**OCTOR APPOINTMENT SYSTEM**

**Documentation Report**

**SUBMITTED BY**

**<Your-Teacher-Name>**

**ADVISED BY**

**<Teacher Name>**

**A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF**

*BECHLOR OF SCIENCE*

*IN*

*COMPUTER SCIENCE*

**DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF SCIENCES**

**UNIVERSITY OF AGRICULTURE FAISALABAD**

**DECLARATION**

I hereby declare that the contents of the report [“**Doctor Appointment System**”] are project of my own research and no part has been copied from any published source (except the references). I further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default the scholar will be proceeded against as per UAF policy.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CERTIFICATE**

To,

The Controller of Examinations,

University of Agriculture,

Faisalabad.

The supervisory committee certify that **Abdul Ahad, M. Abdullah , Toheed Ahmad, under these ag’s respectively 2016-ag-7749, 2016-ag-7708, 2016-ag-7702** has successfully completed his project in partial fulfillment of requirement for the degree of BS Computer Scienceunder our guidance and supervision.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Akmal Rehan

Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Abdul Ahad, M. Abdullah , Toheed Ahmad

Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Salman Asfar

Incharge,

Department of Computer Science

**ACKNOWLEDGEMENT**

I thank all who in one way or another contributed in the completion of this report. First, I thank to ALLAH ALMIGHTY, most magnificent and most merciful, for all his blessings. Then I am so grateful to the Department of Computer Science for making it possible for me to study here. My special and heartily thanks to my supervisor, *[Akmal Rehan]* who encouraged and directed me. His/her challenges brought this work towards a completion. It is with his/her supervision that this work came into existence. For any faults I take full responsibility. I am also deeply thankful to my informants. I want to acknowledge and appreciate their help and transparency during my research. I am also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, I also thank my family who encouraged me and prayed for me throughout the time of my research. May the Almighty God richly bless all of you

**ABSTRACT**

I’m going to develop a “Clinic Appointment

System” that covers all the aspects of making appointment of doctors. It enables healthcare

providers to improve operational effectiveness, reduce costs, reduce medical errors, reduce time

consumption and enhance delivery of quality of care.

This system help reduce the problems occur when using the manual system and helps

patients to skip endless queues. The important thing is it will become easier for the data record and

retrieval. This software also stores all the patient details, patients profile, prescriptions etc. This

system enables doctors and clinic assistant to manage patient records and appointments. User can

enter their details, update their profile and they can select doctors to make appointments. Other

than that, the system is user friendly and it can help the clinic to manage their appointments. The

system helps to avoid making duplicate appointments. Users can view available doctors and their

timings and can make appointments according to it. Users also get an option to cancel their

appointments. Users can view their upcoming appointments and past appointments are deleted

automatically.

The system also allows doctors to log in. Doctors can edit their profile and view their

upcoming appointments, patients etc. They can even send prescriptions to their patients by

selecting their patients from the dropdown list.

The system also has an administrator section, where only a single person can manage the

whole system. Administrator can add/remove patients, doctors and departments and search for

appointments.

The system features a very intuitive and responsive design that fits devices of all screen

size (Smartphones, PCs, Tablets, and Notebooks etc.). The application automatically adapts its

layout to match user’s screen size.

Table of Contents

[Chapter 1 - INTRODUCTION 5](#_Toc536663558)

[1.1 Background: 5](#_Toc536663559)

[1.2 Description: 5](#_Toc536663560)

[1.3 Problem Statement: 6](#_Toc536663561)

[1.4 Scope: 6](#_Toc536663562)

[1.5 Objectives: 6](#_Toc536663563)

[1.6 Feasibility: 6](#_Toc536663564)

[1.7 Requirements: 7](#_Toc536663565)

[1.7.1 Functional Requirements 7](#_Toc536663566)

[1.7.2 Non- Functional Requirements 8](#_Toc536663567)

[1.7.3 Hardware Requirements 8](#_Toc536663568)

[1.7.4 Software Requirements 8](#_Toc536663569)

[1.8 Stakeholders: 8](#_Toc536663570)-9

1.9 [Work break down structure: 10](#_Toc536663570)

[Chapter 2 – MATERIALS & METHODS 10](#_Toc536663571)

[2.1 Process Model: 10](#_Toc536663572)

[2.2 Tools & Technologies 10](#_Toc536663573)

[2.3 Design: 10](#_Toc536663574)

[2.3.1 Use Case Diagrams: 11](#_Toc536663575)-22

[2.3.2 Sequence Diagram: 2](#_Toc536663577)3

[2.3.3 Class Diagram: 23-25](#_Toc536663578)

[2.3.4 Context Diagram: 26](#_Toc536663578)

[2.3.5 Data Flow Diagram: 26](#_Toc536663579)

[2.3.5.1 Zero Level Diagram: 27](#_Toc536663579)

[2.3.6 ER Diagram: 27-28](#_Toc536663580)

[2.3.7 Database Model: 29](#_Toc536663581)

[2.3.8 Architecture: 29-31](#_Toc536663582)

[Chapter 3 - RESULTS & DISCUSSION 32](#_Toc536663583)

[3.1 Testing: 32](#_Toc536663584)

[3.2 Test Cases: 33](#_Toc536663585)

[3.3 Conclusion: 34](#_Toc536663587)

[Chapter 4 – Application Screenshots 35](#_Toc536663588)-41

List of Figures

[Figure ‎1.1 Stakeholders 9](#_Toc536625484)

[Figure ‎1.2 Work break down structure 10](#_Toc536625485)

[Figure ‎2.1 Agile Activities 10](#_Toc536625485)

[Figure 2.‎2 Use Case Diagram 13](#_Toc536625486)

[Figure 2.‎3 Use Case Diagram for Patient 14](#_Toc536625486)

[Figure 2.‎4 Use Case Diagram for Doctor 17](#_Toc536625486)

[Figure 2.‎5 Use Case Diagram for Admin 21](#_Toc536625486)

[Figure 2.6 Sequence Diagram 23](#_Toc536625487)

[Figure 2.7 Class Diagram 25](#_Toc536625488)

[Figure ‎2.8 Context Diagram 26](#_Toc536625489)

[Figure ‎2.9 Level 0 DFD 27](#_Toc536625490)

[Figure ‎2.10 Entity Relationship Diagram 28](#_Toc536625492)

[Figure 2.11 Database Model 29](#_Toc536625493)

[Figure ‎2.12 Applications's Architecture 31](#_Toc536625494)

[Figure ‎4.1 Home page 35](#_Toc536625495)

Figure ‎4.2 Navbar 35

Figure ‎4.3 Login page 36

[Figure ‎4.4 Signup page 37](#_Toc536625495)

[Figure ‎4.5 Categories Features 38](#_Toc536625495)

[Figure ‎4.6 All Doctor 38](#_Toc536625495)

[Figure ‎4.7 Patient Dashboard 39](#_Toc536625495)

[Figure ‎4.8 Patient Requesting for Appoitnment 40](#_Toc536625495)

[Figure ‎4.9 Doctor Dashboard 41](#_Toc536625495)

[Figure ‎4.10 Mail Confirmation 41](#_Toc536625495)

[Figure ‎4.11 Database 41](#_Toc536625495)

List of Tables

[Table 3. 1: User login Test Case 33](#_Toc536658805)

# CHAPTER 1 - INTRODUCTION

## 1.1 Background:

A Faisalabad based hospital was facing the problem of hiring one secretary for each doctor to manage his/her appointments. What they needed was a software that can manage there doctoring staff and there appointments. That way they can save a lot of human hours and money. As well as system would provide there patients an easy way to book appointments rather than waiting in line.

