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  Ruthvik Tunuguntla  Aarsh Ashish Walavalkar  Purav Jangir  Ankit Kaushik  Bhaumik Chawda  Tanush Goel  Devansh Agarwal | 220714  220924  220013  220837  220158  220292  221131  220340 | nilayag22@iitk.ac.in  ruthvikjs22@iitk.ac.in  aarshaw22@iitk.ac.in  puravj22@iitk.ac.in  kankit22@iitk.ac.in  bhaumik22@iitk.ac.in  tanushg22@iitk.ac.in  devanshag22@iitk.ac.in |

|  |  |
| --- | --- |
| Course: | CS253 |
| Mentor TA: | *Somesh Vas* |

Content

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

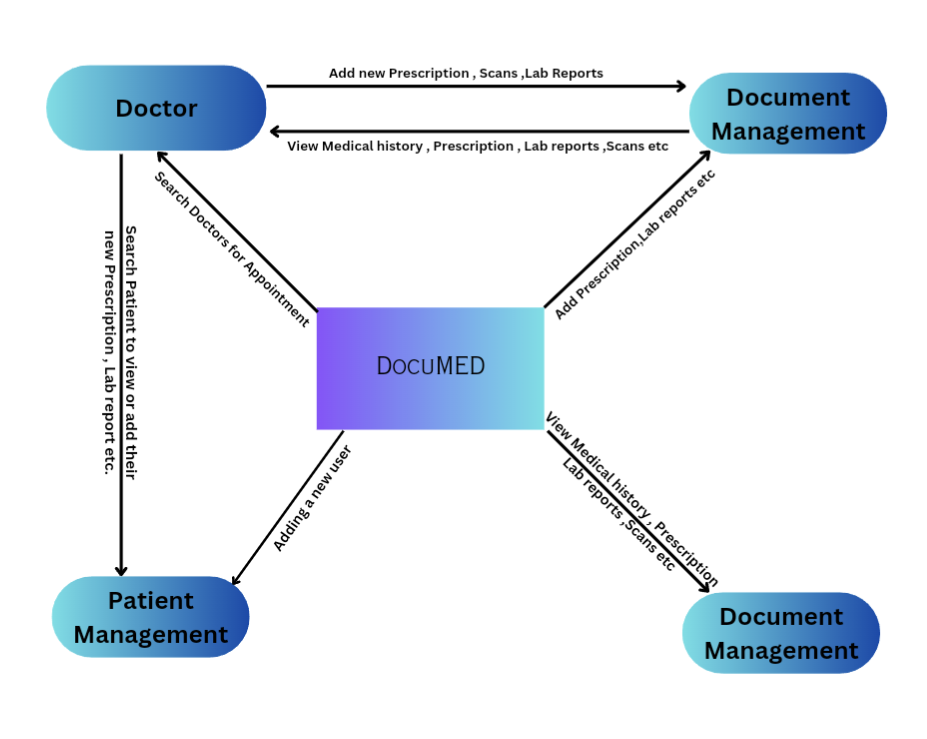
Revisions

| 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 |

# Context Design

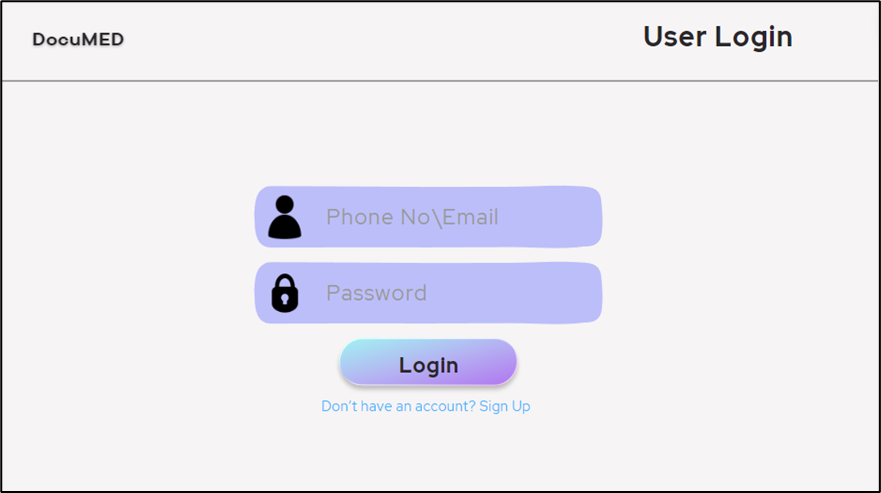
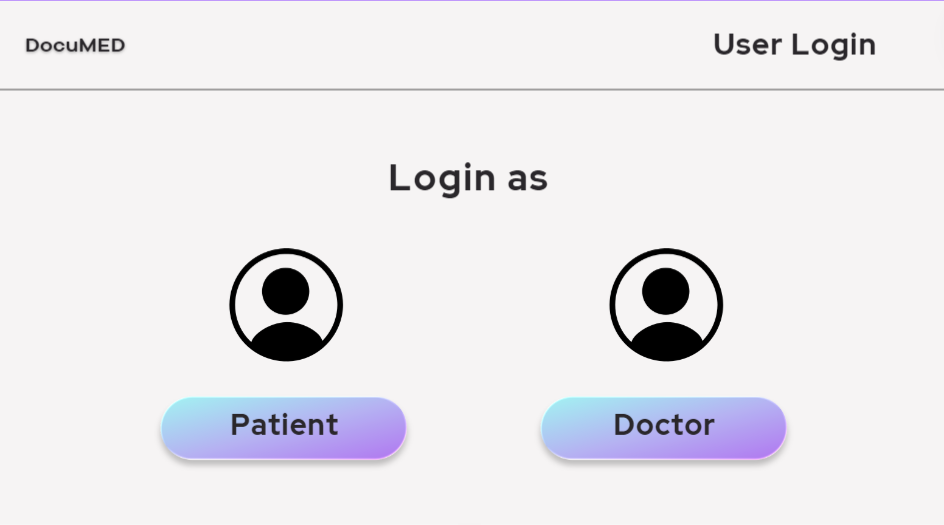
## Context Model

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.

