**Introduction**

**1.1 Motivation**

Sometimes the task becomes very tedious for the compounder or doctor himself in manually allotting appointments for the customer as per their availability. The patients faces many problem while requesting a appointment. Hence this project offers an effective solution where customers can view many booking slots available and select preferred date and time. Therefore we came with this project.

**1.2 Overview**

The project Doctor Appointment System is to provides patients or any customer an easy way of booking a doctor’s appointment. This system overcomes the issue of managing and booking appointments according to customer’s choice or requirements. Patients first select preferred date and hospital. After it can see the doctors list. After selecting hospitals and doctors you can see the slots. In this project customers can view many booking slots available and select the preferred date and time. The already booked space will be marked red and will not be available for anyone else for the specified time. The booked slot will be yellow and available slots will be green. This system also allows users to cancel their booking. This system also allow patients as well as doctors to update their details anytime. Doctor can accept the request of patients and patients also can cancel their booking. Doctors and patients can connect wih the admin via sending email from contact us page and clear their issues.

The Administrator can add doctors and can see all appointments. Admin have a authority of to remove patients as well as doctors from their database. Admin will add the doctor and provide him a usename and password. Doctor can update his profile later. Admin can see list of patients. Admin can see the appointments list also.

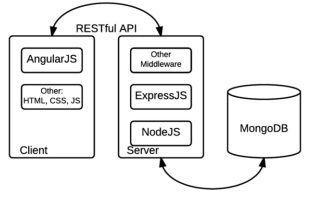
**Introduction of Technology**

**2.1 Introduction of MEAN Stack**

**MEAN** is used to build modern web applications. In this blog, we are going to get a brief introduction on MEAN and how to install it in your system.

As the acronym of MEAN is [MongoDB](https://acadgild.com/database/mongodb-training-certification), Node.js, Express, and Angular, it includes all the 4 technologies combined into it. It is designed to give a quick and organized way to develop MEAN based web apps, websites, web services and APIs.

Using MEAN, developers can set up their work with a set of popular tools, which were carefully combined, so the developers need not concentrate on never ending work of system administration, package management, libraries, etc. Instead, they can concentrate on development completely.



Now let’s look into individual technologies included in MEAN.

**2.2 Division of MEAN**

### MEAN is a collection of MongoDB, Express, Angular, Node.js. It includes all the 4 technologies combined into it. Let’s see each in brief.

### MongoDB :

### It’s simple and is used for storing database. MongoDB is a cross-platform document-oriented database classified as NoSQL. MongoDB acts as a database that stores data for your web application. MongoDB provides some interesting features for your application and architecture that makes it popular. MongoDB supports rich query to fetch data from the database. It supports Server-side JavaScript execution which allows any developer to use a single programming language for both client and server side code. MongoDB is easy to install.

Some points regarding MongoDB are givrn below:

1. MongoDB **stores data in flexible, JSON-like documents**, meaning fields can vary from document to document and data structure can be changed over time
2. The document model **maps to the objects in your application code**, making data easy to work with
3. **Ad hoc queries, indexing, and real time aggregation** provide powerful ways to access and analyze your data
4. MongoDB is a **distributed database at its core**, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
5. MongoDB is **free to use**. Versions released prior to October 16, 2018 are published under the AGPL.
6. Getting involved in the MongoDB community is a great way to build relationships with other talented and like minded engineers, increase awareness for the interesting work that you are doing, and sharpen your skills. To learn about the MongoDB community,

### **Express :**

### Express is one the most popular and widely used web frameworks in Node.js development zone. Express is a minimal web server built on Node.js that provides all the essential functionality required for delivering web applications to the browser and mobile devices. ExpressJS allows you to handle Routes, Server, and I/O stuff very easily.

### 

### Angular :

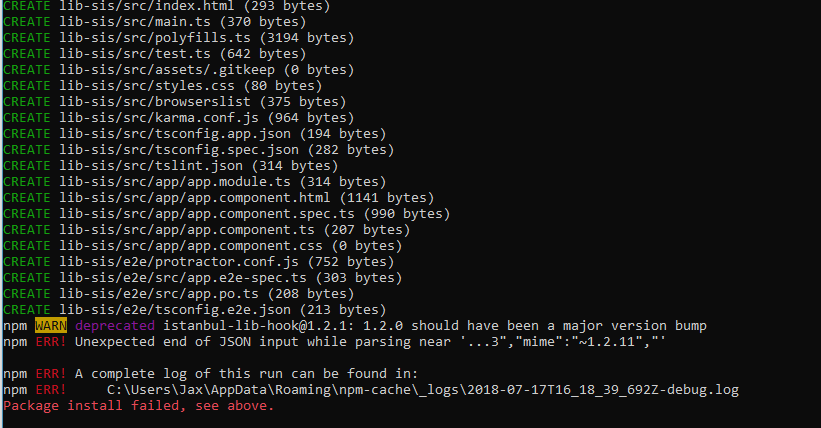
### Angular used to be the “golden child” among JavaScript frameworks, as it was initially introduced by Google corporation. It was built with the [Model-View-Controller concept](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) in mind, though authors of the framework often called it “Model-View.

The framework, written in pure JavaScript, was intended to decouple an application’s logic from DOM manipulation, and aimed at dynamic page updates. Still, it wasn’t very intrusive: you could have only a part of the page controlled by Angular. This framework introduced. many powerful features allowing the developer to create rich, single-page applications quite easily. Specifically, an interesting concept of[**data binding**](https://docs.angularjs.org/guide/databinding) was introduced that meant automatic updates of the view whenever the model (data) changed, and vice versa.

**Anguar new project steps:**

1. npm install -g angular-cli
2. ng new my\_first\_angular\_app
3. ng new my\_first\_angular\_app --style=scss
4. cd my\_first\_angular\_app
5. ng serve

Open a browser on http://localhost:4200/; the app greets us with a message:



**Node.js :**

Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

**Used Node packages :**

1. **Cors :** CORS is a node.js package for providing a [**Connect**](http://www.senchalabs.org/connect/)/[**Express**](http://expressjs.com/) middlewar.
2. **Body-Parser :** Parse incoming request bodies in a middleware before your handlers, available under the req.body property.
3. **Nodemailer:** nodemailer package is used to sending emails.
4. **Multer :** Multer is used to upload the images to the databases.
5. **Sha1:** sha1 is used to encrypt the password fields before save in database .

**2.3 Other Used Languages :**

**1.HTML : HTML**stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages.

**2. CSS : C**ascading **S**tyle **S**heets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

**3. Bootstrap 4 :** Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites. It solves many problems which we had once, one of which is the cross-browser compatibility issue.

**4. JavaScript :** JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. **JavaScript** is very easy to implement because it is integrated with HTML. It is open and cross-platform. Javascript is the most popular **programming language** in the world and that makes it a programmer’s great choice. Once you learnt Javascript, it helps you developing great front-end as well as back-end softwares using different Javascript based frameworks like jQuery, Node.JS etc.

**Software Requirement Analysis**

**3.1 Problem Statement**

Sometimes the task becomes very tedious for the compounder or doctor himself in manually allotting appointments for the customer as per their availability. The patients faces many problem while requesting a appointment. Hence this project offers an effective solution where customers can view many booking slots available and select preferred date and time. Therefore we came with this project.

The project Doctor Appointment System is to provides patients or any customer an easy way of booking a doctor’s appointment. This system overcomes the issue of managing and booking appointments according to customer’s choice or requirements. In this project customers can view many booking slots available and select the preferred date and time. The already booked space will be marked red and will not be available for anyone else for the specified time. This system also allows users to cancel their booking. This system also allow patients as well as doctors to update their details anytime.

There are three types of user

**Administrator:** Administrator can login through the adminpanel.The administrator will be able to add doctors..He/she will be able to see the appointment requests.The administrator can also see the feedback given by the patients.

**Patients:** First patient will sign up and create a new account by entering required details. After creating account customer can login using username and password.After successfully login patient can select particular hospital and doctor then he/she can book an appointment according to the availability of the doctor.

**Doctors :**After the doctor is added by admin the doctor can login using the username and password provided by the admin. After successfully login doctor can update his/her profile and can see the appointment request made by the patients and able to accept and reject them.

**3.2 Modules and their functionalities**

**Admin Panel :**

Below are the some functionalities provided in the admin panel.

**1.Login:** Only admin can login through this panel using unique email and password. After login successfully the admin will relocate to dashboard page.

**2. Change Password:** This functionality provides the admin to change their password.

**3. Hospitals :** This module contains all the hospitals. This module allows the admin to add the hospitals.

**4. Doctors:** This module contains all the doctors that are available on the website. In this admin can add the doctors.

**5. Feedback :**This module contains feedbacks given by patients from the front end. Admin can read the feedbacks of patients.

**6.Appointments:** This module contains all the appointment requests that are made by patients from the main website. Admin can see the all the appointment requests.

**8. Logout:**By clicking on it admin logout successfully.

**Front End(Main Website) :**

Through this front end doctor as well as patient can login and can perform the operations for which they are authorized respectively.

Functionalities provided to the doctors are defined below

**1)Login :** Doctor can login using the username and password provided by the admin.

**2)Profile :**This module contains all the information related to that doctor like education details ,specialization, availability timing and through this module doctor can update the profile.

**3)Appointments :** This module contains all the request that are made by the patients for that particular doctor. Now, doctor can accept or reject the request.

**4)Logout :**By clicking on this button doctor sign out successfully.

Functionalities provide to patient are defined below

**1) Sign Up :** Patient can sign up by entering all the details required in the sign up form .

**2) Login :** Patient can login using the username and password entered at the time of signup.

**3) Profile :** This module contains the information related to that patient like name, age, address, patient image and can update his profile.

**4) Book Appointment :** Patient first select the hospital and the appointment date then after clicking on show doctor button a list of doctor of that hospitals will be displayed. Patient can select the particular doctor and then after clicking on the button show availability then all the slots are visible. Now, patient can select any available slot which is marked in green colour and finally made the request.

**5) My Appointments :**This module contains all the appointments made by the patients and in this module patient can see whether the request is accepted or rejected by the doctor and can also cancel the request.

**6) Feedback :** In this module patients can give his feedback which will be displayed on to the admin panel section.

**Software Design**

**4.1. Use case Diagram :**A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different **use** cases in which the user is involved.

**ADMIN**

**PATIENT**

**DOCTOR**

**4.2. Dataflow Digrams:** Data Flow diagrams show the flow of data from external entities into the system, and from one process to another within the system.

1. **Level -0 Diagram:**The level 0 diagram provides a conceptual view of the process and its surrounding input, output and data stores. It is called context level Data flow diagram also.

**Appointment Management**

**Patient/Doctor**

**Login System**

**Administraion Panel**

**Hospital Management**

**Feedback**

**Update Profile**

**Fig. 2** **Level Zero Dataflow Diagram (DFD)**

**b) Level -1 Diagram:** The level -1 diagram provides a more detailed and comprehensive view of the interaction among the sub-processes within the system

**Admin**

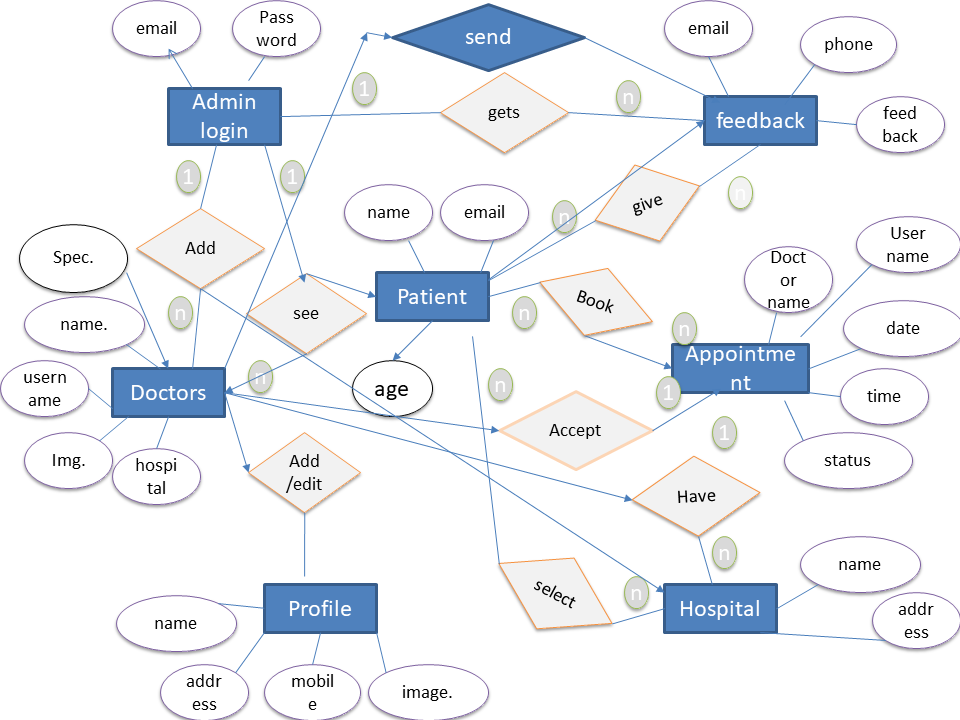
**Feedback**

**Doctor**

**Patient**

**4.3. Entity-Relationship Diagram:**

An **entity-relationship diagram** is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An **entity-relationship diagram**  is a conceptual and representational model of data used to represent the entity framework infrastructure.



As We used MongoDB as a database, So there is no table structutes. We are giving the description of databases.

**1.adminsignup:**

const adminsignup = new Schema({

name:String,

username:String,

email:String,

password:String

});

**2.appointment:**

const appointment = new Schema({

date:String,

doctorid:String,

patientid:String,

pname:String,

dname:String,

time:String,

hospital:String,

staus:String,

accept:String

});

**3. avail:**

const avail = new Schema({

date:String,

doctorid:String,

slotarr:Array

});

**4.doctor:**

const doctor = new Schema({

name:String,

username:String,

email:String,

password:String,

id:String,

education:String,

meetingtime:String,

mobile:Number,

hospital:String,

specilization:String,

image:String

});

**5.feedback:**

const feedback = new Schema({

name:String,

email:String,

subject:String,

messege:String

});

**6.hospital:**

const hospital = new Schema({

name:String,

address:String,

mobile:Number,

image:String

});