## 1.2 Description:

## Health care is one of the fastest growing industry all over the world. Before the last few years, medical appointments were usually taken on the phone calls or by visiting the hospitals in person. This process needed the involvement of individuals so, the ability to take appointment was restricted to the availability of schedulers, phone lines or the physical presence of a person. With the growth of time, everybody demanded timeless and efficient medical care delivery because manual appointments (that requires the physical presence of both individuals) and long waiting lines have formed an irritating situation for the healthcare institutions. So, it created a need for such an integrated health care system that could deliver seamless care to both outpatients as well as inpatients. The emergence of online appointment system offered timeless and efficient access to health care services. Therefore, for hospitals and other medical societies, online appointment booking has a great importance and a subject of interest (Koole, 2007). Booking appointment online has become a new trend in the past few years and is considered as one of the key processes in the healthcare industry. Bailey (1952) considered scheduling system as a trade-off or a compromise between a doctor and patient’s waiting times. Patients who get late for the appointments or who fails to come becomes the reason for the underutilization of a doctor’s time. Idle time and underutilization of doctor’s time are also resulted by gaps in the appointment times (Bailey, 1954). Different researchers agreed that main patient dissatisfaction is caused by long waiting times. Cayirli (2003) defined access time as the time between patients’ request for the appointment and the time he is checked up. According to Veral, waiting time is the time between consultation and the scheduled time while neglecting the early arrival of a patient (Veral, 2003). Different researchers defined waiting /access time in different ways. A well-designed appointment system supposed to improve patients’ satisfaction by reducing cost and time of clinics and hospitals especially in the busy lives we are leading today. 7 With the growing population need for more efficient ways to access a medical treatment is also growing. Through an online appointment scheduling system, a user gets access to the doctor's online webpage and can make an appointment with online software. Patient/user can also provide additional medical history in advance, giving adequate time to the doctor to prepare the necessary information for consultation. In this way, online appointment scheduling systems are helping doctors and the patients and making the healthcare delivery efficient. Nowadays there are many kinds of online appointment tools available in the market which are easy to set up and not too much expensive. Online scheduling system offers value added services and lots of benefits to the doctors and patients. It makes the patient appreciated by eliminating the hassle of long waiting times. Online appointment systems are also getting popular because of its low-cost availability.

## 1.3 Problem Statement:

## The reason behind creating this system was the “trend of private medical clinics” and manual medical file keeping system in Pakistan. Online appointment and database management system aims to improve quality medical care by bringing all medical clinics of the city at one platform, eliminating long waiting lines and replacing manual medical file keeping with an online database.

## 1.4 Scope:

This system is implemented for all the individuals who want to get treated by the city practitioners. The users can participate only if they have created an account through the registration form and have provided their medical history. Once they get registered themselves further they would not need to update their record as it would be done automatically after each doctor’s visit.

## 1.5 Objectives:

## The main objective of the thesis is to provide quality medical care to the patients by making the hospital appointment system online, so that they can choose the date and time of there choosing for appointment. The second objective is to replace the current manual file keeping system with an online medical patient database system.

## 1.6 Feasibility:

The project we are going to build is easy to manage and cheap to implement. As it does not require more than 2 persons to manage the system. Every doctor and user will perform his/her actions himself.

1.6.1 Technical Feasibility – Technical resources being used in the project are easy to use and cheap to use.

1.6.2 Schedule Feasibility – All the necessary resources are available. Project will be complete by the given deadline.

1.6.3 Economic Feasibility – As project is going to be developed by a small team and all the resources being used are cheap. So, the project is economically faesible.

1.6.4 Cultural Feasibility – Most of the people in cities have access to smart phones and internet. So, the system is implementable.

1.6.5 Legal/Ethical Feasibility – There is no legal or ethical restriction in making this system.

1.6.6 Resource Feasibility – Yes, hospital has enough resources to back the development of this system.

1.6.7 Operational Feasibility – There are tones of help on internet about every difficulty we feel on the course of development. So, I hope, this will be easy to build this project.

## Requirements:

### 1.7.1 Functional Requirements

This system is going to register new users, doctors and administrators to to facilitate the appointment booking easily and more precisely.

**FR01:** Provide user name and password to log in

|  |  |
| --- | --- |
| FR01-01 | System shall get Username and Password from patient, admin and doctors. |
| FR01-02 | System should authenticate users name and password |
| FR01-03 | System shall let the user to log in if information is valid |
| FR01-04 | If information is not valid then system will display error message and redirect back to the login page. |

**FR02:** Create admin account

|  |  |
| --- | --- |
| FR02-01 | System shall allow a user to sign up and login as administrator. |
| FR02-02 | System shall collect necessary details in this regard. |

**FR03:** Create patient account

|  |  |
| --- | --- |
| FR03-01 | System shall allow a patient to create an new account. |
| FR03-02 | System shall collect necessary details in this regard. |

**FR04:** Add new doctor

|  |  |
| --- | --- |
| FR04-01 | System shall allow admin to create accounts for doctors. |
| FR04-02 | System shall collect necessary details in this regard. |

### 1.7.2 Non- Functional Requirements

System shall implement all the latest up-to date technologies for enhancing security and user experience with the application so that every user can interact with application easily.

**NFR01:** System shall remain available 24/7 to its users.

**NFR02:** System shall have three types of users i.e.patient, admin and doctor.

**NFR02:** System shall provide easy navigation between pages.

### 

### 1.7.3 Hardware Requirements

Processor: Intel Pentium

Monitor size: 14 inch

RAM: 256MB

Hard Disk: 80GB

### 1.7.4 Software Requirements

Technology : Web Application.

Front end : HTML, Java script , React, Bootstrap & CSS.

Back end : Node js(express) , Mongo db.

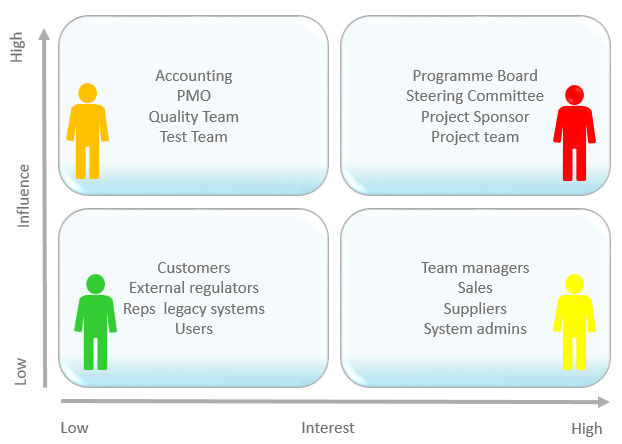
Web browser : Mozilla Firefox, Google Chrome, Apple Safari or any Gecko/Web Kit based

browser.

## Stakeholders:

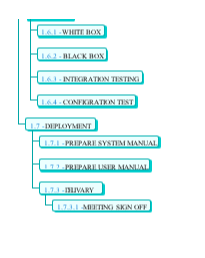
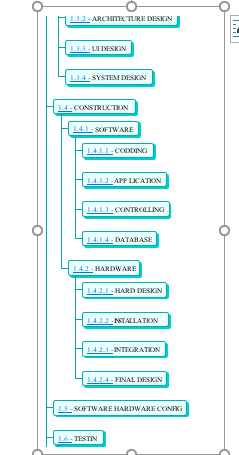
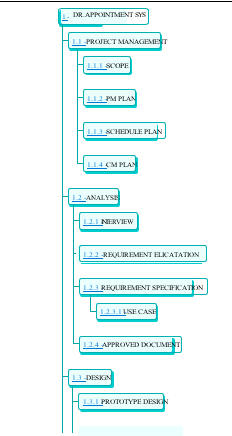
Stakeholders are different people who would be interested in the software or Stakeholders are all those with an interest or role in the project or who are impacted by the project.

We need to understand that it is the actual user who will eventually use the system and hence accept or reject the product. Therefore, ignoring the needs of any user class may result in the system failure. Below picture depicts involvement level of different categories of stakeholders. Identify your project’s stakeholders carefully.

****

*Figure 1.1 Stakeholders*

* 1. **Work Break Down Structure:**

****

*Figure 1.2 Work break down structure*

# Chapter 2 – MATERIALS & METHODS

## 2.1 Process Model:

We are going to use Agile model in this project. The **Agile model** adopts Iterative **development**. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers.

*Figure 2.1 Agile Activities*

## 2.2 Tools & Technologies

Following tools will be used in this Web based Appointment System.

HTML: used for basic document structure.

JAVASCRIPT,REACT: used API calls and Front-End user interactions.

CSS: used for styling the documents.

NODEJS(EXPRESS): used for developing Back-End of my Project.

MONGO DB: for database needs.

HEROKU: A hosting service we are going to use for our app deployment.

## 2.3 Design:

Following are some software design diagrams.

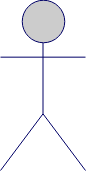
### 2.3.1 Use Case Diagrams:

A use case is a functionality the users need from the system. A use case diagram depicts the relationships among the actors and use cases. Generally, a single use case is supposed to cover all the actions or events that an actor can perform on the system at one go. The size of use case should not be very large or very small. For example, *Add User, Manage Profile, View Sales Report, Update Order* etc. are good medium size use cases. Whereas *Enter Password, Display Error Message* etc. are very small use cases and *Manage Sale & Purchase* is a very large use case.

The components in a use case diagram include:

Actors:

Actors are first thing you need to find for the use case diagram. Actors represent external entities of the system. These can be people or things, such as external hardware that interact with the system. For example, if an online store is being modeled there can be more than one actor that interacts with the store functionality. Such as the Customer and stocker will be the actors in the system. It is represented simply by a stick figure with its name at the bottom of it.



