Software Requirements Specification

for

Smart Travel Companion

Version 1.0 approved

Prepared by

Mounika, Ruthvika, Hemanth

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 2

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 5

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 6

3.3 Software Interfaces 6

3.4 Communications Interfaces 7

4. System Features 7

4.1 Package List Generator 7

4.2 Smart Lock PhraseBook 8

4.3 Mini Trip Planner 8

4.4 TimeZone Converter 8

4.5 Quick Landmark Overview 8

4.6 Dark Mode & Language Support(i18n) 8

4.7 Admin Features 8

5. Other Nonfunctional Requirements **9**

5.1 Performance Requirements 9

5.2 Safety Requirements 9

5.3 Security Requirements 10

5.4 Software Quality Attributes 10

5.5 Business Rules 11

6. Other Requirements **11**

6.1 Database Requirements 11

6.2 Internationalization Requirements 11

6.3 Legal & Compliance Requirements 12

6.4 Reuse Objectives 12

6.5 Future Enhancements (TBD) 12

Appendix A: Glossary **12**

Appendix B: Analysis Models **13**

Appendix C: To Be Determined List **13**

Revision History

Name	Date	Reason For Changes	Version
			1.0

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the **Smart City Companion** platform. It serves as an intelligent assistant that helps citizens and travellers with real-time transport updates, smart navigation, local phrasebooks, and city event notifications. The system aims to improve urban mobility, enhance safety, and provide personalized recommendations. This SRS outlines the functional and non-functional requirements, ensuring smooth implementation and usability for diverse users including residents, administrators, and tourists.

1.2 Document Conventions

This document follows the IEEE standard template for Software Requirement Specifications. Headings and subheadings are numbered for clarity. Functional requirements are denoted as **REQ_xx**, while non-functional requirements are listed under respective categories. Simple, clear language is used to ensure readability for both technical and non-technical stakeholders.

1.3 Intended Audience and Reading Suggestions

The intended audience includes **citizens, travellers, city administrators, developers, and testers** who will use or manage the Smart City Companion platform. Citizens and travellers will focus on features like navigation, transport updates, and local phrasebooks. Administrators will use the system for monitoring and decision-making. Developers and testers will refer to functional and non-functional requirements for implementation and validation. Readers may begin with the **Introduction** for context, then proceed to **System Features** and **Requirements** based on their role.

1.4 Product Scope

The **Smart City Companion** is a digital platform that enhances urban living by integrating services like **real-time transport updates, smart navigation, local language phrasebook, city event notifications, and emergency assistance**. It supports both citizens and travellers by offering personalized recommendations and seamless access to city resources. The platform contributes to **smarter mobility, improved safety, and better city engagement**. By leveraging modern technologies, it aligns with smart city goals of **efficiency, sustainability, and user convenience**, ensuring a connected and intelligent urban experience.

1.5 References

1. Government of India – Smart Cities Mission: https://smartcities.gov.in

2. Google Maps Platform Real-TimeNavigation: https://developers.google.com/maps

3. World Bank – Smart City Framework Report, 2022

4. City of Barcelona – Smart City Strategy Documentation

5. IEEE Standard 830-1998: Recommended Practice for Software Requirements Specifications

2. Overall Description

2.1 Product Perspective

The **Smart City Companion** is part of a larger smart city ecosystem that integrates transport, communication, and public services. It acts as a **mobile and web-based assistant**, connecting users to real-time data such as traffic updates, transport schedules, emergency alerts, and city events. The platform interacts with **external systems** like GPS, public transport APIs, language databases, and government service portals. It is designed to be scalable, interoperable, and adaptable, making it a key component of smart urban infrastructure.

2.2 Product Functions

The system will provide the following key functions:

1. Allow users to **navigate the city** with real-time traffic and transport updates.
2. Provide a **local phrasebook** with translations and pronunciations for travellers.
3. Notify users about **city events, alerts, and emergencies**.
4. Enable **personalized itinerary planning** based on user preferences.
5. Offer **safety assistance features** such as nearby hospitals, police stations, and emergency contacts.
6. Support **multi-language accessibility** for diverse citizens and tourists.

