|  |  |
| --- | --- |
| **Table of Contents** | **Page No** |
| 1. Title of the Project | **1** |
| 2. Introduction and Objectives of the Project | **1** |
| 3. Project Category (RDBMS/OOPS/Networking/Multimedia/Artificial Intelligence/Expert Systems etc.) | **1** |
| 4. Tools/Platform, Hardware and Software Requirement specifications | **2** |
| 5. Goals of Implementation | **3** |
| 6. SDLC Process Applied | **4** |
| 7. Data Model | **5** |
| 8. Functional Requirements (Use Case Diagram) | **5** |
| 9. Nonfunctional Requirements | **6** |
| 10. Feasibility Study | **10** |
| 11. Project Planning | **11** |
| 12. Project Scheduling | **13** |
| 1. Software Engineering Paradigm applied   a. Data Flow Diagram (DFD) | **14** |
| 14. Database design | **15** |
| 15. User Interface Design | **16** |
| 16. Coding | **18** |
| 17. Testing | **27** |
| 18. System Security measures (Implementation of security for the project developed) | **36** |
| 19. Database/Data security | **41** |
| 20. Creation of User profiles and access rights | **41** |
| 21. Cost Estimation of the Project along with Cost Estimation Model | **41** |
| 22. Reports (sample layouts should be placed) | **41** |
| 23. Future scope and further enhancement of the Project | **42** |
| 24. Bibliography |  |

### Title of the Project

Online Doctors Appointment.

1. **Introduction and Objectives of the Project**

Have you ever had one of those moments when you had to make an appointment real fast with your provider but, couldn’t because of simple reasons, like the secretary not being available or the phone line being busy, or the secretary answers your call and put you on hold forever? How about going through the infamous

interactive voice response hoping that one of the ten options will be just the one you want? This certainly has happened to most of us. These moments are extremely annoying and waste your precious time. But you know that you have to make that appointment because of its extreme importance. So what would you do? Waste your precious time and money to try to make all those calls? Or would you look up for an easier and far less time consuming way? The answer is in the human nature that all of us always look for the easier way. The easier solution, in this case, is online appointment scheduling. To make an online appointment is far easier than any other ways. Most of us have internet connections at our homes, office, or on a mobile

device we carry… Why should we not use this immense facility that we have at our disposal? We can easily make appointments through the internet using appointment scheduling software this could, in turn, make our life a lot easier and faster. There are a lot of reasons why making an appointment online is a lot easier.

### 3.A patients should be able to

* register and login to the system.
* view doctors list according to department as well as doctors’ specifications.
* Fix appointment and cancel their appointment.
* Once the appointment is fixed the system notifies the patient confirm the appointment.

**Admin should be able to**

* Manage the doctors.
* Manage the doctors schedule.
* And confirm appointment.

1. **Project Category**

Web Application

1. **Tools/Platform, Hardware and Software Requirement specifications. Tools**
   1. Visual Studio Code , Microsoft Office

##### Platform

1.Microsoft Windows 7/8/10

**Hardware Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **Browser** | Any standard browser with Javascript interpreter | **Software** | Apache |
| **Client side mark up / scripting languages** | HTML, JAVASCRIPT,  CSS, PHP | **Database Management System Software** | MySQL |
|  |  | **Specification** | MySQL 4.1 |

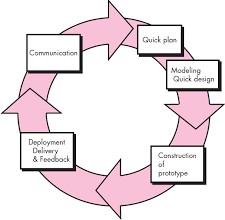
**Software Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **HDD** | 200 MB | **HDD** | 320 GB |
| **Processor** | Pentium 4 or newer processor that supports SSE2 | **Processor** | Dual Core or newer processor |
| **Memory** | 512 MB | **Memory** | 2 GB |

1. **Goals of Implementation**

The implementation aims at seamless document sharing across the institution.

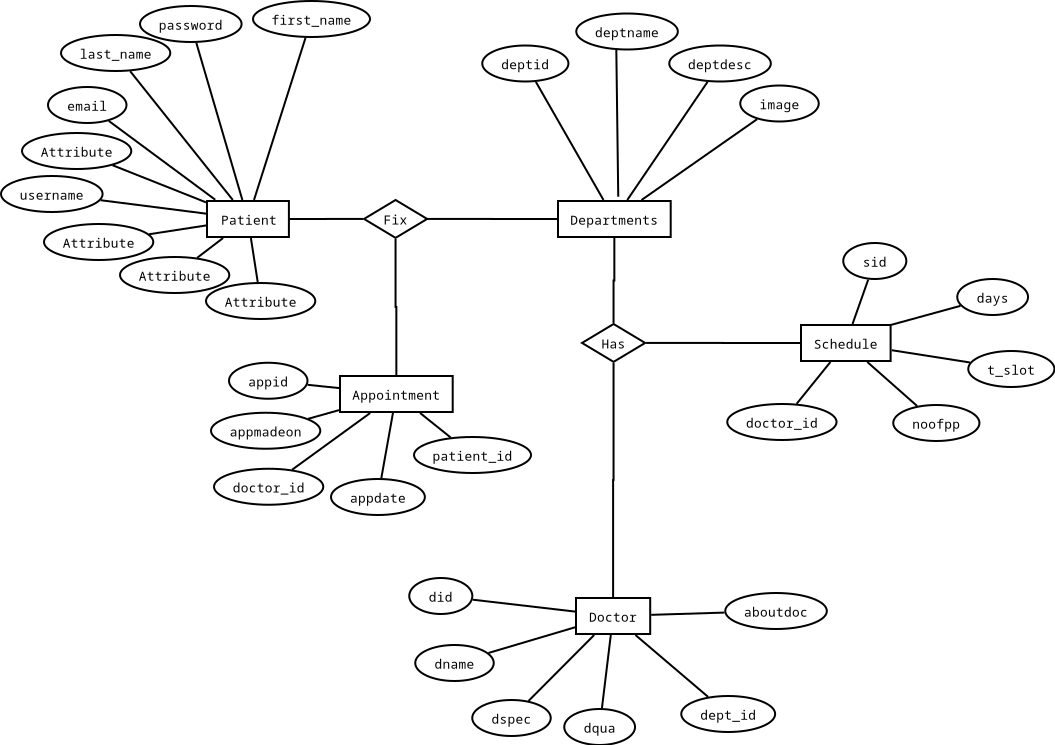
1. **SDLC Process Applied**

****

Often, a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.

The prototyping paradigm begins with **requirements gathering**. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A "**quick design**" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

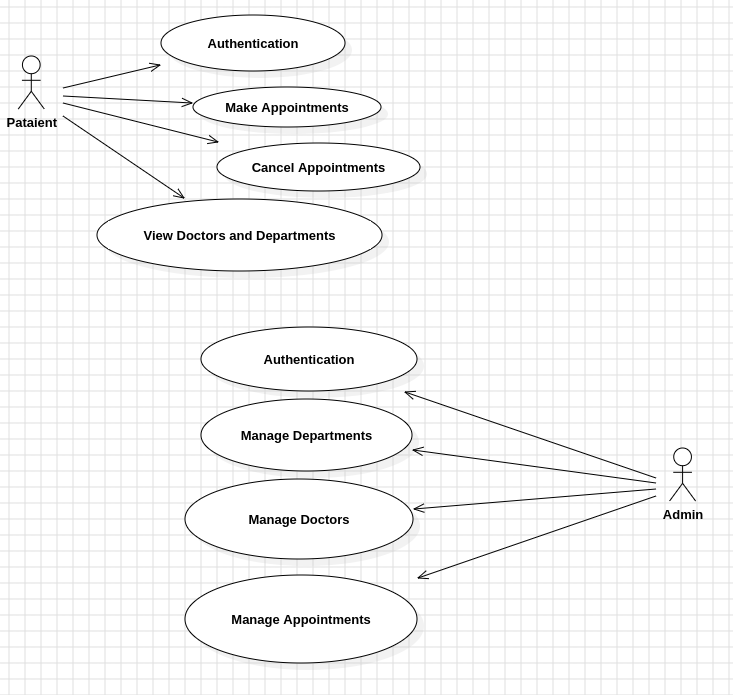
1. **Data Model**

****

1. **Functional Requirements**

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

**Use Case Diagram**

****

**Use Case Descriptions**

|  |  |
| --- | --- |
| **Use Case Name:** | Authentication |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Patient is connected to the Internet and on the Online Doctors Appointment home page |

|  |  |
| --- | --- |
| **Basic Path** | 1. Patient enters username and password. 2. The username and password is matched with the record in the database. 3. If the authentication parameters are correct the patient is directed to the patient’s main page, otherwise an error message is displayed |
| **Alternate Path** | NA |
| **Post Condition** | The user is on the Patient Home Page |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Make Appoinntment |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | User is connected to the Internet and on the user’s main page |
| **Basic Path** | 1. User selects a doctor 2. User clicks on the Fix Appointment button. 3. The server side program receives the doctors details and saves it in the   server system’s. |
| **Alternate Path** | NA |
| **Post Condition** | The patient fix appointment. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Cancel Appointment |
| Priority | Essential |

|  |  |
| --- | --- |
| **Trigger** | Menu selection |
| **Precondition** | Patient is connected to the Internet and on the patient’s main page |
| **Basic Path** | 1. Patient Select All Appointment. 2. The server side program returns the list of Fix appointment details. |
| **Alternate Path** | NA |
| **Post Condition** | The patient has the list of all Appointment and Cancel  Appointment. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Manage Appointment |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Admin is connected to the Internet and on the admin’s main page |
| **Basic Path** | 1. The server program returns list of submitted non confirm appointment. 2. Admin selects the appointment he/she wishes to confirm. 3. The confirm acknowledgment goes to the patient whose appointment is confirm. |
| **Alternate Path** | NA |
| **Post Condition** | The admin confirm a appointment |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Manage Doctor |
| **Priority** | Essential |
| **Trigger** | Menu selectio |
| **Precondition** | Admin is connected to the  Internet and on the admin’s main page |
| **Basic Path** | 1. Admin insert doctors details. 2. Admin clicks on the link. 3. The server side program receives the request and sends   the doctor details. |
| **Alternate Path** | NA |
| **Post Condition** | The Admin views the doctors |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Manage Schedule |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Admin is connected to the Internet and on the admin’s main page |
| **Basic Path** | 4. The server program returns list of all doctors.  .5. Admin selects the doctors he/she wishes to add schedule.  6. The addition acknowledgement goes to the admin of schedule done. |
| **Alternate Path** | NA |
| **Post Condition** | The admin add schedule. |

|  |  |
| --- | --- |
| **Exception Path** | there is a connection failure the server returns to the wait state. |

1. **Non Functional Requirements**

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually DO anything, but are important characteristics nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility. Aren’t a "feature" of the system, but are a required characteristic. You can't write a specific line of code to implement them; rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in English. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making trade offs, e.g., between performance and maintainability. In the APPENDIX you must include an engineering analysis of any significant decisions regarding trade offs between competing attributes

. Here are some examples of non-functional requirements:

### Performance requirements

Requirements about resources required, response time, transaction rates,throughput, benchmark specifications or anything else having to do with performance. For better performance the application will restrict the document size to 5 MB.

