Amrita School of Computing

23CSE311 Software Engineering

Lab Worksheet 2:

Software Requirements Specification

Date:

CO2	Apply requirement engineering principles to analyze, model, and validate requirements for effective software solutions.

Objective:

To clearly document all functional and non-functional requirements of the system, ensuring a shared understanding among stakeholders and serving as a reference for design, development, testing, and maintenance.

Team Information:

Team Member Name	Student ID	Role in Project
Shajith	AM.SC.U4CSE23231	• UI Design • Frontend Developer • Testing
Tejaswini	AM.SC.U4CSE23219	• Backend APIs • Integration with Frontend • Documentation
Guna Abhiram	AM.SC.U4CSE23236	• Database Design • Backend Queries • API Testing
Musafirunnisa Begum Syed	AM.SC.U4CSE23264	• Geospatial Data Processing • Map Integration • Deployment

1. Introduction

1.1 Purpose

The purpose of this app is to help users identify restricted zones, access nearby emergency services, and plan safe routes between a source and destination. It also enables citizens to navigate efficiently, discover essential facilities, and receive timely alerts about disasters or critical updates.

Functionalities:

- Users can enter a source and destination to generate the optimal route.
- Turn-by-turn guidance is provided for visually impaired users.
- Citizens can see disaster-affected areas, restricted zones, and border alerts on the map.
- The app provides safe alternate paths if hazards or restricted zones are detected along the route.
- Users can search for nearby emergency services (hospitals, police stations, fire stations).
- Authorities can update restricted zones and emergency alerts.
- Admins monitor user activities and manage zone/service data.

1.2 Document Conventions

Defines symbols, abbreviations, and notations used in the document.

- UI – User Interface
- API – Application Programming Interface
- GDACS API – Global Disaster Alert and Coordination System **–** Application Programming Interface

1.3 Intended Audience and Reading Suggestions

Who will use this document?

- **Developers** – Implement features such as restricted zone mapping, location tracking, and integration with emergency services.
- **QA Testers –** Validate the accuracy of maps, alerts, and ensure smooth functioning of location-based services.
- **City Authorities & Emergency Services –** Understand how restricted zones and emergency response information are presented to the public.
- **End Users (Citizens & Travelers) –** Learn how to navigate the city, access emergency services, and stay updated about restricted areas.

1.4 Product Scope

A high-level description of the software.

- Enables users to **view restricted zones** (such as disaster-affected or high-security areas).
- Allows users to **track their current location** on an interactive map.
- Integrated with mapping and location APIs for **real-time directions and alerts**.

1.5 References

Any relevant research, standards, or regulations.

- **GDACS (Global Disaster Alert and Coordination System) API** – for real-time disaster alerts and information.
- **Google Maps JavaScript API & Places API** – for map rendering, location search, and navigation services.
- **Industry Reports on Civic Tech & Smart City Solutions –** For understanding trends in urban disaster management technologies.

2. Overall Description

This section provides a general overview of the software's functionality.

2.1 Product Perspective

Describes how the app fits into existing solutions.

- The system is a multi-user disaster management platform, integrating government agencies, NGOs, and citizens.
- It connects with third-party services like GDACS API for disaster alerts, Google Maps API for location tracking and visualization.

2.2 Product Features

Lists the app's core functionalities.

- **Location-based Notifications –** Alerts tailored to the user's current or saved location using GPS.
- **Emergency Helpline Integration –** Quick access to verified emergency contacts and hotlines.
- **Resource Finder –** Locate nearby shelters, hospitals, and relief centers.
  Feedback & Reporting – Citizens can report incidents or request help in real time.

2.3 User Classes and Characteristics

Defines the different types of users and their roles.

- Citizens / Travelers: Navigate safely using maps and voice guidance, check for disasters along the route, and receive alternate paths if needed.
- Authorities (City Admins, Disaster Management Officials, Border Security): Provide official data on restricted zones, disaster alerts, and border restrictions via APIs; the application fetches this data.
- Emergency Services (Hospitals, Fire Stations, Police, Relief Centers): Display contact numbers of nearby services; users can call directly in emergencies.
- System Administrators: Maintain the system, monitor performance, manage APIs, and ensure security and reliability of the platform.

