Detailed Project Plan

Project Objectives

The primary objective of this project is to develop a women's safety app that offers reliable, quick, and accessible emergency support, especially during night-time or in vulnerable situations. The app is designed with the following goals:

- **Enhance Personal Safety**: Provide an easy-to-use, fast response tool for women, with features like one-tap SOS alerts, discreet shake-to-alert, and location sharing.
- **Facilitate Rapid Response**: Reduce emergency response times by integrating real-time location tracking and notifications to both trusted contacts and local authorities.
- **User-Centered Design**: Create a minimalist, user-friendly interface that's accessible even in high-stress situations.
- **Privacy and Data Security**: Protect sensitive user data (e.g., location, contact information) through encryption, secure storage, and controlled access.
- **Community Awareness and Feedback**: Engage with the community to incorporate real needs and continuously improve the app's functionality.

Timeline and Milestones

This project will follow a phased timeline to ensure organized development and timely completion:

Week 1-2: Research and Requirements Gathering

Conduct in-depth research on existing safety solutions and gather requirements for a comprehensive understanding of user needs.

Week 3: Design Phase

Create detailed wireframes and UI mockups of the app, focusing on ease of use and essential emergency features. Review and iterate based on feedback.

Week 4-5: Core Development

Develop primary features such as SOS alert functionality, real-time location sharing, and shake-to-alert. Set up backend APIs and integrate with cloud services.

Week 6: Backend and Cloud Services Integration

Deploy the backend to Vultr or similar cloud services, ensuring scalability and security. Set up push notification services and real-time data handling.

Week 7: Testing and Feedback

Conduct thorough testing to identify and resolve bugs. Collect user feedback on usability and refine the app based on insights.

Week 8: Final Deployment and Documentation

Complete documentation, finalize the app, and deploy the final version. Prepare for submission with all required deliverables.

Deliverables

- **Prototype/Initial Codebase**: A functional prototype with core features like SOS alerts, location sharing, and the shake-to-alert feature.
- **Final App**: Fully functional, tested, and user-ready version of the app.
- **Technical Documentation**: Detailed documentation covering system architecture, key components, API specs, and setup instructions.

High-Level Architecture Diagram

The app architecture will include the following components:

- **User Interface (UI)**: Frontend screens allowing users to access the SOS alert, real-time tracking, and setup options for emergency contacts.
- **Backend Server**: Handles data processing, manages API calls, and interacts with the database for location updates and emergency contact management.
- **Database**: Stores user data, including contacts, app settings, and historical logs of SOS alerts.
- **Notification Service**: Manages real-time notifications and alerts to emergency contacts.
- **Cloud Infrastructure (Vultr)**: Manages scalability, security, and data redundancy, ensuring reliable, always-on support.

Component Diagram

Each core feature is represented in separate components:

1. **SOS Alert Component**: Manages emergency alerts and notifications.
2. **Location Sharing Component**: Uses GPS to send real-time location updates.
3. **Fake Call Feature**: Provides a simulated call interface for the user's protection.
4. **Settings and Configuration**: Allows users to add or modify trusted contacts and customize app features.
5. **Backend APIs**: Endpoint integration for triggering alerts, managing contacts, and updating user location.

Network Topology

If Vultr's cloud infrastructure is used, the network topology would include:

- **User Device**: Connects to the app's frontend and backend over the internet.
- **Vultr Compute Instance**: Hosts the backend server, handling data processing and API requests.
- **Object Storage and Database**: Cloud storage for user data and application logs.
- **Notification Service (e.g., Firebase)**: Routes notifications securely to user contacts.

Technical Documentation

System Architecture and Design

The app's system architecture is designed for high availability, data security, and low latency. The backend, deployed on Vultr's cloud services, handles API requests and data processing, while real-time notifications are managed through integrated cloud messaging services.

- **User Flow**: When the user taps the SOS button or activates shake-to-alert, an emergency alert is immediately triggered. This initiates location sharing, which is processed by the backend and sent to emergency contacts.
- **Data Security**: All sensitive data is encrypted before storage. User data, such as emergency contacts and location history, is stored securely in a cloud-based database with restricted access.

Key Components and Modules

1. **SOS Alert Module**: Activates emergency features with a single tap, contacting authorities and sharing the user's location with trusted contacts.
2. **Location Tracking Module**: Uses GPS to continuously update the user's location and share it with trusted contacts until the user deactivates it.
3. **Shake-to-Alert**: Allows users to discreetly activate the SOS alert by shaking their phone, providing an easy way to call for help without direct interaction.
4. **Notification Service**: Manages the routing of alerts to contacts in real-time, ensuring notifications are promptly received.

API Documentation (if applicable)

Include basic API documentation for key endpoints:

- **/api/sos**: POST request to trigger SOS alert.
- **/api/location**: POST request to share real-time location.
- **/api/contacts**: GET/POST requests to manage user's trusted contacts.

Setup and Usage Instructions

1. **Installation**: Download the app package, install it on a mobile device, and complete the initial setup.
2. **Configuration**: Open settings to add trusted contacts and configure features like shake-to-alert.
3. **Using SOS Alert**: In an emergency, press the SOS button to notify contacts or shake the device to activate the alert discreetly.
4. **Viewing Location**: Access location updates in real-time by checking linked contacts.

Prototype or Initial Codebase (GitHub Repository)

- **GitHub Repository Link**: https://github.com/Shubh-Dhariwal/Emergency-Call-Application
- **Contents**:
  Source Code: Organized by module for easier understanding and collaboration.
  Build Instructions: Step-by-step guide on compiling and running the app, with and necessary environment setup instructions.
  README: Project overview, feature list, setup instructions, and known issues.