Design Document

for

DocuMed

Version 1.0

Prepared by

Group 11 Group Name: Ctrl+C Ctrl+V

Shah Divit Ritesh	220995	divits22@iitk.ac.in
Devansh Bansal	220341	bdevansh22@iitk.ac.in
Shaurya Sharma	221007	shauryas22@iitk.ac.in
Nilay Agarwal	220714	nilayag22@iitk.ac.in
Ruthvik Tunuguntla	220924	ruthvikjs22@iitk.ac.in
Aarsh Ashish Walavalkar	220013	aarshaw22@iitk.ac.in
Purav Jangir	220837	puravj22@iitk.ac.in
Ankit Kaushik	220158	kankit22@iitk.ac.in
Bhaumik Chawda	220292	bhaumik22@iitk.ac.in
Tanush Goel	221131	tanushg22@iitk.ac.in
Devansh Agarwal	220340	devanshag22@iitk.ac.in

Course: CS253

Mentor TA: *Somesh Vas*

CONTENTS ....

REVISIONS ....

1 CONTEXT DESIGN ....

1.1 CONTEXT MODEL ....

1.2 HUMAN INTERFACE DESIGN ....

2 ARCHITECTURE DESIGN ....

3 OBJECT-ORIENTED DESIGN ....

3.1 USE CASE DIAGRAM ....

3.2 CLASS DIAGRAM ....

3.3 SEQUENCE DIAGRAM ....

3.4 STATE DIAGRAM ....

4 PROJECT PLAN ....

5 OTHER DETAILS ....

APPENDIX A - GROUP LOG ....

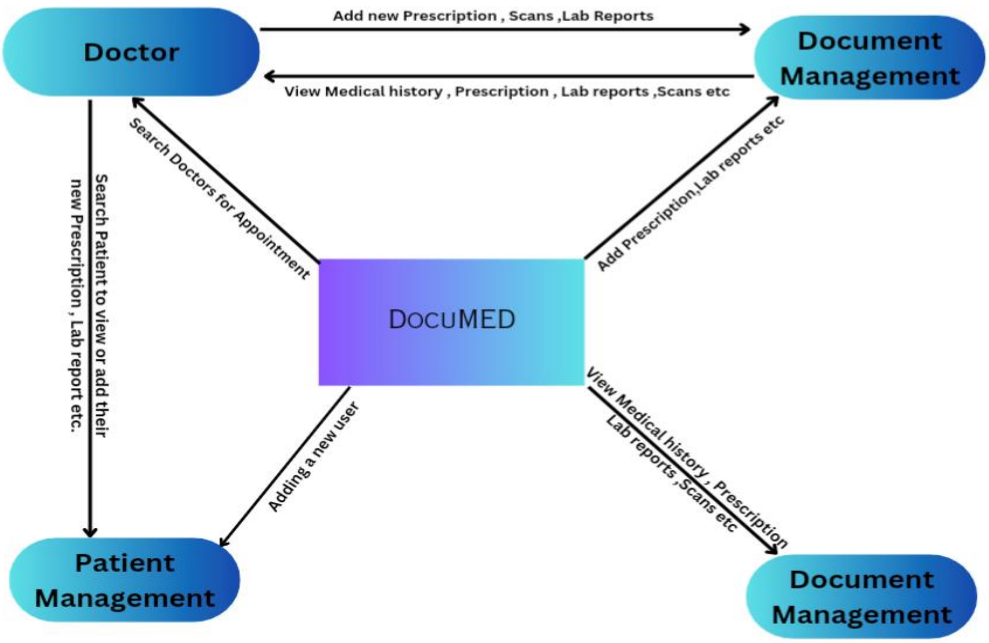
Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Tanush Goel Ruthvik Tunuguntla Devansh Bansal Devansh Agarwal Ankit Kaushik Nilay Agarwal Shaurya Sharma Shah Divit Ritesh Purav Jangir Bhaumik Chawda Aarsh Walavalkar	First Draft	09/02/24

1 Context Design

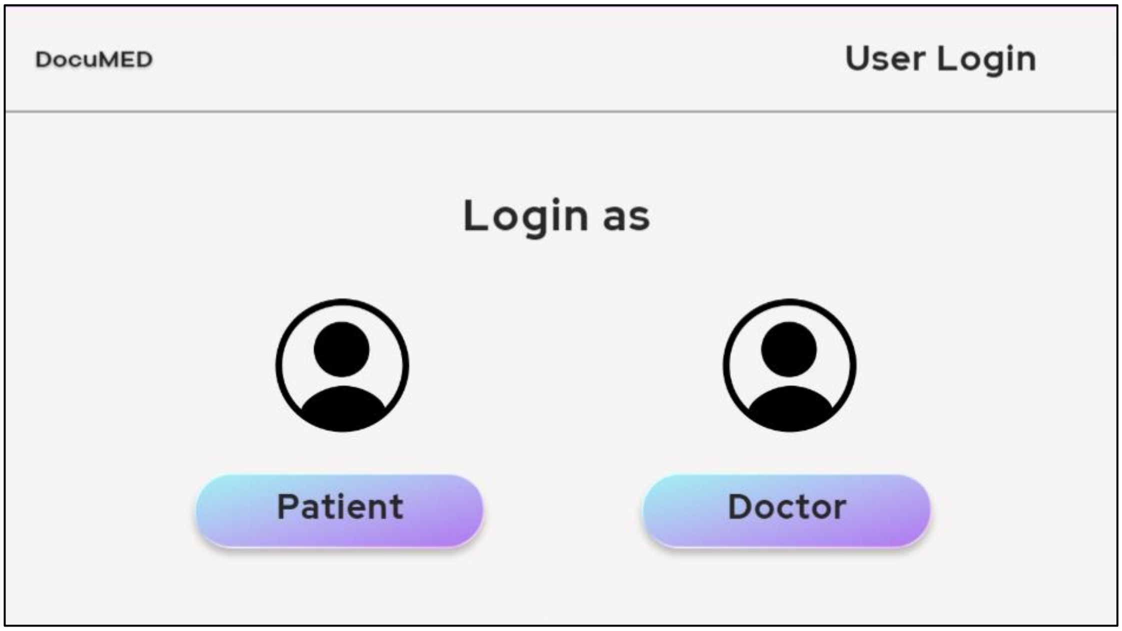
1.1 Context Model

The context model shows the various aspects of the system which interact with each other. The entities are doctor, patient and the management system for both types of user. The database stores all the data of the system. There will be an OTP API which sends unique codes for login and registration purposes.

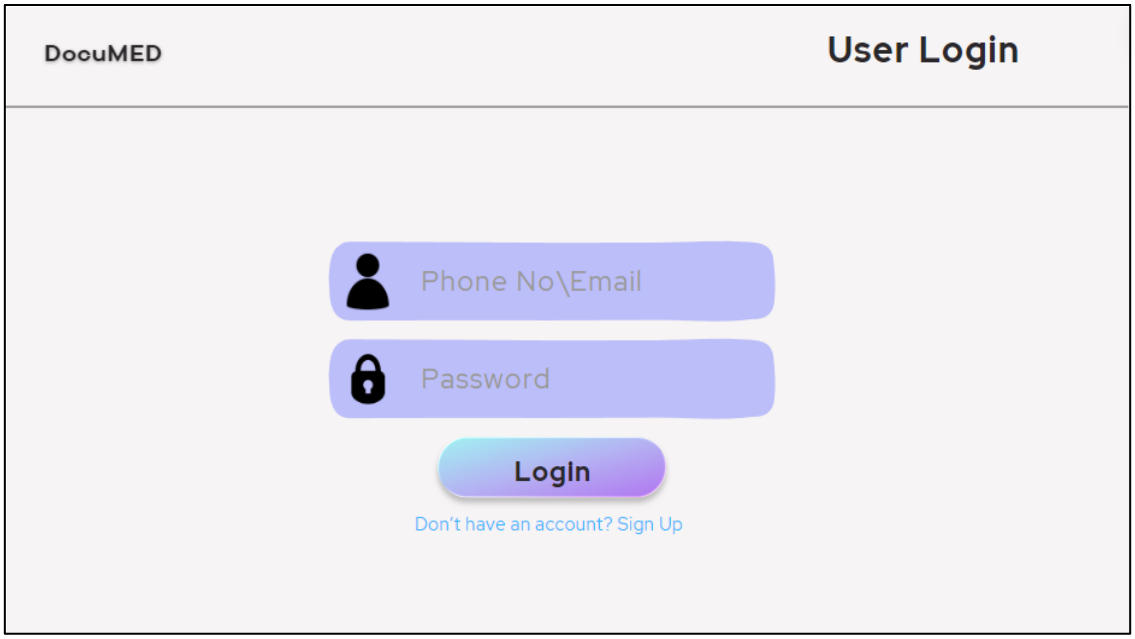

Doctor
Add new Prescription , Scans ,Lab Reports
View Medical history , Prescription , Lab reports ,Scans etc
Document Management
Search Doctors for Appointment
Add Prescription,Lab reports etc
Search Patient to view or add their new Prescription , Lab report etc.
DOCUMED
Adding a new user
View Medical history , Prescription Lab reports ,Scans etc
Patient Management
Document Management