#### Operating constraints

List any run-time constraints. This could include system resources,people, needed software, The application must run without any manual intervention

**Platform constraints** Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.Since the application will be developed in PYTHON it is platform independent.

#### Accuracy and Precision

Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

#### Modifiability

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Minimal

**Portability**

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Minimal

**Reliability**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

**Security**

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time, ...) to break into the system.

Do not discuss solutions (e.g. passwords) in a requirements document. Only secured users can access the application.

No one can go to any independent page without logging in.

**Usability**

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics. Legal There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

1. **Feasibility Study**

You should provide a feasibility report in the following format:

* **Product:** A general statement of the product; give a brief description of what the proposed system will do, highlighting where the proposed system meets the specifiedbusiness requirements of the organization.
* **Technical Feasibility:** Will the proposed system perform to the required specification? Outline technical systems options you propose to use, which will give a technical solution satisfying the requirements and constraints of the system, as outlined in the terms of reference.
* **Social Feasibility:** Consideration of whether the proposed system would prove acceptable to the people who would be affected by its introduction. Describe the effect on users from the introduction of the new system; consider whether there will be a need for retraining the workforce. Will there be a need for relocation of some of the workforce? Will some jobs become deskilled? Will the current workforce be able to perform effectively any new tasks introduced by the proposed system? Describe how you propose to ensure user co-operation before changes are introduced.
* **Economic Feasibility:** Consider the cost/benefits of the proposed system. Detail the costs that will be incurred by the organization adopting the new system; consider development costs and running costs. Detail benefits that the new system will bring, direct economic benefits such as reduced costs, and indirect benefits, such as improved management information and better customer service. Illustrate the cost/benefit of the new system by applying a suitable cost/benefit analysis method such as the payback method.
* **Market Research**: A comprehensive market research identifying a need for the product. Detail all market research you carried out, listing sources of information. Justify any conclusions you have drawn from your research. Identify the potential customer base for your product, together with evidence of customer need for the product. Describe how you propose to compete with similar products on the market.
* **Alternative Solution:** Consideration of alternative solutions should be documented. At least two alternative business or technical systems options should be considered. Detail the differences between these options and the proposed system. Justify your choice of the proposed system and the reasons for rejecting the alternative options. At this point, all of the planning for the project has been done and if the feasibility study has shown that the project is likely to succeed within its constraints, then it only remains for us to start the requirements analysis and thus proceed with the project.

|  |  |
| --- | --- |
| **Feasibility Study** | |
| System:  **ONLINE DOCTORS APPOINTMENT** |  |
| Author:  **Ayaan Al Bari , Debalina Bagchi** |  |
| **Product** | |
| The project requires a web application to be developed that will allow online knowledge/document/paper sharing. | |
| **Technical Feasibility** | |
| The web application will be developed using Python and MySQL. The team is competent in that. | |
| **Social Feasibility** | |
| Some training for the users/admin are required but all users are IT literate | |
| **Market Research** | |
| Market research says that this application would be useful for the users as it could seamlessly help them to share documents | |
| **Economic Feasibility** | |
| The application can be developed within budget | |
| **Alternate Solution** | |
| Could be a desktop system but that would not allow documents to be shared online | |

## Project Planning

Project planning is concerned with identifying the following for every project.

* Activities
* Milestones
* Deliverables.

A plan must be drawn up to guide the development towards the project goal. A plan is drawn up at the start of a project. This plan should be used as the driver for the project. The initial plan is not static, and must be modified as the project progresses .Planning is required for development activities from specification through to delivery of the system.

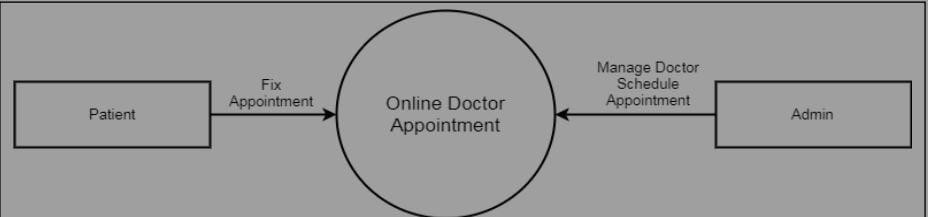
1. **Project Scheduling**

##### GANTT chart

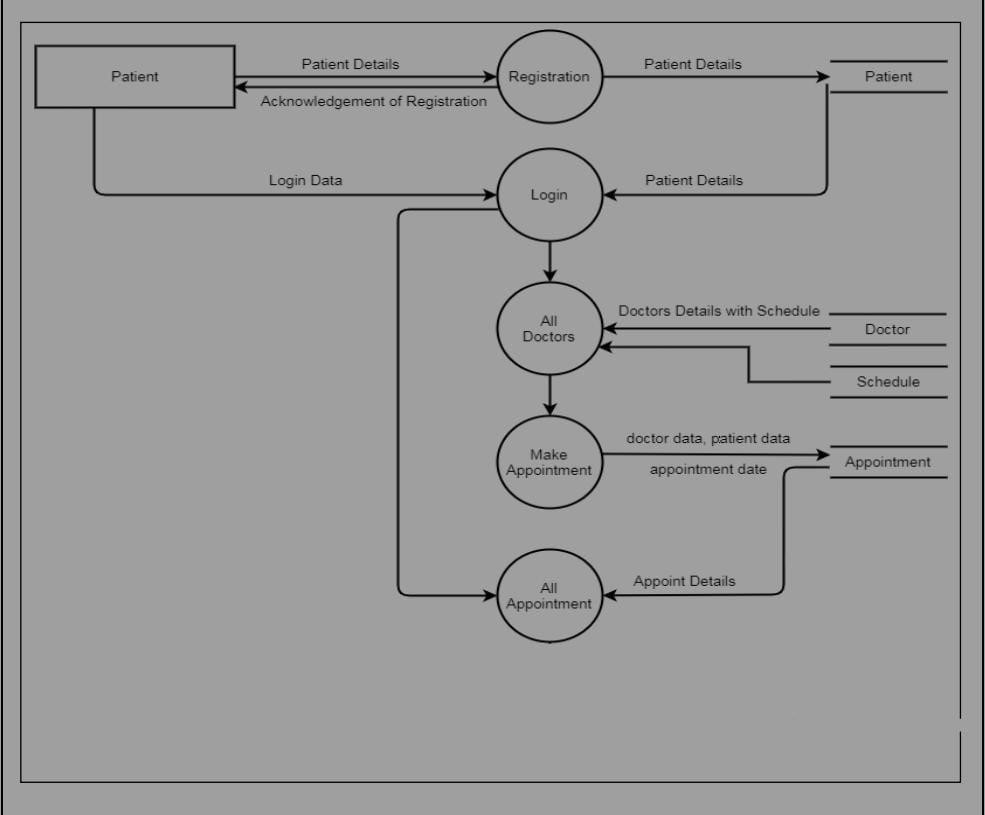
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | erson(s) Responsible | Week 1 | Week 2 | Week 3 | Week 4 |
| Communication |  |  |  |  |  |
| Quick Plan |  |  |  |  |  |
| Modeling Quick Design |  |  |  |  |  |
| Construction of Prototype |  |  |  |  |  |
| Deployment, Delivery and Feedback |  |  |  |  |  |

1. **Software Engineering Paradigm Applied Data Flow Diagrams**

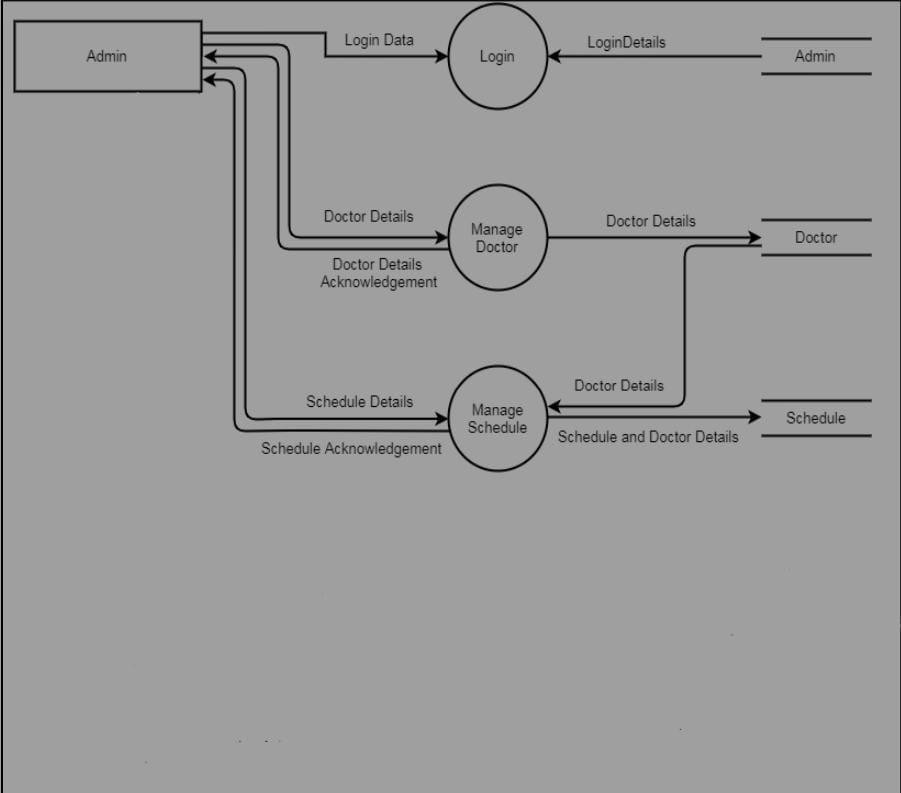
**Level - 0**



**Level -1 patients :**

****

**Level – 1 Admin**

****

1. **Schema/Database Design (project) :-**

**Table Name Appointment:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Type Attribute** | **Data Type** | **Key** | **Description** |
| **app\_id** | **Int(11)** | **Primary Key** | **Appointment ID** |
| **d\_id** | **Int(11)** | **Foreign Key** | **Doctor ID** |
| **u\_id** | **Int(11)** | **Foreign Key** | **Users ID** |
| **app\_date** | **Varchar(12)** |  | **Appointment Fixed date** |
| **app\_made** | **Varchar(12)** |  | **Appointment made**  **date** |
| **time\_slot** | **Varchar(255)** |  | **Appointment time slot** |