*Actor*

Use Cases:

Use cases are functional parts of the system. They figure out what actions/functionalities a user will perform. Use cases are basically the functional requirements that you have pointed out in the functional and non-functional requirements topic. For example: The customer "browses the catalog", "chooses items to buy", and "pays for the items". Here browse catalog, buy item and pay for item are the use cases. Many actors can share a single use cases. The notation for a use case is an ellipse. As it is displayed below:



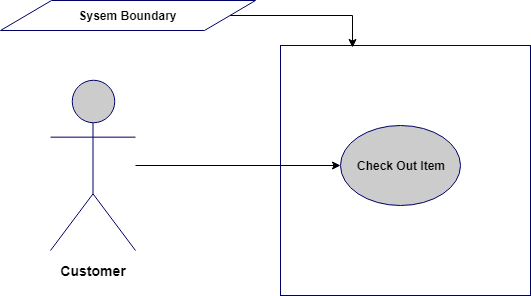
Associations:

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever there is direct interaction between actor and use case.   Associations are modeled as lines connecting use cases and actors to one another, with an arrowhead on one end of the line.



System boundary:

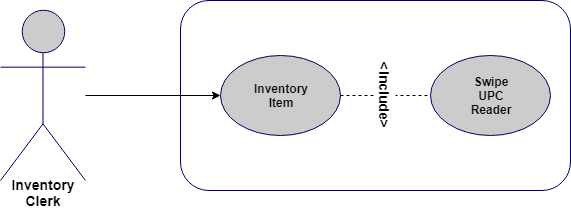
System’s boundary is drawn by a rectangle that contains use cases. The actors are placed outside the system boundary and use cases inside it.



**Relationship between Use cases:**

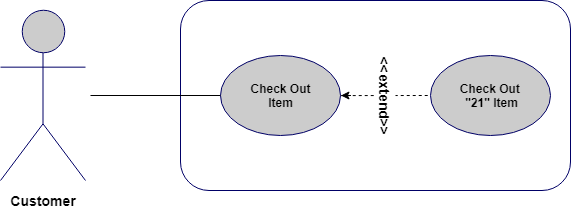
1. **Include/Uses:**

Include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). <<include>> use cases must be **used** by the use cases that **use** them before the latter can be complete. It is displayed in the diagram editor as a dashed line with an open arrow pointing from the base use case to the inclusion use case. The keyword «include» is attached to the connector.

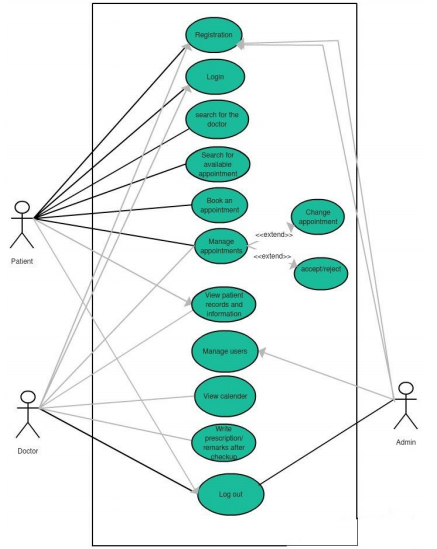


1. **Extend:**

A use case **extends** another use case to do more than the latter. It extends the functionality of one use case to further level. is displayed in the diagram editor as a dashed line with an open arrowhead pointing from the extension use case to the base use case. The arrow is labeled with the keyword «extend».

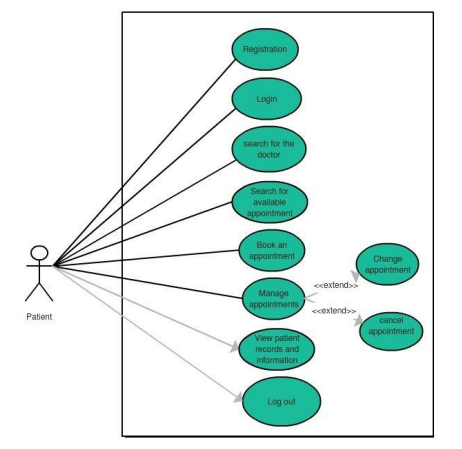


**Use Case Diagram:**



*Figure 2.2 Use Case Diagram*

**Patient Use Case:**

****

*Figure 2.3 Use Case Diagram for Patient*

**Use-Case 1: Registration**

**Primary-actor:** generic user/patient.

**Description:** To make an appointment, user registration is required.

**Precondition:** National ID card and valid email address.

**Basic use-case flow:** For the registration process, the user/patient needs to give some information by filling the form. Some of the required information includes the following:

First name

Last name

Gender

Email address

National ID no.

**Main scenario:** The user/patient will go onto the patient sign-up button either from the main page or from the drop-down menu. After clicking the sign-up button, a registration form will appear, where the user must give his personal information i.e. name, gender, email address, CNIC, any medical history etc. After giving the required information user will submit the form. If all the fields are filled including the valid email address and CNIC, the user will be registered onto the system. In case of any missing entry or invalid format of an email or CNIC, the error occurs onto the page.

**Exception:** Expired/invalid National ID card no or email address.

**Use-case 2:** Login

**Primary-actor:** Patient/user.

**Description:** Before taking any appointment or get access to his medical record, the user must have to provide his username and password.

**Precondition:** the user must have a valid username and password.

**Basic use-case flow:** a valid username with a password must be entered by the user.

**Main scenario:** To be able to get into the system, the user needs to enter his username and password either from the main page or from the drop-down menu from the top of the page. After clicking onto the login button, authentication request will be forwarded to the system.

**Exception:** wrong/invalid entered username or password.

**Use-case 3:** search for a doctor

**Primary-actor:** User patient/generic user

**Description:** In this use-case, any registered or unregistered user, can look and search for a doctor of any desired specialty.

**Precondition:** this use-case has no exception.

**Basic use-case flow:** By going into the category of any specific specialty, the user can view the list of all registered doctors.

**Main scenario:** The user will go to the down menu at the top of the page. A list of all specialties will appear in a drop-down list. The user will hit on the desired specialty. After that, a page will open with all doctors of that specific specialty. Now the user has the choice to choose any doctor based on qualification, experience, location etc.

**Exception:** no exception for this use-case.

**Use-case 4:** Take Appointment

**Primary-actor:** User/patient

**Description:** After choosing a doctor user will go further to send an appointment request from the available timings.

**Precondition:** the user must be login

**Basic use-case flow:** The patient/user hits the button for taking an appointment for the doctor. A list of available timings will appear for the chosen date. The user will select the suitable time for him and send the request for approval.

**Main scenario:** The user will hit the button for "take appointment". List of available timings will appear for a chosen date. The user selects the suitable time. The user will hit the submit button to send the request for approval.

**Exception:** this use-case has no exception.

**Use-case 5:** View history

**Primary-actor:** User/patient

**Description:** the user can view his medical history.

**Precondition:** the user must be signed in.

**Basic use-case flow:** The user/patient will click on the name of the patient and it opens the patient's profile. By going into it the user can view the history by clicking on the button for the patient’s old reports/history.

**Main scenario:** The user clicks on the name of the patient and then press the button “view history” to view old reports.

**Exception:** no exception for this use-case.

**Use-case 6:** Log out

**Primary-actor:** User/patient

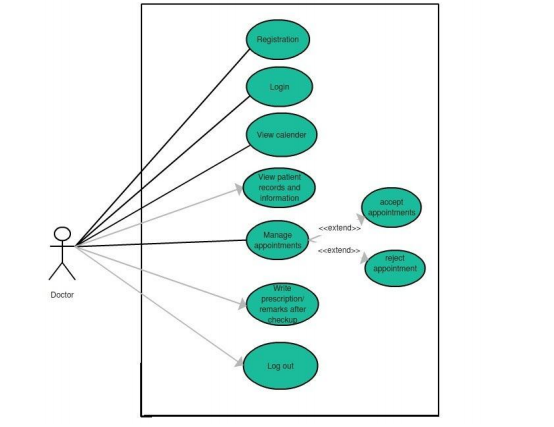
**Description:** the user will log out from the system.

**Precondition:** the user must be logged in Basic use-case flow: the user can sign out himself from the system.

**Main scenario:** the user clicks on the log out button. The system will bring the user to the main page for the generic user.

**Exception:** no exception for this use-case.

### Doctor’s Use Case:



*Figure 2.4 Use Case Diagram for Doctor*

**Use-case 1:** Registration

**Primary-actor:** Generic user/doctor

**Description:** To get online appointment requests, the doctor must register himself as a user on the application/system.