1.2 Human Interface Design

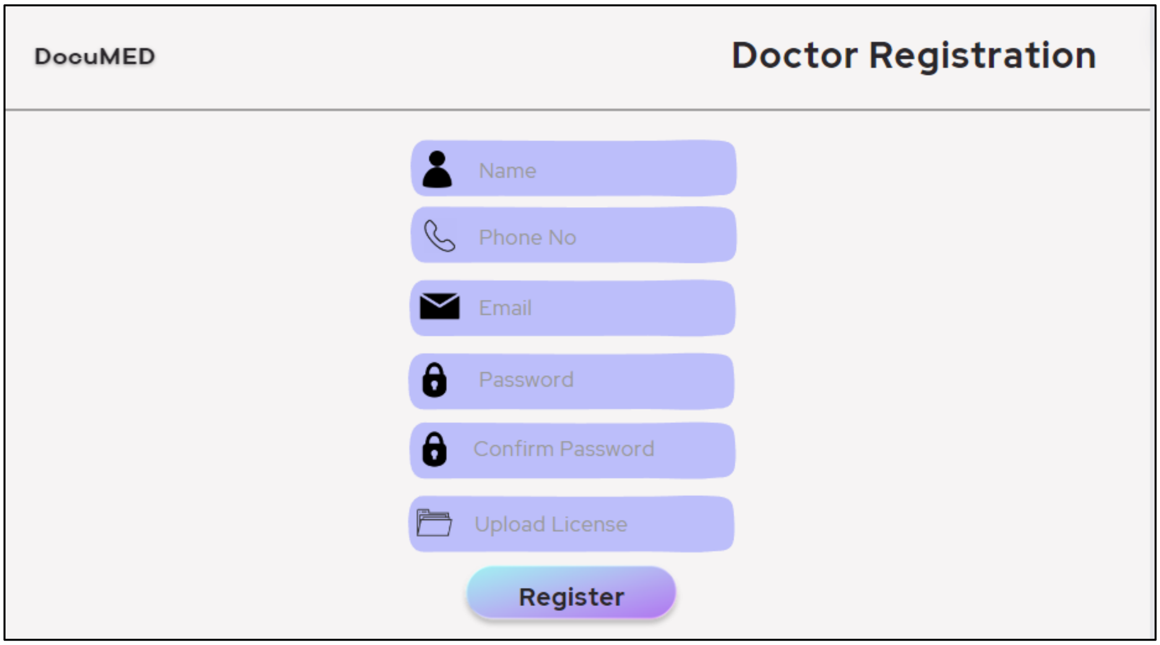
Landing page
user_type(is_doctor, is_patient) *User selects whether they are a doctor or patient*



Sign in page
verify_credentials(phone no, password) *For both doctors and patients to log in*

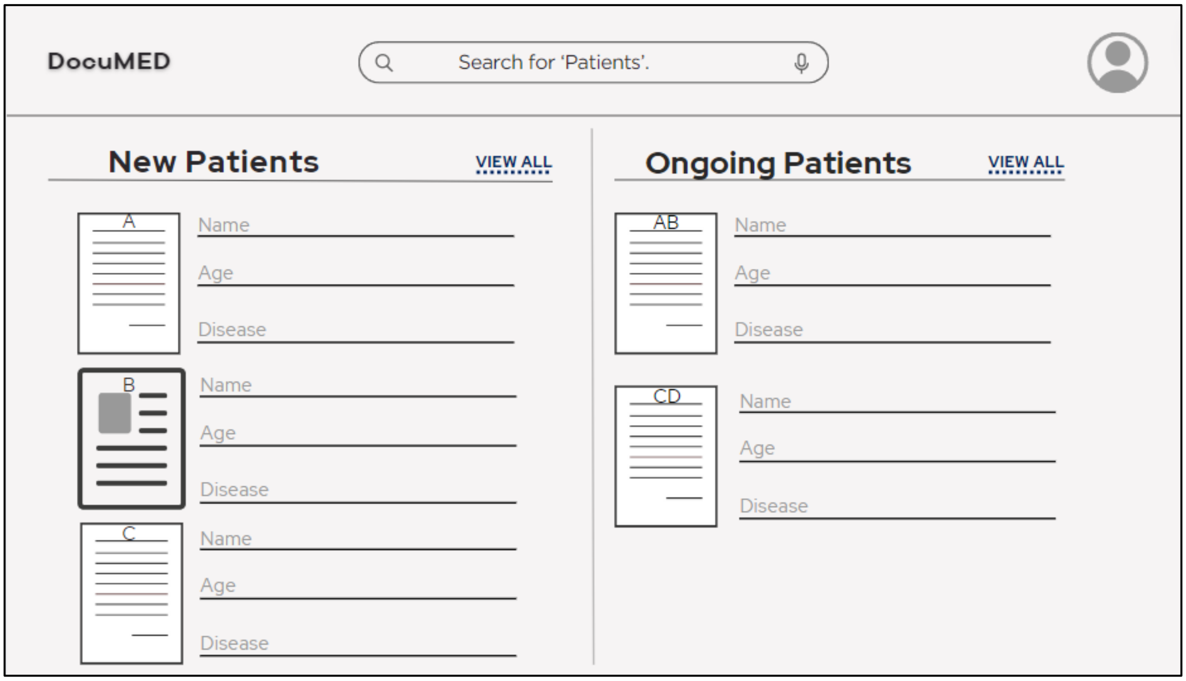


Doctor register page
doctor_id = new_doctor(doctor_name, password, phone no., email_id, license_image) *Create new account* **doctor_details(doctor_id, doctor_specialization , hospital_associated_with, working_days, working_hours)** *Doctor adds details about practice* **verify_license(doctor_id, license_image)** *Doctors verify their medical license*



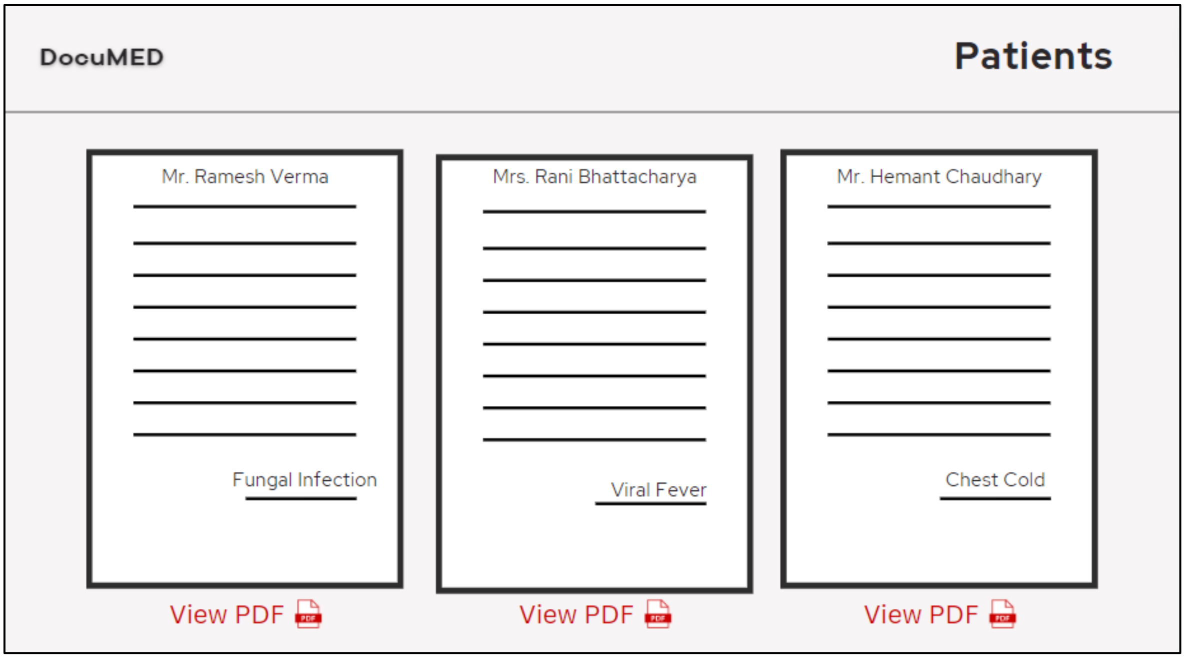
Doctor home page

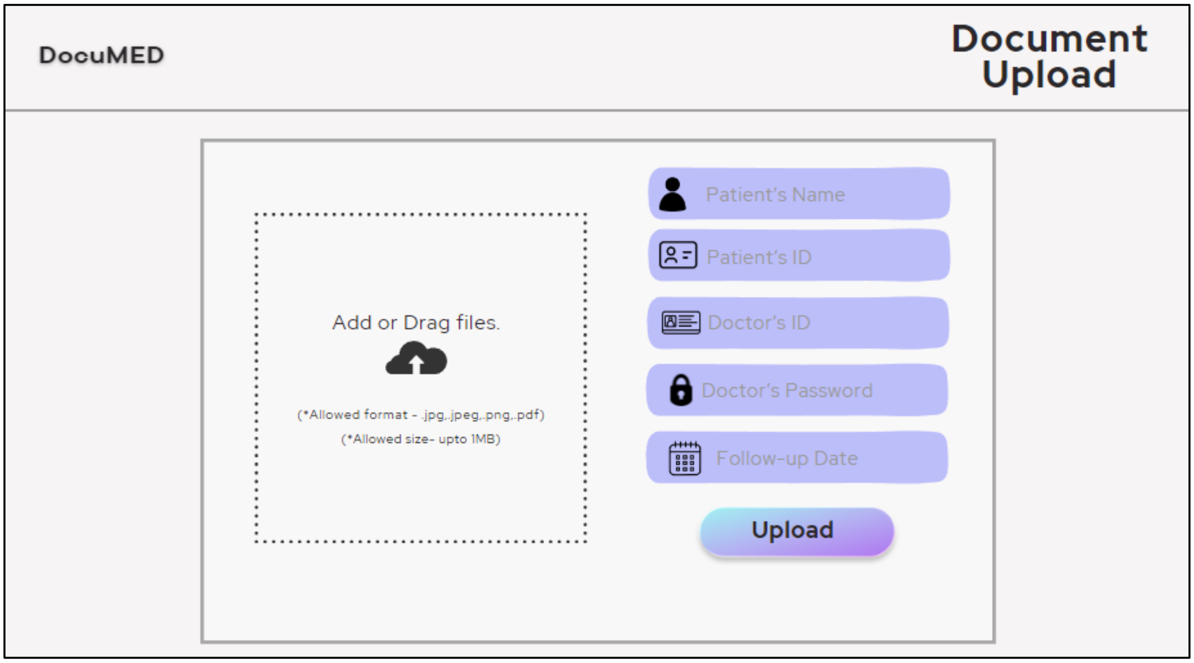
update_password(doctor_id, password, new_password)
Doctors can update their password
 	
update_timing(doctor_id, working_days, working_hours)
Doctors could edit their working days and working hours.
 	