**7.patients:**

const patient = new Schema({

name:String,

email:String,

password:String,

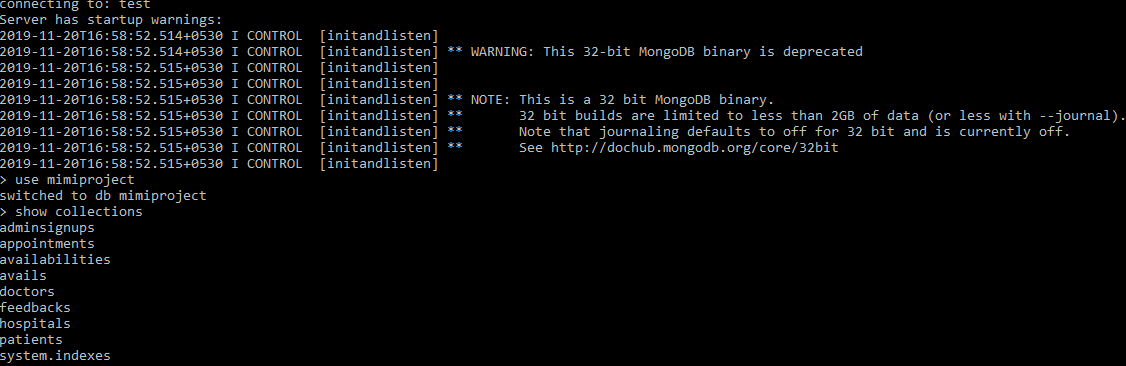
address:String,

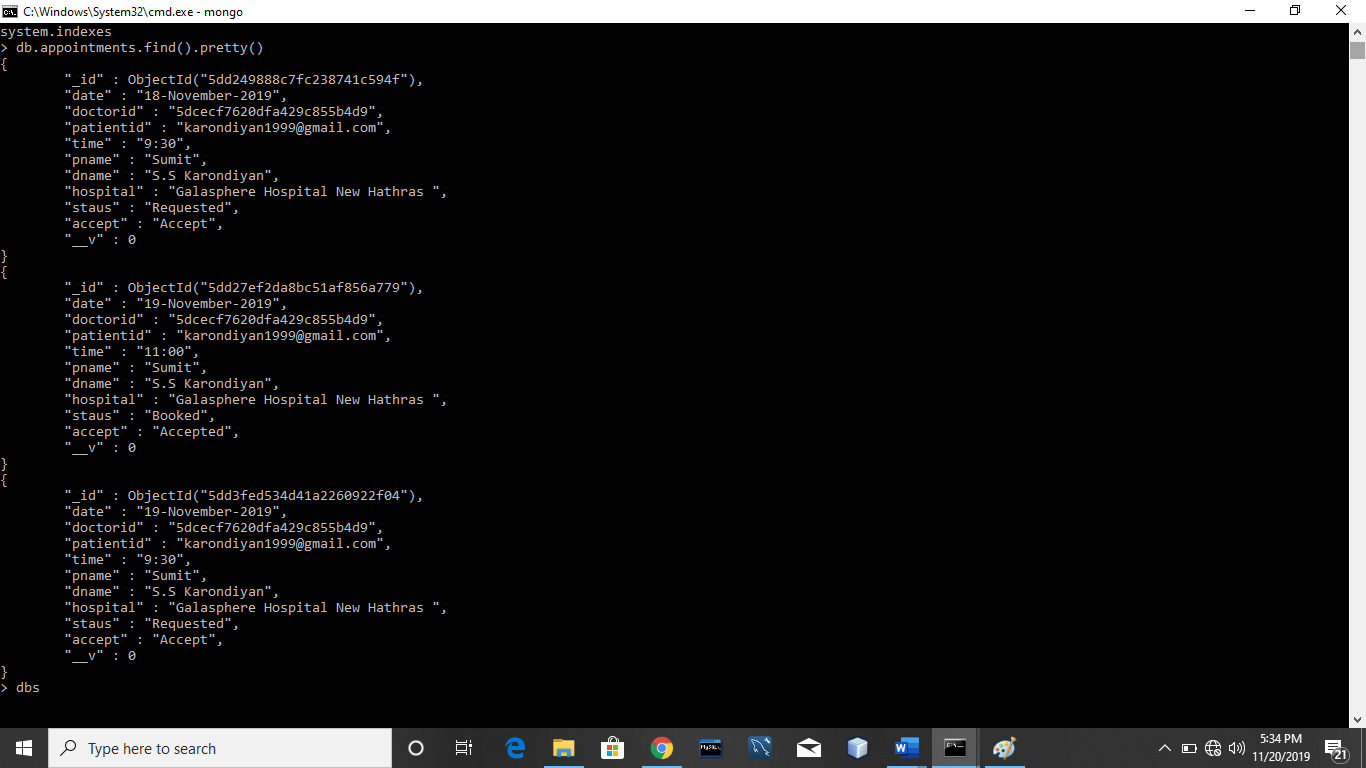
mobile:Number,

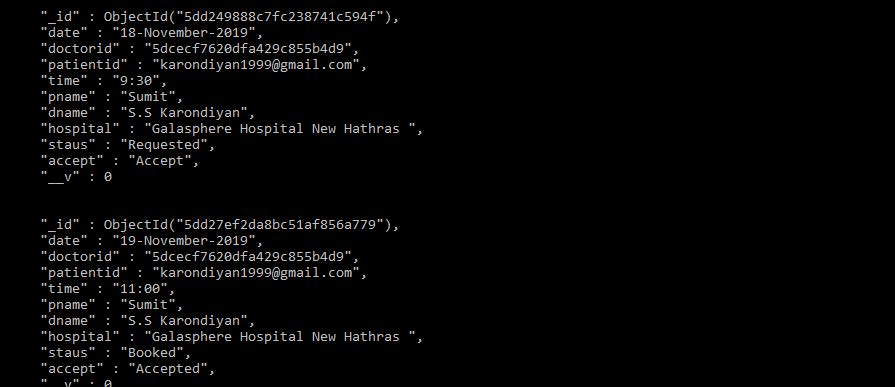
Age:Number,

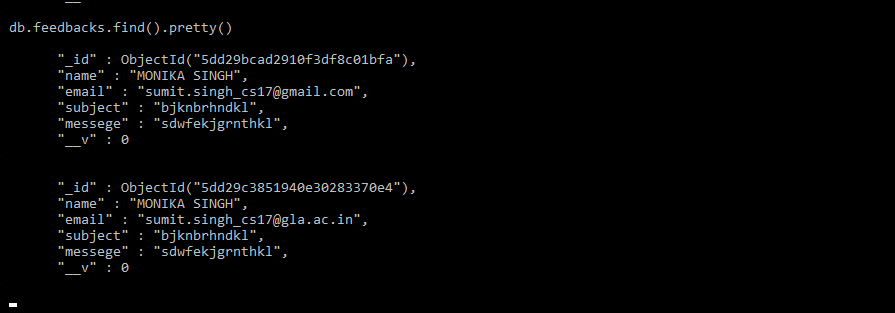
image:String}

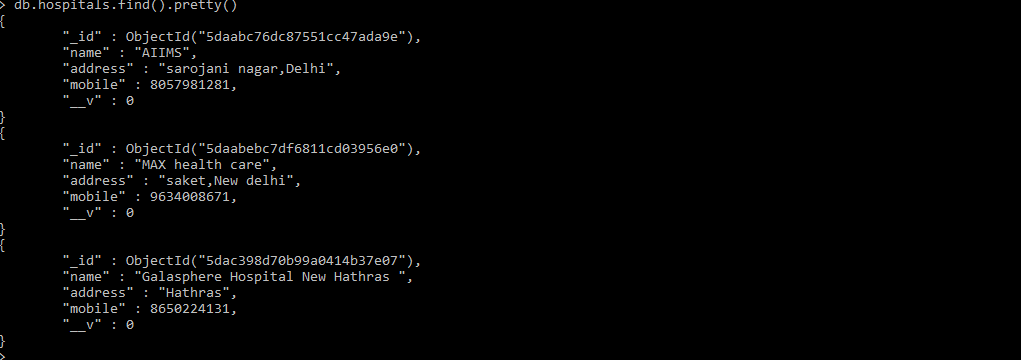
**4.4 Database Data :**

****

****



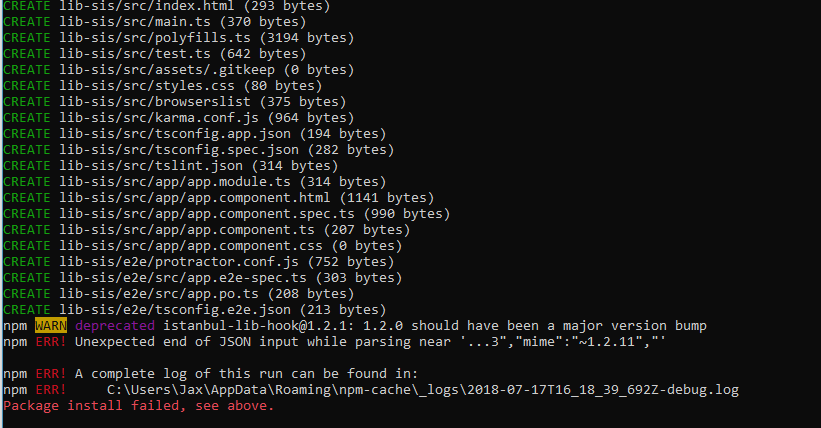




**Coding Process**

We did all the coding process in detailed and clear method. Here we are specifying all the important steps of coding that we have done.

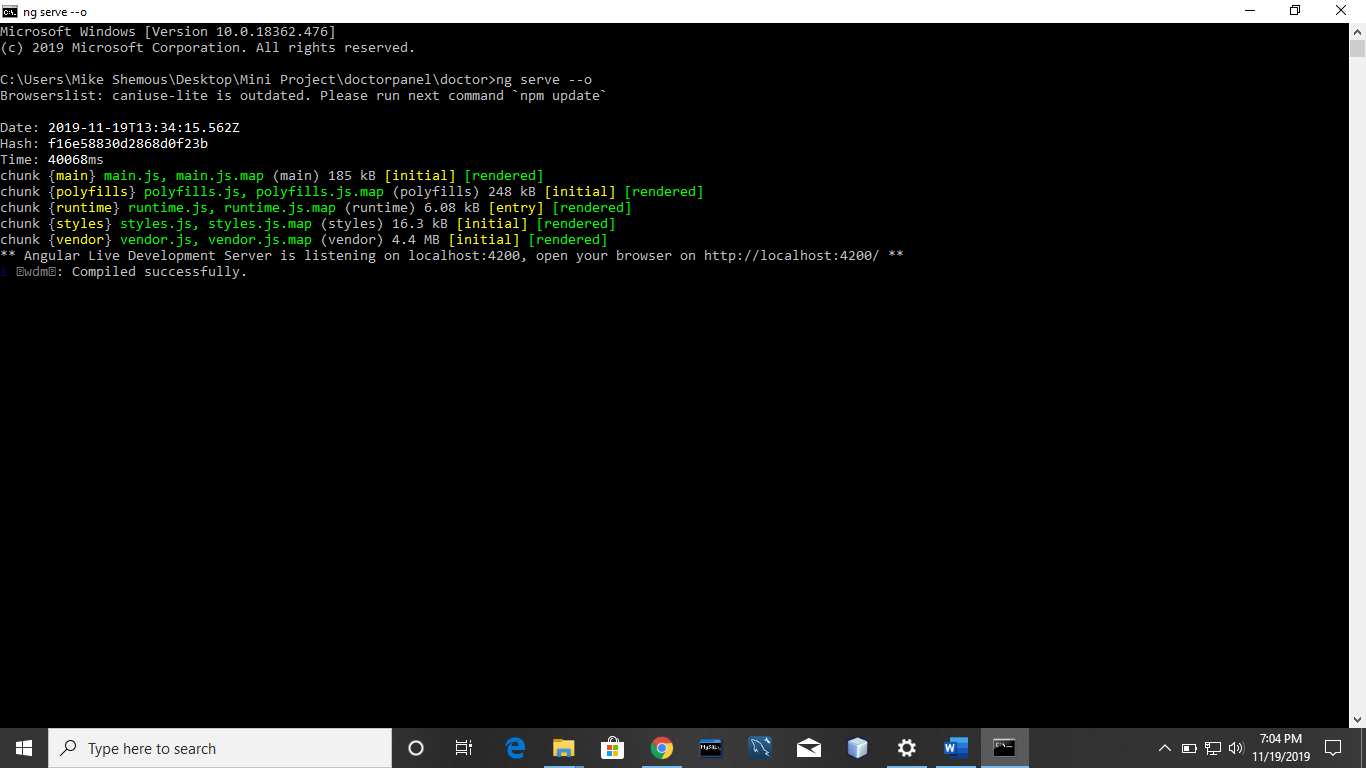
**1.Intialization Step:** In this step we made a angular-cli project for our angular module.We run command on cmd after installing angular-cli on our PC ,**“ng new project\_name”.** This command cretead a angular project.



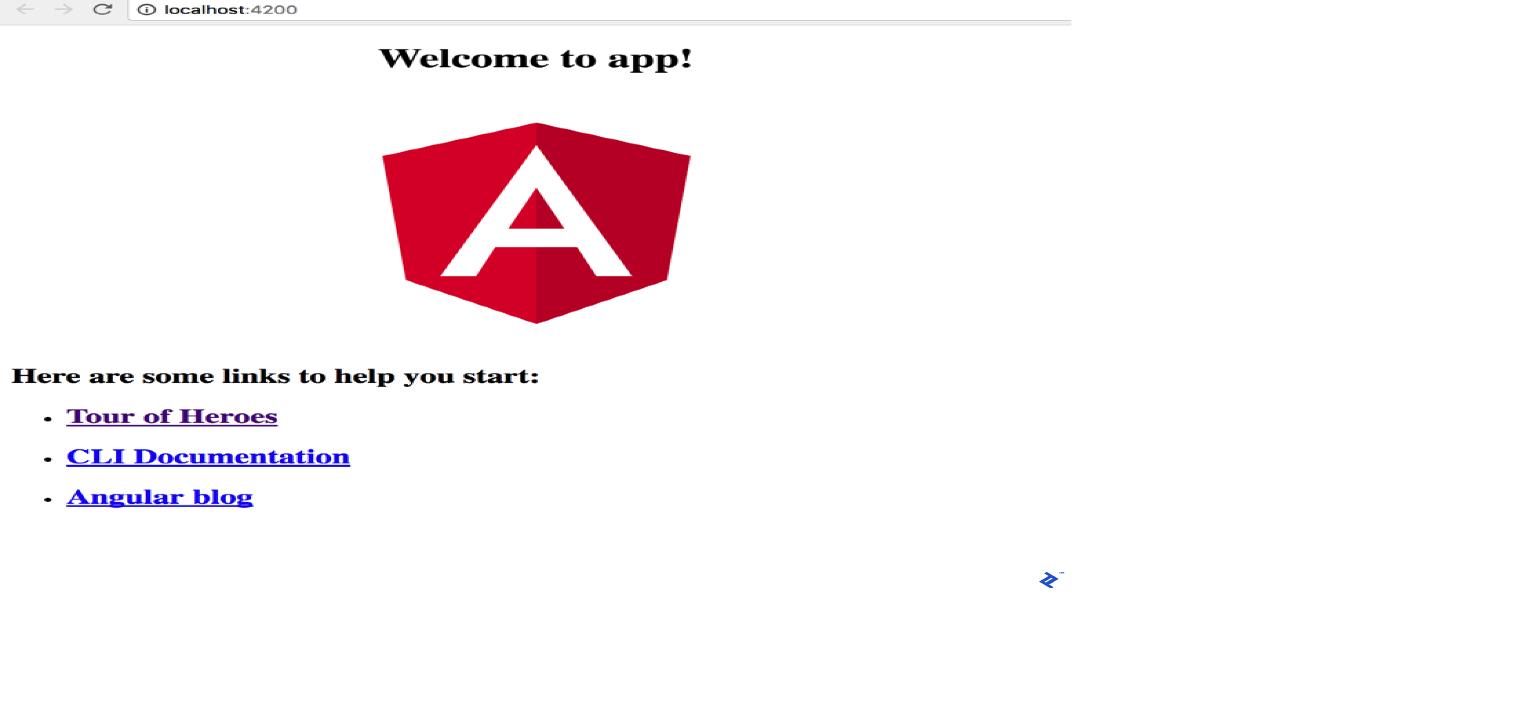
After making angular-cli project. For running of it we have to run a command **“ng serve --o”.**

This command open your angular project on port no 4200 i.e. **http://localhost:4200** .

The functioning of “ng serve –o” command is provided below by a snapshot.



After completion of this command on [**http://localhost:4200**](http://localhost:4200)the first page of your project will open. i.e. The default page of angular project which is given below.



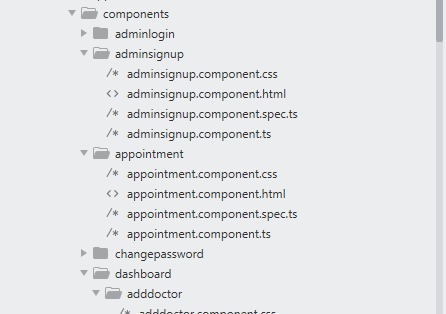
**2.Made components:** In this step we divide our project in various components and we create all the components via command **“ng generate component component\_name ”.** and after making component we did the routing of each components using router outlet.

These components has 4 files in it.