**Precondition:** National ID card and active email address.

**Basic use-case flow:** To register as a doctor, the user must fill the form. Required fields include the following information:

Full name

Email address

National identity number

Qualification

Experience

Valid PMDC certificate (Practicing license issued by the PMDC authority)

Clinic address

Clinic timings

**Main scenario:** The user/ doctor has to go onto the doctor’s sign-up button either from the drop-down menu at the top or from the main page. After clicking the sign-up button, a registration form will appear, where doctor/user has to give his personal as well as professional information i.e. name, clinic address, qualification, working experience etc. The user must give valid PMDC certificate for the registration process. After giving the required information user/ doctor will submit the form. If all the fields are filled, request for registration will be sent to the admin. As soon as admin receives a request for a doctor's registration he will verify his documents and only then the user will be accepted /registered. In case of any missing entry or invalid format of email or CNIC, an error occurs onto the page. PMDC certificate is a license or a proof that an individual is allowed by the authority to practice medicine. Any user fails to give valid PMDC certificate issued by PMDC authority will not be allowed to register as a doctor. PMDC certificate is important to avoid any scam.

**Exception:** False documents, expired or invalid PMDC certificate, missing fields, expired national ID card or inactive email.

**Use-case 2:** log in

**Primary-actor:** User/doctor

**Description:** For further functions, the user must have to provide his email address and password.

**Precondition:** the user must enter the username and password.

**Basic use-case flow:** user/doctor should provide his username and password to log in.

**Main scenario:** To be able to get into the system, the user needs to enter his username and password either from the main page or from the drop-down menu from the top of the page. After clicking onto the login button, an authentication request is forwarded to the system.

**Exception:** occurs if fails to provide username and password.

**Use-case 3:** View calendar

**Primary-actor:** User/doctor

**Description:** User would be able to view his calendar.

**Precondition:** The user must be signed in.

**Basic use-case flow:** After logging in, the user/patient selects the date from the calendar to filter out the appointment.

**Main scenario:** After signing in, the user/doctor can view the page of his calendar. The user will select the month and date to see the appointments of that period or date. The user can view all the patient's appointments of any date.

**Exception:** no exception for this use case.

**Use-case 4:** Accept or reject a request

**Primary-actor:** User/doctor

**Description:** User/doctor can accept or reject any patient's request.

**Precondition**: User/doctor must be signed in.

**Basic use-case flow**: User/doctor selects any date and decides to accept or reject any appointment request.

**Main scenario**: After logging in, all the patient's appointment requests will be appearing on the calendar according to the dates. User /Doctor will select the date from the calendar. User /Doctor can view all the appointment requests. User /Doctor can select the option to accept or reject the appointment request from the drop-down menu.

**Exception**: no exception for this use case.

**Use-case 5**: View patient

**Primary-actor**: User/doctor

**Description**: User/doctor would be able to view the patient's detailed information including his/ her medical history.

**Precondition**: User/doctor must be signed in

**Basic use-case flow**: After selecting the date, user/doctor selects the patient. User/doctor can view the patient's detail and medical history.

**Main scenario**: After selecting the date, list of patients of that particular date will appear. User /Doctor will select the patient. User (doctor) can view his all information including the medical history by clicking on the "view patient" button

**Exception**: no exception for this use case.

**Use-case 6**: Add comments

**Primary-actor:** User/doctor

**Description**: User/doctor would be able to write his own comments into the patient's medical profile.

**Precondition**: User/doctor must be signed in

**Basic use-case flow**: After clicking on the "write comments" button, user/doctor can add his own reviews after checking the patient.

**Main scenario**: User/doctor will hit the "write comments" button. User (doctor) will write his or her comments. User/doctor will hit the submit button.

**Exception**: no exception for this use case.

**Use-case 7**: log out

**Primary-actor**: User/doctor

**Description**: the user/doctor will log out from the system.

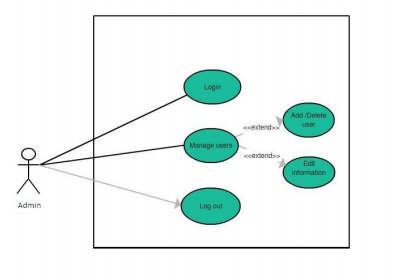
**Precondition**: the user/doctor must be signed in

**Basic use-case flow:** the user/doctor can sign out himself from the system.

**Main scenario**: the user/doctor clicks on the log out button. The system will bring the user to the main page for the generic user.

**Exception**: no exception for this use case

### Admin Use Case:



*Figure 2.5 Use Case Diagram for Admin*

**Use-case 1:** Login

**Primary-actor:** Admin/administrator

**Description**: A user, who possesses admin username and password, owns administrative rights. **Precondition**: username and password.

**Basic use-case flow**: Admin/user needs to provide the username and a password.

**Main scenario**: Admin/user enters valid username and password. After verification, the user will be logged in as an admin and can use all the administrative rights.

**Exception**: occurs if user provides wrong username/password.

**Use-case 2:** Manage users

**Primary-actor**: Admin/user

**Description:** User/admin can use his administrative rights.

**Precondition**: the user must be logged in as an admin.

**Basic use-case flow**: Admin/user can view, edit or delete any user information.

**Main scenario**: Admin can view, edit or delete any user or any information related to user doctor or user-patient.

**Exception**: no exception for this use case.

**Use-case 3:** log out

**Primary-actor:** Admin/user

**Description**: Admin will log out from the system.

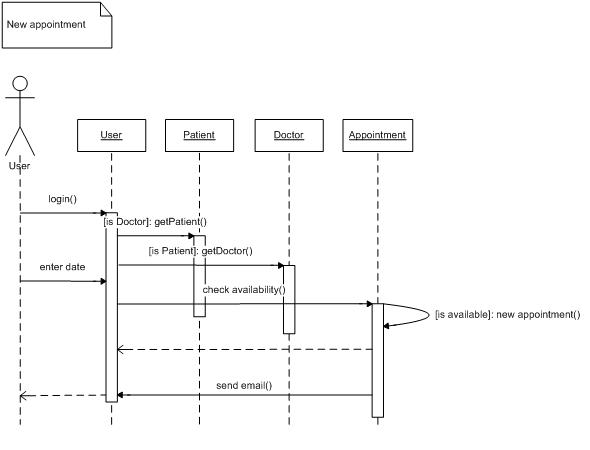
**Precondition**: Admin must be signed in

**Basic use-case flow**: User signs out himself “as an admin".

**Main scenario**: User clicks onto the logout button. The system will remove information from the local storage and brings the user to the homepage.

**Exception**: no exception for this use case.

**2.3.2 Sequence Diagram:**

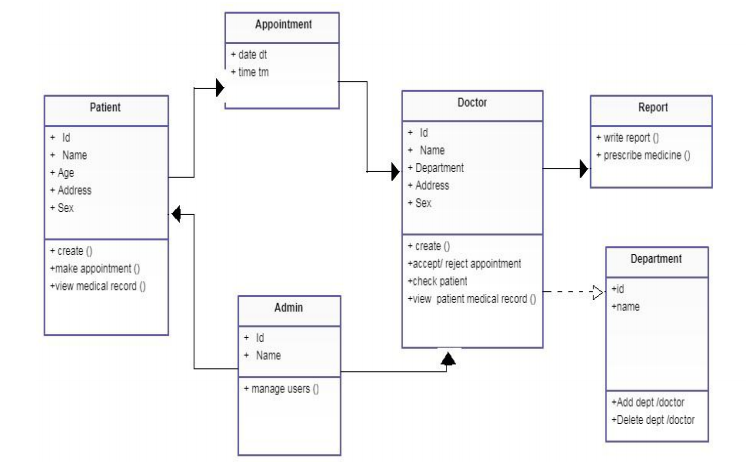


*Figure 2.6 Sequence Diagram*

### 

### 2.3.3 Class Diagram:

The class diagram is chosen to explain the design phase of the system. A class diagram describes classes of the system, attributes, and operations and relationships of the classes in a better way. We can also say that class diagrams are used to justify the structure or behavior of use cases of the system. Class diagrams best explain the conceptual model of the system in terms of entities and their relationships. The class diagram looks like a shape of a rectangle, comprising three compartments stacked vertically. The first top box comprises the class name, the second middle box contains the attributes of the class and third the last box contains the methods or functions performed by that class. The first compartment /box of the name is compulsory while rest of the two can be omitted to simplify the diagram. So, in any class diagram first compartment must be drawn while the second two compartments are optional. The class "patient" contains multiple parameters (such as id, name, age, address), which depict the information of all the registered patients. The user class also contains the methods performed by these users such as get appointment, view/ create own medical record etc. In the same way, the class "doctor" has the parameters id, name, department, address possessing all the required information of the users registered as a doctor on to the system. Methods include accept/reject the appointment, check the patient, view a medical record of any patient etc. These methods are the functions performed by the users registered as a doctor on the system. The class “appointment” has the parameters of date and time, explaining what time or day patient user has requested for the 33 appointment to the doctor. The class “department” has the parameters id and name and methods include add/ delete doctor and add or delete department. Every doctor user must belong to any department class. The “report” is another class containing methods like write report or prescribes medicine. In the end, class “admin” contains the parameters like id and name and methods of this class include manage users.