view_patient_docs(patient_id, is_accessible)
Doctors can view patient documents



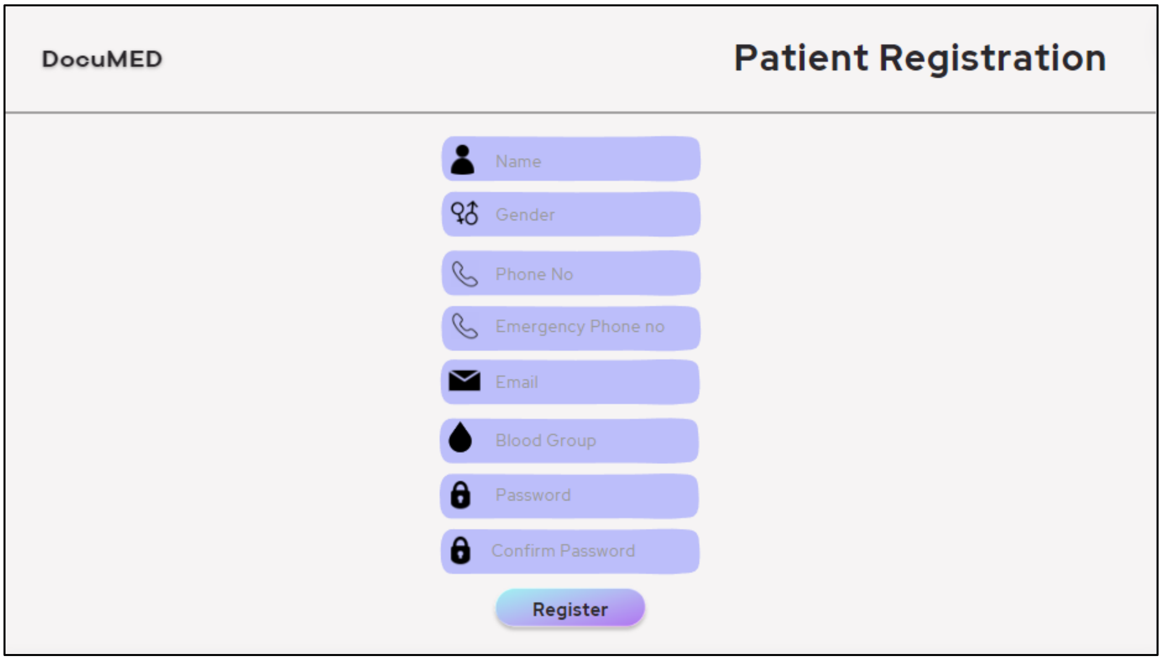
Patient view (doctor interface)

add_certificate(doctor_id, patient_id, certificate_type)
Doctors can add Medical Certificate and Disability Certificate
 	
add_scans(doctor_id, patient_id, scan_type)
Doctors can add X-Rays, MRI and Ultrasound, CT scan and PET-CT
 	
edit_prescription(doctor_id, patient_id, is_accessible)
Doctors can edit prescriptions for patients
 	
next_appointment(doctor_id,patient_id)
Doctors can schedule next appointment
 	
download_docs(patient_id)
Doctors can download the documents





Patient register page
patient_id = new_account(name, gender, blood_group, phone_no, emergency_phone_no, email, password) *Patient could create a new account.* **upload_reports(patient_id, past_report)** *Patient can upload their past reports in PDF/Image format.* **upload_prescriptions(patient_id, past_prescription)** *Patient can upload their past prescriptions in PDF/Image format.* **add_ongoing_medical_status(patient_id, current_medicines, current_ailments)** *Patient can add their current medicines and ailments.*



Patient home page

update_details(patient_id, phone_no, emergency_phone_no, email, blood_group)
Patient can change their personal details.

update_password(patient_id, old_password, new_password)
Patient can change their account password

upload_reports(patient_id, past_report)
Patient can upload their past reports in PDF/Image format.

upload_prescriptions(patient_id, past_prescription)
Patient can upload their past prescriptions in PDF/Image format.

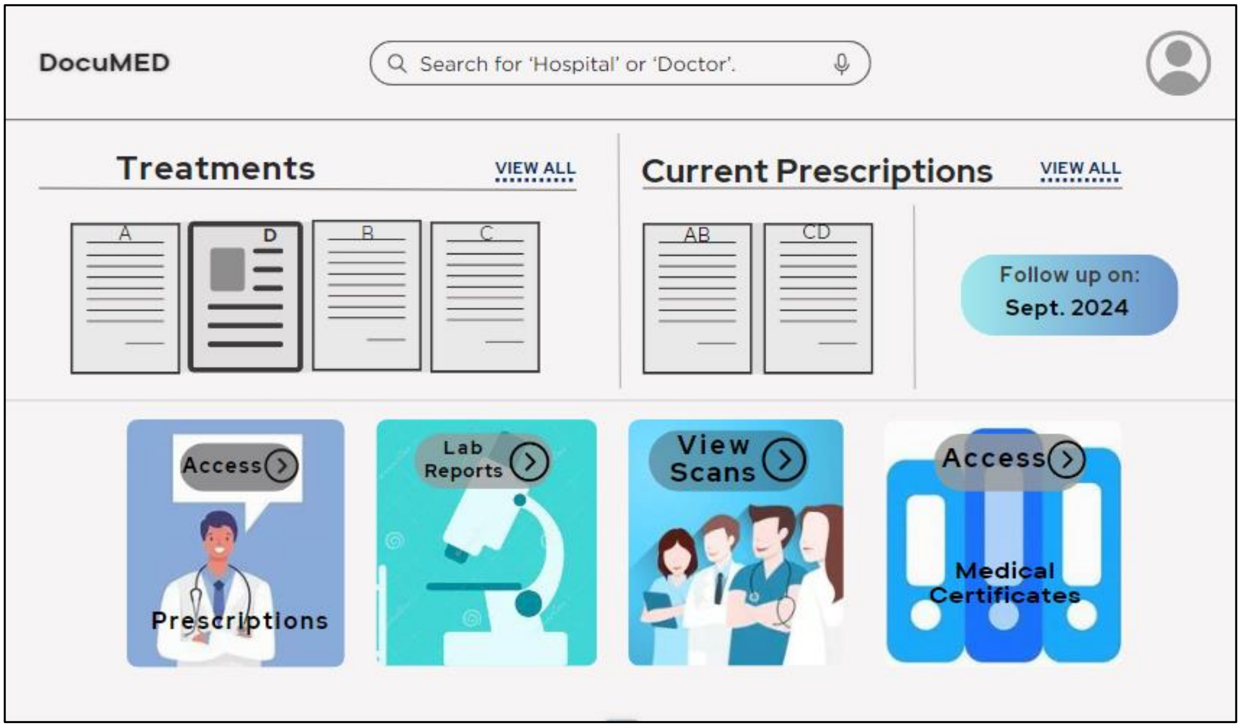
add_ongoing_medical_status(patient_id, current_medicines, current_ailments)
Patient can add their current medicines and ailments.

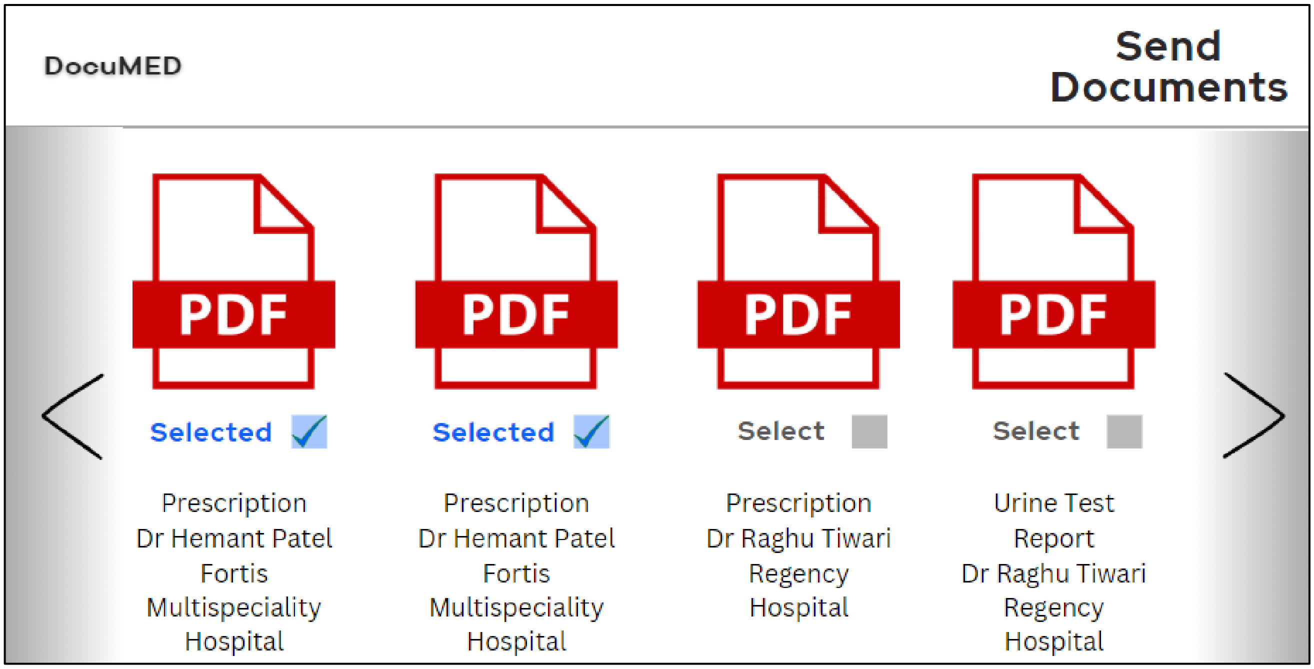
view_self_docs(patient_id, is_accessible)
Patients can view their documents and reports

