IT632 Software Engineering

Group 1

Universal Patient Record

Members:

Anil Mangtani (202112005)

Parth Soni (202112007)

Khusboo Chhabra (202112014)

Kalp Vora (202112018)

Ravi Agarwal (202112021)

Nickhil Chhabria (202112026)

Arpitha Sreenivasan (202112030)

Vivek Doshi (202112063)

Gautam Patel (202112086)

Reniz Shah (202112126)

Guided by

Prof. Saurabh Tiwari

Mr. Varun Shah

1. Introduction

In the existing system patients are given physical copies of the prescription. Patients need to maintain files and prescriptions given by different doctors and there are high chances of losing some prescriptions.

In this project, we are creating a website that will allow patients to store their medical records at a centralized location and will be accessible to them whenever they want. Along with patients, doctors can view the medical records during the consultation which will help them in their diagnosis. The chances of losing a record are very low so doctors can view a patient's entire medical history.

2. Overall Description

2.1. Project Scope Description

The primary objective of our project is to develop a system that permits patients to store their medical records at centralized locations and they can access the records from any location. This enables the doctors to access the medical record of any patient from digital medium which reduces paperwork and chances of losing important medical records are also minimized.

To develop our system we have used the following technologies -

A. For frontend - React Js
B. For backend - Django
C. For database - SQLite

Regarding risks involved in our project, we are dealing with sensitive information like Aadhaar Card number, Doctor license number, patient medical records, etc. All this information is being stored in the database itself.

2.2. Users and Stakeholders

End Users:
Any person with a valid Aadhaar id can register as a patient in the system.
Any doctor with a valid license number and Aadhaar id can try to register in the system.
Admin is exclusive for the stakeholder representatives.

Stakeholders:
Government health bodies are the stakeholders for the system as the patient health details should be handled and viewed by the government bodies only.

2.3. Possible Features

- ➔ The medical records of the patients are stored/fetched to/from the database.
- ➔ Patients can view/add their medical records.
- ➔ Doctors can add prescriptions for various registered patients.
- ➔ Doctor and admin can view the medical record of any patient.
- ➔ Admin can block any account of a doctor/patient.

2.4. Requirement elicitation technique

A. Brainstorming

For our system we have used Brainstorming, a group problem-solving technique that involves the spontaneous contribution of ideas from all members of the group.

In Brainstorming we explain the main agenda of our project then group discussion is held and each individual comes up with a different idea.

The main advantage of using brainstorming is that multiple ideas come very quickly. Promoting equal participation means each member can share their ideas and thoughts which helps in finding the best solution quickly.

In our system we tried to keep things as user friendly as possible with all the required functionalities that would improve the efficiency of our website.

B. Use Cases

In any system there are multiple interactions of the user and the application. These interactions can be described as Use cases which would cover the complete sequence i.e starting from the beginning to end.

According to our system's requirements we have designed various use cases. For each use case, multiple instances are created, also called the scenarios.

2.5. Process Model

The requirements of the systems are very straightforward and well-understood. Since they are well-understood there wouldn't be any change in the requirements at a later stage. Thus we chose the Waterfall model as our process model.

3. Functional Requirements

3.1. Functional Requirements

Registration : Patients need to register using Aadhaar since it's a universal record so multiple registrations of the same Patient are not possible. Doctors need to register and provide a doctor's license and other details which will be used for background checks. Only after getting verified by the admins, doctor's account will be created

Login :

1. The registered Patient/Doctor/Admin needs to log in to access the functionalities of the system.
2. Patient and Doctor can edit his/her profile.
3. Patient and Doctor can view his/her profile.
4. Doctor and Admin can view medical records of any Patient

Patient:

1. After registration is completed Patients will fill in the basic details and initial medical details.
2. Patients can view his/her medical records with timelines.
3. Patients can add previous prescriptions/consultations as well.
4. Patients can add any prescription/consultation he/she got from a doctor not registered to our application.

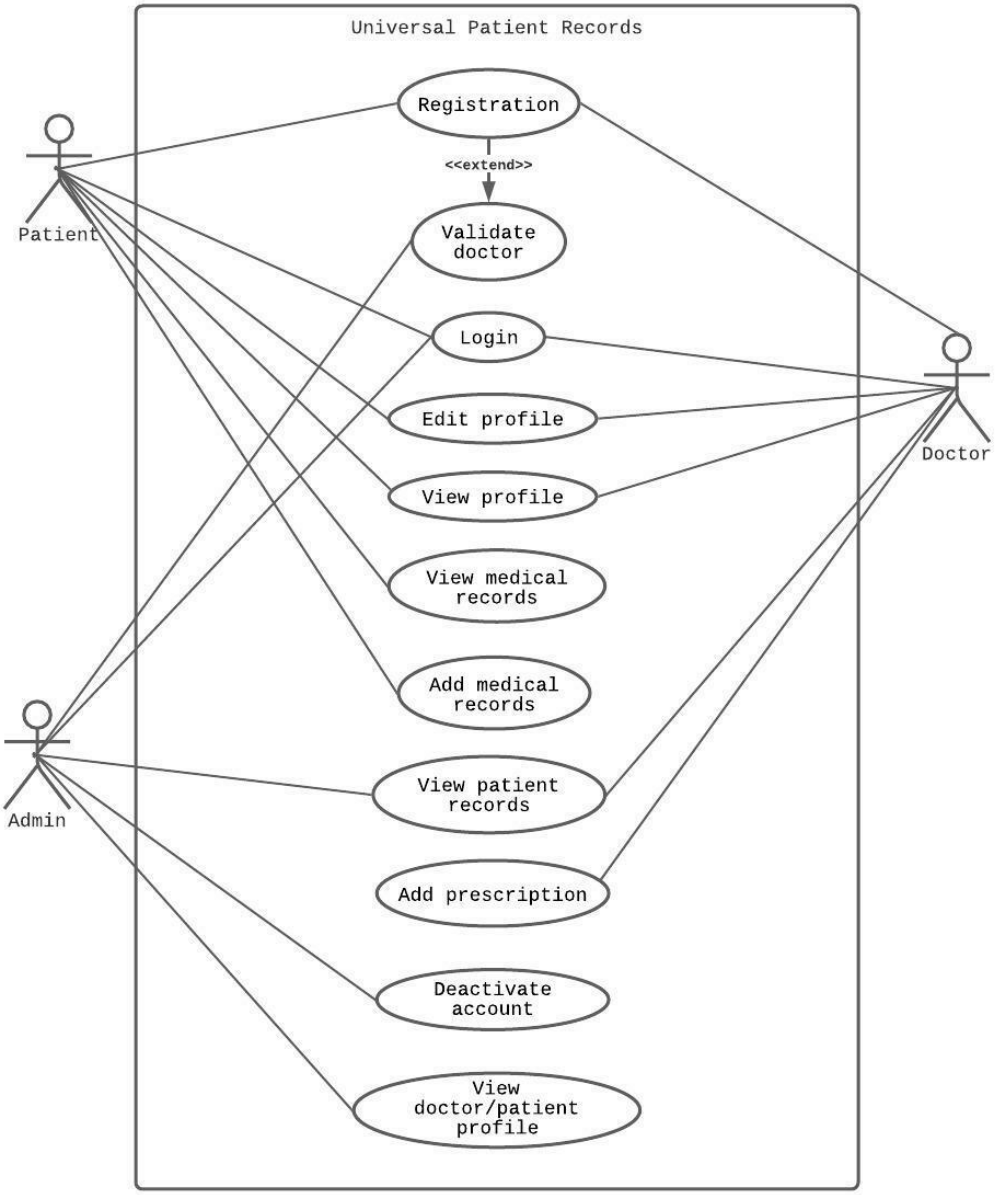
Doctor:

1. Doctors can add a new prescription/consultation.

Admin:

1. Admin will verify the doctor that has registered to the system and only after his/her approval doctor's account will be created else doctor's account will not be created.
2. Admin can deactivate the patient/doctor accounts.
3. Admin can view the profile of doctors and patients

3.2. UseCase Diagram



3.3. UseCase Description

1. Registration (U1)

    1.1 Introduction: User will be able to register themselves with the application

    1.2 Actors:

        1.2.1 Patient

        1.2.2 Doctor

    1.3 Post-conditions:

        1.3.1 Account created for Patient

1.3.2 Account created for Doctor

1.4 Basic flow:

1.4.1 User navigates to **sign up**

1.4.2 User enters the details

1.4.3 System checks if user exists

1.4.4 Account created

1.5 Alternate flow:

1.5.1 For patient

1.5.1.1 User will have to enter patient centric details

1.5.1.2 Account will be created for patient

1.5.2 For doctor

1.5.2.1 Admin manually verifies the doctor

1.5.2.2 Account will be created for doctor

1.6 Exceptional flow:

1.6.1 If user already has an account sign up will be unsuccessful

1.6.2 If verification for doctor is unsuccessful then account for doctor will not be created

1.6.3 If the user clicks "cancel" in the middle of the process the account will not be created.

1.7 Special requirements:

1.7.1 Password should have at least 8 characters, at least 1 non alphanumeric character. It can contain a-z, A-Z, 0-9, @, #.

1.7.2 Patient and Doctor must have a valid Aadhaar id.

1.7.3 Doctors should have a valid medical practitioner's license.

1.8 Use case relationships:

1.8.1 extends "validate doctor" (U2)

2. Validate doctor (U2)

2.1 Introduction: Doctor will be verified

2.2 Actors:

2.2.1 Admin

2.3 Pre-conditions:

2.3.1 Doctor should have entered all the necessary details.

2.4 Post-conditions:

2.4.1 The doctor will be eligible for an account

2.5 Basic flow:

2.5.1 starts when doctor registers into the application

2.5.2 the admin verifies the account using doctor's license number

2.5.3 doctor is verified

2.6 Exceptional flow:

2.6.1 if doctor hasn't entered all the necessary details the verification process fails

2.6.2 if so problem has been found during manual background check by admin verification fails

2.7 Use case relationships:

2.7.1 extended from registration(U1)

3. Login (U3)

3.1 Introduction: Users will be able to log into their respective accounts with username and password.

3.2 Actors:

3.2.1 Patient

3.2.2 Doctor

3.2.3 Admin

3.3 Pre-conditions:

3.3.1 User should have a verified account

3.4 Post-conditions:

3.4.1 User will be able to use services provided by the system

3.5 Basic flow:

3.5.1 Starts when the user enters username and password and clicks the "Login" button.

3.5.2 The user credentials are verified by the system

3.5.3 The user is logged in

3.6 Alternate flow:

3.6.1 Reset password

3.6.1.1 User clicks on reset password

3.6.1.2 The user will receive a code via email

3.6.1.3 User enters the code

3.6.1.4 User submits the code

3.6.1.5 User will be asked for new password

3.6.1.6 User enters new password

3.6.1.7 User submits the password

3.6.1.8 User logs in

3.7 Exceptional flow:

3.7.1 If username or password is incorrect the login will be unsuccessful

3.7.2 If password reset is unsuccessful then login will be unsuccessful

3.7.3 If no account for the user is found then login will be unsuccessful

4. Edit profile (U4)
    - 4.1 Introduction: User can edit profile in the system
    - 4.2 Actors:
        - 4.2.1 Patient
        - 4.2.2 Doctor
    - 4.3 Pre-conditions:
        - 4.3.1 User should be logged in
    - 4.4 Post-conditions:
        - 4.4.1 User profile will be updated
    - 4.5 Basic flow:
        - 4.5.1 User navigates to **Edit Profile page**
        - 4.5.2 User selects the field to be updated
        - 4.5.3 Changes the field
        - 4.5.4 System updates the field
    - 4.6 Exceptional flow:
        - 4.6.1 If a user tries to change "Aadhaar number", the profile will not be edited.
        - 4.6.2 If user cancels the process the profile will not be updated

5. View profile (U5)
    - 5.1 Introduction: User will be able to view their profile
    - 5.2 Actors:
        - 5.2.1 Patient
        - 5.2.2 Doctor
        - 5.2.3 Admin
    - 5.3 Pre-conditions:
        - 5.3.1 User should be logged in
    - 5.4 Post-conditions:
        - 5.4.1 User will be able to view their profile details
    - 5.5 Basic flow:
        - 5.5.1 starts when user navigates to "profile"
        - 5.5.2 User will be able to view their profile details
6. View medical records (U6)
    - 6.1 Introduction: Patient will be able to view their medical records
    - 6.2 Actors:
        - 6.2.1 Patient
    - 6.3 Pre-conditions:
        - 6.3.1 User should be logged in
    - 6.4 Post-conditions:
        - 6.4.1 User will be able to view their medical records

6.5 Basic flow:

6.5.1 Starts when user navigates to "View Medical Records"

6.5.2 User will be able to view their medical records

7. Add medical records (U7)

7.1 Introduction: Patient will be able to add their medical records

7.2 Actors:

7.2.1 Patient

7.3 Pre-conditions:

7.3.1 user should be logged in

7.4 Post-conditions:

7.4.1 User's medical records will be updated with the new data

7.5 Basic flow:

7.5.1 Starts when user navigates to "Add Medical Records"

7.5.2 User can type or attach the documents for medical records

7.5.3 User confirms the changes

7.5.4 Medical records are updated

7.6 Exceptional flow:

7.6.1 If user discards the changes instead of confirming them, the medical records will not be updated

7.6.2 If user submits the changes without making any, the medical records will not be updated

8. View patient records (U8)

8.1 Introduction: Describes how doctor/admin can view record of patient

8.2 Actors:

8.2.1 Doctor

8.2.2 Admin

8.3 Pre-conditions:

8.3.1 doctor/admin should be logged into their accounts

8.4 Post-conditions:

8.4.1 doctor/admin will be able to view patient records

8.5 Basic flow:

8.5.1 Doctor/Admin clicks on view patient record

8.5.2 Doctor/Admin enter's the patient ID

8.5.3 Doctor/Admin can view the patient records

8.6 Exceptional flow:

8.6.1 If the doctor/admin enters a wrong patient ID the patient records will not be retrieved.

9. Add prescription (U9)
    9.1 Introduction: doctor will be able to add prescription for a patient
    9.2 Actors:
        9.2.1 Doctor
    9.3 Pre-conditions:
        9.3.1 Doctor should be logged into their account
    9.4 Post-conditions:
        9.4.1 Patient records will be updated with the new prescription entered by the doctor
    9.5 Basic flow:
        9.5.1 Doctor clicks on Add Prescription button
        9.5.2 Doctor enters the details in the form
        9.5.3 Doctor confirms the changes
        9.5.4 The Patient's medical records will be updated with the new prescription entered by the doctor
    9.6 Exceptional flow:
        9.6.1 If the doctor cancels the changes then the prescription will not be added into the patient's medical records.

