Project Report Outline

Patient Tracker Management System

Midpoint Deliverable

Mani Kishan Ghantasala Satya Sai Prudhvi Krishna Nikku Satya Sriram Potluri
Srimathi Mahalingam

Github Repo Link: https://github.com/sh4nnu/patient-tracker-system

Recording link: video2033540104.mp4

UI Mockups : Mockup UI

1. Requirements

1.1. Overview

Healthcare ecosystem is one of the complex and highly data traffic, every hour a lot of documents are generated and needed to be processed for every event that happens starting from incident response, to consulting, treatment, appointments, insurance etc., with such heavy traffic transactions going on both in money and information. Resorting to paperwork is going to slow the process by a lot and limits the throughput of the services for patients in need.
Hence, digitizing the patient tracking system is of high importance. It replaces the manual paperwork, and mitigates the risks of potential errors, time-consuming processes, and limited accessibility.

The Patient Tracker System is a web-based application designed to streamline the management process of patient information and medical records within the contemporary healthcare landscape. This project is aimed to provide doctors and administrative staff of a healthcare institute with a streamlined tool and features to enhance managing, decision-making, increasing productivity and improve the quality of patient care. The application follows best practices in software development in producing a viable, reliable, efficient and secure product. This solution not only streamlines the process but also opens opportunities for data analysis to improve patient care, availability , trend analysis which helps the practitioners and everyone involved to make data-informed decisions. Such a system would help and ease the work and also is reliable.

Objectives:

The objectives of this project can be split as below:

1. **Accuracy and Efficiency:**

The paramount objective of the Patient Tracker System is to significantly improve the efficiency and accuracy of patient data management. Eliminating the laborious and error-prone process with manual paper work and decreasing the errors in data, and also improving the accessibility of data to make decisions quickly and efficiently.

2. **Real-time Updates:**

In today's dynamic environment and the vast amount of data available, instant access to patient records is not merely a convenience but a necessity. The application will be designed to provide real-time updates for the medical records. And enables the patients to update their personal records wherever and whenever.

3. **Management and supply chain:**

Third and most important objective is to cater to the supply chain needs of a healthcare facility. Giving ability to book and schedule appointments, visualize them, send messages or notifications to patients and doctors all these tasks included with other management features and analytics on the data available enhances and enables ease of the management of patients and process. Makes the life of patients, admins and doctors etc., easier.

Intended Audience:

The intended audience for the Patient Tracker System encompasses several key stakeholders:

1. **Doctors and Healthcare Professionals:** The system is primarily tailored to the needs of healthcare practitioners, offering them a comprehensive and easy-to-use platform for managing patient information, scheduling appointments, and making data-driven medical decisions
2. **Administrative Staff:** Helps in streamlining overall operations of the healthcare clinic.
3. **System Administrators:** will be tasked with managing user roles and ensuring the system's smooth operation, including data security and system reliability
4. **Patients:** The stakeholders whose data and for whose treatment and process the application is made for.

Summary and Motivation:

The motivation behind the Patient Tracker System is to address the inherent challenges in the current healthcare ecosystem, marked by laborious manual paperwork, potential data inaccuracies, and the need for efficient and accessible patient data and process management tools. The project is driven by the conviction that technology can empower healthcare practitioners to deliver higher-quality care, reduce administrative overhead, and enhance the overall patient experience.

In summary, the Patient Tracker System is a transformative project that aspires to make a significant impact on the healthcare industry by providing healthcare professionals with the tools they need to offer more efficient and accurate patient care, ultimately leading to improved patient outcomes. And enabling the people at different levels to make better decisions, operate and manage effectively improving the quality of care.