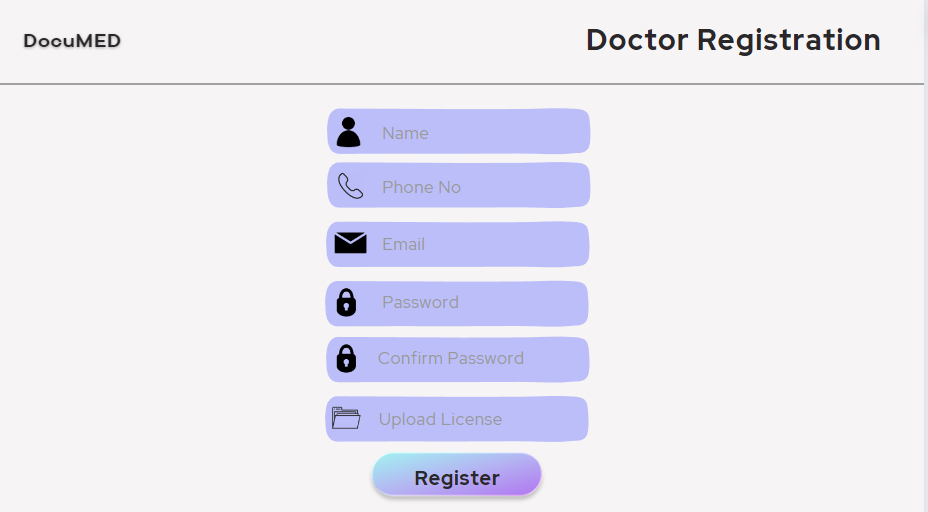


## Human Interface Design

|  |
| --- |
| Landing page |
| **user\_type(is\_doctor, is\_patient)**  *User selects whether they are a doctor or patient* |

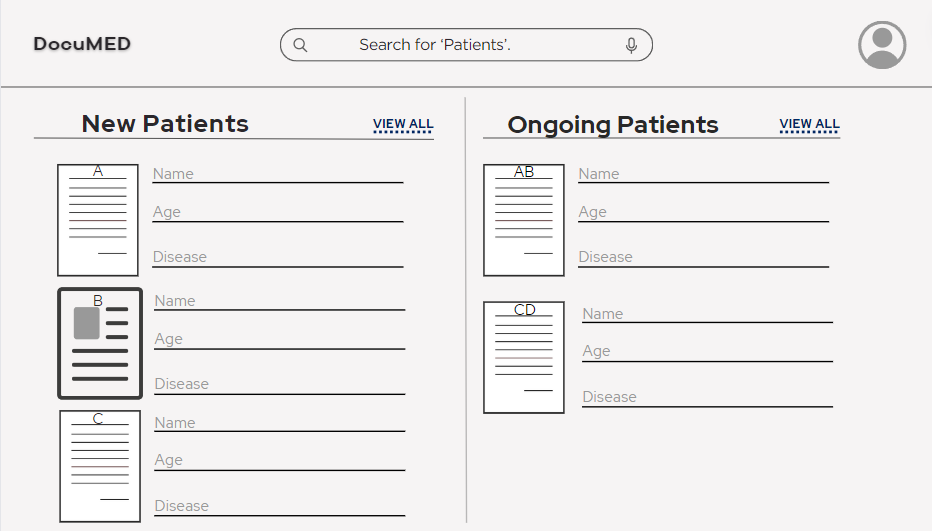


|  |
| --- |
| Sign in page |
| **verify\_credentials(phone no, password)**  *For both doctors and patients to log in* |

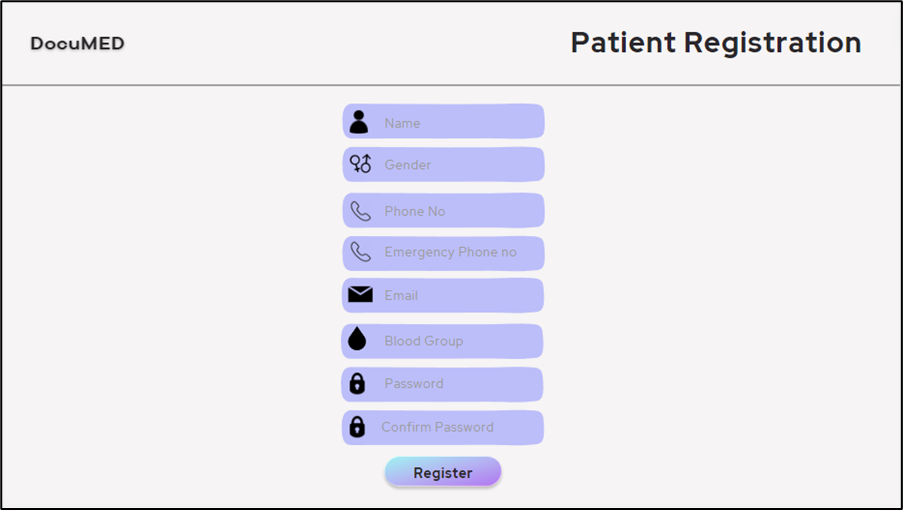
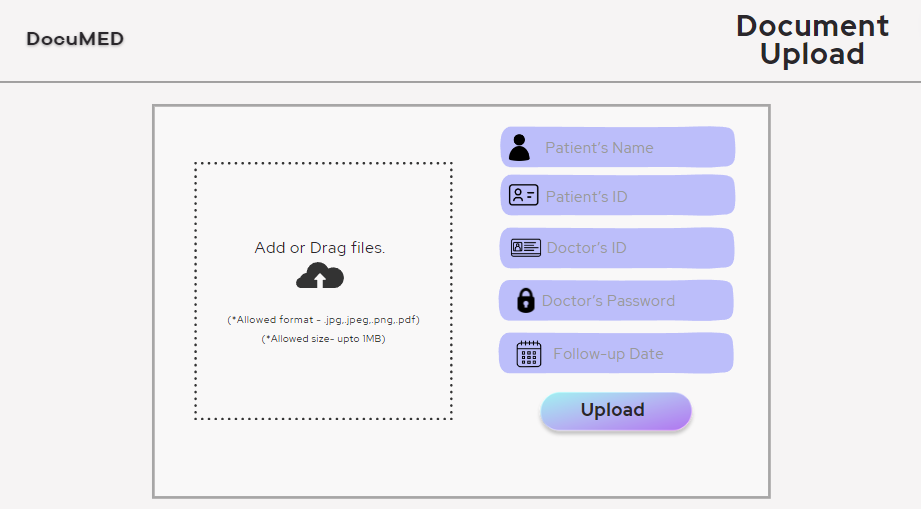


