Software Requirements Specification – QuakeAlert

Table of Contents

1.Product Perspective

QuakeAlert is a stand-alone mobile application designed using Expo (React Native). It operates independently by integrating with external APIs to retrieve earthquake data and deliver alerts to users.

2.Product Functions

- Fetch real-time earthquake data from open APIs
- Send push notifications to users based on location
- Display safety guidance during an emergency
- Show a list of recent earthquake events
- Allow users to configure notification settings

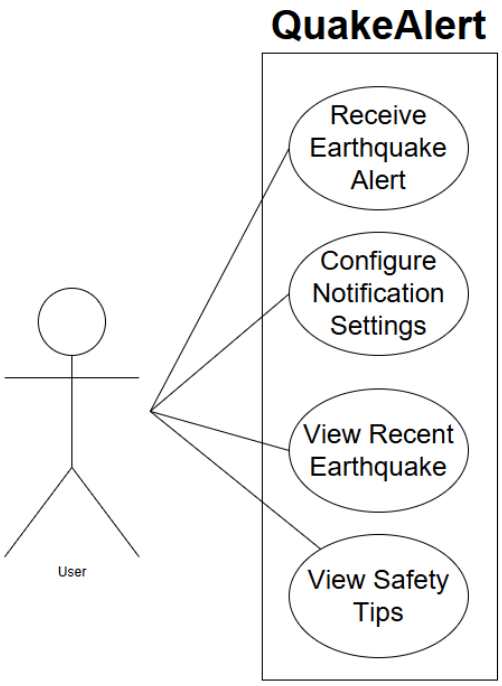
3.User Characteristics

- No technical expertise required
- Familiarity with standard mobile apps
- Users reside in earthquake-prone regions (e.g., Turkey)

4.Constraints

- Mobile-only (Android/iOS)
- Limited to public earthquake APIs
- Notifications depend on internet and device permission
- Language: Turkish & English

5.System Features (Use Case Based)



Use Case 1: Receive Earthquake Alert

- **Title:** Real-time Alert Notification
- **Main Actor:** System
- **Goal:** Deliver an alert when a nearby earthquake is detected
- **Preconditions:** Device has internet connection and notifications enables
- **Main Flow:**
    1. Earthquake data is fetched
    2. System checks user location
    3. If relevant, a push notification is sent
- **Postconditions:** User receives timely alert with safety info

Use Case 2: Configure Notification Settings

- **Title:** Manage Alert Preferences
- **Main Actor:** User
- **Goal:** Enable or disable earthquake notifications
- **Preconditions:** App installed and running
- **Main Flow:**
    1. User opens settings
    2. Switch toggle is updated
    3. Preference is stored locally
- **Postconditions:** System respects user's preferences

Use Case 3: View Recent Earthquakes

- **Title:** Access Earthquake History
- **Main Actor:** User
- **Goal:** Check list of recent seismic events
- **Preconditions:** Internet access
- **Main Flow:**
    1. App fetches latest earthquake data
    2. Displays them in a scrollable list
- **Postconditions:** User sees past events

Use Case 4: View Safety Tips

- **Title:** Emergency Guidance Display
- **Main Actor:** User
- **Goal:** Read “Drop, Cover, Hold On” tips
- **Preconditions:** App is open
- **Main Flow:**
    1. User clicks “Safety Info”
    2. Textual/visual guidance appears
- **Postconditions:** User reads safety instructions

6.Non-Functional Requirements

Usability: Intuitive UI, designed for fast comprehension

Performance: Notifications should be delivered <10 seconds after detection

Portability: Compatible with Android and iOS

Reliability: App should not crash during high-traffic API requests

7.External Interface Requirements

- AFAD or USGS API for real-time data
- Expo push notification service

8.User Interfaces

Home Screen, Settings Screen, Safety Info Screen, History Screen

9.Software Interfaces

- JavaScript/TypeScript environment
- Expo Notifications SDK
- Axios or Fetch API for backend integration

10. Task Matrix

Task	Responsible Member
UI description and system features	Altar
Use case design and flow writing	Talha
Non-functional requirements	Samet
External/software interfaces	Yakup
Review and structure	Altar