10. Deactivate account (U10)
    10.1 Introduction: Admin will be able to Deactivate the accounts of a particular doctor/patient.
    10.2 Actors:
        10.2.1 Admin
    10.3 Pre-conditions:
        10.3.1 Admin should be logged into their account
    10.4 Post-conditions:
        10.4.1 Patient/doctor's account will be Deactivate
        10.4.2 Patient/doctor will not be able to log into their accounts
    10.5 Basic flow:
        10.5.1 Admin navigates to deactivate option in the patient/doctor's profile
        10.5.2 Admin confirms deactivation
        10.5.3 patient/doctor account will be deactivated
    10.6 Exceptional flow:
        10.6.1 If the patient/doctor account don't exist the account will not be deactivated
        10.6.2 If Admin doesn't confirm the deactivation the account will not be deactivated

11. View patient/doctor profile (U11)
    - 11.1 Introduction: Admin will be able to view the patient/doctor's profile
    - 11.2 Actors:
        - 11.2.1 Admin
    - 11.3 Pre-conditions:
        - 11.3.1 Admin should be logged into their account
    - 11.4 Post-conditions:
        - 11.4.1 Admin will be able to view the patient/doctor profile details
    - 11.5 Basic flow:
        - 11.5.1 Admin navigates to view patient/doctor profile
        - 11.5.2 Admin searches for patient/doctor by entering their userID
        - 11.5.3 Admin will be able to view their profile details
    - 11.6 Exceptional flow:
        - 11.6.1 If the user doesn't exist then the admin will not be able to view their details.

4. Non Functional Requirements

1. Reliability: While inserting any data into the field the system must ensure that data is not altered before storing it in the database.

2. Design Constraints: The system should be developed using HTML 5, React, in the front-end side, and on the server-side Django will be used, and for the Database, mySQL is used. The system will be developed as a web application that can be used with all the latest browsers.

3. Availability: The system can be used or accessed at any time of the day. It'll be operational for the whole day.

4. Efficiency: The system must be efficient enough to handle multiple requests from doctors and patients from multiple locations.

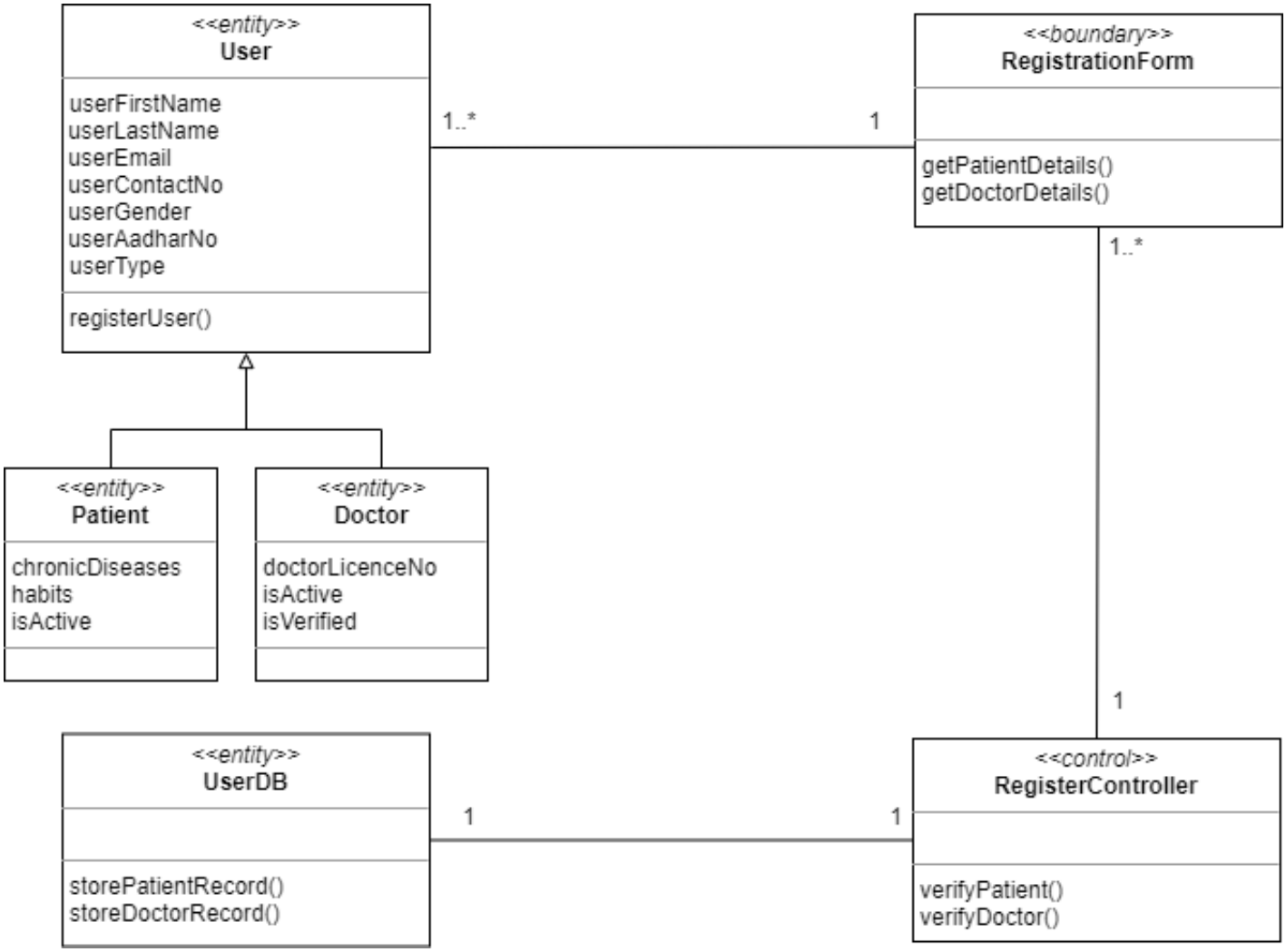
5. Usability: The User-Interface must be simple and clean so that it becomes easier for every patient and doctor to use.

6. Capacity – The database must be capable enough to hold all the data like the patient's personal information and medical records along with doctor details.
7. Regulatory – The system will work within the boundaries of the law.

