

Chapter 1:- Introduction

The proposed project is a smart appointment booking system that provides patients or any user an easy way of booking a doctor's appointment online. This is a web based application that overcomes the issue of managing and booking appointments according to user's choice or demands. The task sometimes becomes tedious and consumes time of the doctor in manually allotting appointments for the patients as per their availability. Hence this project offers an effective solution where users can view various time slots and select the preferred date and time. This system also allows patients to cancel their booking anytime. The web application is developed using HTML, CSS and Bootstrap as a front-end and PHP and MySQL database as the back-end.

Modules:

- Admin
- User/Patient
- Doctor

Chapter 2:- System Analysis

2.1. Study of current System:

Few patients do not book their appointment in advance due to this it gets crowded in hospital/clinic and patients have to wait until their turn comes. And this causes waste of time and unwanted waiting period for patients.

Also there are some related systems available right now but no one is totally similar. Many of the system have some limitations. From the study of this similar project I got interested to develop this system.

2.2. Problem and weakness of Current System

- No user login option.
- Payment before Appointment.
- Doctors were getting many requests which they were not able to fulfil.
- Difficult for layman.
- Users could only see doctor's location and visit-time.

2.3. Requirement analysis of New System

Software Requirements:-

- Operating System – Windows
- Front end – HTML, CSS, Bootstrap, Javascript
- Back end – MySql, PHP
- Server – XAMPP
- Web Browser – Google chrome, Microsoft edge, etc.

Hardware Requirements:-

- RAM – at least 2 GB
- Processor - 1.9 gigahertz

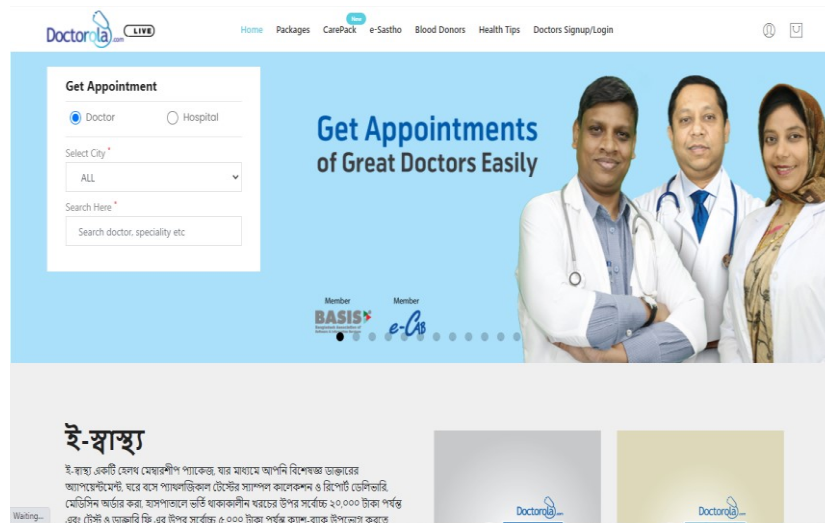
Functional Requirements:-

- FR01: The user should be able to register and login into the system.
- FR02: The user should be able to search doctor, book appointment, delete appointment, and download appointment slip.
- FR03: The user can change his profile info at any time.
- FR04: Database has to store all the information efficiently without any information loss.
- FR05: Doctors should be able to accept/reject appointment request sent by patients.
- FR06: Doctors should be able to schedule/update their time slots.
- FR07: Admin should be able to accept/reject registration request sent by doctors.
- FR08: Admin should be able to manage doctors and patients.
- FR08: Admin should be able to view appointments of system.

Non Functional Requirements:-

- **Performance Requirements:**
 - The system need to be reliable
 - If unable to process the request then appropriate error message is printed.
 - Web pages are loaded within few seconds
- **Safety Requirements:**
 - Doctors and patients must be authenticated.
- **Security Requirements:**
 - After entering the password and email the user can access his profile.

2.4. Brief Literature Review



[Fig-2.4.1: Doctorrola.com]

<http://doctorola.com/>

I have explored few websites which are related to doctor appointment booking system, first my attention caught in ‘Doctorrola.com’. In their system user need to search for doctor or hospital from different location and get them for booking appointment. There is no user login option and personal profile so users are totally detached from getting extra facilities in future purpose. And ‘Doctorsbd.com’ site provides only doctors list. Users can only able to know their service location from this site.

2.5. Design: Analysis, Design Methodology and Implementation Strategy

- I have used Iterative Model for my project in which I started with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.
- The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

- Also testing and debugging is easy during smaller iteration and parallel development can done. Risks are identified and resolved during iteration.

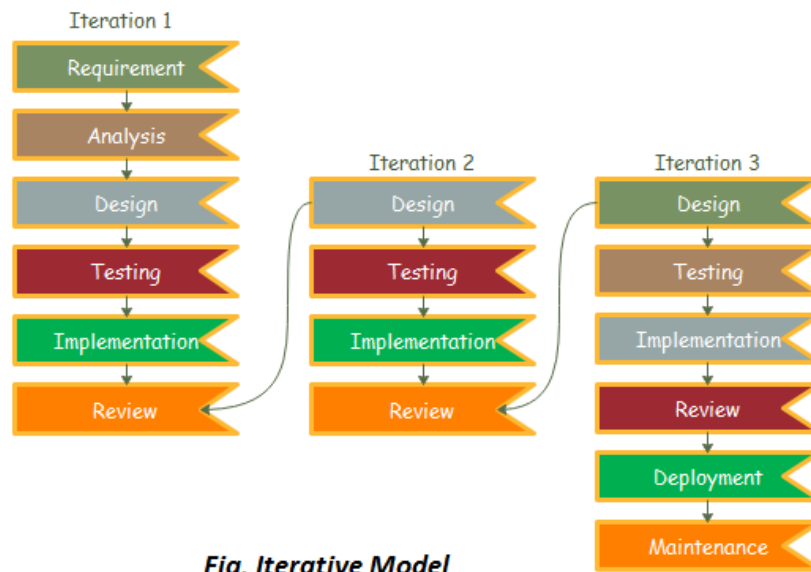


Fig. Iterative Model

[Fig-2.5.1: Iterative model]

- **Planning & Requirements:** As with most any development project, the first step is go through an initial planning stage to map out the specification documents, establish software or hardware requirements, and generally prepare for the upcoming stages of the cycle.
- **Analysis & Design:** Once planning is complete, an analysis is performed to nail down the appropriate business logic, database models, and the like that will be required at this stage in the project. The design stage also occurs here, establishing any technical requirements (languages, data layers, services, etc) that will be utilized in order to meet the needs of the analysis stage.
- **Implementation:** With the planning and analysis out of the way, the actual implementation and coding process can now begin. All planning, specification, and design docs up to this point are coded and implemented into this initial iteration of the project.

- **Testing:** Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have have cropped up.
- **Evaluation:** Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on.

Advantages of Iterative Model –

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Risk analysis is better.
- It supports changing requirements.

Disadvantages of the Iterative Model –

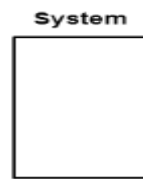
- Each phase of an iteration is rigid with no overlaps.
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle.

Chapter 3: - System Modeling

3.1 Use Case Diagram:

System boundary:

A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined. A system boundary of a use case diagram defines the limits of the system.

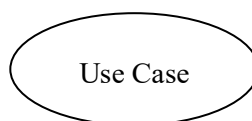


Use case:

Each use case is represented by an ellipse with the name of the use case written inside the ellipse, named by verb.

All the use cases are enclosed with a rectangle representing system boundary. Rectangle contains the name of the system.

It identifies, clarifies and analyzes the functional requirements of the system.



Actor

An actor is anything outside the system that interacts with it, named by noun.

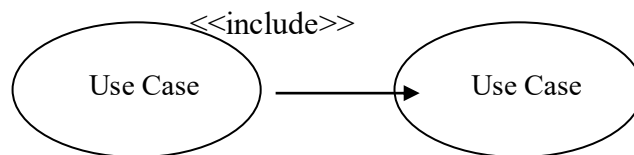
Actors in the use case diagram are represented by using the stick person icon. An actor may be a person, machine or any external system.

In use case diagrams, actors are connected to use case by drawing a simple line connected to it. Actor triggers use case.