|  |
| --- |
| Doctor register page |
| **doctor\_id = new\_doctor(doctor\_name, password, phone no., email\_id, license\_image)**  *Create new account*  **doctor\_details(doctor\_id, doctor\_specialization , hospital\_associated\_with, working\_days, working\_hours)**  *Doctor adds details about practice*  **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* |

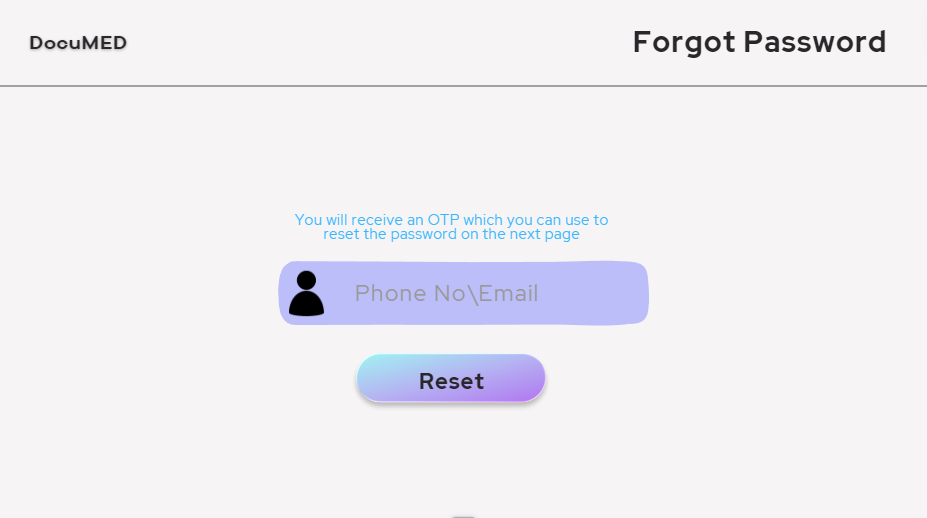
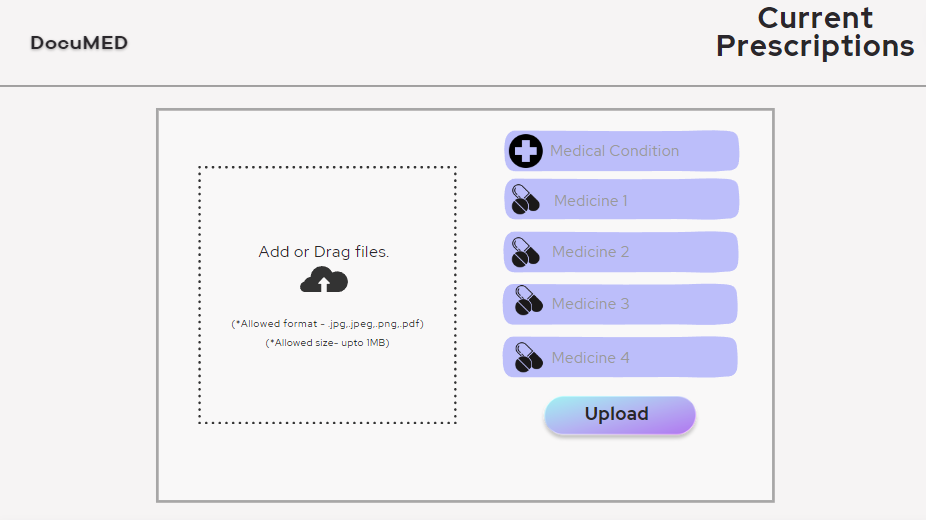
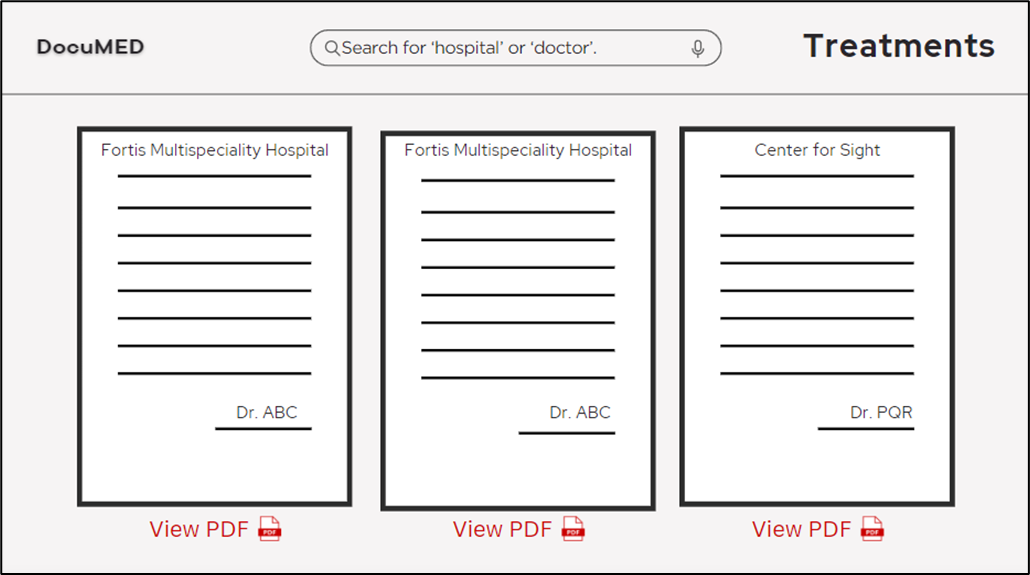
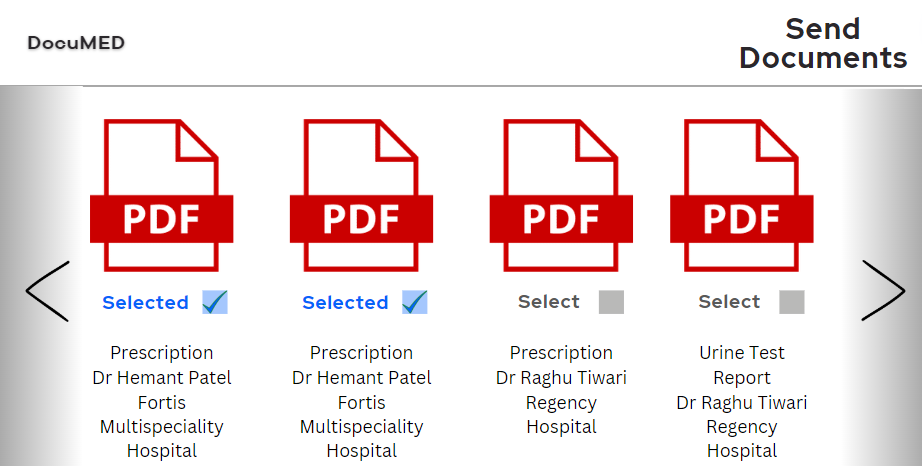
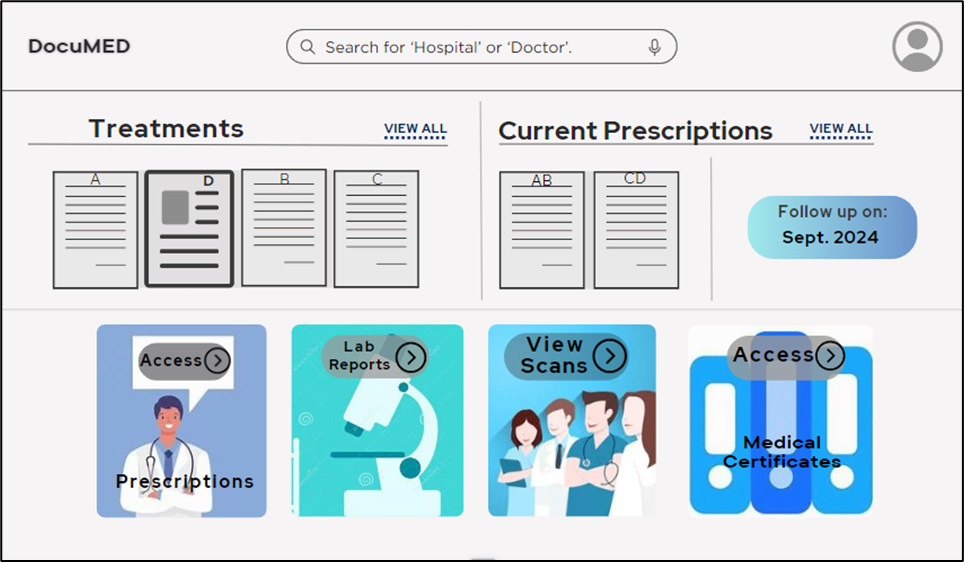


|  |
| --- |
| Patient view (doctor interface) |