**Table Name Doctors:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Type Attribute** | **Data Type** | **Key** | **Description** |
| d\_id | **Int(11)** | **Primary Key** | **Doctors ID** |
| d\_name | **Varchar(255)** |  | **Doctors Name** |
| Degree | **Varchar(255)** |  | **Doctors Degree** |
| Specification | **Varchar(255)** |  | **Doctors Specification** |
| cnumber | **Varchar(12)** |  | **Doctors Mobile** |
| YOE | **Varchar(2)** |  | **Doctors Year Of Experience** |

**Table Name User:-**

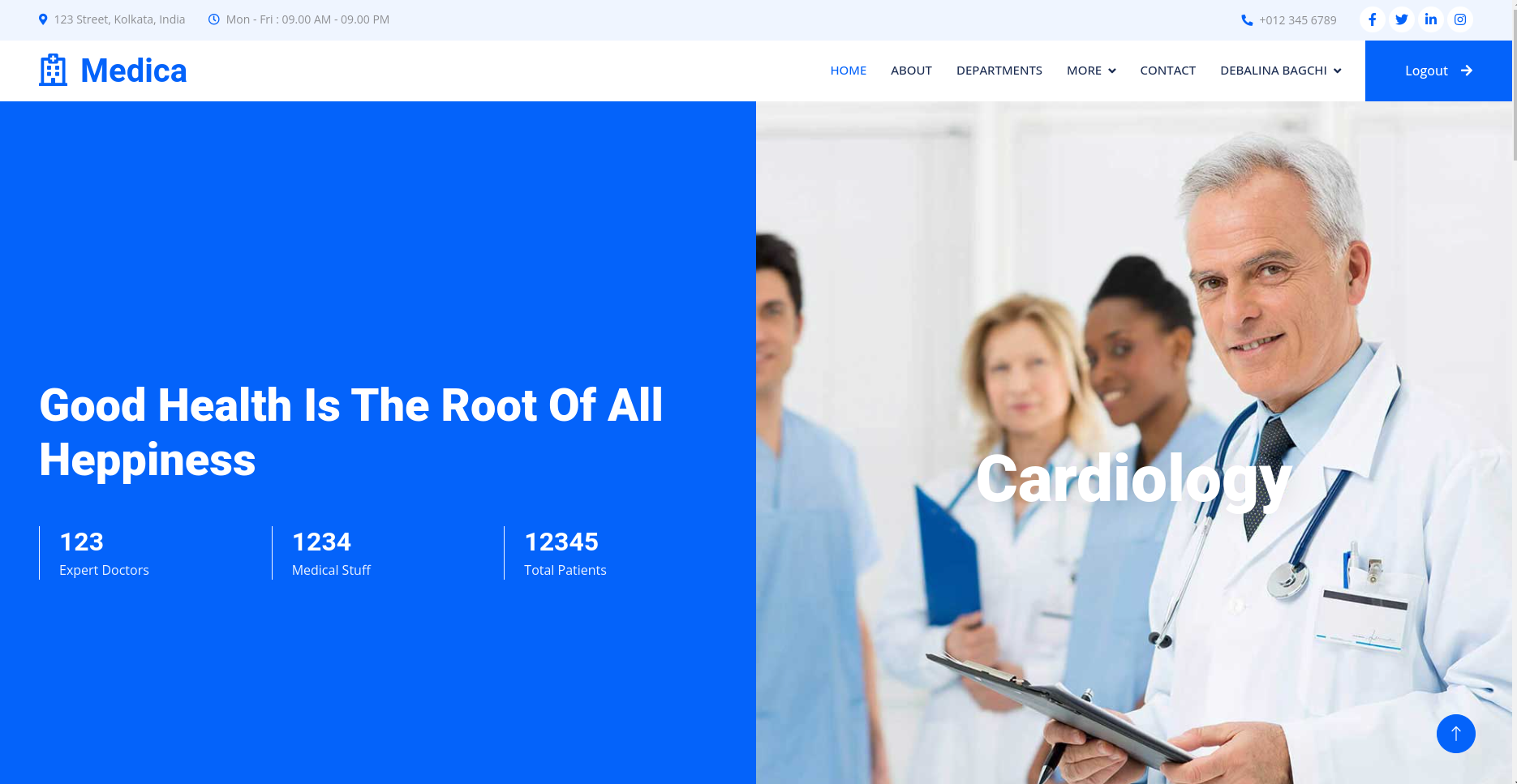
|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Type Attribute** | **Data Type** | **Key** | **Description** |
| u\_id | Int(11) | Primary Key | users ID |
| F\_ame | Varchar(50) |  | Users first Name |
| L\_name | Varchar(50) |  | Users last Name |
| Email\_id | Varchar(255) |  | Users Email |
| Pass | Varchar(50) |  | Users Password |
| Mobile | Varchar(15) |  | Users phn no |

|  |  |  |  |
| --- | --- | --- | --- |
| **Gender** | Varchar(6) |  | Users Gender |
| **Age** | int(3) |  | Users Age |
| **Address** | Varchar(255) |  | Users Address |
| **U\_type** | Enum(‘admin,’patient’) |  | Users type |

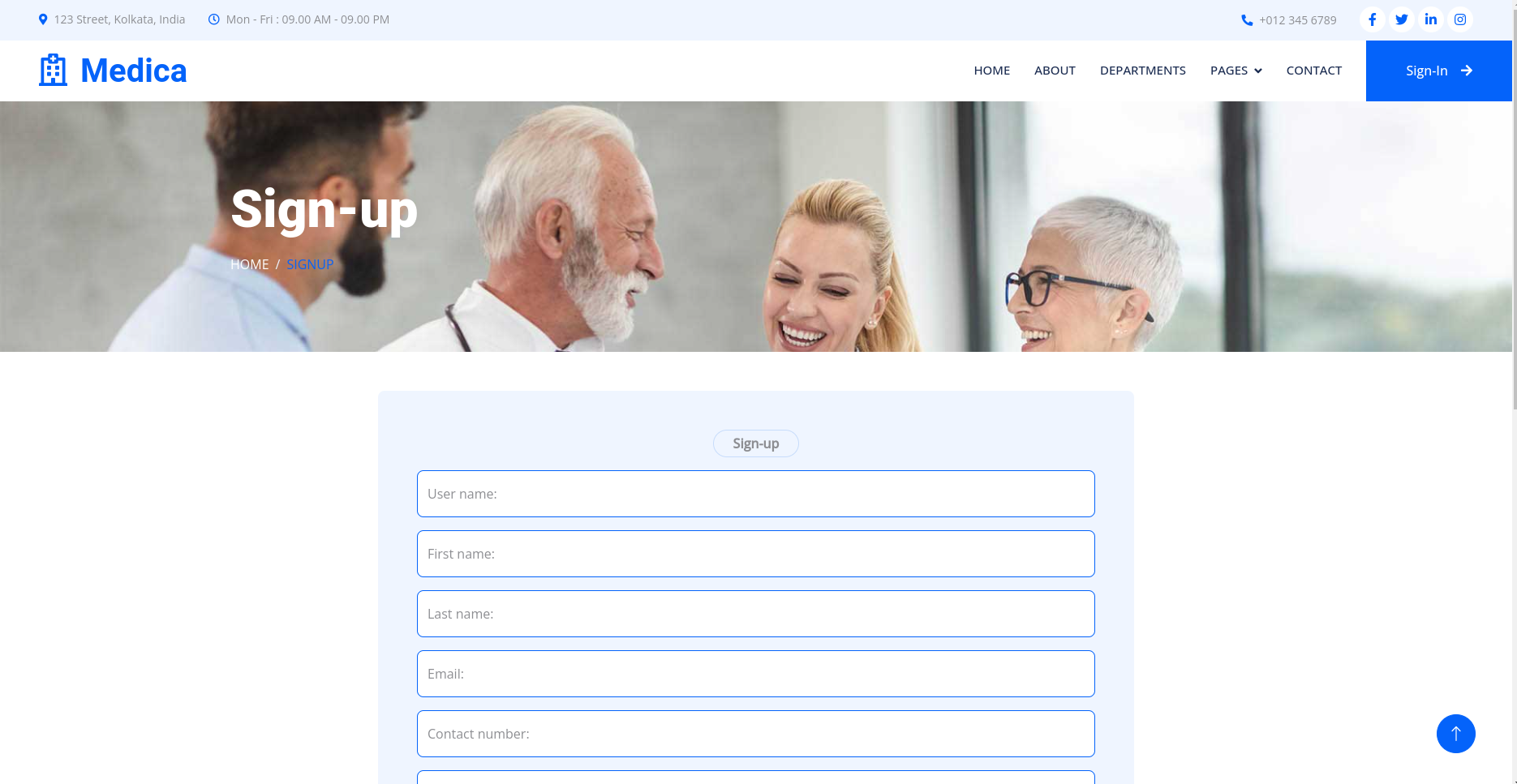
**Table Name Schedule:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Type Attribute** | **Data Type** | **Key** | **Description** |
| s\_id | Int(11) | Primary Key | Schedule id |
| d\_id | Int(11) | Foriegn Key | Doctor id |
| Days | Varchar(255) |  | Schedule day |
| Time\_slot | Varchar(255) |  | Schedule time |

