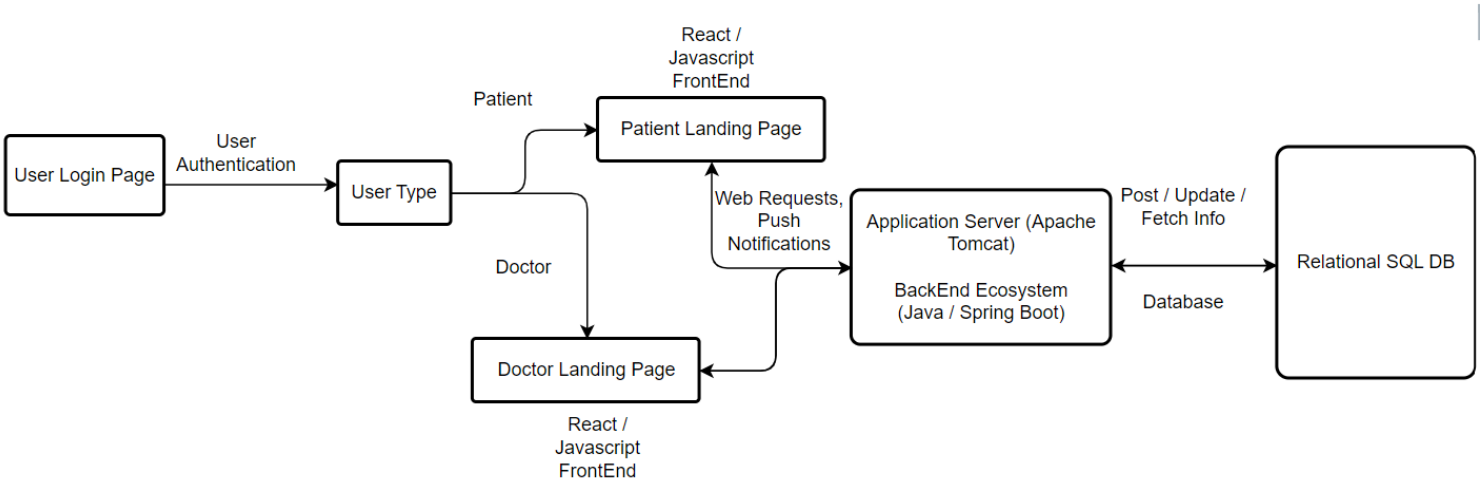
Design

2.1. Architecture Diagram



2.2. Technology Stack with Justification

1) Frontend: JavaScript and React

JavaScript is a widely used programming language for web development supported by all modern web browsers and a vast developer ecosystem for creating interactive and dynamic user interfaces. We can implement classes also as per need to make the code structure object oriented.

React, a JavaScript library for building user interfaces, is an easy to use component driven framework. Its component-based architecture makes it easy to create reusable UI elements and manage the frontend efficiently.

Major Advantages:

-> Virtual DOM feature that enhances performance by minimizing direct manipulation of the actual DOM, resulting in faster UI rendering.

-> An active community and good online documentation that contributes towards learning about the libraries, tools and resources to expedite development.

-> React supports server-side rendering, which generally speaking is beneficial for initial load times.

-> Its component reusability and a data flow make it easier to maintain and scale your frontend.

2) Backend: Java and Spring Boot

Java is a robust, mature, widely adopted and reliable object oriented programming language, known for its scalability, security, and extensive libraries. In a healthcare application like the Patient Tracker System, data security and scalability are paramount. Additionally, the extensive Java ecosystem provides various libraries and frameworks for efficient development. Also being an object oriented language, it allows us to structure our code keeping in important principles like modularity, reusability and other OOPS principles

Spring Boot is also a Java framework that simplifies the development of Java based applications. It provides a range of features that are well-suited for our project, including:

-> Ease of Development: It simplifies the development process by providing defaults and auto-configurations, which reduce the need for extensive setup.

-> Microservices Support: If our project needs to scale or extend its capabilities with microservices in the future, Spring Boot provides excellent support for building microservices.

-> Security: Spring Boot includes robust security features, including authentication and authorization, which are critical for protecting patient data.

-> Database Integration: Spring Boot makes it easy to integrate with various databases, which is essential for storing and managing patient records.

-> Community Support, Documentation: Spring Boot has a large and active community and well documented help websites, so we can find plenty of resources and support as we develop our application.

-> Compliance with Java EE Standards: Spring Boot adheres to Java EE standards, which ensures our application's compatibility with established best practices.

While developing a Java based app, we'll also need a web server or application server to run our Java application. Since you're using the Spring Boot framework for your Java application, we can take advantage of its embedded servers like Tomcat. Spring Boot simplifies the deployment process and offers flexibility in choosing the embedded server that suits our needs.

3) Database: RDBMS

Relational Database Management Systems (RDBMS) are known for their data consistency, ACID compliance, and mature query languages. They are suitable for healthcare applications where structured data is important, and relationships between different data points need to be maintained.

Features:

-> Structured Data: Since our patient data is highly structured and follows a well-defined schema with many relationships (e.g., patients, doctors, appointments, medical history), an RDBMS is a natural fit. SQL databases excel at maintaining data integrity and enforcing relationships between tables.

-> ACID Compliance: RDBMS systems are known for their strong ACID (Atomicity, Consistency, Isolation, Durability) compliance, making them suitable for applications where data consistency and reliability are critical, such as healthcare.

-> Complex Queries: If our application requires complex queries, reporting, and analysis of the patient data, SQL databases provide powerful querying capabilities and join operations.

-> Data Integrity: Healthcare applications demand a high level of data integrity and security. SQL databases provide robust security mechanisms and are well-suited for maintaining patient data privacy.