| **add\_certificate(doctor\_id, patient\_id, certificate\_type)**  *Doctors can add Medical Certificate and Disability Certificate*  **add\_scans(doctor\_id, patient\_id, scan\_type)**  *Doctors can add X-Rays, MRI and Ultrasound, CT scan and PET-CT*  **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* |

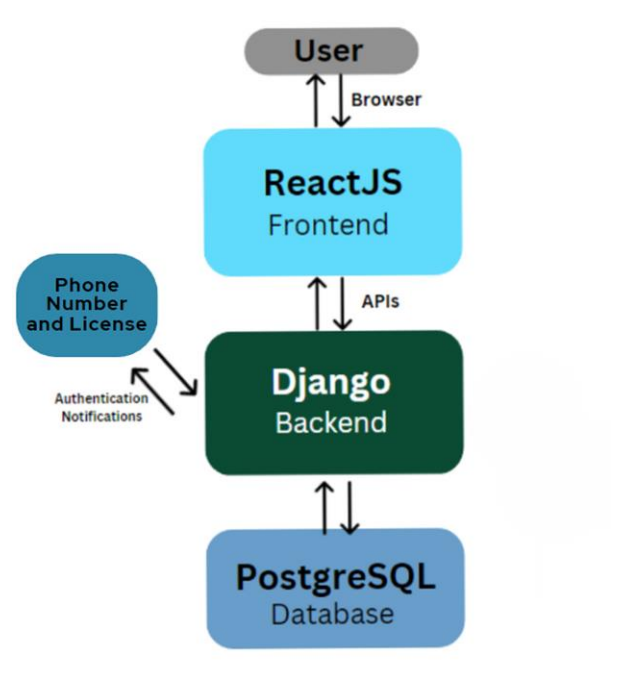


|  |
| --- |
| Patient register page |
| **patient\_id = new\_account(name, gender, blood\_group, phone\_no, emergency\_phone\_no, email, password)**  *Patient could create a new account.*  **upload\_reports(patient\_id, past\_report)**  *Patient can upload their past reports in PDF/Image format.*  **upload\_prescriptions(patient\_id, 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* |