2.3 User Classes and Characteristics

There will be three main classes of users for the Smart City Companion system:

1. **Citizens** – Regular residents of the city who use the app for daily commuting, receiving alerts, and accessing local services. They require simple, intuitive interfaces and reliable real-time data.
2. **Travelers/Tourists** – Visitors who rely on the system for navigation, translations, phrasebooks, and discovering local attractions. They may have limited technical expertise and require **multi-language support**.
3. **City Administrators/Authorities** – Officials who monitor transport data, issue city alerts, and analyse usage patterns. They require **advanced dashboards, secure access, and reporting tools**.

Each class interacts with different features, ensuring the platform serves both everyday needs and administrative oversight.

2.4 Operating Environment

PC or Laptops:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Mac OS X
- Linux

Smart TV's, Tablets (Android or IOS) etc.

2.5 Design and Implementation Constraints

1. The platform must support **Android, iOS, and web browsers** to ensure wide accessibility.
2. It should integrate with **third-party APIs** such as Google Maps, public transport databases, and emergency services.
3. The system must comply with **government data privacy and security regulations**.
4. Performance is constrained by **internet connectivity and server capacity**, especially during peak hours.
5. The design should support **multi-language content** for both citizens and international travellers.
6. The backend will use **scalable cloud infrastructure** to handle real-time updates and large user volumes.

2.6 User Documentation

The Smart City Companion will include the following user documentation:

1. **Mobile App Guide** – Step-by-step instructions for installing, registering, and using the app on Android/iOS.
2. **Web Portal Manual** – Instructions for accessing services via browsers.
3. **Quick Start Guide** – A simplified guide covering navigation, phrasebook use, and emergency assistance.
4. **Online Help & FAQs** – Built-in help section with searchable topics and troubleshooting tips.

5. **Administrator Manual** – Detailed instructions for city officials on monitoring dashboards, issuing alerts, and managing system data.

All documentation will be available in **digital format (PDF, in-app help, and website)** for easy access.

2.7 Assumptions and Dependencies

- It is assumed that users will have access to **internet-enabled smartphones or computers**.
- The system depends on **third-party APIs** (e.g., Google Maps, public transport databases, weather services) for real-time updates.
- Availability of **accurate city data** (traffic, transport schedules, emergency contacts) is assumed to be provided by city authorities.
- The application assumes **multilingual support resources** (dictionaries, translation APIs) will remain accessible.
- The system depends on **cloud infrastructure** for scalability, uptime, and real-time data synchronization.
- It is assumed that city administrators will **regularly update alerts and events** to keep the platform relevant.

3. External Interface Requirements

3.1 User Interfaces

The Smart Travel Companion will be a **responsive web application** accessible via desktop, tablet, and mobile browsers.

Traveller UI:

- **Home/Trip Dashboard:** Displays saved trips, option to create a new trip, and “Surprise Me” feature.
- **Trip View:** Contains modules for packing checklist, phrasebook, mini planner, and landmarks.
- **Checklist Screen:** Editable list with add/remove items, option to export PDF.
- **Phrasebook Screen:** Categorized phrases (greetings, emergencies, transport, dining) with translations and audio playback.

- **Planner Screen:** AI-generated itinerary with ability to adjust activities.
- **Landmark Screen:** Grid/list view of landmarks with images, descriptions, and filters.
- **Settings:** Dark/light theme toggle, language switch, reset to defaults.

Admin UI:

- **Login Page:** Secure admin login.
- **Admin Dashboard:**
  a. Manage landmarks (add/update/delete/bulk upload).
  b. Manage phrasebooks (add/edit phrases).
  c. View usage statistics.
- **Forms:** Validated forms for uploading content (text, images, CSV).

UI Design Guidelines:

- Should follow **WCAG 2.1 accessibility standards**.
- Must load within **3 seconds** under normal conditions.
- Support multiple languages (initially English & Hindi).
- Consistent navigation bar and breadcrumb trail across screens.

playback in the web browser.

3.2 Hardware Interfaces

The system is a browser-based web application and does not require special hardware beyond the client's device.

- **End-user devices:**
  - Desktops/Laptops with ≥4GB RAM.
  - Smartphones/Tablets with ≥2GB RAM.
- **Server environment:**
  - Cloud-hosted virtual machine(s) with at least:
  - 4-core CPU, 8GB RAM, 50GB storage.
  - PostgreSQL database server.
  - Backup storage for admin-uploaded content (images, CSVs).

- No proprietary external hardware is required.