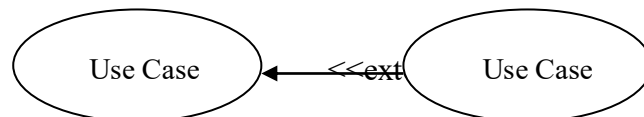


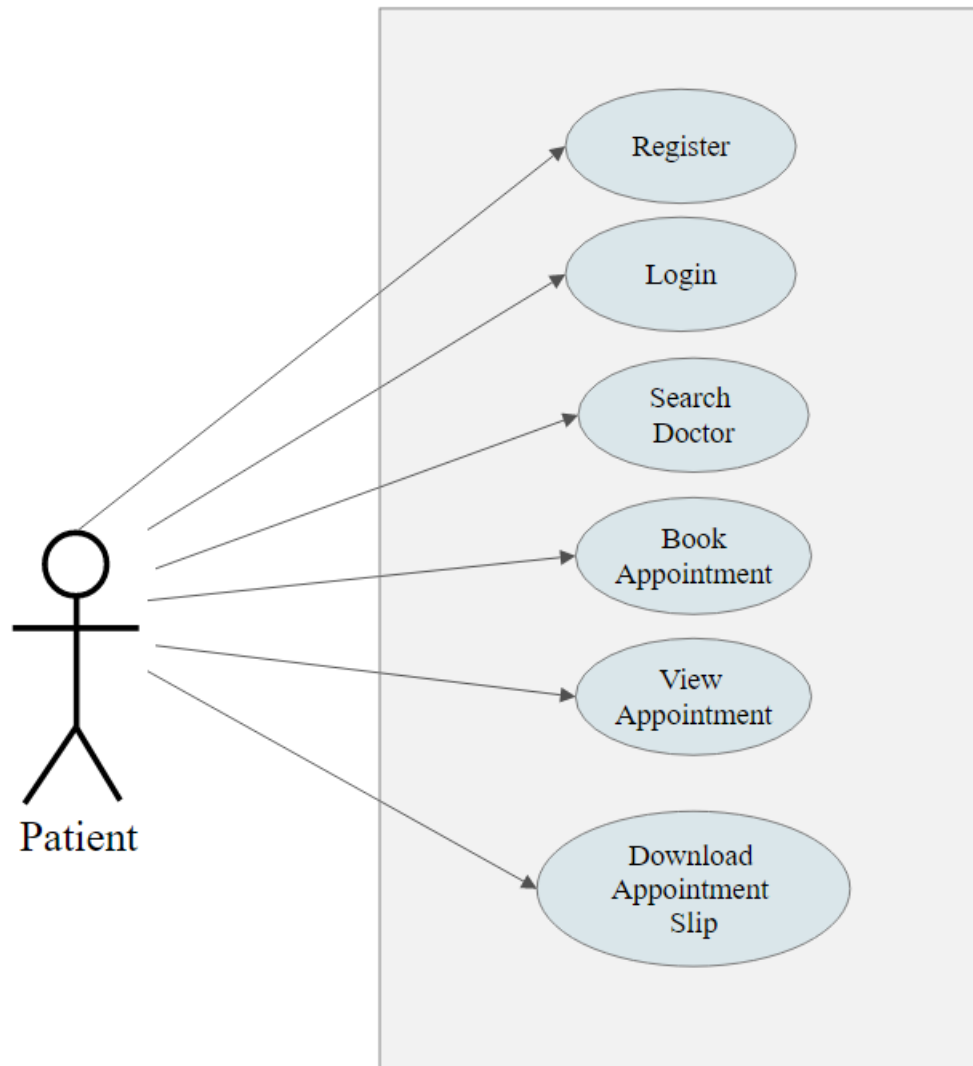
Relationship:

Include: When a use case is depicted as using the functionality of another use case in a diagram, this relationship between the use cases is named as an *include* relationship. Literally speaking, in an *include* relationship, a use case includes the functionality described in another use case as a part of its business process flow. An include relationship is depicted with a directed arrow having a dotted shaft.

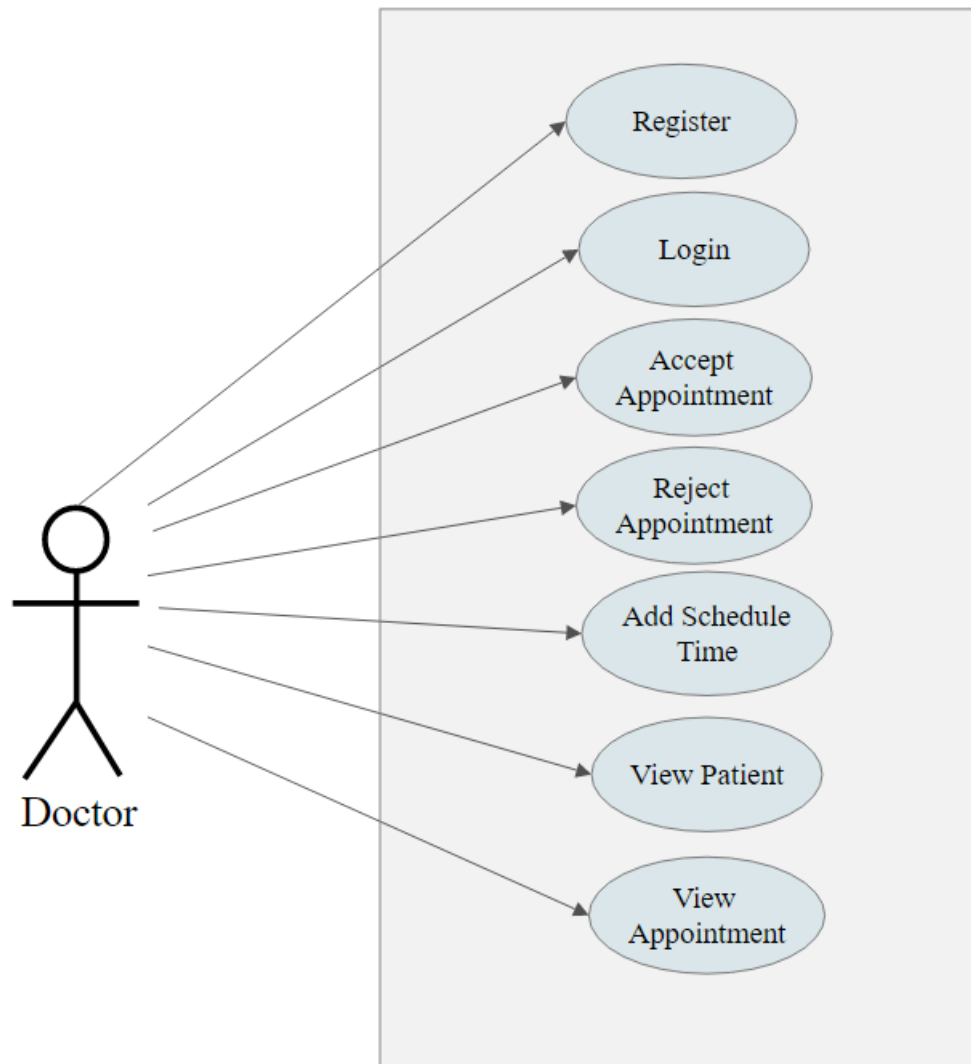


Extend: In an *extend* relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case. An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow.

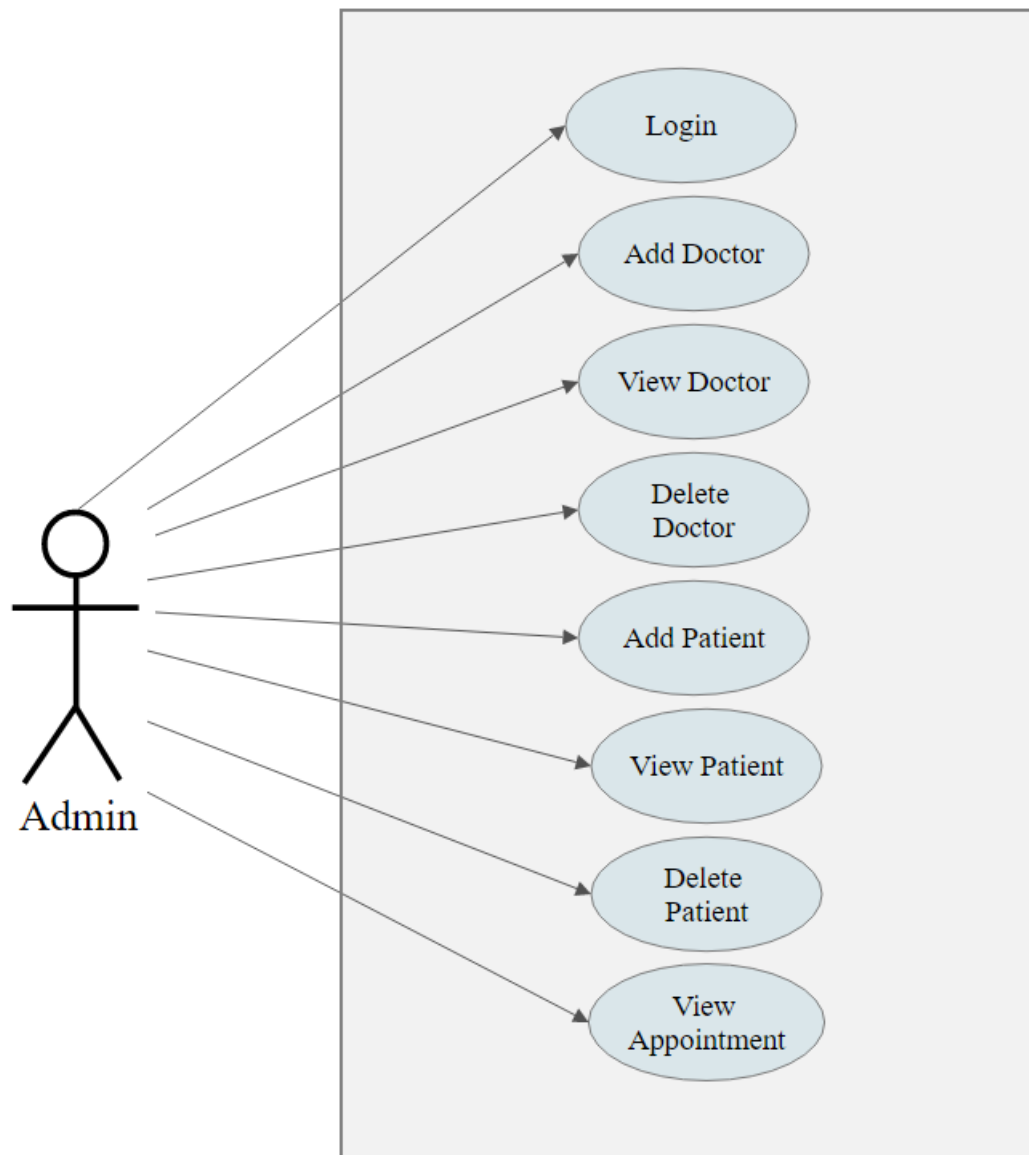




[Fig-3.1.1: Use case Diagram- Patient]