# Architecture Design



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

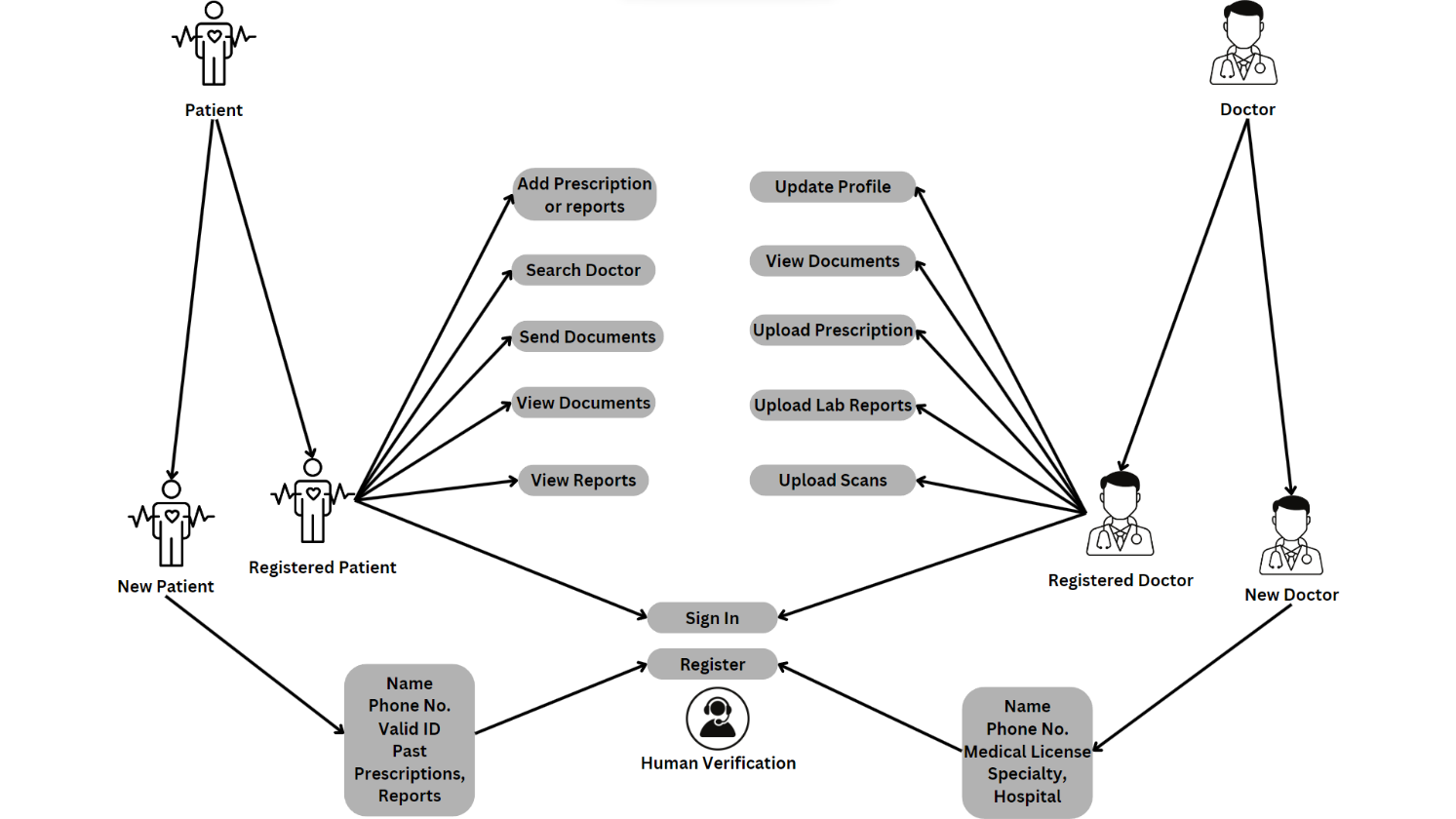
* **Scalability:** The architecture is scalable, meaning it can handle an increased number of users and data. The backend and database can be scaled horizontally by adding more instances, and the frontend can be scaled vertically by increasing the resources of the server. To implement horizontal scaling, 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.