find_doctor(doctor_name, hospital_associated_with, doctor_id)
Patients can search for doctors using either name or hospital or doctor_id

docs_access(patient_id, doctor_id, docs_id)
Patients can give selective document access to a particular doctor

download_docs(patient_id)
Patients can download the documents





DocuMED

Send Documents

Selected ☑

Prescription
Dr Hemant Patel
Fortis
Multispeciality
Hospital

Selected ☑

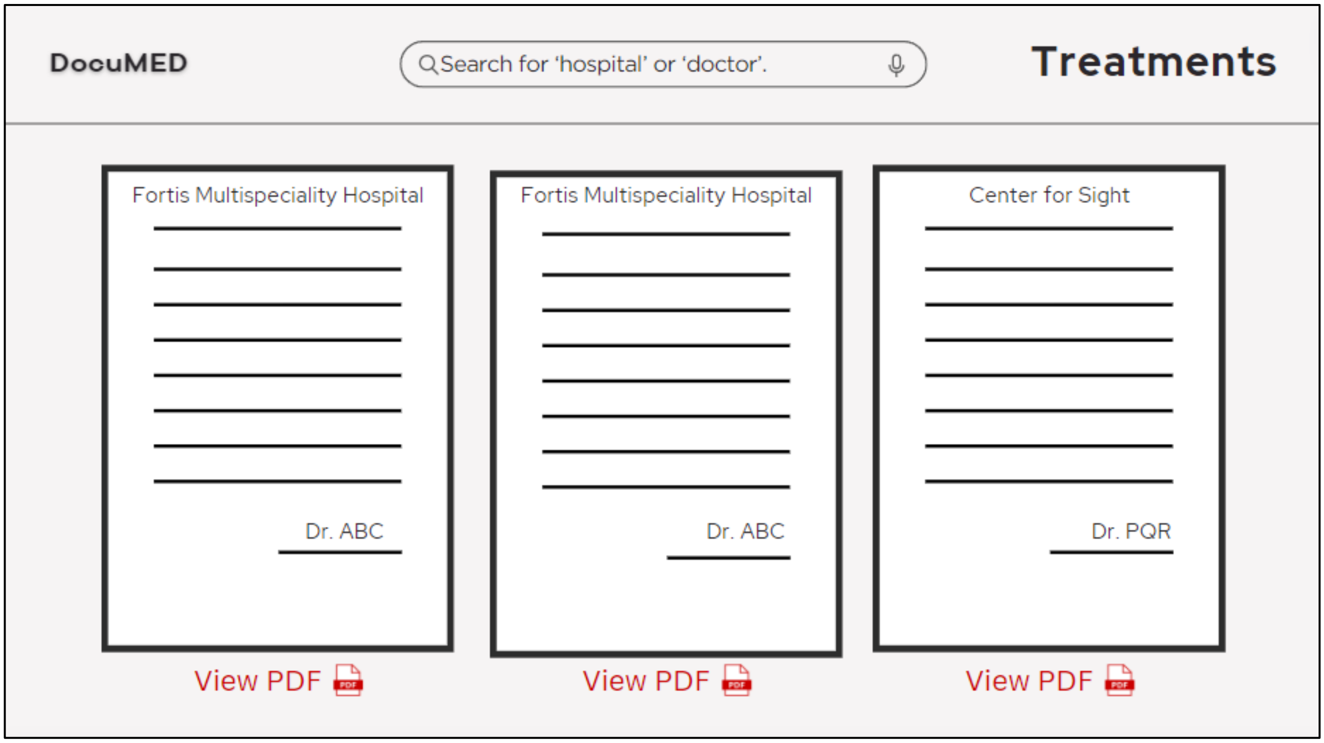
Prescription
Dr Hemant Patel
Fortis
Multispeciality
Hospital

Select □

Prescription
Dr Raghu Tiwari
Regency
Hospital

Select □

Urine Test
Report
Dr Raghu Tiwari
Regency
Hospital



DocuMED

Search for ‘hospital’ or ‘doctor’.