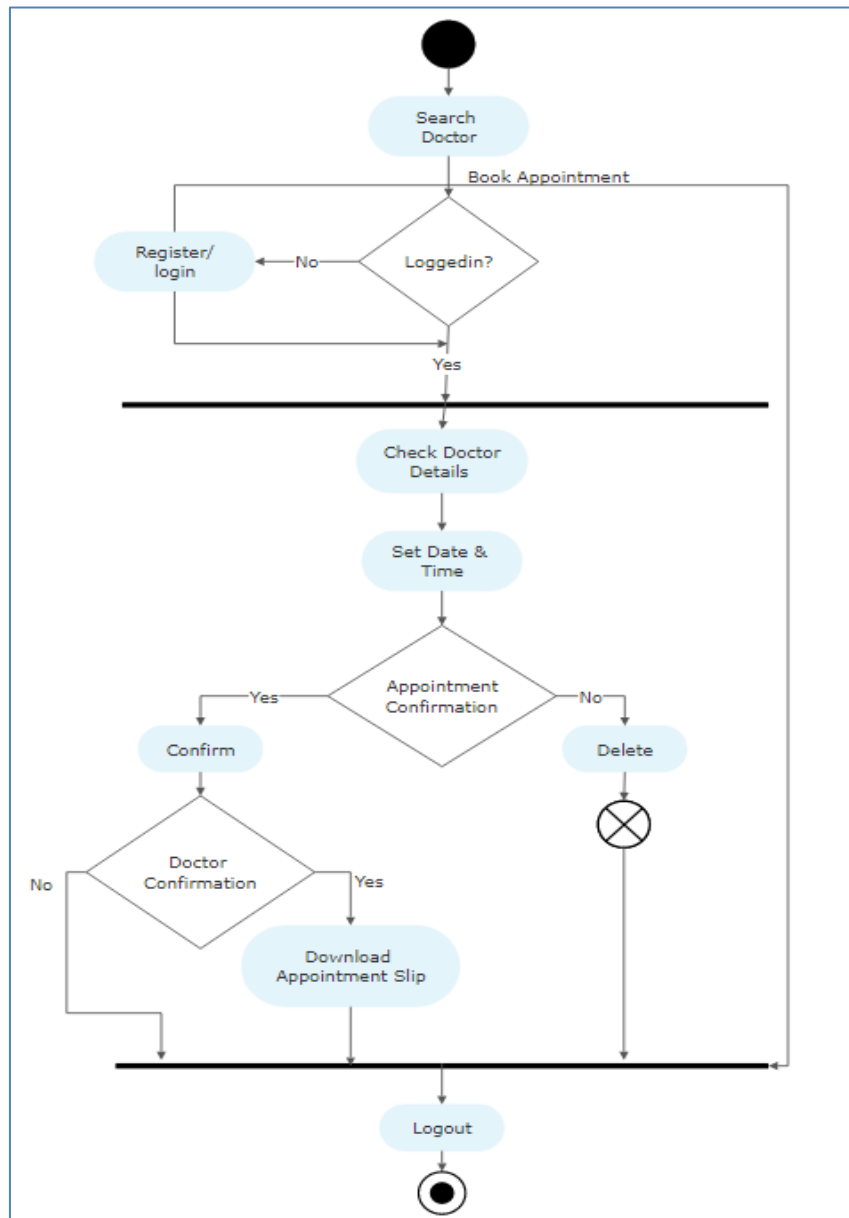
[Fig: 3.1.2- Use case Diagram- Doctor]



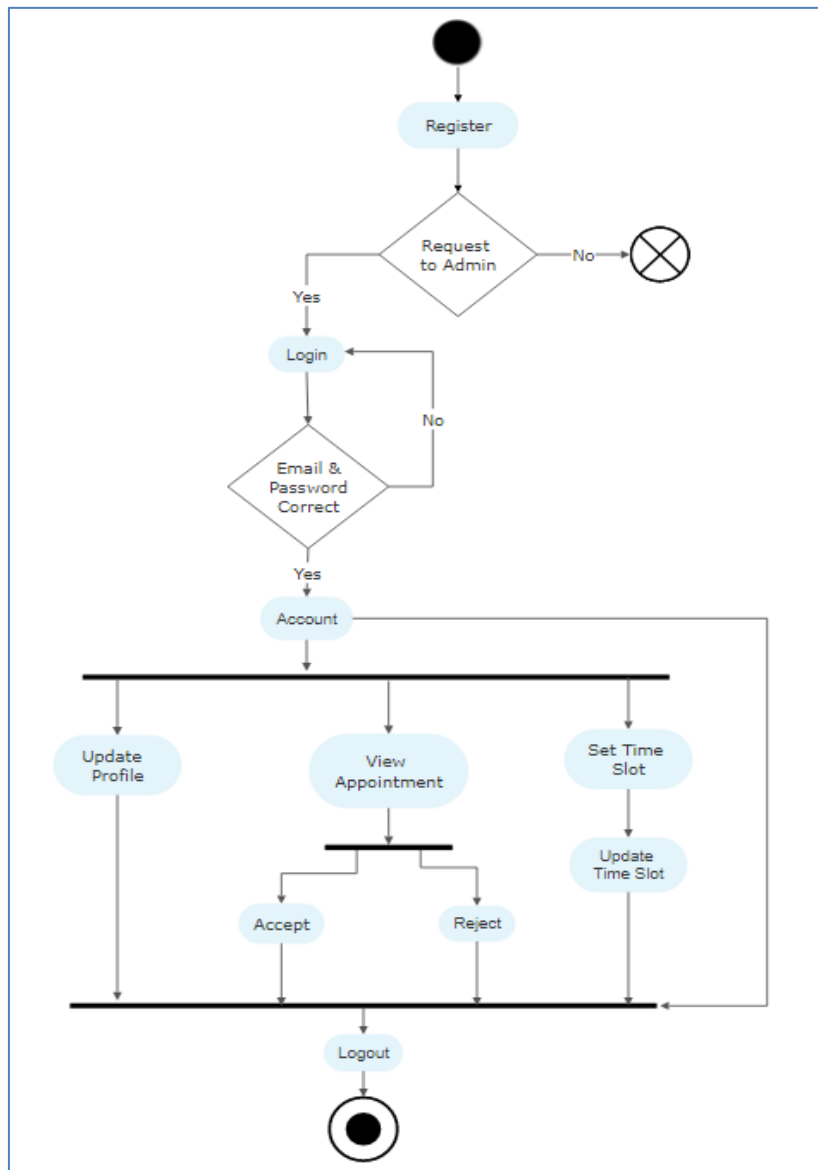
[Fig: 3.1.3- Use case Diagram- Admin]

3.2 Activity Diagrams

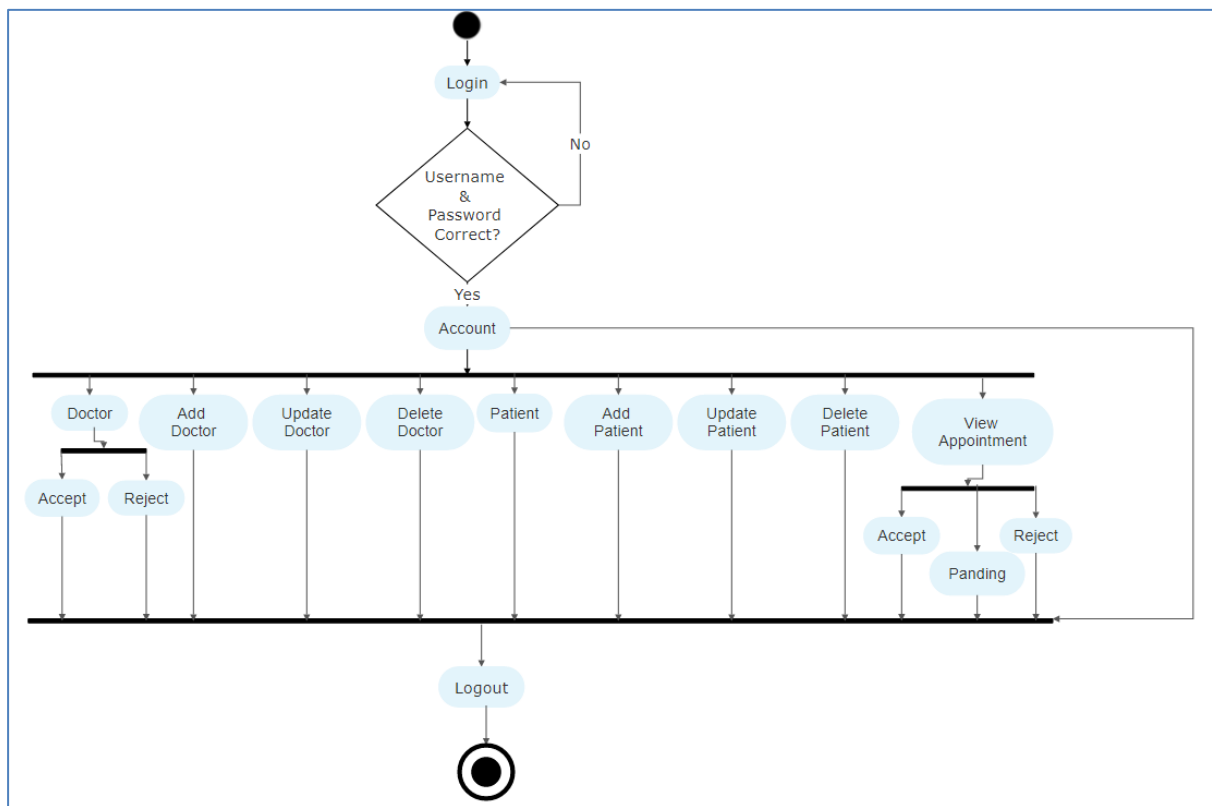
Activity diagram is used to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.