2.4 Operating Environment

Lists system requirements.

- **Mobile App:** Android 9+ / iOS 13+
- **Web App:** Chrome, Firefox, Safari

- **Database:** MYSQL
- **Cloud Hosting:** AWS / Google Cloud

2.5 Design and Implementation Constraints

Any limitations or restrictions.

- Secure Communication – All API communications must be encrypted using SSL/TLS to ensure data security and privacy.
- Real-time Location Updates – Integration with Google Maps API must provide location updates within 3–5 seconds for accurate, timely alerts and notifications
- Cross-Platform Compatibility – The mobile and web application must be fully compatible with Android, iOS, and modern web browsers.
- Scalability Constraints – The system design must handle many concurrent users during disaster situations without significant performance degradation.

2.6 Assumptions and Dependencies

Conditions that affect development.

- Users' mobile devices and browsers must support the latest application requirements for smooth performance.
- The application depends on the continuous availability of third-party services such as **Google Maps API**, **notification services**, and **cloud hosting providers**.

3. Specific Requirements

Defines the exact system functionalities.

3.1 Functional Requirements

Lists detailed functional features.

- Location Search & Navigation
    - Users can search for places (hospitals, police stations, government offices, tourist attractions, etc.) by name, category, or nearby location.
    - Integration with Google Maps & Places API to show accurate details.
    - Users can get step-by-step navigation from their current location.

- Categorized Services Directory
    - Locations are grouped into categories (Healthcare, Safety, Transport, Education, Government, Tourism, etc.).
    - Users can filter and sort results by distance, rating, or service type.

3.2 External Interface Requirements

Defines interactions with external systems.

- **User Interfaces**:
    - Web Application accessible via modern browsers (Chrome, Firefox, Safari, Edge).
    - Responsive and interactive UI with map-based navigation.

- **Hardware Interfaces**:
  - GPS-enabled devices (mobile phones, tablets, laptops with location services) to detect user location.
  - Touchscreen compatibility for mobile and tablet users.
  - Desktop compatibility with mouse and keyboard navigation.
- **Software Interfaces**:
  - Google Maps API (for map rendering, directions, and location search).
  - Database (MYSQL) for storing user profiles, preferences, and search history.

- **Communication Interfaces**:
  - Real-time updates via REST APIs or WebSocket's for live location tracking and directions.

3.3 Performance Requirements

Defines app speed, scalability, and efficiency.

- Map rendering and location search results should load within 2 seconds.
- The system must support scalability for up to 100,000 users during peak hours.
- Route generation and navigation updates should refresh within 3–5 seconds.

3.4 Security Requirements

Ensures data protection and security.

- All user credentials must be encrypted using strong hashing algorithms (e.g., crypt).
- Integration with Google Maps API must follow OAuth 2.0 secure authentication.
- User location and personal data must only be shared with explicit user consent.

3.5 Software Quality Attributes

Non-functional requirements that affect user experience.

- Usability – The app should have an intuitive, simple UI with clear navigation for users of all age groups.
- Reliability – The system should maintain 99.9% uptime to ensure continuous access to maps and services.
- Maintainability – The codebase should follow modular architecture to allow easy updates and bug fixes.
- Portability – The app should work seamlessly across web, Android, and iOS platforms.

3.6 Other Non-Functional Requirements

Additional system requirements.

- Compatibility – The application must run smoothly on major browsers (Chrome, Firefox, Safari, Edge) and mobile platforms (Android and iOS).

- Accessibility – The app should follow WCAG standards to ensure usability for people with disabilities.
- Maintainability – The system should be designed with modular architecture to ensure that updates, bug fixes, and feature enhancements can be implemented efficiently without downtime.

Instructor's Signature & Comments (For Lab Use):