3.3 Software Interfaces

- **Frontend:** React with Tailwind CSS, i18next for language support.
- **Backend:** Flask REST APIs.
- **Database:** PostgreSQL for persistent storage.
- **External APIs:**
  - OpenWeatherMap → for weather forecasts (checklist generation).
  - WorldTimeAPI → for time zone conversion.
  - Unsplash API → for landmark images.
- **File Formats Supported:**
  - Checklist export as PDF.
  - Bulk uploads via CSV/Excel for admin data entry.

3.4 Communications Interfaces

- **Protocols:** All client-server and API communication will use **HTTPS** for secure transmission.
- **Internet Requirement:** A stable internet connection is required for:
  a. Real-time weather, images, and time zone retrieval.
  b. Syncing user data and admin updates.
- **Offline Support:**
  a. Cached phrasebooks and checklists are accessible offline.
  b. Data re-synced when the internet is restored.
- **Performance Expectations:**
  a. Round-trip response time from API servers should be < 2 seconds in stable conditions.
  b. The system should handle at least 100 concurrent users during peak times.

4. System Features

4.1 Packing Checklist Generator

Priority: High
Description: Generate destination + weather-aware packing checklist.
Stimulus/Response:

1. User enters destination + travel dates.
2. The system fetches weather + city type.
3. Checklist generated + downloadable as PDF.

Functional Requirements:

- REQ_01: System shall allow travelers to create trips.
- REQ_02: System shall generate checklist within 3 seconds.
- REQ_03: System shall allow PDF export.

4.2 Smart Local Phrasebook

- REQ_04: Display phrases in the local language with translation + audio.
- REQ_05: Allow offline access to cached phrasebooks.

4.3 Mini Trip Planner

- REQ_06: Generate daily itinerary from city + dates.
- REQ_07: Suggest activities with time slots.

4.4 Time Zone Converter

- REQ_08: Show source + destination time zones.
- REQ_09: Sync with WorldTimeAPI.

4.5 Quick Landmark Overview

- REQ_10: Display top landmarks with image, description, category.
- REQ_11: Allow saving landmarks to trip notes.

4.6 Dark Mode & Language Support (i18n)

- REQ_12: Switch UI language dynamically.
- REQ_13: Toggle dark/light themes.

4.7 Admin Features

- REQ_14: Admin can add/edit landmarks.
- REQ_15: Admin can add phrases.
- REQ_16: Changes should reflect to travellers within 1 minute.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **PR-1 :** API response time (weather, time zone, image fetch) shall not exceed 3 seconds under stable 4G/WiFi conditions.
- **PR-2 :** PDF checklist generation and download shall complete within 2 seconds.
- **PR-3 :** The system shall support at least 1000 concurrent users without performance degradation.
- **PR-4 :** Trip data auto-save shall occur within 500ms after user edits to prevent data loss.

5.2 Safety Requirements

5.2.1 Data Integrity

- User data (trips, checklists, phrasebook entries) shall be auto-saved and restored after unexpected shutdowns.
- All writes must be atomic; partial updates are rolled back.

5.2.2 Graceful Failure

- On third-party API failures (weather, time zone, landmarks), the system shall **fall back to cached or default data**.
- The system shall prevent **crashes, hangs, or infinite loading loops**.

5.2.3 User Action Safety

- Destructive actions (e.g., delete trip, reset checklist) shall require **confirmation dialogs**.
- An **Undo option** shall be available for at least **5 seconds**.

5.2.4 Content & Guidance

- AI-generated checklists and itineraries shall include a **disclaimer** that suggestions must be verified with official travel advisories.
- Hazardous conditions (storms, extreme heat/cold) shall trigger **warning messages and safe recommendations**

.

5.2.5 Accessibility

- The system shall comply with **WCAG 2.1 AA standards** (contrast, keyboard navigation, screen-reader support).
- All images and icons shall have **text alternatives**.
- No flashing content above **3 Hz** shall be used.

5.2.6 Offline & Recovery

- Cached data shall be available in **read-only mode offline**.
- Sync operations after reconnection shall be **idempotent** (no duplicates).
- The system shall maintain **daily backups** (RPO ≤ 15 minutes, RTO ≤ 60 minutes).

5.3 Security Requirements

- All communication must use HTTPS with TLS 1.2+ encryption.
- User authentication shall use hashed passwords (SHA-256 with salt).
- Admin privileges must be role-restricted, with two-factor authentication (2FA) support.
- User data (trip history, saved phrases) must be encrypted at rest in the database.
- The system shall comply with GDPR-like privacy principles (data minimization, user control of stored data).