[Fig: 3.2.1- Activity diagram of patient]



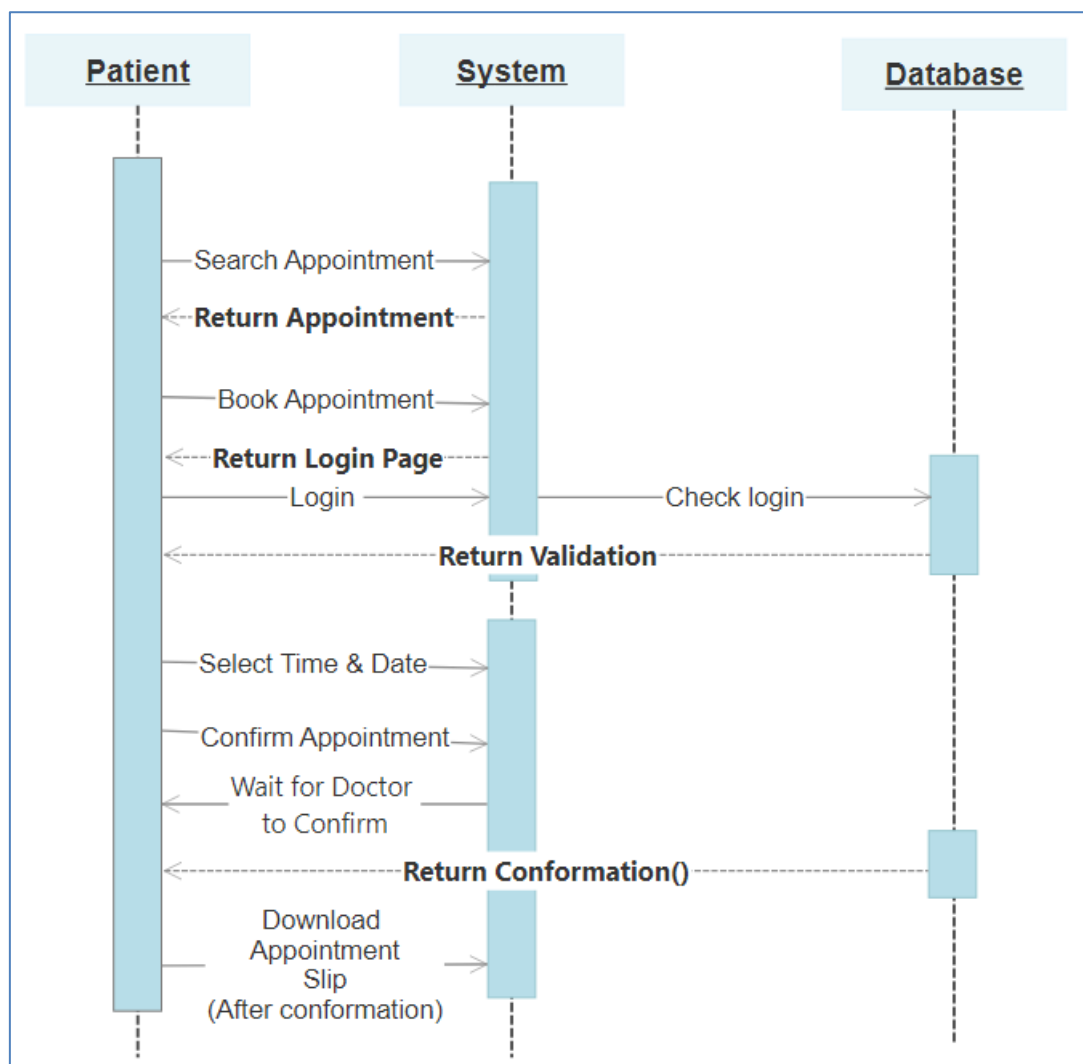
[Fig: 3.2.2- Activity diagram of doctor.]



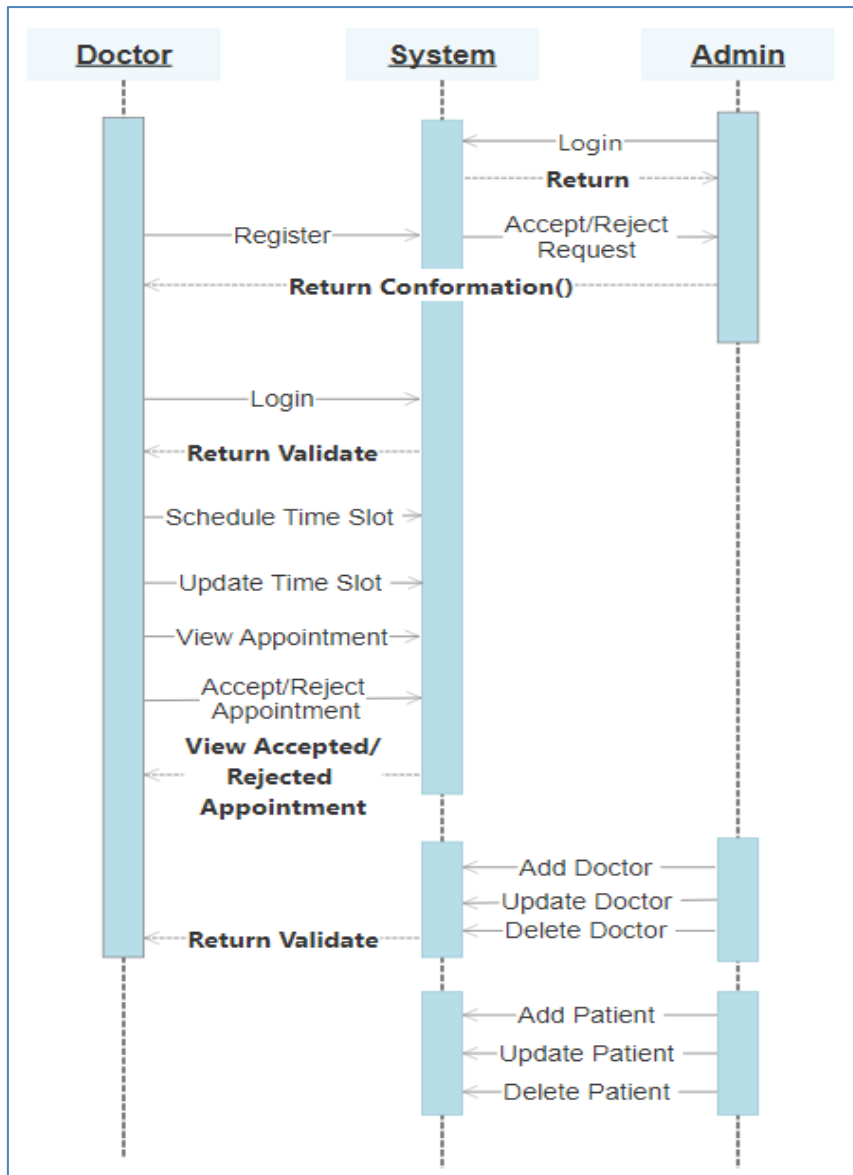
[Fig: 3.2.3- Activity diagram of admin.]

3.3 Sequence Diagrams

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.



[Fig: 3.3.1- Sequence diagram of patient.]

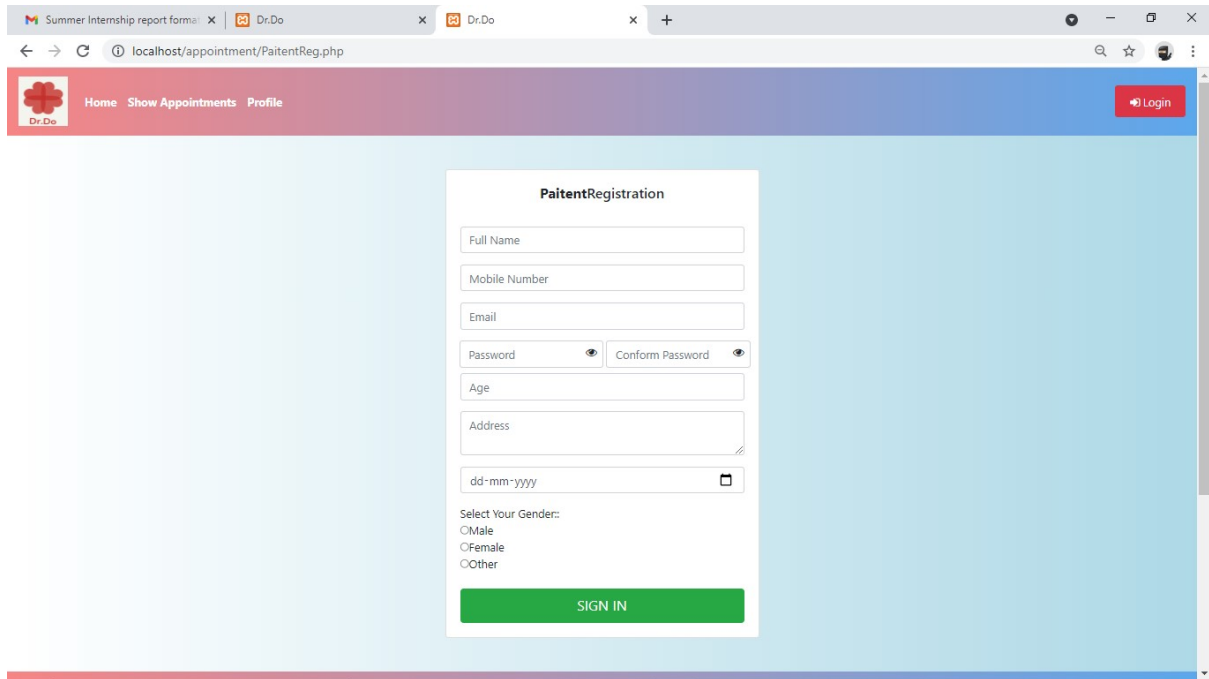


[Fig: 3.3.2 - Sequence diagram of doctor and admin.]