1.2. Features

The following are the main features of the project:

1. **User Authentication**: A secure login system to authenticate both doctors and patients, ensuring that only authorized individuals can access the application.
2. **Patient Profile Managemen**t: Allow system admins to create, update, and maintain their personal and medical profiles within the application. This includes personal details, medical history, and medication records.
3. **Appointment Scheduling**: Enable doctors to schedule and manage patient appointments, facilitating efficient time management.
4. **Real-time Medical Records**: Provide a platform for healthcare professionals to access and update patient medical records in real time during patient checkups. Also, allow users to update their records and information and doctors to monitor and get notified on improvements etc.,
5. **Notifications and Alerts**: A notification system that allows doctors to send and receive real-time notifications and alerts related to patient appointments, test results and updates.
6. **Dashboard and Analytics**: Variety of dashboards that offer data visualization and analytics features, empowering healthcare professionals to make data-driven decisions.
7. **Access Control**: Ability to define user roles and access levels, ensuring that each user has the appropriate level of access to patient data and system features to maintain data security and privacy.

1.3. Functional Requirements (Use cases)

Detailed functional use-cases of the application by different users.

Doctor (any health care professional) -Related Use Cases:

1. **Doctor Authentication**: As a doctor, I want to log in securely to the system using my credentials.

2. **View Appointments**: As a doctor, I want to view a list of patients scheduled for the day so that I can prepare for consultations.

3. **Access Patient Records**: As a doctor, I want to access and review the complete medical history, including diagnoses, medications, and test results, for a specific patient during a visit.

4. **Update Patient Records**: As a doctor, I want to update a patient's medical records, including adding new diagnoses, changing medications, and recording new observations or recommendations during a patient visit.

5. **Receive Patient Updates**: As a doctor, I want to receive real-time updates when a patient submits new information or modifies existing records, ensuring I stay informed about any developments.

6. **Access Reports**: As a doctor, I want to access data analytics and reports to gain insights into clinic performance and patient data for informed decision-making.

9. **Communication & send Notification** : As a doctor, I want to send secure messages or notifications to patients regarding their appointments, prescription updates, or other important information.

Patient-Related Use Cases:

1. **Patient Authentication**: As a patient, I want to log in securely to the system using my credentials.

2. **View Appointments**: As a patient, I want to view and confirm scheduled appointments, ensuring I'm aware of my healthcare appointments.

3. **Access Personal Medical Records**: As a patient, I want to access and review my complete medical history, including diagnoses, medications, and test results stored in the system.

4. **Update Personal Information**: As a patient, I want to update my personal information, such as contact details, insurance information, and medical history within the system.

5. **Upload Medical Records**: As a patient, I want to upload medical records, test results, and relevant documents electronically/ by filling a form into the system, ensuring that my healthcare data is centralized and up to date.

6. **Receive Updates and Notifications**: As a patient, I want to receive real-time notifications when a doctor generates or updates a treatment, allowing me to view medication instructions and track my medication history along with notifications on my upcoming appointments.

Administrative-Related Use Cases:

1. **Profile Creation:** As an administrator I want to create profiles for both doctors and patients.
2. **Role Assignment:** As an administrator I want to assign roles to certain users so the features and data they have access to are decided.
3. **Row level security based on role:** Assign role and security level of data to be open for a given user.
4. **Rescheduling appointments:** In case of a doctor's absence I need the access to reschedule appointments or cancel them or route them to different doctors.
5. **Send Notifications:** Send notifications to patients or doctors on anything they should be knowing.