Treatments

Fortis Multispeciality Hospital

Dr. ABC

View PDF

Fortis Multispeciality Hospital

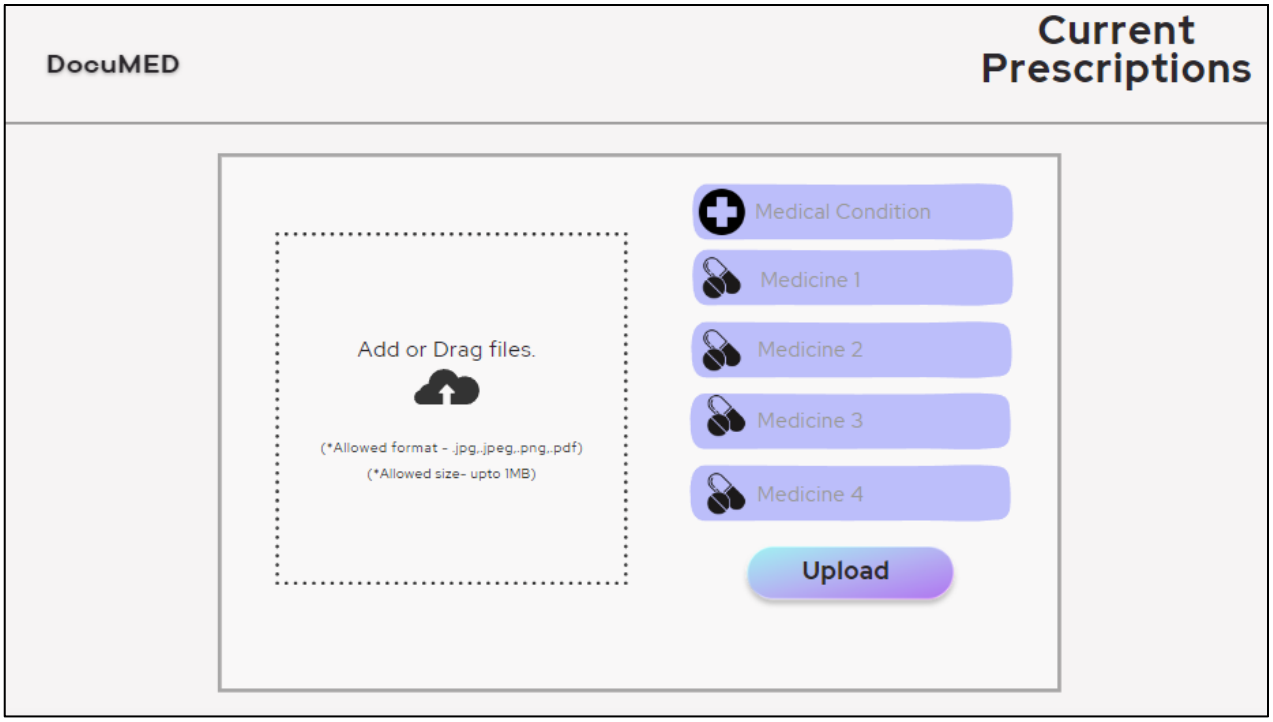
Dr. ABC

View PDF

Center for Sight

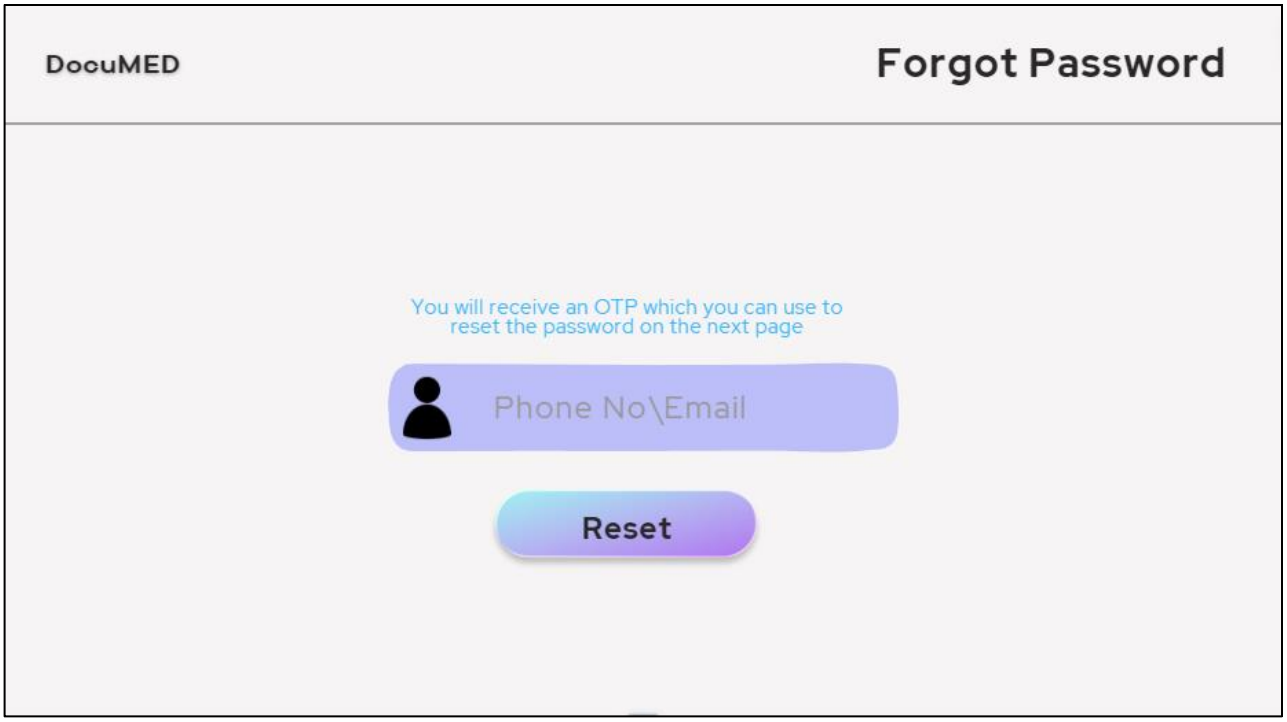
Dr. PQR

View PDF



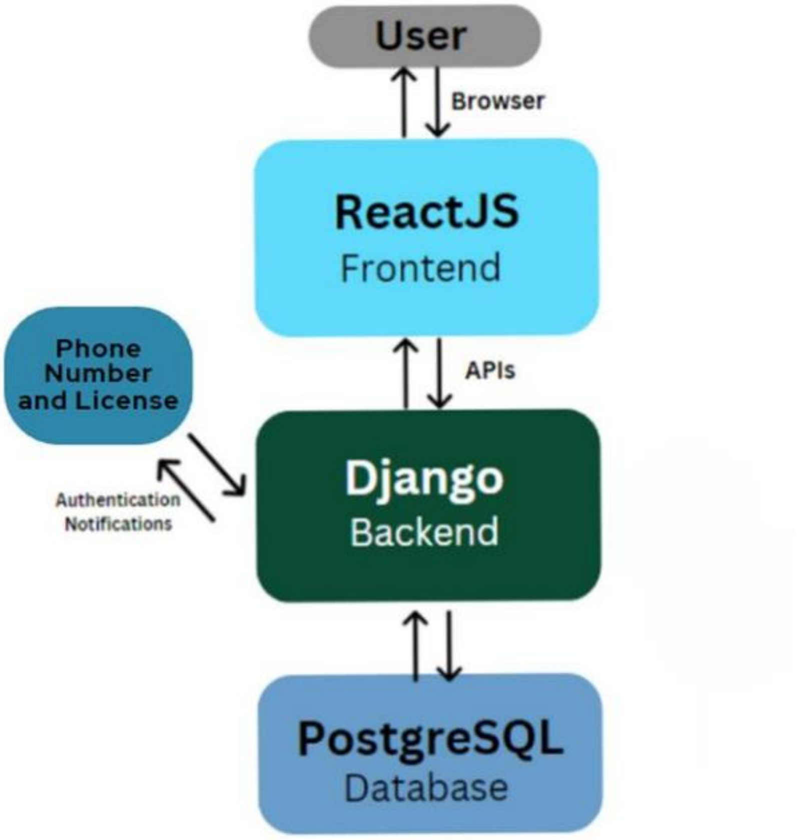
Add or Drag files.

(*Allowed format - .jpg,.jpeg,.png,.pdf)
(*Allowed size- upto 1MB)



You will receive an OTP which you can use to reset the password on the next page

2 Architecture Design



This architecture should meet the non-functional requirements as described in the SRS document as follows:

- **Scalability:** The architecture is scalable, meaning it can handle an increased number of users and data. The backend and database can be scaled horizontally by adding more instances, and the frontend can be scaled vertically by increasing the resources of the server. To implement horizontal scaling, we can increase the number of servers which are managing database handling that run our application and distribute the workload among them. This can handle more requests and tolerate failures or outages of individual servers. Apart from this, if need be, we can also use cloud-based services for managing database requests while maintaining privacy of the users.

- **Performance:** The architecture is designed for high performance while being simple to understand and implement, with a separation of concerns between frontend and backend, and the use of APIs to communicate between them, along with a powerful PostGre SQL database for data retrieval and management. This allows for optimized data access and processing.

- **Safety and Security:** Security is of prime importance in our software, since we are handling sensitive private information of the users such as medical reports and diagnostics. For this reason, our architecture implements security measures such as the protection of sensitive data and enforcing access control. This is done by devising an authentication system which verifies the phone number and license (in case of doctor) of the user and prevents unauthorised access, all of which is implemented using the backend framework.

- **Interoperability:** The given architecture uses React.js and Django for the frontend and backend respectively, both of which are extremely popular and reliable frameworks for software development, and hence compatible across almost all major browsers on the internet. This allows for smooth functioning of the software on all servers and for all users, irrespective of the devices or browsers used.

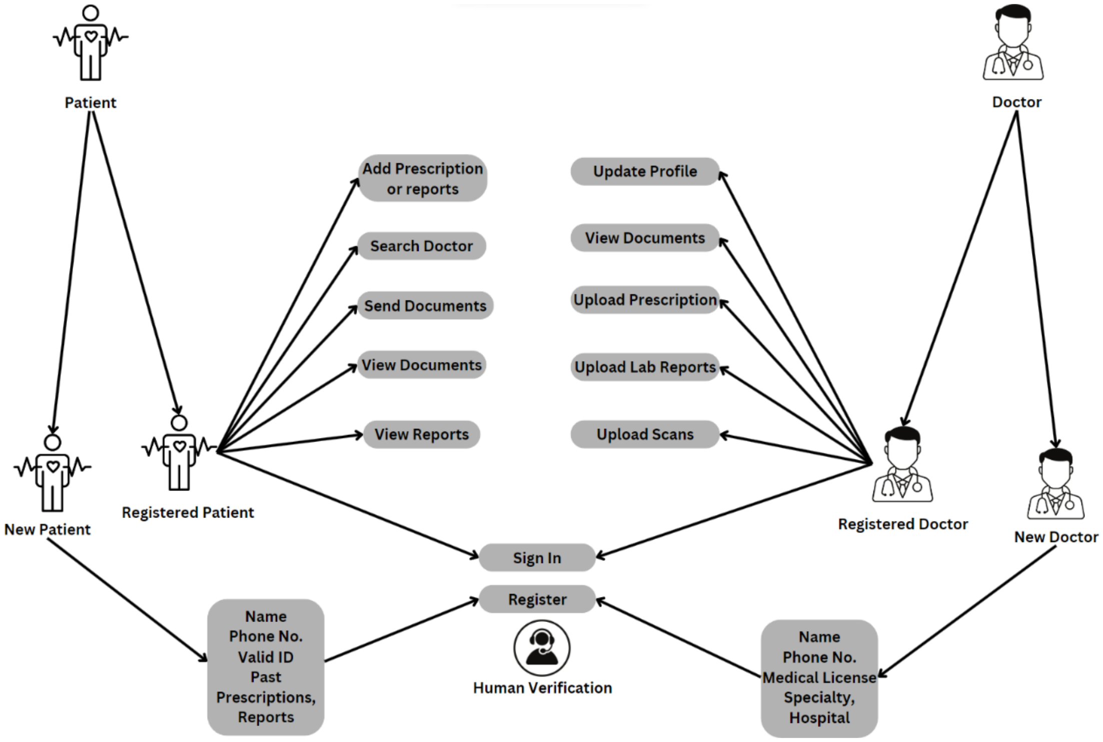
- **Reliability:** Through rigorous testing and beta testing, as well as a robust design achieved by using enhanced and flexible code quality, we plan to provide continuous integration and deployment within our architecture so that any bugs or glitches are prevented from occurring and the software works smoothly. We also plan to cover all details through thorough and up-to-date documentation, which will make it easy to use and convenient for users, while being accessible most of the time and dynamically updated as well.