Chapter 4 – Implementation

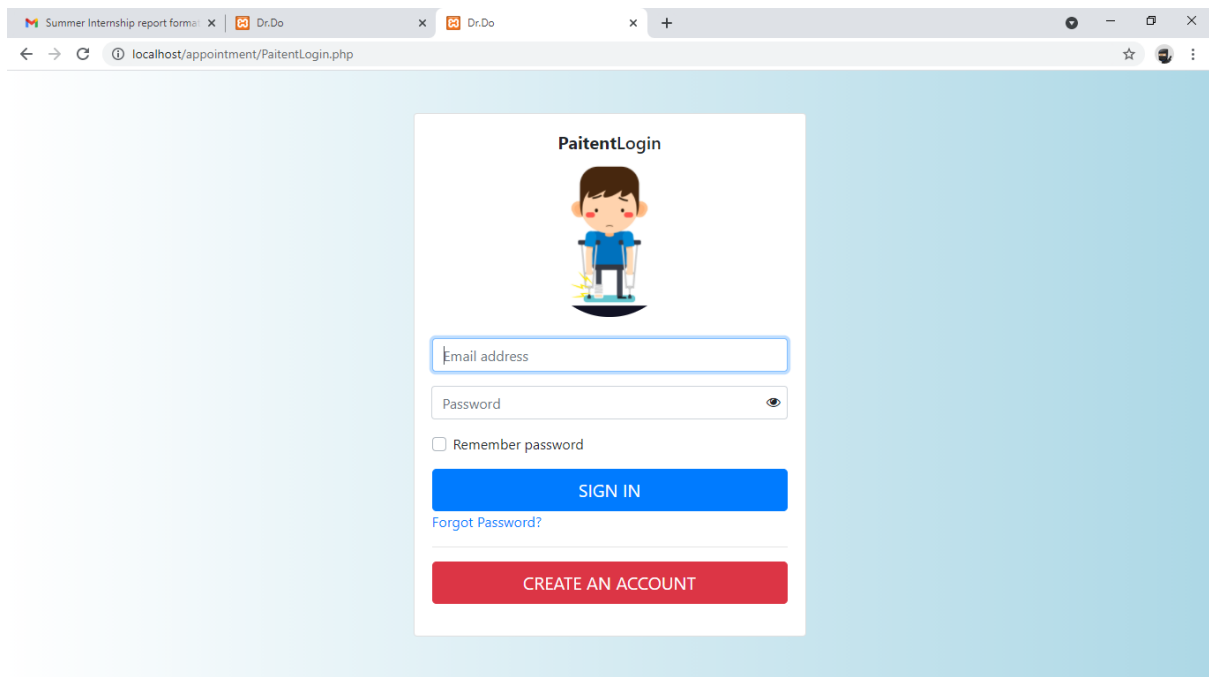
4.1. Snapshots of project

- **Patient side snapshots**



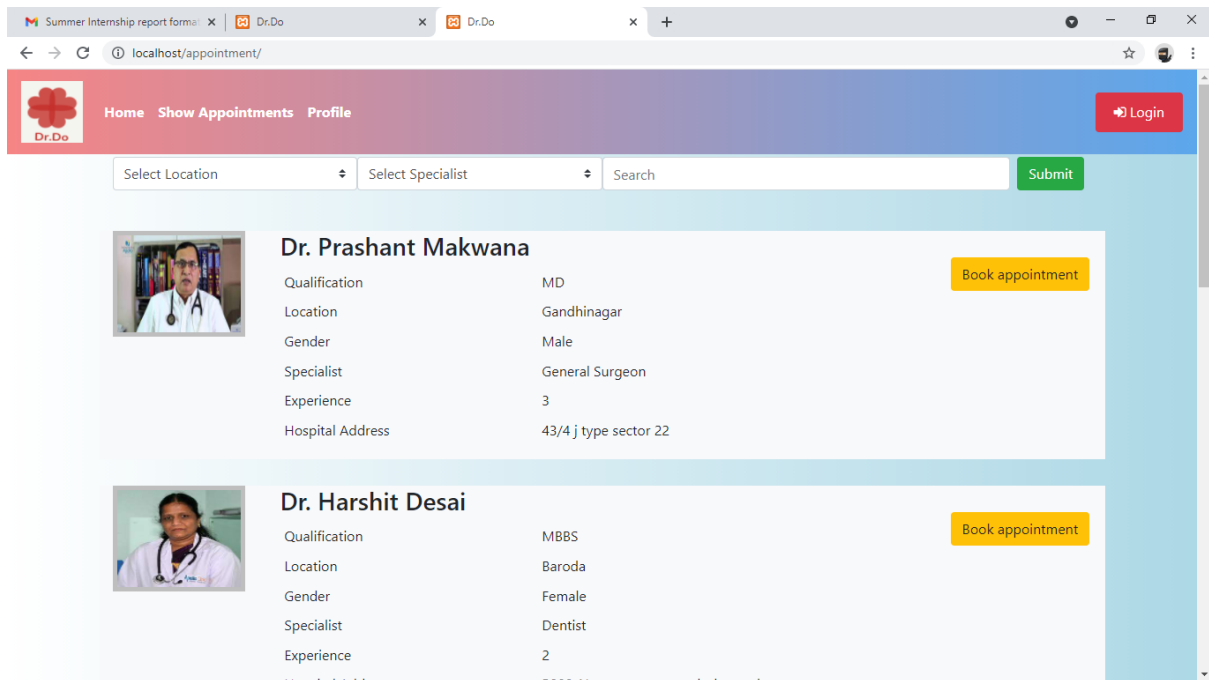
The screenshot shows a web browser window with the URL `localhost/appointment/PatientReg.php`. The page features a header with a red cross logo, navigation links (Home, Show Appointments, Profile), and a red 'Login' button. The main content area is a light blue gradient. Centered on the page is a white 'PatientRegistration' form. The form includes input fields for Full Name, Mobile Number, Email, Password, and Confirm Password (with a toggle icon), Age, and Address. Below the address field is a date picker set to 'dd-mm-yyyy'. At the bottom of the form are radio buttons for 'Select Your Gender:' with options Male, Female, and Other. A green 'SIGN IN' button is positioned at the bottom of the form.

[Fig: 4.1.1- Patient registration page.]

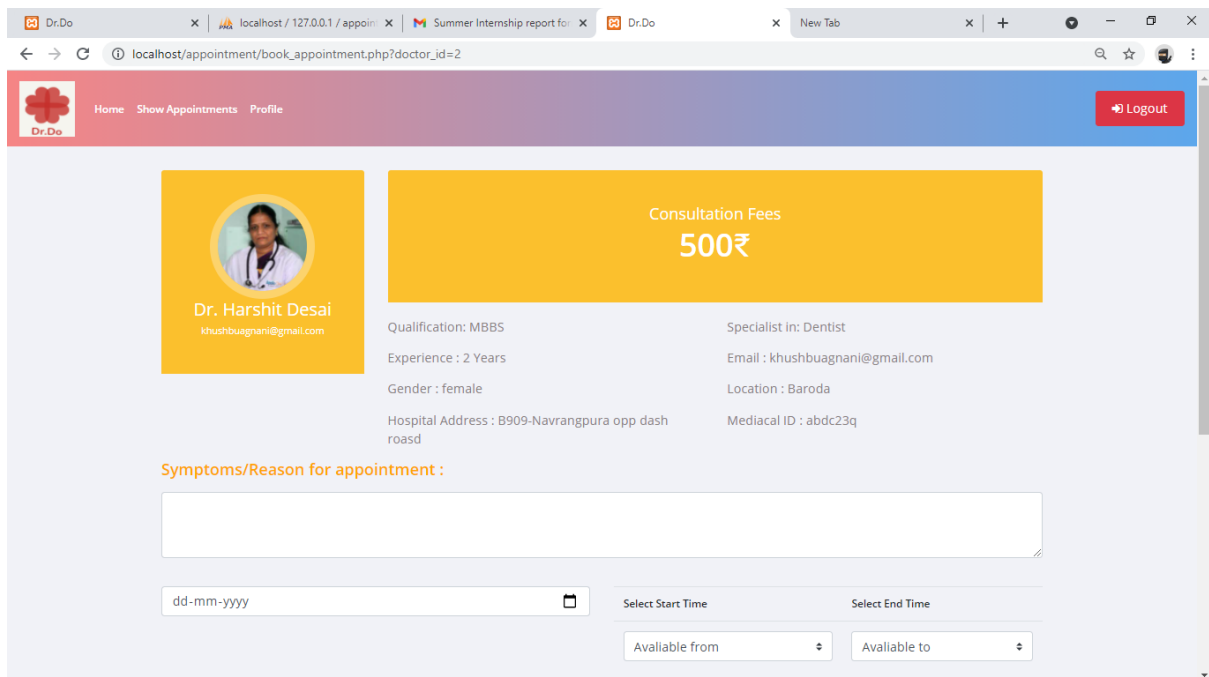


The screenshot shows a web browser window with the URL `localhost/appointment/PatientLogin.php`. The page features a header with a red cross logo, navigation links (Home, Show Appointments, Profile), and a red 'Login' button. The main content area is a light blue gradient. Centered on the page is a white 'PaitentLogin' form. At the top of the form is a cartoon illustration of a boy with brown hair, wearing a blue shirt and white pants, standing on a black oval base. Below the illustration are input fields for 'Email address' and 'Password' (with a toggle icon). There is a checkbox labeled 'Remember password'. A blue 'SIGN IN' button is positioned below the checkbox. Below the button is a blue link that says 'Forgot Password?'. At the bottom of the form is a red 'CREATE AN ACCOUNT' button.

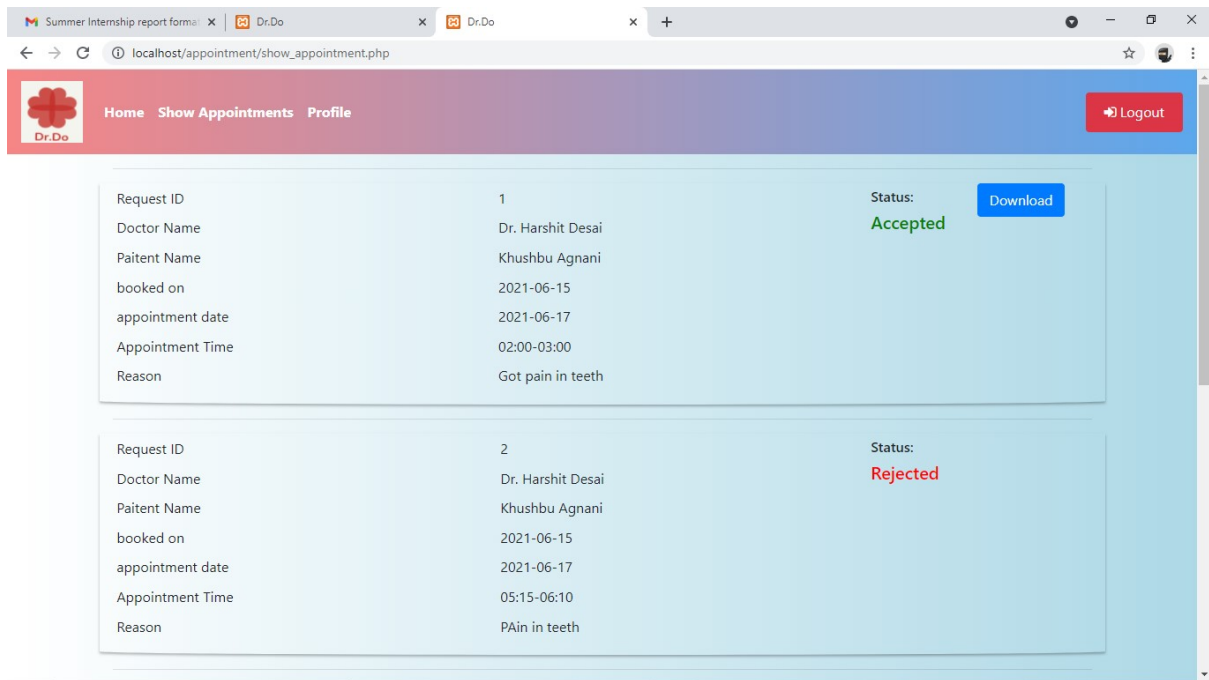
[Fig: 4.1.2- Patient login page.]