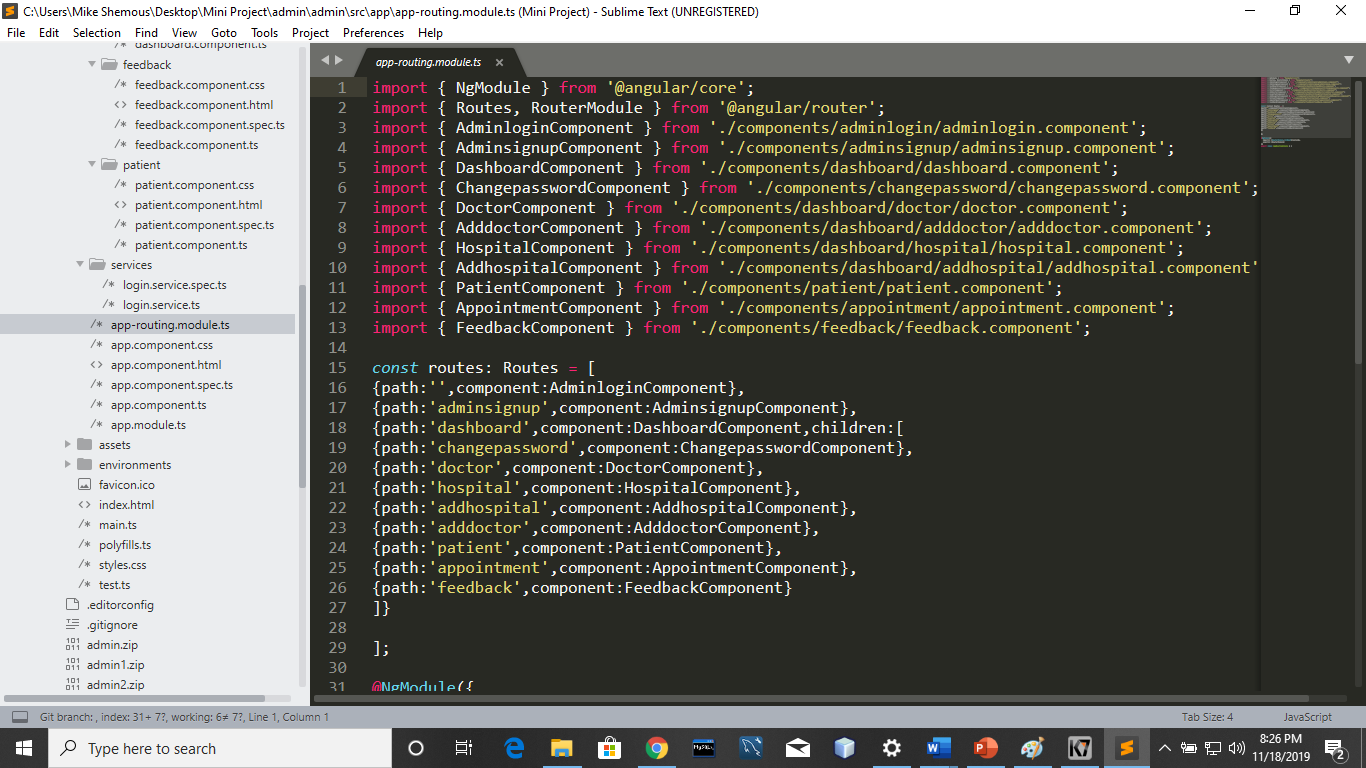
**a).html file:** For html code **b).ts file :** For typescript code

**c).css file** : For CSS 4) .spec.ts file

1. **.html file:** It is the file in which we write html code for our specific component. We make all the frontend in this file.
2. **.ts file:** This file is used to get the html data . It is called Typescript file. We call services from this file using .subscribe method.
3. **.css file:** We can use this file for external css file . We do not need the external file linking. In angular this CSS file is inbuilt.
4. **.spec.ts :** This is only file from which we do not touch.



Here is the snapshot of all the components(Routing Modules) and their routing.



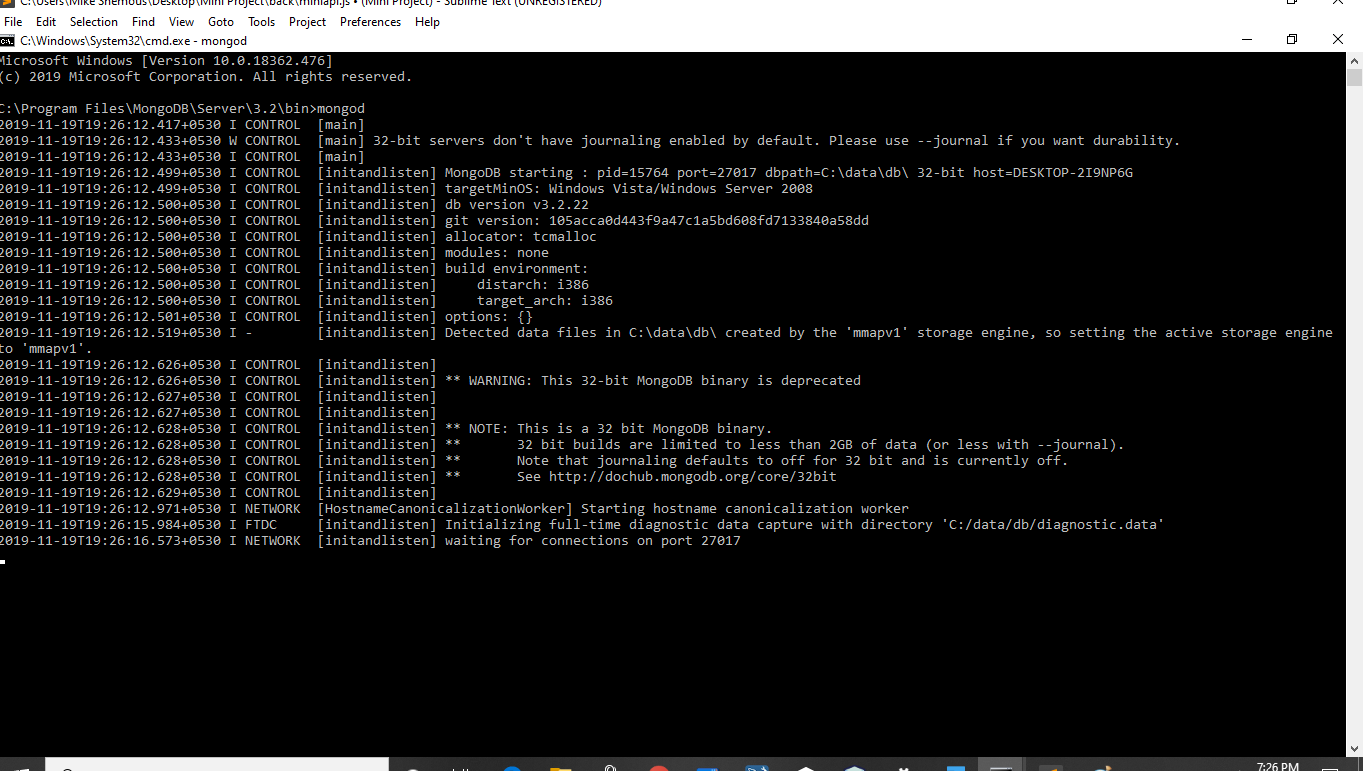
**<router-outlet></router-outlet>:**

**Router**-**outlet** in Angular works as a placeholder which is used to load the different components dynamically based on the activated component or current route state. Navigation can be done using **router**-**outlet** directive and the activated component will take place inside the **router**-**outlet** to load its content

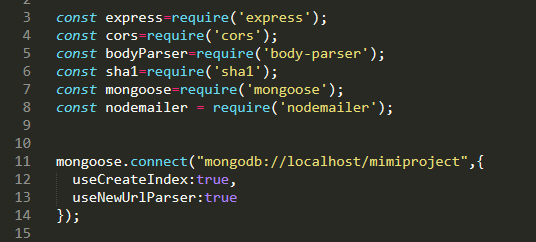
**3.Made Service:** After dividing components we have to make services file for api’s of sending and extracting data from database(connecting backend).We made the service file using command **“ng generate service service\_name”.** The methods called from .ts file are defined in this service file.From here we made the get and post api’s for connecting backend.



1. **Backend File (Database Conenction ):** After creation of API’S we used Node for our backend connection. First we installed node on our PC. And installed **Node package manager(NPM).** Installed all the packages which are required (nodemailer, multer,body-parser,cors etc.). We coonect the mongoDB with our Node. For running of monoDB .fisrt we launchd the command **mongod** on CMD.

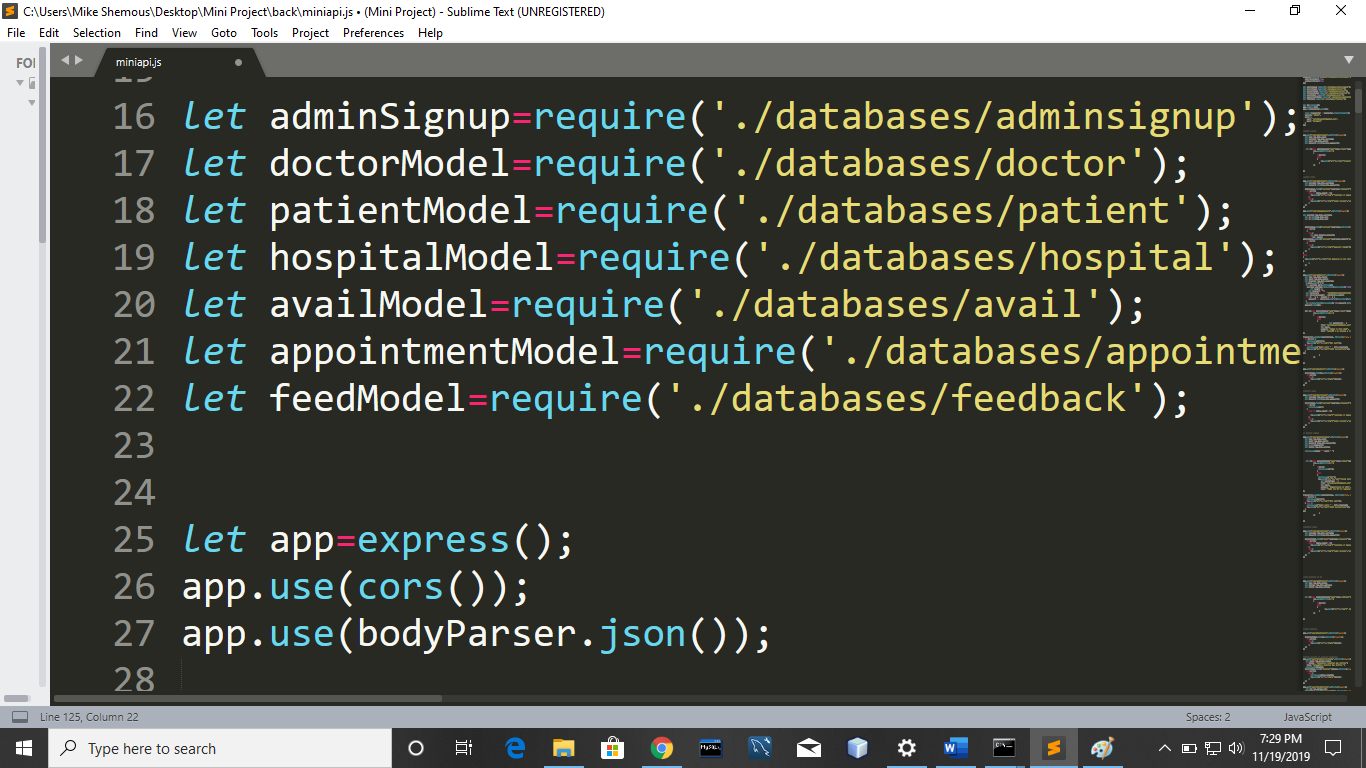


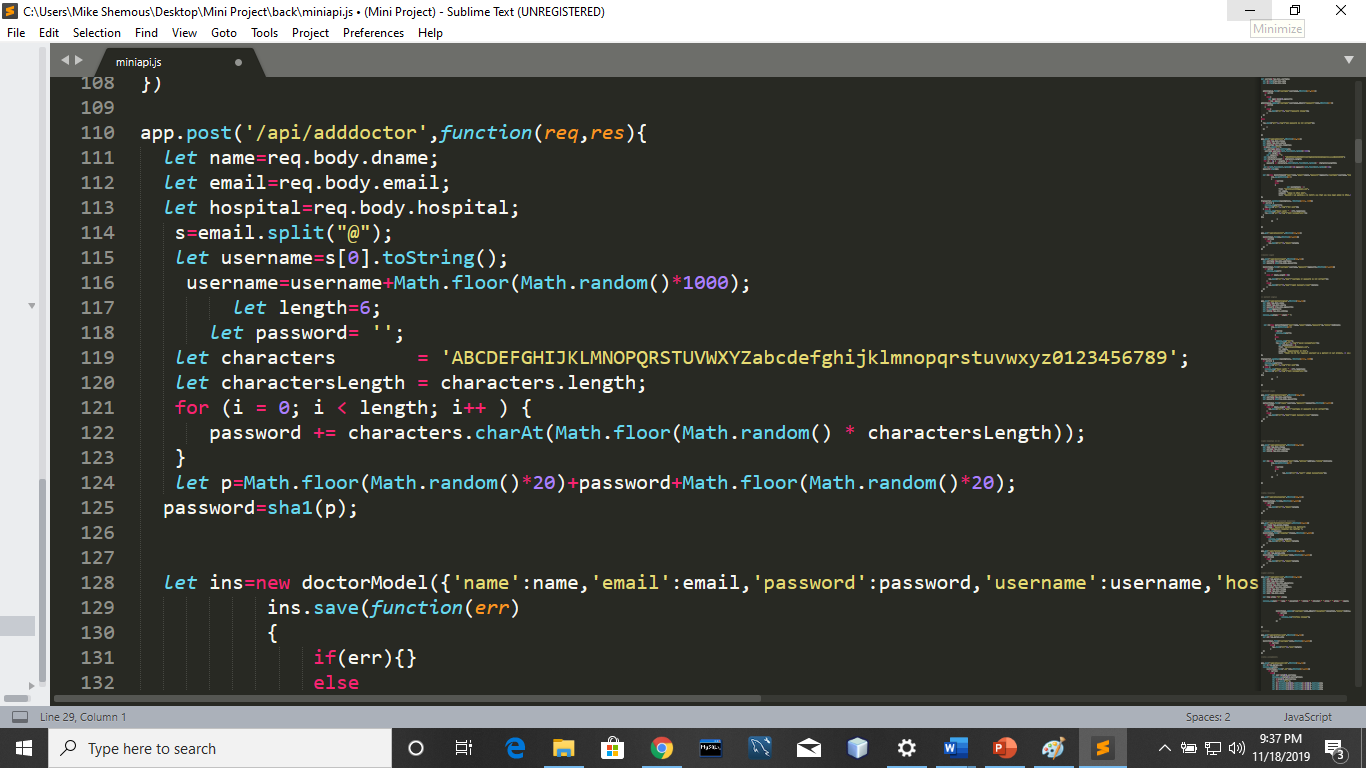
Now after installing all the packages using NPM(Node package manager). We attach these method from the require method of node.