# Object Oriented Design

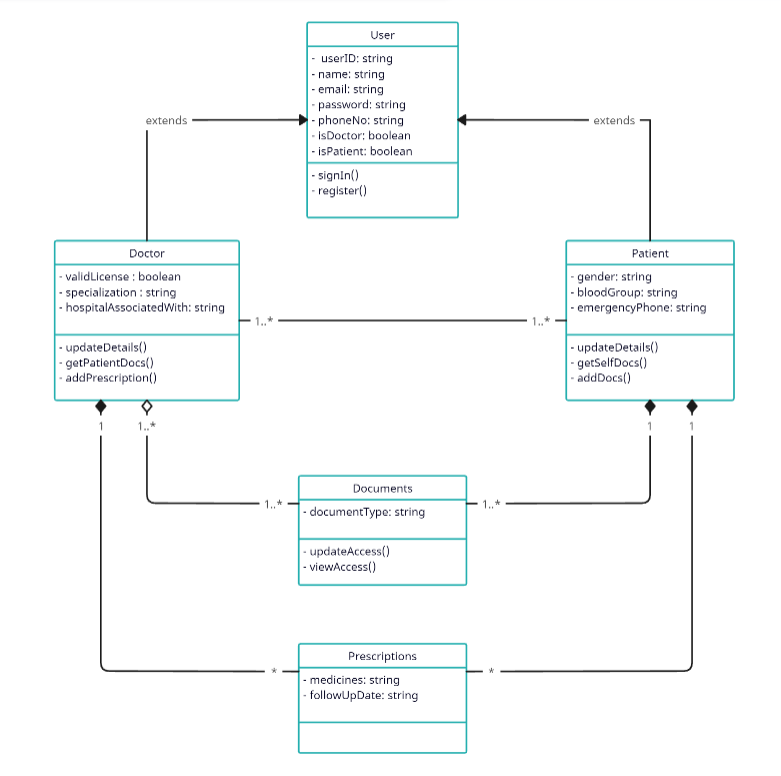
## Use Case Diagrams

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

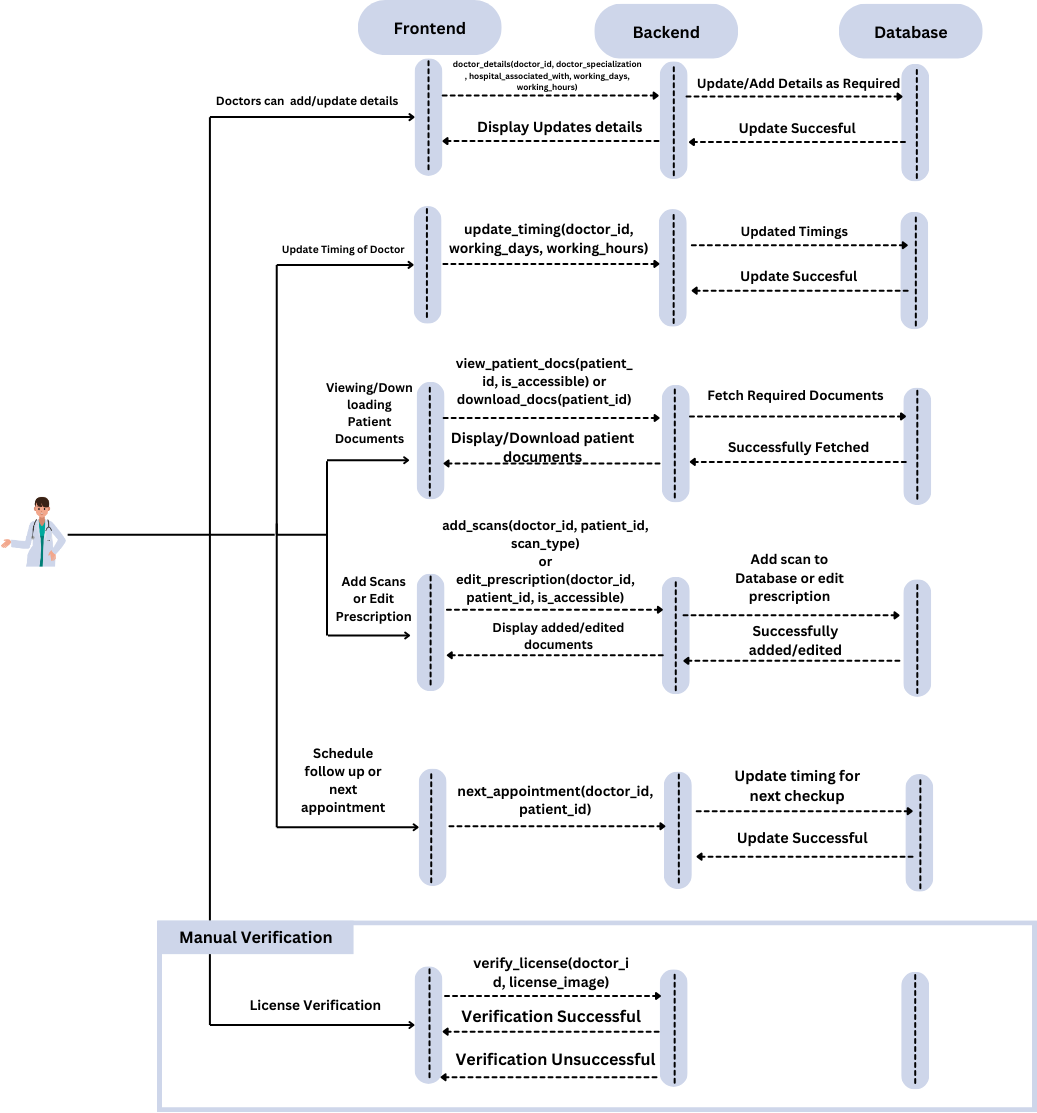
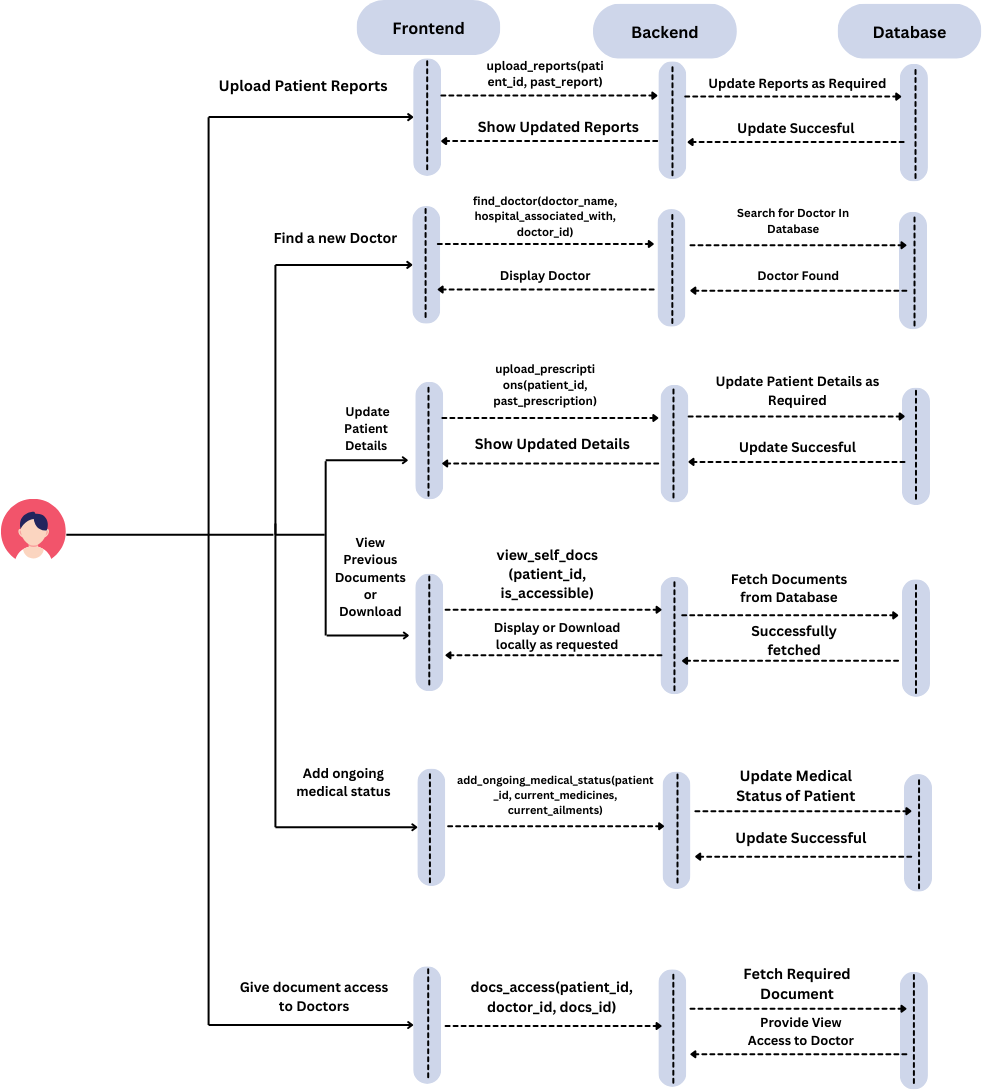
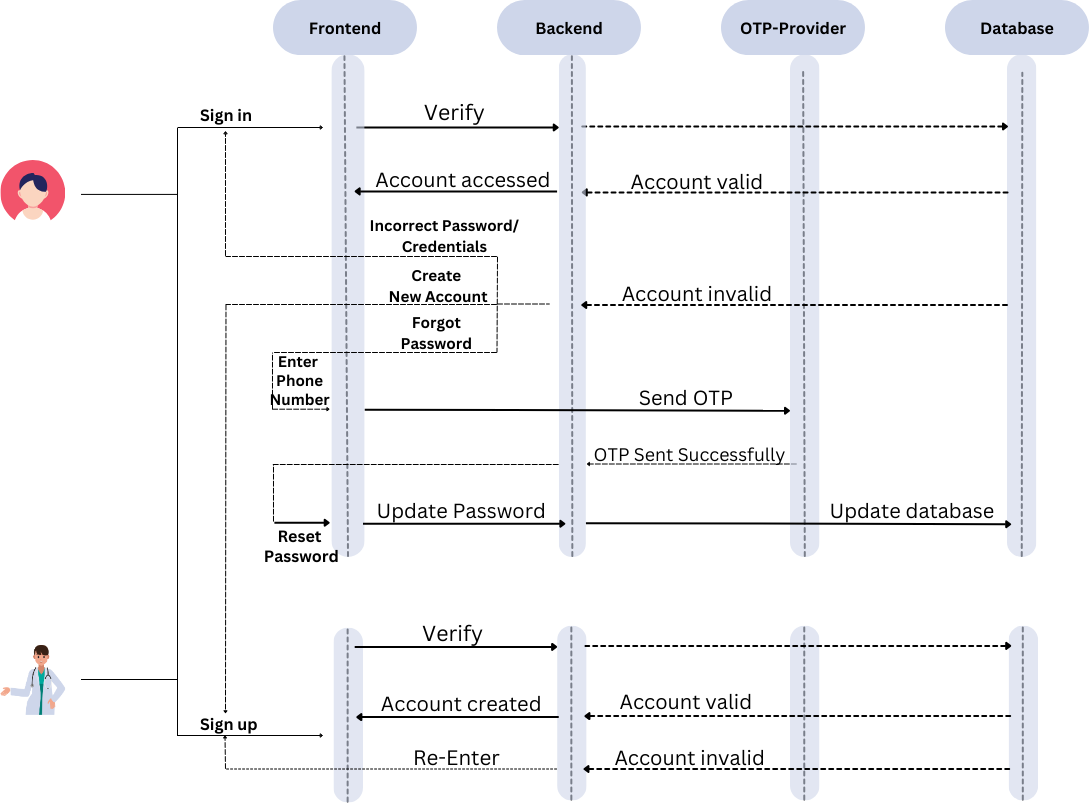
* New Patient and Doctor Registration.
* Existing Patient and Doctor sign-in.
* Patients can add prescriptions or Reports.
* Patients can search for a Doctor.
* Patients can send desired documents to the Doctor.
* Patients can view prior Documents and Reports.
* A Human Manually verifies a doctor's License.
* 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