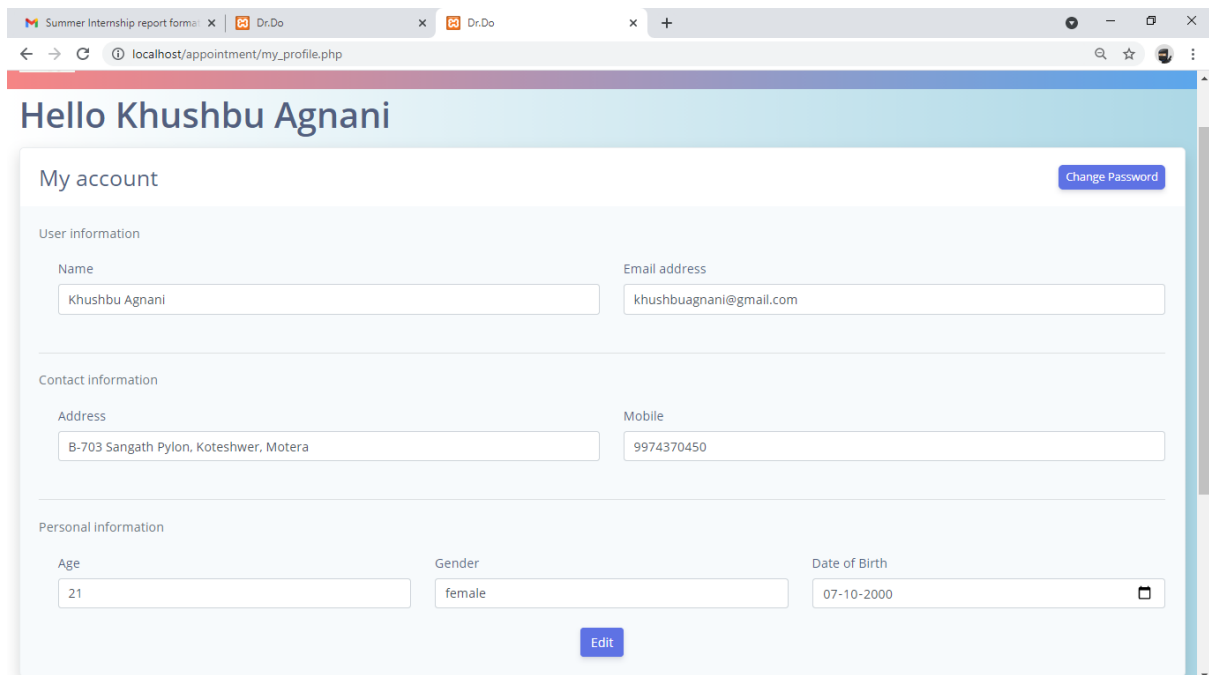
[Fig: 4.1.3 - Patient index page.]



[Fig: 4.1.4 - View doctors details.]



[Fig: 4.1.5- Display booked appointments.]



[Fig: 4.1.6- Patients view/edit profile page.]

- **Doctor side snapshots:**

DoctorRegistration

Full Name

Mobile Number

Email

Password Confirm Password

Select Your Gender:

☐ Male

☐ Female

☐ Other

Select DOB

dd-mm-yyyy

Specialist in

General Physicians

Experience

City

Hospital/Clinic Address

Medical ID

Fees


Qualification

Choose File No file chosen

SIGN IN

[Fig: 4.1.7- Doctor registration page.]

DoctorLogin



Email address

Email address

Password

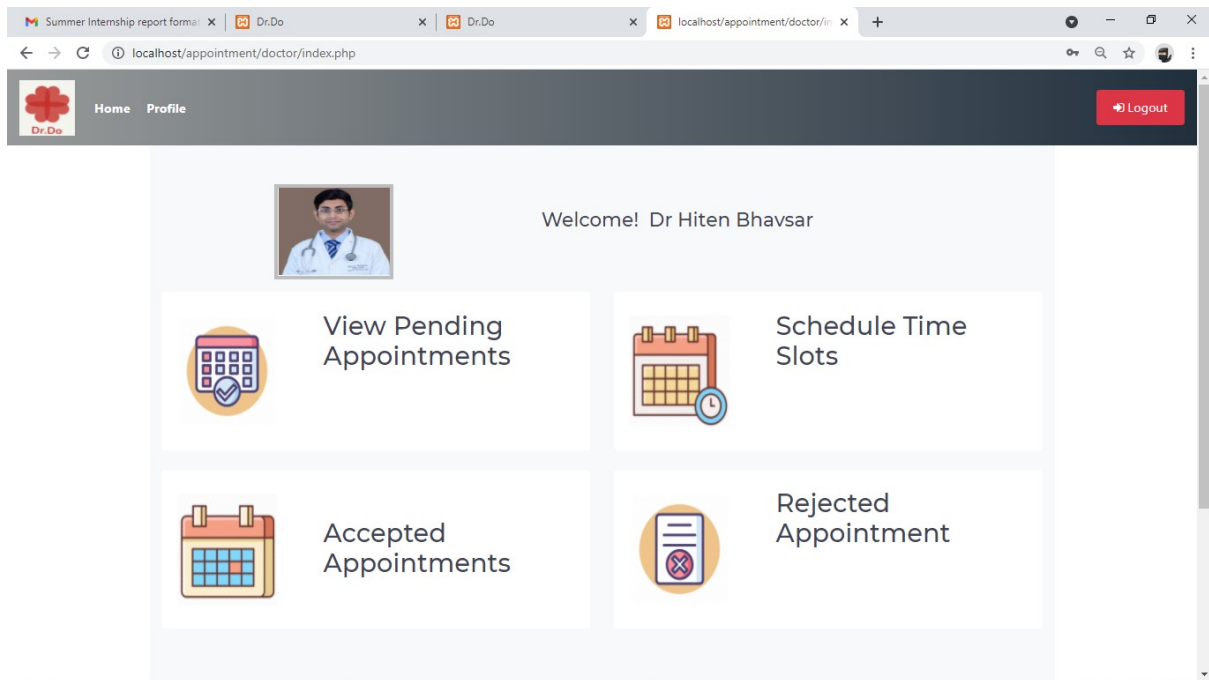
Remember password ☐

SIGN IN

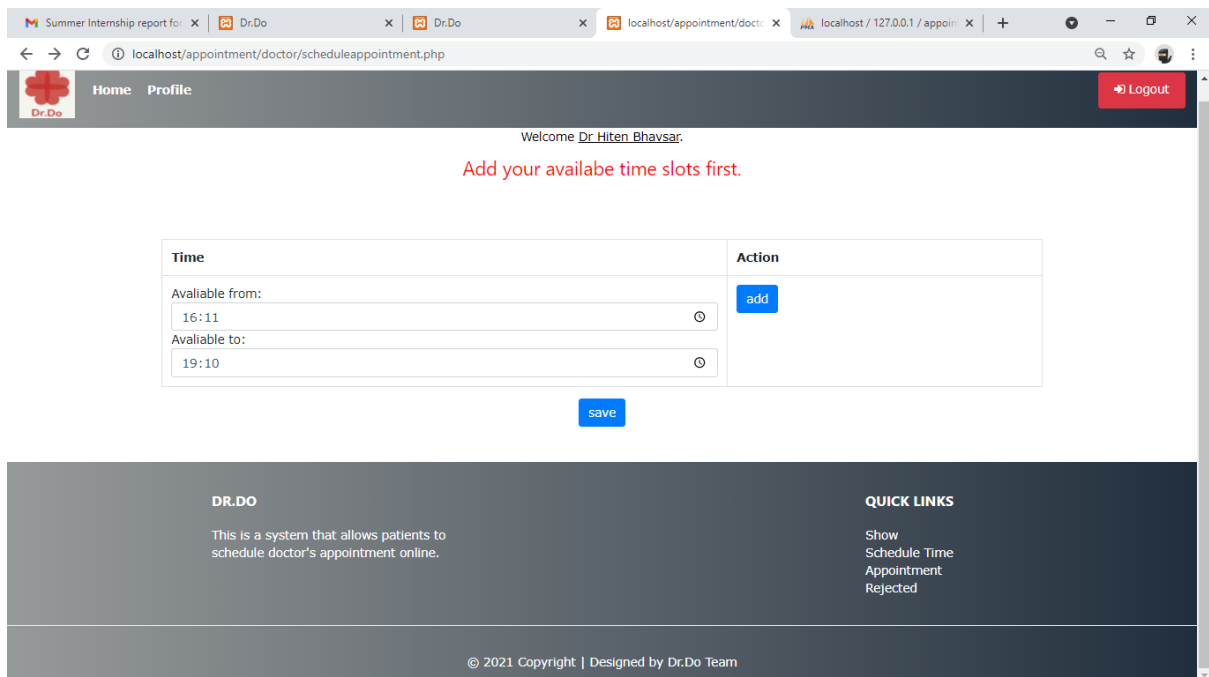
[Forgot Password?](#)