3 Object Oriented Design

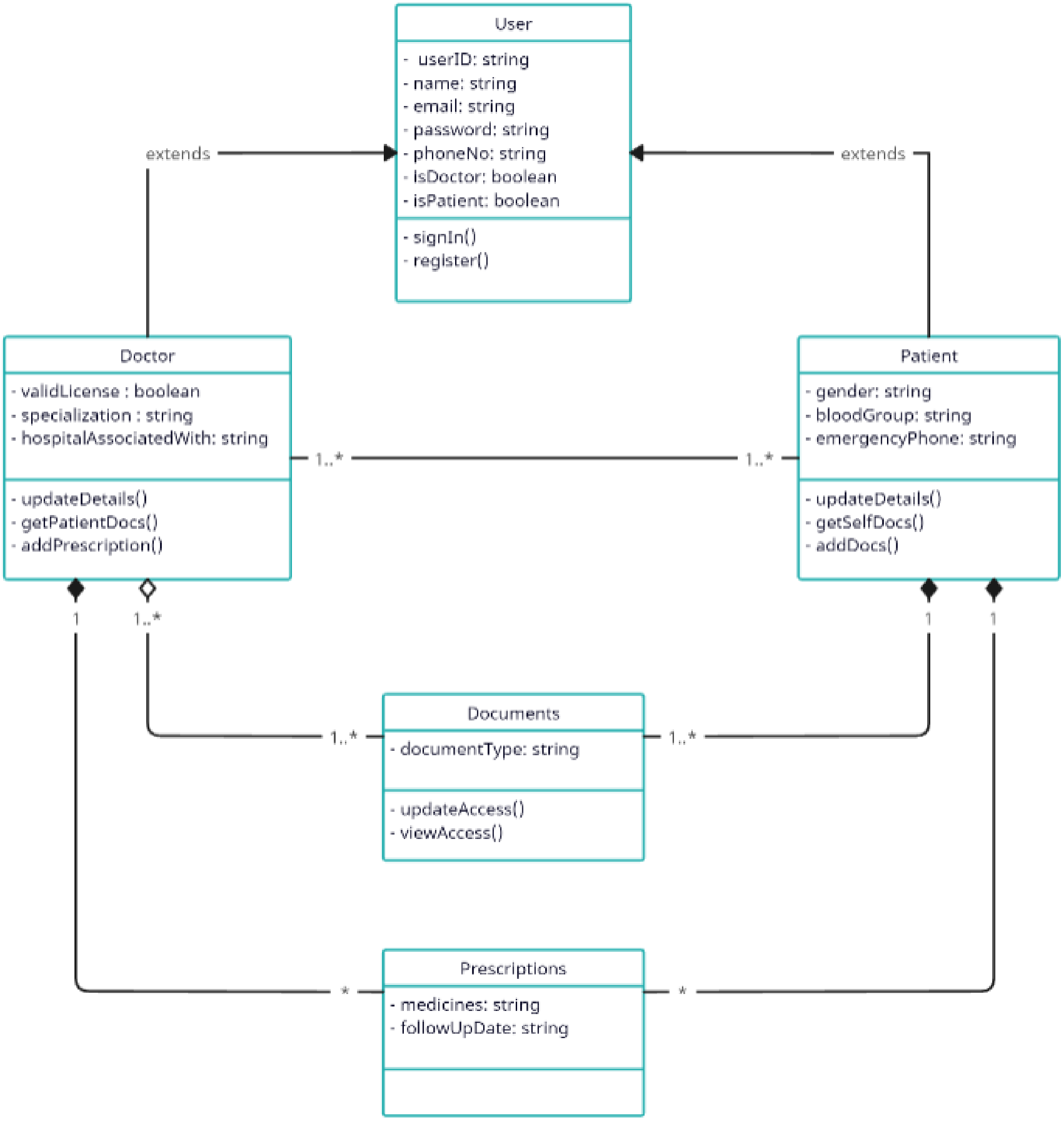
3.1 Use Case Diagrams

In the diagram shown below, we have demonstrated different use cases.

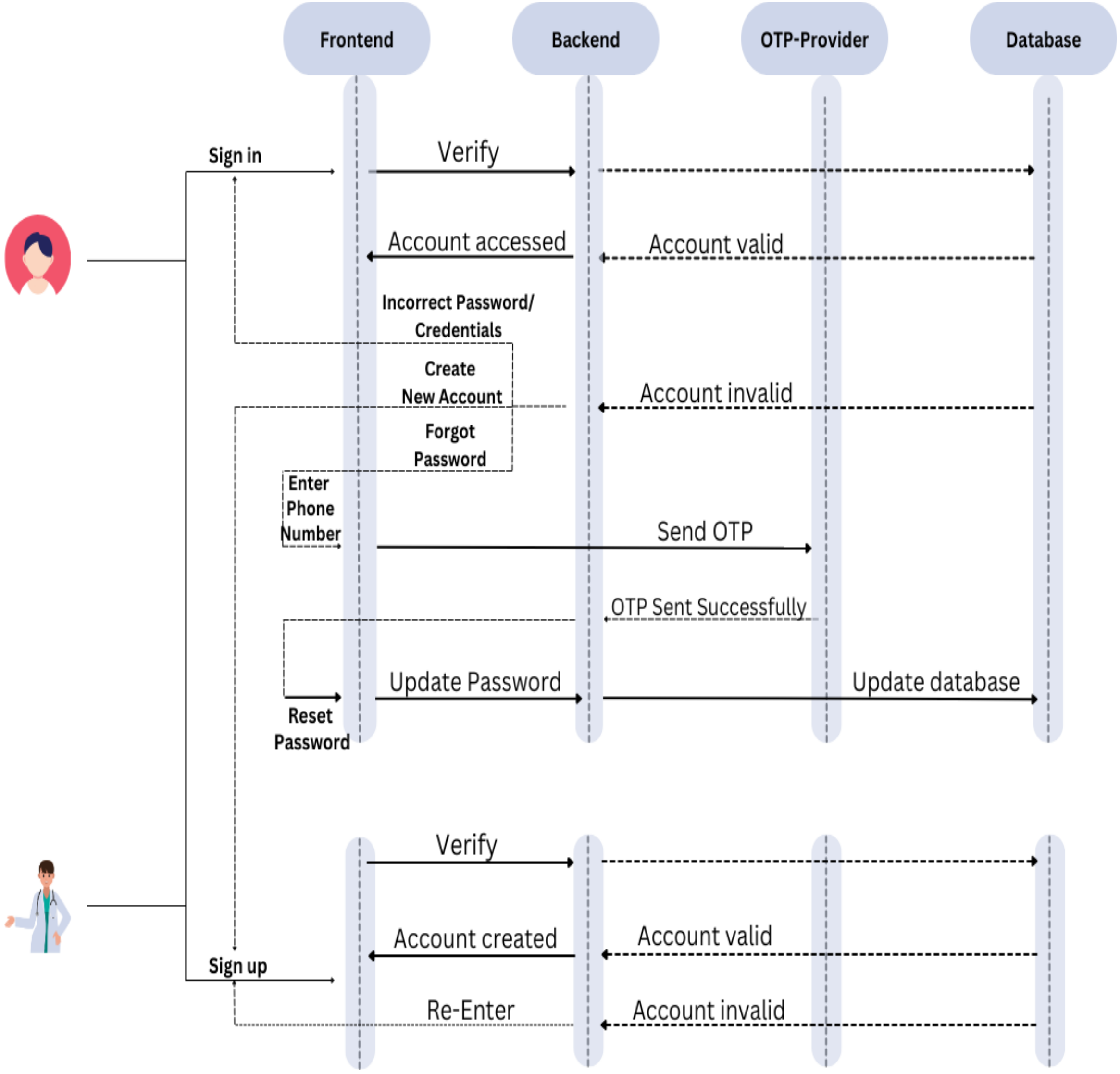
- New Patient and Doctor Registration.
- Existing Patient and Doctor sign-in.
- Patients can add prescriptions or Reports.
- Patients can search for a Doctor.
- Patients can send desired documents to the Doctor.
- Patients can view prior Documents and Reports.
- A Human Manually verifies a doctor's License.
- Doctors can update their profiles.
- Doctors can view the documents/reports given access to them by the patient.
- Doctors can update the patient's prescription.
- Doctors can upload lab reports/ scans of the patient