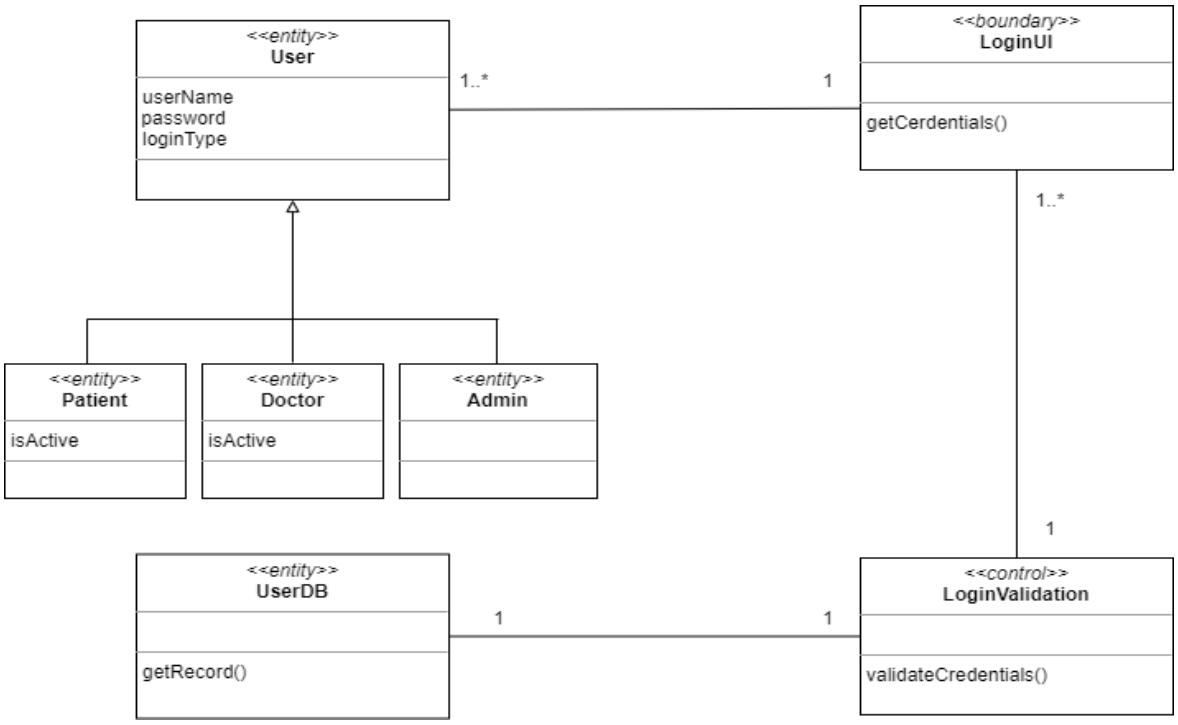
5. Analysis Design Documents

5.1. Analysis Class Diagrams

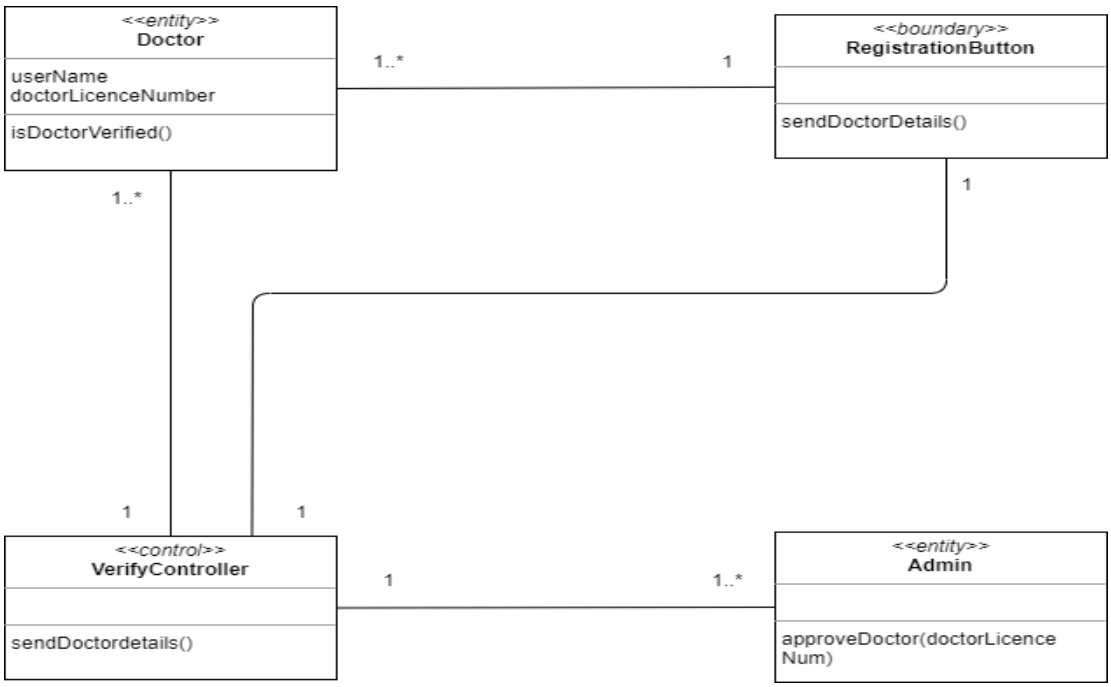
5.1.1. Registration



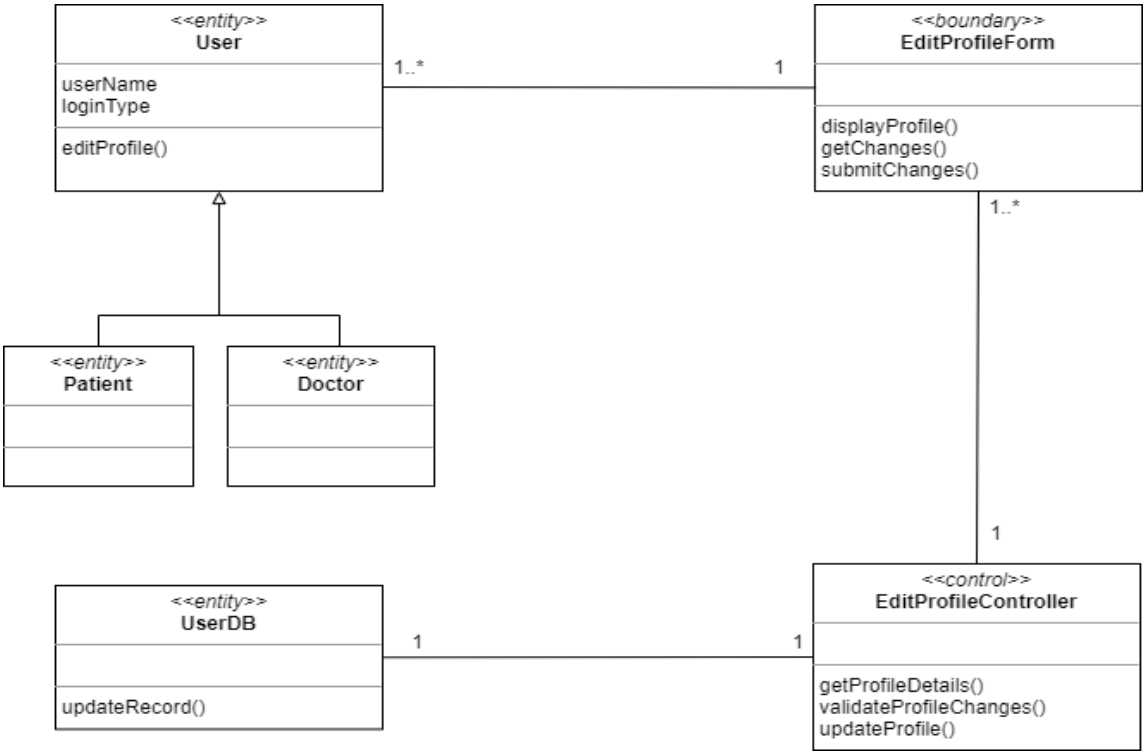
5.1.2. Login



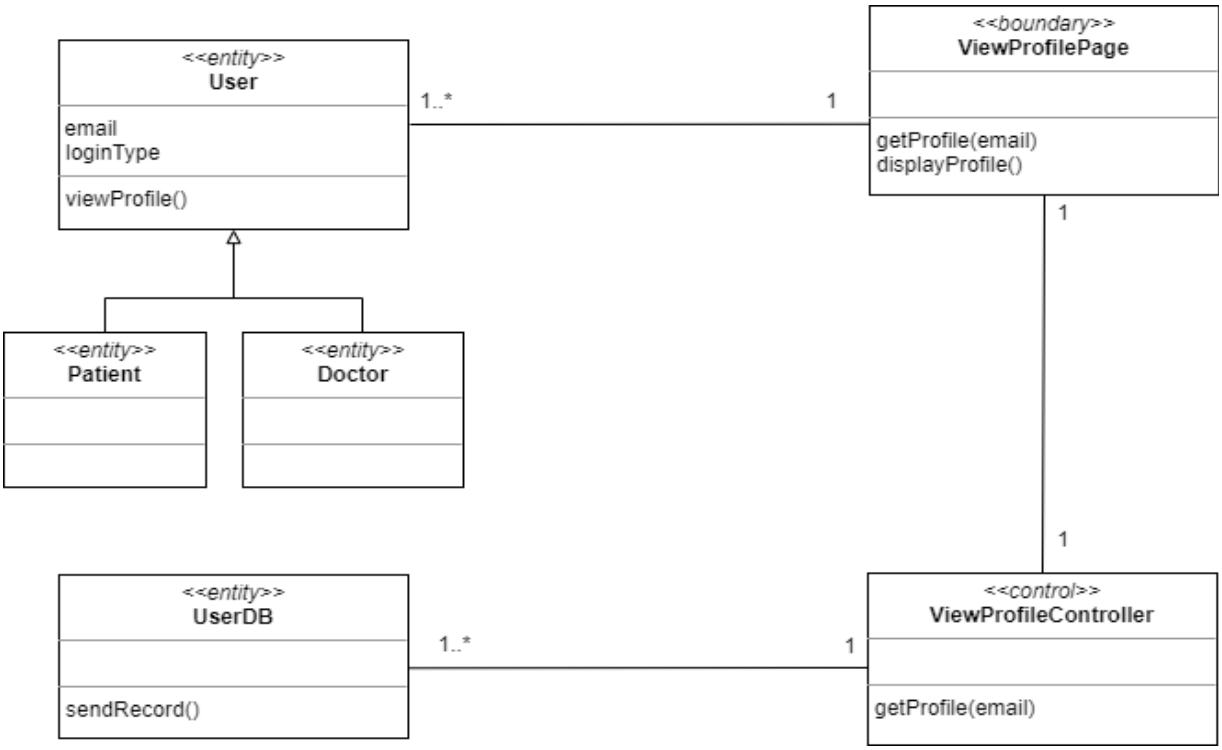
5.1.3. Validate Doctor



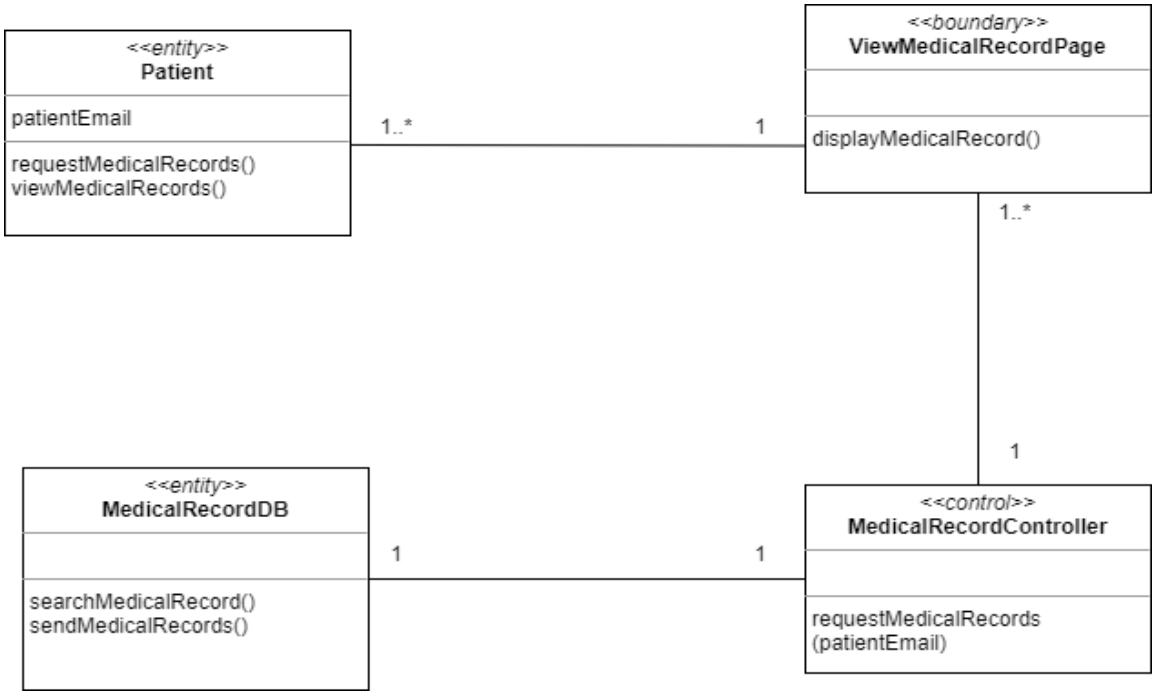
5.1.4. Edit Profile



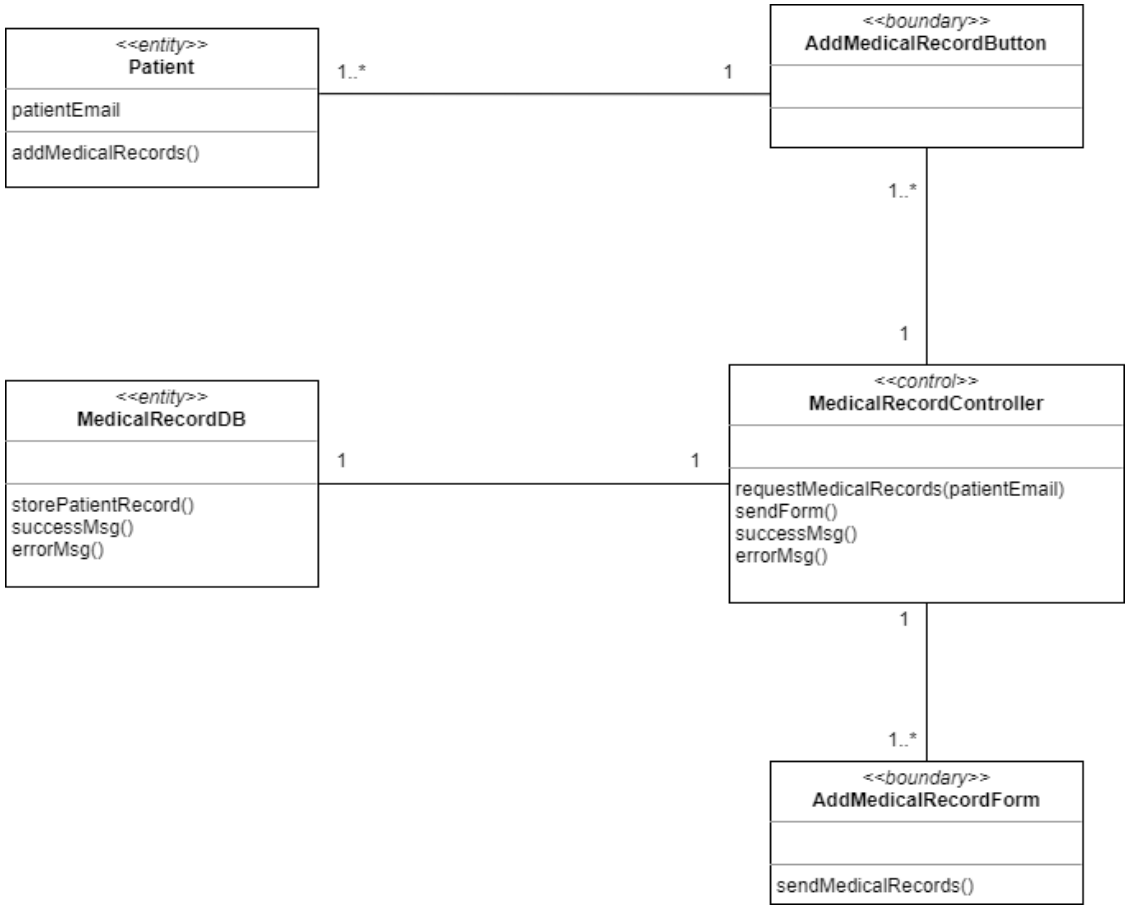
5.1.5. View Profile



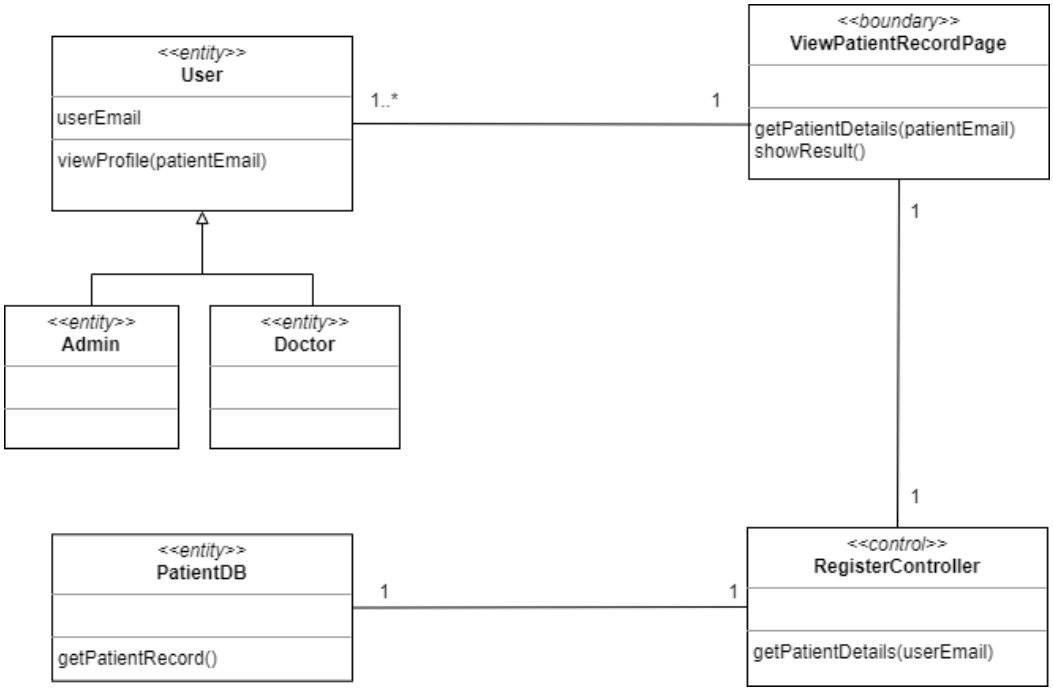
5.1.6 View Medical Record



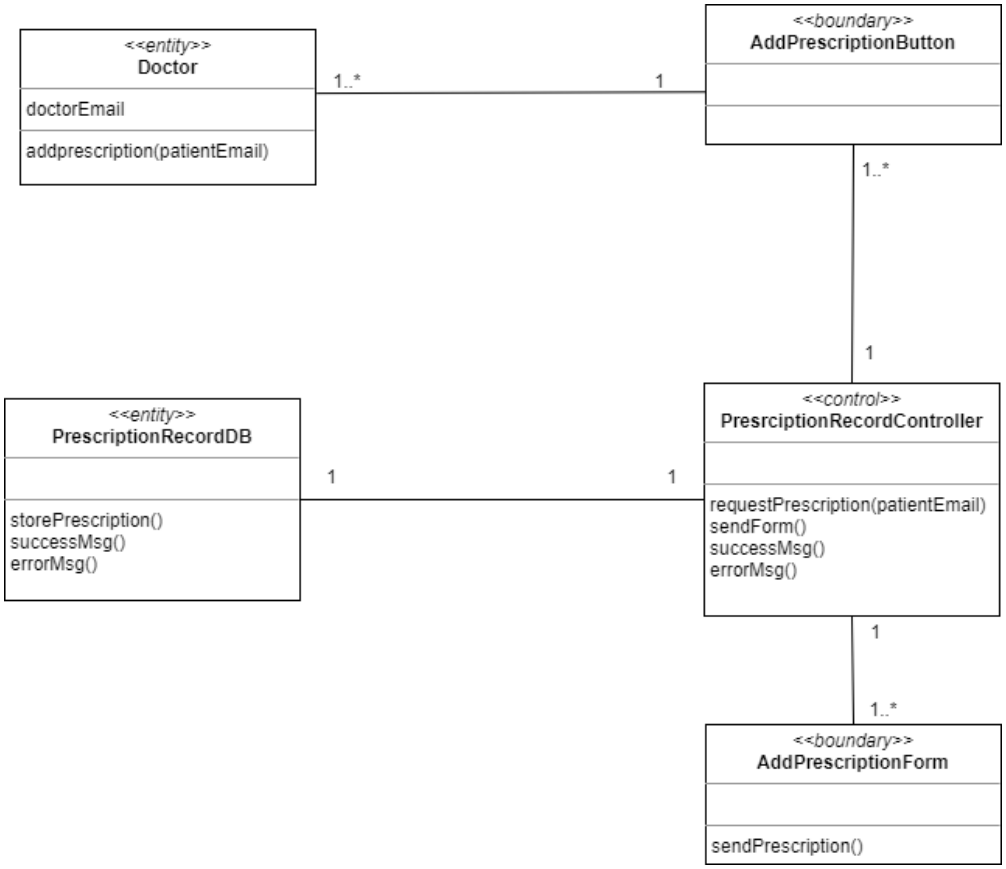
5.1.7. Add Medical Record



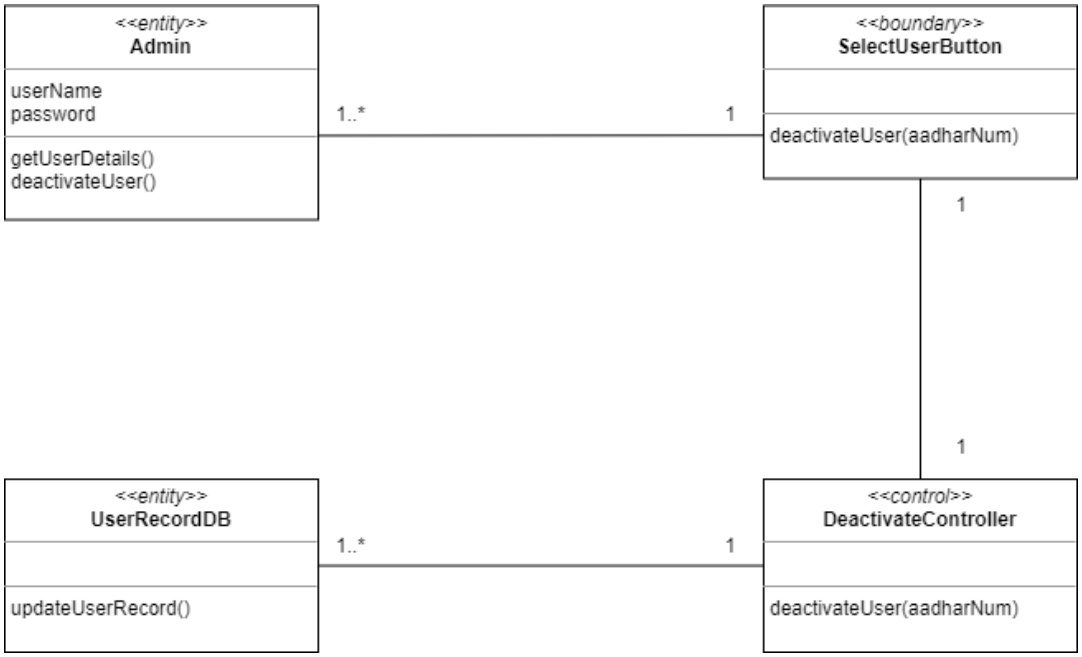
5.1.8. View Patient Record



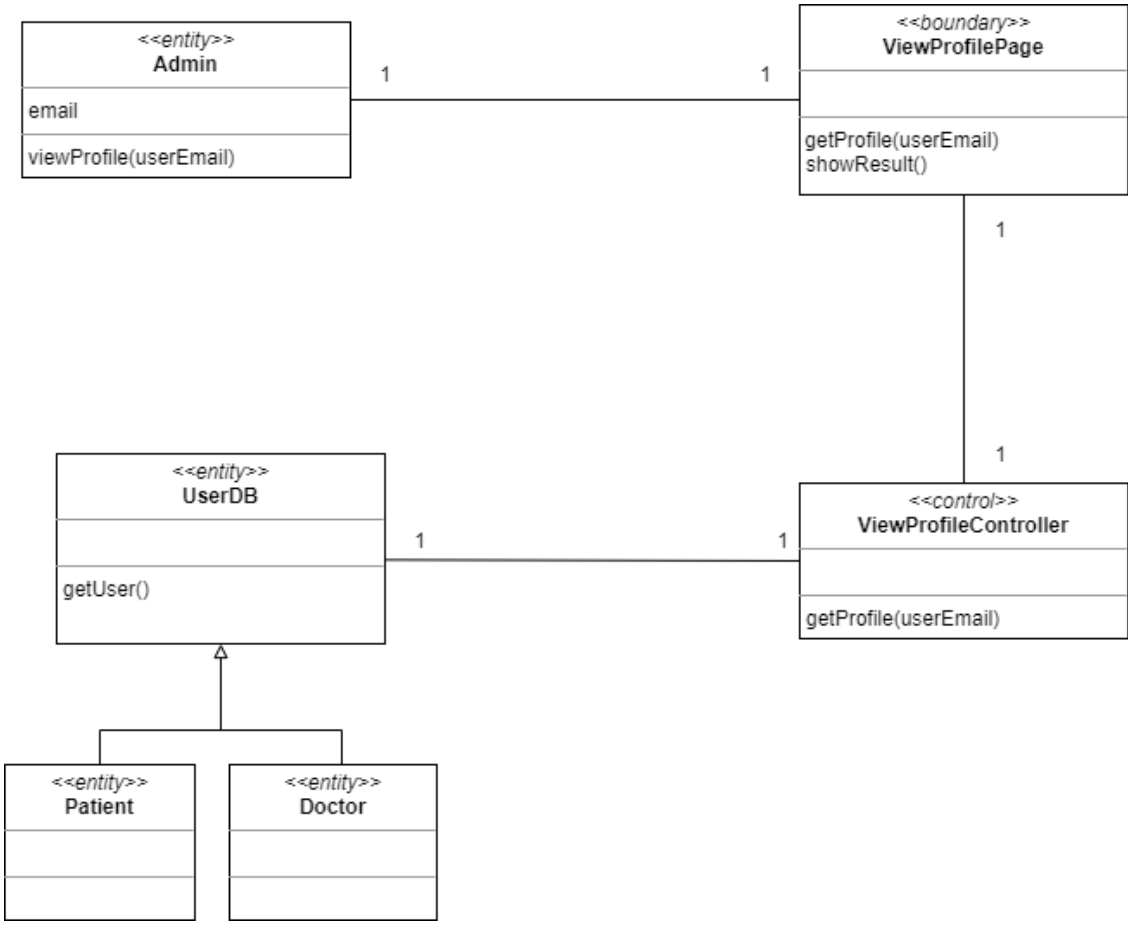
5.1.9. Add Prescription



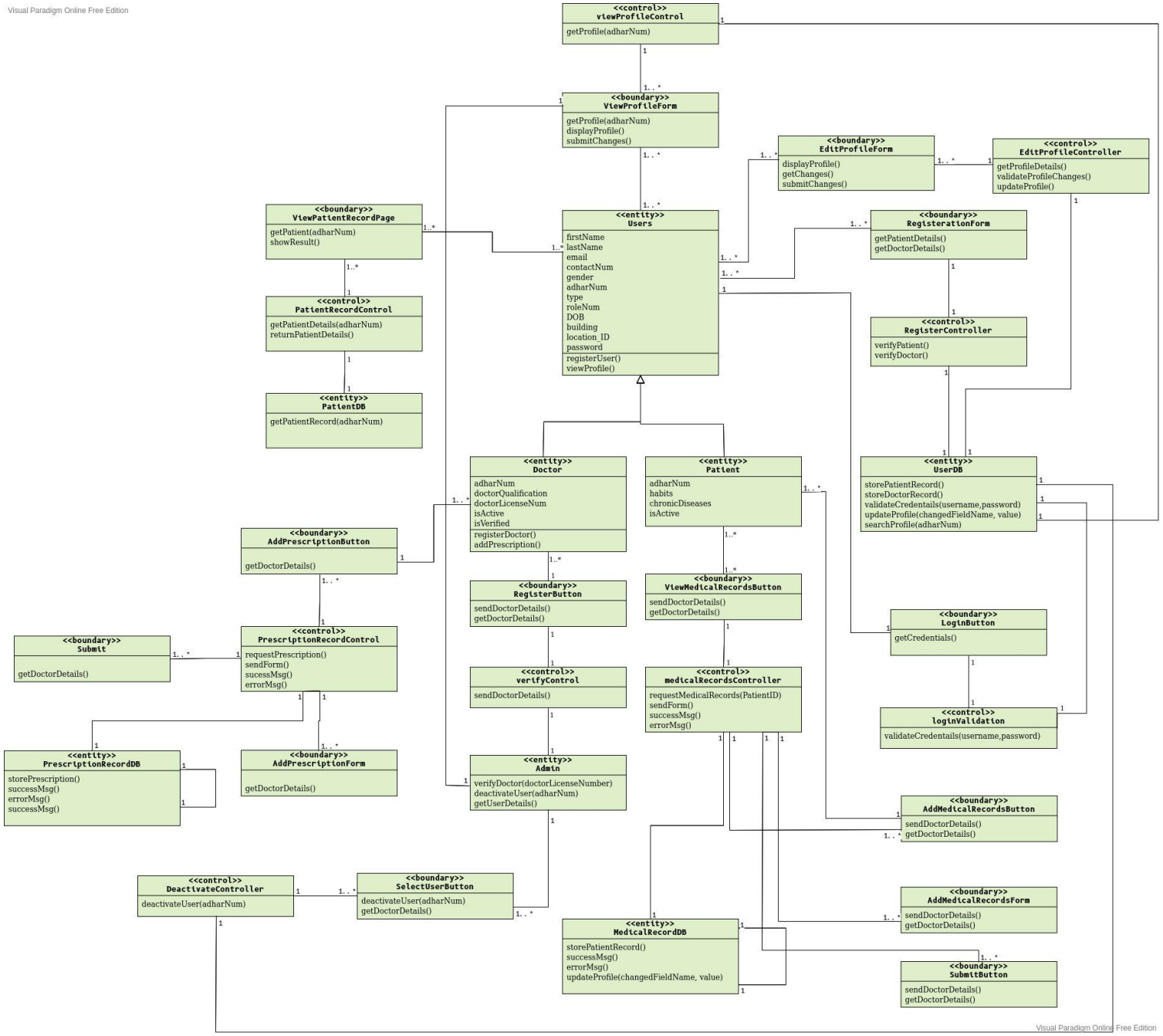
5.1.10. Deactivate Account