[Create an account](#)

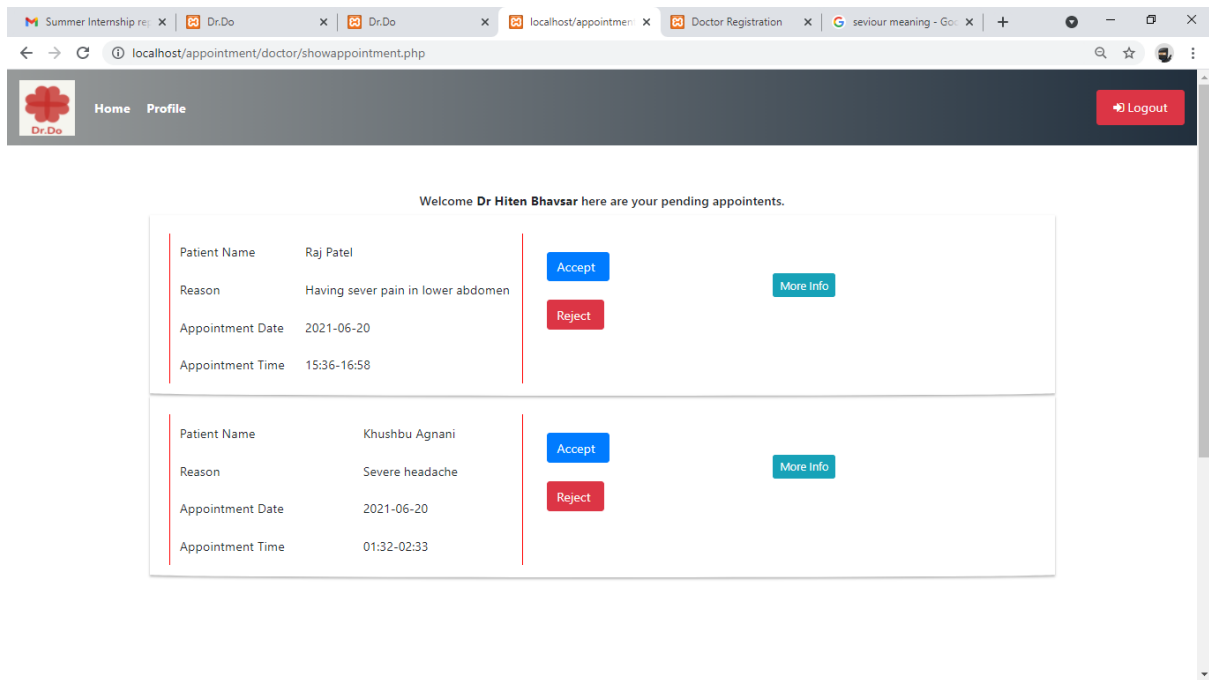
[Fig: 4.1.8- Doctor login page.]



[Fig: 4.1.9- Doctor index page.]

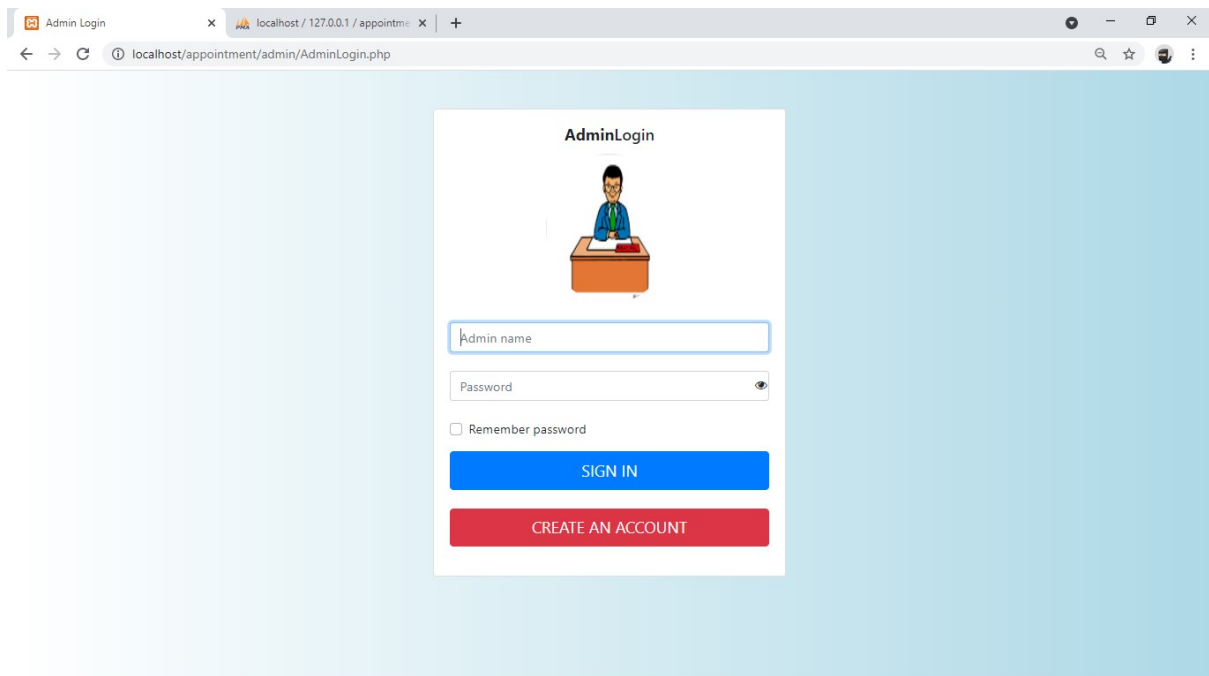


[Fig: 4.1.10-Schedule time slot.]

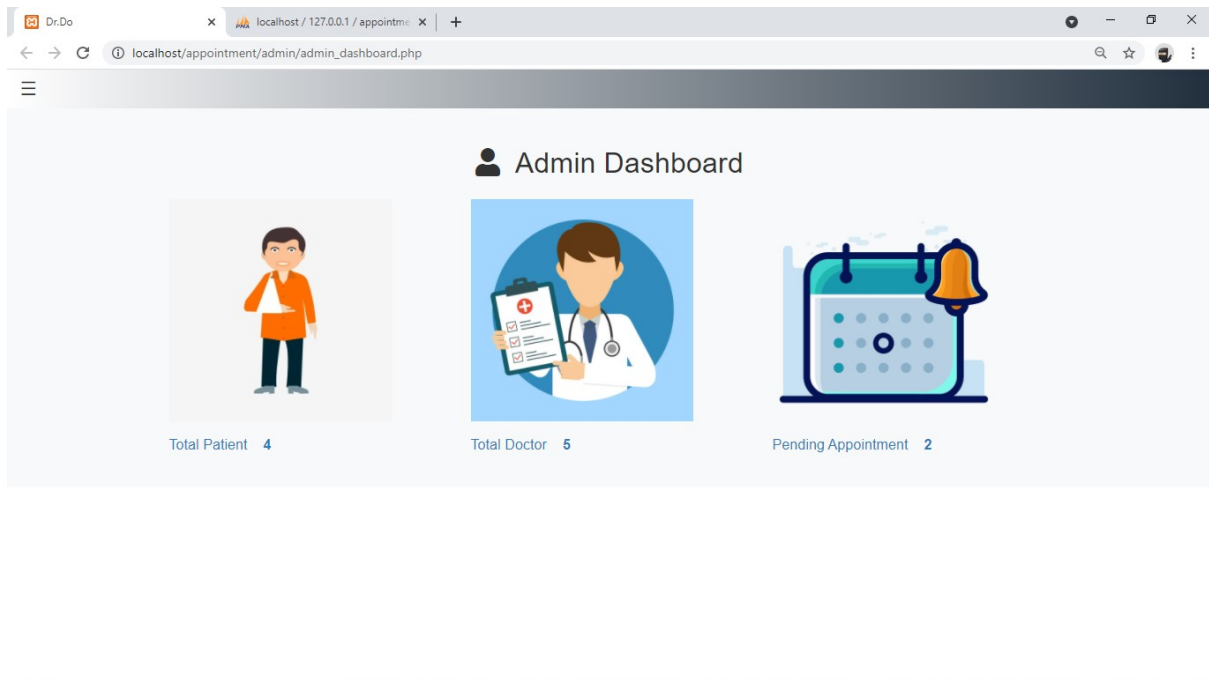


[Fig: 4.1.11- Doctor pending appointments.]

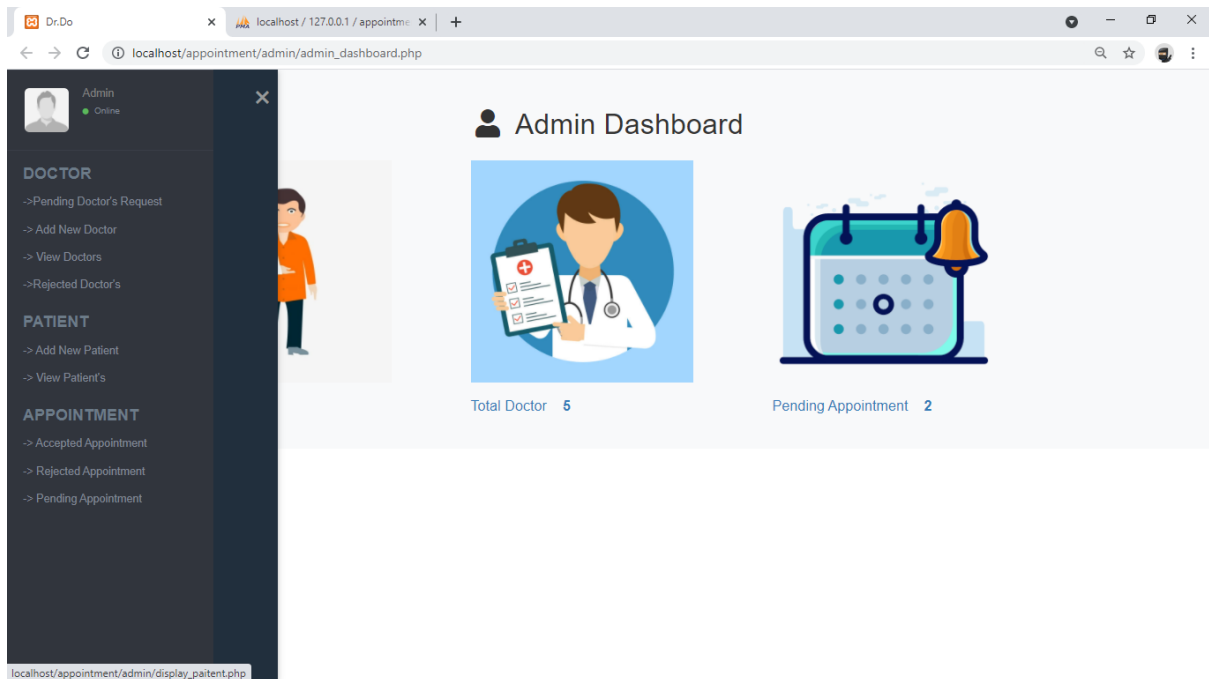
- Admin side snapshots:**



[Fig: 4.1.12- Admin Login]



[Fig: 4.1.13- Admin index page.]



