Patient Tracking System

Midpoint Deliverable

Devyani Mishra Dhruvi Tahilramani Shrutiya Mohan Tejaswini Amaresh

Github Repo Link: https://github.com/devyanimishra/Sprinters520

1. Requirements

1.1. Overview

A Patient Tracking System is a platform to allow the complete management of the services regarding a patient in a healthcare scenario. It is intended to be used by medical professionals like doctors, nurses, patients and administrative members alike. Each user has a separate view of the system and different credentials to register. Its primary purpose is to allow medical staff members an efficient way to manage and track patient information including associated functions like booking appointments between patient and doctor.

It is imperative to digitalize the patient tracking system in the current healthcare landscape. Relying on manual paperwork and traditional record-keeping has been exhausting, tedious and prone to vulnerabilities. Thus is it necessary to bring on a digital change. This project aims to solve this problem by providing an efficient solution for tracking patient data. This change ensures an improved healthcare system which is easier to manage for the users. It allows real time data tracking so that medical professionals will have access to the latest most accurate data which will allow better treatment plans and prescription administration.

Another feature which enables the system to be more efficient and convenient is the ability to book appointments in a digital format. This provides a time-saving and convenient approach which is advantageous to the patient since they will not have to manually schedule an appointment. This project emphasizes data encryption as well, which maintains the confidentiality and privacy of the user information.

Finally, this project aims to enhance the experience of the users by allowing them to easily access the medical records and prescription history. In conclusion, this system is meant to eliminate the shortcomings of outdated tracking techniques and introduce a new digitized effective method to do the same.

1.2. Features

- **User Authentication** - To authorize users that are either doctors, admin staff or patients to login using their credentials.
- **Patient Registration and Profile Management** - To allow a user to login with credentials and create a profile including medical history, residing address and other such details.

- **Appointment Scheduling** - To allow users to schedule appointments with the doctors based on their availability and on a specific date and time.
- **Medical Records Management** - To maintain a record of all the medications prescribed for all the patients, even maintain their backup to prevent any data loss.
- **User-friendly Dashboard** - Provide a simple dashboard which allows the user to view the details they require such as patient medical history, patients prescribed medications, doctor availability etc

1.3. Functional Requirements (Use cases)

DOCTOR

1. As a doctor, I want to log in to the system with my credentials.
2. As a doctor, I want to view the list of patients scheduled for the day.
3. As a doctor, I want to prescribe medication to a patient and have it recorded.

ADMINISTRATOR

4. As an administrative staff member, I want to log in to the system with my credentials.
5. As an administrative staff member, I want to reschedule an appointment for a patient.
6. As an administrative staff member, I want to view a list of all registered patients.
7. As an administrative staff member, I want to create, update or edit details of the patient.
8. As an administrative staff member, I want to create, update or edit details of the doctor.

PATIENT

9. As a patient, I want to register with my personal information.
10. As a patient, I want to schedule an appointment for myself.
11. As a patient, I want to access my medical history and test results.
12. As a patient, I want to search for a specific doctor's availability.

1.4. Non-Functional Requirements

- **Security of Data**: The data should be encrypted when it is stored and there should be secure transmission of data and there should not be any information leakage as a patient's health records need to be private and confidential. All users of the system must have their own login credentials in order to access the specific data.
- **Availability**: There should be very little to no downtime as patient data is crucial and should be available. As patient information is very critical at times and we need to prevent any denial of service attack. It should provide fast response times for real-time updates and data retrieval.

- **Usability:** The UI should be interactive and user friendly with easy navigation. The user should be able to use the system with ease and should have a simple UI that is user friendly.
- **Scalability:** The system should be able to manage a large amount of data. As the system will be storing patient data, doctor data, administrative staff's data, it should not lag and should be able to provide information accurately and quickly. The system should be able to accommodate the future growth of the patients, doctors and the administrative staff.
- **Prevent data loss:** There should be backups of the database in case of some database corruption to prevent any sort of data loss.
- **Maintainability:** The system should have a complete backup present at all times in case of any data error, unexpected outage or system failure. The system should track each transaction and maintain logs of it for future reference. The system should be constantly available for use.

