Amrita School of Computing

23CSE311 Software Engineering

Lab Worksheet Number 1

Software Project Finalization, Planning & Feasibility Study

Date: 19 - 08 -2025

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

Objective:

To design and develop a navigation and safety-focused software system that provides shortest path routing, detects disaster zones, issues national border alerts, suggests alternate routes, delivers live hazard notifications, offers voice-guided navigation, and locates nearby emergency services by integrating real-time data from reliable APIs for enhanced travel safety and efficiency.

Team Information:

Team Member Name	Student ID	Role in Project
SHAJITH	AM.SC.U4CSE23231	• UI Design • Frontend Dev • Testing
G. TEJASWINI	AM.SC.U4CSE23219	• Backend APIs • Integration with Frontend • Documentation
GUNA ABIRAM	AM.SC.U4CSE23236	• Database Design • Backend Queries • API Testing
SD. MUSAFIRUNNISA BEGUM	AM.SC.U4CSE23264	• Geospatial Data Processing • Map Integration • Deployment

1. Project Title:

CIVIGUIDE

A smart civic-safety navigation system that combines shortest path routing with real-time hazard detection, border alerts, and emergency service locators to enhance travel safety.

2. Project Description:

This project aims to develop a comprehensive navigation and safety application that enhances travel efficiency while prioritizing user safety. The system will provide shortest path routing from source to destination, detect disaster-prone zones such as floods or landslides, and issue alerts when approaching national borders . It will also suggest alternate routes to avoid hazards and offer voice-guided navigation for hands-free operation. Additionally, it will help users locate nearby emergency services such as hospitals and police stations. By combining real-time geospatial data, API integrations, and intuitive mapping interfaces, the application will serve as a reliable travel companion for both everyday commuting and emergency situations.

3. Relevance of the Software:

The demand for intelligent navigation tools that ensure safety and efficiency has grown globally, especially in areas prone to natural disasters or geopolitical restrictions. CiviGuide enables individuals, travelers, and emergency responders to navigate safely, avoid hazardous zones, and access emergency services in real time.

4. Existing Related Software:

Existing Tools: Google Maps, Waze, HERE WeGo
Limitations: Limited hazard zone integration, lack of national border alerts, and minimal disaster-specific route planning.
Our Improvement: Incorporates real-time hazard data, geofencing for border alerts, alternate safe routes, and emergency service locators — features not combined in traditional navigation apps.

5. Novelty / Innovation in Your Approach:

- Real-time disaster zone detection using NASA EONET & GDACS
- National border crossing alerts via geofencing
- Alternate safe route suggestions avoiding hazards/restricted areas
- Integrated emergency services locator for hospitals, police stations, etc.
- Voice navigation for hands-free guidance

6. Front-End Technologies:

- HTML5, CSS3, JavaScript
- Leaflet.js (for maps)
- Bootstrap / (for UI styling)

7. Back-End Technologies:

- **Programming Language:** Python

- **Framework:** FastAPI (lightweight, fast backend for REST APIs)
- **Routing & Geospatial Processing:**
- OpenRouteService API (routing)
- Shapely (geofencing & spatial calculations)
- GeoPandas (for handling geographic datasets)

Disaster Data Integration:

- NASA EONET API
- GDACS API
- **Data Storage :** PostgreSQL with PostGIS extension

8. Techniques & Tools to be Used:

Techniques:

- API integration for real-time data
- Geofencing (lat/long restricted zones
- Hazard mapping with GeoJSON
- Dynamic safe route suggestions
- Voice navigation using browser Web Speech API

Tools:

- **Frontend:** HTML5, CSS3, JavaScript, Leaflet.js, Bootstrap
- **Backend:** Python, FastAPI, Requests, Shapely, GeoPandas
- **Version Control:** Git & GitHub
- **Testing:** Postman for API testing, Pytest for backend unit testing
- **Data Formats:** JSON, GeoJSON

9. Software Requirements (Functional & Non-Functional):

Functional Requirements

- Shortest path routing from source to destination
- Hazard zone alerts (flood, earthquake, landslide, etc.)
- Border alerts with geofencing
- Alternate safe route suggestions

- Voice-guided navigation
- Emergency services locator
- Offline mode (basic map access)

Non-Functional Requirements

- Scalability – should handle multiple API requests
- Performance – fast route calculation (< 2s)
- Usability – accessible across devices
- Reliability – accurate hazard data from official sources
- Security – protect user location data

10. Feasibility Study – Guided Questions

Problem Definition

- Problem: Unsafe navigation in disaster/border-prone areas.
- Users: Tourists, daily commuters, emergency responders.

Solution Strategy

- Real-time hazard integration + routing + alerts.
- Platform: **Web app (primary)**; mobile app possible later.

Alternative Evaluation

- Build from scratch (higher effort, but full control).
- Extend Google Maps API with overlays (lower effort, but less unique).

Technical Feasibility

- Yes, the team has web dev + Python API + GIS basics.
- Challenge: handling real-time API data + map rendering.

Resource Feasibility

- Tools: Open-source (React/Leaflet, FastAPI, GitHub).
- Hardware: Laptops are sufficient.

Cost/Benefit Analysis

- Cost: Minimal (time + effort, free tools).
- Benefit: Unique hazard-aware navigation system.

Constraints Identification

- Time: ~10–11 weeks project timeline.
- Data: Dependent on free APIs (NASA, GDACS).
- Hosting: Local deployment for demo.

Risk Assessment

- API downtime → fallback to cached data.
- Map rendering lag → optimize markers.
- Data inaccuracy → cross-verify multiple APIs.

11. Planning Phase / Project Timeline:

Phase	Week	Activities
Requirements Gathering	Week 1	Brainstorming, finalize APIs, feasibility
Design	Week 2-3	UI mockups, map design, API schema
Development – Phase I	Week 4-6	Basic map, routing, alerts
Development – Phase II	Week 7-9	Hazard integration, emergency locator, voice nav
Testing	Week 10	Unit testing, bug fixing
Deployment & Finalization	Week 11	Deploy on localhost, prepare report

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