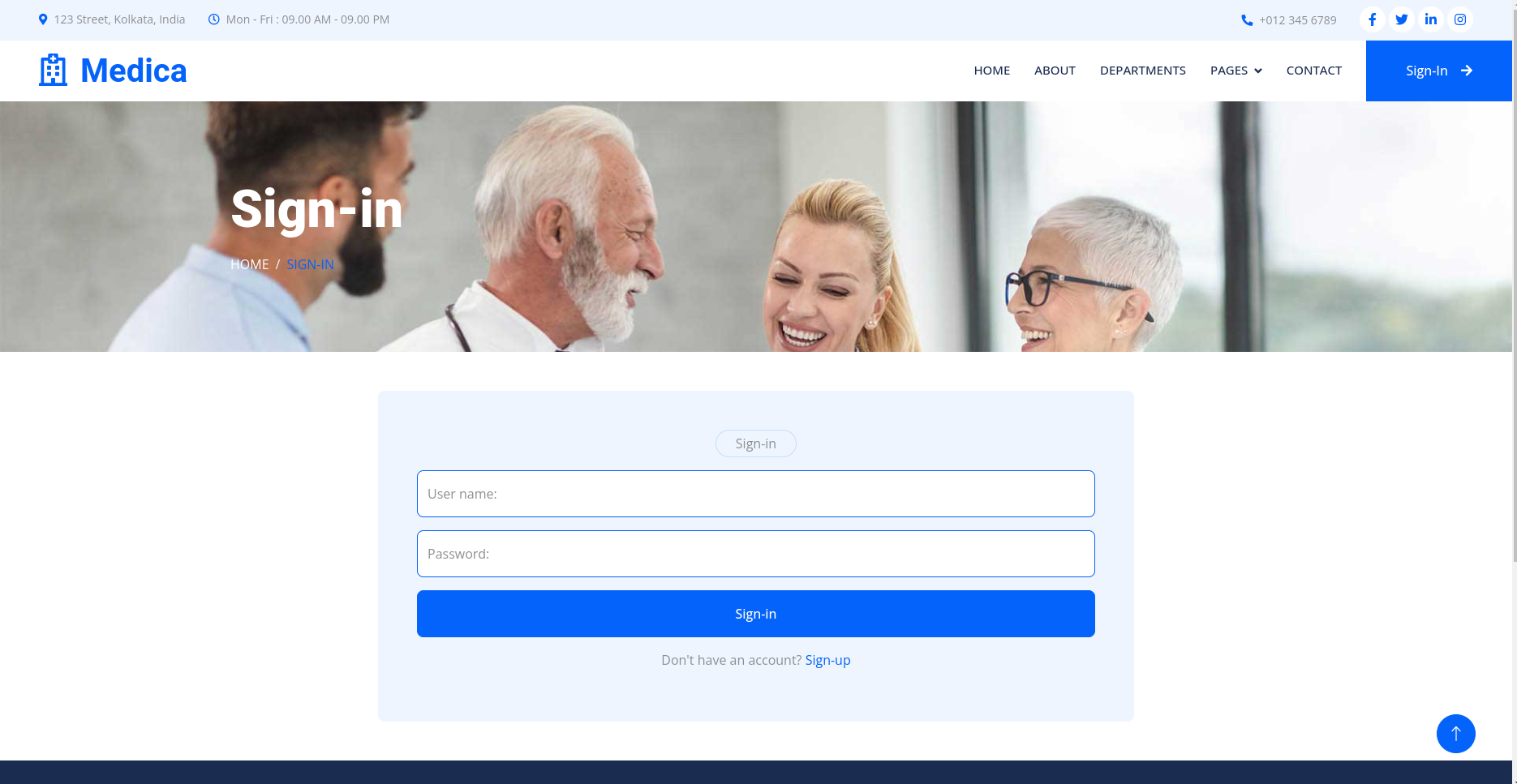
**User Interface Design Home Page**

****

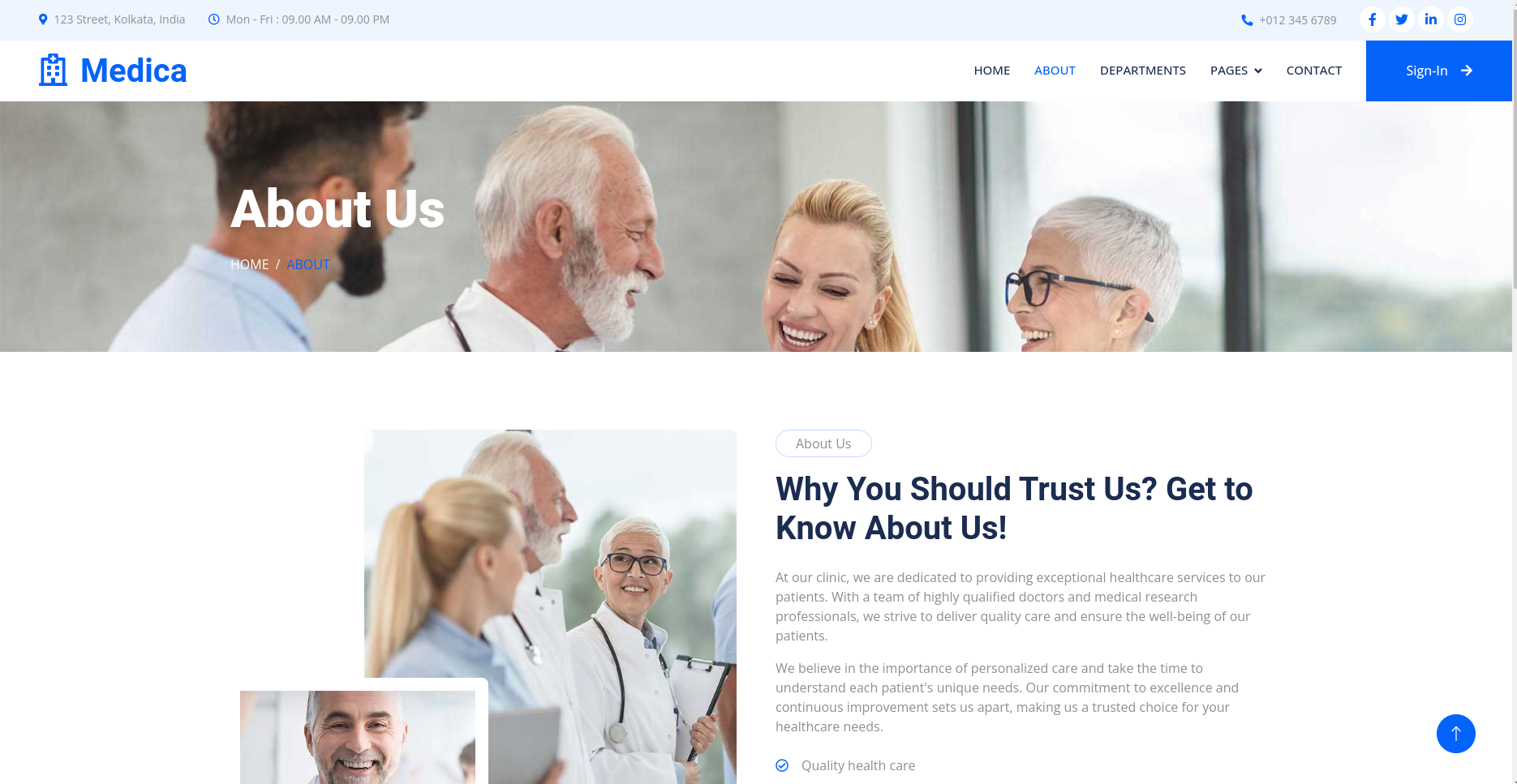
**Patients Registration:-**

****

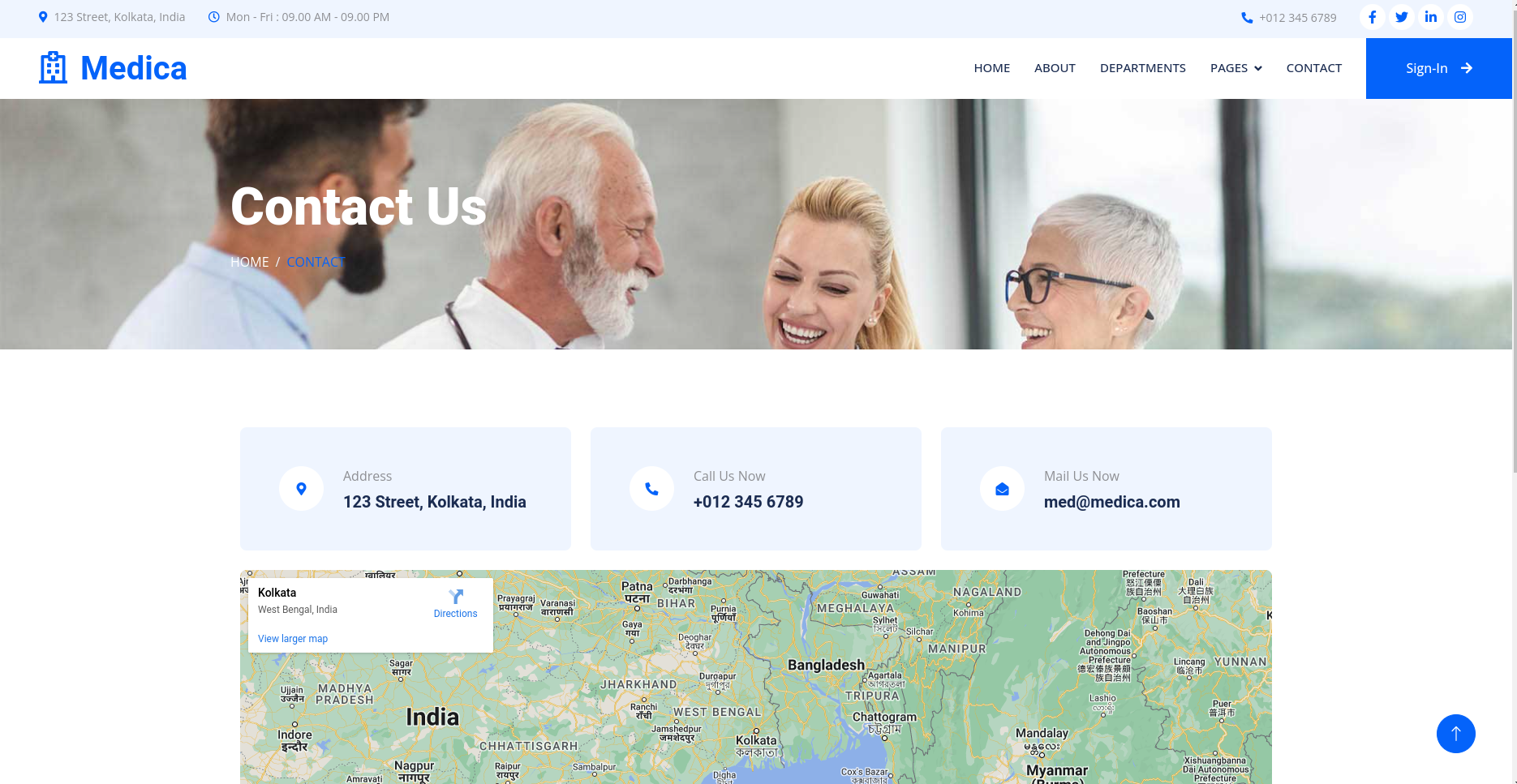
**Login:-**

****

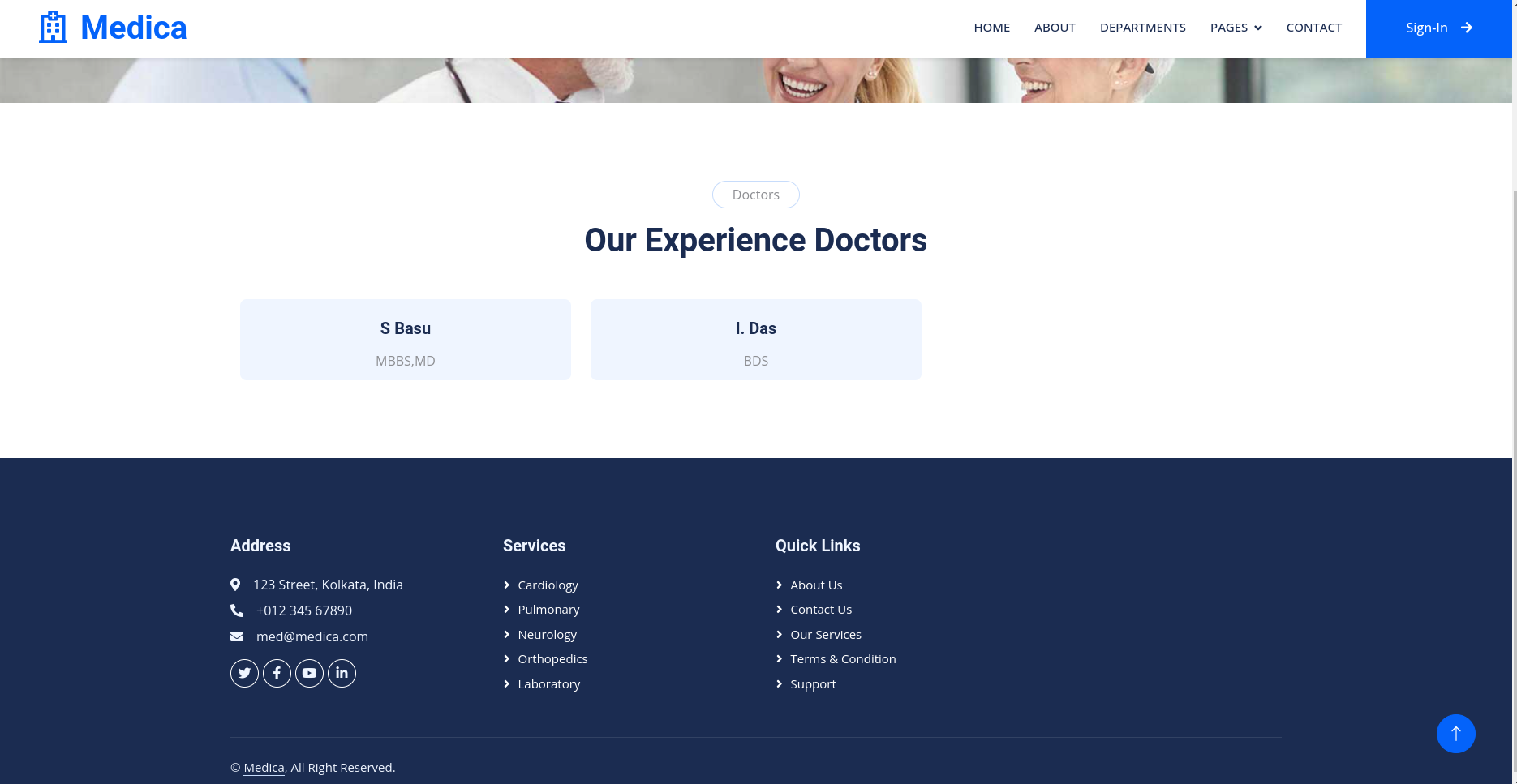
**About us:-**

****

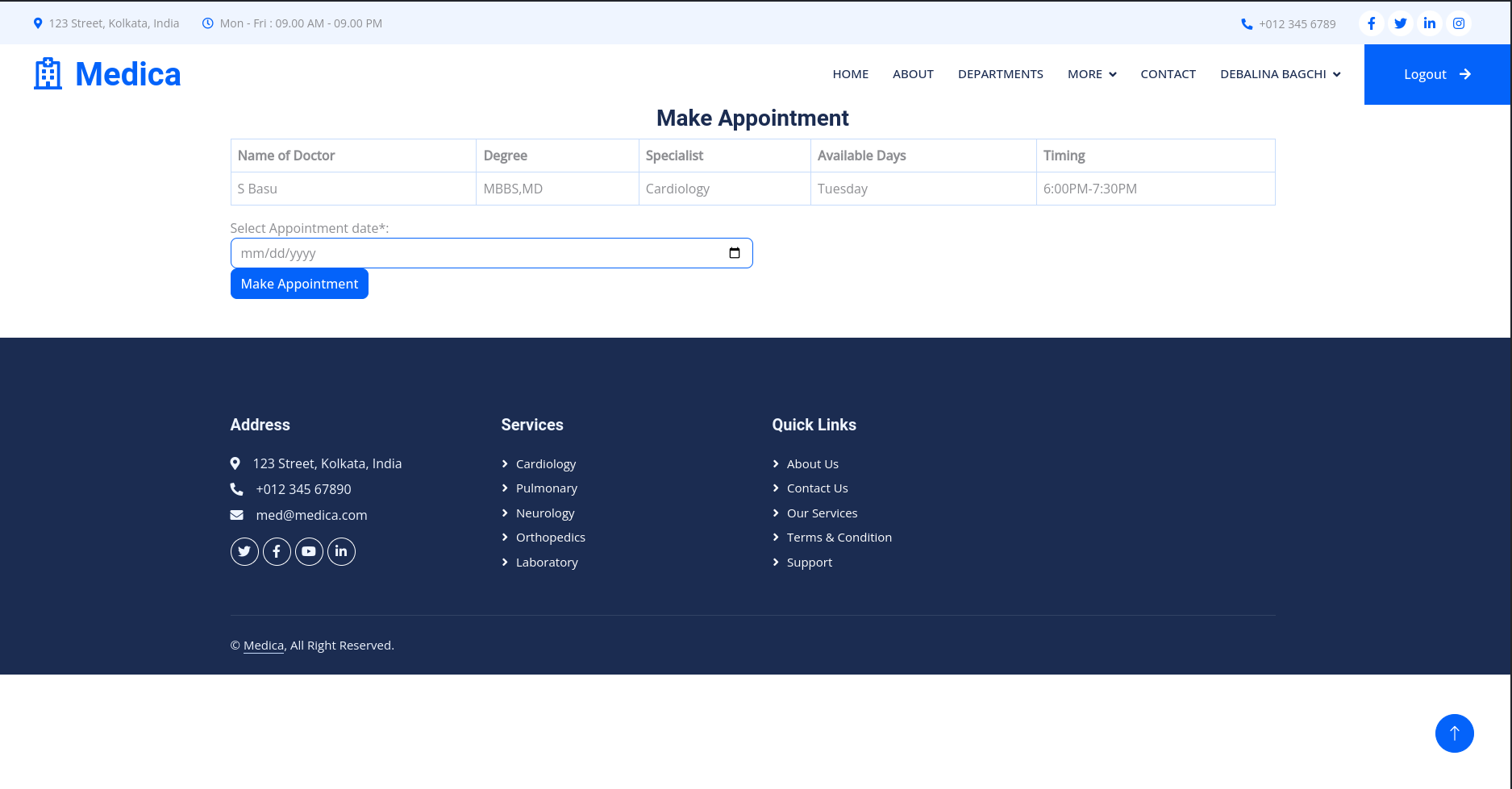
**Contact US:-**

****

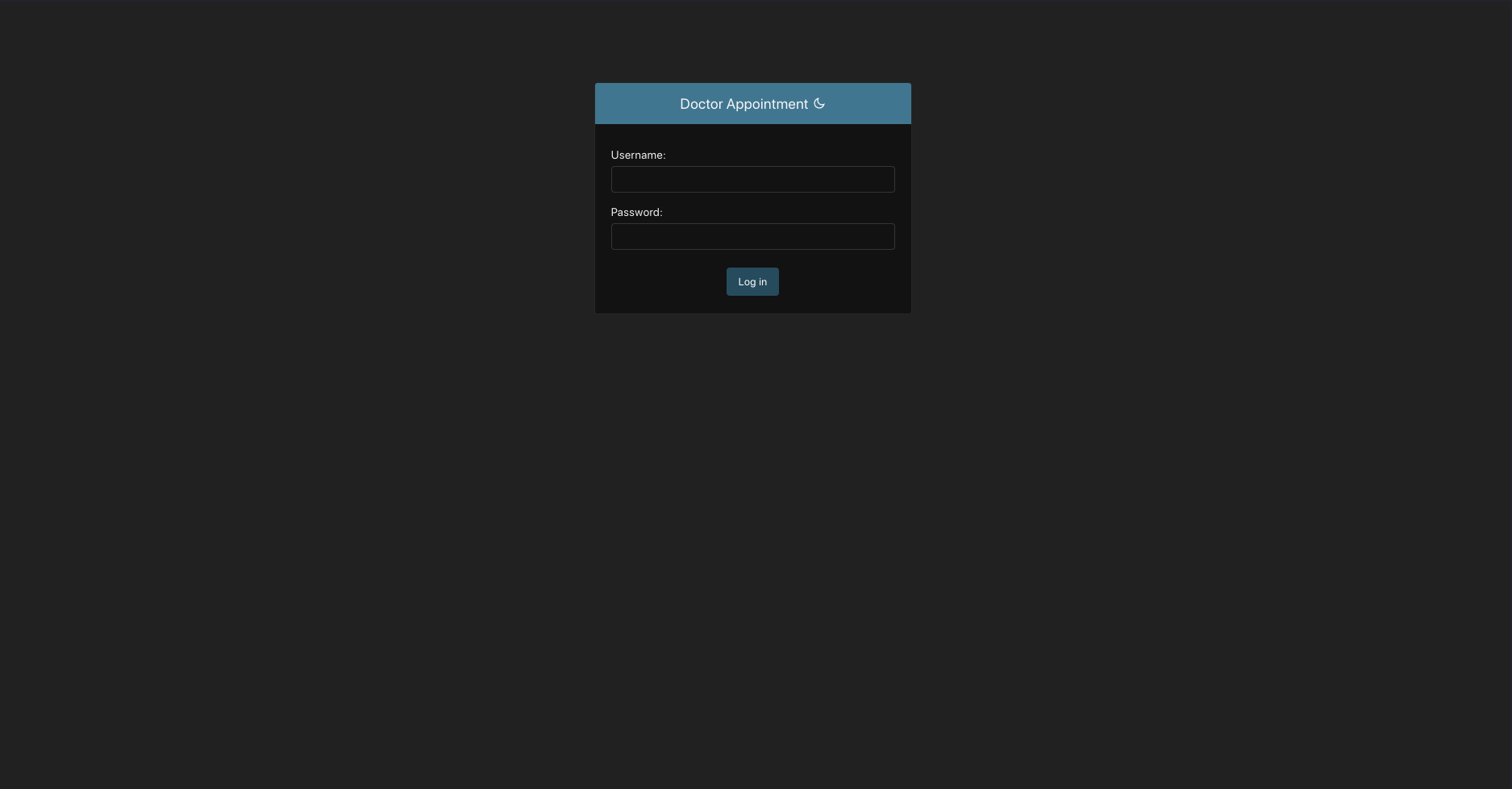
**All Doctors:-**

****

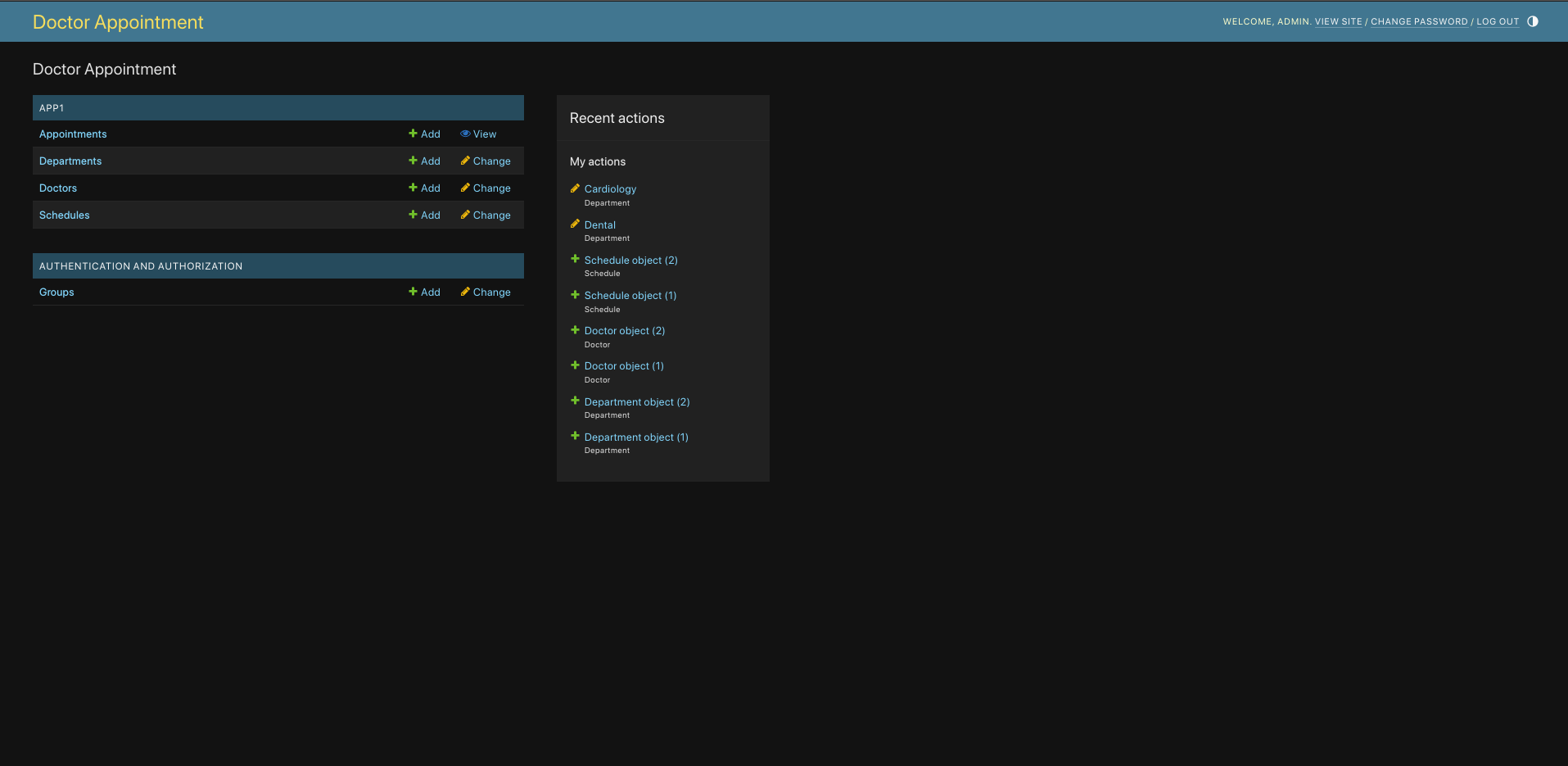
**Fix Appointment:-**

****

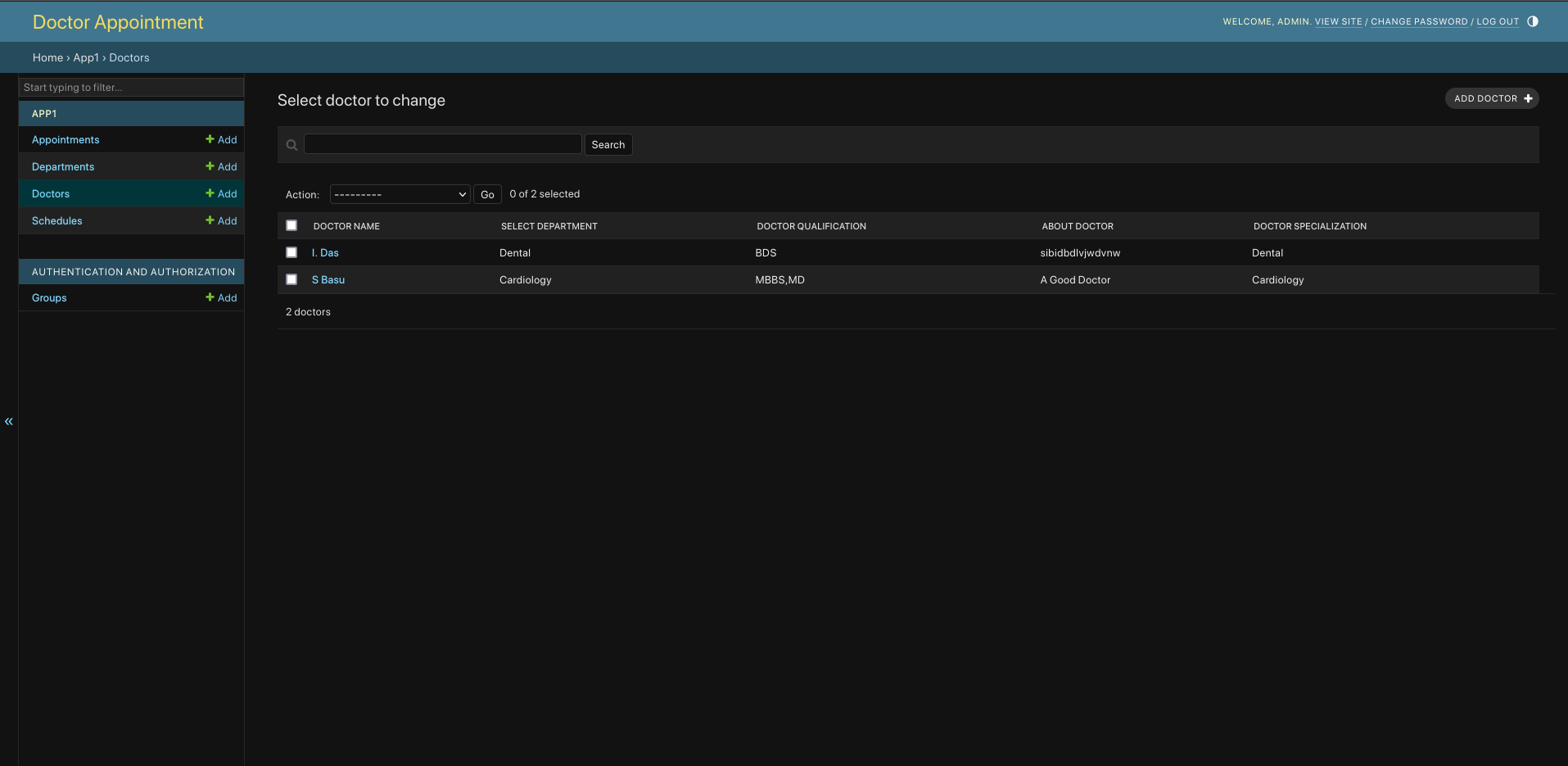
**Administration Login:-**

****

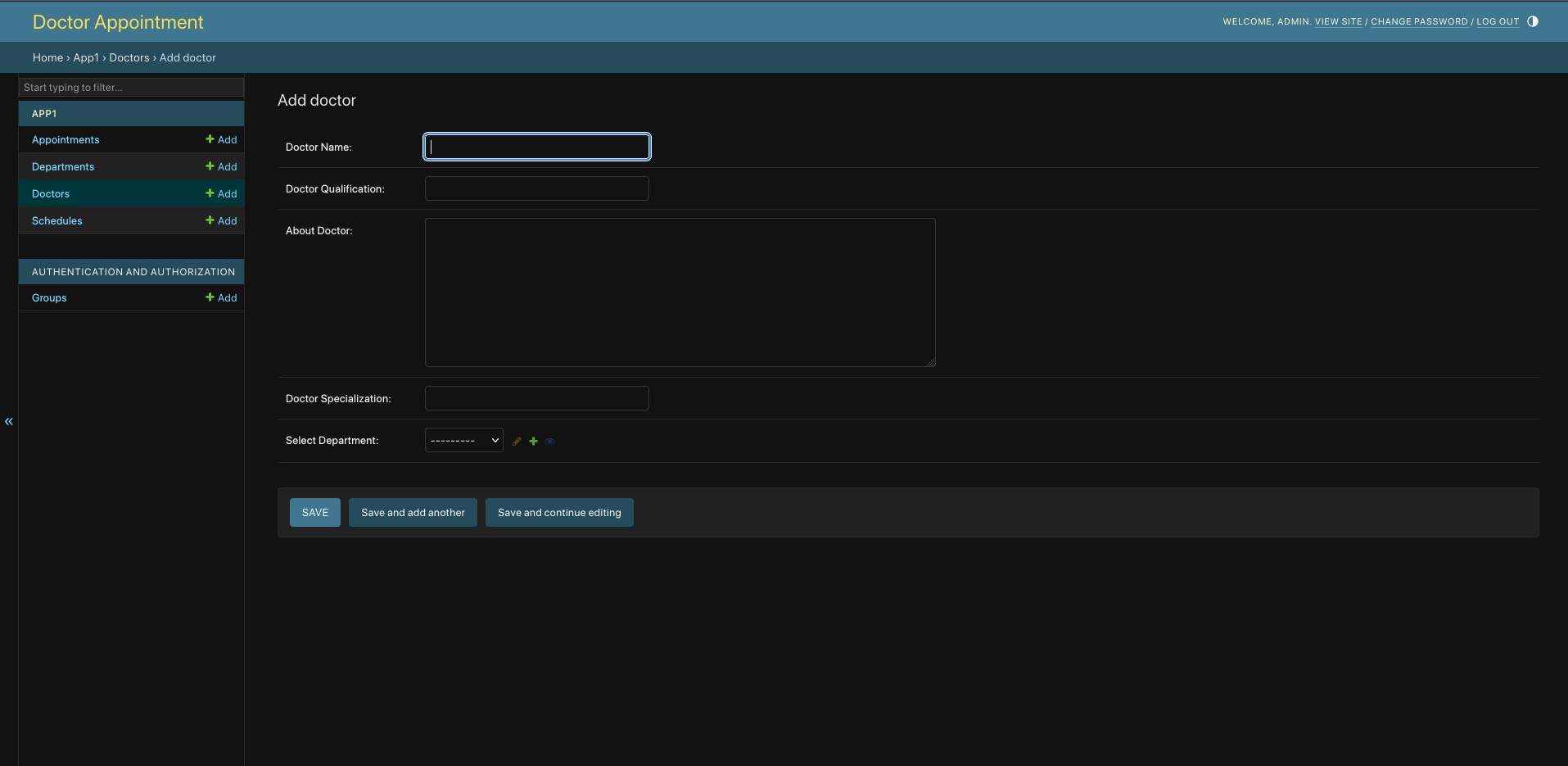
**Admin Dashboard :-**

****

**All Doctors:-**

****

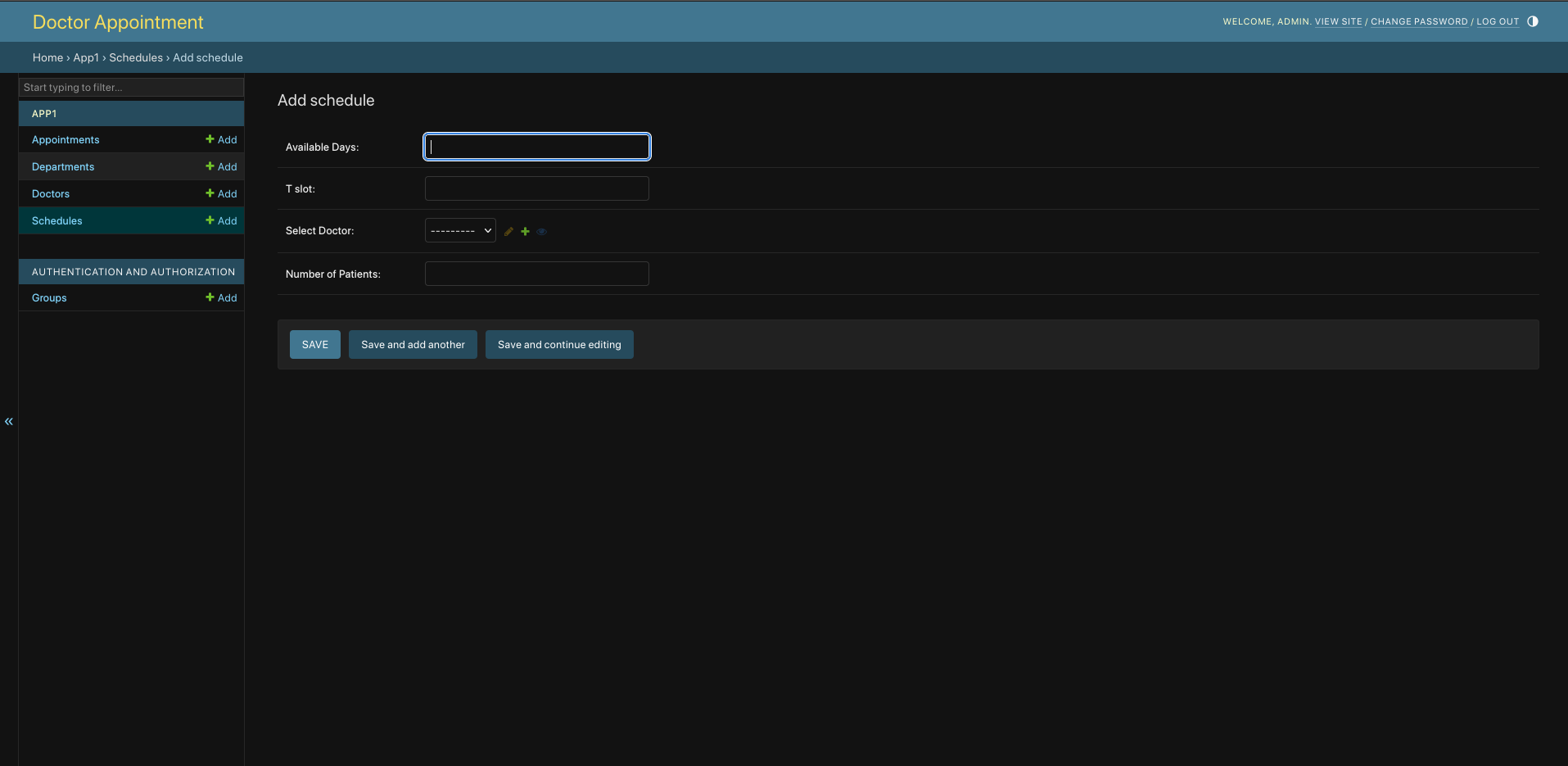
**Add New Doctor :-**

****

**All Doctors Schedule:-**



**Add New Doctor schedule:-**



1. **CODING:-**

**views.py CODING:-**

**from django.contrib.auth import authenticate, login, logout,update\_session\_auth\_hash**

**from django.shortcuts import render, redirect**

**from . forms import