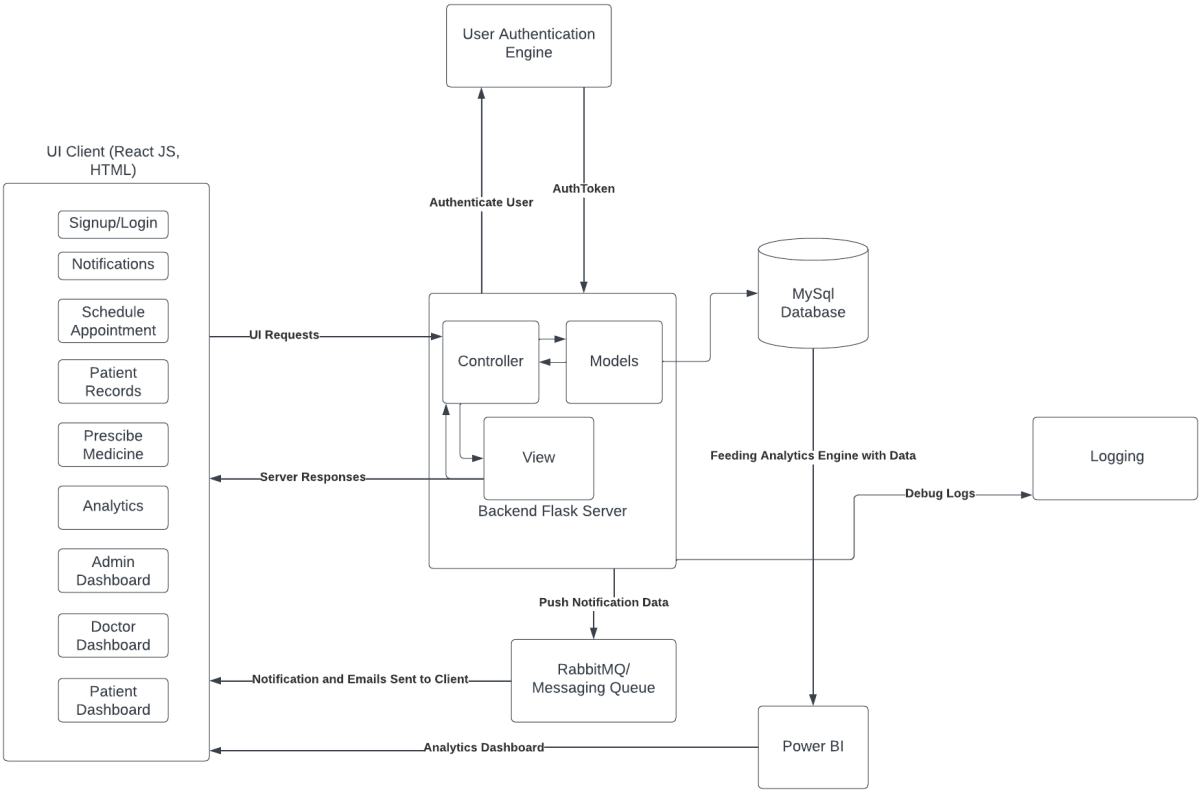
1.4. Non-Functional Requirements

The following are the non-functional requirements of the project.

1. **Data Security and Encryption:** The system should ensure the security and privacy of patient data. All patient records, personal information, and medical history should be encrypted both at rest and during the transmission to prevent unauthorized access.
2. **Real-time access to data:** Request to access any record or file should be catered immediately i.e, no queues or mails. The application should be showing them the data requested as per the user's access level to the data.
3. **User-Friendly Interface:** The user interface should be intuitive and user friendly for both the doctors and patients to perform their tasks with minimal effort.
4. **Following the HIPAA rules:** All the health and patient related data should follow HIPAA rules and regulations and are controlled, stored , requested, masked accordingly.
5. **Understandability:** The project will prioritize comprehensive documentation and meticulous commenting across all components, ensuring a clear understanding of the entire codebase and facilitating seamless collaboration among team members.
6. **Debuggability:** To enhance application debuggability, the project will integrate a robust logger module, enabling efficient error tracking and streamlined debugging processes for enhanced system maintenance and troubleshooting.
7. **Modularity:** The project development will adhere to a modular approach, emphasizing the creation of independent and reusable components to ensure flexibility, scalability, and ease of maintenance throughout the software lifecycle.

