NIT-3003

Secure Medical Records Portal-

(MediAccessHub)

Project Proposal: Opening Section

Group Members:

Member 1(Admin Portal): Md Mahmudul Hasan Saif

Member 1 ID: s8089637

Member 2 (Monitoring & Reports): Abyead Faisal Arnob

Member 2 ID: s8098828

Member 3(Doctor Portal): Tashfiqul Prodhan

Member 3 ID: s8116509

Member 4(Patient Portal): Prateek Maharjan

Member 4 ID: s8113219

Introduction

General Background (Prateek):

In this modern world, healthcare has become more digitalized. Digital health is transforming how care is delivered around the world (World Health Organization, 2021). It has many important factors. One of them is how medical records are stored and shared. In past times, the records were written and stored on paper using different systems, which was not very convenient since it made it hard for doctors and patients to find or share important information quickly. Now, as the world is changing, it has been apparent that there is a strong need for a system that is secure, easy to use, and allows people to manage medical records online.
Here we bring in our **MediAccessHub** which is a web-based portal that helps solve this problem. This platform helps doctors to update and manage their patient records without risking mixing up patients, dates etc. Also, patients could now view their health history and request appointments without worrying about missing appointments and forgetting dates, and admins manage users and system activity. The system we are to create is secure, with access controls that make sure each user can only see what they are allowed to.

Market Analysis (Tashfiqul):

We can see there is a growing need for these kinds of solutions as they decrease errors and help save lives in that area. The electronic health record market was valued at over $29 billion in 2023 and is expected to grow to $47.6 billion by 2030 (Fortune Business Insights, 2024). Recently, a lot of hospitals and medical centers already use digital systems, but some smaller clinics and private practices often don’t have access to simple and secure tools. MediAccessHub is designed to fill this gap.

Competitor Analysis (Mahmudul Hasan):

There are a lot of digital healthcare sites like my health record, practiceHub and Doxy.me, but each of them comes with a lot of limitations. These platforms assist healthcare providers and patients in many ways but fall short in terms of usability and functionality.

My Health Record allows users to link their medical history with Medicare. However, the system can be difficult for some users to navigate. PracticeHub is effective in managing clinic operations, compliance, and documentation. But it does not provide patients with direct access to their medical records. Doxy.me offers secure video consultations but lacks the capability to store or manage medical records for long-term use.

MediAccessHub is designed to overcome these limitations by offering a user-friendly, role-based system that allows doctors, patients and admins to manage, view and interact with medical data in a secure, accessible and integrated environment.

Project Aims and Unique Selling Proposition (Arnob):

Why is MediAccessHub better? We are different because our application is built around four user roles: doctors, patients, admins, and a system reporting module. All parts are designed and built by a different team member, including the front end, back end, database, and security. This makes the system well organized and easier to manage (Sommerville, 2016). By focusing on clear design and strong security, MediAccessHub offers a simple and modern way to handle medical records safely online.

Summary: our system offers these following things:

Access based on roles and secure system (Saif): There is a secure login system for every role assigned. Patients can view their medical history and make appointments; Doctors can update every patient's records. Admins look after the whole system and update the database. And the monitoring team monitors the system and makes reports.

Centralized Medical record (Tashfiqul): Doctor can manage and update patient data following the system structure and centralized platform. This highly reduces the risk of errors such as record duplication and ensuring data integrity and reliability.

Patient Empowerment and Accessibility (Parteek): Patients have full access to their medical history and their appointments, which are secure. This interface helps patients stay updated with their previous and ongoing records.

Administrator Control and Oversight (Saif): Admins manage user activities, update the system, and keep the system running smoothly.

Intergrated System Reporting Module (Arnob): Authorized users and admins monitor and report about the usage, appointment and records management enabling decision making which are driven by the data. The reports view current patterns of usage and also view traffic on the system.

Functional Requirements for MediAccessHub

Patient Portal – Functional Requirement (Prateek)

The main function of the patient portal is to be able to register into the portal, view their medical records while being able to download the data, request appointments with doctors and manage/request change in the appointment. The patients are allowed to login or register using their credentials i.e., Name, Email, contact no, Password and confirm password (For registration) and Email or username and password (For login).

The patient's dashboard mainly contains greetings to the patients with two main features, like Registering oneself to the MediAccessHub portal, Medical Records with option to download data and Appointment Request and change the appointment details.

1. **Register a New Patient:**

   - **Front-end:** The first step in registering an account is to accesses the registration page and fill out a form with fields like name, date of birth, email, phone, and password. Then the patient clicks on "Register" where all inputs are validated for format. The validated data is sent via POST request to the server.

   - **Server-end:** On the server end, the server receives the registration data. It checks if the email already exists in the database. And If the data is valid, the server encrypts the password and stores the user record. Finally, the server sends a success or failure response to the front-end.

   - **Database:**

Patient_ID	Full Name	Email	Phone	Password_hash	DOB
123	XX	XX	XX	XX	XX

2. View Medical Records:

- Front-end

  Here, our patients are shown a list of their previous reports and their uploaded files in a table form. Every row displays report title, date, description and a Download file button. Everything displayed is accessed from the server when the page loads.

- Server-end

  As the patient logs in the dashboard, the server identifies the patient by their e-mail and login credentials, then it displays their medical history and records.

- Database Tables

Record_id	Full Name	Patient_email	Doctor_email	Title	Description	Date
369	XX XXX	XX	XX	Blood test	All normal	03-02-2021

3. Request Appointment:

- Front-end

  Users are allowed to access a form where they can Select a doctor, choose a date and time from calendar, write up their reason for appointment and click to submit button.

- Server-end

  Here, the server is going to receive the data then verify the patient's identity. Then it checks if the requested doctor exists and if the time slot is valid. If it is valid, then the appointment is kept in the database with a "Pending" status. An email or message is sent to the patient after confirmation.

Then the form is sent to the server using a secure post request.

- Database Tables

Appointment_ ID	Full Name	Patient_ email	Doctor_email	Appointment_d ate	Reason	Status
246	XX XXX	XX XXX	XX XXX	10-10-2025; 10AM	Headache	Pending

Doctor Portal Component (Tashfiqul)

1. Introduction

Key objectives of this system include providing secure login and role-based access for patients, doctors, and administrators, and to enable doctors to manage availability, medical records, and appointments for their assigned patients. The system is designed around distinct user roles to ensure data privacy and ease of

use. Its unique value lies in a modular development approach, allowing for a well-organized and highly secure platform that fills a gap in the market for smaller clinics and private practices

2. Functional Requirements

1) Log in (Doctor - Tashfiqul)

- **Function:** The system shall allow a doctor to log in using secure credentials.
- **Front-end:** A login form will be displayed with fields for the doctor's email and password. The user enters their credentials and clicks a "Login" button.
- **Server-end:** The server receives the credentials, hashes the provided password, and compares it against the stored hash in the database for the given email. Access is granted upon a match, and the user is redirected to the Doctor Dashboard.
- **Database:** Interacts with the Users table to verify credentials.

UserID	Email	FullName	HashedPassword	Role
101	dr.smith@email.com	John Smith	a9b1c7d3...	Doctor
201	p.jones@email.com	Peter Jones	f4d8a25e...	Patient

2) View & manage records (Doctor - Tashfiqul)

- **Function:** The system shall allow a Doctor to view, create, edit, and delete medical records for their assigned patients.
- **Front-end:** On a selected patient's record page, the system displays their medical history. Buttons for "Add New Record", "Edit", and "Delete" are available. A form is used for data entry.
- **Server-end:** Receives requests (POST, PUT, DELETE), validates data, confirms the doctor's authorization for the patient, and performs the corresponding database operation.
- **Database:** Interacts with the MedicalRecords table.

RecordID	PatientID	DoctorID	Diagnosis	Notes	DateCreated

5001	201	101	Common Cold	Advised rest...	2025-06-10

3) Update availability (Doctor - Tashfiqul)

- **Function:** The system shall allow a Doctor to set and update their available hours for appointments.
- **Front-end:** Displays a weekly calendar on a "My Availability" page. The doctor can click on a day to open a form to input their start and end times or mark the day as unavailable. Clicking "Save" sends the data to the server.
- **Server-end:** Receives availability data, validates it (e.g., start time is before end time), and saves or updates the entries for the specific doctor and date in the database.
- **Database:** Interacts with the DoctorAvailability table.

AvailabilityID	DoctorID	AvailableDate	StartTime	EndTime
1	101	2025-06-16	9:00:00	17:00:00

4) View scheduled appointments (Doctor - Tashfiqul)

- **Function:** The system shall allow a doctor to view their upcoming scheduled appointments.
- **Front-end:** Displays a weekly or monthly calendar view showing all the doctor's scheduled appointments. Each entry shows the patient's name and appointment time.
- **Server-end:** Queries the database for all appointments linked to the doctor's ID and provides the data to the front-end calendar component.
- **Database:** Interacts with the Appointments table.

AppointmentID	PatientID	DoctorID	AppointmentDate	Status
801	202	101	2025-06-18 10:00	Confirmed

Admin Portal – Functional Requirements (Saif)

The admin portal provides admin with a secure login and dashboard interface. This portal allows the admin to view the full list of user accounts and status. The admin should have access to manage and view Doctor and patient profiles for database updates.

And the Admin portal makes sure there is admin only access to sensitive data.

1. **Admin Login**

    1.1. **Front-End:** Admin Login Page:

        1.1.1. Admin Email

        1.1.2. Admin Password

        1.1.3. Sign in button for admin

        1.1.4. Forgot password? Button for admin

    1.2. Back-End:

        1.2.1. POST/admin/login

        1.2.2. Verify admin login credentials

        1.2.3. Generate session for admin

        1.2.4. Show admin dashboard upon successful login

    1.3. Database:

Admin ID	Email	Password
1	xxx	xxx

2. **Manage Doctor Registrations**

    2.1. **Front-End:** Admin dashboard will show all the registration request of doctors, and the table includes:

2.1.1.Doctor Name

2.1.2.Email

2.1.3.Specialty

2.1.4.Status

2.1.5.Approve button

2.1.6.Reject button

2.2.Back-End:

2.2.1.GET /admin/Doctors?Status=pending – view all pending reqeust

2.2.2.PUT /admin/doctor/{id}/approve - Set status to approve

2.2.3.PUT / admin/doctor/{id}/reject - Set status to rejected

2.3.Database:

Doctor Name	Email	Specialty	Status	Is Active
xx	xx	xx	xx	xx

3. **Manage User Profiles**

3.1.**Front-End:** Admin Dashboard will show user management tab for both doctor and patient

3.1.1.Doctor: ID, Name, Specialty, Email, Status [Edit][Deactivate]

3.1.2.Patient: ID, Name, Email, Gender, Status [Edit][Deactivate]

3.2.Back-End:

3.2.1.GET /admin/user?Type=doctor | patient

3.2.2.PUT /admin/user/{ID} - for updating info or to check is active

3.2.3.Admin cannot delete user accounts only set status to inactive

3.3.Database:

Doctor ID	Name	Specialty	Email	Status
xx	xx	xx	xx	xx
Patient ID	Name	Email	Gender	Status
xx	xx	xx	xx	xx

4. **Create and Manage Doctor Schedule**

4.1. **Front-End:**

4.1.1.Interface showing doctor availability

4.1.2.List of available slots

4.1.3.Admin selects slots to create schedule

4.1.4.Admin click publish schedule

4.1.5.UI shows confirmation and changes

4.2.Back-End:

4.2.1.GET /admin/availability?Status=approved

4.2.2.POST /admin/schedule - Publish schedule

4.2.3.PUT /admin/schedule/{ID} - Update status for appointment marked as booked

4.3.Database:

Availability ID	Doctor ID	Date	Time slot	Status
xx	xx	xx/xx/xx	xx	x

xx	xx	xx/xx/xx	xx	x

5. **Approve Doctor availability changes mid-week**

    5.1. **Front-End:** Admin will be notified for pending availability changes, and the table will include the following

        5.1.1. Doctor Name

        5.1.2. Date

        5.1.3. Time

        5.1.4. Change Time

        5.1.5. [Approve][Reject] Button

    5.2. Back-End:

        5.2.1. GET /admin/availability?status=pending

        5.2.2. PUT /admin/availability/{id} - set status = approve/rejected

    5.3. Database:

Doctor	Date	Time	Change Time
xx	xx	xx	xx

6. **Approve appointment change reqeusts**

    6.1. **Front-End:** Admin will have appoitment changes tab where admin can see pending appointment change requests adn the table will include:

        6.1.1. Appointment ID

        6.1.2. Doctor

        6.1.3. Patient

6.1.4.Old Time

6.1.5.Requested Time

6.1.6.Modified by

6.1.7.[Approve][Reject] button

6.2.Back-End:

6.2.1.GET /admin/appointment?status=change_requested

6.2.2.PUT /admin/appointment/{id} - update status to approve or rejected

6.3.Database:

Appointment ID	Doctor	Patient	Old Time	Requested Time	Modified by
xx	xx	xx	xx	xx	xx

7. View Appointment Records and schedule:

7.1.**Front-End:** Admin dashboard will have a view records page which can be filtered by;

7.1.1.Doctor Name

7.1.2.Patient Name

7.1.3.Date Range

7.1.4.Status

7.2.Back-End:

7.2.1.GET /admin/appointment

7.2.2.GET /admin/schedules

7.3.Database:

Doctor Name	Patient Name	Date Range	Status
xx	xx	xx	xx

Monitoring & Reports (Arnob)

The main task of this module is to display a dashboard which will show the number of logins, requests per doctor, and many other current trends. This module also formats the data in downloadable forms.

- **Viewing Dashboard:**
    1. Front-end: The dashboard must consist of a view of total logins per day and will have options to view log in details on a weekly and monthly basis. It will also view the number of daily appointments requests per doctor.
    2. Back-end: Save number of logins. The database shall consist of different folders for each doctor and must save the number of appointment requests made for each doctor to their own respective folders.
- **Exporting records:**
    1. Front-end: After clicking the export option, a list of patient records will be visible to the doctor. For the patients, only their records will be visible and can be exported. For the doctors, they can only access the records of the patients they treated. They will be able to choose a specific patient and export their records.
    2. Back-end: Look through the database for related data of patients and summaries in a list. Fetch through the database for the specific patient data and arrange them on another table. When export is required for the patient's data, it gathers the data and formats it in a PDF/CSV format and returns it to download.

3. Patient record list table as below:

Patient ID	Full Name	Age	Gender	Total Appointments	Assigned Doctor	Last Appointment
xxx	xxx	xx	xx	xx	xxxxxx	xx-xx-xxxx

4. Exportable data table below:

Patient ID	Date	Diagnosis	Prescription	Comments	Doctor
xxx	xx-xx-xxxx	xxxx	xxxx	xxxx	xxxx

Non-Functional Requirements:

Patient Portal - (Prateek):

Function 1: Register a new user

1. Security:
The emails and passwords must me transmitted securely using encryption protocols such as HTTPS. All the passwords must be hashed before storing in database which ensures authorized access.

2. Response Time:
The system should ensure smooth experience for the user and it must reduce chances of user drop-off.

Function 2: View Medical Records

1. Privacy and security:
For security and privacy reasons, only authenticated patients are allowed to access the medical records. The system should be built in such a way that no one other than the patient can access medical records.

2. Performance:
Even when there are multiple users, the users should be able to access their medical records within a certain time frame so that no inconvenience is caused.

Function 3: Request Appointment

1. Easy to Use:
As the user can vary in age or knowledge level, the appointment form should be easy and simple to understand and complete. This will help to minimize any errors for wrong input. This will also save time for the patients.

2. Data Checking:
Any invalid or incomplete form request should not be submitted. The system stops users from using dates that have already passed or the doctors that are not available at the time. The system should only allow users to request time that doctors are available.

Doctor's Portal – (Tashfiqul):

1. Log in

- **Security:** Passwords must be securely hashed before being stored and during comparison to protect user data and ensure integrity.
- **Performance:** Login verification should be processed quickly, ideally within a few seconds, even under standard load.

2. View & manage records

- **Performance:** Record access and updates should be fast, loading within a few seconds to ensure an efficient clinical workflow.
- **Security:** The system must enforce strict access control, ensuring doctors can only see and manage records for patients specifically assigned to them.

3. Update availability

- **Reliability:** When a doctor saves their availability, the system must save it correctly without errors to ensure patients see accurate appointment slots.
- **Usability:** The calendar interface for setting availability should be intuitive and easy for doctors to use.

4. View scheduled appointments

- **Usability:** The appointment calendar should be clear, well-organized, and straightforward for the doctor to read.
- **Performance:** The schedule must load quickly to avoid delaying the doctor.

ADMIN- (Saif):

1.Admin access and role control: The system will restrict any unauthorized user to access in admin role; they will be denied access to the admin portal and receive an error message.

2.Appoinment Management consistency: The system must make sure any action done by admin is immediately updated across all connected modules and these actions could be status change, doctor patient linking and etc.

3.Interface Performance: the system must operate under standard conditions to make sure whenever viewing doctor schedules, appointments requests, user list

the system must load all data withing 3 seconds and provide admin with smooth use experience.

4. Real Time synchronization: Any update done by admin must be updated across the entire system withing standard timing ensuring real time accuracy for doctors and patients.

5. System Scalability: The Admin portal should be able to handle a high volume of data without any problems with performance.

6. Security: encryption must be used by HTTPs protocol in admin portal to all the data and data validation must be enforced.

Monitoring & Reports (Arnob):

1. **Security:**
   - Access to the patients' records and monitoring dashboards must be restricted.
   - Patient data must not be visible to unauthorized personnel in any error message.
2. **Performance:**
   - The dashboard should run without any crashes or errors even with a large number of users.
   - The system must support real-time or near-real-time updates.
   - The export process should not take more than 5 seconds.
   - If the export is failed, it must show a clear error message and should have an option to try again.
3. **Usability:**
   - The dashboard UI should be easy to navigate for the users, considering the majority of the patients can be elderly people.
   - Patient records must be in a readable and sorted format for viewing and exporting.
4. **Maintainability:**
   - All the code must follow a clean and updatable format so that any updates or edits will be easy to make.

Use Case for MediAccessHub

Use Case Descriptipn

Use Case Description for Patient (Prateek)

1. Registering a new user

Use Case Name	Patient Registration and Login
Participating Actor	Patient (Prateek)
Goal	The goal of this Use case is to allow user to create an account and login securely to enter in the system
Flow of Events	1. The user goes to MediAccessHub portal. 2. A form with the patient's details is filled in and completed. 3. The system then checks for invalid data and after completion, a new account is created. 4. The user is then sent to login page where they can enter their credentials and log into the account. 5. The system verifies the input; • If incorrect: asks user to retry. • If correct: allows user to the dashboard.
Entry Condition	• The patient is required to visit the portal. • The registration form is filled.
Exit Condition	• A new account has been created. • Patients can now successfully login. • If there are any errors, the user is alerted, and resubmission is required.