PatientForm,SignInForm ,ChangeProfileFrm,ChngePassFrm**

**from django.contrib import messages**

**from .forms import AppointmentForm**

**from .models import Appointment ,Department, Doctor**

**from app1.models import Schedule**

**from datetime import date, timedelta**

**import datetime**

**# Create your views here.**

**def home(request):**

**return render(request, 'app1/index.html')**

**def about(request):**

**return render(request, 'app1/about.html')**

**def contact(request):**

**return render(request, 'app1/contact.html')**

**def services(request):**

**return render(request, 'app1/service.html')**

**def team(request):**

**allDoc=Doctor.objects.all()**

**return render(request, 'app1/team.html', {'allDoc':allDoc})**

**def testimonial(request):**

**return render(request, 'app1/testimonial.html')**

**def feature(request):**

**return render(request, 'app1/feature.html')**

**def signup(request):**

**form=PatientForm()**

**if request.method=='POST':**

**form=PatientForm(request.POST)**

**if form.is\_valid():**

**try:**

**form.save()**

**messages.success(request, 'Account created successfully')**

**except Exception as e:**

**messages.error(request, 'Account creation failed')**

**else:**

**form=PatientForm()**

**return render(request, 'app1/signup.html', {'form':form})**

**def signin(request):**

**if request.POST:**

**frm=SignInForm(request=request, data=request.POST)**

**if frm.is\_valid():**

**uname=frm.cleaned\_data['username']**

**upass=frm.cleaned\_data['password']**

**user=authenticate(username=uname, password=upass)**

**if user is not None:**

**login(request, user)**

**return redirect('/profile/')**

**else:**

**frm=SignInForm()**

**return render(request, 'app1/signin.html', {'form':frm})**

**def userLogout(request):**

**logout(request)**

**return redirect('/signin/')**

**def profile(request):**

**if request.user.is\_authenticated:**

**alldept=Department.objects.all()**

**return render(request, 'app1/profile.html', {'alldept':alldept})**

**else:**

**return redirect('/signin/')**

**def makeApp(request,did):**

**if request.user.is\_authenticated:**

**if request.POST:**

**schedule=Schedule.objects.get(doctor=did)**

**sDays=schedule.days.split()**

**appdate=datetime.datetime.strptime(request.POST.get('appdate'), "%Y-%m-%d").date()**

**appSDay=appdate.strftime('%A')**

**# print(appSDay)**

**# print(sDays)**

**frm=AppointmentForm(request.POST)**

**if frm.is\_valid():**

**f=0**

**for d in sDays:**

**if appSDay==d:**

**f=1**

**if f==1:**

**instance=frm.save(False)**

**instance.doctor\_id = did**

**instance.patient\_id=request.user.id**

**instance.save()**

**messages.success(request, 'Your appointment has been made successfully')**

**else:**

**messages.error(request, 'Doctor will not be available that day')**

**else:**

**frm=AppointmentForm()**

**allDoc=Schedule.objects.raw("SELECT s.\*, d.\* FROM app1\_schedule s INNER JOIN app1\_doctor d ON s.doctor\_id=d.did WHERE d.did={}".format(did))**