[Fig: 4.1.14- Side navigation bar.]

View Doctors

Name	Gender	Dob	Category	Experience in Years	City	Fees	Qualification	View More	Update	Delete
Dr. Prashant Makwana	male	2001-09-05	General Surgeon	3	Gandhinagar	350	MD			
Dr. Harshit Desai	female	2000-07-10	Dentist	2	Baroda	500	MBBS			
Dr. Hiten Bhavsar	male	1995-08-03	General Surgeon	3	Ahmedabad	600	PHD			
Dr. Raman Patel	male	1998-08-28	ENT Specialist	4	Ahmedabad	1500	PHD			
Dr. Manju Pawar	female	1995-10-07	Gynecologist	5	Ahmedabad	750	MBBS			

[Fig: 4.1.15- Doctor Management page.]

More Doctor Details

ID	8
Name	Dr. Parul Patel
Mobile	9898444629
Email	16ce001@gmail.com
Password	Parth123@
Gender	female
Date Of Birth	1997-01-29
Category	General Surgeon
Experience	4
City	Baroda
Hospital Address	XYZ-007 Nobal nagar. opp street city mall.
Medical ID	1211
Fees	350
Qualification	PHD
Profile photo	

[Fig: 4.1.16- Pending doctor requests.]

4.2. Data dictionary

Field Name	Data type	Field Length	Constraint	Description
id	Int	10	Primary Key	Appointment id
form_patient_id	int	10	Not Null	Id for patient
to_doctor_id	int	10	Not Null	Id for doctor
booked_on	date	-	Not Null	Appointment booking date
appointment_date	date	-	Not Null	Appointment date
appointment_start	varchar	200	Not Null	Appointment start
appointment_end	varchar	200	Not Null	Appointment end
Reason	varchar	300	Not Null	Patient appointment reason
Status	varchar	100	Not Null	Accept or reject appointment status

[Table-4.1: book_appointment]

Field Name	Data type	Field Length	Constraint	Description
Id	int	11	Primary Key	Id for patient
Name	varchar	200	Not Null	Patient name
Mobile	big int	30	Not Null	Patient contact detail
Email	varchar	60	Not Null	Patient mail id
Password	varchar	25	Not Null	Login password for

				patient
Age	int	10	Not Null	Patient age
Address	varchar	160	Not Null	Patient living address
Gender	varchar	15	Not Null	Gender male/female
DOB	date	-	Not Null	Date of birth

[Table-4.2: patient]

Field Name	Data type	Field Length	Constraint	Description
Id	int	20	Primary Key	Unique Id for doctor
Name	varchar	150	Not Null	Doctor name
Mobile	big int	15	Not Null	Doctor contact detail
Email	varchar	150	Not Null	Doctor mail id
Password	varchar	20	Not Null	Login password for doctor
Gender	varchar	15	Not Null	Gender male/female
Dob	date	-	Not Null	Date of birth
Category	varchar	30	Not Null	Speciality of doctor
Experience	int	10	Not Null	Doctor medical work experience
City	varchar	100	Not Null	Location of doctor

hospital address	varchar	170	Not Null	Address of hospital in which doctor work
medical id	Varchar	40	Not Null	Doctor medical id/ license id
Fess	Int	20	Not Null	Doctor consultation fees
qualification	Varchar	200	Not Null	Doctor's qualification
Photo	Varchar	200	Not Null	Doctor photo
Status	Varchar	100	Not Null	Accept or reject appointment status

[Table-4.3: doctor]

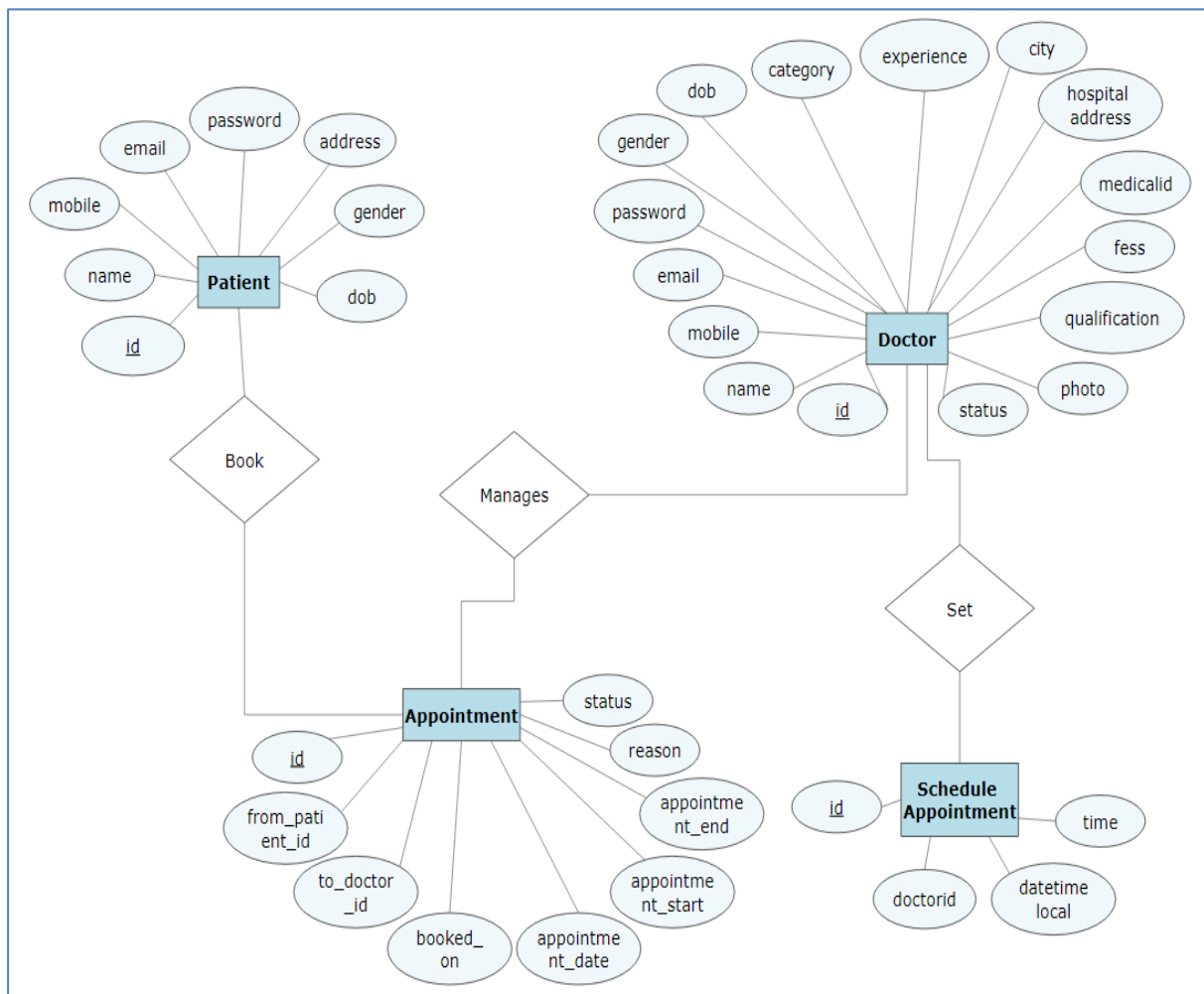
Field Name	Data type	Field Length	Constraint	Description
Id	Int	10	Primary Key	Unique ID
Doctore_id	Int	20	Not Null	Registered doctor id
datetimelocal	Varchar	300	Not Null	Start time of slots
Time	Varchar	300	Not Null	End time of slots

[Table-4.4: schedule appointment]

Field Name	Data type	Field Length	Constraint	Description
admin_name	Varchar	200	Not Null	Admin name
admin_pwd	Varchar	200	Not Null	Admin password

[Table-4.5: admin]

4.3. Database Relationship Diagram



[Fig:4.3.1- Database Relationship Diagram(ER-Diagram)]

Conclusion:

Doctor appointment booking system is a very exciting topic to work. After going through the work, I faced many challenging tasks. Day by day healthcare system is becoming important part of our society. So I decided to build this system. This is a web based software application and it will be easily accessible from any computer/laptop by using internet. I tried to complete the software within time limit. And almost I can do it. The proposed system will easily be accessible and it will be well organized and delivered the right information in the right place.

Future scope:

- Implement Security system for project.
- Allow patient to make payment online.
- Doctors should be able to
- Add patient - doctor chatting option.
- Notify if doctor forgets to accept/reject appointment before a day or two.
- Notify patient to reschedule appointment if rejected by doctor or few days prior to next appointment's due date.

References:

1. Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
2. PHP & MySQL Novice to Ninja – by Kevin Yank
3. HTML & CSS: The Complete Reference, Fifth Edition – by Thomas A. Powell
4. <https://www.w3schools.com/php/DEFAULT.asp>
5. <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
6. <https://www.tutorialspoint.com/javascript/index.html>