2. View and download Medical Records

Use Case Name	Access Medical records and download reports.
Participating Actor	Patient (Prateek)
Goal	The goal here is to allow users to access their medical records and optionally download the required data.
Flow of Events	1. The user logs into the portal. 2. In dashboard, user clicks on “View Medical Record” 3. The system gets the required record from the database. 4. The records are then displayed in a tabular form. 5. Each row contains download button at the end 6. The user can download medical records as required.
Entry Condition	• The patient is successfully logged in.
Exit Condition	• The records are displayed properly. • Downloading is easily available. • Errors are reported for missing any files or access denial.

3. Request and Modify Appointment

Use Case Name	Request and Manage Appointment
Participating Actor	Patient (Prateek)
Goal	The goal of this use case is to allow the user to request, view and change appointments with doctor.
Flow of Events	1. The user logs into the portal. 2. User clicks to “Appointment” page 3. Patient can select to “Request Appointment” or “Upcoming Appointment” 4. In “Request Appointment” page, user can select doctor, date and time then submit the form. 5. Then the system checks availability and stores the form as pending. 6. After confirming, it is displayed to the user as a Confirmed Appointment. 7. Patient can also select “Upcoming Appointment” 8. Here the system displays all Confirmed and Pending Confirmation. 9. User also has option to click on “Request change” 10. Here, the system processes the update and sends confirmation of new Appointment.
Entry Condition	• The login credentials are correct. • Appointment form is submitted.
Exit Condition	• Appointments are successfully created. • User receives confirmation. • Stop the process if any error occurs.

Category	Description
Use Case Name	View and Manage Records
Participating Actor	Doctor (Tashfiqul)
Goal	To allow a doctor to create, edit, and delete medical records for their assigned patients.
Flow of event	1. The Doctor selects an assigned patient to view their medical history, and the System displays the records page. The Doctor then performs a management action: • Create: Fills and submits a new record form. The System validates and saves it. • Edit: Selects and modifies an existing record. The System validates and updates it. • Delete: Selects a record and confirms its removal. The System deletes it.. Finally, the System updates the display to reflect the successful action.
Entry Condition	- The Doctor must be successfully logged into the MediAccessHub system. - The Doctor must have selected an assigned patient whose records are to be managed.
Exit Condition	- If successful: A new medical record is created, an existing one is updated, or a record is deleted, with changes reflected in the patient's history. An audit log is recorded. - OR: An error message is displayed if the action fails.
Category	Description
Use Case Name	Update availability
Participating Actor	Doctor (Tashfiqul)
Goal	To allow a doctor to set and update their available hours for patient appointments.
Flow of event	1. The Doctor navigates to the "My Availability" page. 2. The System displays a weekly calendar interface showing the Doctor's current schedule. 3. The Doctor selects a day or time block, then inputs their start and end times or marks the period as unavailable. 4. The Doctor clicks "Save" to submit the changes. 5. The System validates the new availability data (e.g., ensures start time is before end time). 6. The System saves the updated availability to the database.> 7. The System displays a confirmation message to the Doctor.
Entry Condition	The Doctor must be successfully logged into the MediAccessHub system.
Exit Condition	The Doctor's availability schedule is successfully updated in the database for the specified dates. The changes are available for the patient appointment system to use. If the update fails, an error message is displayed.

Category	Description
Use Case Name	Update availability
Participating Actor	Doctor (Tashfiqul)
Goal	To allow a doctor to set and update their available hours for patient appointments.
Flow of event	1. The Doctor navigates to the "My Availability" page. 2. The System displays a weekly calendar interface showing the Doctor's current schedule. 3. The Doctor selects a day or time block, then inputs their start and end times or marks the period as unavailable. 4. The Doctor clicks "Save" to submit the changes. 5. The System validates the new availability data (e.g., ensures start time is before end time) 6. The System saves the updated availability to the database. 7. The System displays a confirmation message to the Doctor.
Entry Condition	The Doctor must be successfully logged into the MediAccessHub system.
Exit Condition	The Doctor's availability schedule is successfully updated in the database for the specified dates. The changes are available for the patient appointment system to use. If the update fails, an error message is displayed.

Category	Description
Use Case Name	View scheduled appointments
Participating Actor	Doctor (Tashfiqul)
Goal	To allow a doctor to view their upcoming confirmed appointments in an organized format, such as a calendar.
Flow of event	1. The Doctor navigates to the "Appointments" or "My Schedule" page within the portal. 2. The System sends a request to the server to fetch the doctor's appointment data. 3. The server queries the database for all appointments linked to the logged-in doctor's ID. 4. The System displays the retrieved appointments in a weekly or monthly calendar view. 5. Each entry on the calendar clearly shows the patient's name and the appointment time.
Entry Condition	The Doctor must be successfully logged into the MediAccessHub system.
Exit Condition	The Doctor has successfully viewed their schedule of upcoming appointments. No data in the system is changed by this action. If the schedule cannot be loaded, an error message is displayed.

Use case descriptions for Admin (Saif)

Admin login

Use Case Name	Admin Login
Participating Actor	Admin (SAIF)
Goal	Login with secure credential to admin dashboard.
Flow of Events	1. Admin visits the system Login 2. The system will display a login page requiring login credentials. 3. Admin will enter a valid username and password. 4. System verifies login information. 5. Admin is granted access to dashboard upon successful verification 6. Admin is granted access and dashboard is displayed.
Entry Condition	Admin access the portal through login page
Exit Condition	Admin successfully logs in after successful verification and dashboard is loaded.

Profile Management:

Use Case	Profile Management
Participating Actor	Admin (SAIF)
Goal	Manage user profile for both Doctor and Patient

Flow of Events	1. Admin access list of all registered doctors and patient 2. Admin can view or edit doctor and patient profile 3. Admin will approve/reject doctor registration 4. Admin updates user information as needed 5. Admin can active/inactive user profile 6. Admin can terminate doctor profile upon resigning
Entry Condition	Admin logged into the dashboard.
Exit Condition	Updates made to user profiles are saved in the system database.

Schedule Management

Use Case	Create/update/manage Schedule
Participating Actor	Admin (SAIF)
Goal	Manage appointment table, doctor availability and approve any updates to the system
Flow of event	1. Admin logs into the dashboard 2. Admin view updated Doctor availability and create schedule 3. Admin receives notifications of changes 4. Admin update timetable after an appointment is booked 5. Admin approve/reject doctor availability change mid-week 6. Admin approves/rejects any changes to appointments made by doctor or patient. 7. Admin reviews and take decision for any system update 8. System applies approved updates and reflects the changes 9. Notifications are sent to the users

Entry Condition	Admin is logged into the system and access create/update/manage schedule section
Exit Condition	All updates and changes are approved and applied and saved into the database

Admin View Record

Use Case	View Medical Records
Participating Actor	Admin (SAIF)
Goal	Access and view all medical records
Flow of Events	1. Admin logs into the system 2. Admin views medical record section 3. Admin views patient medical history 4. Admin view appointment history and upcoming appointment 5. Admin views other records such as prescriptions, test results, etc.
Entry Condition	Admin is logged in and open view record dashboard
Exit Condition	Medical records are displayed for admin viewing only

Use case Description for Monitor (Arnob)

Use Case Name	Create dashboard for doctor and admin view
Participating Actor	Monitor (Arnob)
Flow of event	**1.** Monitor logs into the system. 2. System loads the recent log in history and appointment activity. 3. Monitor views the dashboard with login, appointments stats. 4. Option to filter by date or user role. 5. Data is updated real-time on the dashboard.
Entry Condition	Monitor logs into the system successfully.

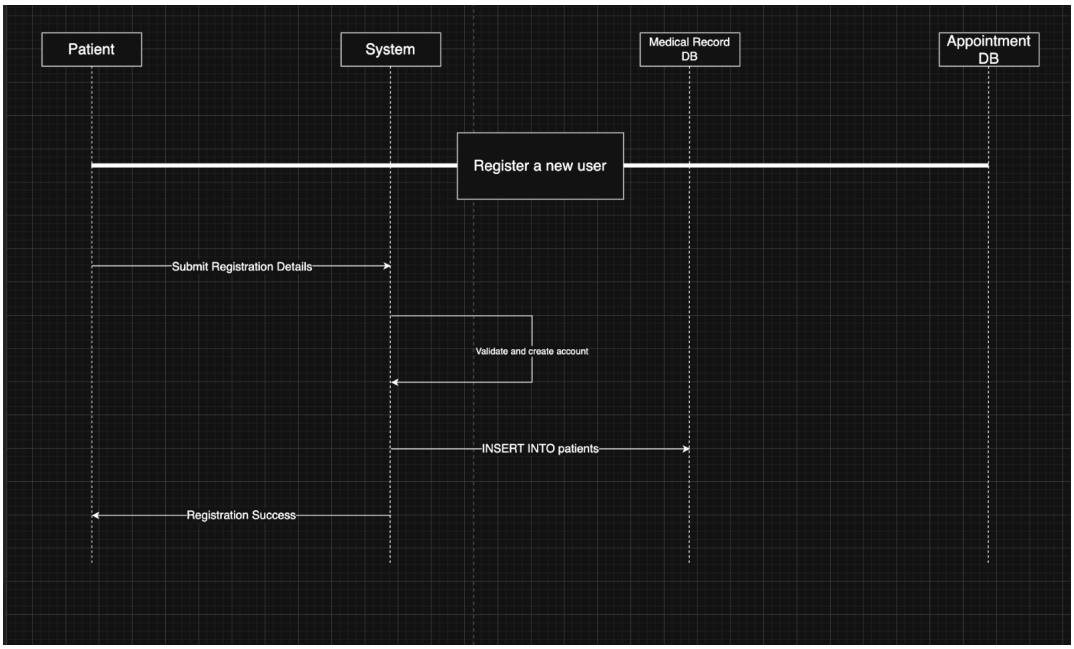
Exit Condition	Dashboard is loaded and the stats are displayed clearly. Filters are applied when needed.

Use Case Name	Arrange data in an exportable format
Participating Actor	Monitor (Arnob)
Flow of event	**1.** Monitor selects a patient from the dashboard list. 2. Medical records for the selected patient can are collected from the database. 3. Monitor clicks Export and gets the option to choose from PDF or CSV. 4. Data is arranged and formatted. 5. Exported file is generated and downloaded.
Entry Condition	Patient list and recird are visible
Exit Condition	Exported file is downloaded successfully or try again if not successfull.

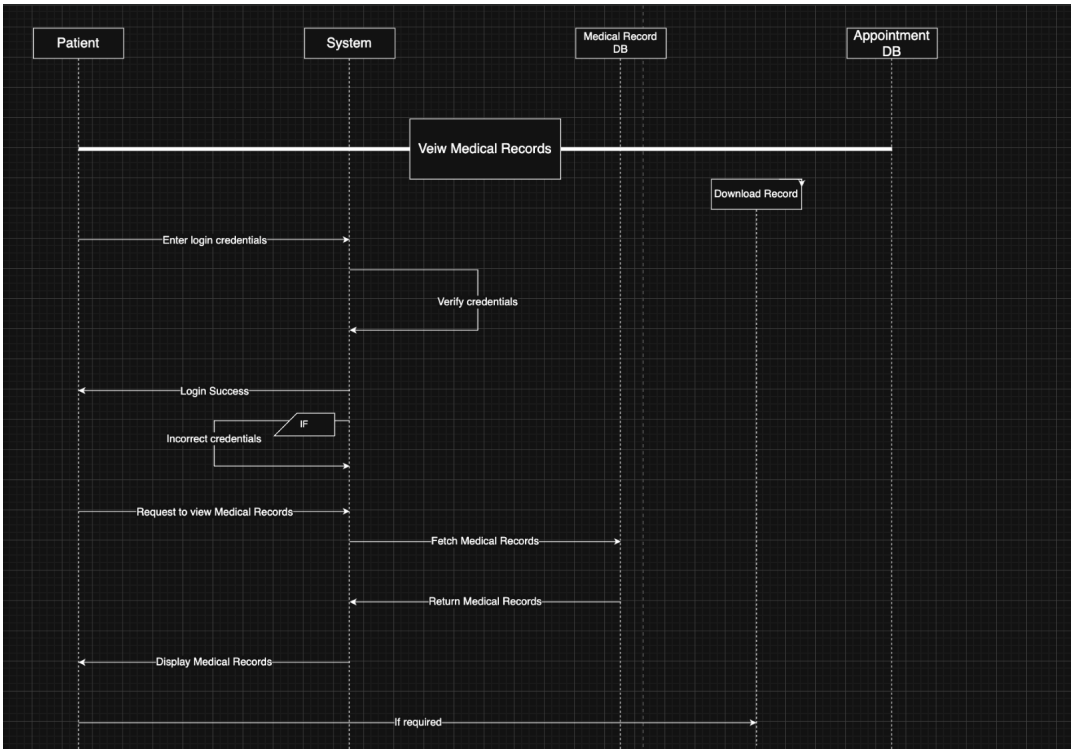
Sequence Diagram for MediAccessHub

Sequence Diagram for Patient(parteek)