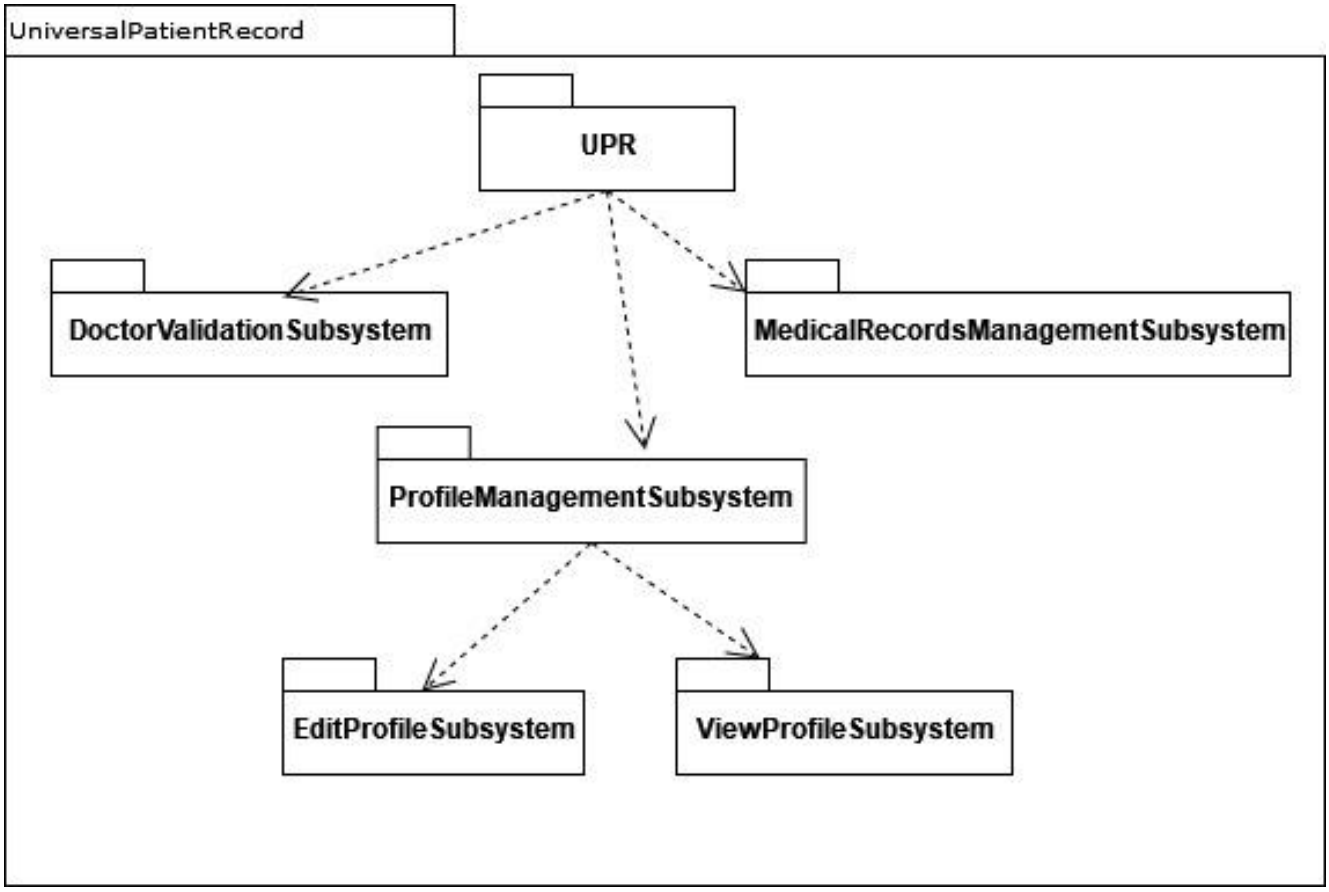
5.1.11View Doctor/Patient Profile



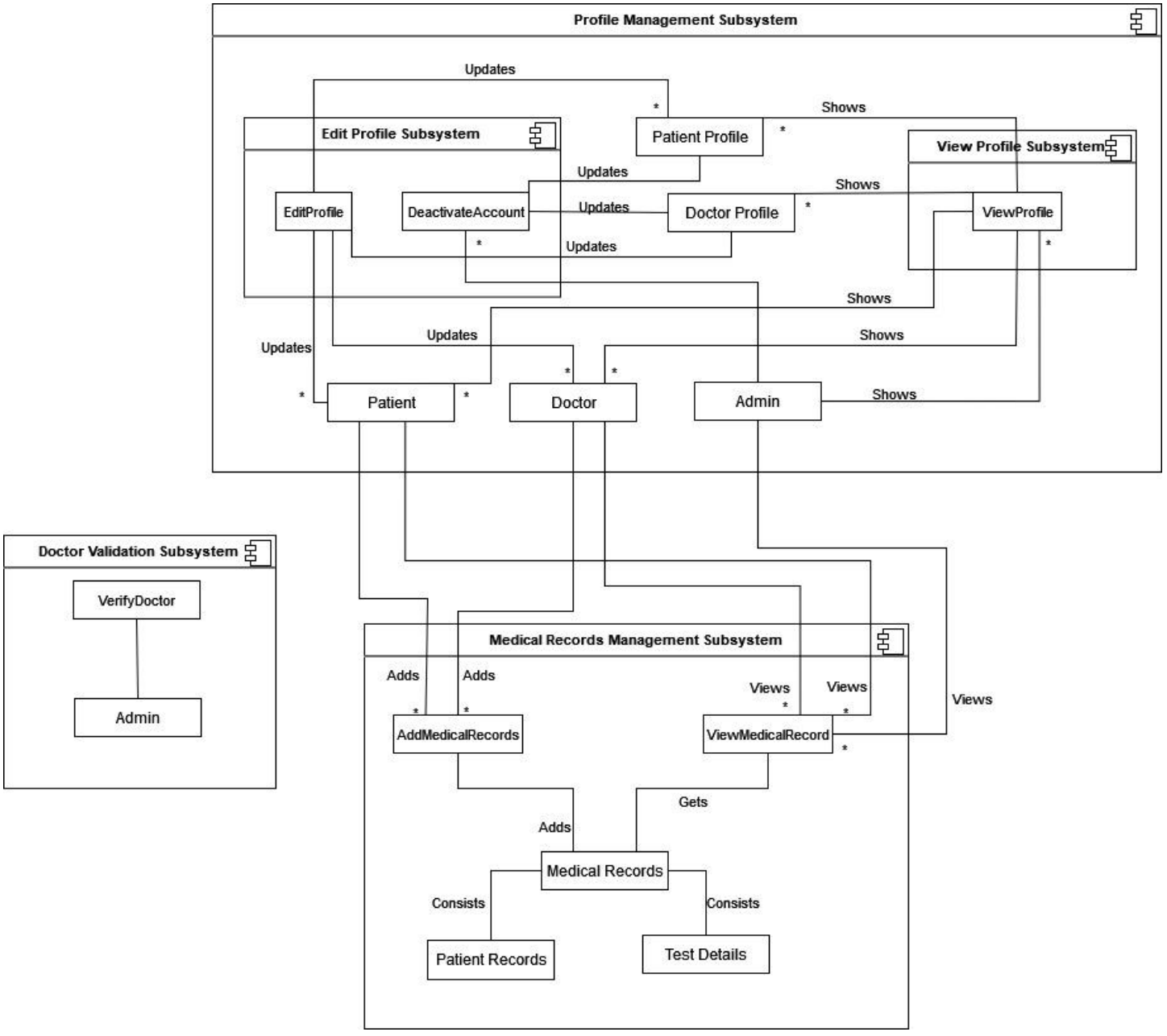
5.2. Complete Analysis Class Diagram



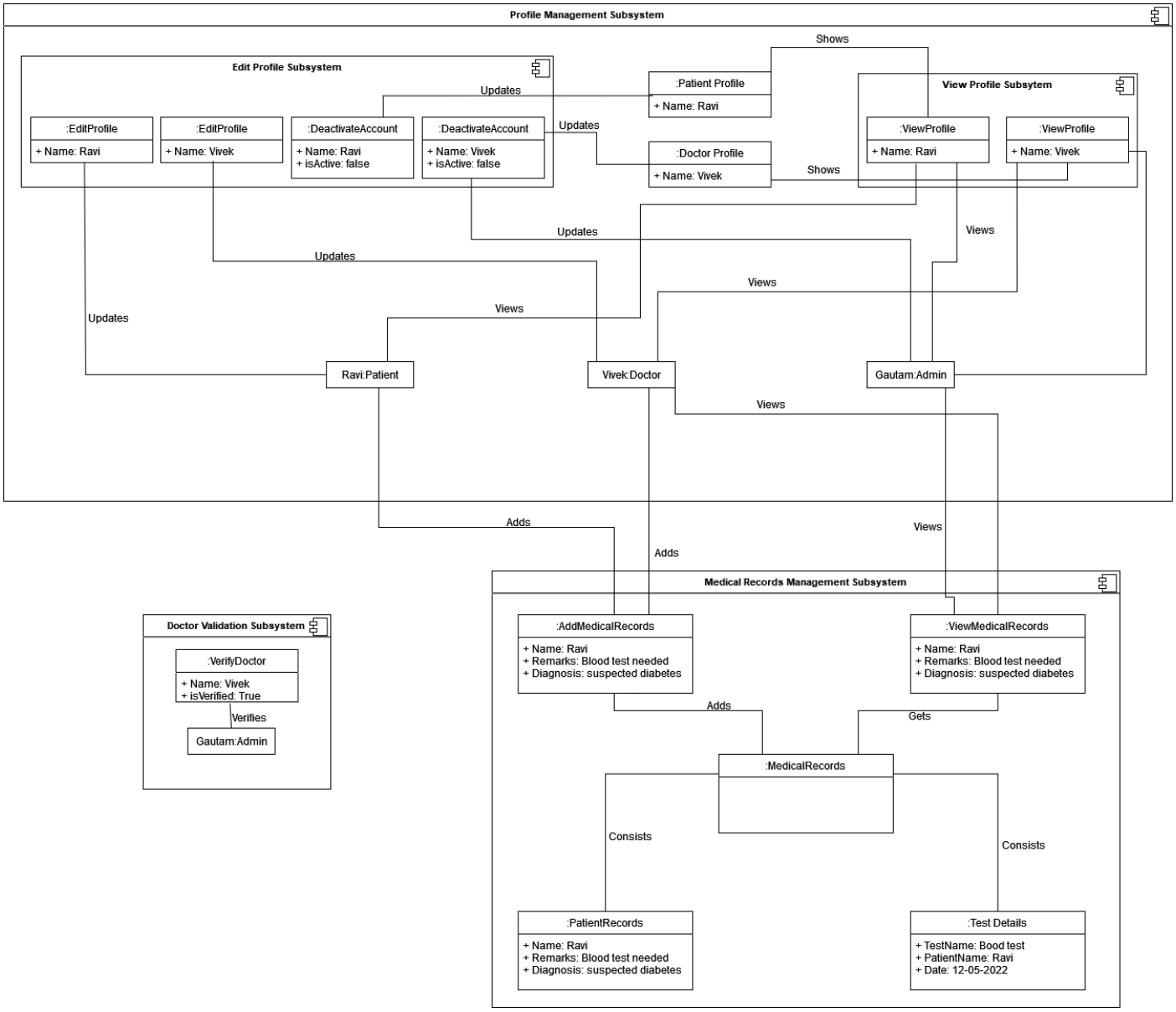
6. System Design



6.1. Sub System Design



6.2. Object Design



7. Testing Plan

Unit Testing:-

For unit testing we are using postman to test route functions and check the response returned from the server is correct or not.

System Testing -

For system testing we are using black box testing techniques like equivalence classes and boundary value analysis to test the UI validations

8. Testing Strategies and Framework(White Box testing and Black Box testing)

White Box testing:-
Unit testing:-

Test Case: Add Patient

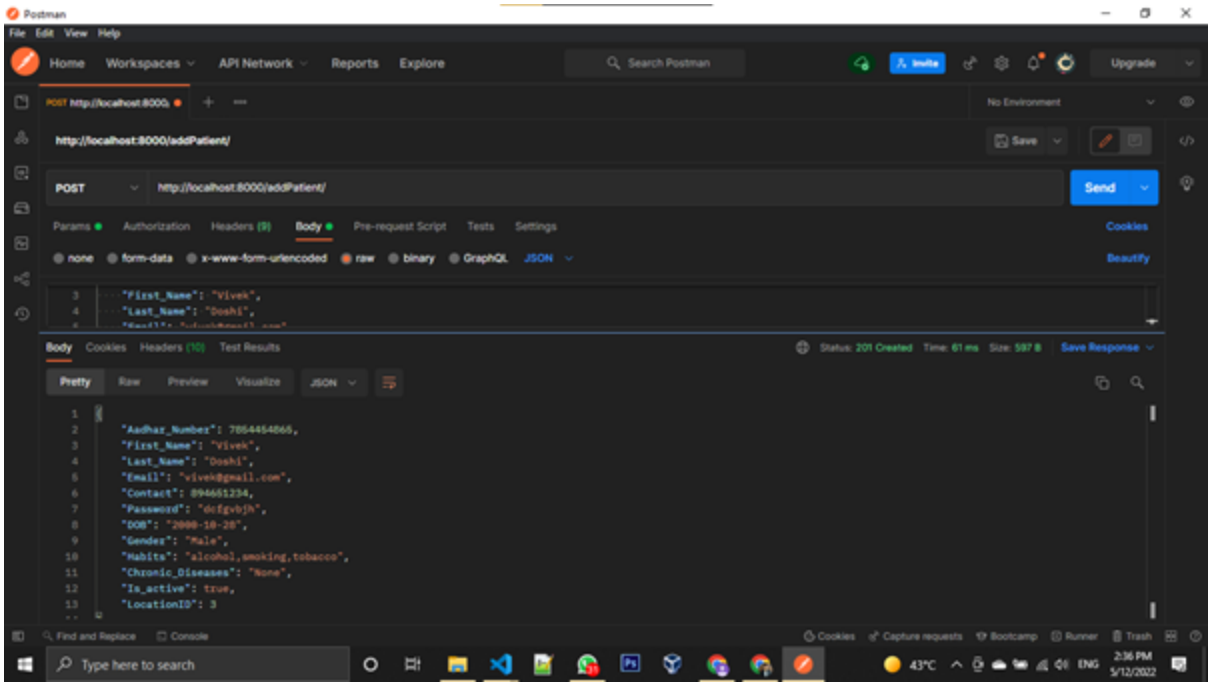