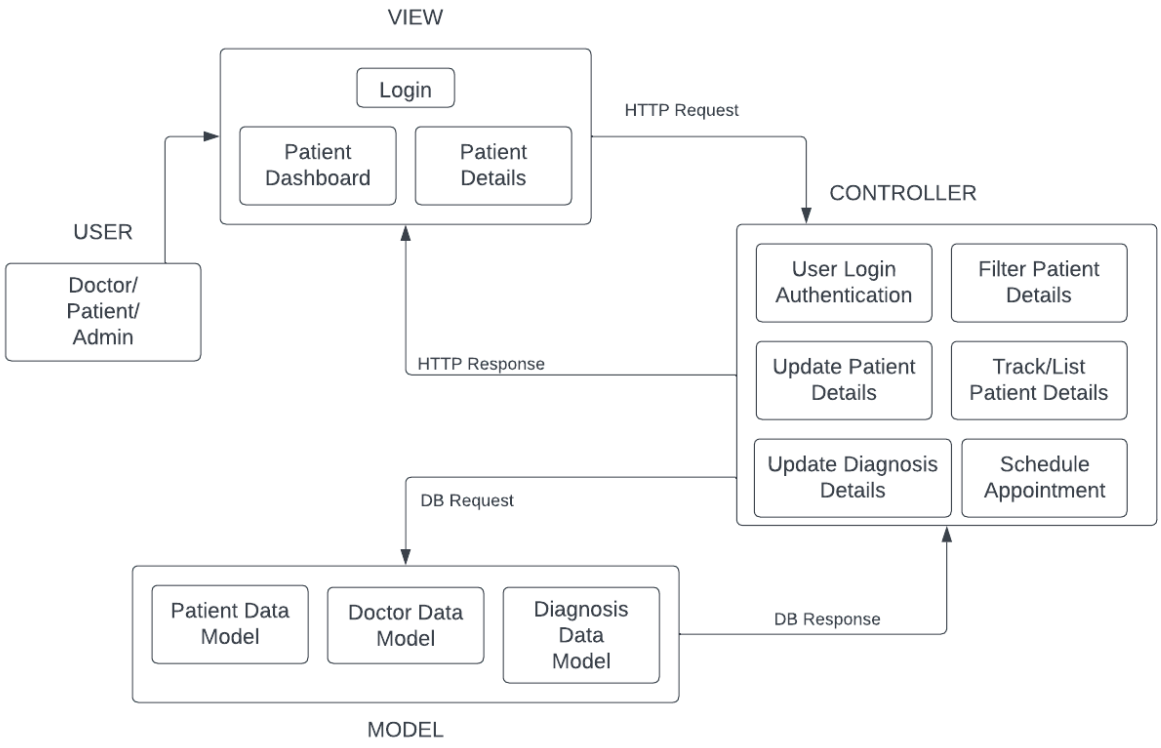
1.5. Stakeholders

- **Users:** The end users are the primary stakeholders who use the application to track medical history, prescriptions, schedule/reschedule/cancel appointments. The user population includes Patients and Doctors.
- **Developers:** The developers create the application and incrementally add/update features, improving the functionality of the application.
- **Investors:** Investors who have invested in the development and the growth of the application.

2. Design

2.1. Architecture Diagram

Patient Tracker System is based on the MVC architecture pattern represented by the below diagram.



For the implementation of the project, the object-oriented programming language python will be used.

Major Components of the Patient Tracker System are described below –

- **Model** – The model will be populated with Patient personal and login details, diagnosis details, doctor login details, which are required for the view and the application logic.
- **Controller** – The application logic resides in the controller and has multiple components.
    - Populate Model – Make use of DB connections to update the data in the model. For example: Add/Update Patient personal data, Add/Update Patient Diagnosis, Add Doctor/Patient Login data.
    - Secure Data – Authenticate the login data entered by the user and encrypt the passwords.
    - Search/Filter/List Data – Search and filter based on the list of the diagnosis and the number of previous checkups. Updates the view based on the user input.
- **View** – Essentially displays the model on the UI as requested by the user.

- Interactive Screen – Home Page with navigable tabs and dashboards.
- User Login – Can be logged in by a doctor, patient or admin.
- Dashboard – Displays the patient details scheduled for a doctor on that day.
- Detailed Patient View – Each patient list item is clickable and displays the detailed history of diagnosis for the selected patient.
- Schedule Appointment – Schedule appointment on a particular day with a doctor for a patient.

2.2. Technology Stack with Justification

The technology stack for the Patient Tracking System can be divided into the Frontend component,the backend component and the database.

