# Software Architecture Overveiw– QuakeAlert

## Table of Contents

# 1.Architecture Style

**Layered Client-Server Architecture**

QuakeAlert follows a layered architecture, where the mobile frontend communicates with external APIs and notification services over HTTP. The system consists of loosely coupled layers to ensure maintainability, flexibility, and scalability.

# 2.Major Components

**Mobile Client:** Expo (React Native) frontend that handles UI, user preferences, and notifications

**Earthquake API Layer:** Connects to AFAD/USGS or any real-time seismic data provider via HTTP

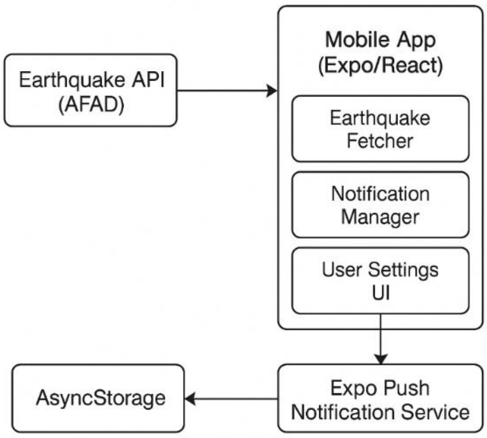
**Notification Service:** Expo Push Notification system that delivers alerts to users' devices

**Preference Storage:** Local device storage (AsyncStorage or SecureStore) for saving user settings

# 3.Component Interactions

- The mobile app fetches earthquake data periodically or via polling from AFAD/USGS API.
- When an earthquake relevant to the user's location is detected, the app triggers a push notification using Expo's notification service.
- The user's notification settings are stored locally and used to filter out alerts.
- The app UI displays recent earthquakes and safety tips via clean component-based screens.

# 4.Architecture Diagram



# 5. How Architecture Support Use Cases

- **Receive Earthquake Alert:** Notification Manager listens for new data and uses Expo to push alerts
- **Configure Notification:** Settings saved locally in AsyncStorage, used to filter alert logic
- **View History:** Earthquake Fetcher gets latest data from API and shows on screen
- **View Safety Tips:** Static content displayed via predefined UI screen in the app

# 6.Task Matrix

| Task | Responsible Member |
|---|---|
| Architecture layer description | Altar |
| Component interaction explanation | Talha |
| Diagram structure and layout | Samet |
| Mapping use cases to components | Yakup |
| Final review | Altar |