Url : http://localhost:8000/addPatient/

Expected response : Patient will be added to the database

Body :


```
{
  "Aadhar_Number": 7854454865,
            "First_Name": "Vivek",
            "Last_Name": "Doshi",
            "Email": "vivek@gmail.com",
            "Contact": 894651234,
            "Password": "dcfgvbjh",
            "DOB": "2000-10-28",
            "Gender": "Male",
            "Habits": "alcohol,smoking,tobacco",
            "Chronic_Diseases": "None",
            "Is_active": true,
            "LocationID": 3
            }
```


Response :



Output: 201 response with record inserted hence test case passed

Test Case: Add Doctor

Url : http://localhost:8000/addDoctor/

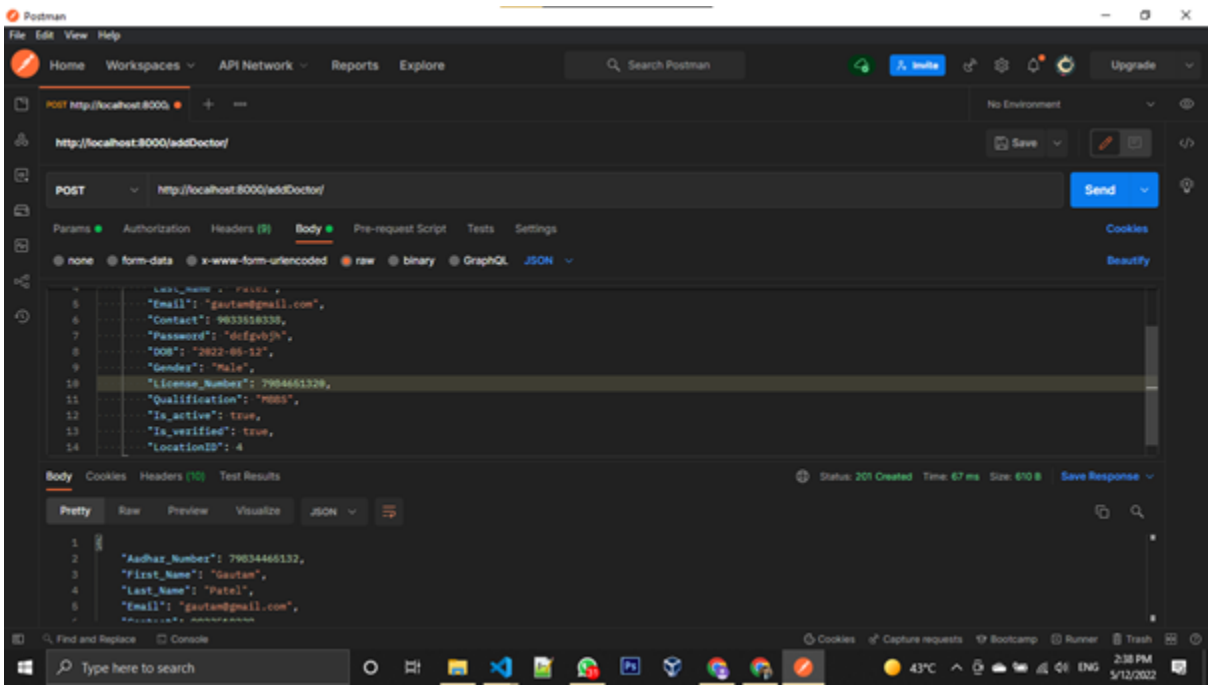
Expected Response : Doctor will be added in the database