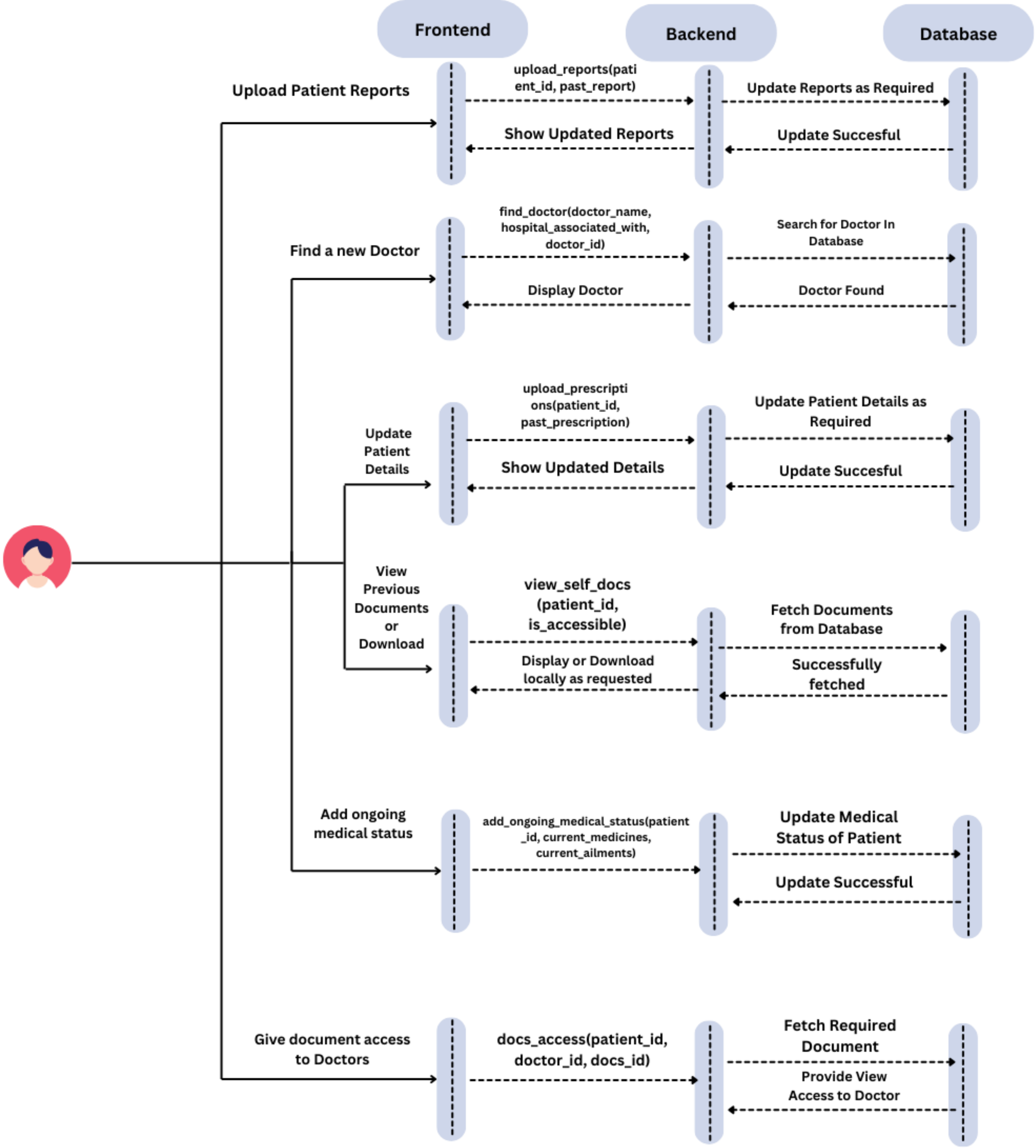


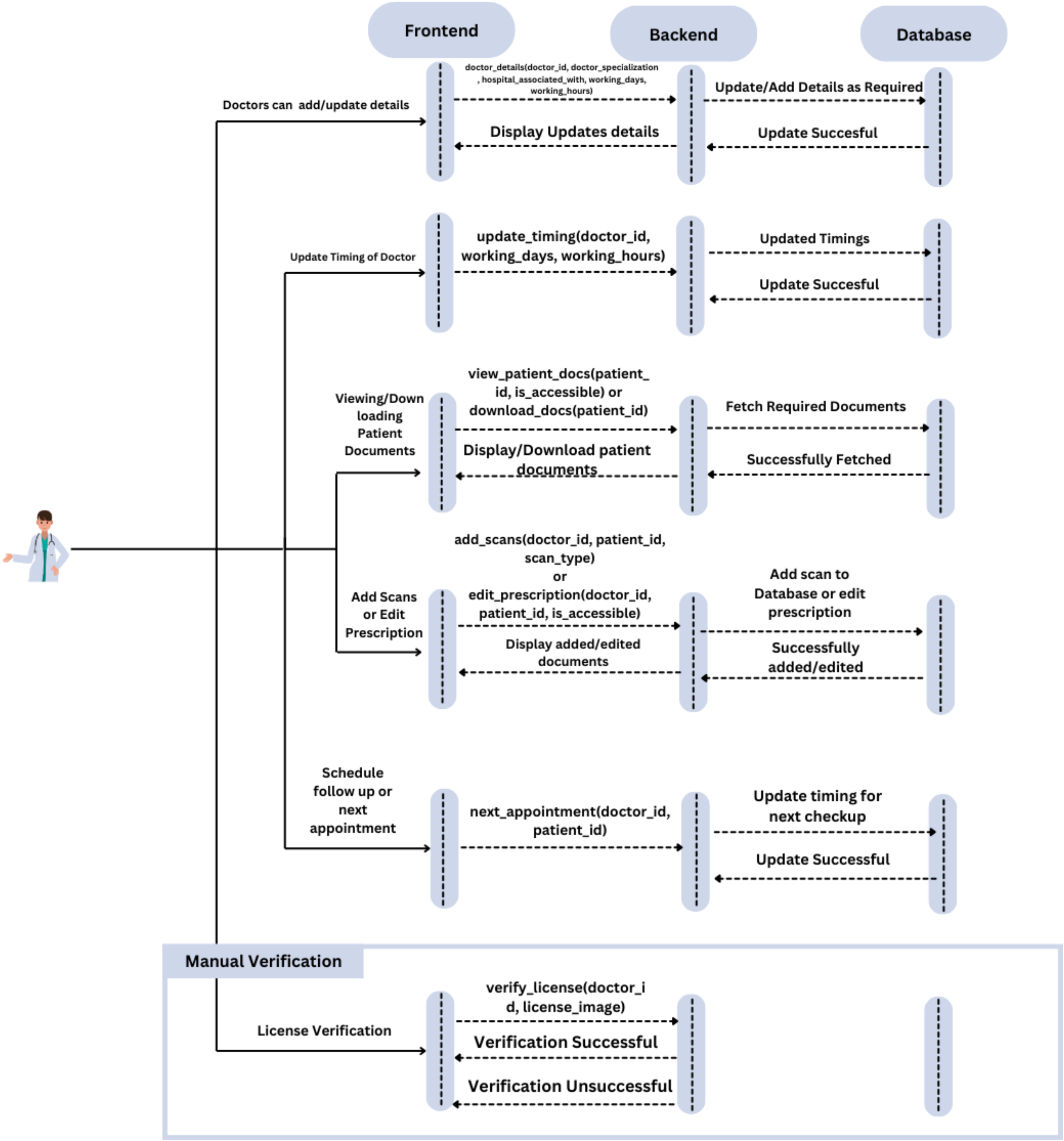
3.2 Class Diagrams



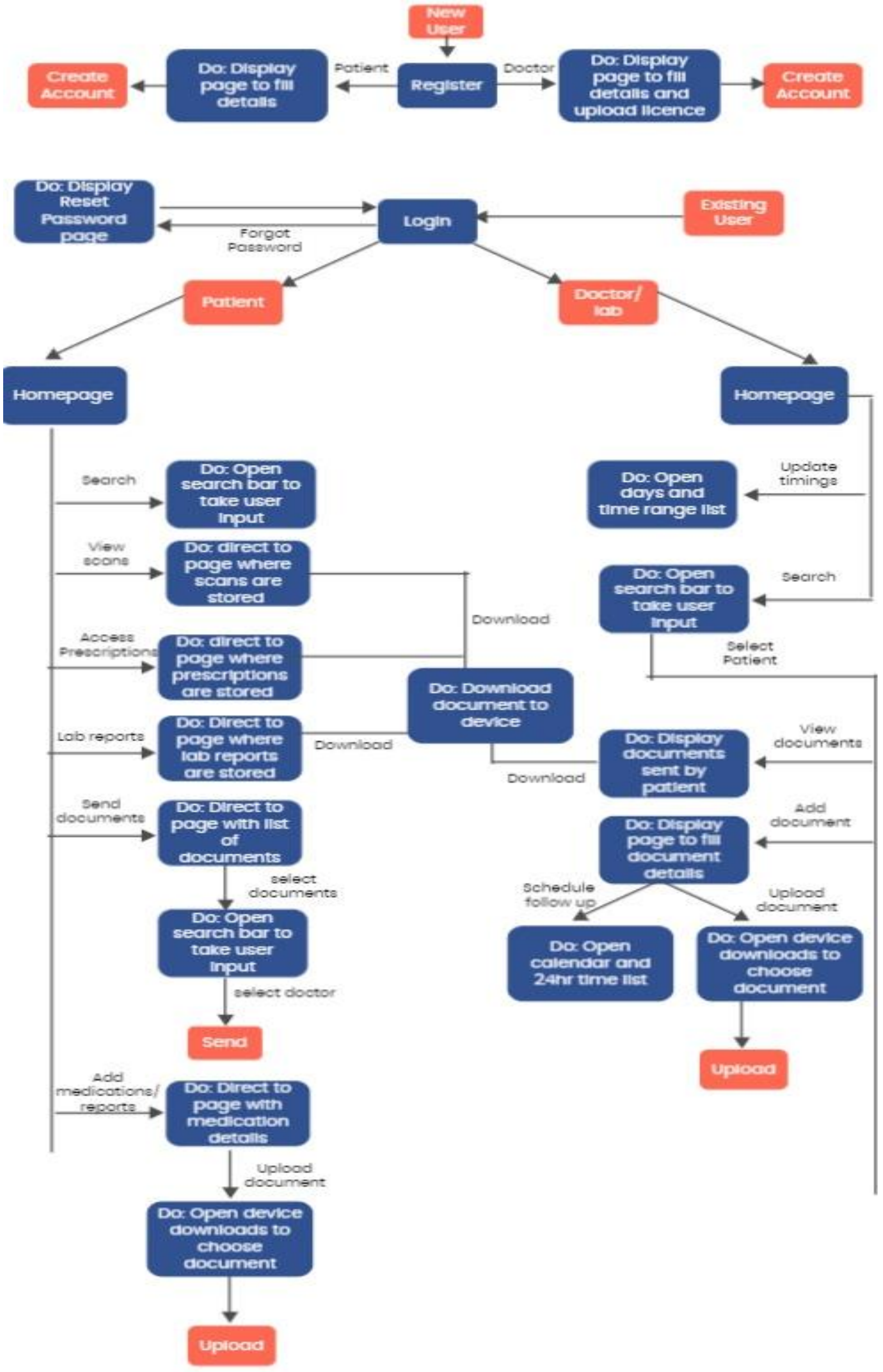
3.3 Sequence Diagrams



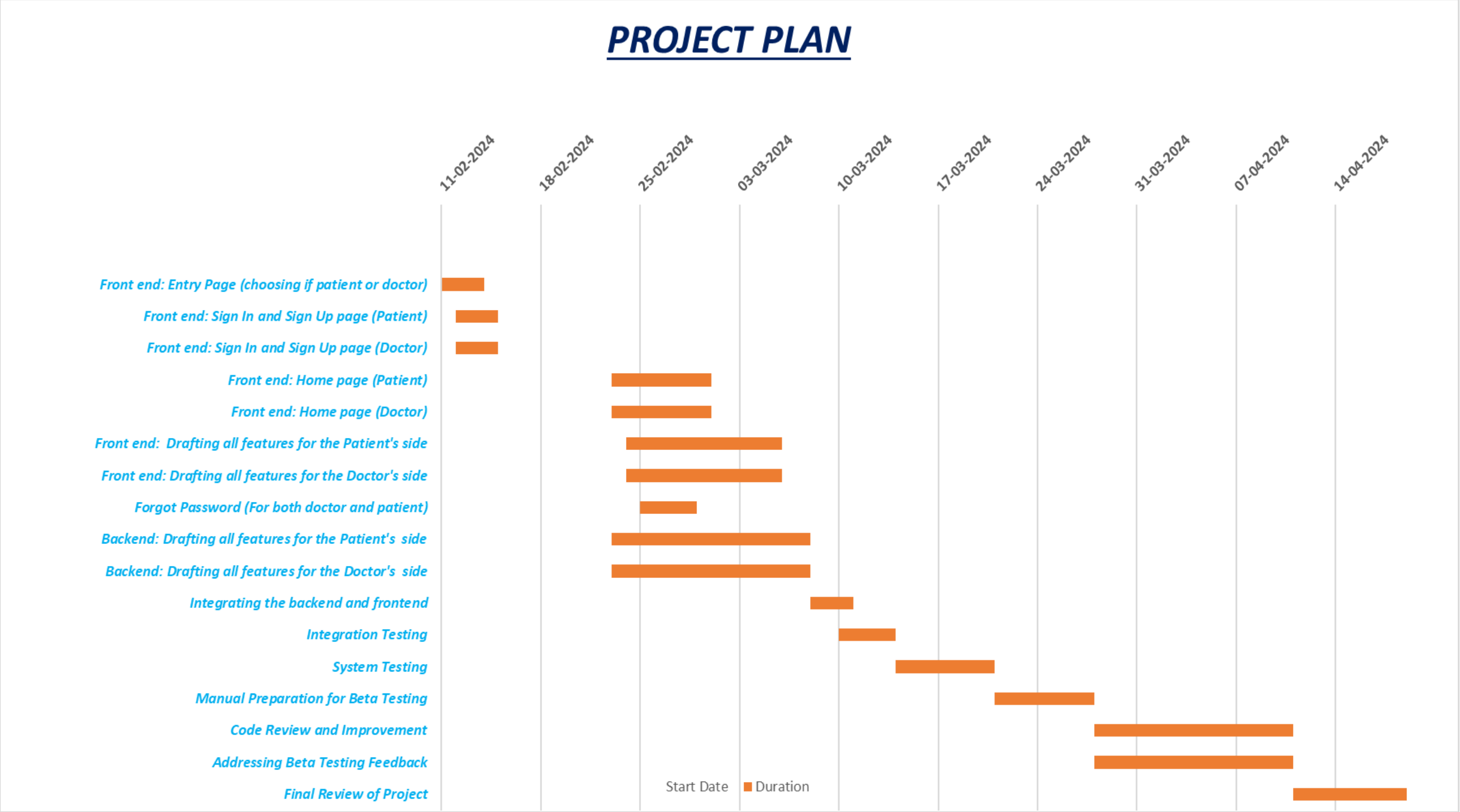

Frontend
Backend
Database
Upload Patient Reports
upload_reports(patient_id, past_report)
Update Reports as Required
Show Updated Reports
Update Succesful
Find a new Doctor
find_doctor(doctor_name, hospital_associated_with, doctor_id)
Search for Doctor In Database
Display Doctor
Doctor Found
Update Patient Details
upload_prescriptions(patient_id, past_prescription)
Update Patient Details as Required
Show Updated Details
Update Succesful
View Previous Documents or Download
view_self_docs (patient_id, is_accessible)
Fetch Documents from Database
Display or Download locally as requested
Successfully fetched
Add ongoing medical status
add_ongoing_medical_status(patient_id, current_medicines, current_ailments)
Update Medical Status of Patient
Update Successful
Give document access to Doctors
docs_access(patient_id, doctor_id, docs_id)
Fetch Required Document
Provide View Access to Doctor


Frontend
Backend
Database
Doctors can add/update details
doctor_details(doctor_id, doctor_specialization
, hospital_associated_with, working_days,
working_hours)
Display Updates details
Update/Add Details as Required
Update Succesful
Update Timing of Doctor
update_timing(doctor_id,
working_days, working_hours)
Updated Timings
Update Succesful
Viewing/Down
loading
Patient
Documents
view_patient_docs(patient_
id, is_accessible) or
download_docs(patient_id)
Display/Download patient
documents
Fetch Required Documents
Successfully Fetched
Add Scans
or Edit
Prescription
add_scans(doctor_id, patient_id,
scan_type)
or
edit_prescription(doctor_id,
patient_id, is_accessible)
Display added/edited
documents
Add scan to
Database or edit
prescription
Successfully
added/edited
Schedule
follow up or
next
appointment
next_appointment(doctor_id,
patient_id)
Update timing for
next checkup
Update Successful
Manual Verification
License Verification
verify_license(doctor_i
d, license_image)
Verification Successful
Verification Unsuccessful

3.4 State Diagrams


New User
Register
Patient
Doctor
Do: Display page to fill details
Create Account
Do: Display page to fill details and upload licence
Create Account
Do: Display Reset Password page
Login
Forgot Password
Existing User
Patient
Doctor/ lab
Homepage
Homepage
Search
Do: Open search bar to take user input
View scans
Do: direct to page where scans are stored
Access Prescriptions
Do: direct to page where prescriptions are stored
Download
Do: Download document to device
Lab reports
Do: Direct to page where lab reports are stored
Download
Send documents
Do: Direct to page with list of documents
select documents
Do: Open search bar to take user input
select doctor
Send
Add medications/ reports
Do: Direct to page with medication details
Upload document
Do: Open device downloads to choose document
Upload
