Product Requirements Document (PRD) for ER Wait Time App

---

1. Introduction

1.1. Purpose

The purpose of this document is to outline the requirements, features, and roadmap for the development of an ER wait time application. This app will provide real-time emergency room wait times for hospitals, using data extracted programmatically from known URLs. The application aims to help patients make informed decisions by providing the most up-to-date wait times and suggesting the optimal hospital based on their location and current wait times.

1.2. Scope

The scope of this document includes the core functionalities of the ER wait time app, including data extraction, user interface, API integrations, and future enhancements. It will serve as a roadmap for the development team to ensure a clear understanding of the project requirements and milestones.

2. Project Overview

2.1. Objectives

- Provide real-time ER wait times for hospitals.
- Offer a user-friendly interface for accessing wait time data.
- Integrate with Google Maps API to provide estimated travel times.
- Suggest hospitals with the shortest wait times and optimal travel routes.
- Schedule regular updates to ensure data accuracy.

2.2. Key Features

- Real-time wait time extraction from hospital websites.
- Map view displaying nearby hospitals with wait times.
- Estimated travel times and directions.
- Suggested hospitals based on location and wait times.
- Regular data updates via cron jobs.
- Additional hospital information and services.

3. Functional Requirements

3.1. Data Extraction

- **Manual URL List**: Initially, the app will use manually collected URLs where hospitals post ER wait times.

- **Programmatic Data Extraction**: Develop a method to programmatically extract wait times from these URLs.
- **Data Parsing**: Parse the extracted data to identify and store wait times.

3.2. User Interface (UI)

- **Home Screen**: Map view with current location and nearby hospitals.
- **Hospital Information**: List of hospitals with wait times and services offered.
- **Wait Time Display**: Real-time wait times for each hospital.
- **Suggested Hospitals**: Recommendations based on shortest wait times and estimated travel times.

3.3. API Integrations

- **Google Maps API**: Integrate to provide map functionality, estimated travel times, and directions.
- **Traffic Data API**: Use traffic data to improve travel time estimates.

3.4. Data Update Mechanism

- **Cron Jobs**: Schedule data extraction and updates every 15 minutes to ensure accuracy.
- **Error Handling**: Implement error handling for failed data extraction attempts.

3.5. Additional Features

- **Urgent Care Integration**: Expand to include urgent care centers.
- **Insurance Integration**: Allow users to filter hospitals based on in-network providers.
- **Hospital Services**: Display additional services provided by each hospital.

4. Non-Functional Requirements

4.1. Performance

- The app should load wait time data within 3 seconds.
- Updates should run efficiently in the background without impacting user experience.

4.2. Scalability

- The system should be designed to handle an increasing number of hospitals and users.
- Future integrations (e.g., urgent care, insurance) should be easily incorporable.

4.3. Security

- Ensure data privacy and secure handling of user information.
- Protect against potential scraping blocks and CAPTCHAs.

4.4. Usability

- The UI should be intuitive and easy to navigate.
- Provide clear instructions and feedback to users.

5. Technical Requirements

5.1. Platform

- Web application, initially with a potential mobile app in the future.

5.2. Technology Stack

- **Frontend**: React.js or Vue.js for the user interface.
- **Backend**: Python (Flask/Django) for data extraction and API handling.
- **Database**: PostgreSQL or MongoDB for storing hospital data and wait times.
- **APIs**: Google Maps API, Traffic Data API.

6. Milestones and Roadmap

6.1. Phase 1: Proof of Concept

- Collect URLs and manually extract wait times.
- Basic web app with map view and wait time display.
- Programmatic data extraction and parsing.

6.2. Phase 2: Core Functionality

- Integrate Google Maps API.
- Develop recommendation algorithm.
- Implement cron jobs for regular updates.

6.3. Phase 3: Enhancements

- Add urgent care centers.
- Integrate insurance provider filters.
- Display additional hospital services.

6.4. Phase 4: Future Expansions

- Mobile app development.
- Additional features based on user feedback and market needs.

7. Assumptions and Dependencies

- Hospital websites consistently update their wait times.
- APIs (Google Maps, Traffic Data) remain accessible and reliable.
- Adequate resources and expertise available for development.

8. Risks and Mitigations

- **Data Inconsistency**: Implement regular checks and error handling.
- **API Limitations**: Monitor usage and optimize API calls.
- **Scalability Issues**: Plan for cloud infrastructure to handle growth.