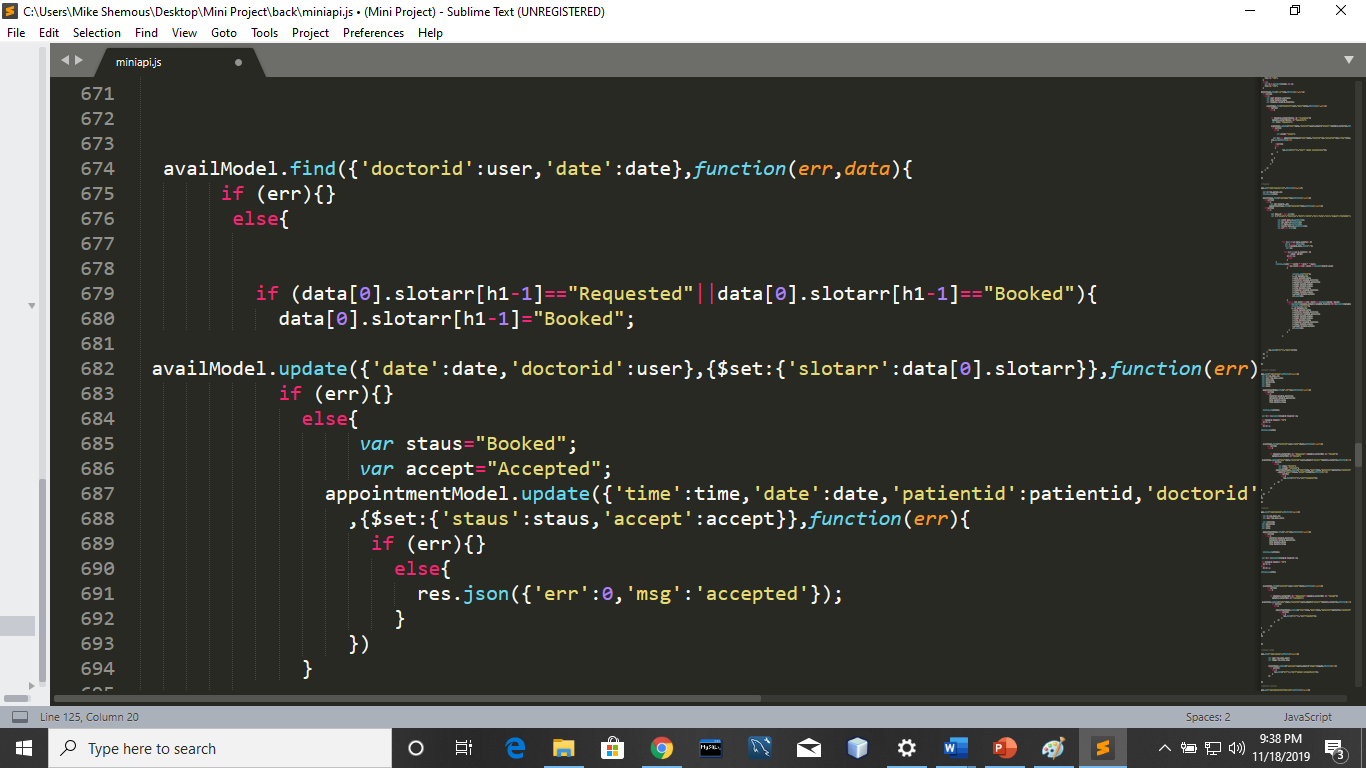
****

**5.Writing Code for database :** This section of coding contains all the logic and algorithms behind our project. Here we store the data into the database and retrieve back it from databases.This section contain almost **1000 lines** of code for all API’s. We are giving a look file.

We require all the databases also.



****

****

**Testing**

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is[Defect](https://www.guru99.com/defect-management-process.html)free. It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a [White Box](https://www.guru99.com/white-box-testing.html) and [Black Box Testing](https://www.guru99.com/black-box-testing.html).

In simple terms, Software Testing means Verification of Application Under Test.

**White Box Testing:**

White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

**Working process of white box testing:**

**Input:** Requirements, Functional specifications, design documents, source code.

**Processing:** Performing risk analysis for guiding through the entire process.

**Proper test planning:** Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.

**Output:** Preparing final report of the entire testing process.

White box testing is very thorough as the entire code and structures are tested.It results in the optimization of code removing error and helps in removing extra lines of code.It can start at an earlier stage as it doesn’t require any interface as in case of black box testing.it is easy to automate.

**Black Box Testing :**

Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Black box testing can be done in following ways:

**1. Syntax Driven Testing –** This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers,language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

**2. Equivalence partitioning –** It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of class would also result into same error.

The technique involves two steps:

1. **Identification of equivalence class –** Partition any input domain into minimum two sets: **valid values** and **invalid values**. For example, if the valid range is 0 to 100 then select one valid input like 49 and one invalid like 104.
2. **Generating test cases –**

(i) To each valid and invalid class of input assign unique identification number.  
(ii) Write test case covering all valid and invalid test case considering that no two invalid inputs mask each other.

To calculate the square root of a number, the equivalence classes will be:  
**(a) Valid inputs:**

* + Whole number which is a perfect square- output will be an integer.
  + Whole number which is not a perfect square- output will be decimal number.
  + Positive decimals

**(b) Invalid inputs:**

* + Negative numbers(integer or decimal).
  + Characters other that numbers like “a”,”!”,”;”,etc.

Now Let’s have a look some of the test cases for our project.

**TestCase 1:**

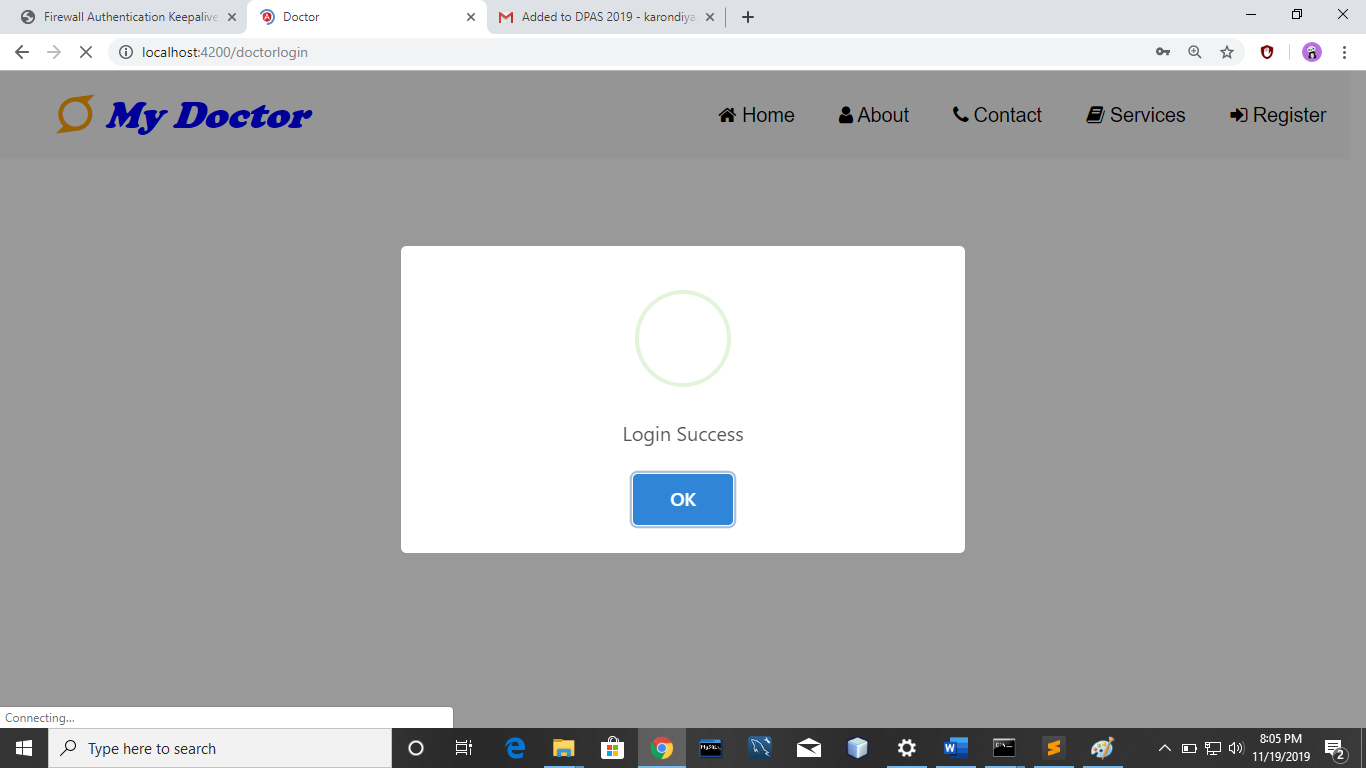
**Login :** Every person related to project have a username and password using that he/she can login to your account.

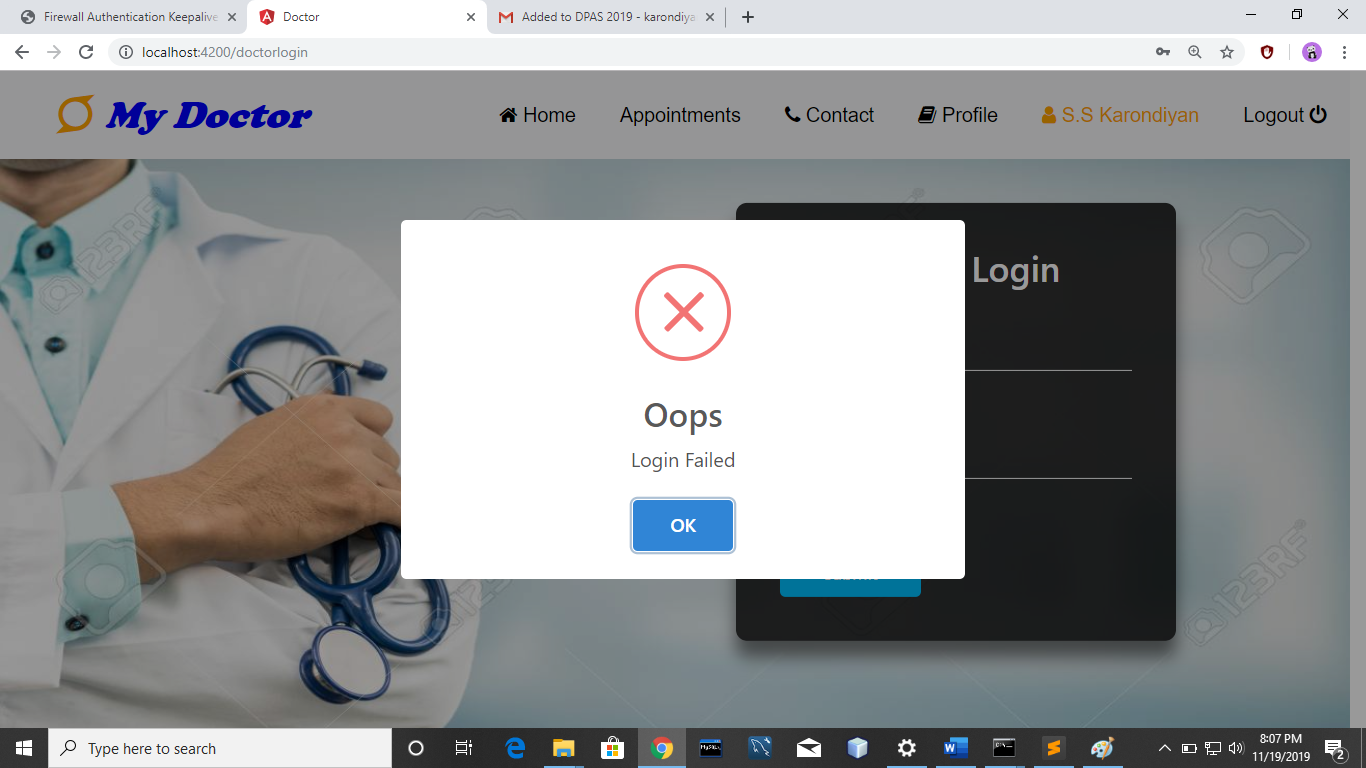
**If (username is coorect && password is correct)**

**Login success**

**Else**

**Login fail**

****

****

**Test Case 2:**

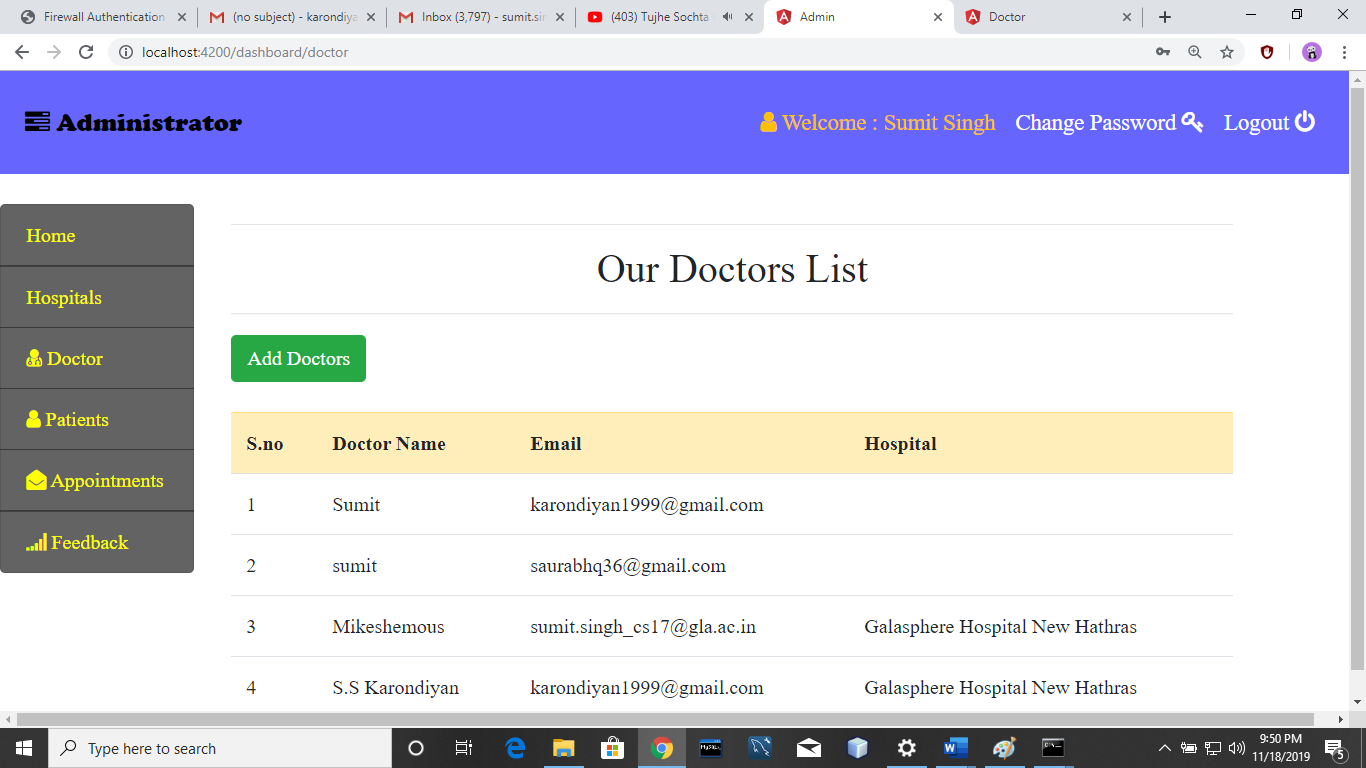
**Book Appointment :** If slot is available then appointment is requested otherwise not.