Body :


```
{
    "Aadhar_Number": 79834465132,
    "First_Name": "Gautam",
    "Last_Name": "Patel",
    "Email": "gautam@gmail.com",
    "Contact": 9033510338,
    "Password": "dcfgvbjh",
    "DOB": "2022-05-12",
```

```
"Gender": "Male",

"License_Number": 7984651320,

"Qualification": "MBBS",

"Is_active": true,

"Is_verified": true,

"LocationID": 4

}
```


Response :



Output: 200 response with record inserted hence test case passed

Test Case: Add Consultation

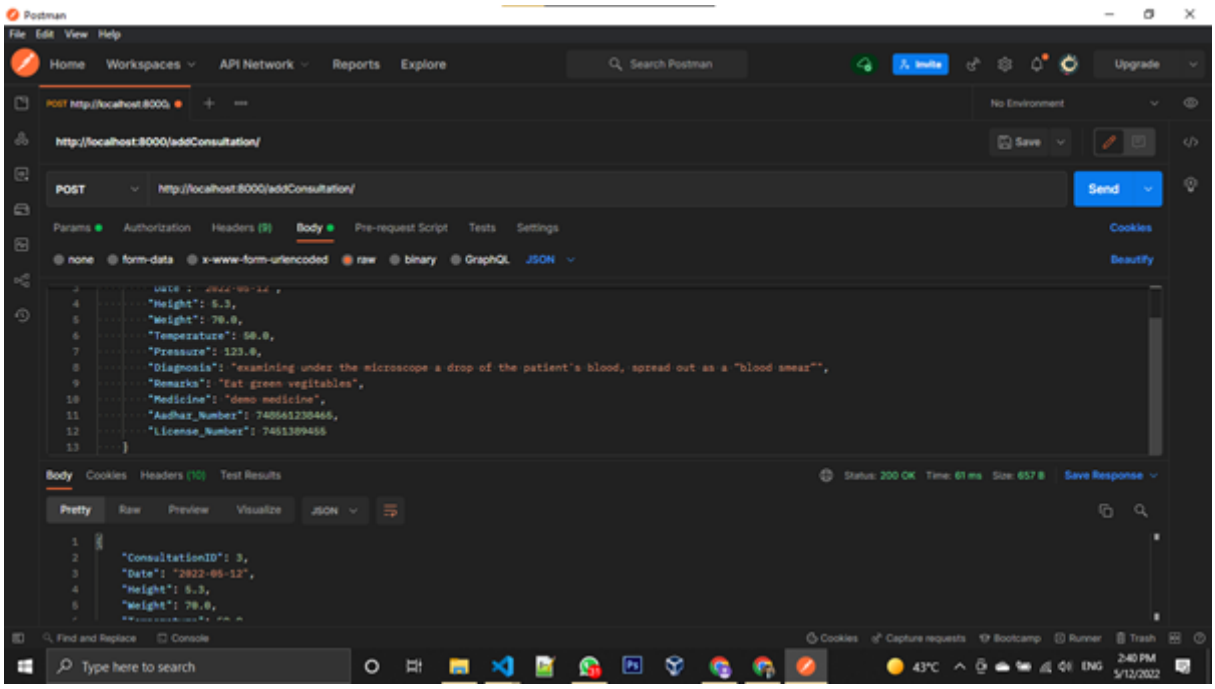
Url : http://localhost:8000/addConsultation/

Expected Response : Consultation record will be added in the database

Body :


```
{

        "Date": "2022-05-12"

        "Height": 5.3,

        "Weight": 70.0,

        "Temperature": 50.0,

        "Pressure": 123.0,

        "Diagnosis": "examining under the microscope a drop of the patient's blood, spread out as a “blood smear”",

        "Remarks": "Eat green vegetables",

        "Medicine": "demo medicine",

        "Aadhar_Number": 748561238465,

        "License_Number": 7451389455

        }
```


Response :



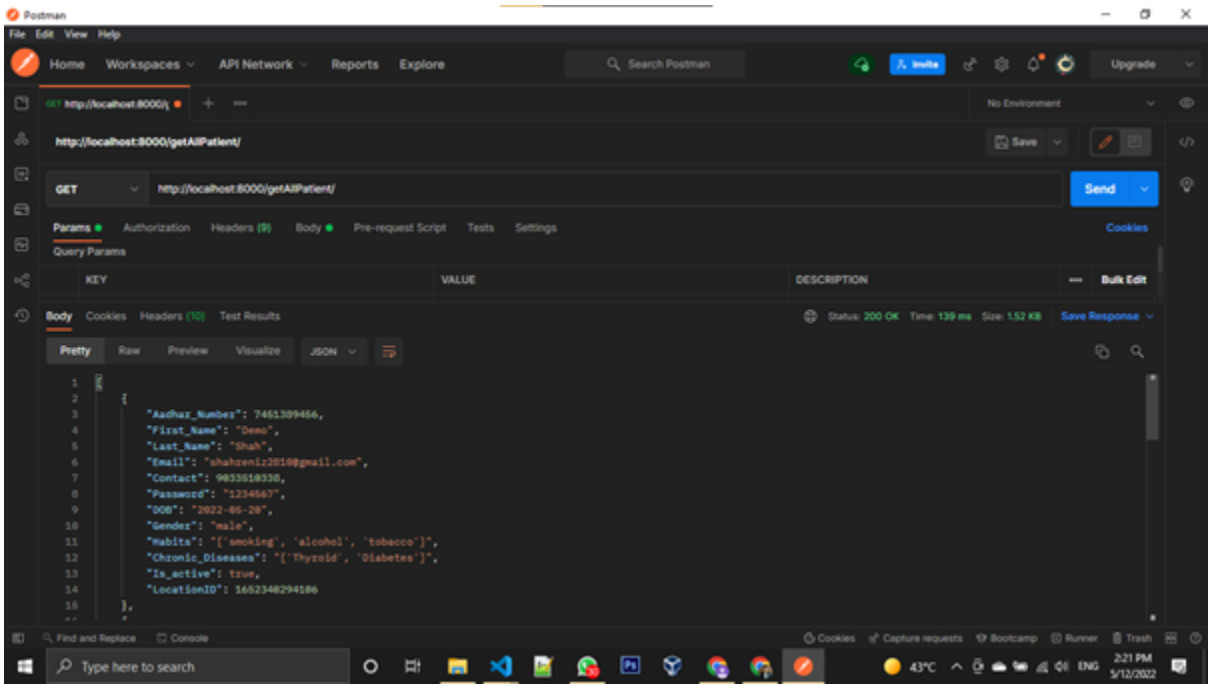
Output: 200 response with record inserted hence test case passed

Test Case: Get All Patients

Url: http://localhost:8000/getAllPatient/

Expected Response: JSON object of all Patients

Response :



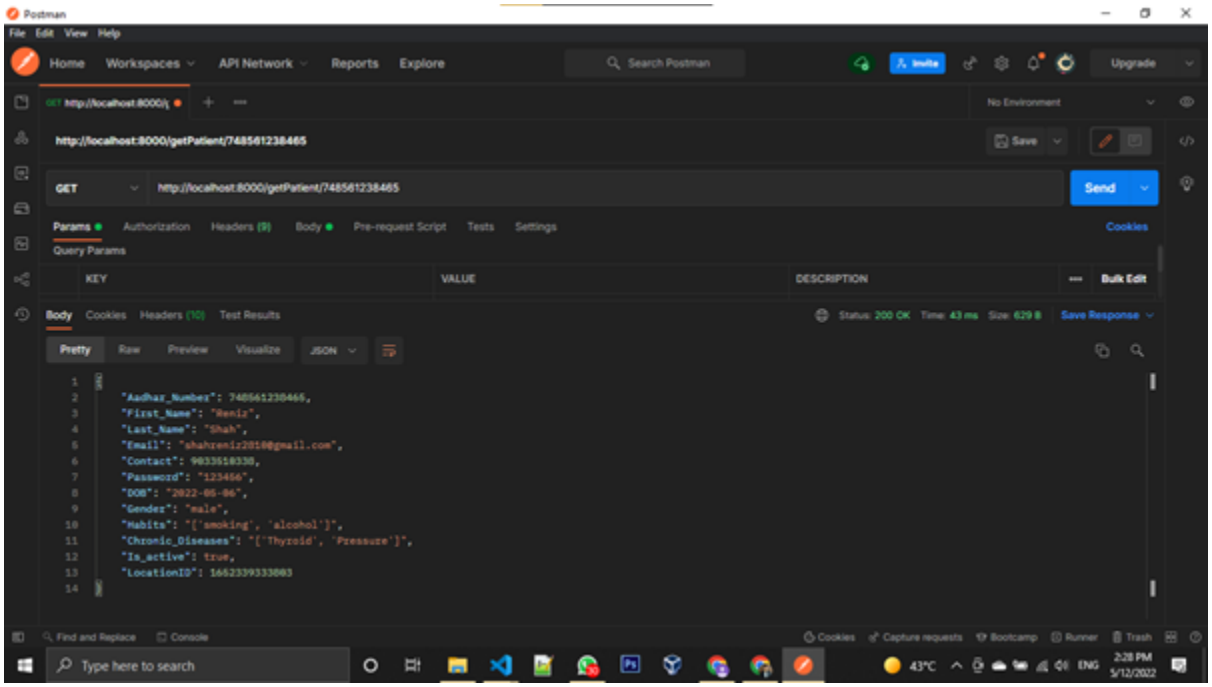
Output : Response received with all patient records hence test case passed.

Test Case: Get a Patient

Url : http://localhost:8000/getPatient/id/

Expected Response : JSON object of requested patient

Response :



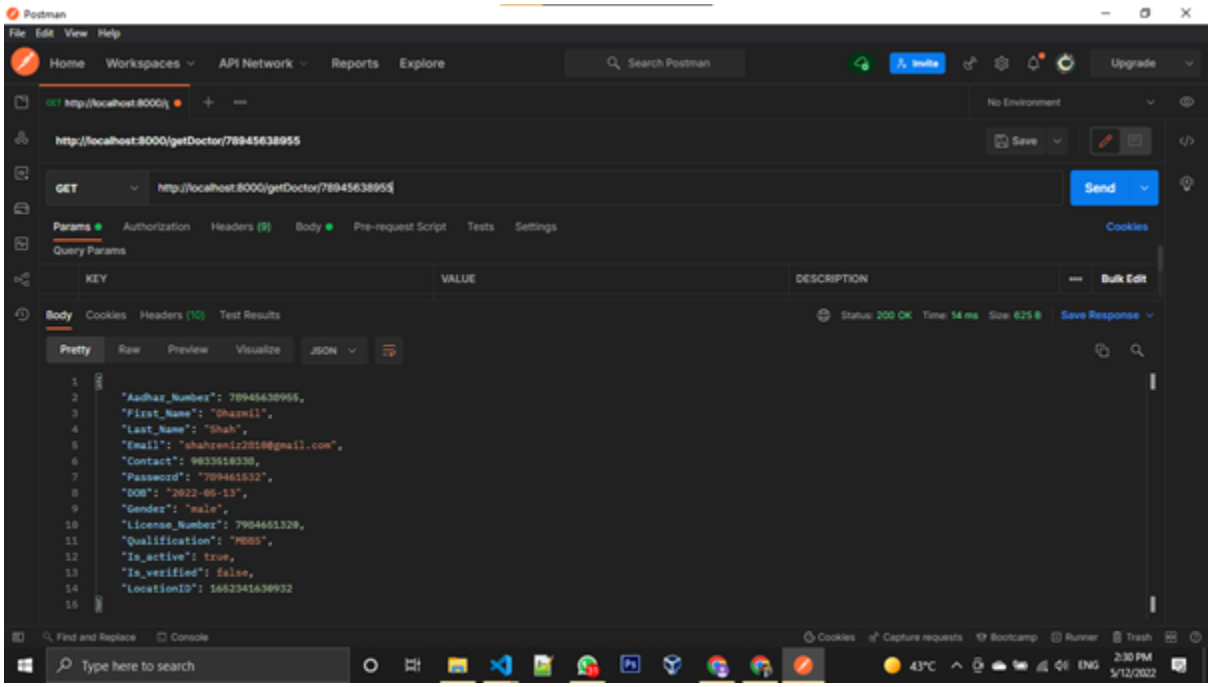
Output: Response received with only one patient record hence test case passed.