Frontend: Since the Patient Tracking System is a browser-based application the frontend tech stack will be implemented using HTML, CSS and JavaScript

JavaScript- JavaScript is an open source front end scripting language and can be used to implement light weight, event driven applications.

HTML/ CSS- HTML is used to create the User Interface structure and is useful for creating consistent layouts in the Patient Tracker System.

Backend: The Backend of the Patient Tracker System is implemented using Flask and Python.

Flask- A light-weight Python based web framework to develop web applications with minimal code. Since the framework utilizes Python, it is portable and easily deployable.

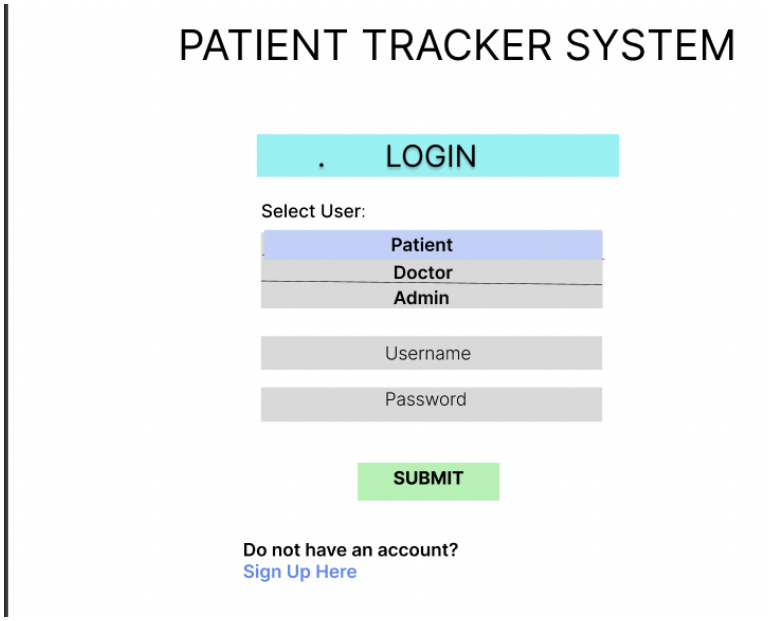
Python- A high level programming language that facilitates developing readable code with multifarious functionalities and modules. The flexibility and simplicity that Python offers makes it ideal to utilize for extensive web-based applications, such as the Patient Tracker System.

Database: A database component to store and retrieve all the details pertaining to Patients/Doctors/Admins is extremely crucial. The database component of the application will be incorporated using MongoDB

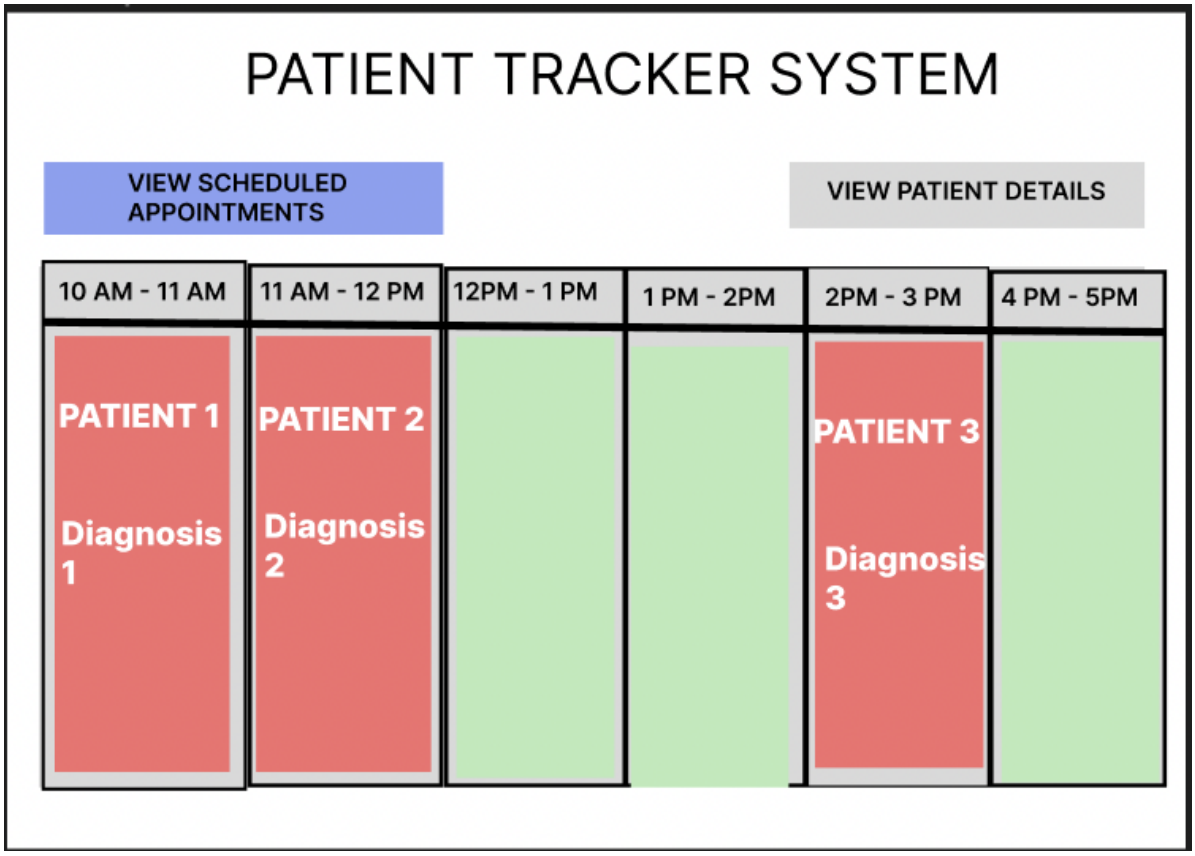
MongoDB- A NoSQL Database Management System that is scalable and efficient in storing and retrieving large volumes of data. The flexibility of MongoDB makes it the ideal database management system for the Patient Tracker System.

