision History

- Version 1.0 (01 February 2026);

1.5. Document Structure

Mainly the current document is divided into six chapters:

1. **Introduction:** provides an overview of the document, outlining its purpose, scope, and relevance to the project.
2. **Implemented Functionalities and Requirements:** details the functionalities and requirements that have been implemented in the project.
3. **Adopted Development Frameworks:** describes the development frameworks utilized in the project, explaining their roles and benefits.
4. **Source Code Structure:** outlines the organization and structure of the source code, facilitating understanding and navigation.
5. **Testing Strategy:** presents the testing methodologies and strategies employed to ensure the quality and reliability of the software.
6. **Installation Instructions:** provides step-by-step guidance on how to install and set up the software.
7. **References:** lists the references and resources used in the creation of the document and the project.

8. **Effort Spent:** details the distribution of work and time spent by each team member throughout the project.

2 | Implemented Functionalities and Requirements

2.1. Product Functions

This section outlines the essential functionalities and detailed requirements of the platform, structured to support the key objectives defined in the scope of the product.

- User Management:
- Trip Management:
- Path Management:
- Report Management:
- Statistics Management:
- Weather Integration:

2.2. Requirements

The following requirements have been implemented
The following requirements have NOT been implemented

3 | Adopted Development Frameworks

3.1. Adopted Frameworks

3.1.1. Frontend

React Native Expo React Paper -> Theming Lucide Icons -> Icons React Native Maps -> Maps Axios -> Api Calls Zod -> Data Validation Zustand -> State Management Expo Router -> Navigation

3.1.2. Backend

NestJS Openmeteo service

3.1.3. Data Layer

PostgreSQL Prisma

3.2. Adopted Programming Languages

TypeScript

3.3. Development Tools

Docker + Shared Nginx + Cloudflare + Docker

3.4. API Calls

Any API not included in the DD should be mentioned here.

4 | Source Code Structure

4.1. Code Organization

4.1.1. Frontend

Here it is a walkthrough from the notes, should be written bette, but this is the content to include:
Directory Walkthrough (src)
Root files app.json / app.config.js Expo project name, icon, splash, extra env vars. package.json scripts + dependencies (Expo 54, React Native 0.81, Paper 5, Zustand, etc.). tsconfig.json aliases @/ to src, JSX config. jest.config.js React Native preset extended to include tests/unit + tests/integration. jest.setup.ts global mocks for Expo Router, SecureStore, SafeArea, Paper UI, AppButton/TextInput/Popup shims, console suppression. tests/mocks/expoMock.ts empty stub returned when some modules import expo.
app/ (Expo Router) _layout.tsx wraps the entire navigation tree with PaperProvider, injects the Lucide icon set, calls useAuthStore.initAuth() and enforces redirects (unauth users forced to welcome, loggedin users prevented from staying in (auth)). +notfound.tsx generic 404 screen. (auth)/_layout.tsx minimal layout for welcome/login/signup flows. (auth)/welcome.tsx hero screen with CTA buttons and "Guest Mode". (auth)/login.tsx and signup.tsx React Hook Form + Zod validation wiring, call useAuthStore actions. (main)/_layout.tsx houses BottomNav, LoginPromptProvider, and the loginrequired popup (guest flows). (main)/home.tsx central map/route experience: search inputs, result sheet, Create Path FAB, report modal, navigation progress, start/complete trip actions. (main)/trips.tsx trip history, metrics, stats cards, entry points to detail screens. (main)/paths.tsx curated path list with filters/tags. (main)/createpath.tsx wizard/modal for usergenerated paths. (main)/profile.tsx profile header, stats, privacy preferences, entry to editprofile. (main)/editprofile.tsx full edit form with Zod validation, updateProfile call, success/error popups. (main)/settings.tsx settings toggles, theme/preferences. Folder Structure Explained
app/
This is the main routing folder Expo Router turns its structure into navigation automat-