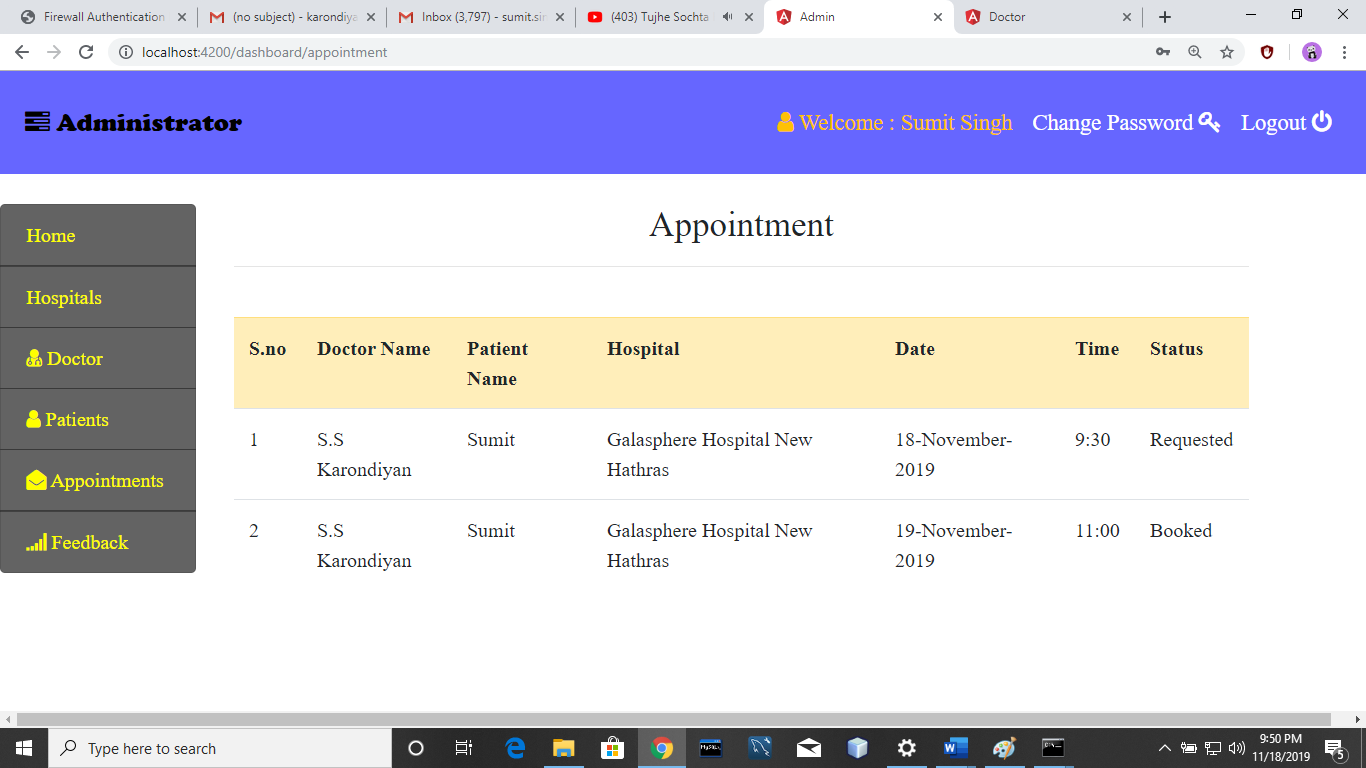


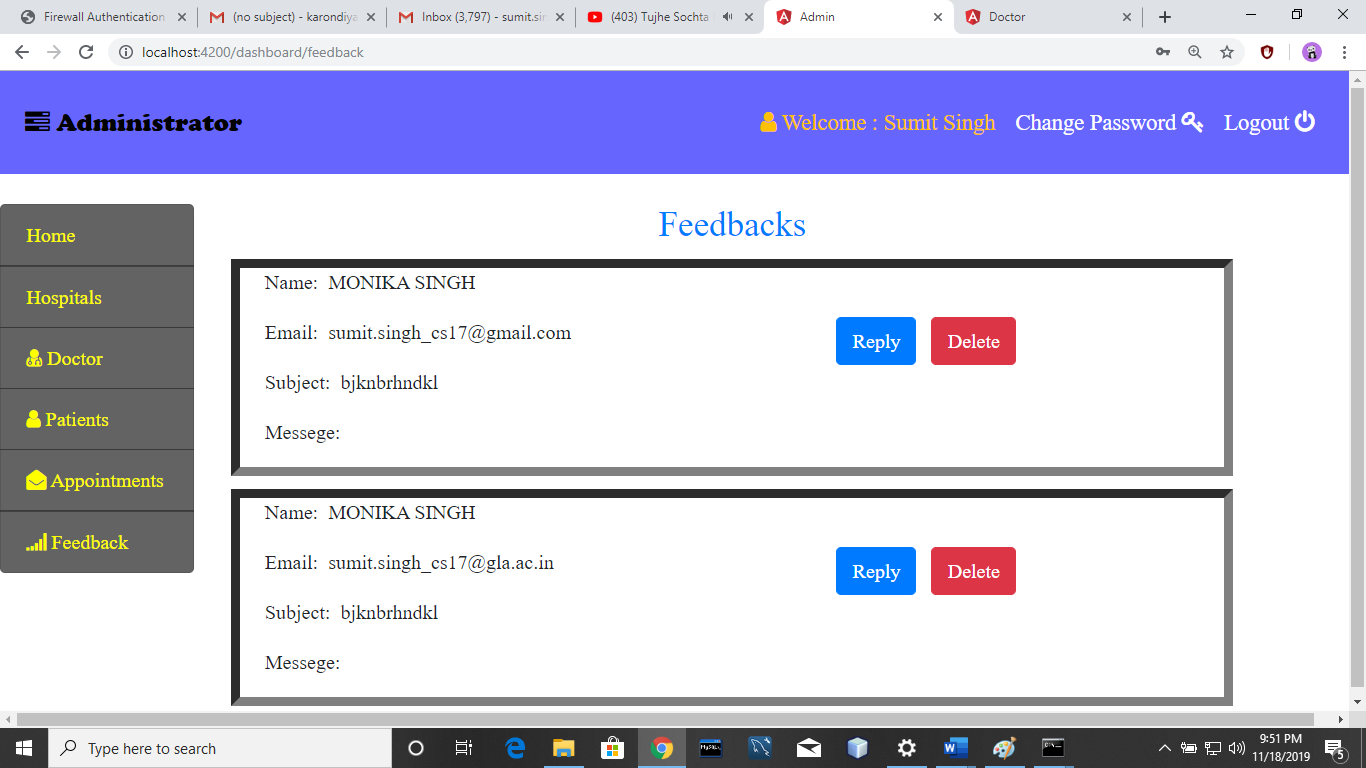
Like above we completed our testing.