2. Design

2.1. Architecture Diagram



2.2. Technology Stack with Justification

Technology Stack with Justification:

Backend Framework:

Flask: We chose Flask for the backend development of the Patient Tracker System. Flask is known for its simplicity, flexibility, and lightweight nature, which aligns with the project's need for rapid development and easy integration. It allows us to build RESTful APIs efficiently and offers a wide range of extensions and libraries for various functionalities.

Frontend Framework:

React: React is our choice for the frontend development. React is a popular JavaScript library that enables the creation of dynamic and interactive user interfaces. Its component-based

architecture, strong community support, and virtual DOM make it suitable for building a responsive and real-time user interface.

Database Management

MySQL: We have selected MySQL as the relational database management system. MySQL is a widely-used and reliable database system known for its performance and scalability. It is suitable for handling healthcare data and can provide data integrity, security, and efficient data retrieval.

2.3. UI Mockup

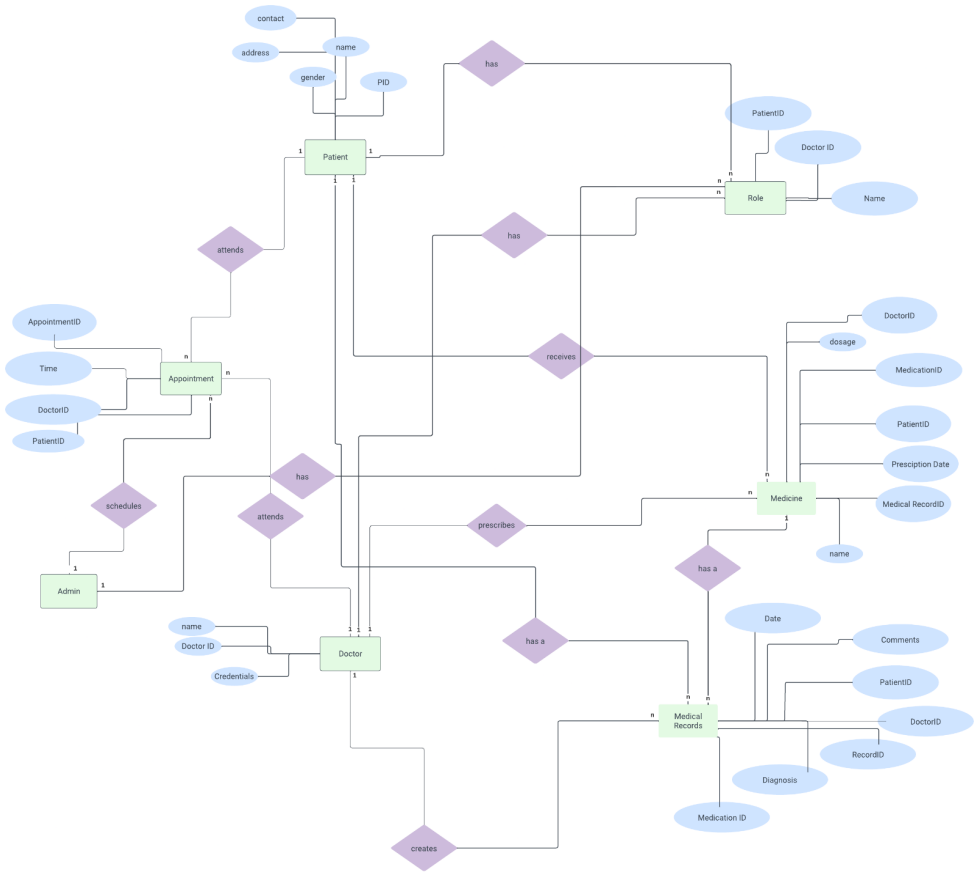
UI Mockup:

<Attached as separate files>

Mockup UI - One drive link

2.4. Data Model

ER Diagram:



3. Implementation

Outline the coding process, methodologies to be followed, and potential challenges.