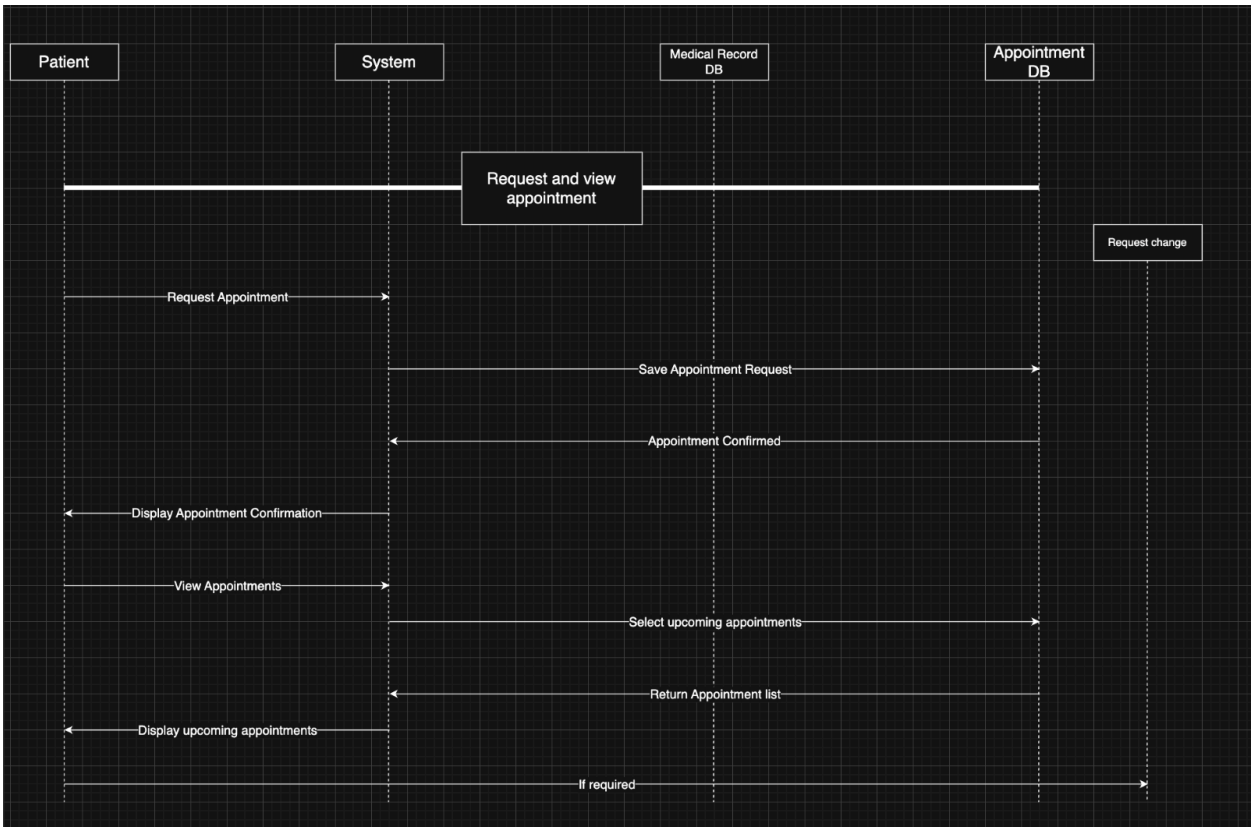
1. Registering a new patient/user



2. Viewing Medical records and downloading if required

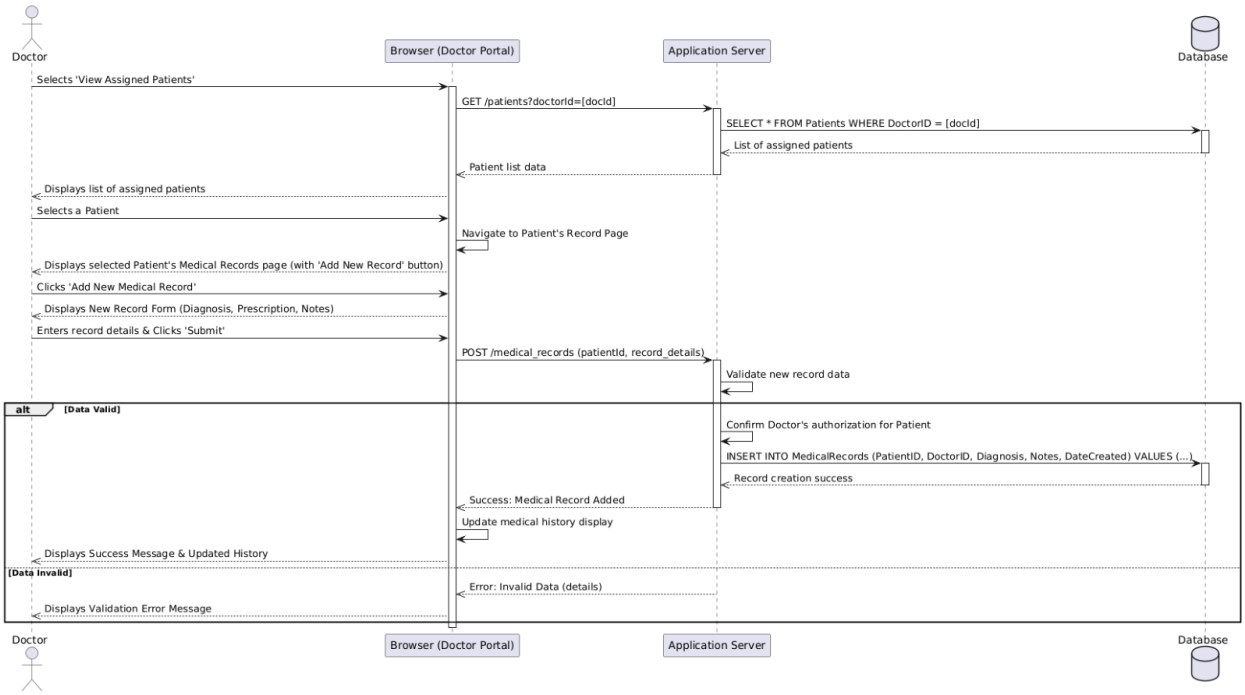


3. Making and viewing appointments

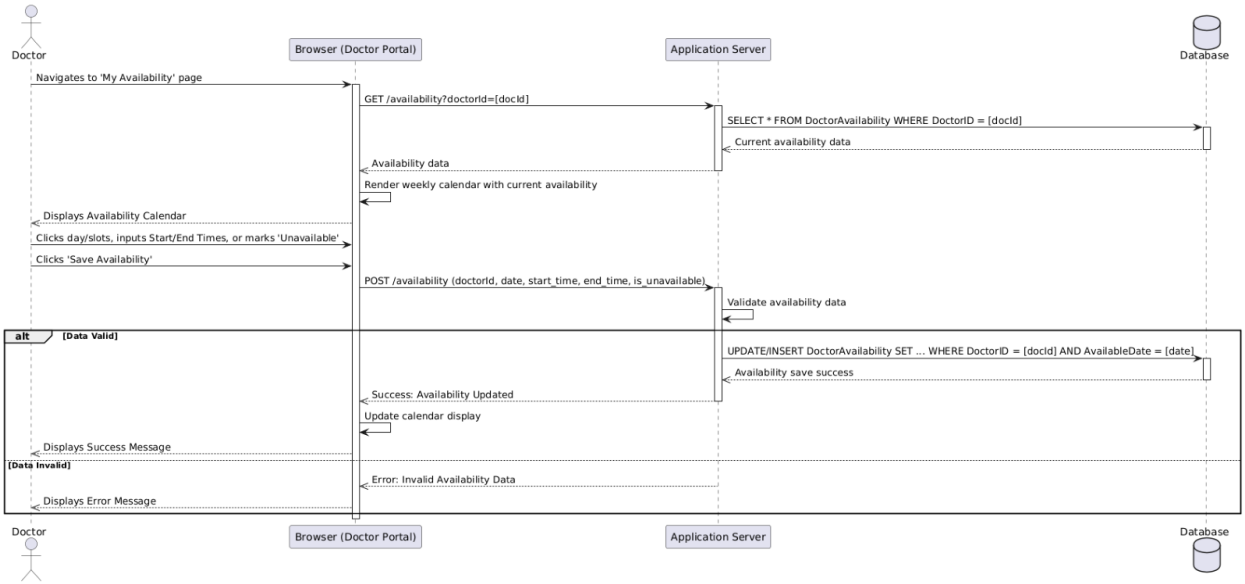


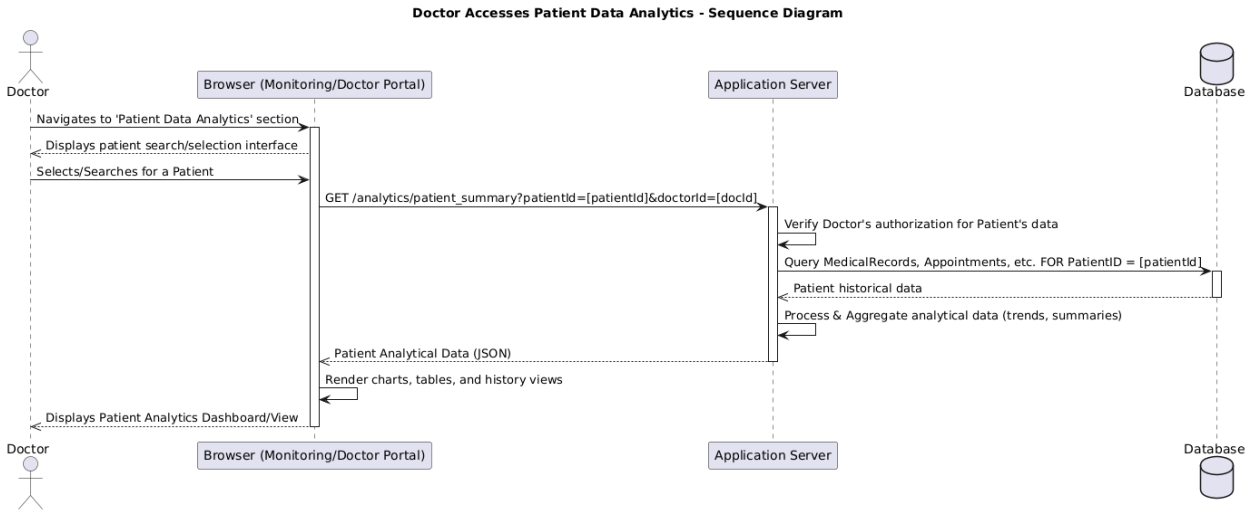
Sequence Diagram for Doctor:

Doctor Adds New Medical Record - Sequence Diagram

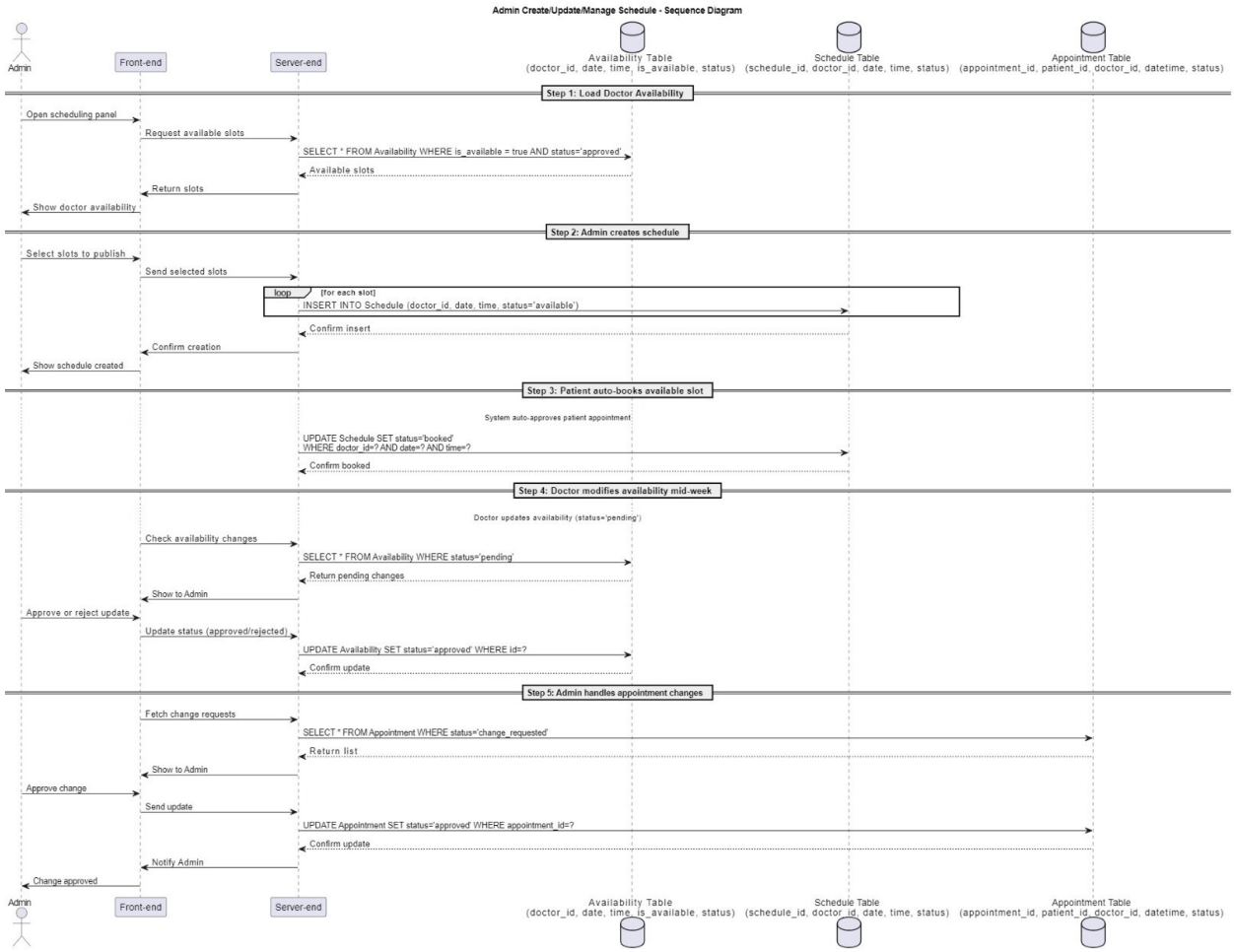


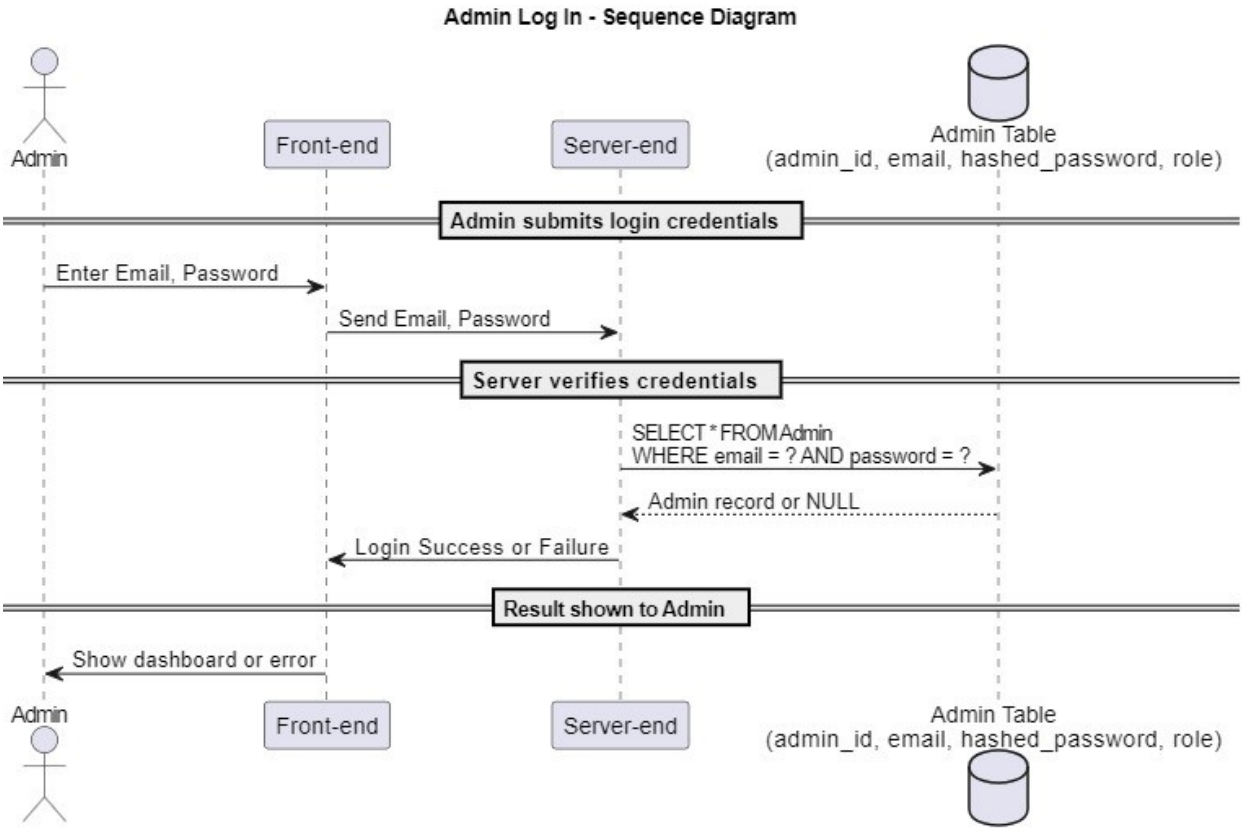
Doctor Sets Weekly Availability - Sequence Diagram

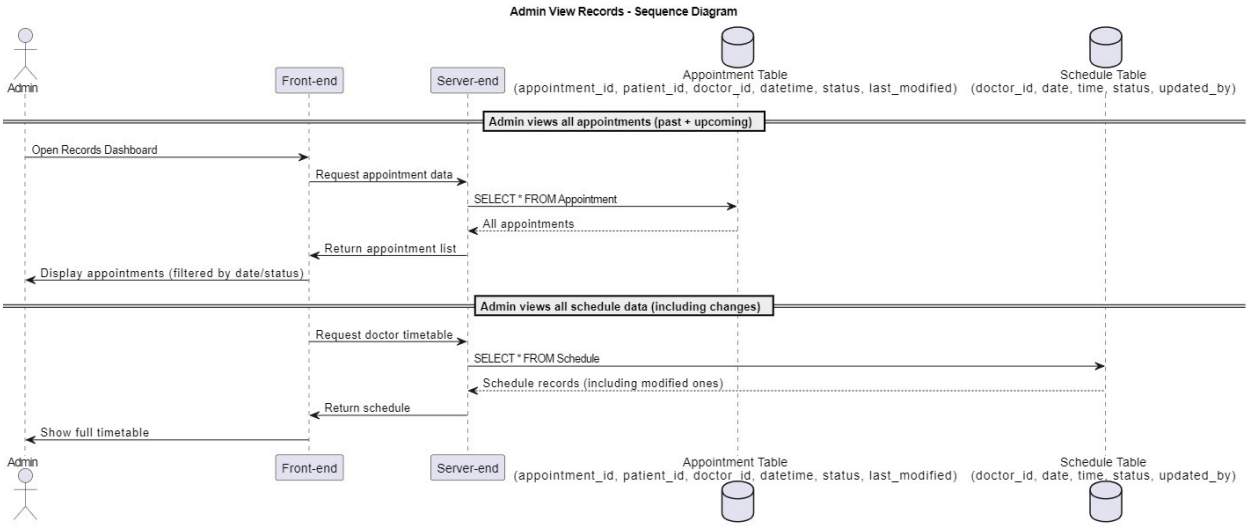


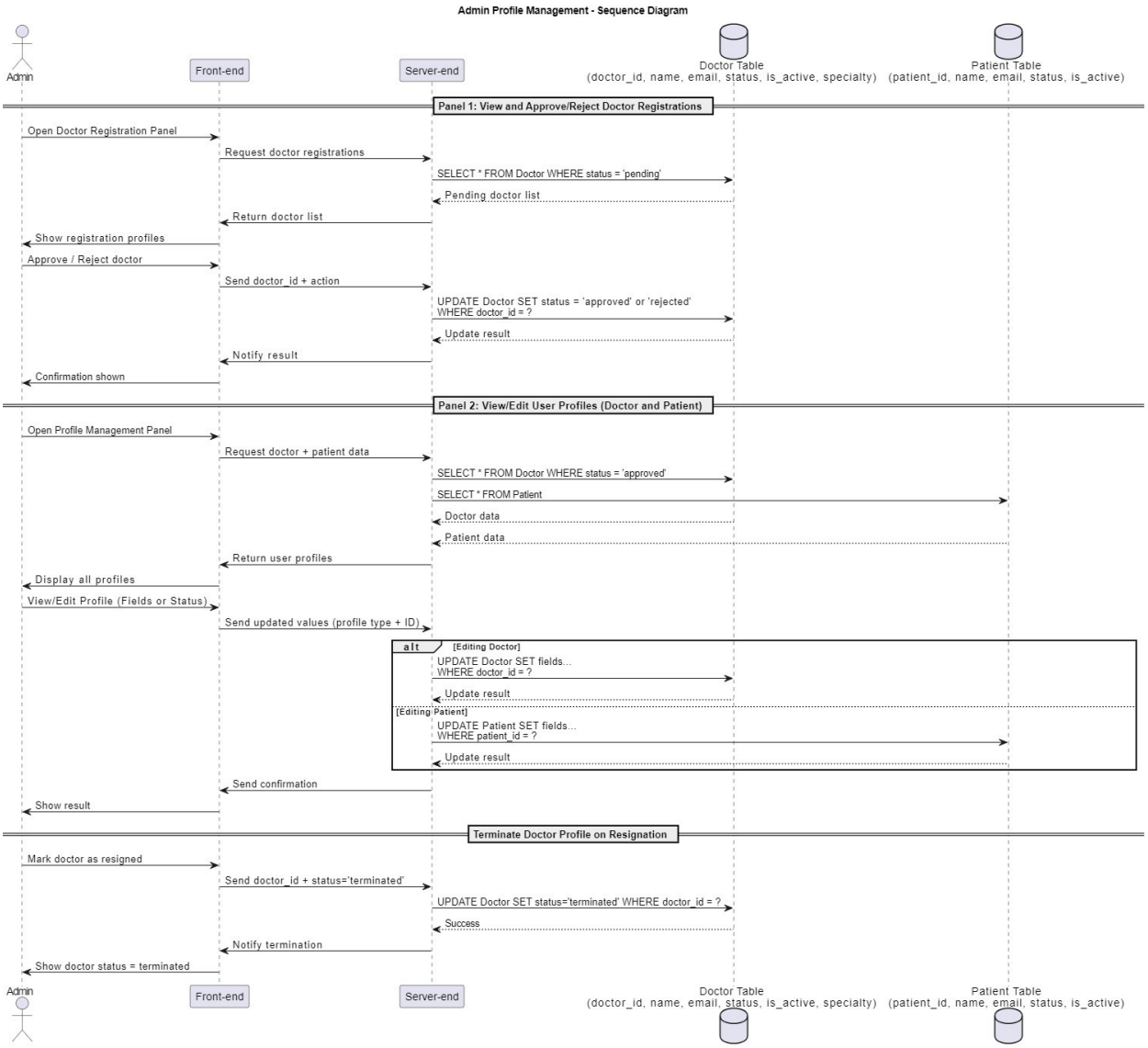


Sequence Diagram for Admin (Saif):