****

*Figure 2.7 Class Diagram*

### 

**2.3.4 Context Diagram:**

User

Doctor Appointmentsystem

System

Ask for action

Assign task

Perform task

Give fusibility status

*Figure 2.8 Context Diagram*

**2.3.5 Data Flow Diagram:**

Generally, DFD’s are used as a design notation to represent architectural design (External design) and top level design (internal design) specifications. DFD’s represent the system in hierarchical manner with one top level and many lower level diagrams with each representing separate parts of the system. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel Since diagrammatic representations are easier to interpret as compared to the technical descriptions, the non-technical users can also understand the system details clearly. DFD consists of four basic notations which help to depict the information in the system. These notations are rectangle, circle, open-ended rectangle, and arrows.

**2.3.5.1 Zero Level DFD**

Patient

Admin

Doctor Appointment

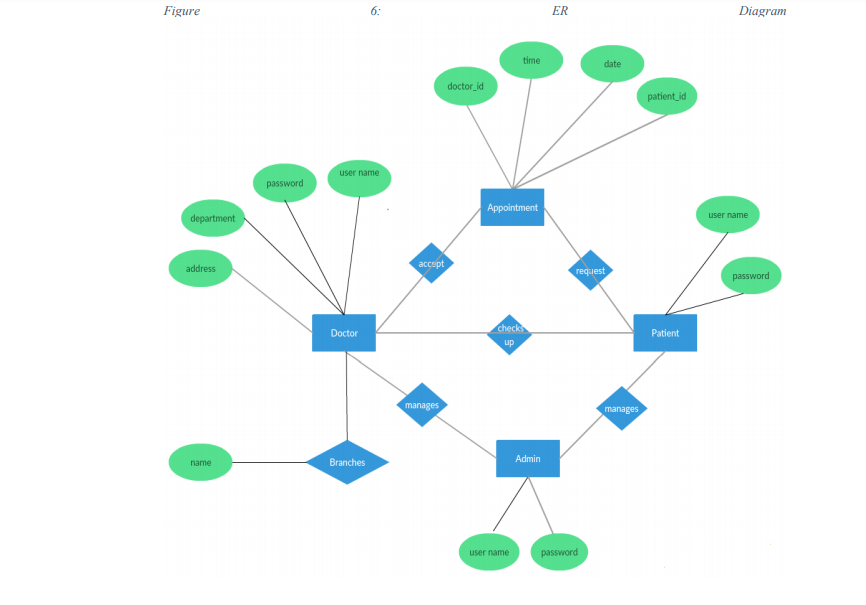
System

Doctor

*Figure 2.9 Zero Level DFD*

### 2.3.6 ER Diagram:

A basic ER model consists of objects called entities and specifies relationship among those entities. Purpose of this diagram is not to define any functionality rather show association and dependency among entities. ER diagram is drawn with "rectangular boxes" as entities and the "straight lines" showing the relationship between these boxes. An entity is an object or a thing that has an independent existence and can be easily differentiated from others. Each entity has some attributes like name, age, address, department etc. In the following diagram the doctor, patient, appointment, admin etc, all are different entities. So, an entity can be a person, animal, plant, event or a company. 34 Entities consisting of similar attributes make the entity sets. These entities have some association among each other which make a relationship. These relationships can be "one to one" or "one to many" or "many to many". For example, a doctor and department can have "one to many" relationships, means one department can have many doctors but one doctor is related to only one department.

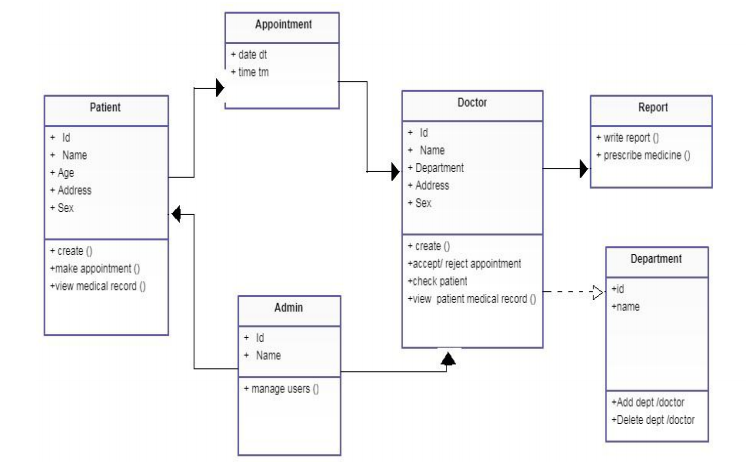


*Figure 2.10 Entity Relationship Diagram*

### 

### 2.3.7 Database Model:

This database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed in my project. Individual database models are designed based on the rules and concepts of data model that I used. Most data models are represented by an accompanying database diagram. Below is database model for my Appointment management system.



*Figure 2.11 Database Model*

### 

### 2.3.8 Architecture:

**3-Tier:**

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This added flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system.

**Presentation Tier-**

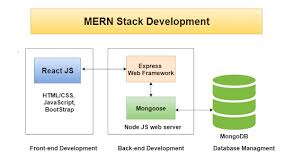
The presentation tier is the front-end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, React CSS, or through other popular web development frameworks, and communicates with other layers through API calls.

**Application Tier-**

The application tier contains the functional business logic which drives an application’s core capabilities. It is written in Nodejs (express).

**Data Tier-**

The data tier comprises of the database/data storage system and data access layer. In my project “Doctor Appointment System” this third tier is built using Mongo db. And API calls are used to access the data from server.



*Figure 2.12 Application Architecture*

# 

# Chapter 3 - RESULTS & DISCUSSION

**3.1 System Testing**

For the sake of software quality assurance, system testing is a very essential thing to do. It is a process by which we try to make the system error proof by performing the program to find an error. The goal is to run the program, find errors or bugs and then fix them. Testing is considered a very essential step in software development and any system is not considered to be complete without this process. Different developers follow different approaches for system testing. System testing is followed to make sure that system is working fine and meets the requirements.

**3.2 Test Cases:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Test case name | Precondition | Priority | Input test data | Steps to be executed | Expected Result | Successful or not |
| 1 | User\_reg | no | high | Valid email address or correct national ID card no | Need to fill the registration form and submit | Registration successful | OK |
| invalid email address or incorrect national ID card number | ERROR Message | ERROR |
| 2 | User\_login | Must be registered earlier | high | Valid username and password | Enter username and password and submit | A successful login to the system | OK |
| Invalid username or  password | Login failed, and the system displays an error message | ERROR |