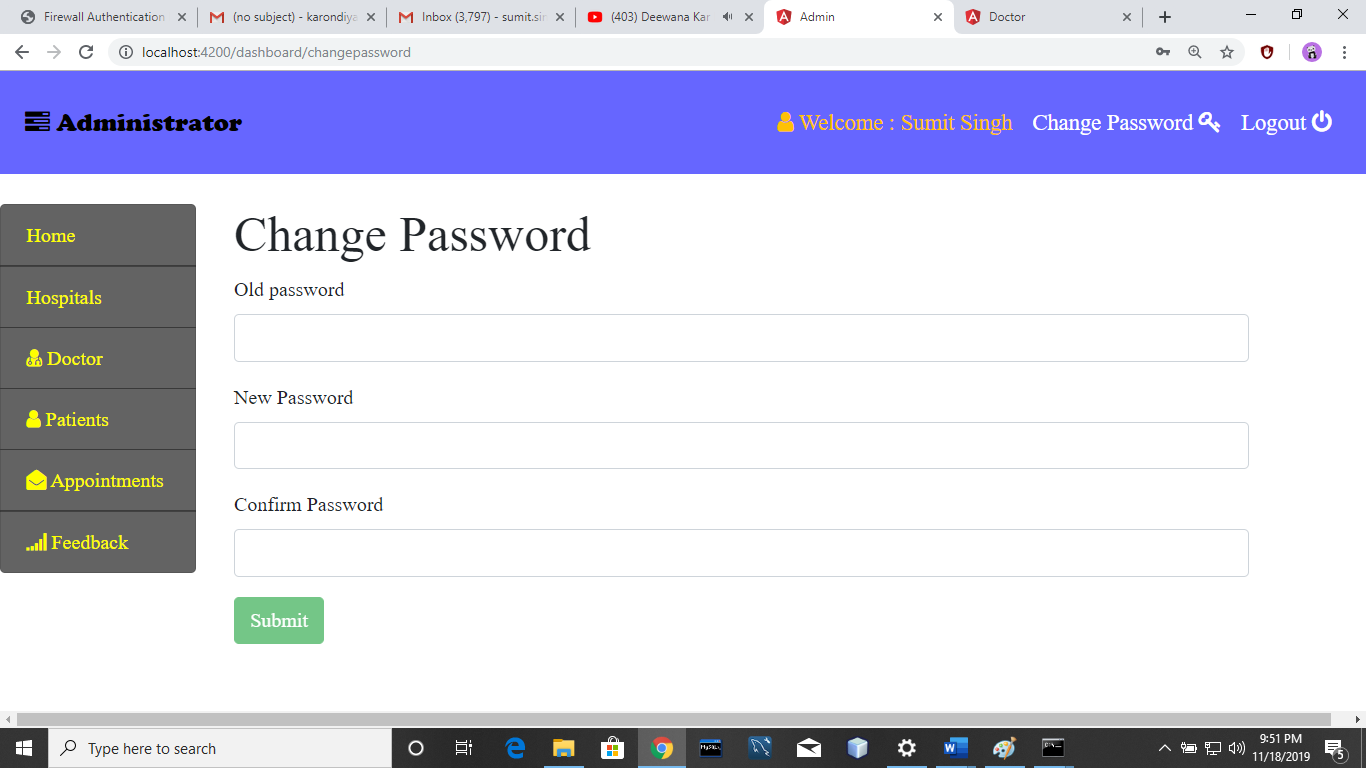
**User Interfaces**

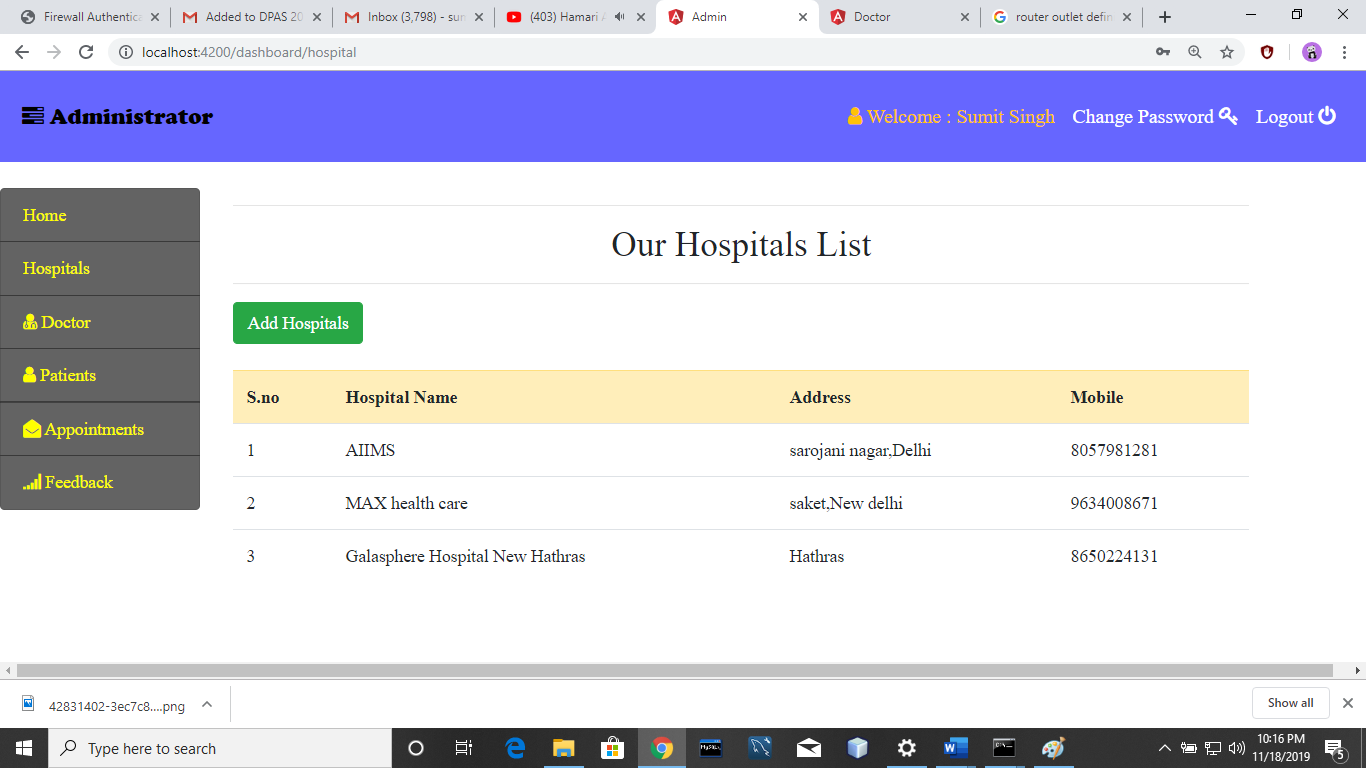
* 1. **Admin panel :**

****

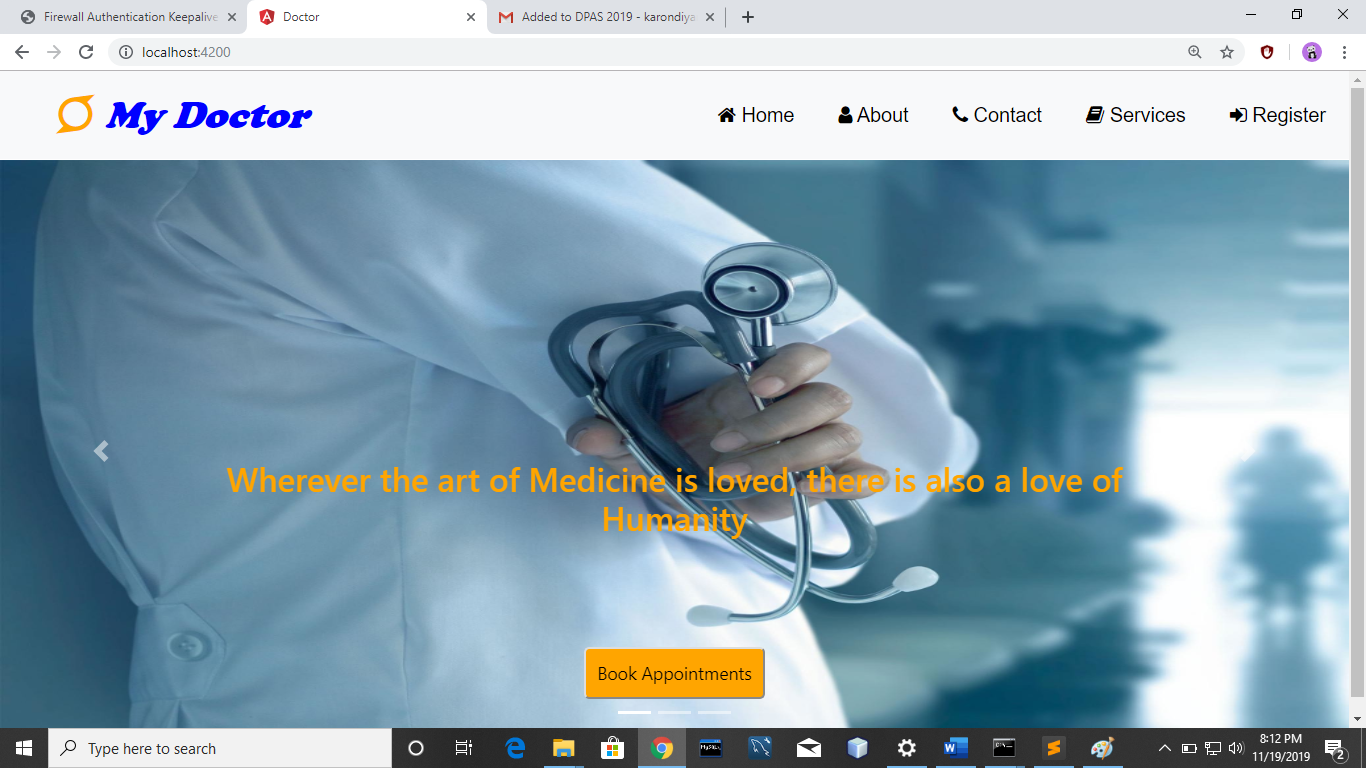
****

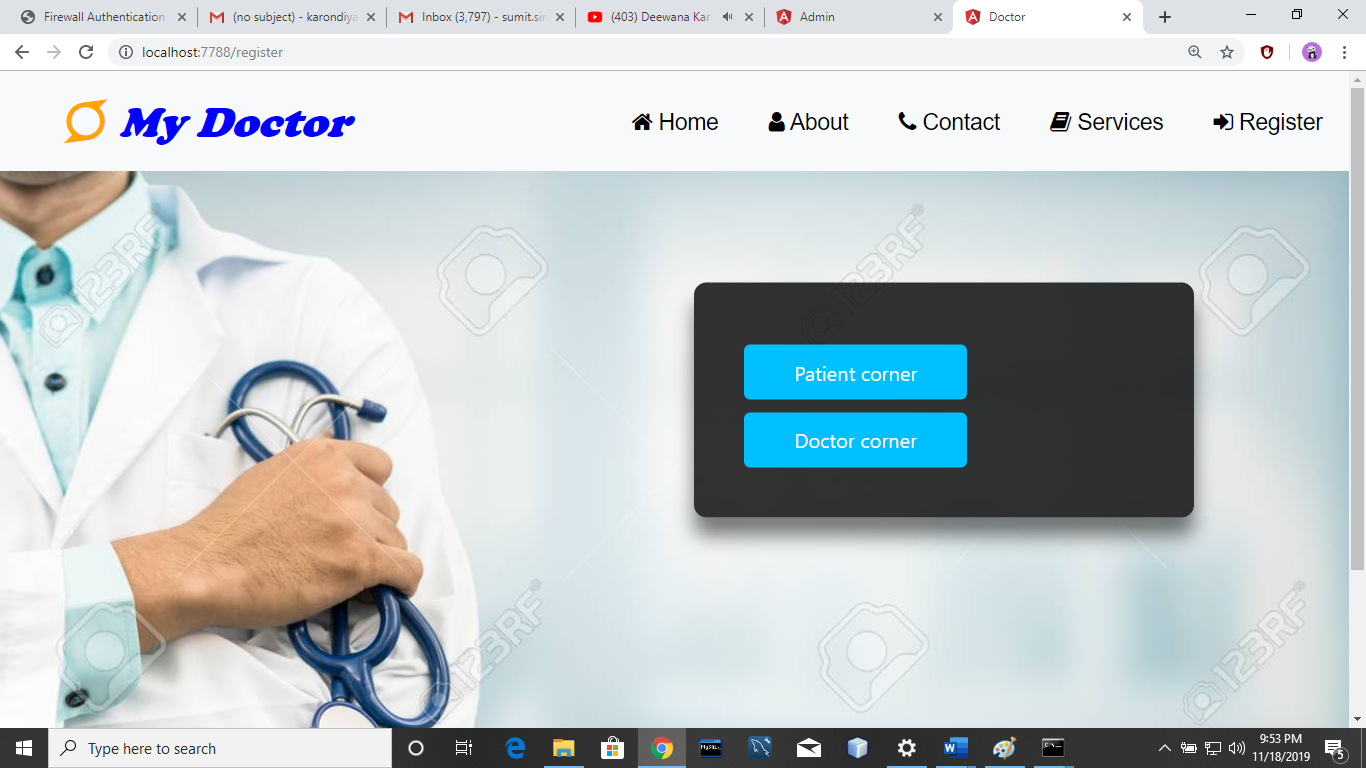
****

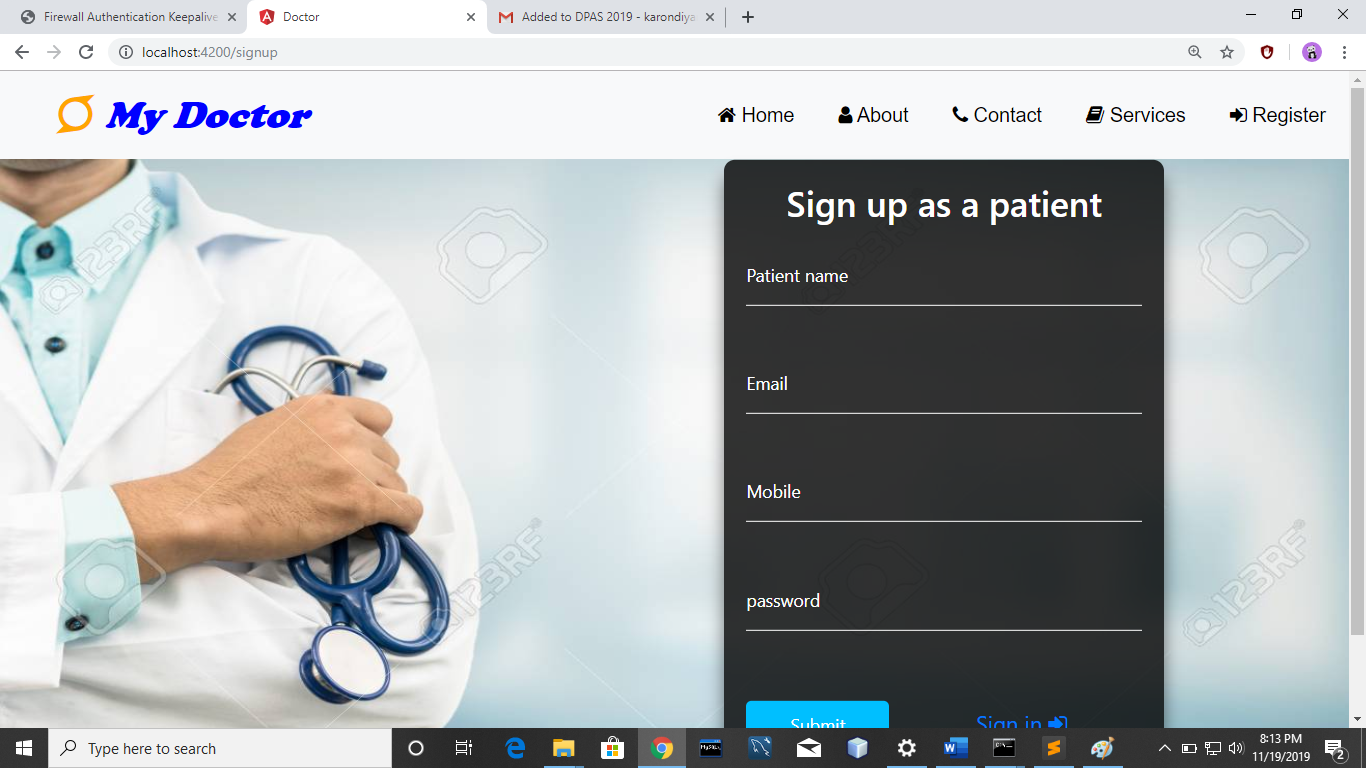
****

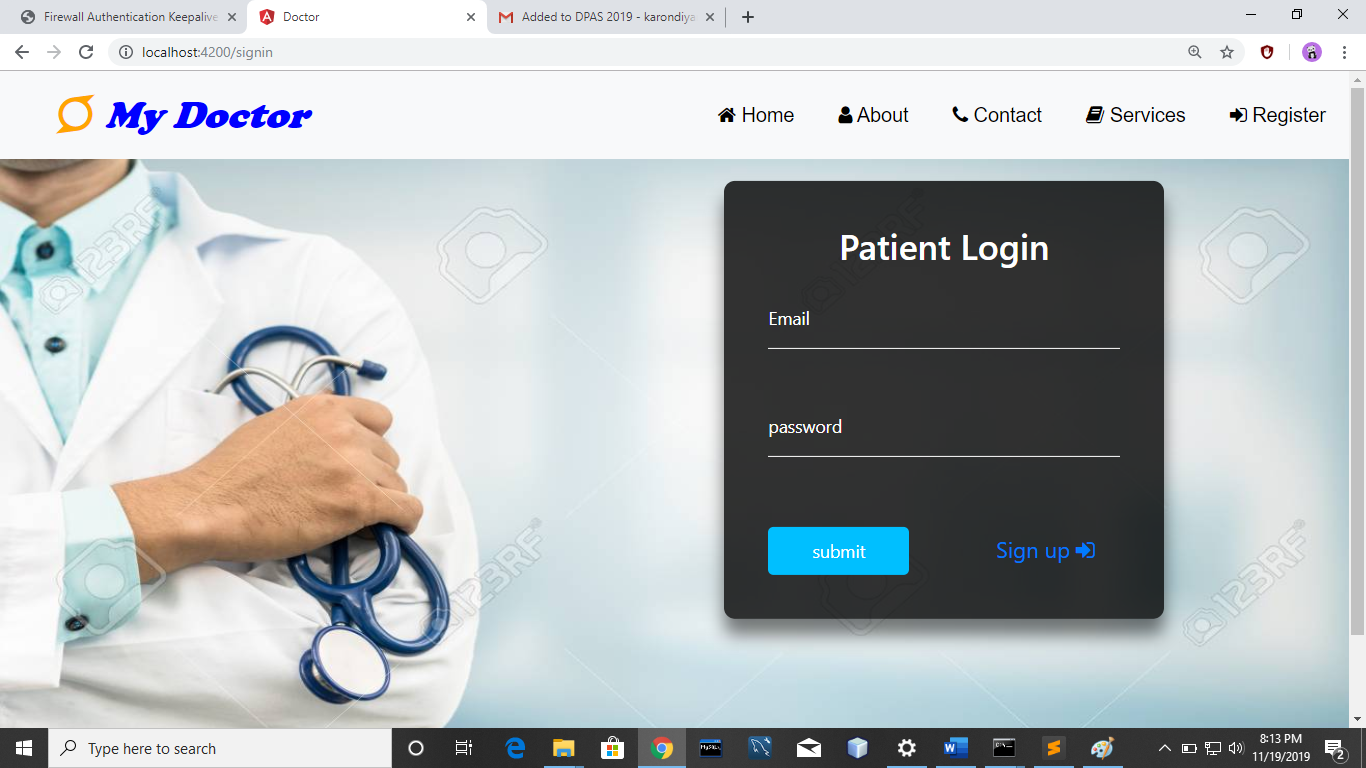
****

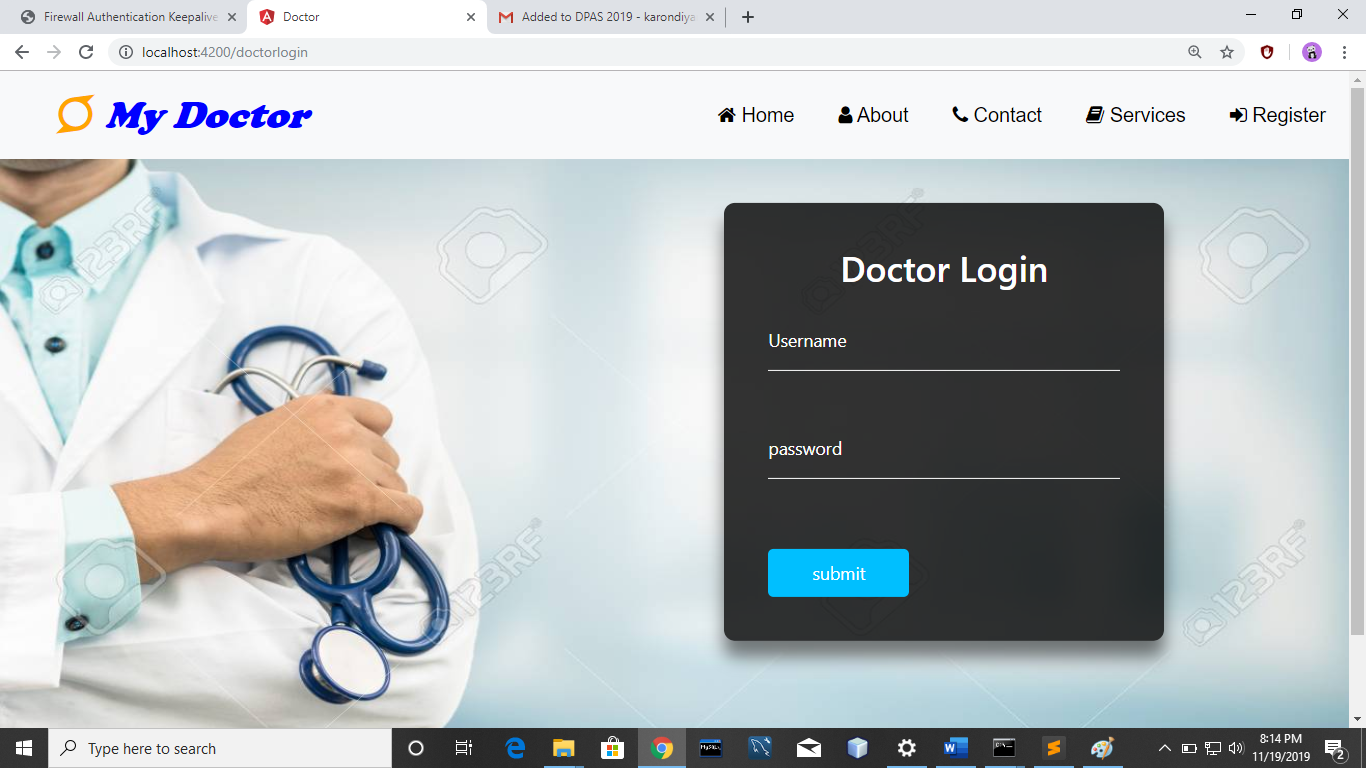
**7.2.Front Side :**

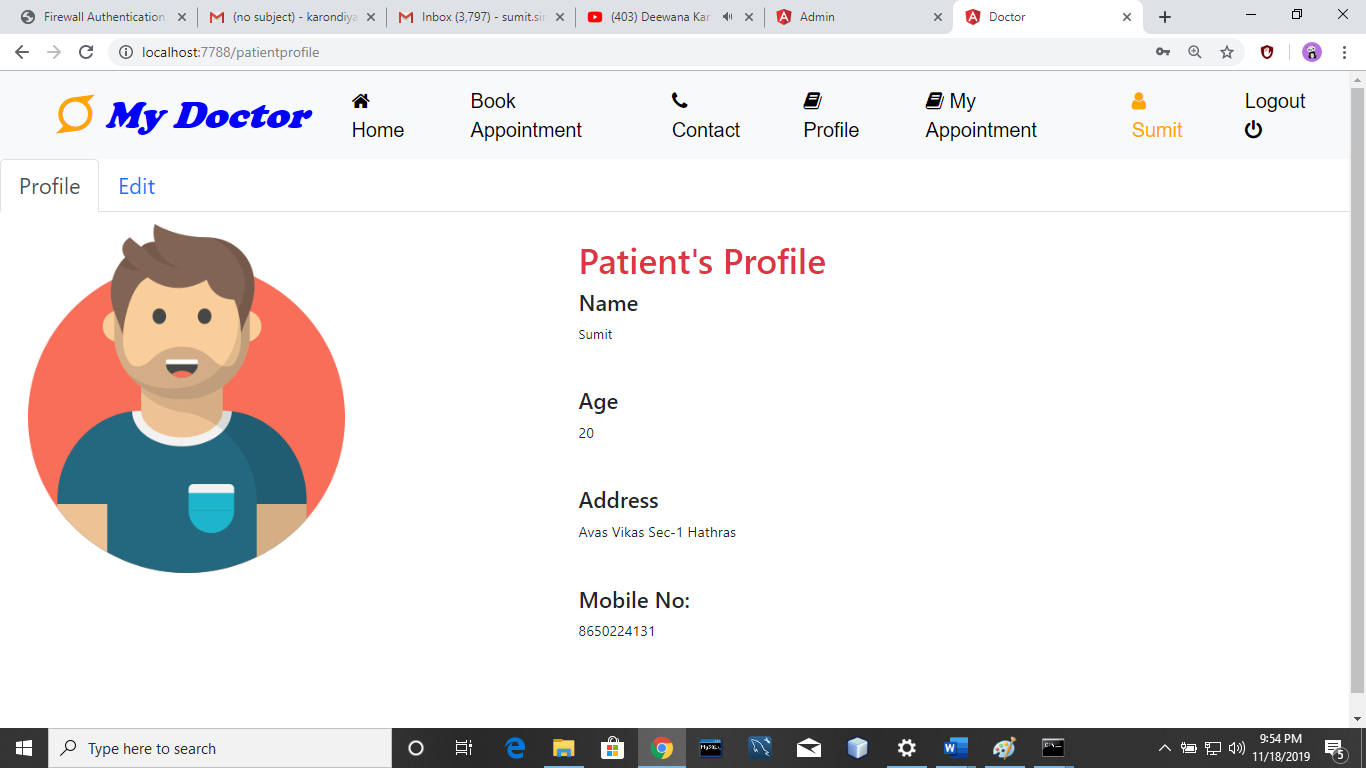
****

****

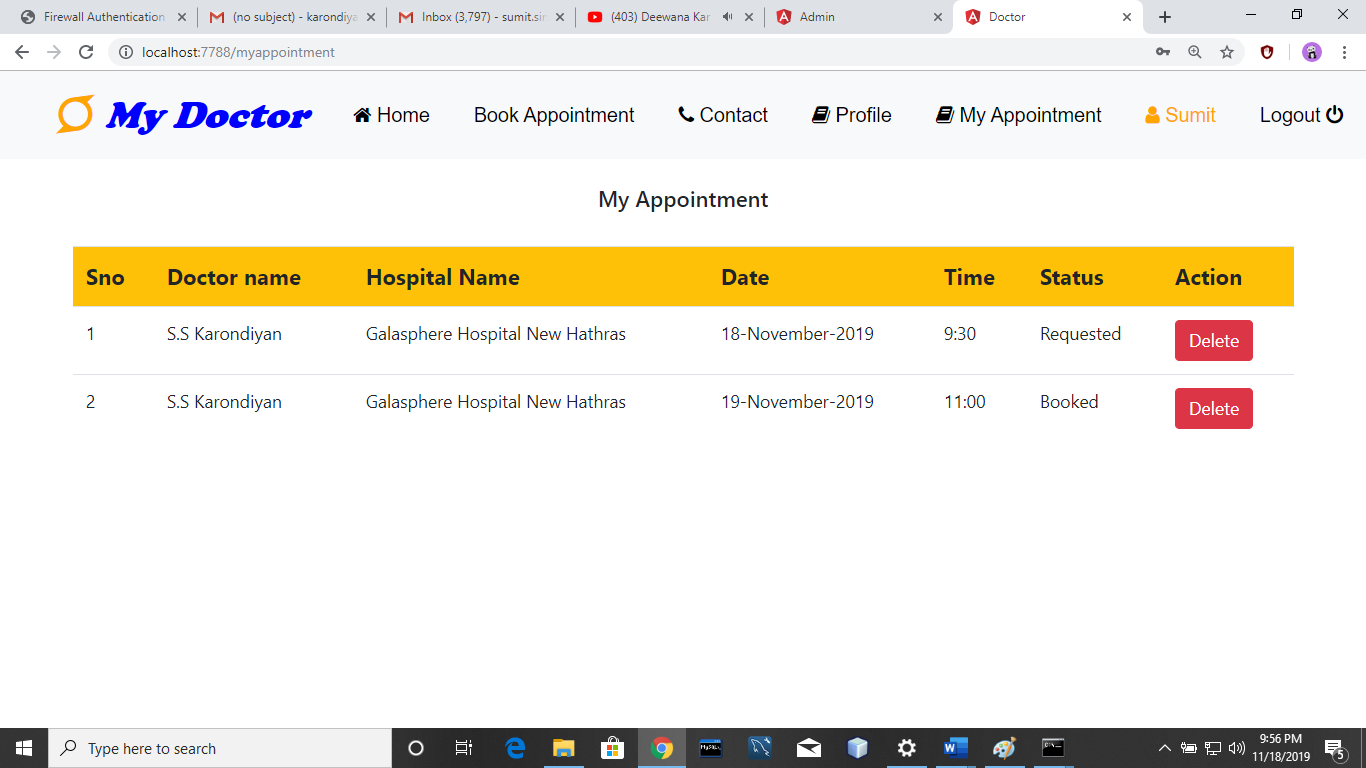
****

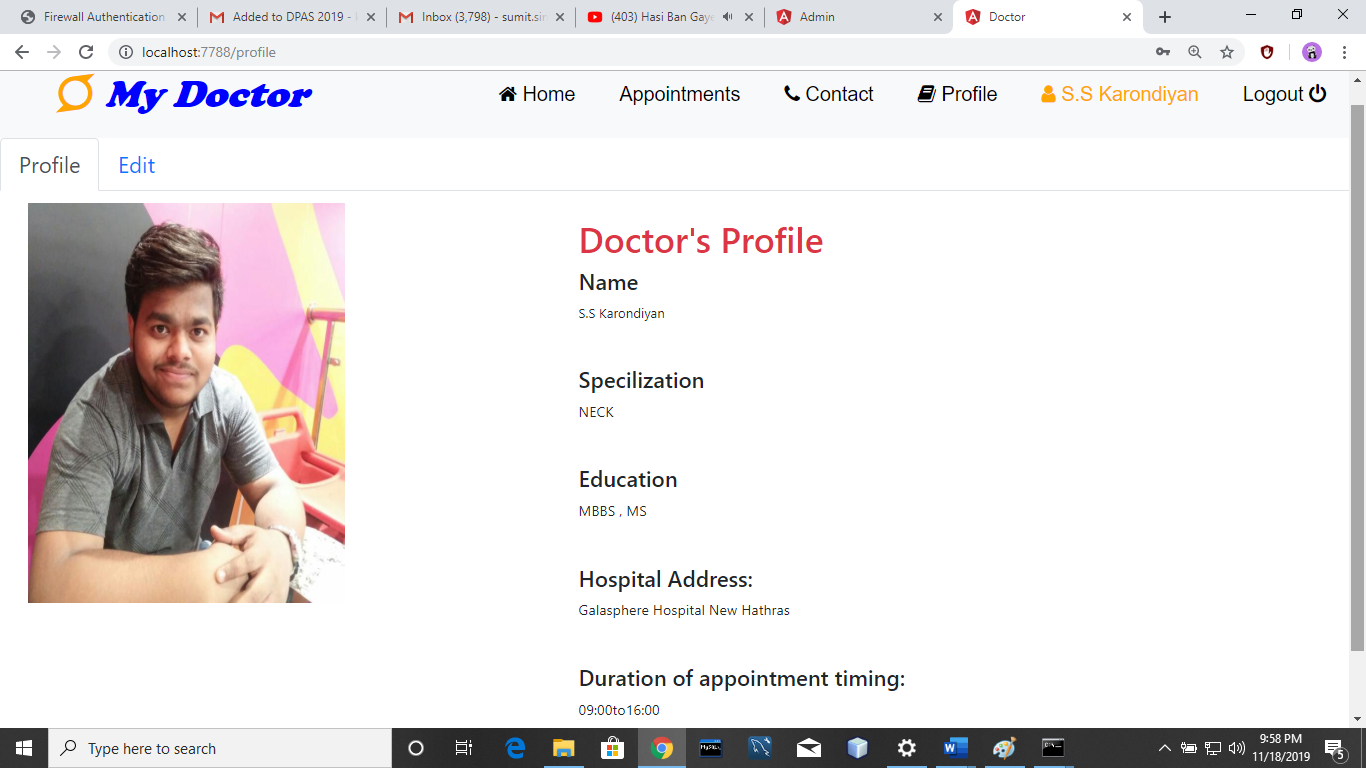


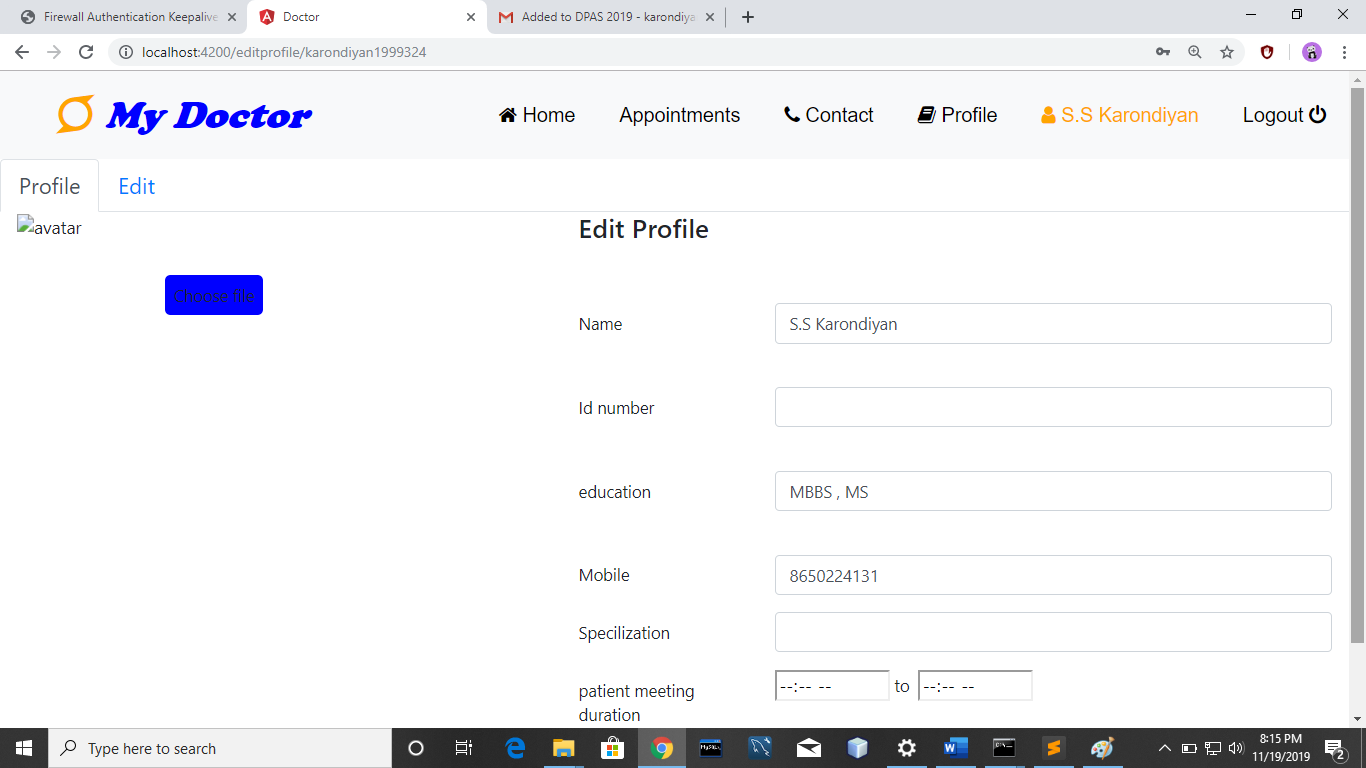


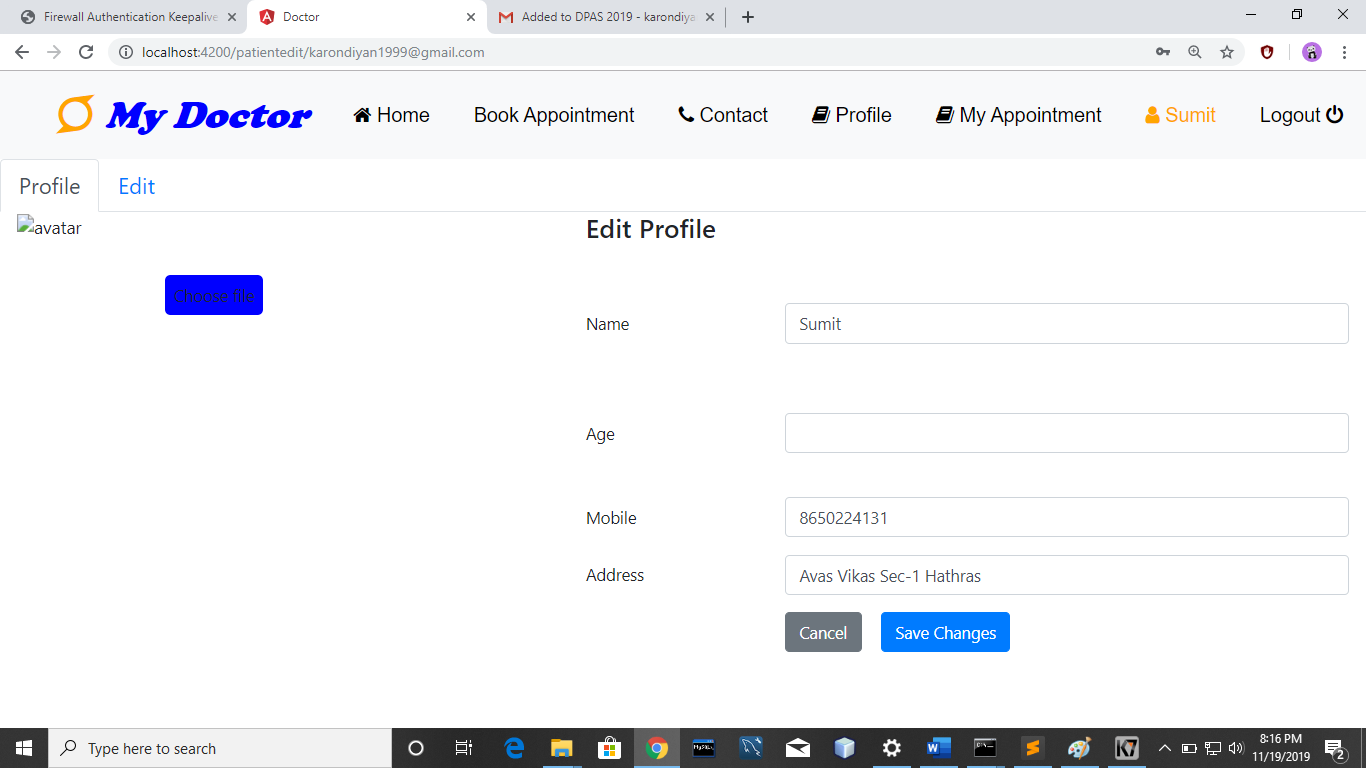
****

****

****



****

****

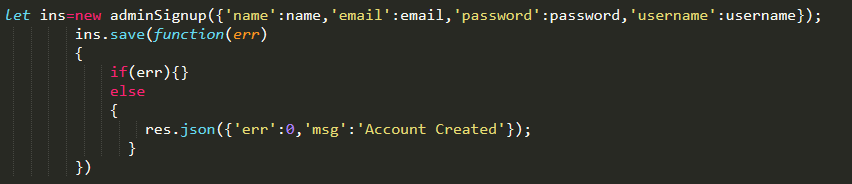