2.3. UI Mockup

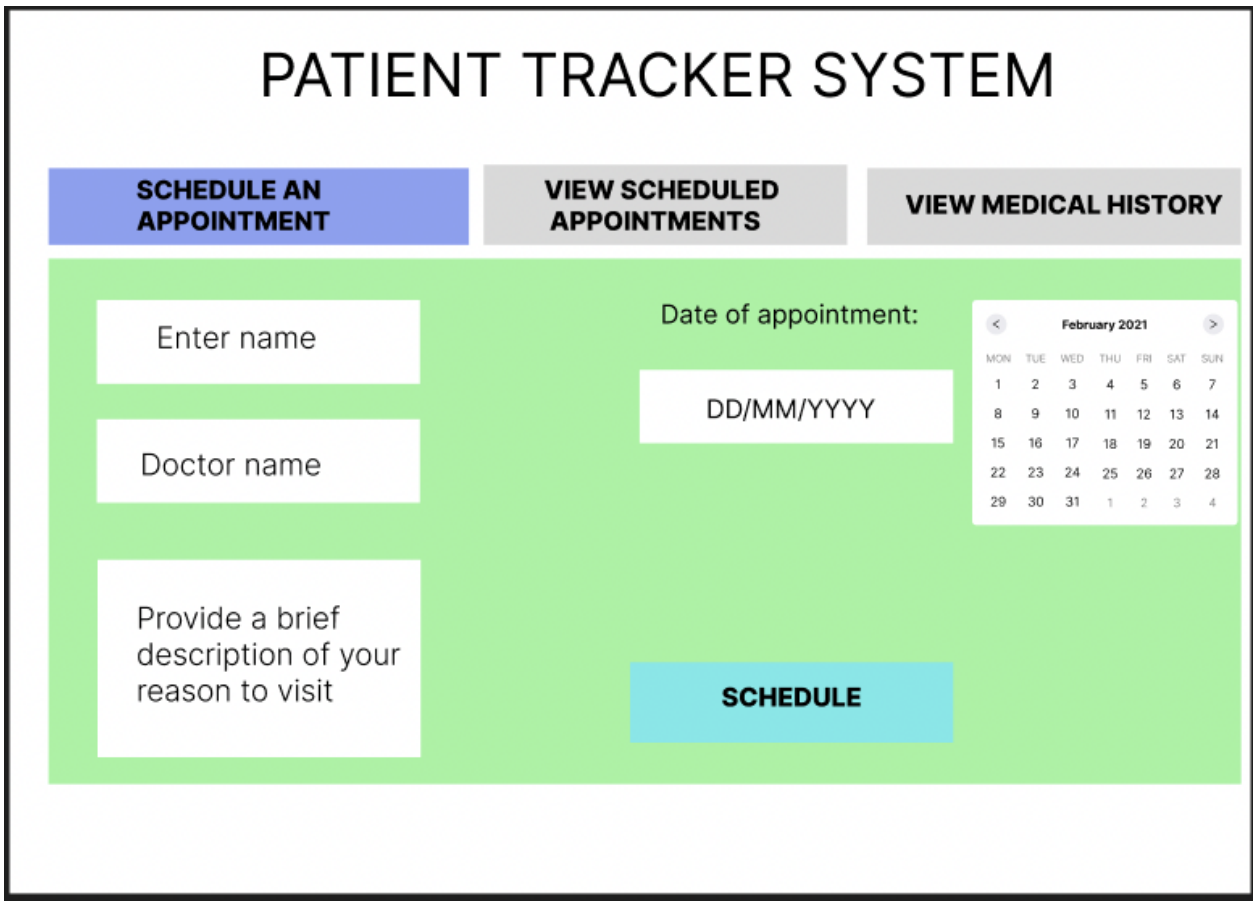
Login Page:



Doctor Landing Page:

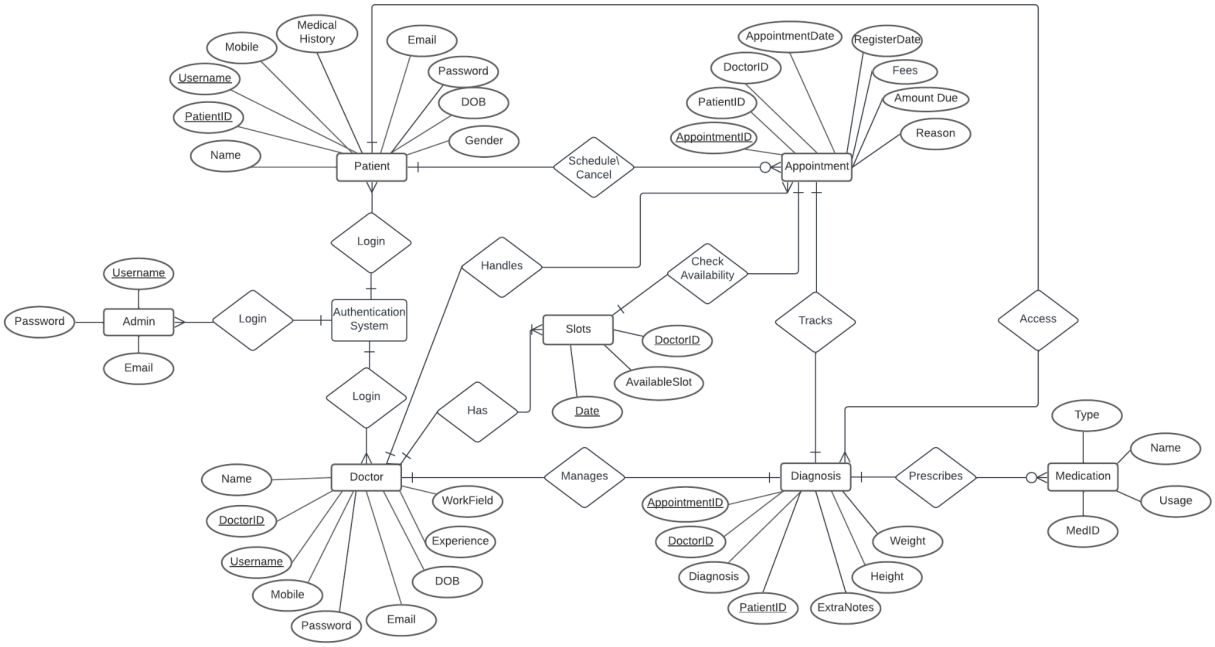


Patient Landing Page:



2.4. Data Model

Patient Tracker System Database ER Diagram:



3. Implementation

This project will be following an MVC architecture. The model will be interacting with the database and handling the data logic. The View will be the user interface presented to the client ie, medical professionals like doctors, nurses and administrator staff as well as patients. The controller will be the one interacting between the model and the view and fetching the requests.

We will be following the sprint-wise Agile methodology for this project. The development for this project will be divided into multiple short processes called sprints. Small bursts of development will be done, tested and feedback taken in order to improve the next sprint development. There will be week-wise increments made to the code along with a peer code review which emphasizes continuous code improvement.