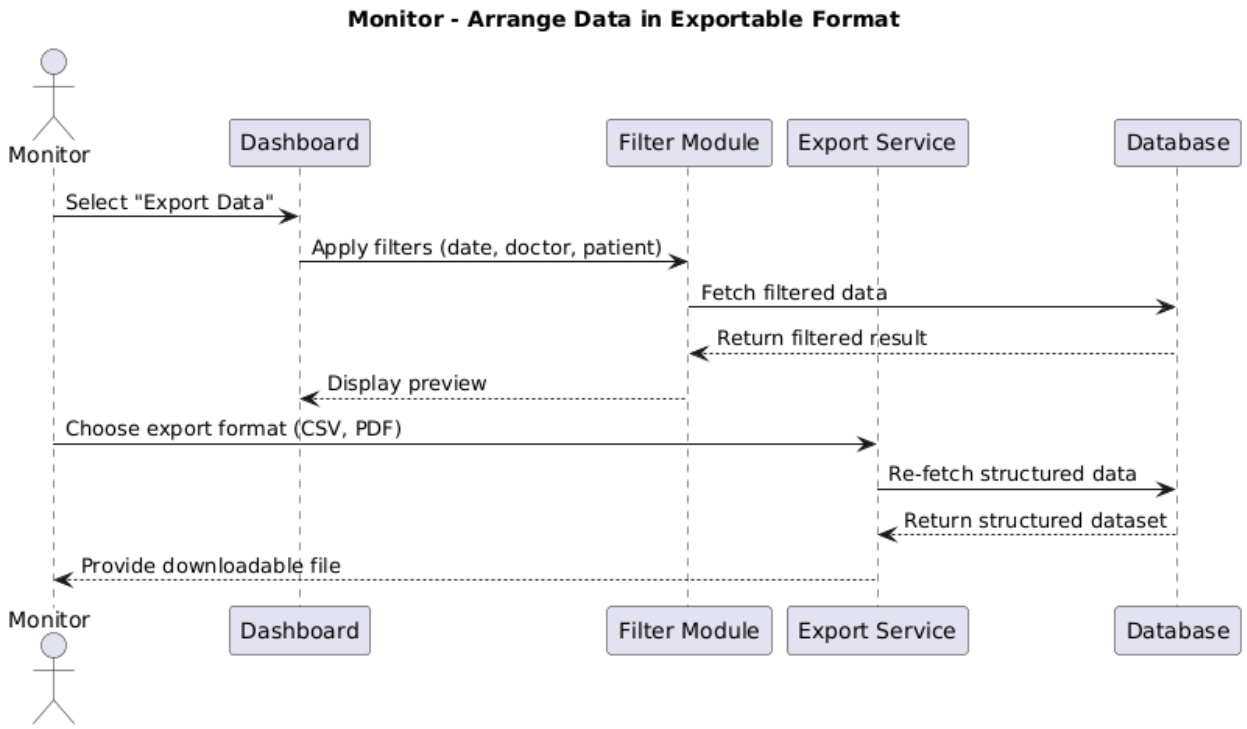

Admin Create/Update/Manage Schedule - Sequence Diagram
Admin
Front-end
Server-end
Availability Table
(doctor_id, date, time, is_available, status)
Schedule Table
(schedule_id, doctor_id, date, time, status)
Appointment Table
(appointment_id, patient_id, doctor_id, datetime, status)
Step 1: Load Doctor Availability
Open scheduling panel
Request available slots
SELECT * FROM Availability WHERE is_available = true AND status='approved'
Available slots
Return slots
Show doctor availability
Step 2: Admin creates schedule
Select slots to publish
Send selected slots
loop
[for each slot]
INSERT INTO Schedule (doctor_id, date, time, status='available')
Confirm insert
Confirm creation
Show schedule created
Step 3: Patient auto-books available slot
System auto-approves patient appointment
UPDATE Schedule SET status='booked'
WHERE doctor_id=? AND date=? AND time=?
Confirm booked
Step 4: Doctor modifies availability mid-week
Doctor updates availability (status='pending')
Check availability changes
SELECT * FROM Availability WHERE status='pending'
Return pending changes
Show to Admin
Approve or reject update
Update status (approved/rejected)
UPDATE Availability SET status='approved' WHERE id=?
Confirm update
Step 5: Admin handles appointment changes
Fetch change requests
SELECT * FROM Appointment WHERE status='change_requested'
Return list
Show to Admin
Approve change
Send update
UPDATE Appointment SET status='approved' WHERE appointment_id=?
Confirm update
Notify Admin
Change approved

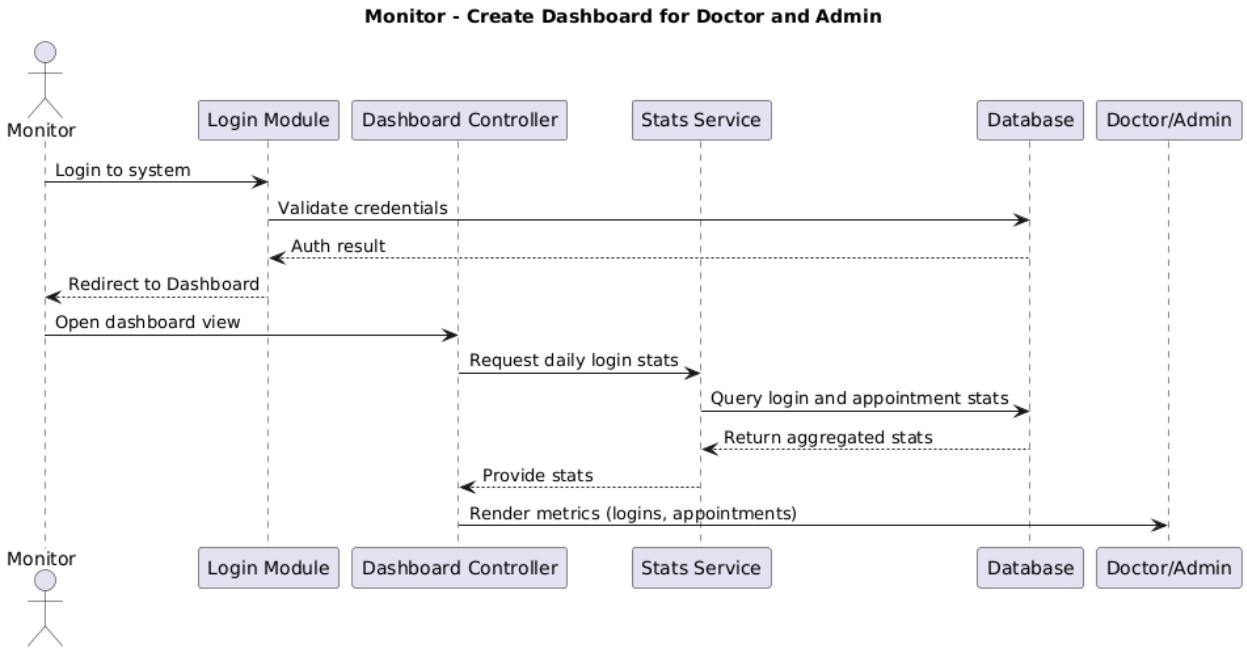

Admin Log In - Sequence Diagram
Admin
Front-end
Server-end
Admin Table
(admin_id, email, hashed_password, role)
Admin submits login credentials
Enter Email, Password
Send Email, Password
Server verifies credentials
SELECT * FROM Admin
WHERE email = ? AND password = ?
Admin record or NULL
Login Success or Failure
Result shown to Admin
Show dashboard or error
Admin
Front-end
Server-end
Admin Table
(admin_id, email, hashed_password, role)


Admin View Records - Sequence Diagram
Admin
Front-end
Server-end
Appointment Table
(appointment_id, patient_id, doctor_id, datetime, status, last_modified)
Schedule Table
(doctor_id, date, time, status, updated_by)
Admin views all appointments (past + upcoming)
Open Records Dashboard
Request appointment data
SELECT * FROM Appointment
All appointments
Return appointment list
Display appointments (filtered by date/status)
Admin views all schedule data (including changes)
Request doctor timetable
SELECT * FROM Schedule
Schedule records (including modified ones)
Return schedule
Show full timetable
Admin
Front-end
Server-end
Appointment Table
(appointment_id, patient_id, doctor_id, datetime, status, last_modified)
Schedule Table
(doctor_id, date, time, status, updated_by)


Admin Profile Management - Sequence Diagram
Admin
Front-end
Server-end
Doctor Table
(doctor_id, name, email, status, is_active, specialty)
Patient Table
(patient_id, name, email, status, is_active)
Panel 1: View and Approve/Reject Doctor Registrations
Open Doctor Registration Panel
Request doctor registrations
SELECT * FROM Doctor WHERE status = 'pending'
Pending doctor list
Return doctor list
Show registration profiles
Approve / Reject doctor
Send doctor_id + action
UPDATE Doctor SET status = 'approved' or 'rejected'
WHERE doctor_id = ?
Update result
Notify result
Confirmation shown
Panel 2: View/Edit User Profiles (Doctor and Patient)
Open Profile Management Panel
Request doctor + patient data
SELECT * FROM Doctor WHERE status = 'approved'
SELECT * FROM Patient
Doctor data
Patient data
Return user profiles
Display all profiles
View/Edit Profile (Fields or Status)
Send updated values (profile type + ID)
alt
[Editing Doctor]
UPDATE Doctor SET fields...
WHERE doctor_id = ?
Update result
[Editing Patient]
UPDATE Patient SET fields...
WHERE patient_id = ?
Update result
Send confirmation
Show result
Terminate Doctor Profile on Resignation
Mark doctor as resigned
Send doctor_id + status='terminated'
UPDATE Doctor SET status='terminated' WHERE doctor_id = ?
Success
Notify termination
Show doctor status = terminated

Monitor sequence diagram(arnob):





Resource Management Document – MediAccessHub

1. Introduction(prateek)

Resource Management ensures that all modules of MediAccessHub — including Doctor, Patient, Admin, and Monitoring components — are developed and maintained efficiently. It focuses on optimal use of human resources, technology, and time while ensuring each team member can contribute effectively toward their respective sections.

2. Human Resources

Team Member	Role	Responsibility Summary
Tashfiqul Rhidoy Prodhan	Doctor Portal Developer	Handles doctor login, dashboard, availability update, patient record operations
Md Mahmudul Hasan Saif	Admin Portal Developer	Develop admin interface, user management, schedule system and approval flows
Prateek Maharjan	Patient Portal Developer	Manages patient registration, appointments, record access, and interface usability.
Abyead Faisal Arnob	Monitoring & Reporting Developer	Implements dashboards, data tracking, visualization, and export functions.

3. Technical Resources (tashfiqul)

Resource Type	Tools & Details
Development Tools	Visual Studio Code, IntelliJ, or any IDE suitable for web & backend development
Version Control	GitHub – to track source code and enable collaborative development
Hosting/Test Server	Localhost or cloud environments like Heroku, Vercel, or Firebase (free tiers)

Database	MySQL / Firebase Realtime Database / MongoDB (based on selected backend)
Other Dependencies	Node.js / Express / Django (backend frameworks as required)

4. Financial Resources (Arnob)

- **Estimated Cost:** $0
- **Remarks:** All components will utilize **open-source** or **freemium tools**. No paid resources are required unless upgraded features or deployments are later desired.

5. Module-Wise Resource Assignment

Doctor Portal (Tashfiqul)

- **Assigned Developer:** Tashfiqul Rhidoy Prodhan
- **Responsibilities:**
    - Build secure login system for doctors
    - Design and implement dashboards
    - Enable CRUD operations for patient records
    - Manage doctor availability using calendar interface
    - View scheduled appointments
- **Resources Used:**
    - PC/Laptop
    - Coding software (IDE)
    - Secure database with role-based access
    - GitHub for code tracking

Patient Portal (Prateek)

- **Assigned Developer:** Prateek Maharjan

- **Responsibilities:**
    - Enable user registration and secure login
    - Display medical records with download options
    - Implement appointment request and modification flows
- **Resources Used:**
    - HTML/CSS, React or Vue.js for frontend
    - Database for patient records and appointments
    - Secure encryption libraries for password hashing
    - Form validation logic

Admin Portal (saif)

- **Assigned Developer:** Md Mahmudul Hasan Saif
- **Responsibility:**
    - Manage all users (Doctors and patients)
    - Approve/Reject doctor registration and availability
    - Manage system schedule and appointment conflicts
    - Maintain system-Wide access control and monitoring
- **Resources Used:**
    - Admin dashbaord framework (e.g, AdminLTE, Bootstrap)
    - Role-based access control mechanisms
    - Notifications module for approvals
    - Audit trail loggin system

Monitoring & Reports Module (Arnob)

- **Assigned Developer:** Abyead Faisal Arnob
- **Responsibilities:**
    - Implement dashboards showing login and appointment statistics
    - Export data in CSV or PDF
    - Real-time system usage tracking

- **Resources Used:**
  - Charting libraries (e.g., Chart.js, D3.js)
  - Export libraries (e.g., jsPDF, PapaParse)
  - File generation tools
  - Backend query optimization for reporting

6. Risk Management and Contingencies (Saif)

- **Data Loss**: GitHub version control and local backups mitigate this
- **Team Unavailability**: Team members can hand over documentation/code access
- **Performance Bottlenecks**: Load testing to be done during development to avoid crashes
- **Security Threats**: Secure password hashing, role-based access and HTTPS enforcement

Pseudocode for MediAccessHub

Pseudocode for Patient (Prateek):

1. Register a new user

```
START RegisterNewUser


  DISPLAY "Enter Full Name, Email, Password, Confirm Password"
  INPUT fullName, email, password, confirmPassword


  IF password == confirmPassword THEN
    CALL ValidateInput(fullName, email, password)
    IF input is valid THEN
      HASH password
```

```
        INSERT newUser INTO PatientDatabase
        DISPLAY "Registration Successful"
      ELSE
        DISPLAY "Invalid Input. Please check your details."
      ENDIF
    ELSE
      DISPLAY "Passwords do not match."
    ENDIF

  END RegisterNewUser
```

2. Login

```
  START PatientLogin

    DISPLAY "Enter Email and Password"
    INPUT email, password

    WHILE loginAttempts < 3 DO
      IF CredentialsAreValid(email, password) THEN
        DISPLAY "Login Successful"
        BREAK
      ELSE
        DISPLAY "Invalid credentials. Try again."
```

```
        loginAttempts = loginAttempts + 1
    ENDIF
  ENDWHILE

  IF loginAttempts == 3 THEN
    DISPLAY "Too many attempts. Try again later."
  ENDIF

END PatientLogin
```

3. View Medical Record

```
START ViewMedicalRecords

  IF userIsAuthenticated THEN
    FETCH medicalRecords FROM MedicalRecordDatabase WHERE patientEmail = user.email
    DISPLAY records in table format
    DISPLAY "Download" button for each record
  ELSE
    DISPLAY "Access Denied. Please login first."
  ENDIF
```

END ViewMedicalRecords

4. Request appointment

START RequestAppointment

DISPLAY "Select doctor and preferred time slot"

INPUT doctorName, appointmentDate, appointmentTime

IF selectedSlot IS available THEN

SAVE appointment TO AppointmentDatabase

DISPLAY "Appointment Confirmed"

ELSE

DISPLAY "Selected slot is not available. Please choose a different time."

ENDIF

END RequestAppointment

5. View Upcoming Appointments & Request Change

START ViewAppointments

FETCH appointments FROM AppointmentDatabase WHERE patientEmail = user.email

DISPLAY list of upcoming appointments

```
  IF patient selects "Request Change" THEN
     DISPLAY form to select new date/time
     INPUT newDate, newTime

     IF new slot IS available THEN
        UPDATE appointment WITH newDate, newTime
        DISPLAY "Appointment Updated Successfully"
     ELSE
        DISPLAY "New slot unavailable"
     ENDIF
  ENDIF

END ViewAppointments
```

Pseudocode: Doctor Portal

1. Doctor Login

This pseudocode describes the logic for securely logging in a doctor.

```
FUNCTION doctorLogin(email, password)
    // Find the user in the database with the provided email
    user = Database.findUserByEmail(email)

    // Check if a user with that email was found
    IF user IS NOT found THEN
        RETURN "Error: No account found with that email."
    END IF

    // Verify the user's role is 'Doctor'
    IF user.role IS NOT "Doctor" THEN
        RETURN "Error: Access denied for this role."
    END IF

    // Hash the input password and compare it to the stored hash
    IF password_matches(user.hashedPassword, password) THEN
        // If successful, create a secure session
        Session.create(user.id)
        RETURN "Login successful"
    ELSE
        // If passwords do not match, return an error
        RETURN "Error: Invalid password."
    END IF
END FUNCTION
```

2. Display Doctor Dashboard

This logic fetches the necessary data to show the doctor's personalized dashboard.

```
FUNCTION getDashboardData(loggedInDoctorID)
    // Get today's date to find current appointments
    currentDate = today()

    // Get all appointments for the logged-in doctor scheduled for today
    todaysAppointments = Database.query("SELECT * FROM Appointments WHERE
DoctorID = loggedInDoctorID AND AppointmentDate = currentDate")

    // Get the doctor's name for a welcome message
    doctorInfo = Database.query("SELECT FullName FROM Users WHERE UserID =
loggedInDoctorID")
    welcomeMessage = "Welcome, " + doctorInfo.FullName

    // Return all data needed to build the dashboard
    RETURN { welcomeMessage, todaysAppointments }
END FUNCTION
```

3. View Assigned Patients

This logic retrieves the list of all patients assigned to the logged-in doctor.

```
FUNCTION viewAssignedPatients(loggedInDoctorID)
    // Query the database for all patients linked to this doctor's ID
    assignedPatientsList = Database.query("SELECT PatientID, FullName,
DateOfBirth FROM Patients WHERE DoctorID = loggedInDoctorID")

    // Return the list
    RETURN assignedPatientsList
END FUNCTION
```

4. Manage Medical Records

This is a complex functionality broken into smaller parts: Create, Edit, and Delete.