ically.
Inside it:
(tabs)/ > Defines the bottom tab navigation (the bar with multiple pages). _layout.tsx > Defines the navigation bar and icons for each tab. index.tsx > Home screen. two.tsx > Example second screen. (Additional screens can be added here, e.g., trip.tsx, paths.tsx, etc.)
_layout.tsx > Root layout for everything outside tabs (e.g., modals, login screens). **+html.tsx** > Used internally when running on web to render HTML pages (can be ignored). **+notfound.tsx** > The "404" page, shown if a user navigates to a nonexistent route. **modal.tsx** > A demo modal screen (can be removed or replaced later).
assets/
Contains static files bundled with the app:
images/ > App icons, splash screens, etc. fonts/ > Custom fonts (the default project includes SpaceMonoRegular.ttf).
Additional assets such as logo.png or mapmarker.png can be added here and imported in the screens.
components/ ui/ reusable primitives (AppButton, AppTextInput, AppPopup, SelectField, etc.). BottomNav.tsx bottom navigation bar aware of auth/guest state. ScreenHeader, ProfileHeroHeader, PathResultCard, SearchResultsSheet, CreatePathModal, ReportIssueModal, RouteMap, RouteCard, etc. These encapsulate domain UI/logic for routes, paths, profile. components/icons/ LucideIcon wrapper to ensure consistent icon use inside Paper theme.
auth/ authSession.ts inmemory tokens + listener for session changes. storage.ts Zustand store connecting SecureStore, APIs, and UI (init, login, logout, fetch profile, update profile, guest mode).
validation/ auth.ts Zod schemas + types for login, signup, and editprofile flows. path.ts schema for createpath modal. index.ts barrel so consumers can import signupSchema from "@/validation".
api/ client.ts Axios instance with getAccessToken() injection and 401 interceptor that hits refreshAccessToken. auth.ts login/signup/logout/profile/update API wrappers with typesafe mappers. tokenManager.ts manual axios client to refresh tokens, handles concurrency and session clearing on failure.
hooks/ useColorScheme, useThemePreference theme detection/persistence. useLoginPrompt context to trigger global login modal (guest restrictions). useBottomNavVisibility context provider to hide the nav for certain screens (e.g., edit profile). useTrips handles trip fetching/state (caching between navigations). usePrivacyPreference manages selected privacy option and persistence. useBottomNavVisibility context consumed

by modals/screens to hide nav on scrollintensive views.
constants/ and theme/ constants/Colors.ts light/dark palette, accent colors, guest colors, etc. constants/privacy.ts static privacy preference options. theme/layout.ts / theme/typography.ts spacing helpers, radius, font sizes. theme/mapStyles.ts, theme/paperTheme.ts map style JSON and Paper theme definitions.
assets/ assets/images/ icons, splash, adaptive icon. assets/fonts/ custom typefaces (Space Mono).
utils/ utils/apiError.ts normalizes Axios/JS errors into userfacing strings (used by Login/Signup/Edit Profile popups). utils/layout.ts responsive layout helpers (scale, verticalScale, moderateScale). tests/utils/render.tsx (in tests/utils) test helper that wraps React components with PaperProvider + theme icon settings.
tests/ tests/unit permodule jest files. tests/integration screen + navigation flow tests (auth, navigation). tests/utils/render.tsx helper to render components with PaperProvider wrapper in tests. tests/mocks Expo Router / Expo stub modules.

4.1.2. Backend

bash src/ – middleware - Middlewares | -jwt.auth.ts - JWT authentication middleware | -http.logger.ts - HTTP request logging middleware | -... - Other middlewares – errors/ - Custom error classes | -app.errors.ts - Applicationspecific errors – prisma/ - Prisma setup | -schema.prisma - Prisma schema file | -migrations/ - Prisma migrations | -json.types.d.ts - Prisma JSON type definitions with our custom types – services/ - External service integrations | -openmeteo.service.ts - OpenMeteo API integration – managers/ - Business logic | -user/... - Userrelated logic | -query/... - Queryrelated logic | -.../... - Other related logic – routes/ - API route definitions | -v1/ - Version 1 of the API | -auth.routes.ts - Authentication routes | -user.routes.ts - User routes | -... - Other routes | -index.ts - Central router for v1 – types/ - TypeScript type definitions | -error.types.ts - error types/interfaces | -... - Other type definitions – utils/ - Utility functions | -prismaclient.ts - Prisma client instance | -... - Other utility functions – tests/ - Test files | -auth.test.ts - Authentication tests | -user.test.ts - User tests | -... - Other tests – .env - Environment variables – tsconfig.json - TypeScript configuration – jest.config.mjs - Jest configuration – prisma.config.ts - Prisma configuration – package.json - Package configuration – package.lock.json - Package lock file -server.ts - Server entry point