**API’S Working**

In our project we made the API’s for the backend connection. We made API’s for all the operations related to database like insertion,deletion, update. Here we are explaining all the API’s in detail.

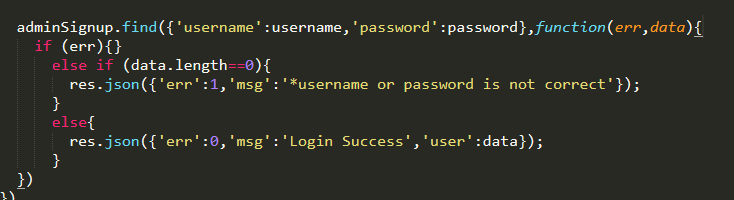
**1.SignUp :** This is for the creation of new account of user.

**app.post('/api/adminsignup',function(req,res)**

We fatched the details and insert into the adminSignup database using insert query of MongoDB database.

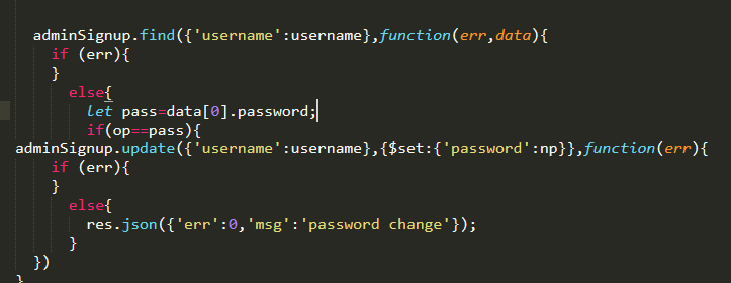


**2. Login** : This is for the login of user. In this we matched the user input value from database and functioned accordingly.