*Table 3.1 user login test case*

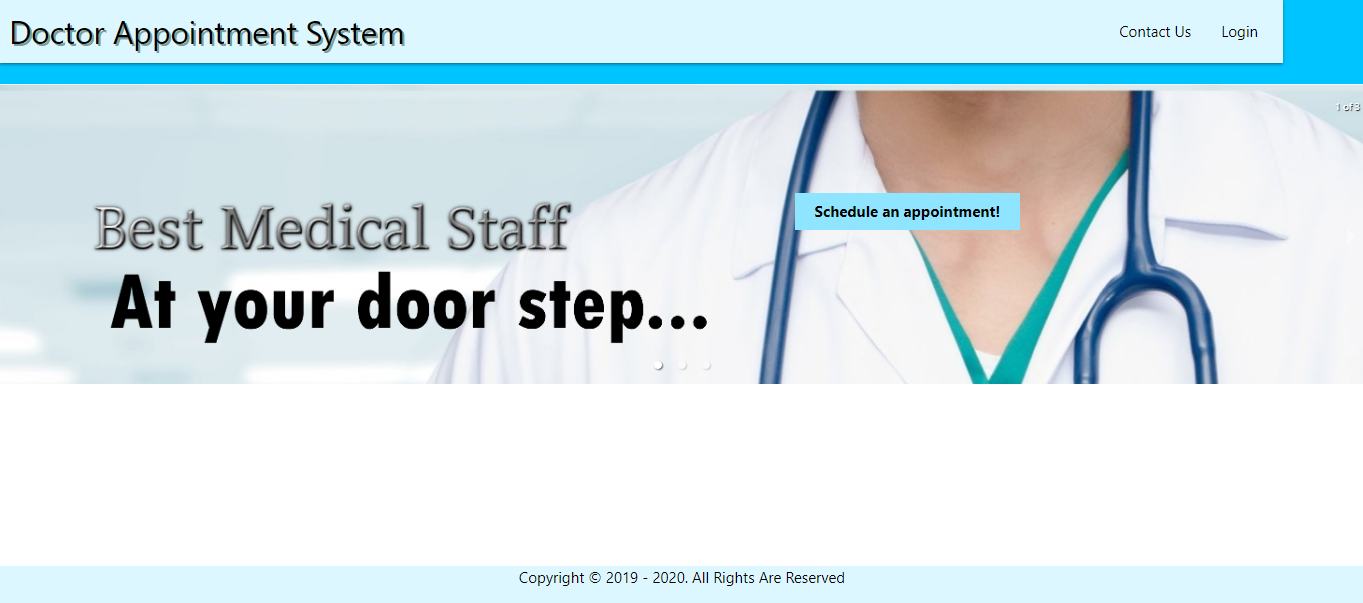
**3.3 Conclusion:**

Making clinic appointments shouldn’t be hard, but often due to the manual way of making appointments, patients find it’s hard to make appointments with their desired doctor. Often patients have to wait in long queues and yet sometimes they won’t be able to book their appointments. Clinic Appointment System is an easy solution for such patients. They don’t have to wait in endless queues or ask someone to do them a favour, because Clinic Appointment system has everything they need to make an appointment with their desired doctor. Clinic Appointment System comes with a clean and responsive interface, so that user can make appointments from every device, all they need is an internet connection. From the hospital/clinic point of view, they often fail to satisfy the needs of their patients/customers. Sometimes, a staff may need to take a leave and it may cause the whole appointment procedure to go down. With the new Clinic Appointment System, all they need to do is deploy the application to their web server and they are done. Once the administrator has added all the departments and doctors list, patient can book their appointments by visiting the system. No need of a third person to process the request. Clinic Appointment System also provides doctors an account to interact with their patients. Now doctors can easily send prescriptions online, or see their patient details in advance or even better, they can cancel/ put the appointment to the pending list. Patient on the other side, don’t have to worry about losing his/her prescription since it’s stored in the hospital’s servers

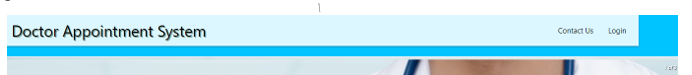
# Chapter 4 – Application Screenshots

Following are some screen-shots of Doctor Appointment App.