**return render(request, 'app1/makeapp.html', {'allDoc':allDoc, 'frm':frm})**

**else:**

**return redirect('/signin/')**

**def appointment(request):**

**if request.user.is\_authenticated:**

**app1=Appointment.objects.raw("SELECT a.\*, d.\*, s.days, s.t\_slot FROM app1\_appointment a INNER JOIN app1\_doctor d ON a.doctor\_id=d.did INNER JOIN app1\_schedule s ON a.doctor\_id=s.doctor\_id WHERE a.patient\_id={}".format(request.user.id))**

**return render(request, 'app1/appointment.html',{'allDoc':app1})**

**else:**

**return redirect('/signin/')**

**def doctor(request,deptid):**

**if request.user.is\_authenticated:**

**allDoc=Schedule.objects.raw("SELECT s.\*, d.\* FROM app1\_schedule s INNER JOIN app1\_doctor d ON s.doctor\_id=d.did WHERE d.dept\_id={}".format(deptid))**

**return render(request, 'app1/doc.html', {'allDoc':allDoc})**

**else:**

**return redirect('/signin/')**

**def changeProfile(request):**

**if request.user.is\_authenticated:**

**if request.POST:**

**frm = ChangeProfileFrm(request.POST, instance=request.user)**

**if frm.is\_valid():**

**try:**

**frm.save()**

**messages.success(request,'Profile Update successfully')**

**except Exception as e:**

**messages.error(request, 'Profile Could Not Update successfully')**

**else:**

**frm=ChangeProfileFrm(instance=request.user)**

**return render(request, 'app1/changepro.html', {'frm':frm})**

**else:**

**return redirect('/signin/')**

**def chagapss(request):**

**if request.user.is\_authenticated:**

**if request.POST:**

**frm = ChngePassFrm(request.user, request.POST)**

**if frm.is\_valid():**

**frm.save()**

**update\_session\_auth\_hash(request, request.user)**

**messages.success(request, 'Password change successfully')**

**else:**

**frm=ChngePassFrm(request.user)**

**return render(request, 'app1/changepass.html', {'frm':frm})**

**else:**

**return redirect('/signin/')**

**def cancelApp(request, appid):**

**if request.user.is\_authenticated:**

**try:**

**appointment = Appointment.objects.get(appid=appid,patient=request.user)**

**appointment.delete()**

**messages.success(request, 'Appointment cancel successfully')**

**except Exception as e:**

**messages.success(request, 'Appointment not cancel successfully')**

**return redirect('/appointment/')**

**else:**

**return redirect('/signin/')**

**models.py CODING :-**

**from django.db import models**

**from django.contrib.auth.models import AbstractUser**

**# Create your models here.**

**class Pataient(AbstractUser):**

**mobile=models.CharField(max\_length=15, verbose\_name='Mobile Number')**

**gender=models.CharField(max\_length =6,verbose\_name='Gender' )**

**age=models.CharField(max\_length=2,verbose\_name='Age')**

**class Department(models.Model):**

**deptid=models.AutoField(primary\_key=True)**

**image=models.ImageField(upload\_to='dept\_img/')**

**deptname=models.CharField(max\_length=255, verbose\_name='Department Name')**

**deptdesc=models.TextField(verbose\_name='Department Description')**

**def \_\_str\_\_(self):**

**return self.deptname**

**class Doctor(models.Model):**

**did=models.AutoField(primary\_key=True)**

**dname=models.CharField(max\_length=150, verbose\_name='Doctor Name')**

**dqua=models.CharField(max\_length=255, verbose\_name='Doctor Qualification')**

**aboutdoc=models.TextField(verbose\_name='About Doctor')**

**dspec=models.CharField(max\_length=255, verbose\_name='Doctor Specialization')**

**dept=models.ForeignKey(Department, on\_delete=models.CASCADE, verbose\_name='Select Department')**

**def \_\_str\_\_(self):**

**return self.dname**

**class Schedule(models.Model):**

**sid=models.AutoField(primary\_key=True)**

**days=models.CharField(max\_length=200, verbose\_name='Available Days')**

**t\_slot=models.CharField(max\_length=200)**

**doctor=models.ForeignKey(Doctor, on\_delete=models.CASCADE, verbose\_name='Select Doctor')**

**noofpp=models.CharField(max\_length=2,verbose\_name='Number of Patients')**

**class Appointment(models.Model):**

**appid=models.AutoField(primary\_key=True)**

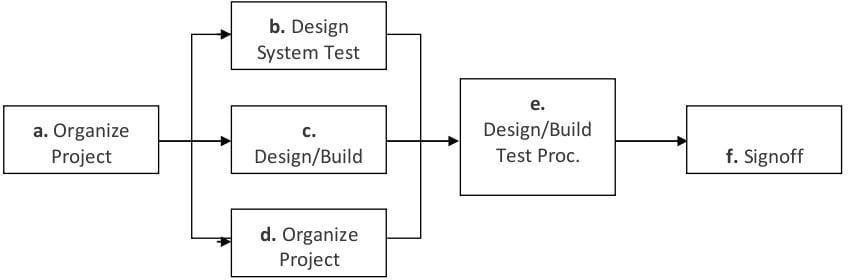
**doctor=models.ForeignKey(Doctor, on\_delete=models.CASCADE, verbose\_name='Doctor')**

**patient=models.ForeignKey(Pataient, on\_delete=models.CASCADE, related\_name='Patient')**

**appmadeon=models.DateField(auto\_now\_add=True, blank=False, verbose\_name='Appointment Made Date')**

**appdate=models.DateField(verbose\_name='Appointment Date')**

**Testing Process**

****

**The diagram above outlines the Test Process approach that will be Folllowed:**

1. Organize Project involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.
2. Design/Build System Test involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.
3. Design/Build Test Procedures includes setting up procedures such as Error Management systems and Status reporting.
4. Build Test Environment includes requesting/building hardware, software and data set-ups.
5. Execute System Tests – The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

##### Testing Strategy

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested,the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use case is shown in Figure 2.1 and onwards.

**Test Cases**

|  |  |
| --- | --- |
| **Tested By:** | **Swati Bhattacharyya** |
| **Test Type** | Unit Testing |
| **Test Case Number Test Case Name** | **1** |
| User Identification |
| **Test Case Description** | The user should enter his/ her accurate userid and password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct userid  ,  password. |
| **Item(s) to be tested** | |
| **1** | Verification of the user id and password with the record in thedatabase. |
| **Specifications** | |
| **Input** | **Expected Output/Result** |
| 1) Correct User id and password | 1) Successful login |
| 2) Incorrect Id or Password | 2) Failure Message |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | **Rakhi Das** |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 2 |
| **Test Case Name** | | Submit Document |
| **Test Case Description** | | The User submit document. Fordocument submission the must  login. |
| **Item(s) to be tested** | | |
| **1** | Check whether the user id logged in. | |
| **2** | Check if the user has selected a document and the size of that is < 5 MB. | |
| **Specifications** | | |
| **Input** | | **Expected Output/Result** |
| 1) Trying to submit document without logging  in. | | 1) The user is redirected to the login page. |
| 2) Check whether document is selected | | 2) A message asks the user to select a document. |
| 3) Check whether the size of the document is < 5 MB. | | 3) Return an error message. |

**Unit Testing:-**

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules.

The unit test cases shall be designed to test the validity of the programs correctness.

**White Box Testing:-**

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply. Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

**Black Box Testing:-**

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

### System Testing:-

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

### System Security measures (Implementation of security for the project developed)

### **Database/Data security:-**

* Database is present in remote machine
* MySQL’s default securities are applied.

**Creation of User profiles and access rights**

* The admin must create users manually

1. **Cost Estimation of the Project along with Cost Estimation Model Analogous estimate of effort or cost**

Used for Early Estimate or Individual Activity Estimate Sample example shown below is for two major deliverables of a software project. You use a previous project as a benchmark for analogous estimation. Using your experience you will estimate a multiplier.

**Multipliers:**

1. Prototyping: 0.75.
2. Testing: 0.5
3. Deployment: 0.5

Finally, if you want to convert to cost, you would use current rates for the resource

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
| **WBS ID** | **Previous Similar**  **Project Activity** | **Previous Effort** | **Current Project Estimate** | **Multiplier** | **Effort (Previous**  **Effort \* 0.75)** | **Cost (Rs.**  **500/hr.)** |
| 1 | Prototyping | 40  WorkHours | Prototyping | 0.75 | 30  Workhours | Rs. 15000/- |
| 2 | Testing | 20  WorkHour | Testing | 0.50 | 10  WorkHours | Rs. 5000/- |
| **Total** |  |  |  |  | **40**  **WorkHours** | **Rs. 20000/-** |
| Note: Effort is also called Size and unit of estimation is called either Work-Hour, person-hours   1. **Future scope and further enhancement of the Project**   Future’s day we are adding mobile calling, email or massage confirm portability. We adding new doctors and more schedule date. We adding the online charting with doctors. We adding some special doctors for special treatment**.**   1. **Bibliography:-**   **I)** | | | | | | |

1. Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.
2. Ian Sommerville. Software Engineering (Seventh Edition). Addison- Wesley, 2004.
3. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley Pub Co; 1st edition (August 2, 1995).
4. Introducing HTML5 by Bruce Lawson,Remy Sharp.
5. HTML5 for Web Designer by Jeremy Keith.
6. HTML for World Wide Web(Visual Quick Start Guide) by Elizabeth Castro.
7. HTML5 Up And Running by Mark Pilgrim.
8. The Definitive Guide To HTML5 by Adam Freeman.
9. Pro HTML5 Programing:Powerful APIs for RcherInternet Aplication Development by Peter Lubbers.
10. Dynamic HTML:The Definitive Reference by Danny Godman.
11. CSS:Definitive Guide by Eric.A.Mayer.
12. CSS Cook Book by Christopher Schmitt.
13. CSS:Missing Manual by David Sawyer Mcfarland. 14. CSS Mastery:Advance Web Standard Sollution by Andy Budd,CameronMoll,Simon Collision. 15. CSS Anthology by Rachel Andrew. 16.HancraftedCSS:More Bulletproof Web Design/Bulletproof Essentialsby Dan Caderholm,EthanMarcotte.