Update timings
Do: Open days and time range list
Search
Do: Open search bar to take user input
Select Patient
View documents
Do: Display documents sent by patient
Download
Add document
Do: Display page to fill document details
Schedule follow up
Upload document
Do: Open calendar and 24hr time list
Do: Open device downloads to choose document
Upload

4 Project Plan



Activity	Start Date	Duration	End date	Team members
Front end: Entry Page (choosing if patient or doctor)	11-02-2024	3	13-02-2024	Bhaumik, Purav
Front end: Sign-in and Sign-Up page (Patient)	12-02-2024	3	14-02-2024	Nilay, Aarsh
Front end: Sign in and Sign-Up page (Doctor)	12-02-2024	3	14-02-2024	Ruthvik, Divit
Front end: Home page (Patient)	23-02-2024	7	29-02-2024	Shaurya, Tanush, Devansh Bansal
Front end: Home page (Doctor)	23-02-2024	7	29-02-2024	Devansh Agarwal, Ankit, Nilay

Front end: Drafting all features for the Patient's side	24-02-2024	11	05-03-2024	Tanush, Aarsh, Bhaumik, Purav, Divit
Front end: Drafting all features for the Doctor's side	24-02-2024	11	05-03-2024	Shaurya, Devansh Bansal, Devansh Agarwal, Ankit, Ruthvik
Forgot Password (For both doctor and patient)	25-02-2024	4	28-02-2024	Devansh Bansal, Devansh Agarwal, Ankit
Backend: Drafting all features for the Patient's side	23-02-2024	14	07-03-2024	Tanush, Shaurya, Nilay, Bhaumik
Backend: Drafting all features for the Doctor's side	23-02-2024	14	07-03-2024	Aarsh, Ruthvik, Purav, Divit
Integrating the backend and frontend	08-03-2024	3	10-03-2024	Entire Team
Integration Testing	10-03-2024	4	13-03-2024	Entire Team
System Testing	14-03-2024	7	20-03-2024	Entire Team
Manual Preparation for Beta Testing	21-03-2024	7	27-03-2024	Entire Team
Code Review and Improvement	28-03-2024	14	10-04-2024	Entire Team
Addressing Beta Testing Feedback	28-03-2024	14	10-04-2024	Entire Team
Final Review of Project	11-04-2024	8	18-04-2024	Entire Team

Appendix A - Group Log

- We have made a WhatsApp group for effective communication amongst ourselves
- We have created a private repository on GitHub on which we would be regularly pushing our codes to maintain efficiency and collaboration.
- We have broadly divided the team into parts, all members would be working on different aspects of frontend and backend so that all members understand the mechanism of software development and are able to implement it.

The following is the group log for the development of this SDD (Software Design Document).

Date	Time	Meet type	Final Outcomes of the Meet
Jan 29, 2024	8.30-9.30	Offline	Forming groups among our team to tackle different aspects of the project
Jan 31,2024	10-10.30	Online	Meet with TA to clear queries pertaining the SDD and review the weekly logs.
Jan 02,2024	10.30-12.00	Offline	Dividing sections of the SDD, and assigning them to members of the team
Jan 06,2024	10.00-11.00	Online	To check on the progress of all team members
Jan 08,2024	10.00-12.30	Offline	Final meet amongst the team members as a robust draft of the SDD documentation