The Coding Process:

For this project, our technology stack will be centered around ReactJS for the front-end components, ensuring an interactive and engaging user interface. On the backend, we have opted for Python's Flask framework, adhering to the MVC architecture to facilitate efficient and scalable development. To enable seamless communication between the client and server, we will employ RESTful APIs, leveraging Flask's robust capabilities. For effective database management, we will integrate SQLAlchemy as an Object-Relational Mapper, enabling smooth interaction with MySQL databases. To enhance the system's debuggability, we will implement a comprehensive logging system, ensuring efficient issue identification and resolution. Additionally, for efficient and reliable message delivery to UI clients, we will leverage RabbitMQ messaging queue for sending notifications, thereby ensuring smooth and timely communication with the end users.

An outline of the best practices and potential risks to address:

Modularity: The project shall be implemented as a modular design approach to promote code reusability and maintainability.

Comments and Documentation: Comprehensive comments and documentation to ensure code readability and facilitate future updates or modifications along with appropriate README.md files to promote understandability.

Version Control: Utilizing a robust version control system such as Git to track changes, collaborate effectively, and maintain a reliable history of the project. Alongside, will be maintaining separate branches for development, and testing servers/environments.

Testing and Quality Assurance: Implementing a comprehensive testing strategy, including unit tests, integration tests, and end-to-end tests, to ensure the reliability and functionality of the code.

Code Reviews: We plan on conducting regular code reviews to identify and rectify any issues or bugs and to ensure adherence to coding standards and best practices.

Security Measures: Implementing robust security practices and protocols to protect user data and prevent potential security breaches. All the default passwords (ex: default password of mysql server) and generic passwords will not be accepted.

3.1. Security and Risks

HIPAA Compliance (in case of Patient Tracker):
Data Encryption:

Ensure that data both in transit (e.g., during data transfer) and at rest (e.g., in the database) is encrypted. Use secure encryption protocols (e.g., TLS for data in transit and strong encryption algorithms for data at rest).

Access Control and Authentication:
Implement strict access controls, user authentication, and role-based access to ensure that only authorized individuals can access and modify patient health information.

Potential Risks:

Technical Risks: Potential technical challenges or complexities that may arise during development, such as compatibility issues, integration problems, or performance bottlenecks.

Security Risks: Vulnerabilities that could compromise the security of the application, leading to data breaches or unauthorized access.

Time Constraints: Risks associated with project delays ,impacting the overall development timeline.

Regulatory Compliance Risks: Risks associated with non-compliance with industry regulations or data protection laws, leading to legal consequences and reputational damage.

By addressing these best practices and potential risks, we can ensure the implementation of a robust development strategy that prioritizes quality, security, and user satisfaction while proactively mitigating potential challenges.

3. Work Plan

High-Level Timeline:

1. Setting up the Flask Project App
2. Setting up a React App
3. Creation of Database using Models.py using SQLAlchemy
4. Parallel development of User Authentication, User Profiles, Admin Module
5. Testing these modules.
6. Development of Appointment Scheduling, Patient Profile Management, and Notification Engine.
7. Testing these modules.
8. Deploying these features into a cloud platform.

Development:

Group Members:

Mani Kishan Ghantasala:

1. Setting up of React App
2. Development of Doctor UserProfile end-to-end
3. Development of Patient Profile Management end-to-end
4. Creating and testing use cases
5. Documentation and logging

Srimathi Mahalingam:

1. Creation of Appointment Scheduling feature
2. Development of Admin Module
3. Creating and testing use cases
4. Setting up Logging Module
5. Documentation and logging

Satya Sriram Potluri

1. Creation of Models using SQLAlchemy
2. Development of Patient Profile end-to-end
3. Setting up RabbitMQ

4. Documentation and logging
5. Creating and testing use cases

Satya Sai Prudhvi Krishna Nikku

1. Setting up Flask App
2. Development of User Authentication feature
3. Development of Notification workflows
4. Creating and testing use cases
5. Documentation and Logging