5.4 Software Quality Attributes

5.4.1 Usability:

- UI must be usable by first-time users without training.
- Accessibility: Must comply with WCAG 2.1 (AA level) for visually impaired users.

5.4.2 Reliability:

- Availability target: 99.5% uptime.
- Auto-recovery: API retries up to 3 attempts before showing fallback.

5.4.3 Maintainability:

- Code must follow modular architecture (React components, Flask services).
- Unit test coverage target: **≥80%**.

5.4.4 Scalability:

- Databases must handle **≥1 million records** (phrases, tips, landmarks) without slow queries (>200ms).

5.4.5 Portability:

- Must run seamlessly on **mobile (Android/iOS), tablet, and desktop browsers**.

5.5 Business Rules

- **BR-1**: Expired trips auto-deleted after 24h.
- **BR-6**: Landmarks categorized (cultural, historical, natural).
- **BR-18**: Phrases must follow a defined format.
- **BR-45**: Minimum three supported languages.

6. Other Requirements

6.1 Database Requirements

- PostgreSQL schema must support:
    a. User Table (profile, saved trips).
    b. Checklist Templates Table (city type, items).
    c. Phrasebook Table (phrase, translation, audio link).
    d. Landmark Table (name, category, image, description).

- DB operations must commit in <200ms for 95% of queries.

6.2 Internationalization Requirements

- All UI text must be externalized via i18next JSON files.
- Must support non-Latin scripts (e.g., Hindi,English).
- Theme and language preferences must persist across sessions.

6.3 Legal & Compliance Requirements

- **GDPR compliance:**
    a. Users must be able to **delete their account & all stored data**.
    b. The system must log **user consent** for storing trip details.

- API usage must respect **third-party provider terms** (OpenWeatherMap, Unsplash, WorldTimeAPI).

6.4 Reuse Objectives

- The **AI mini planner** module should be reusable in other university projects.
- Checklist generation logic should be **API-based** to support mobile apps in the future.

6.5 Future Enhancements (TBD)

- Integration with **local event APIs** (concerts, festivals).
- Offline-first support for **landmark browsing**.
- Push notifications for daily reminders on the mobile **PWA version**.

Appendix A: Glossary

- **AI Mini Planner** – A feature that generates basic trip itineraries automatically using predefined rules or AI models.
- **API (Application Programming Interface)** – A set of functions and protocols that allows the system to communicate with external services (e.g., OpenWeatherMap).
- **Checklist Template** – A predefined list of items associated with a city type (e.g., coastal, urban, mountain).
- **GDPR (General Data Protection Regulation)** – A European Union regulation focused on user data protection and privacy.
- **i18next** – An internationalization framework used to manage translations and multilingual support.
- **Landmark** – A point of interest such as a cultural, historical, or natural site stored in the system database.
- **PWA (Progressive Web Application)** – A type of web application that can be installed on devices and work offline with native-like features.
- **RPO (Recovery Point Objective)** – The maximum acceptable amount of data loss measured in time.
- **RTO (Recovery Time Objective)** – The maximum acceptable downtime before full system recovery.

- **WCAG (Web Content Accessibility Guidelines)** – International accessibility standards published by W3C for making web content usable by people with disabilities.

Appendix B: Analysis Models

- **Data Flow Diagram (DFD):** Illustrates the flow of information between users, external APIs, and the system database.
- **Entity-Relationship Diagram (ERD):** Represents the main entities (User, Checklist, Phrasebook, Landmark) and their relationships.
- **State-Transition Diagram:** Shows user states (logged out, logged in, editing trip, viewing checklist) and transitions triggered by actions.
- **Class Diagram (Optional):** Defines major software components such as UI modules, API services, and database interfaces.

Appendix C: To Be Determined (TBD) List

- **TBD-1:** Final decision on AI model (rule-based vs. ML-based) for the mini planner.
- **TBD-2:** Number of languages to be fully supported at first release (currently English, Hindi).
- **TBD-3:** Scope of offline mode features (currently limited to phrasebook and checklist).
- **TBD-4:** Integration timeline for push notifications in the PWA.
- **TBD-5:** Exact capacity requirements for scaling beyond 2,000 requests per second.