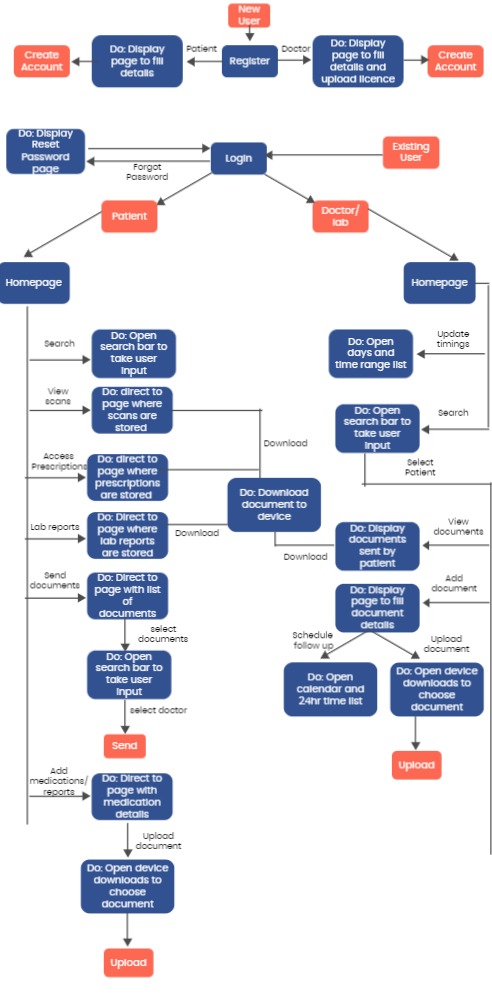
## Class Diagrams



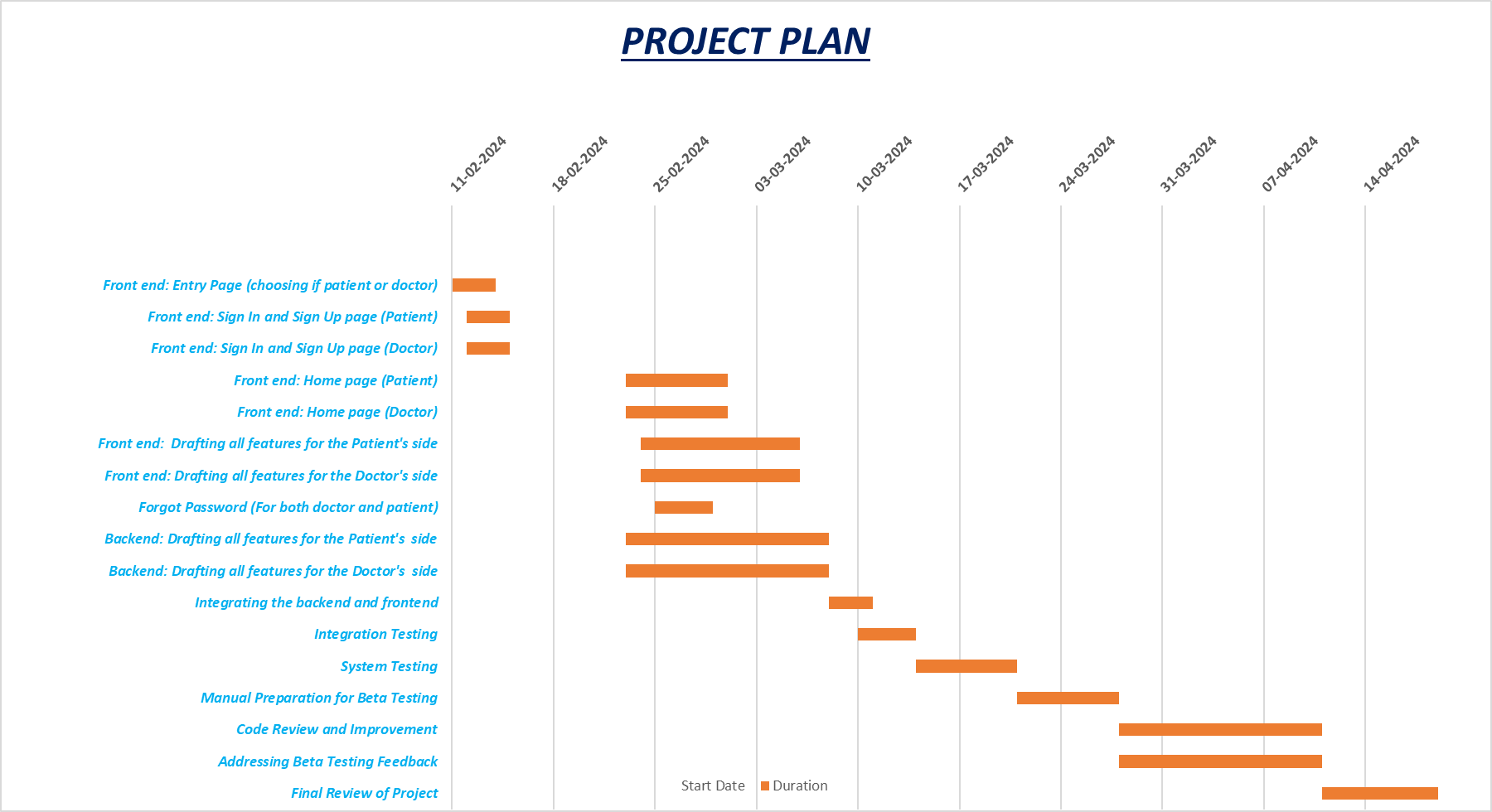
## Sequence Diagrams



## State Diagrams



# 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 |