1. **Home Page**

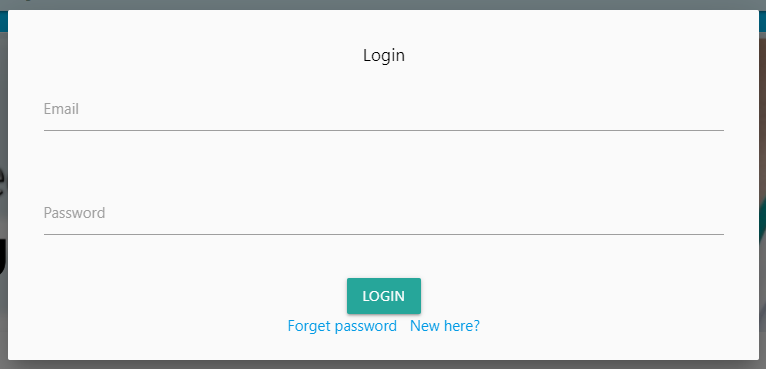
*\***** *Figure 4.1 Home Page*

**2-Navbar**

****

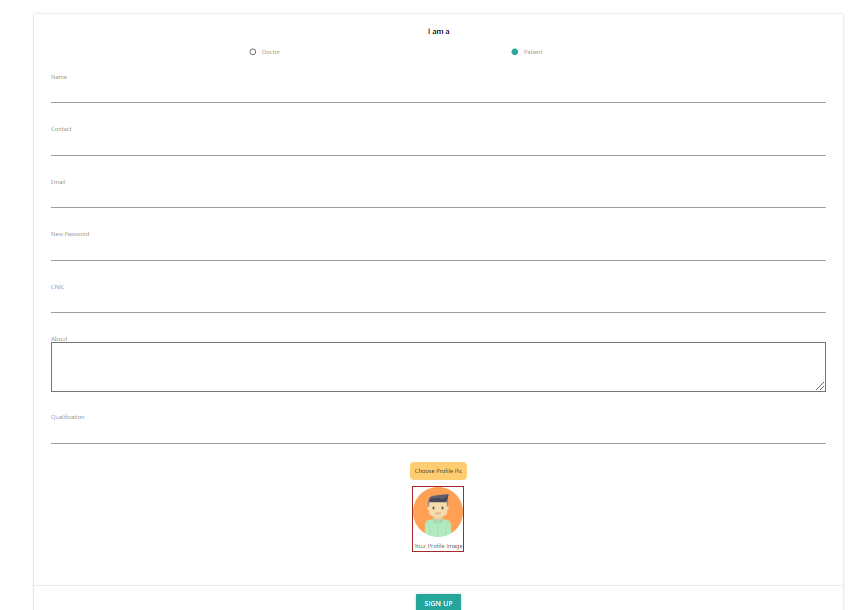
*Figure 4.2 Navbar*

**3-Login page**

****

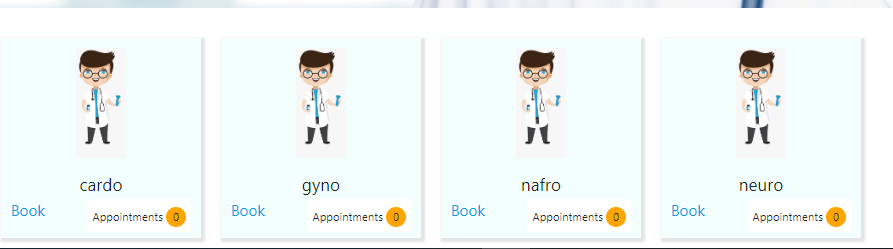
*Figure 4.3 Login Page*

**4- Signup page**

****

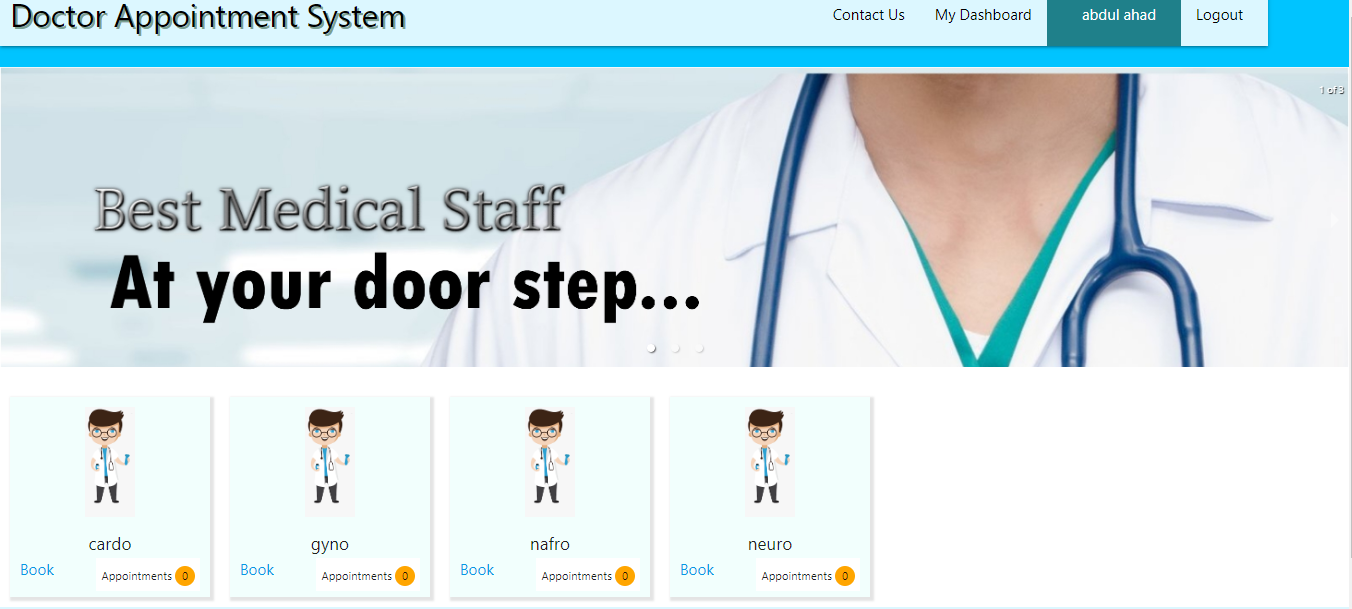
*Figure 4.4 Signup Page*

**5-Categories Features**

****

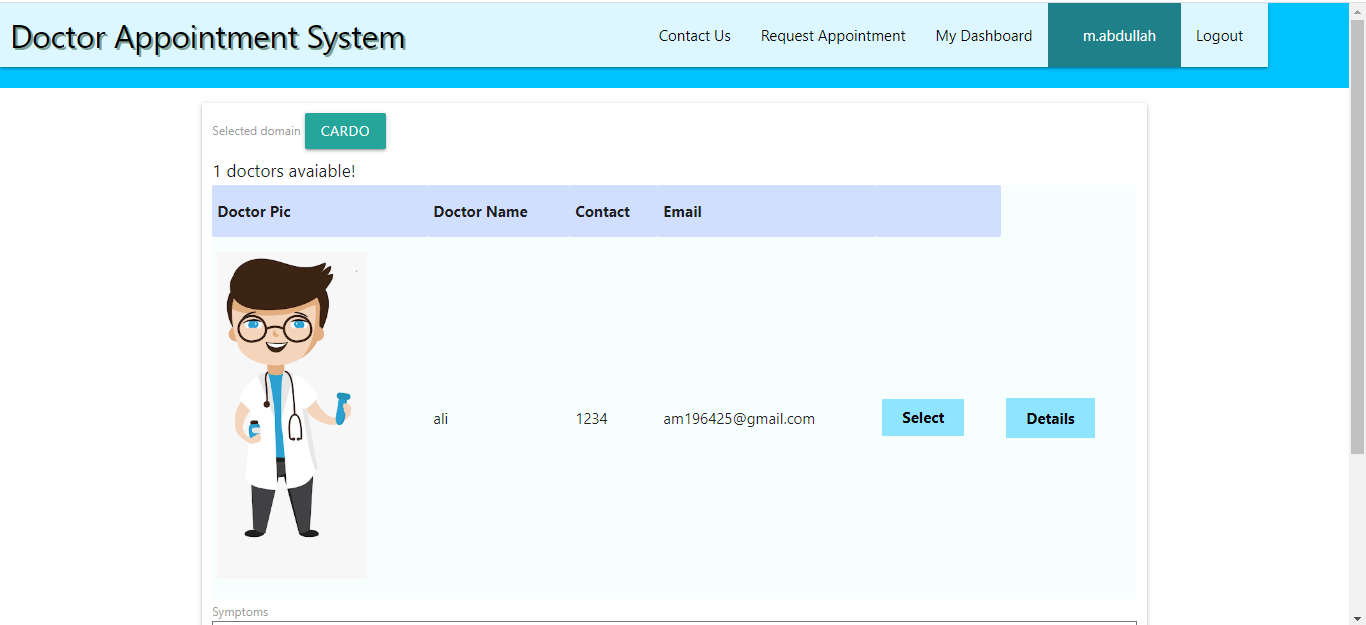
*Figure 4.5. Categories Featured*

**6- All Doctors**

****

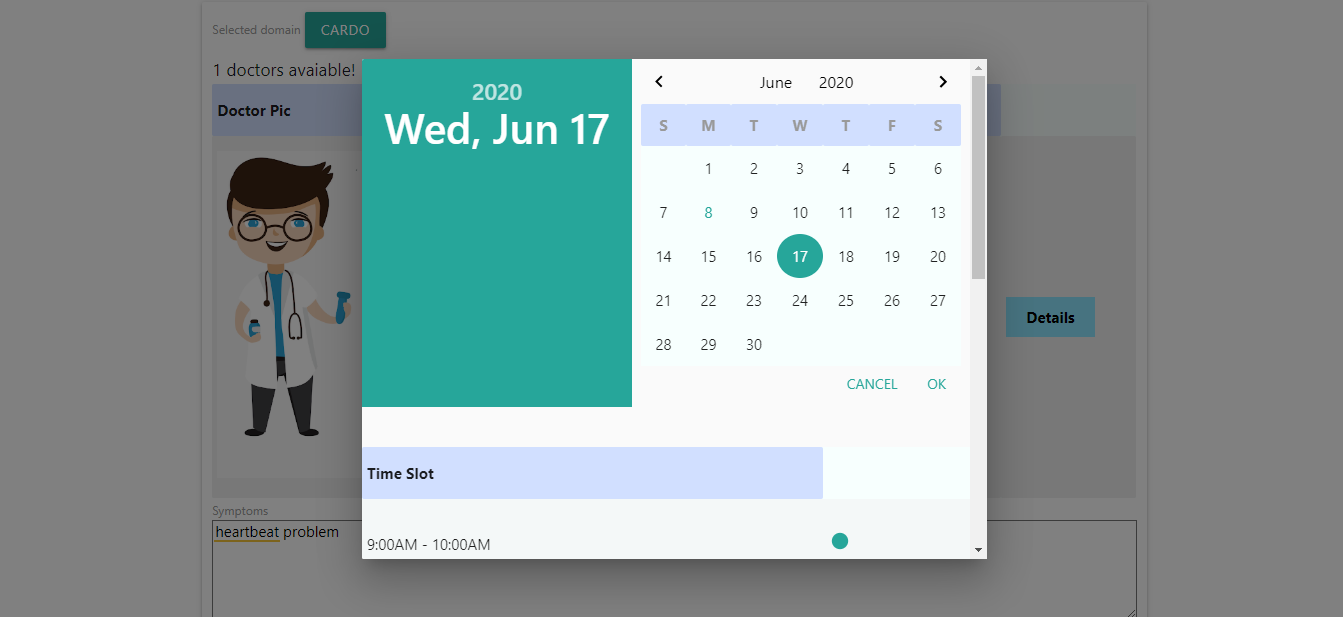
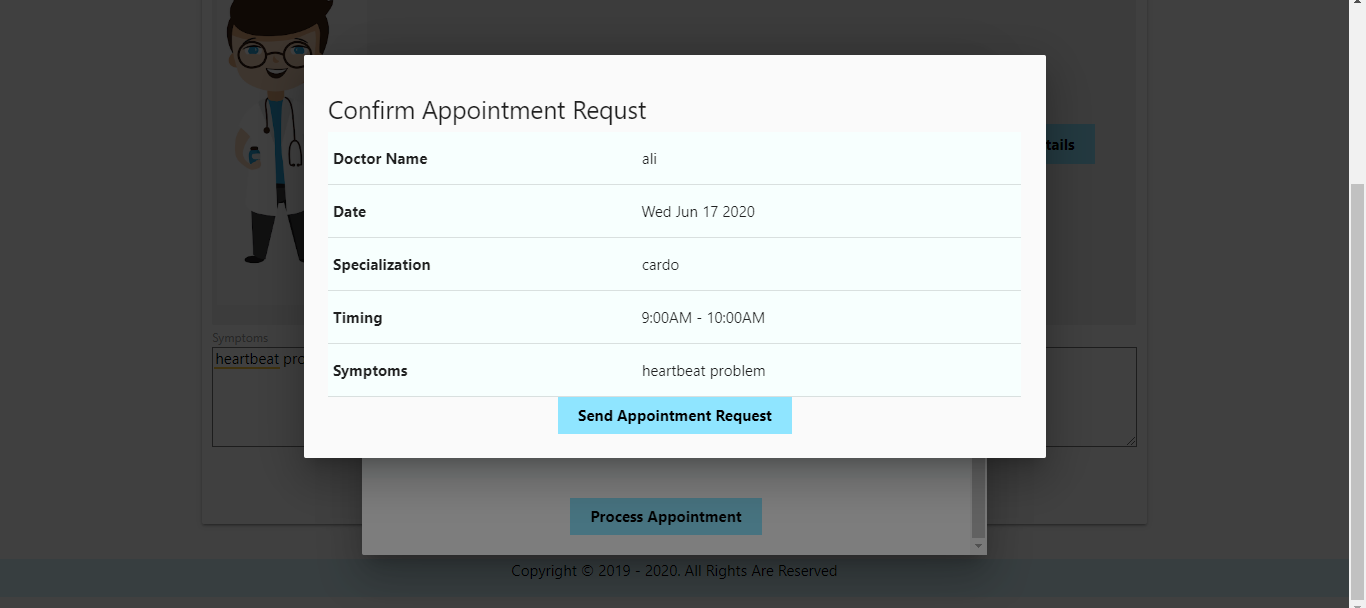
*Figure 4.6 All doctor*