The coding best practices this project aims to achieve are:

- Planning and preparing the code beforehand, so as to ensure efficient method and class creation
- Reduce redundant and unnecessary code statements
- Practice version control using Git
- Writing comments in the code for better understandability
- Using meaningful variable names and camel case for naming conventions
- Creating unit test cases for the methods
- Modularity by dividing the code into easy to read methods

The potential challenges and vulnerabilities that we can face during the development of this project are:

- The project development may not be able to be completed
- The project may not satisfy the requirements of the user
- The technology used may not be sufficient to code some features of the project
- The database could be corrupted
- The code of the project may not be backed up and be lost due to failure
- The system fails the unit tests

3.1. Security and Risks

Description of the potential threats and vulnerabilities

In the Patient Tracker System, the most important information to be safeguarded is the sensitive medical information of the patients. Thus, a variety of the attacks will be targeted to this information. Such threats and attacks include data breaches, exposure of private patient medical history and unauthorized access to the system. Some other potential vulnerabilities in the project are:

- Unencrypted Data: If the confidential information is not encrypted during transmission and storage, it can be intercepted by eavesdroppers.

- Phishing attacks: A legitimate user may be misled into providing their login credentials to attackers which ultimately provides unauthorized access to hackers.
- Weak Access Control: If a user has not set a strong password, it may be accessed using brute force attack by the hackers.

Measures taken to prevent unauthorized data access

To countermeasure the above looming threats, the Patient Tracking system should implement a sturdy security framework. Some other mitigation measures that can be implemented are:

- Data encryption: We must ensure that the data is encrypted during transmission and storage.
- Frequent Security Inspections: We must conduct penetration tests to identify and deal with vulnerabilities.
- Phishing awareness: The users must be educated on the signs of a phishing attack so that they can avoid and report them.
- Access Control: We must implement strict access control methods including role-based access as well as strong password protection.

HIPAA Compliance

HIPPA stands for The Health Insurance Portability and Accountability Act which is a critical part of the legislation of the United States of America that overlooks the protection of a patient's medical records which is very confidential. It sets the national standards to protect the patient's medical records and their personal information. HIPAA is very relevant to the patient tracking system as this application deals with patients' medical records and HIPAA ensures that this information is confidential.

Steps and protocols followed to ensure the system's compliance with HIPAA.

- Access controls: In our application we will be implementing role based access so that confidential information about the patients remain confidential.
- Encryption : By encrypting the data while storing it in the database provides privacy to the patient's confidential information.
- Data backup & recovery : Regularly backup the data so that there is no data loss and provide robust data recovery.
- Auditing & monitoring : Implement auditing and monitoring the data every few days to detect any suspicious activity if any.

3. Work Plan

High-level Timeline:

Since we will be following the agile methodology, we have divided the timeline into sprints.

Sprint 0: Project kickoff - getting started

In this sprint our main focus is to define the requirements of the project to have a proper idea on what we are building. We need to create a software requirement specification that defines the scope of the project and the main objectives we aim to achieve.

We need to draw the ER diagrams and come up with the different use cases that we plan on developing. Also, come up with UI mockup screens that give a better idea on what we are planning to develop. The requirement document also covers the work plan of the entire project.

Sprint 1: Development Phase 1 - Building the core

In this sprint we plan on developing the initial prototype of the entire project, which includes the core functionality and test if it covers all the use cases that were originally planned. To set the database schema and implement it with some basic initial data for testing it.

As soon as the initial prototype is ready we also plan on writing test cases for it, so that we can make sure that everything is working as expected and catch any issues early in the development and keep the project on track.

Sprint 2: Development Backlog and Testing

In this sprint we focus on completing the remaining use case development, and fine tuning the initial prototype to the actual project. We need to make sure that our application is robust in nature and is able to handle any unexpected inputs.

Manual testing and writing test cases will be an ongoing process during this sprint, making sure that the application works fine and provides error handling.

Sprint 3: Wrapping it up

In this sprint we plan on making any final changes and perform all testing to make sure that our application works as expected. Then we will make the video required for the submission that demonstrates the working prototype of the patient tracking system and also make the final project report.

Front End Development : Dhruvi & Devyani

Back End Development : Tejaswini & Shrutiya

Database Maintenance & Creation : Dhruvi & Shrutiya

Debugging and Testing : Devyani & Tejaswini