Create Medical Record

```
FUNCTION createMedicalRecord(loggedInDoctorID, patientID, recordData)
    // First, confirm this doctor is authorized for this patient
    isAuthorized = Database.verifyDoctorPatientLink(loggedInDoctorID,
patientID)
    IF isAuthorized IS FALSE THEN
        RETURN "Error: Not authorized for this patient."
    END IF

    // Validate the incoming data (e.g., check for empty fields)
    isValid = validate(recordData)
    IF isValid IS FALSE THEN
        RETURN "Error: Invalid data provided."
    END IF

    // Save the new record to the database
    Database.query("INSERT INTO MedicalRecords (PatientID, DoctorID,
Diagnosis, Prescription, Notes) VALUES (...)")

    RETURN "Success: Medical record created."
END FUNCTION
```

Edit Medical Record

```
FUNCTION editMedicalRecord(loggedInDoctorID, patientID, recordID,
updatedData)
    // Confirm this doctor is authorized for this patient
    isAuthorized = Database.verifyDoctorPatientLink(loggedInDoctorID,
patientID)
    IF isAuthorized IS FALSE THEN
        RETURN "Error: Not authorized for this patient."
    END IF
```

```
    // Validate the updated data
    isValid = validate(updatedData)
    IF isValid IS FALSE THEN
        RETURN "Error: Invalid data provided."
    END IF

    // Update the specific record in the database
    Database.query("UPDATE MedicalRecords SET ... WHERE RecordID = recordID")

    RETURN "Success: Medical record updated."
END FUNCTION
```

Delete Medical Record

```
FUNCTION deleteMedicalRecord(loggedInDoctorID, patientID, recordID)
    // Confirm this doctor is authorized for this patient
    isAuthorized = Database.verifyDoctorPatientLink(loggedInDoctorID,
patientID)
    IF isAuthorized IS FALSE THEN
        RETURN "Error: Not authorized for this patient."
    END IF

    // Delete the specific record from the database
    Database.query("DELETE FROM MedicalRecords WHERE RecordID = recordID")

    RETURN "Success: Medical record deleted."
END FUNCTION
```

5. Manage Availability

This logic allows a doctor to set or update their work hours.

```
FUNCTION manageAvailability(loggedInDoctorID, availabilityData)
    // availabilityData is a list of dates/times, e.g., [{date: "2025-07-10",
start: "09:00", end: "17:00"}]

    // Loop through each availability entry provided by the doctor
    FOR EACH entry IN availabilityData
        // Validate the entry's data
        IF entry.start_time IS AFTER entry.end_time THEN
            CONTINUE to next entry with error message "Start time must be
before end time"
        END IF

        // Check if an entry for this date already exists for the doctor
        existingEntry = Database.query("SELECT * FROM DoctorAvailability
WHERE DoctorID = loggedInDoctorID AND AvailableDate = entry.date")

        IF existingEntry IS found THEN
            // If it exists, update it with the new times
```

```
            Database.query("UPDATE DoctorAvailability SET StartTime =
entry.start, EndTime = entry.end WHERE ...")
        ELSE
            // If it doesn't exist, insert a new availability entry
            Database.query("INSERT INTO DoctorAvailability (DoctorID,
AvailableDate, StartTime, EndTime) VALUES (...)")
        END IF
    END FOR

    RETURN "Success: Availability has been updated."
END FUNCTION
```

Pseudocode for Admin (Saif):

Admin Login

Front-End (login page)

Function adminLogin() {

username = input_username

password = input_password

send_login_info_to_server(username, password)

}

Back-End (Server)

Function send_login_info_to_server(username, password){

If check_credentials(username, password) == true {

Load_dashboard()

} else {

show_error(“Invalid username or password”)

}

}

Profile Management

Front-End (profile Management Page)

```
Function manageUserProfile() {
Admin_login_dashboard()
User_list = Fetch_all_users() // doctors and patients
Display_user_list(user_list)
Selected_user = Admin_selects_user()
User_profile = Fetch_user_profile(selected_user)
Display_user_profile(user_profile)
If admin_clicks_edit_profile() {
Update_data = Get_update_profile_input()
send_update_profile_to_server(update_data)
}
If admin_reviews_new_doctor_registration() {
registration_info = View_doctor_application()
        Decision = Admin_approve_or_rejected(registration_info)
        Send_registration_decision_to_server(registration_info, decision)
    }
    If admin_toggles_active_status() {
        Status = Admin_select_active_or_Inactive()
        Send_status_update_to_server(selected_user, status)
    }
    If admin_click_terminate_doctor() AND
```

```
Doctord_has_resigned(selected_user) {
        Send_termination_request_to_server(selected_user)
        }
}
```

Back-End (server):

```
Function Fetch_all_users() {
Return Query_all_registration_users() }
Function Fetch_user_profile(user_id) {
Return Query_user_profile_by_id(user_id) }
Function Send_update_profile_to_server(updated_data) {
If validate_profile_data(update_data) == true {
Update_profile_in_database(updated_data) } else {
Show_error ("Invalid profile data")}}
Function Send_registration_decision_to_server(registration_info, decision) {
If decision == "approve" {
Approve_doctor_registration(registration_info)
Notify_doctor("Registration Approved") } else
Reject_doctor_registration(registration_info)
Notify_doctor("Registration Rejected") }}
Function Send_status_update_to_server(user_id, status) {
Update_user_status(user_id, status) // active or inactive
Notify_user_status_change(user_id, status) }
Function Send_termination_request_to_server(doctor_id) {
Doctor_has_resigned(doctor_id) == true {
Mark_doctor_as_terminated(doctor_id)
```

Notify_termination(doctor_id) } else {

Show error(“Doctor not eligible for termination”) }

}

Create/Update/Manage schedule:

Front-End (schedule Management Page)

Function manageSchedule() {

Admin_login_dashboard()

Doctor_availability = Fetch_update_availability()

Display_availability(doctor_availabiltiy)

if admin_wants_to_create_schedule() {

Schedule_input = Get_schedule_input()

Send_schedule_to_server(schedule_input)

}

If admin_recived_notification(“availability_change”) {

Change_details = View_change_request()

decison = Admin_approve_or_reject(change_details)

Send_availability_decision_to_server(change_details, decision)

}

If admin_recieve_notfication(“appointment_change”) {

Appointment_request = view_appointment_change()

decision = Admin_approve_or_reject(appointment_request)

Send_appointment_decision_to_server(appointment_request, decision) }

}

Back-End (Server):

```
Function send_schedule_to_server(shcedule_input) {
        If validate_schedule(schedule_input) == true {
        Save_schedule_to_database(schedule_input)
        Notify_doctor(schedule_input) } else {
        Log_error("Invalid Schedule data") }
        }
Function send_availability_decison_to_server(change_details, decision) {
If decision == "approve" {
Apply_availability_update(change_details)
Notify_doctor_and_update_UI(change_details)
} else {
Reject_availabiltiy_update(change_details)
Notfiy_doctor_rejection(change_details)
} }
Function send_appointment_decision_to_server(appointment_change_request, decision) {
If decision == "approve"
Update_appointment_in_database(appointment_change_reqeust)
Notify_pateint_and_doctor(appointment-change_reqeust) }
}
```

View Medical Records

Front-End (Admin Records Page)

```
Function viewmedicalRecords() {
Admin_login_dahsboard()
Patient_id = Get_input("Enter patient ID")
Send_patient_id_to_server(server_ID)
```

}

Back-End(server)

Function send_patient_id_to_server(patient_id)

Id patient_exists(patient_id) == true {

Medical_history = Fetch_medical_history(patient_id)

Past_appointment = Fetch_pastappointment(patient_id)

Upcoming_appointments = Fetch_upcoming_appointment(patient_id)

Display_to_admin(medical_history, past_appoitnment, upcoming_appointments){

Else {

Show_error("patient not found") }

}

Pseudocode for monitor (Arnob):

Front-end:

1. Load Page and Display Patients List:

    IF user_role IS Admin OR Doctor THEN
        DISPLAY patient list table
        ENABLE search and filter options

    ELSE IF user_role IS Patient THEN

DISPLAY "Export My Records" button

2. Select Patient to View Records:

WHEN user clicks "Select" on a patient row THEN SET selected_patient_id = chosen ID

CALL API: GET /patients/{selected_patient_id}/records

IF response.status IS success THEN
DISPLAY second table with medical records

ELSE
DISPLAY "Access Denied" or "No Records Found"

3. Export the Record:

WHEN user clicks "Export as PDF" OR "Export as CSV" THEN SET export_format = selected option SET patient_id = selected_patient_id

CALL API: POST /export WITH patient_id AND export_format

IF response.status IS success THEN
DOWNLOAD the returned file

ELSE
DISPLAY "You are not authorized to export this data."

Back End:

1. Fetch Patient List:

FUNCTION get_patient_list(user_role, user_id): IF user_role IS Admin THEN RETURN all patients

ELSE IF user_role IS Doctor THEN
RETURN patients assigned to user_id

ELSE IF user_role IS Patient THEN
RETURN only the record for user_id

ELSE
RETURN "Access Denied"

2. Fetch Patient Medical Records:

FUNCTION get_patient_records(requesting_user_role, requesting_user_id, target_patient_id): IF requesting_user_role IS Admin THEN ALLOW access

ELSE IF requesting_user_role IS Doctor THEN
CHECK IF doctor is assigned to target_patient_id
IF NOT assigned THEN
RETURN "Access Denied"

ELSE IF requesting_user_role IS Patient THEN
IF requesting_user_id != target_patient_id THEN
RETURN "Access Denied"

FETCH and RETURN records from database

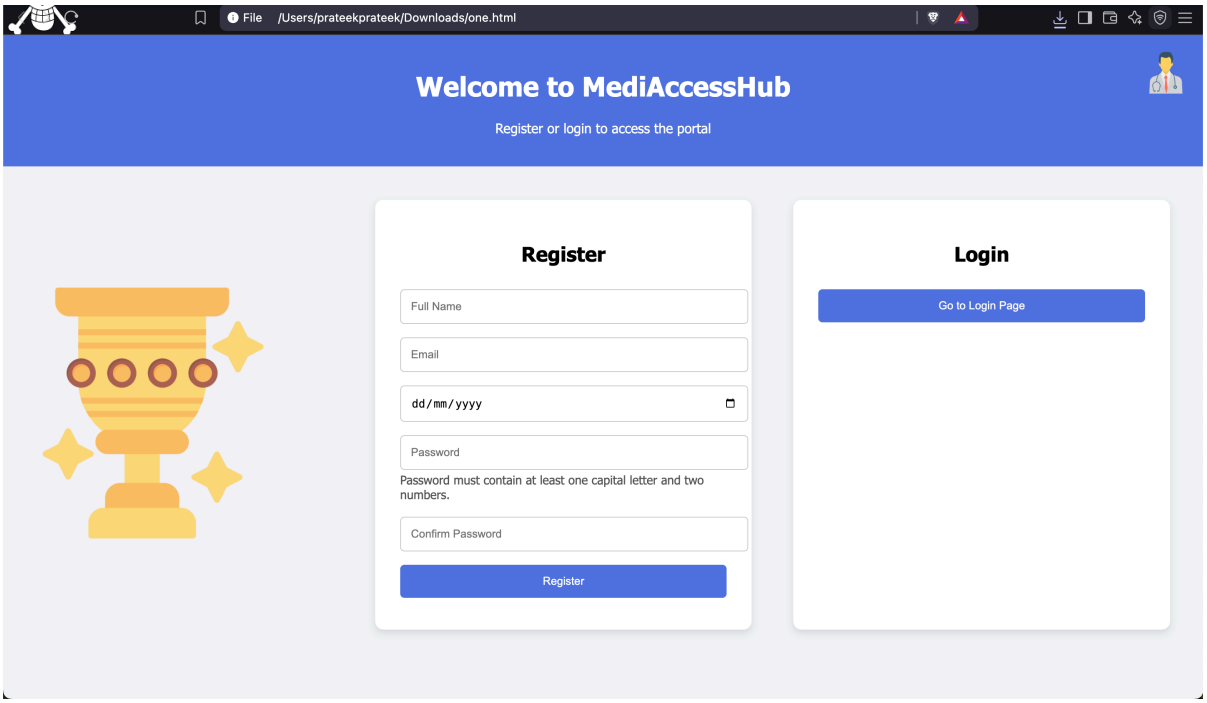
3. Export Record as File:

```
    FUNCTION export_patient_record(user_role, user_id, patient_id, format):
    CALL get_patient_records(user_role, user_id, patient_id)

IF response IS "Access Denied" THEN
   RETURN "Unauthorized"

IF format IS PDF THEN
   GENERATE PDF with patient record

ELSE IF format IS CSV THEN
   GENERATE CSV with patient record

RETURN file for download
```

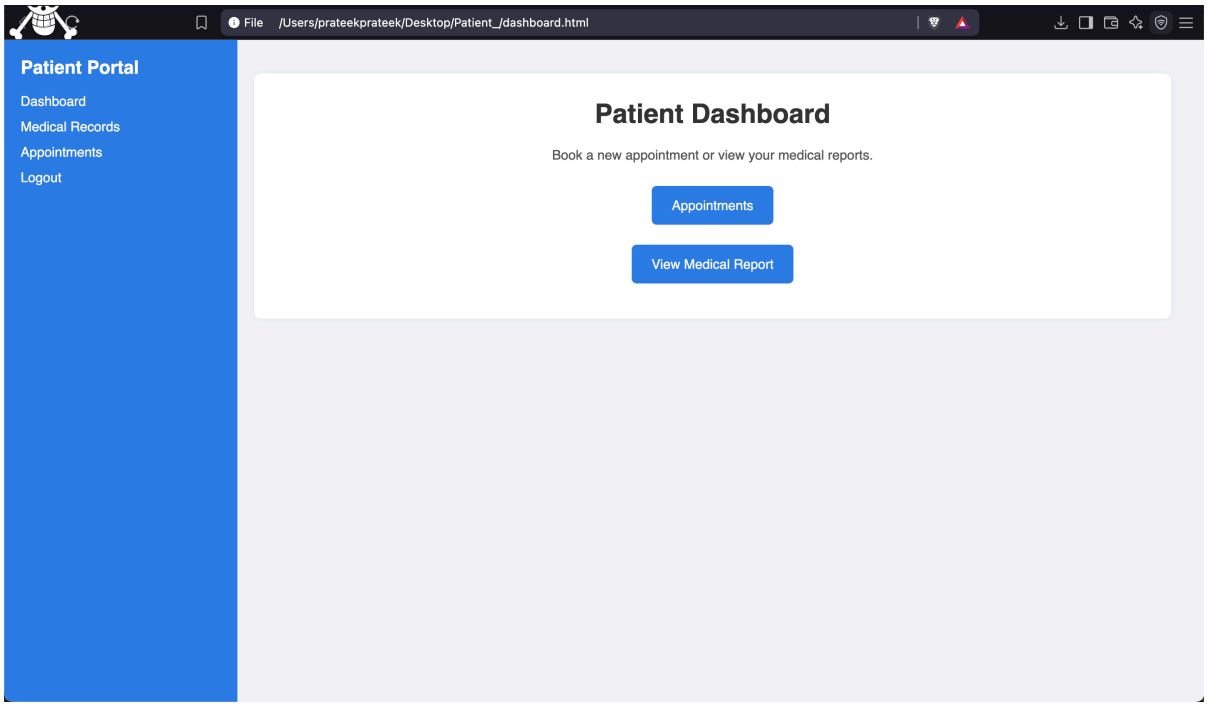
UI Design for MediAccessHub

UI design for patient portal (Prateek):