Test Case: Get a Doctor

Url : http://localhost:8000/getDoctor/id/

Expected Response : JSON object of requested doctor

Response :



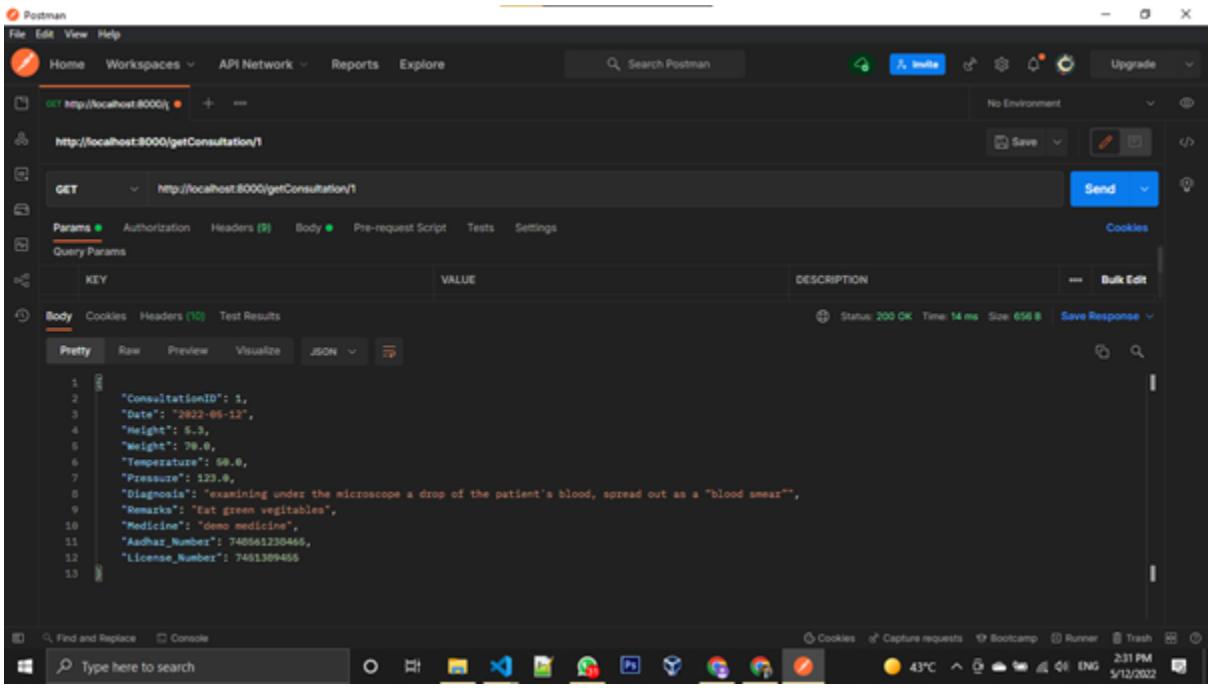
Output: Response received with only one doctor record hence test case passed.

Test case: Get a Consultation

Url : http://localhost:8000/getConsultation/id/

Expected Response : JSON object of requested Consultation record

Response :



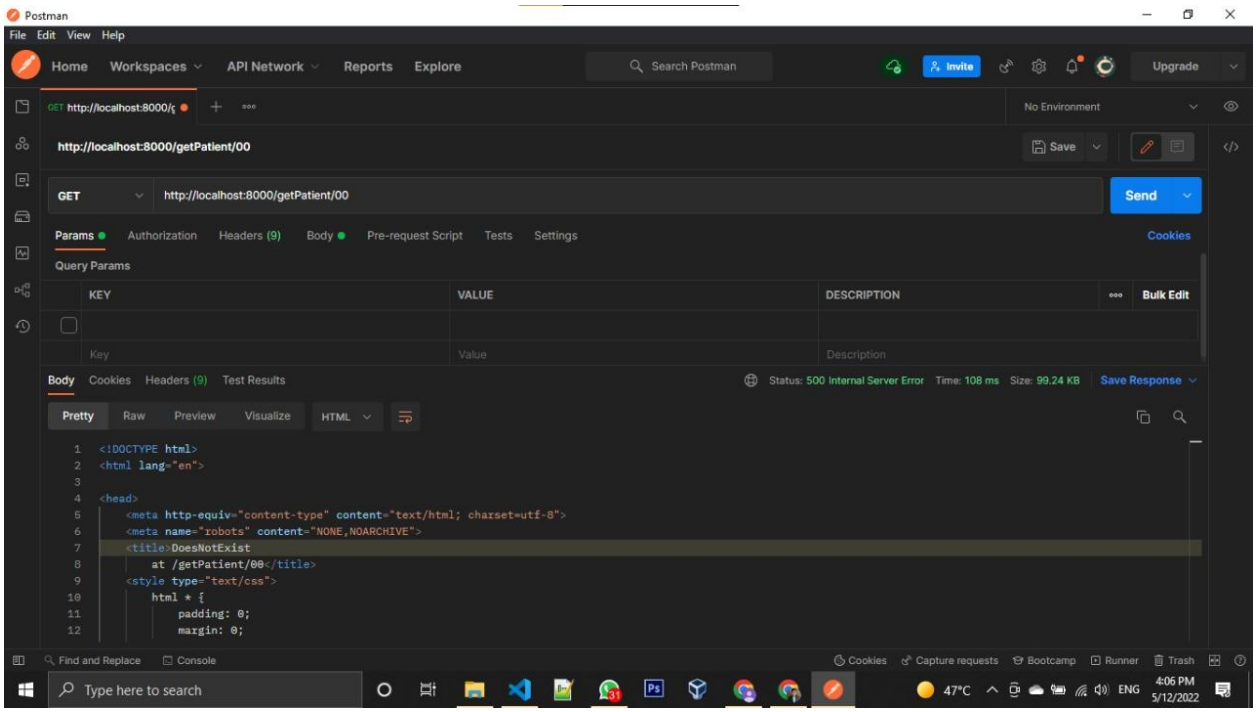
Output: Response received with only one doctor record hence test case passed.

Test Case: Patient not found

Url: http://localhost:8000/getPatient/id

Expected Response: Response containing DoesNotExist message

Response :



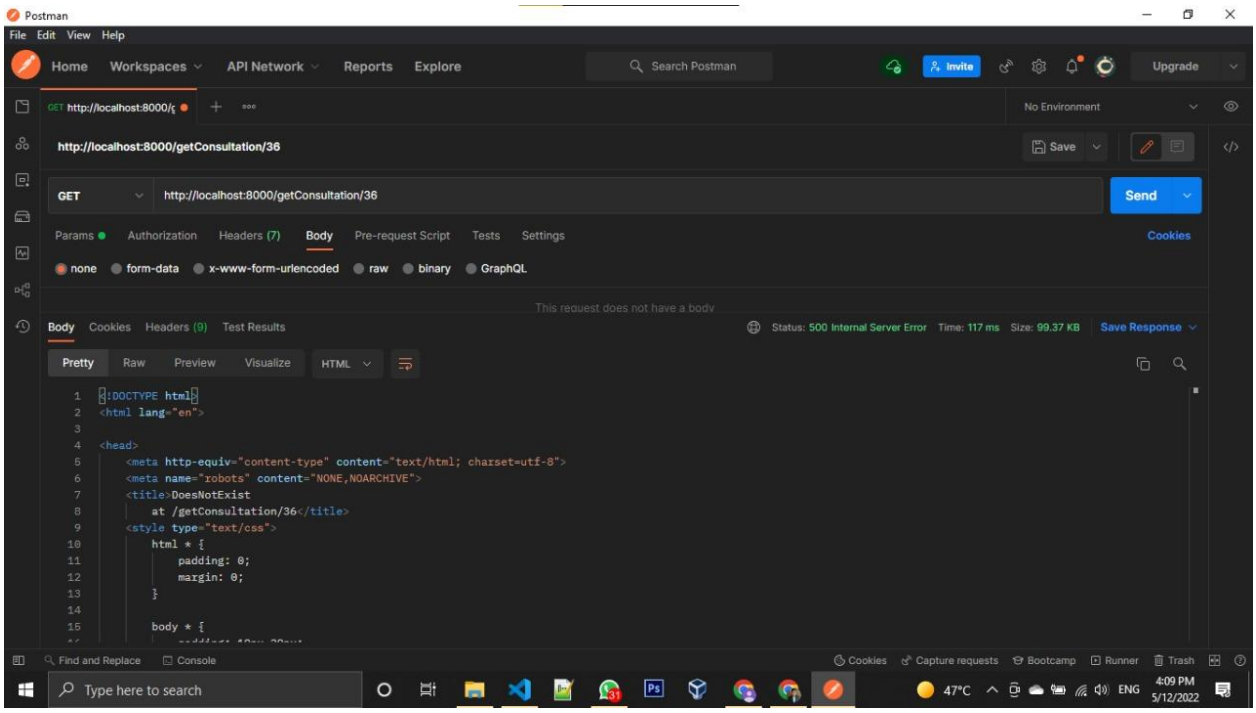
Output: Response with 500 status error indicating patient does not exists.

Test Case: Consultation not found

Url: http://localhost:8000/getConsultation/id

Expected Response: Response containing DoesNotExist message

Response :



Output: Response with 500 status error indicating consultation does not exist.

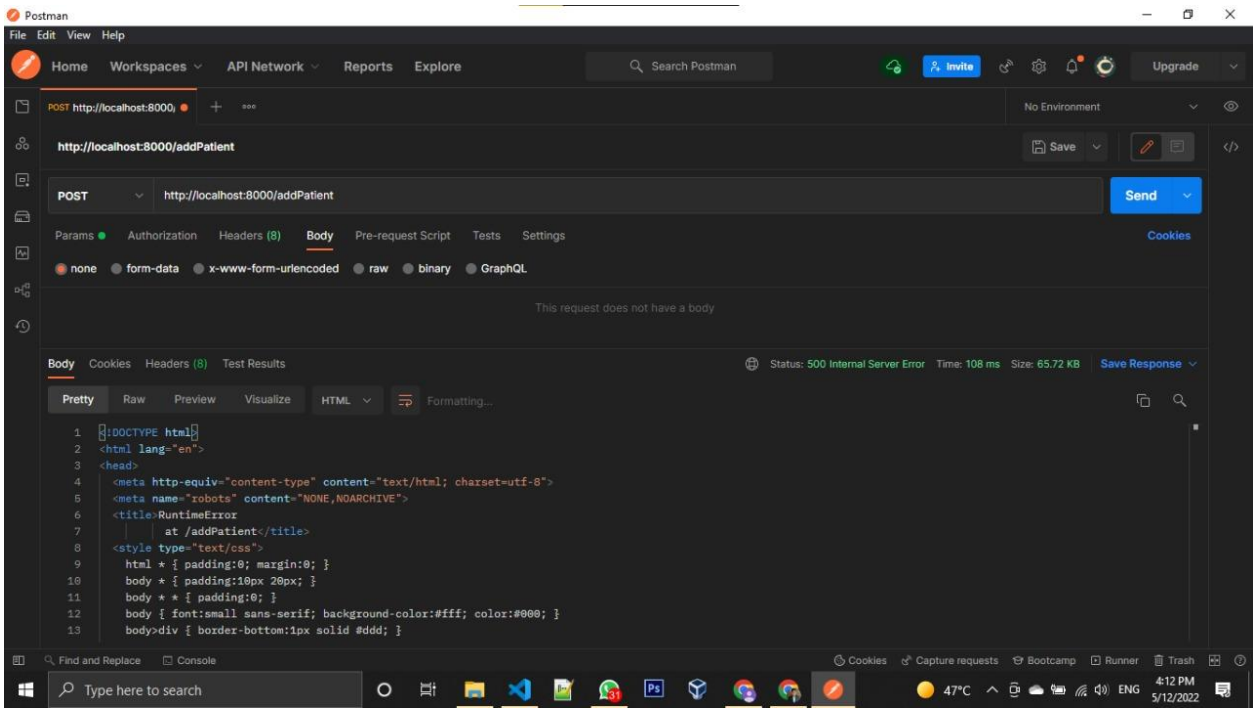
Test Case: Empty patient record post error

Url: http://localhost:8000/addPatient

Body: none

Expected Response: 500 server error

Response :



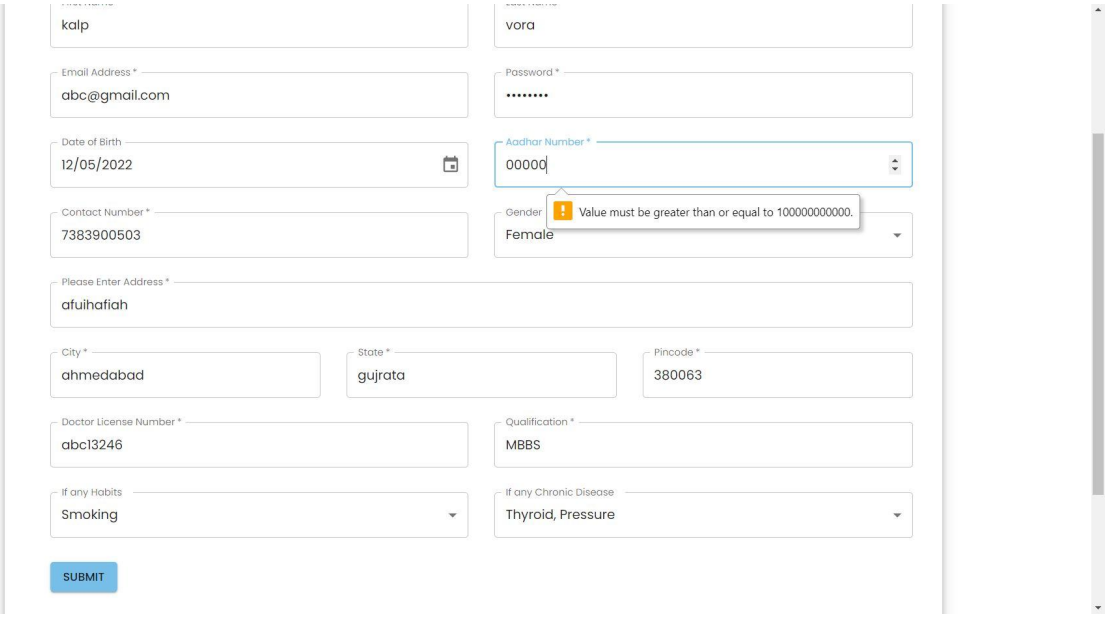
Output: Response with 500 status error indicating patient with empty body is not added.