4.1.3. Server

bash /opt – nginx | – conf.d | | – site.conf | | -api.conf | – ssl | | – residenzaclasmarinaorigin.crt | | – residenzaclasmarinaorigin.key | | – bia3iaorigin.crt | | -bia3iaorigin.key | -log |

– access.log | -error.log | – residenzaclasmarina | -dockercompose.yml | – bbpbackend | – Dockerfile | – dockercompose.yml | -.env

5 | Testing Strategy

5.1. Unit Testing

5.2. Integration Testing

In the notes, both of APP and BACKEND there is explanation.

6 Installation Instructions

6.1. Prerequisites

Node npm

6.2. Backend Setup

Is running on a Personal Server managed via Docker. If you want to run it locally, you need to have PostgreSQL installed and running. Then, clone the repository, install the dependencies with npm install, set up the .env file with the necessary environment variables, and run the migrations with Prisma.

6.3. Frontend Setup

You can build the app, run it on a simulator or physical device using Expo CLI. You can install the apk on Android devices directly.
If you want to use a local server you should change the API URL in the .env file and build it or run it on the emulator, since the built app points to the production server.

7 References

7.1. Reference Documents

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

- IEEE Standard for Software Requirement Specifications [2];
- 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 [6];
- Slides of the Software Engineering II course available on WeBeep [7].

7.2. 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 [5].
- **LaTeX**: typesetting system used to produce the final RASD document in a consistent format [3].
- **Git**: version control used to track changes and support collaborative development [8].
- **GitHub**: remote repository hosting and collaboration platform used for versioning, reviews, and issue tracking [1].
- **Lucidchart**: creation of UML diagrams (use case diagrams, state diagrams, domain class diagram) [4].

7.3. Use of AI Tools

AI tools were used during the project in the same way as other supporting software tools. Their role was not to autonomously generate content, but to assist in improving the presentation of the document, supporting the organisation of ideas and enhancing overall textual coherence.
Their use was mainly limited to the drafting phase, where they helped compare different ways of explaining scenarios, simplify long paragraphs, and check whether certain sentences could be misunderstood. In several cases, interacting with an AI assistant helped clarify the underlying concepts before writing the final version of the text.

7.3.1. Tools Used

The AI tools employed during the project were:

- Gemini
- ChatGPT

7.3.2. Typical Prompts

AI tools were queried using prompts such as:

- "Rephrase this design description to make the interaction flow clearer."
- "Does this explanation of the component interaction sound ambiguous?"
- "Help restructure this paragraph describing a UI flow to improve readability."
- "Format this design description or table using LaTeX"
- "Help debug formatting or build issues related to VS Code or LaTeX"

7.3.3. 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.

7.3.4. 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.

7.3.5. 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.

7.3.6. 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.

8 | 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	Total Hours
Introduction	4 hours	4 hours	5 hours	13 hours
Overall Description	11 hours	7 hours	10 hours	28 hours
Specific Requirements	19 hours	8 hours	12 hours	39 hours
Formal Analysis	7 hours	21 hours	11 hours	39 hours
Final Review & Editing	3 hours	3 hours	3 hours	9 hours
Total Hours	**44 hours**	**43 hours**	**41 hours**	**128 hours**

Table 8.1: Time spent on document preparation

Bibliography

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

[2] ISO/IEC/IEEE. Systems and software engineering - life cycle processes - requirements engineering, 2018.

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

[4] Lucid Software Inc. Lucidchart: Diagramming and visualization tool. Online platform, 2025. https://www.lucidchart.com/.

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

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

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

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

List of Figures

List of Tables