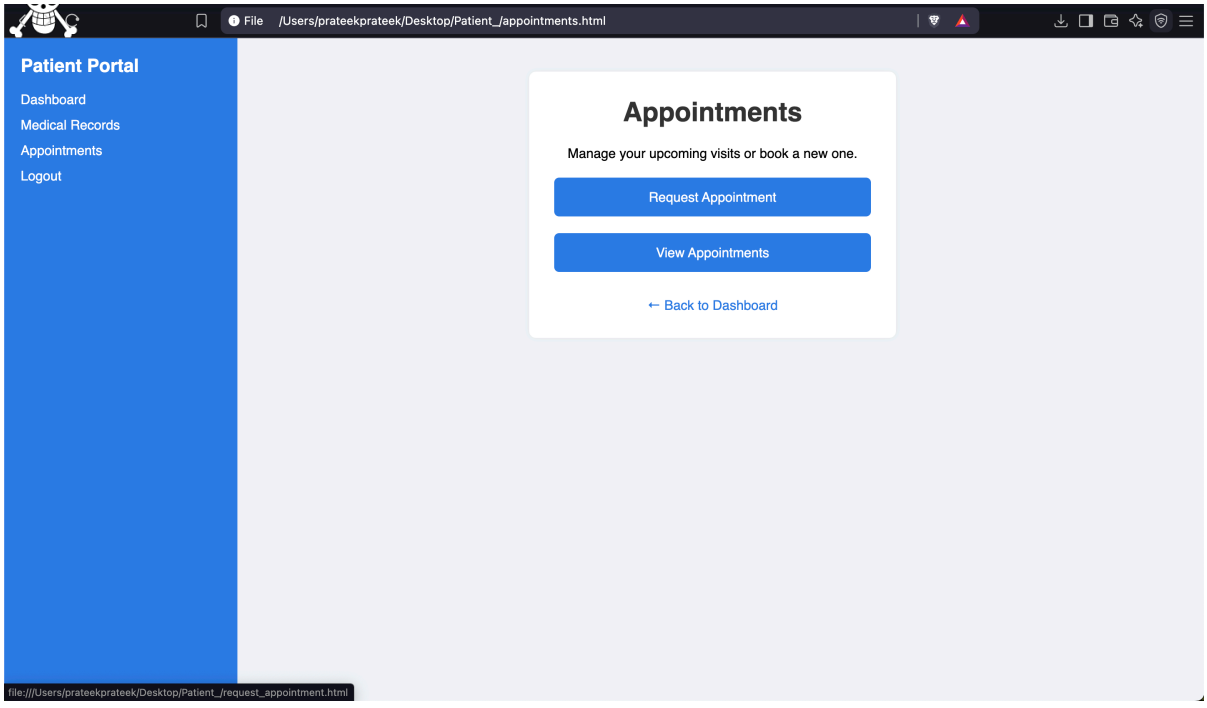
1. The Access Portal



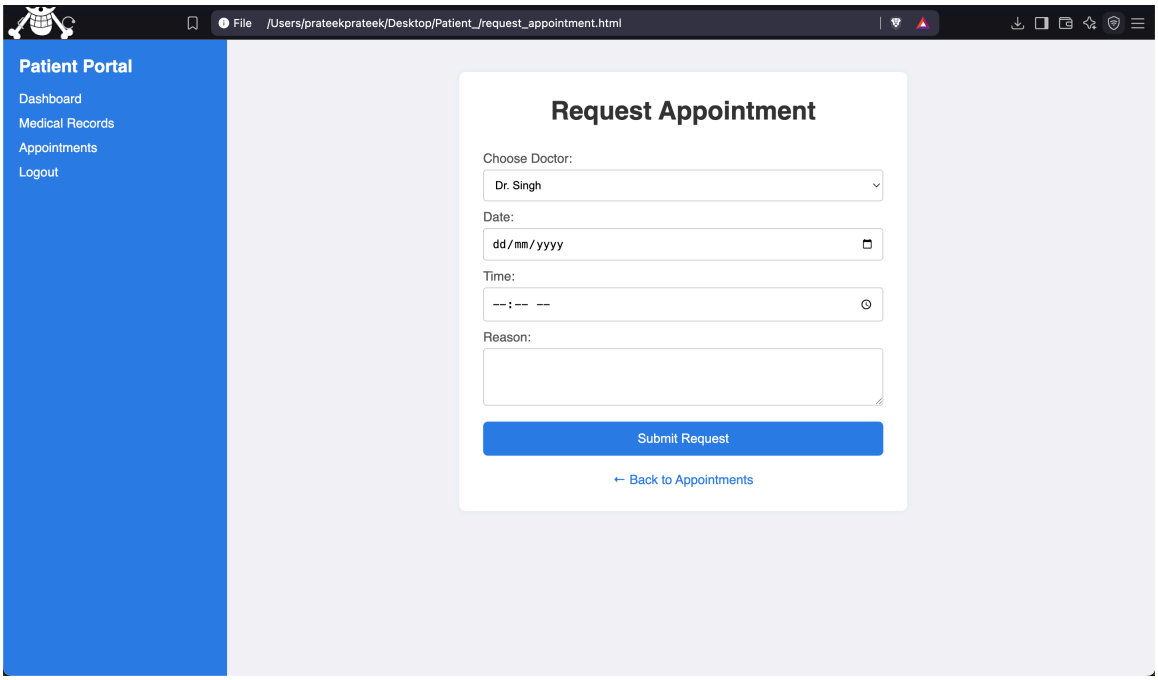
2. **Dashboard**



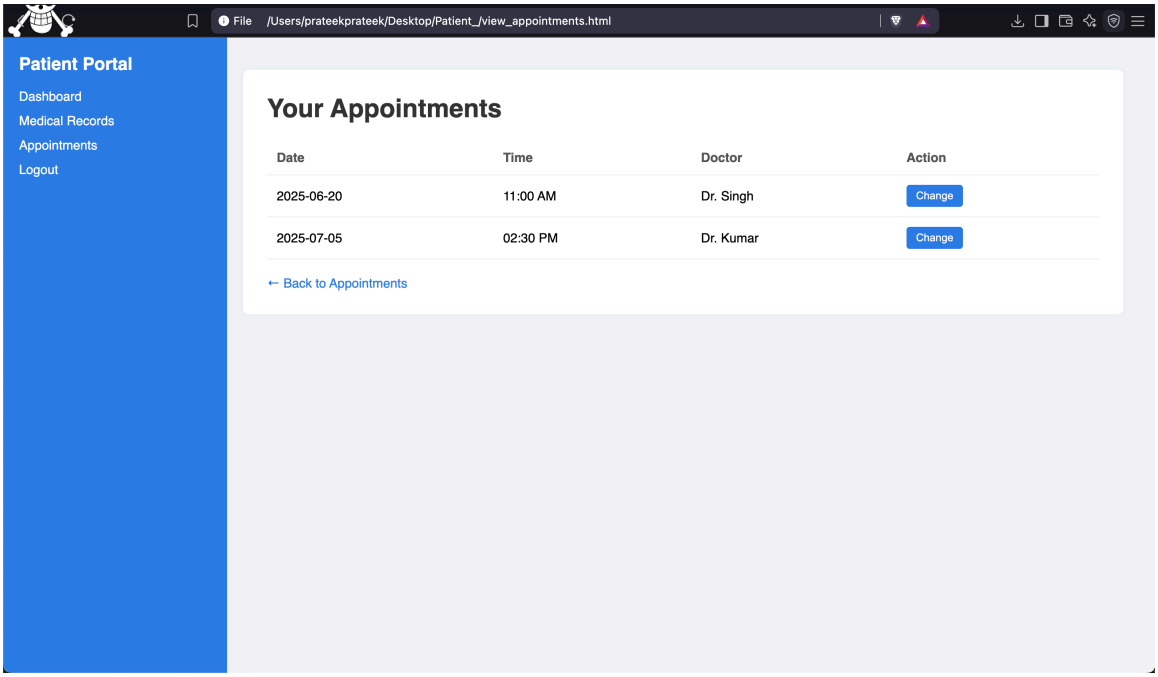
3. **Appointment Portal**



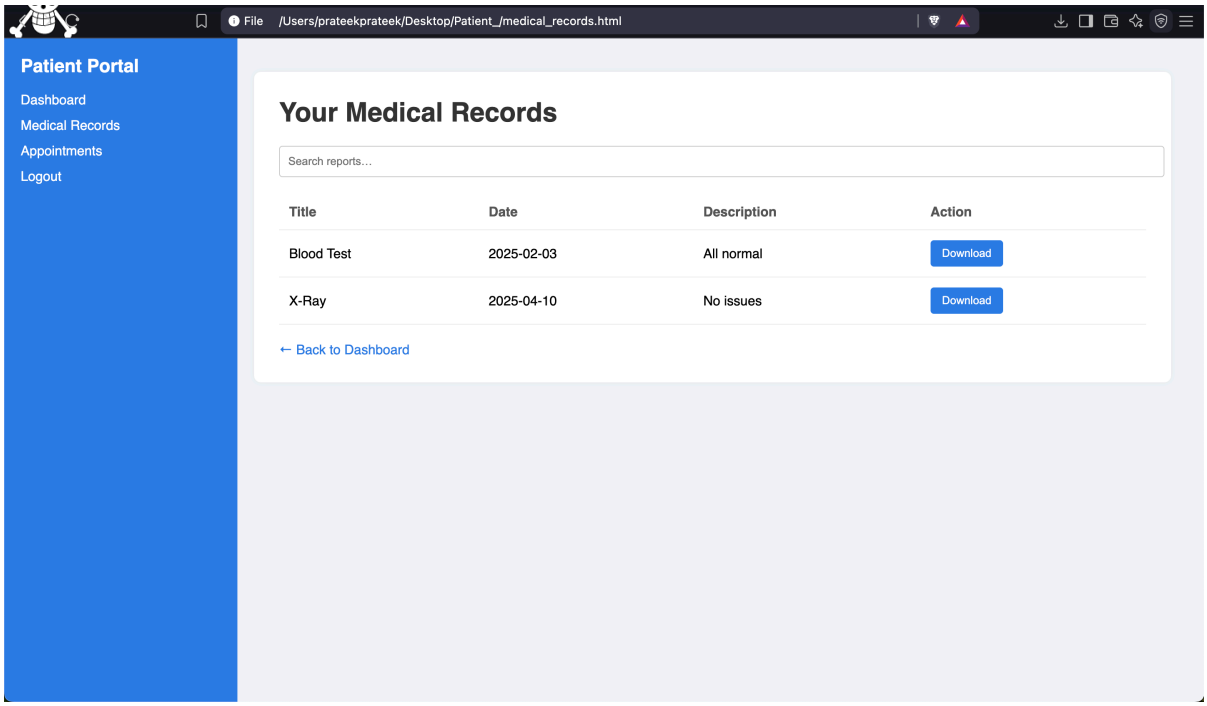
- **Request Appointment**



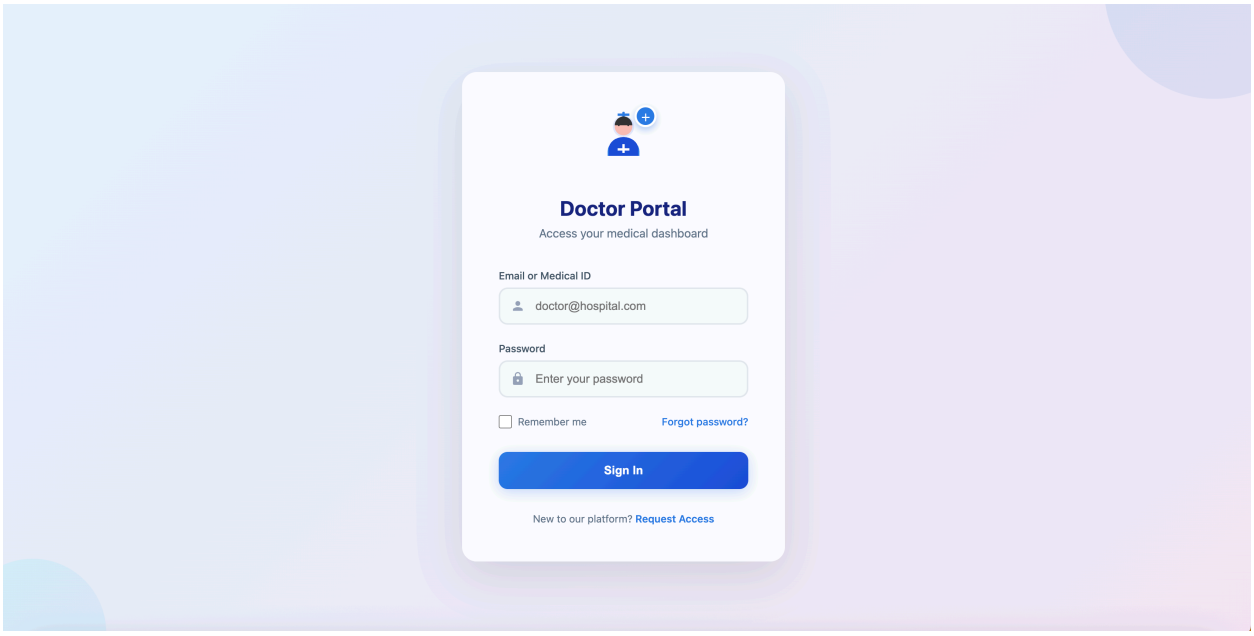
- **View Appointment**

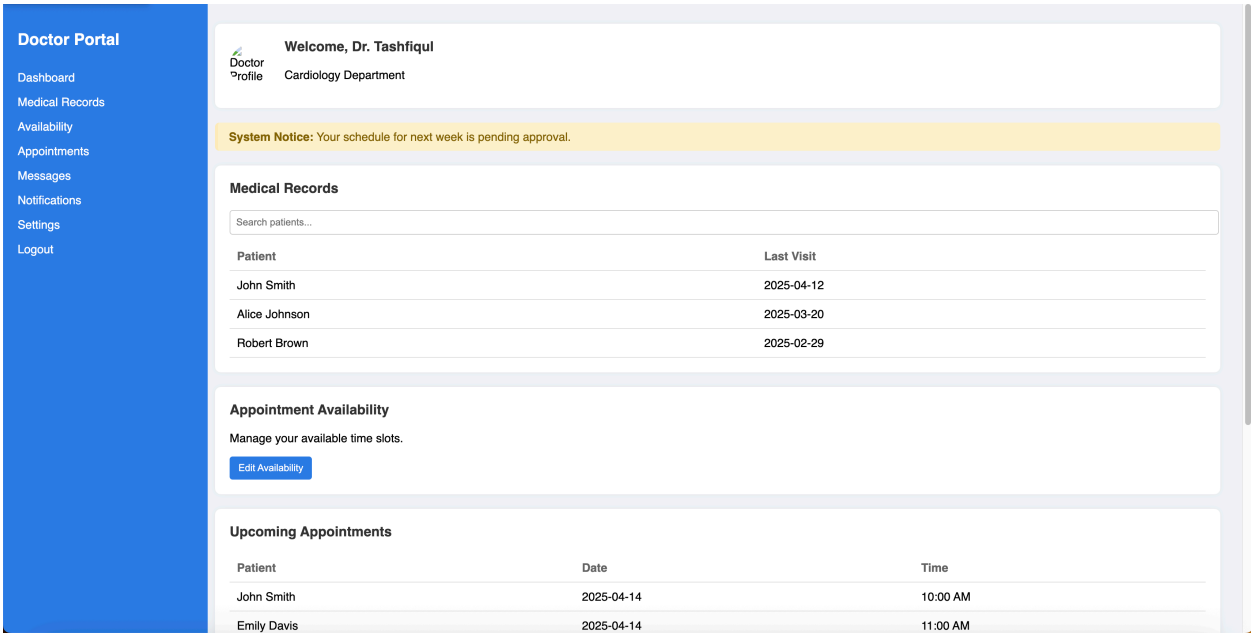


4. **View Medical Report**



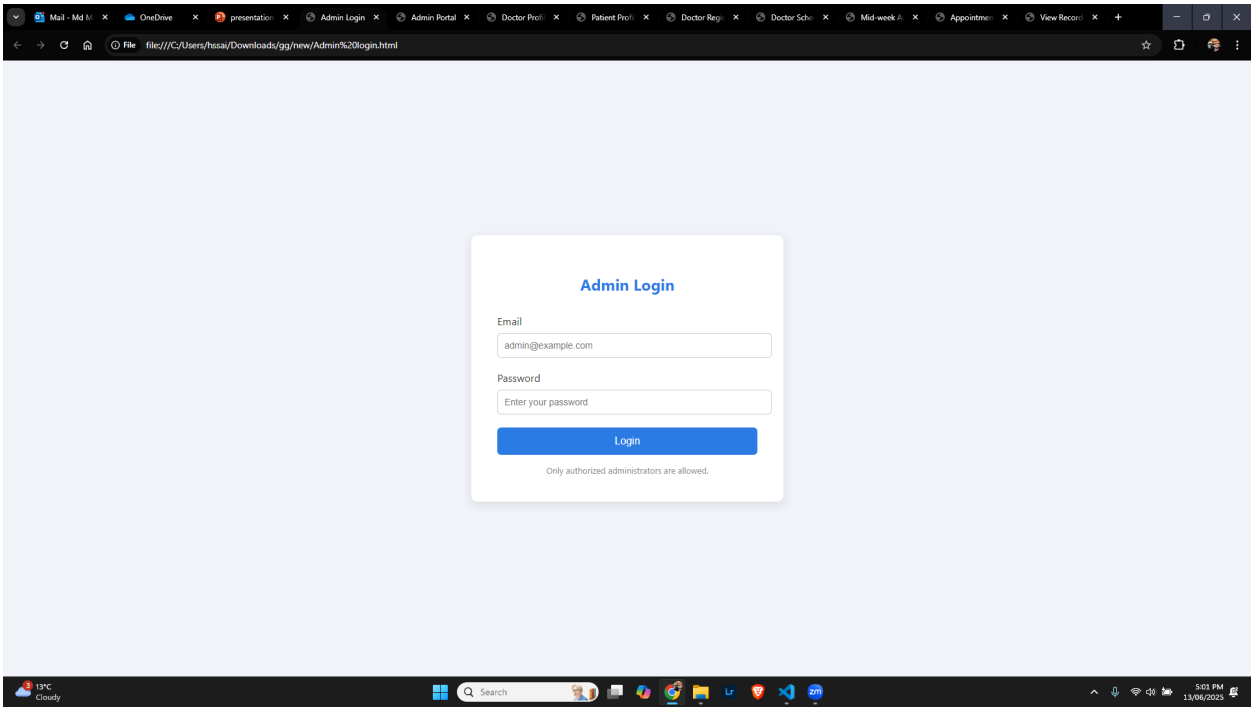
UI design for Doctor’s Portal (Tashfiqul):



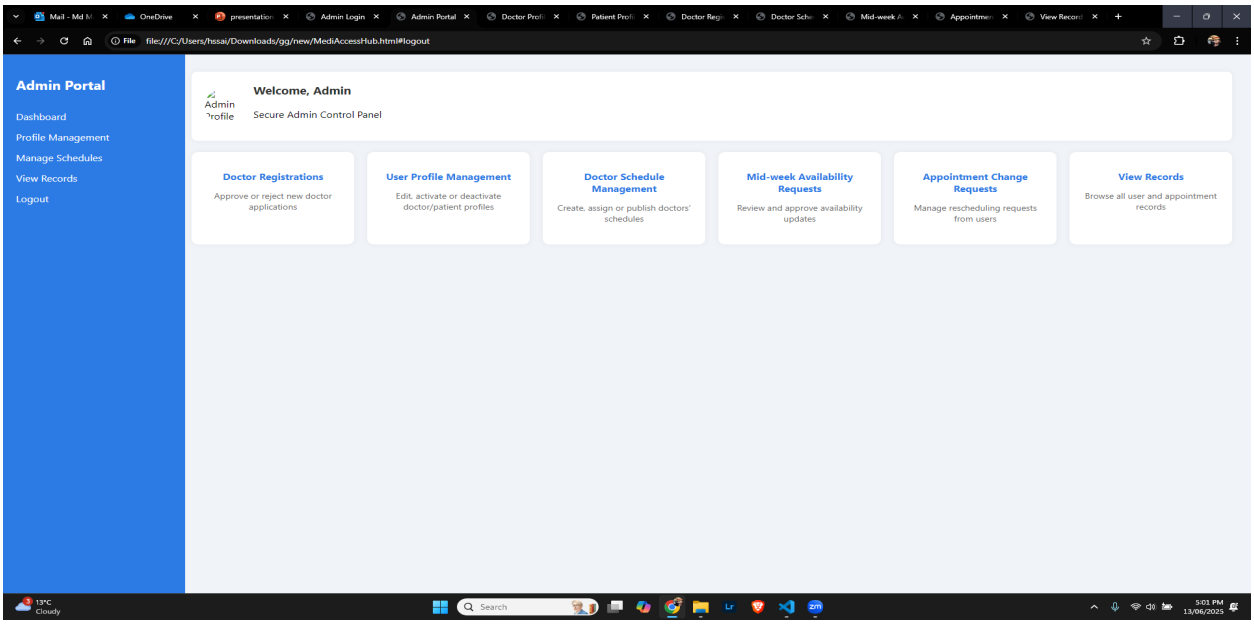


Ui Design for MediAccessHub Admin Portal (Saif):

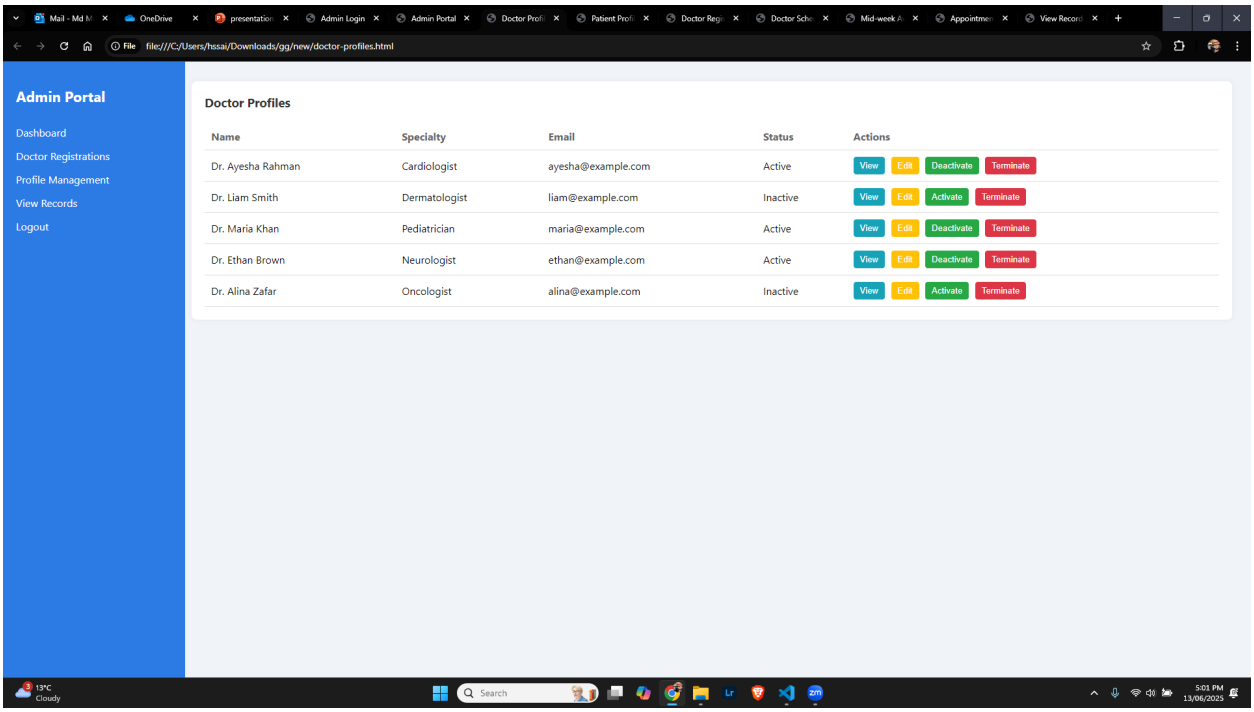
Admin Login:

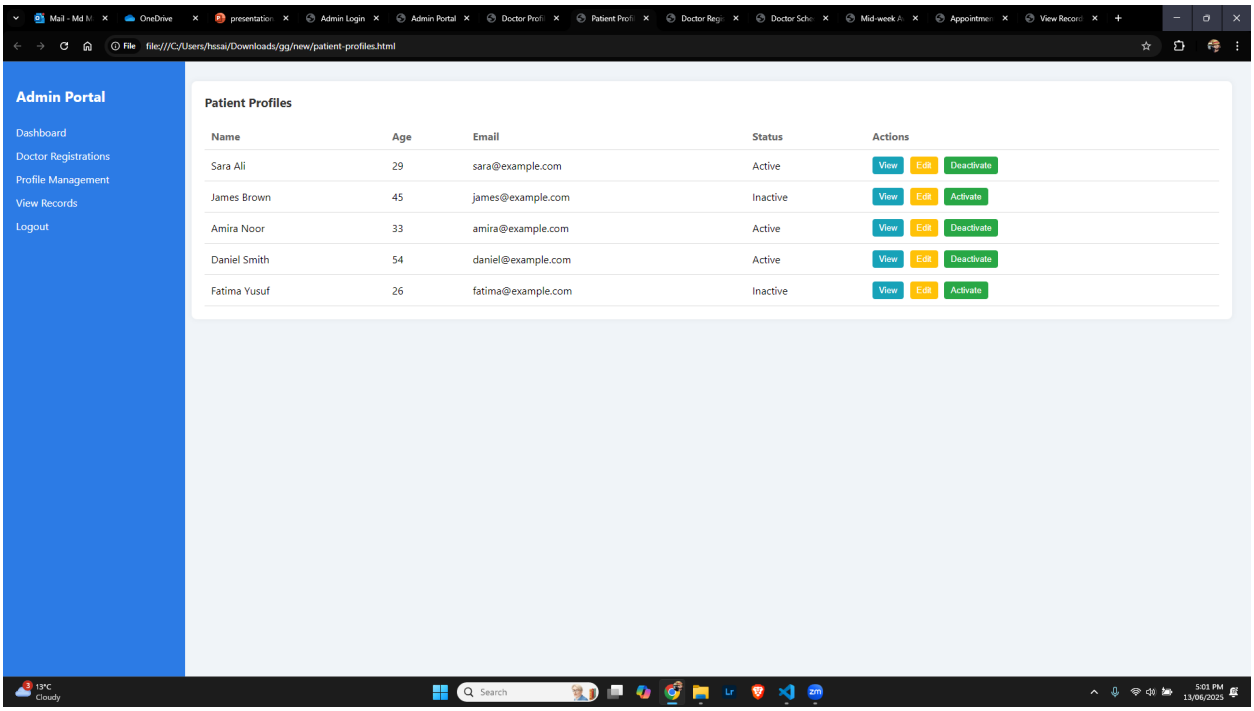


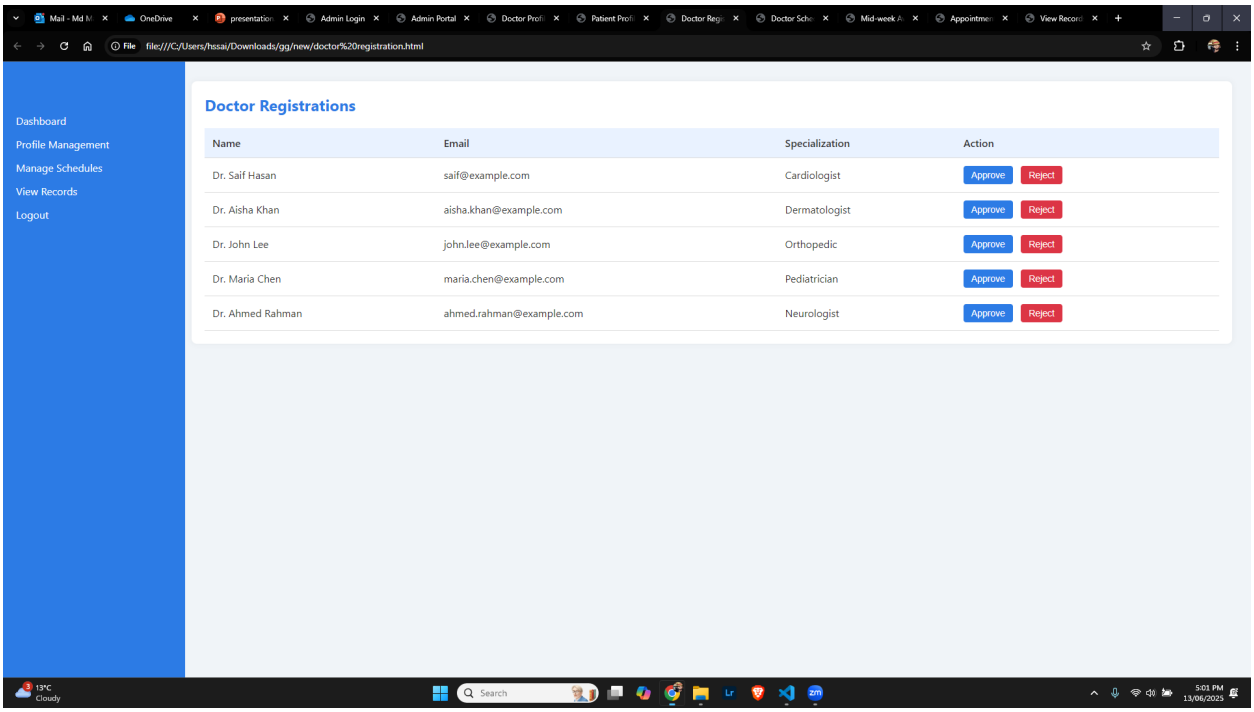
Admin Dashboard:



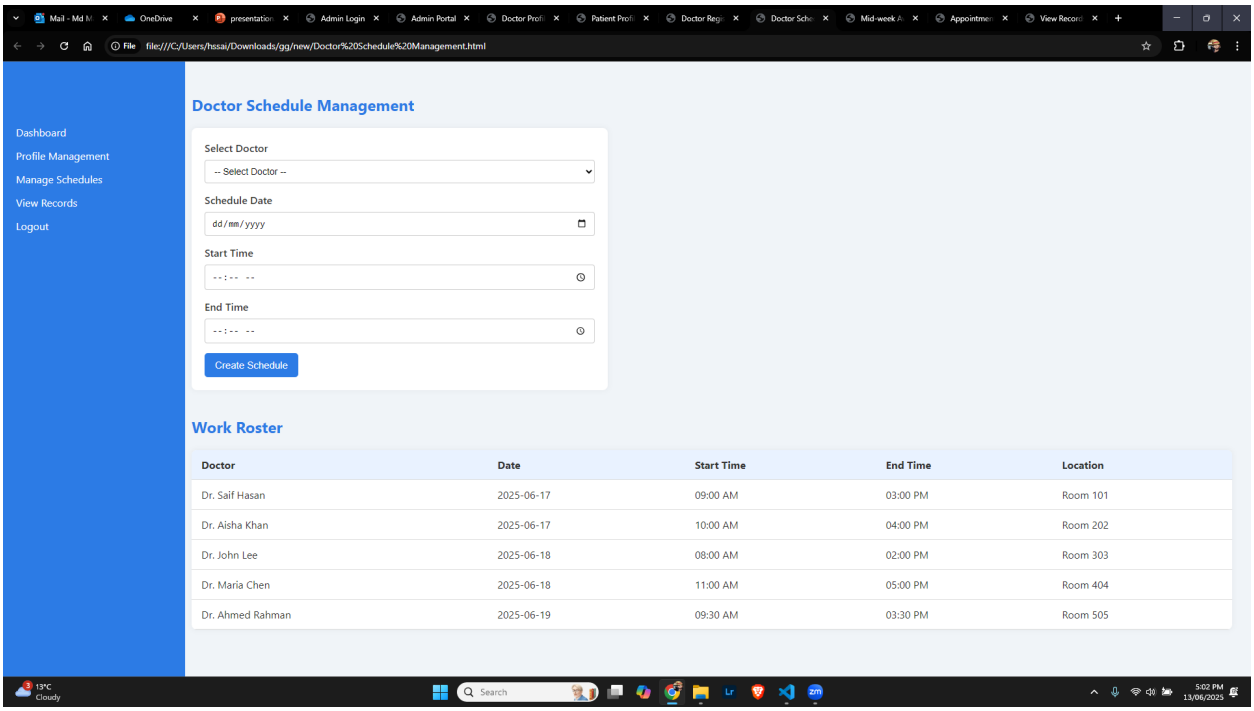
Admin Profile Management:

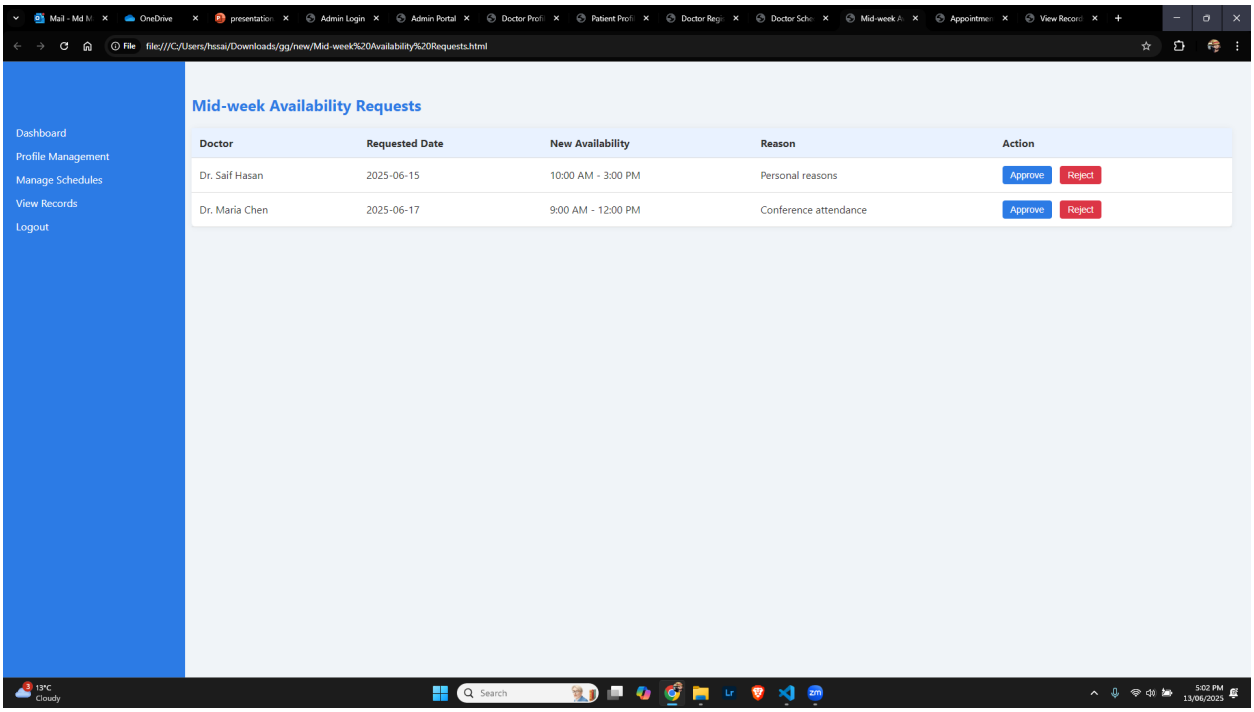


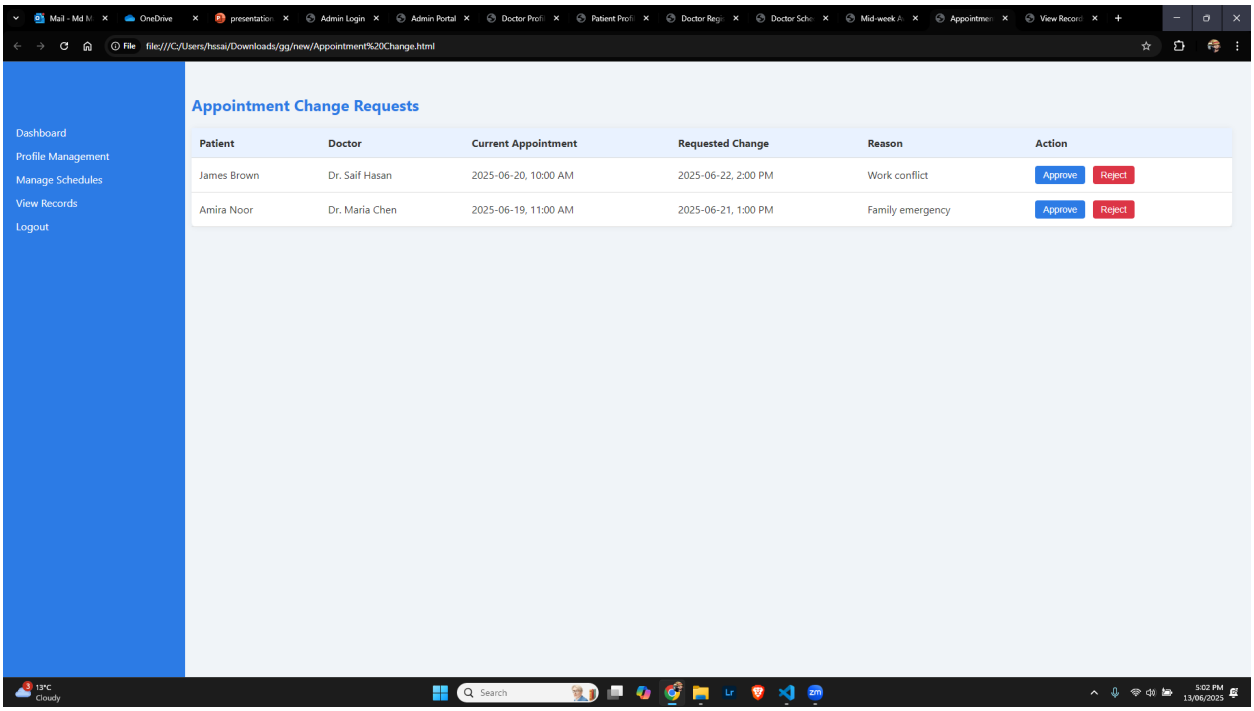




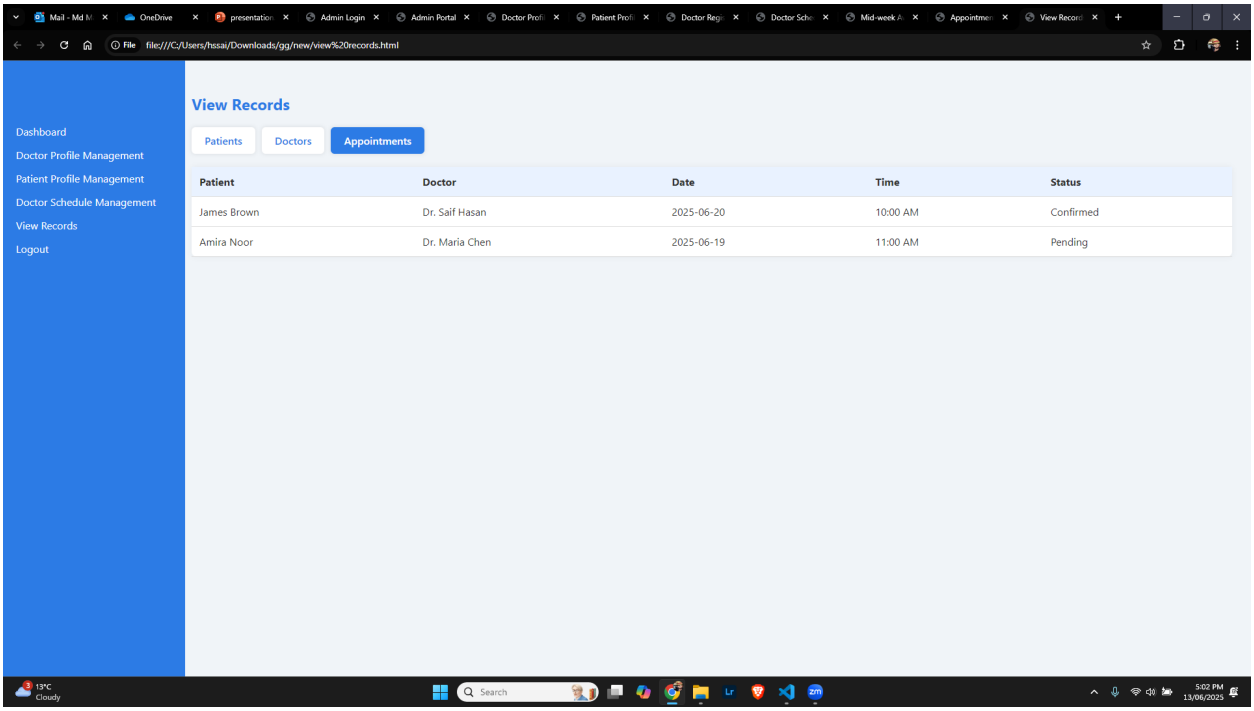
Create/Update/Manage Schedule:


Mail - Md M
OneDrive
presentation
Admin Login
Admin Portal
Doctor Profi
Patient Profi
Doctor Regi
Doctor Sche
Mid-week A
Appointmen
View Record
file:///C:/Users/hssai/Downloads/gg/new/Doctor%20Schedule%20Management.html
Dashboard
Profile Management
Manage Schedules
View Records
Logout
Doctor Schedule Management
Select Doctor
-- Select Doctor --
Schedule Date
dd/mm/yyyy
Start Time
--:-- --
End Time
--:-- --
Create Schedule
Work Roster
Doctor
Date
Start Time
End Time
Location
Dr. Saif Hasan
2025-06-17
09:00 AM
03:00 PM
Room 101
Dr. Aisha Khan
2025-06-17
10:00 AM
04:00 PM
Room 202
Dr. John Lee
2025-06-18
08:00 AM
02:00 PM
Room 303
Dr. Maria Chen
2025-06-18
11:00 AM
05:00 PM
Room 404
Dr. Ahmed Rahman
2025-06-19
09:30 AM
03:30 PM
Room 505
13°C
Cloudy
Search
5:02 PM
13/06/2025


Mail - Md M
OneDrive
presentation
Admin Login
Admin Portal
Doctor Profi
Patient Profi
Doctor Regi
Doctor Sche
Mid-week A
Appointmen
View Record
file:///C:/Users/hssai/Downloads/gg/new/Mid-week%20Availability%20Requests.html
Dashboard
Profile Management
Manage Schedules
View Records
Logout
Mid-week Availability Requests
Doctor
Requested Date
New Availability
Reason
Action
Dr. Saif Hasan
2025-06-15
10:00 AM - 3:00 PM
Personal reasons
Approve
Reject
Dr. Maria Chen
2025-06-17
9:00 AM - 12:00 PM
Conference attendance
Approve
Reject
13°C
Cloudy
Search
5:02 PM
13/06/2025

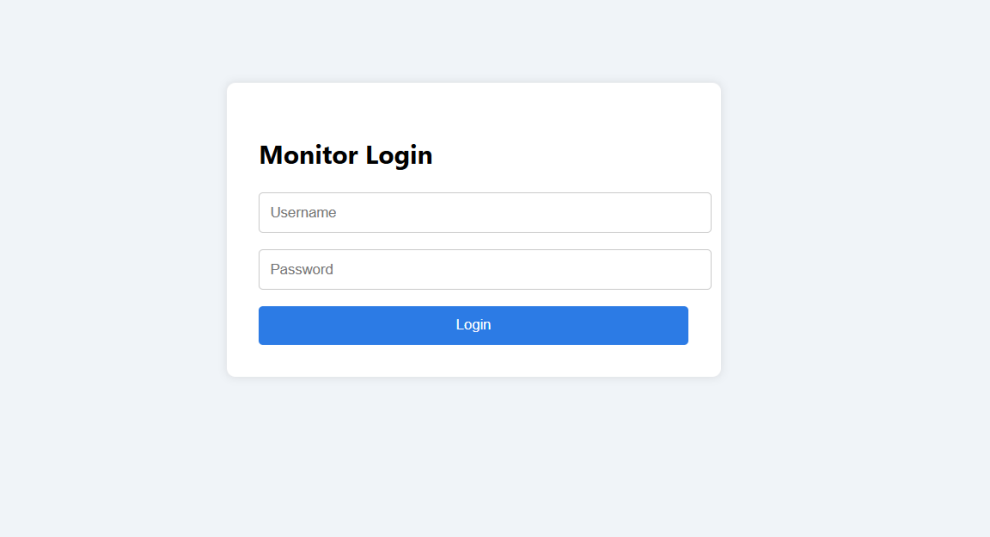


Admin View Record:

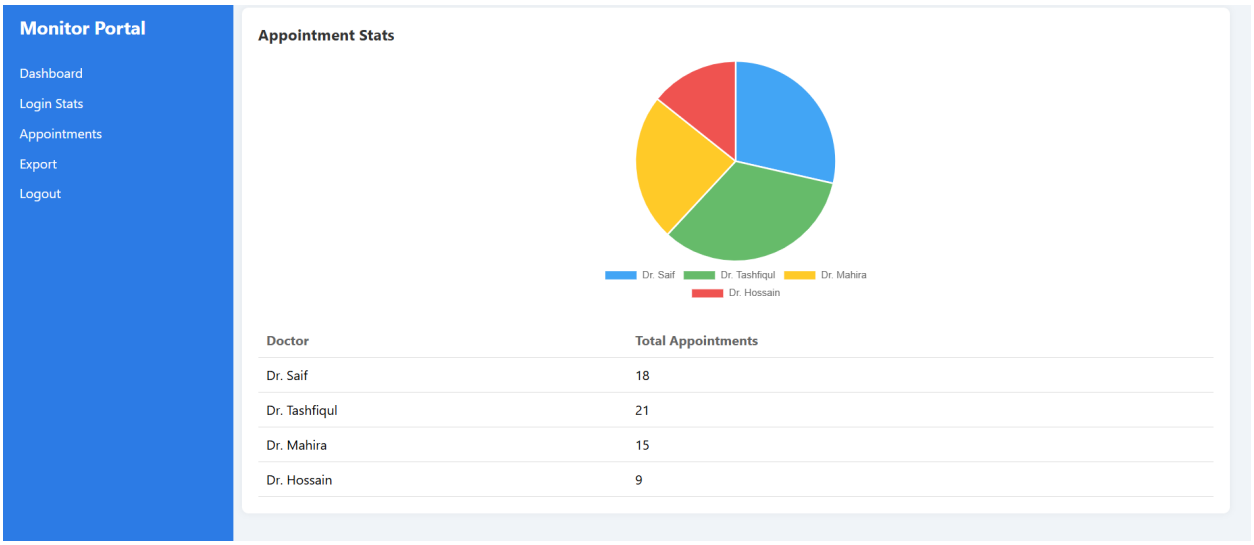


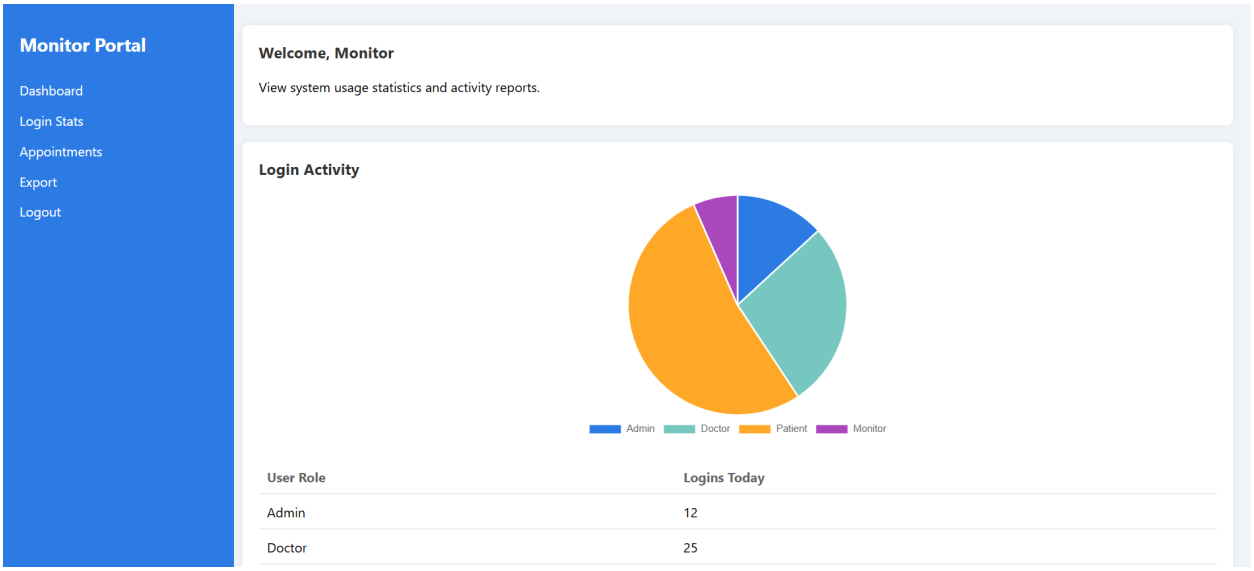
UI design for monitor (Arnob):

Login Portal:

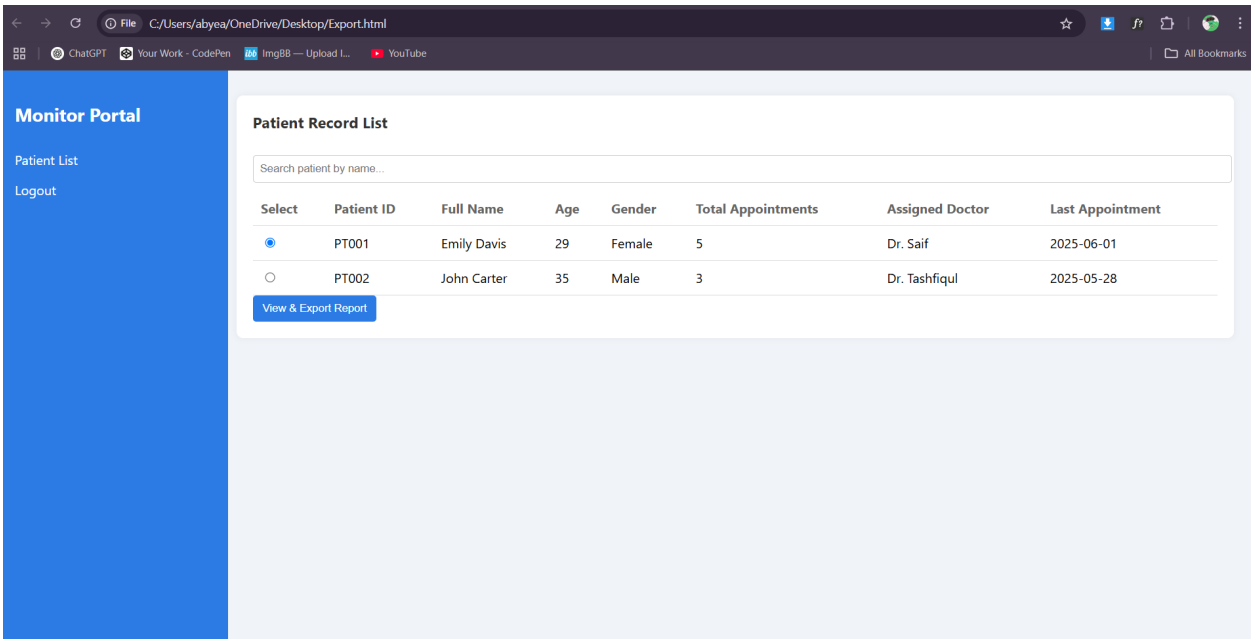


Dashboard:





View Patient list:



Exprot Report:

Risk Management for MediAccessHub

Patient portal (Prateek):

Risk ID	Description	Likelihood	Impact	Risk Level	Category	Mitigation Strategy	Owner
P1	Patient enters incorrect login credentials	Medium	Medium	Medium	User	Add input validation, lockout after multiple failed attempts, and show error hints	Prateek (Patient)
P2	Medical report fails to load or display incorrectly	Low	High	Medium	Technical	Implement fallback data retrieval and display error message to user	Tashfiqul (Doctor)
P3	User submits appointment request for unavailable doctor or past time	High	Medium	High	Functional	1. Implement dynamic calendar to block unavailable dates. 2. Validate input fields in real-time before submission.	Arnob (Monitor)
P4	User with poor digital literacy finds UI difficult to navigate	Medium	Medium	Medium	Usability	1. Use larger buttons, icons, and text with clear labels. 2. Include short tooltips or onboarding steps during first login.	Prateek
P5	Patient accesses another user's data due to a session/token mix-up	Low	High	High	Security	1. Use JWT tokens with unique session IDs. 2. Apply strict backend verification on all data requests.	Saif (Admin)

Doctor (Tashfqul)

Risk ID	Description	Likelihood	Impact	Risk Level	Category	Mitigation Strategy	Owner
D1	Doctor forgets to update availability schedule	Medium	Medium	Medium	Functional	Implement a reminder notification and visual indicator on dashboard for pending schedule updates	Doctor
D2	Medical record update fails or is lost	Low	High	Medium	Technical	Use auto-save functionality, confirmation prompts, and periodic backend sync	Doctor / Dev
D3	Unauthorized access to patient records	Low	High	Medium	Security	Enforce secure login, use encrypted data access, and restrict access using roles	System Admin
D4	UI confusion delays doctor's task flow	Medium	Medium	Medium	Usability	Design a minimal UI with clear call-to-action buttons and tooltips for functions	UI Designer
D5	System downtime during patient check-up	Low	High	Medium	Technical	Add offline fallback (read-only), auto-reconnect messages, and sync data when online	DevOps Team

D6	Doctor misidentifies patient record	Medium	High	High	User	Include patient photo, DOB, and auto-highlight active patient record in UI	Doctor
D7	Missed appointment due to portal error	Low	High	Medium	Functional	Include real-time sync with appointment database and error logs; alert admin on sync issues	Developer
D8	Doctor edits a record without confirmation	Medium	Medium	Medium	Usability	Add warning prompts before editing or deleting sensitive data	Developer

Admin (Saif):

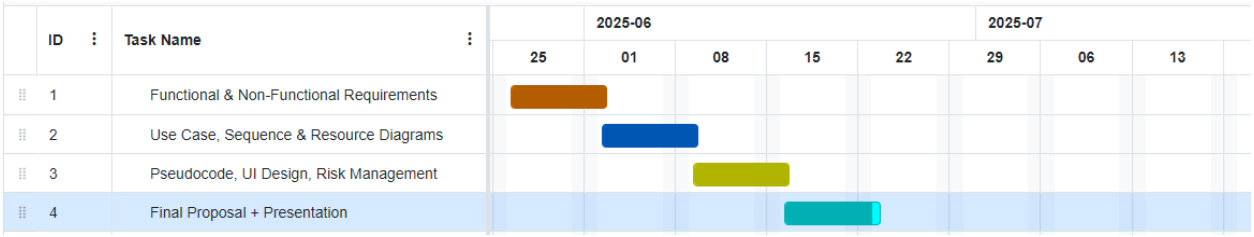
Risk ID	Description	Likelihood	Impact	Risk Level	Mitigation Strategy	Owner
A1	Delay in creating Doctor schedule	Medium	High	High	Set automated reminders for admins to review availability and implement a queue management dashboard highlighting pending approvals	Saif
A2	Unauthorized access attempt to the admin portal could lead to data breaches	Low	High	High	Implement two-factor authentication and strong password policies. Lock account after multiple failed attempts	Saif
A3	Appointment change request pile up during peak hours	High	Medium	Medium	Assign backup admin support during peak period and auto flag urgent request for quick review	Saif
A4	System fails to sync admin updates across all modules in real time	Medium	High	High	Implement auto sync services and conduct daily integration tests to catch sync issues early	Saif
A5	Admin accidentally deactivates an active user	Low	Medium	Low	Add confirmation popups for deactivate actions and maintain an audit trail for recovery	Saif

Monitor (Arnob):

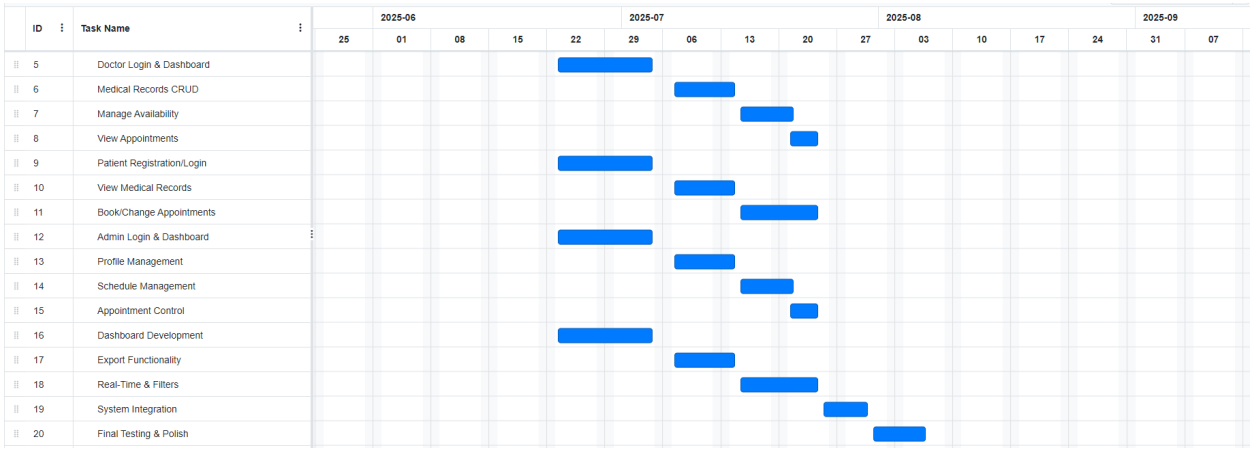
Risk ID	Description	Likelihood	Impact	Risk Level	Mitigation Strategy	Owner	Risk Category
M1	Dashboard fails to update login or activity logs in real time	Medium	medium	High	Implement event-driven logging with retry logic and alerts for delays over 1 minute	Arnob	Tech.
M2	Data export fails or generates corrupted files	Medium	High	High	Use reliable export libraries with data validation and error checks before download	Arnob	Tech.
M3	System lags when monitoring high traffic data	High	Medium	Medium	Optimize backend queries and add server caching and use lazy loading in	Arnob	Tech.
M4	Missing or incomplete log data due to database error	Low	High	Medium	Schedule hourly databse backups and log integrity achecks with anomaly alerts	Arnob	Tech.
M5	Unauthorized access to monitoring or export functions	Low	High	High	Restrict access to specific roles and enable export logging and perform regular permission audits	Arnob	Tech.

TimeLine:

1. NIT3003

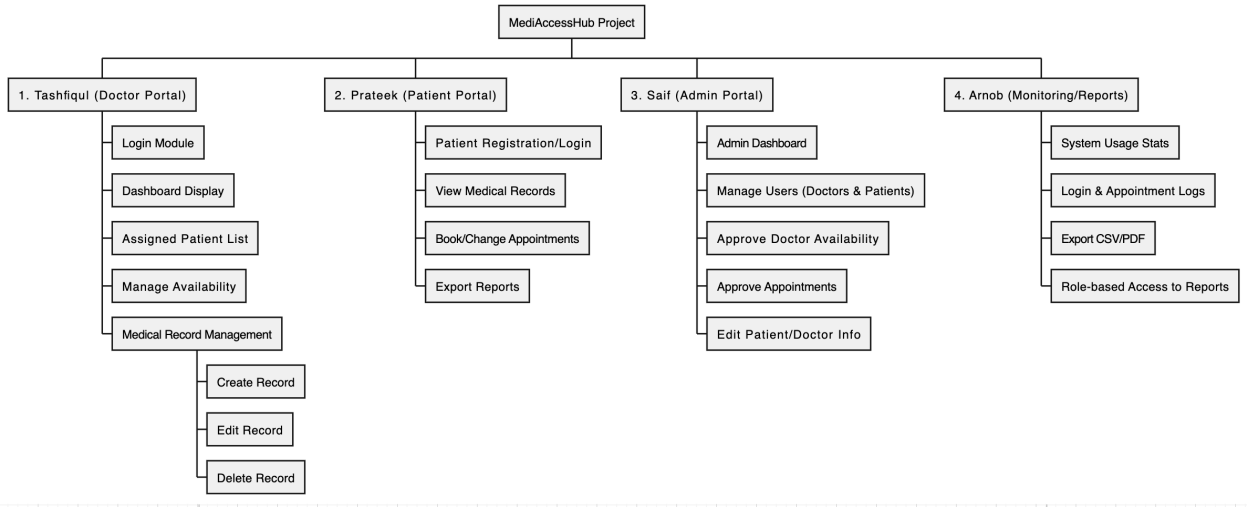


2. Implementation of 8 weeks



3. NIT3004

Work Breakdown Structure:



Conclusion:

Our project for MediAccessHub presents a secure user friendly and comprehensive web application for managing digital medical records and booking appointments with doctors. By dividing roles across patient, doctor, admin and monitor between the team, the system ensures privacy, data integrity and smooth interaction between users. Each model for this project was thoughtfully designed to address real world healthcare challenges through a collaborative and modular approach MediAccessHub stands out as a scalable and secure platform that empowers users and improves the overall efficiency of healthcare delivery.

Contribution:

Name	Student ID	Task	Contribution
Md Mahmudul Hasan Saif	s8089637	Saif was responsible for Admin portal and all task related to admin portal was done by Saif as well as creating the report	27
Abyead Faisal Arnob	s8098828	Arnob was responsible for monitor portal and all task related to monitor portal was done by Arnob and helping to create the report	26
Tashfiqul Prodhan	s8116509	Tashfiqul was responsible for Doctor protal and all task related to doctor portal was done by tashfiqul and helping the team with group task	25
Prateek Maharjan	s8113219	Prateek was responsible with patient portal and all task related to patient portal was done by Prateek	22

References

World Health Organization. (2021). Global strategy on digital health 2020–2025. https://www.who.int/publications/i/item/9789240020924

Fortune Business Insights. (2024). Electronic health records (EHR) market size, share & trends, 2023–2030. https://www.fortunebusinessinsights.com/electronic-health-records-ehr-market-102798

Australian Digital Health Agency. (2023). My Health Record system overview. https://www.digitalhealth.gov.au/initiatives-and-programs/my-health-record

PracticeHub. (2023). Features and solutions. https://practicehub.com.au

Doxy.me. (2023). Simple, free, and secure telemedicine solution. https://doxy.me/en/

Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education. https://books.google.com/books/about/Software_Engineering.html?id=OhL7CgAAQBAJ