Black box testing:-

Registration Flow:-

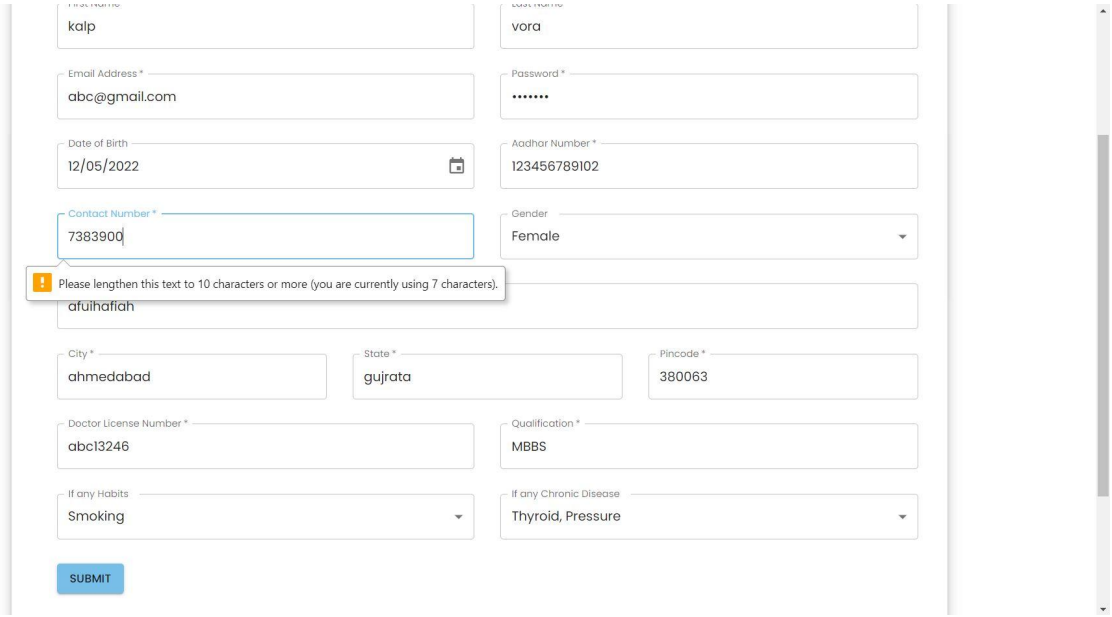
SR	Class	Valid/Invalid
C1	aadharNum.length > 12	Invalid
C2	aadharNum.length = 12	Valid
C3	aadharNum.length < 12	Invalid

Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
aadharNum = 894563251478	Successfully Registered	C2
aadharNum = 8945632514789888	Register Unsuccessful	C1
aadharNum=89456	Register Unsuccessful	C3
Boundary Value Analysis		
aadharNum = 89456325147	Register Unsuccessful	C3
aadharNum = 8945632514789	Register Unsuccessful	C1



SR	Class	Valid/Invalid
C1	phoneNum.length > 10	Invalid
C2	phoneNum.length = 10	Valid
C3	phoneNum.length < 10	Invalid

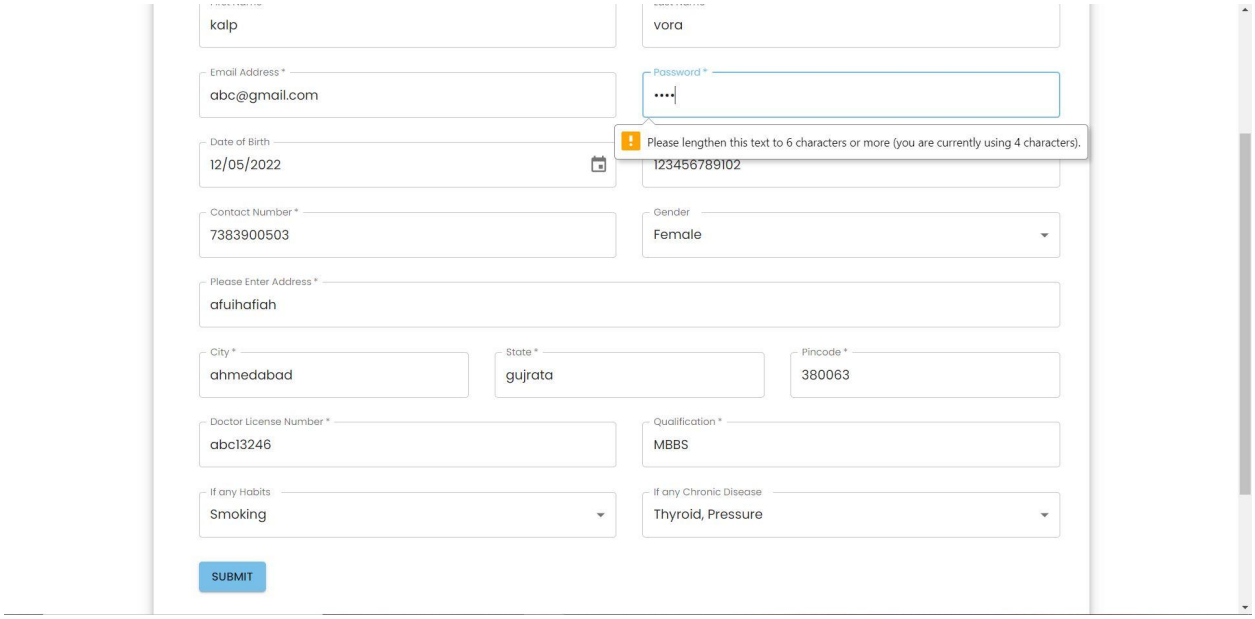
Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
phoneNum = 9978184556	Successfully Registered	C2
phoneNum = 99781845556667	Register Unsuccessful	C1
phoneNum = 99781	Register Unsuccessful	C3
Boundary Value Analysis		
phoneNum = 997818455	Register Unsuccessful	C3
phoneNum = 997818455566	Register Unsuccessful	C1



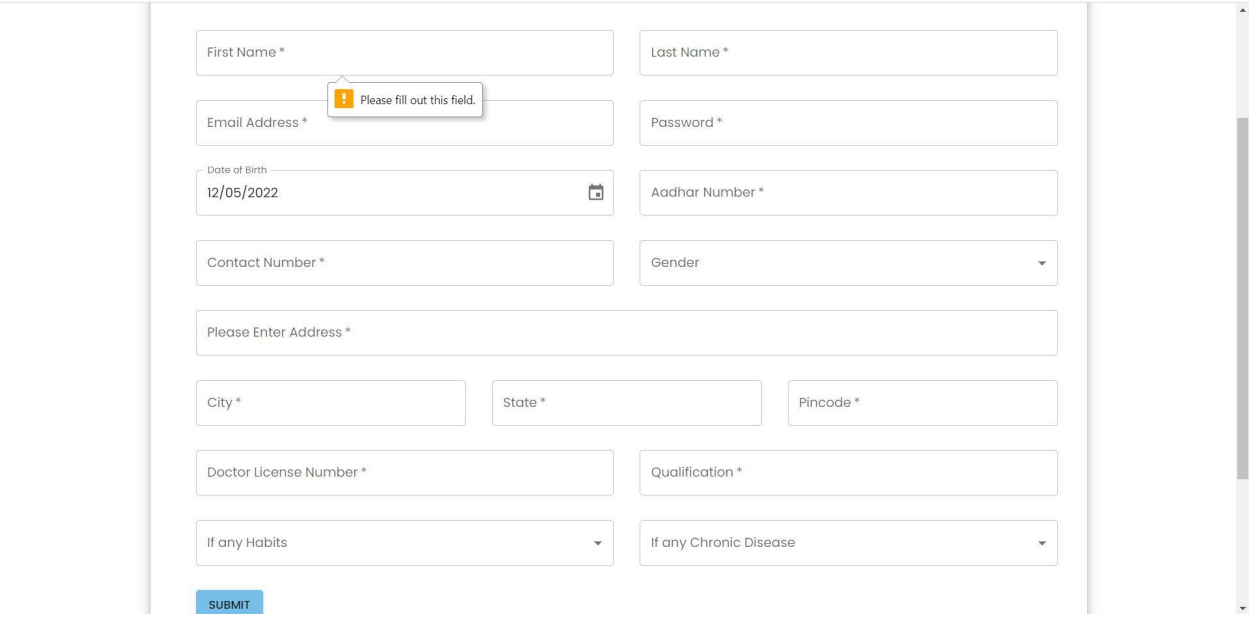
SR	Class	Valid/Invalid
C1	password.length < 6	Invalid
C2	password.length >= 6	Valid

Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
password = abc	Register Unsuccessful	C1
password = abdcdefgh	Successfully Registered	C2
Boundary Value Analysis		
password = abdcdef	Successfully Registered	C2
password = abdcde	Register Unsuccessful	C1

password = abdcdefg	Successfully Registered	C2



SR	Class	Valid/Invalid
C1	Submit Empty fields	Invalid



SR	Class	Valid/Invalid
C1	pincode.length > 6 && pincode contains only numbers	Invalid
C2	pincode.length = 6 && pincode contains only numbers	Valid
C3	pincode.length < 6 && pincode contains only numbers	Invalid
C4	pincode.length > 6 && pincode contains alphabets	Invalid
C5	pincode.length = 6 && pincode contains alphabets	Invalid
C6	pincode.length < 6 && pincode contains alphabets	Invalid

Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
pincode=32444444	Register Unsuccessful	C1
pincode=382475	Successfully Registered	C2
pincode=382	Register Unsuccessful	C3
pincode=38a	Register Unsuccessful	C6
pincode=3244444a	Register Unsuccessful	C4
Boundary Value Analysis		
pincode=3824756	Register Unsuccessful	C1
pincode=38247	Register Unsuccessful	C3
pincode=38247f	Register Unsuccessful	C5

Add medical record:-

SR	Class	Valid/Invalid
C1	Doctor License Number field empty	Invalid
C2	Aadhar Number field empty	Invalid
C3	Date field empty	Invalid
C4	Diagnosis field empty	Invalid
C5	All other fields empty other than Doctor Licence, Aadhar Number, Date and Diagnosis	Valid

Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
Doctor License Number= ABCPQR1234, Aadhar Number= 123456789123, date= 12/5/2022, height= , weight= , pressure= , temperature= , medicine= , Diagnosis= ABC, remarks=	Consultation Added	C5
Doctor License Number= , Aadhar Number= 123456789123, date= , height= , weight= , pressure= , temperature= , medicine= , Diagnosis= ABC, remarks=	Fill out this field	C1, C3
Doctor License Number= ABCPQR1234, Aadhar Number= 123456789123, date= 12/5/2022, height= , weight= , pressure= , temperature= , medicine= , Diagnosis= , remarks=	Fill out this field	C4
Doctor License Number= ABCPQR1234, Aadhar Number= , date= 12/5/2022, height= , weight= , pressure= , temperature= , medicine= , Diagnosis= ABC, remarks=	Fill out this field	C2
Doctor License Number= ABCPQR1234, Aadhar Number= 123456789123, date= 12/5/2022, height=165 , weight=65 , pressure=100 , temperature=93.5 , medicine=XYZ , Diagnosis= ABC, remarks= No remarks	Consultation Added	C5

Edit Profile flow:-

SR	Class	Valid/Invalid
C1	Email changed improper format	Invalid
C2	Email changed proper format	Valid
C3	Phone number changed improper format	Invalid
C4	Phone number changed proper format	Valid

Test Action and Input Data	Expected Outcome	Class
Equivalence Class		
email=agarwalravigmail@.com	Edit Unsuccessful	C1
email=agarwalravil@gmail.com	Successfully Edited	C2
phone=869099342222	Edit Unsuccessful	C3
phone=8690993422	Successfully Editedl	C4

9. Challenges you faced:

During the integration process, we found that there are some anomalies and redundant fields that should be removed from the schema. This helped us in building a normalized database.

It becomes difficult to take all the records/details from the patient. So, finding out the most important fields and taking the details for the same from patients took a lot of brainstorming on what fields to opt for and what fields to avoid.

The admin also holds the right to block/unblock any patient/doctor. So, identifying the process to block/unblock any user and implementing the same was a challenging task.

9.1. Open Issues

Aadhaar based user verification: Since we are taking aadhaar number during registration, user details can be verified.

An optimized verification process for doctor accounts: For now doctor verification is a manual process so we can add a system to validate the basic details of doctors like license number.

10. Lessons Learned

This project helped us understand the software engineering concepts on a deeper level by applying them at different stages of software development.
We also learned different tools for software engineering including postman for backend testing. As the group size was 10 students we learned how to manage and coordinate among ourselves both offline and online.

Contribution:

Name (ID)	Contribution
Anil Mantani (202112005)	Frontend, Documentation,Testing
Parth Soni (202112007)	Frontend, Documentation, Presentation
Khusboo Chhabra (202112014)	Documentation, Testing, Backend
Kalp Vora (202112018)	Frontend, Testing,Integration
Ravi Agarwal (202112021)	Documentation, Testing, Backend
Nikhil Chhabria (202112026)	Documentation, Testing, Backend
Arpitha Sreenivasan (202112030)	Backend(Schema), Documentation
Vivek Doshi (202112063)	Documentation, Backend(Schema), Testing
Gautam Patel (202112086)	Frontend, Testing, Integration
Reniz Shah (202112126)	Backend,Integration, Testing