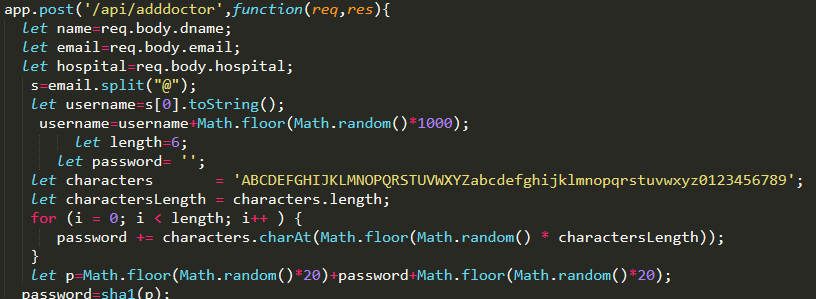
**app.post('/api/adminlogin',function(req,res) **

**3.change password :** This is for changing the current password f the user.

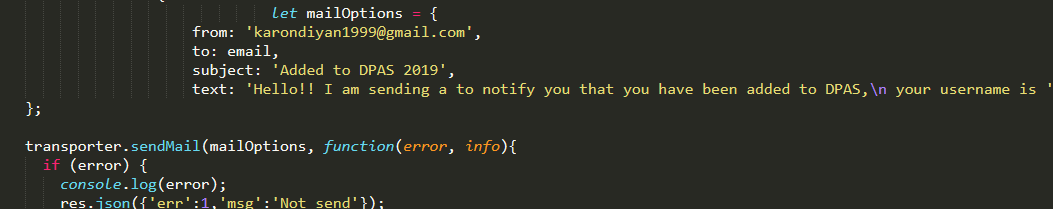
**app.post('/api/changepassword',function(req,res)**

****

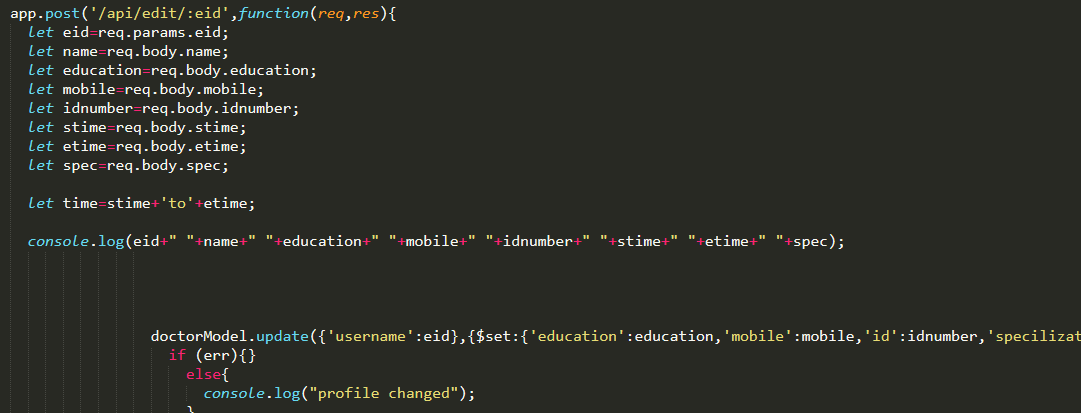
**4.Add Doctor :** This is used for by the admin for adding the doctor.We inserted a new value in doctor database. It auto genetrate the username and password for doctor.



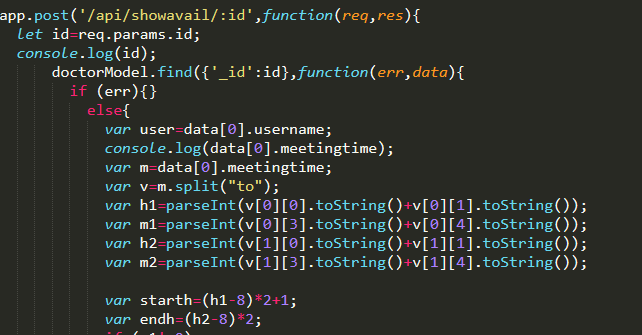
These username and password are sent to the doctor via email using nodemailer package of Node.

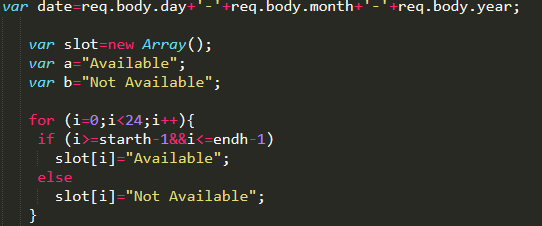


**5.Edit profile :**This API is used for the edit of doctor and patient profile. We fetched the data and update the existing database of doctor.

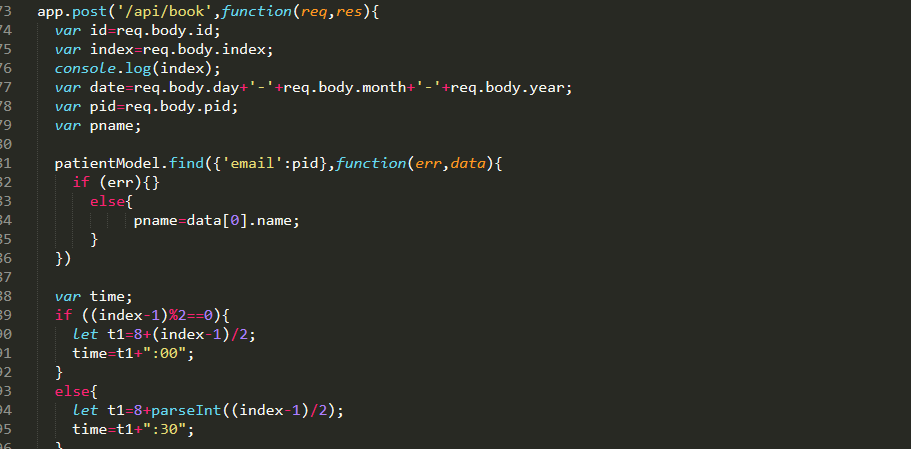
****

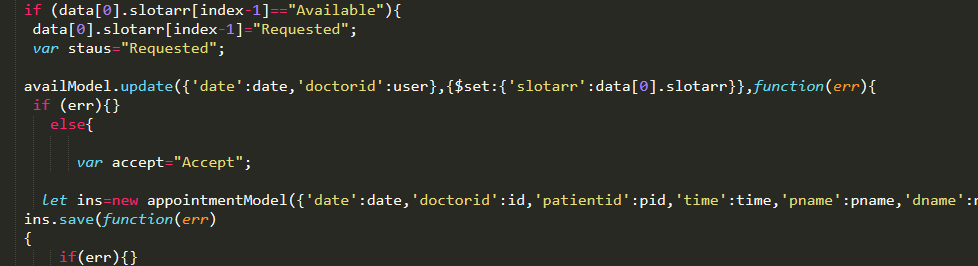
**6.Timing of doctor :** This API is used for the matching the timing of doctor. In this we made array of slots and set available and not available according to the timing of doctor.



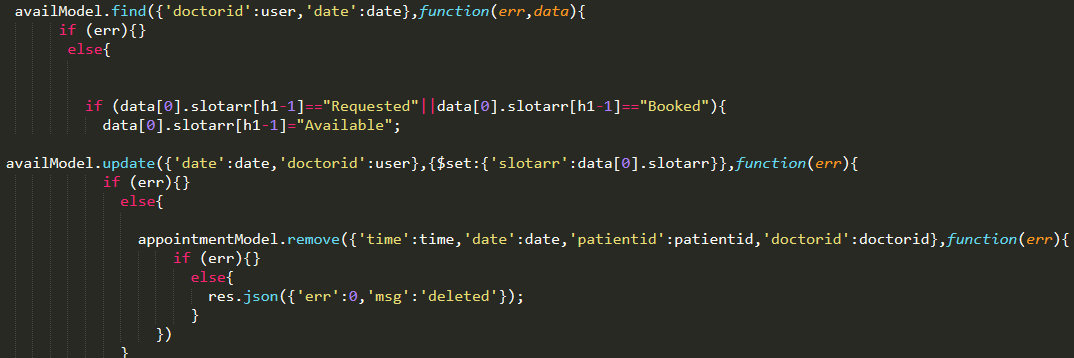


**7.Book Appointment :** This API is used to make appointment with doctor. In this we made a logic which is working properly and book appointment according to patient choice.





**8.Deleteion :** This API is used for the deletion of any record from database . This API is used for the cancelation of appointment.

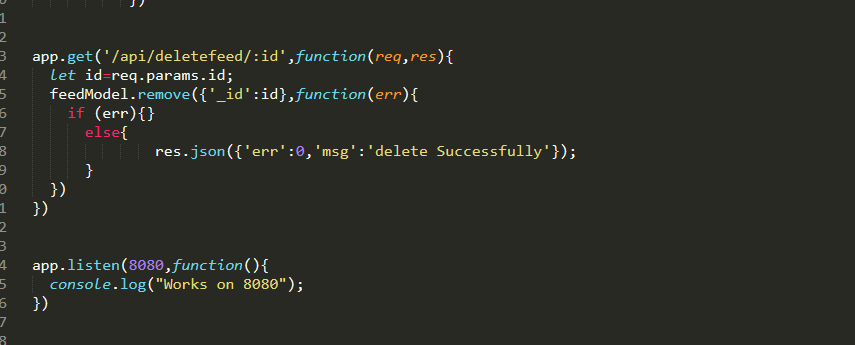


**9.Feedback API :** This API is used for to connect patients and doctors with admin. If any of the user have a issue then he can directly send a email to the admin.



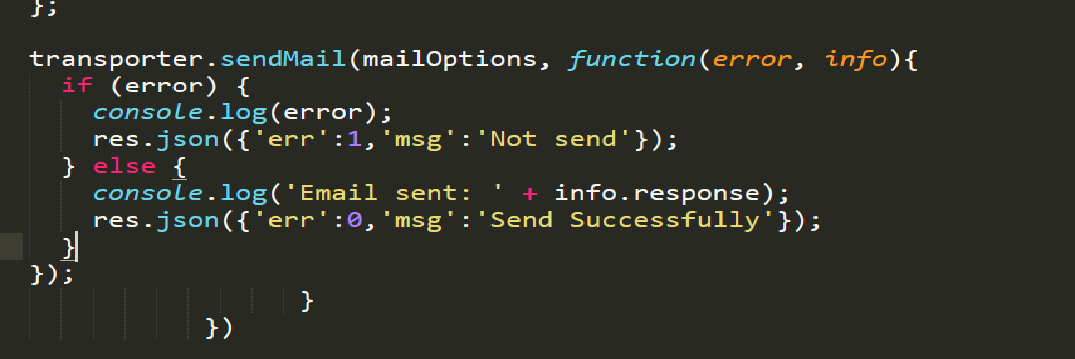
**10. Reply API:** This API is used by the admin for replay to a user to clarify his/her issues.





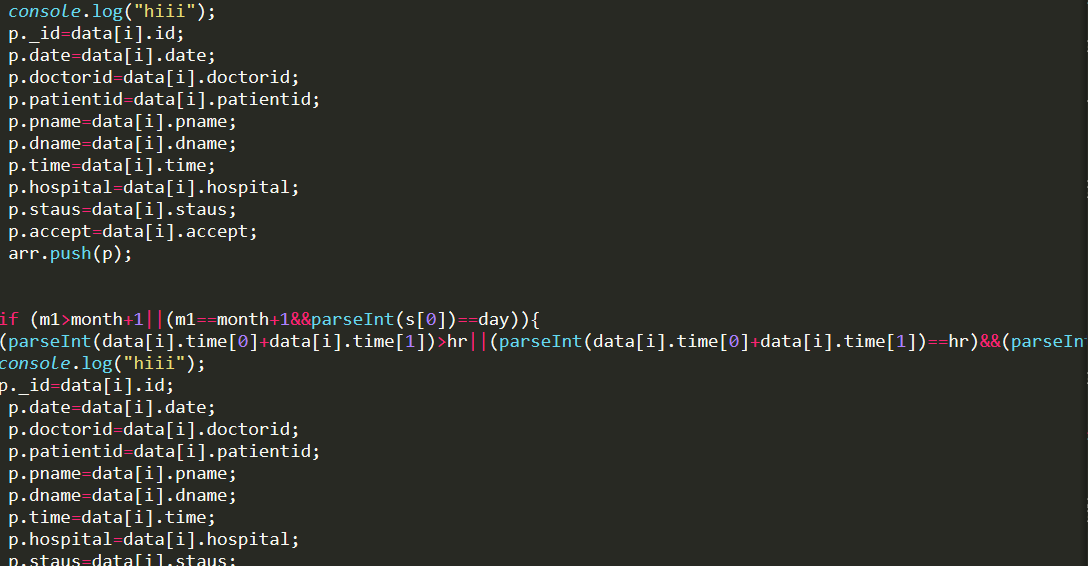
Node mailer working :





If doctor did not accept the request of patients and slot’s time is went off. Then we used this API.

Using this API only those appointment will store who are under their slot timing.



**Contributions Of group members**

Our group member SAURABH KUMAR contributed in this project  in the implementation of both fronted and backend. whole Doctor’s profile  front end has been  made by him by html, Bootstrap and Css. The backend related part of creation of schema of doctors and insertion of doctor’s various field in that schema has been done by him. He has made the validation code to check whether the patient logging in  website is present or not, if patient is not present the message of signing up will be displayed otherwise the patient get logged in and can see his/her profile. Same is the case with doctor while signing into the account. This task is related to sending the entered information on backend  and then check for the presence of entered data in particular collections using query and then send the response as per the response of query. While implementing frontend profile of doctor he designed functionalities for doctors to update his required details as per his wish. He implemented this task using update query by which the data stored in data base can be changed. When admin gives  id and password to doctor he/she can login into his/her account and there the doctor can change their profile details like timing , photo and other details as peer their choice. He also helped in implementation of group task of appointment management. He implemented the task of fetching of doctors when patient select the hospital. As the patient selects the hospital all the doctors present in that hospital are fetched from the database based on the name of the hospital and the fetched data is sent to the frontend.

Our Group member, YOGESH CHANDWANI had contributed in this project by implementing the front end of the main website like navbar , slider, patient and doctor login/signup portal. Also the static pages of our website like about us and contact us pages all these pages are implementing using the HTML5 ,CSS and Bootstrap. He also contributed in the backend of our project by implementing a functionality through which patient click on Book Appointment then the patient is redirecting to another page  from where he can select the particular date on which he wish to book an appointment and then select a particular hospital from all the hospitals which are available in the database then after clicking the show doctors button a list of doctors of the selected hospital fetched from the database and will be displayed on to the screen then patient select a particular doctor and patient redirect to another page. All this work is done using api files which are implemented in javascript   which is responsible for performing the operations like insert, update, delete on to the database. MongoDB queries are very useful for performing CRUD operations.

Our group member SUMIT SINGH had contribute in both frontend and backend. Mainly backend development of the project has done by him. He made the connection of Node to Angular in various API’s.. From this API’s we can transfer the data from one component to another component. He also made the logic for book appointment component which is most important part pf our project. We used Node for backend coding. He made the logic and the algorithms very sharply which are working properly. The code for the book appointment is very long and it took the major time portion of the total time. Sumit had done a great work in this backend section. Our feedback component is also made by him. From this feedback component patients as well as doctors can sent an email for clear his/her issues. The project is also contain the replay facility to the patient and doctors. This is also made by Sumit. He has given the help in the making of cancelation and acceptance of appointment which are useful for both patients as well as doctors. Doctor will accept the request sent by the patients and patients can cancel the request before accepting the request by the doctors. Sumit has also helped in the development of the profile update of the doctor and patients. The image of the doctors and patient are store in the form of string in MongoDB database. This method of uploading image in the database is good than any other method. This method is completed by him,Although he took the help of his friend Manish Chahar but he did it in very well manner.

At last we would like to say that this project is the output of all our team members (Sumit , Saurabh and Yogesh )’s hard work and dedication. We are thankful to each other that we made this project with good team work. We all know all the things that developed by us and we completed in very well manner. We have a great team and we wish that we will continue with this team in future as well.

**Future Scope**

This project is not fully completed yet. Doctor Appointment System has more than this.If we continue this project in future then we can add different functionalities in it.

1. We can assign a master card for patients .From that card patients and his/her family can know the history of patients.
2. This project currently does not have the facility of old appointments. It can be add later.
3. We can add on the emergency service and ambulance service also.
4. We can add the report section of patients in which doctor can attach the report of patients.

There are many facilities like above which can be provided in future.

**References**

This project is the output of work of our team member and guidance of our mentor. But there are some website which helped us a lot. They are :

1. [www.angular.io](http://www.angular.io)
2. [www.w3schools.com](http://www.w3schools.com)
3. [www.nodejs.org](http://www.nodejs.org)
4. [www.monogodb.com](http://www.monogodb.com)
5. [www.youtube.com](http://www.youtube.com)