**7-Patient Dashboard**

****

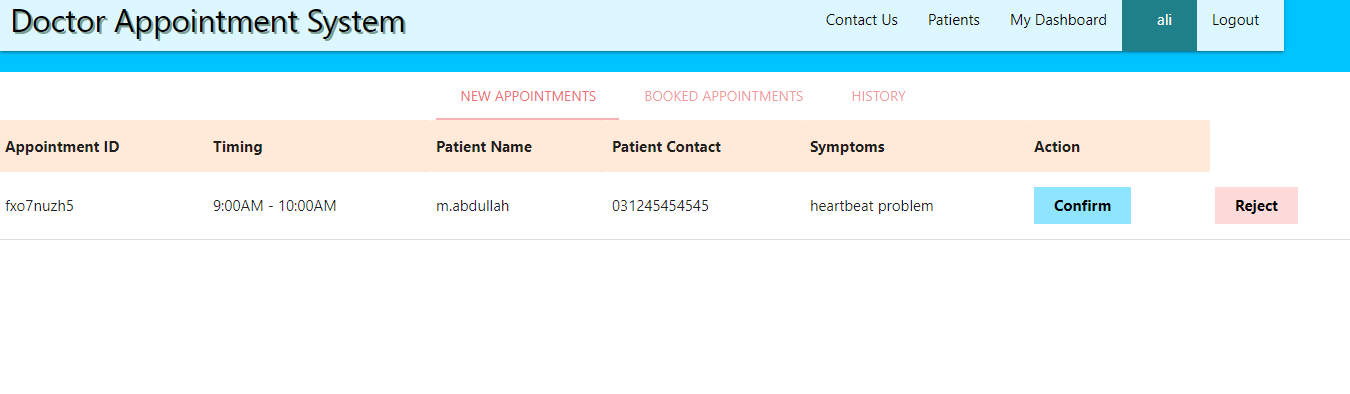
*Figure 4.7 Patient Dashboard*

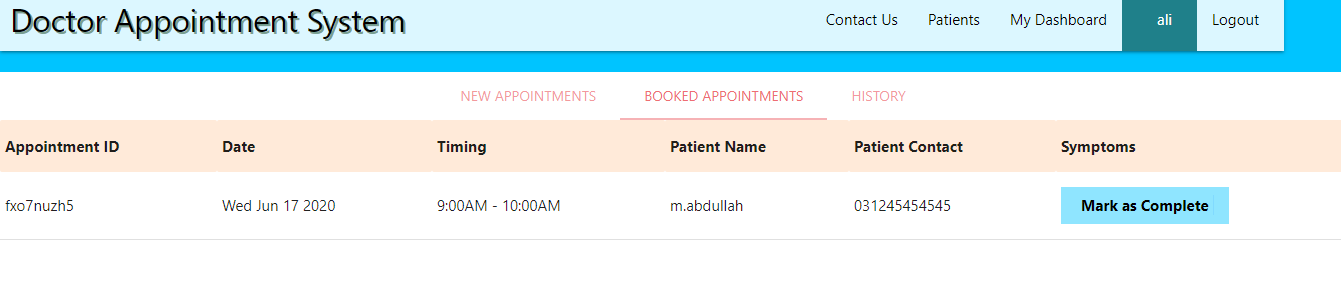
**8-Patient Request for Appoitnment**

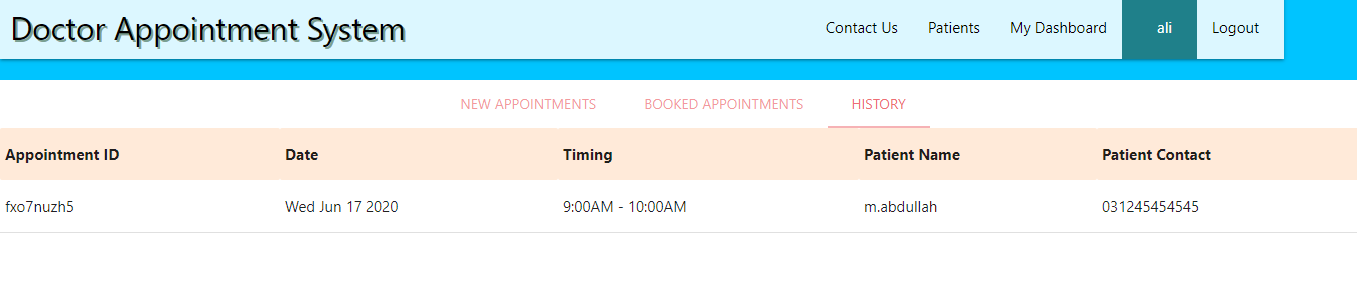
****

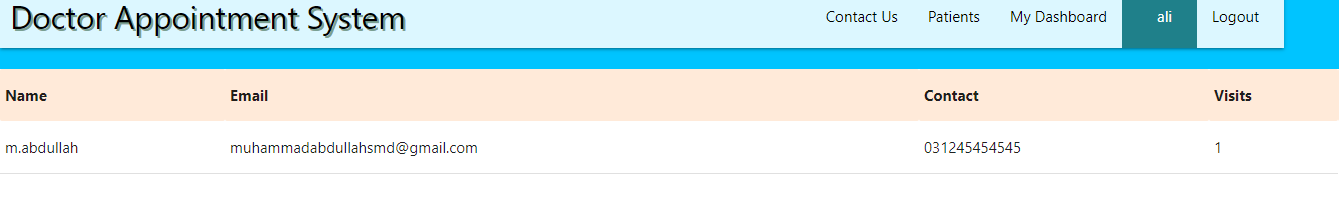
*Figure 4.8 Patient Request for appointment*

**9-Doctor Dashboard**

****

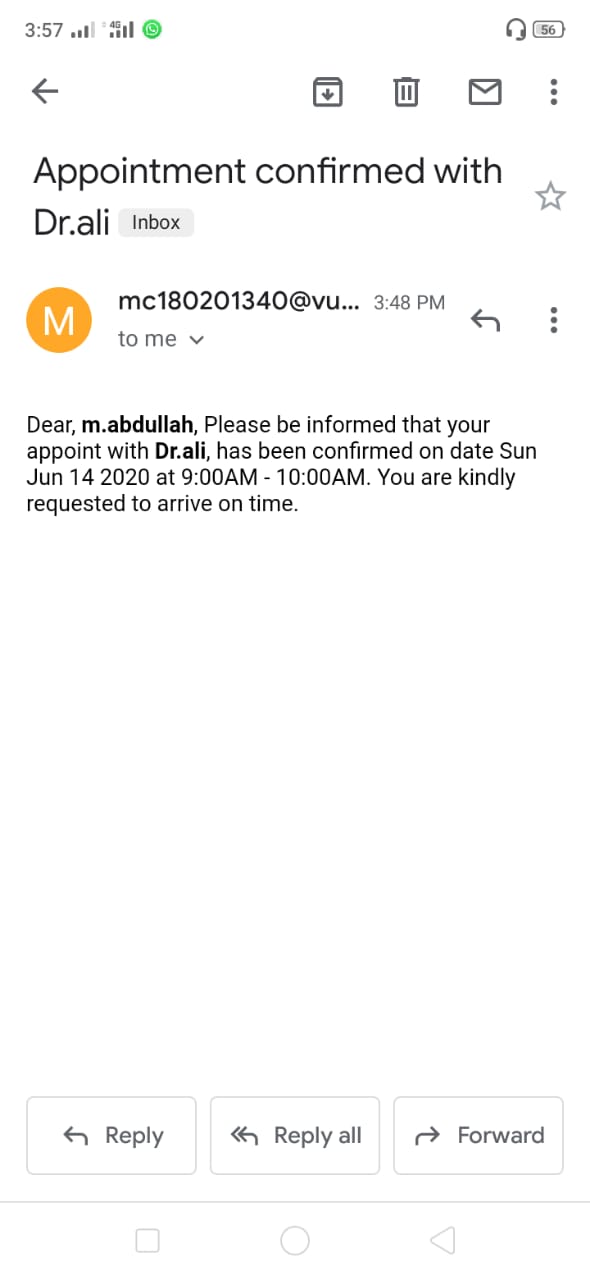
****

****

****

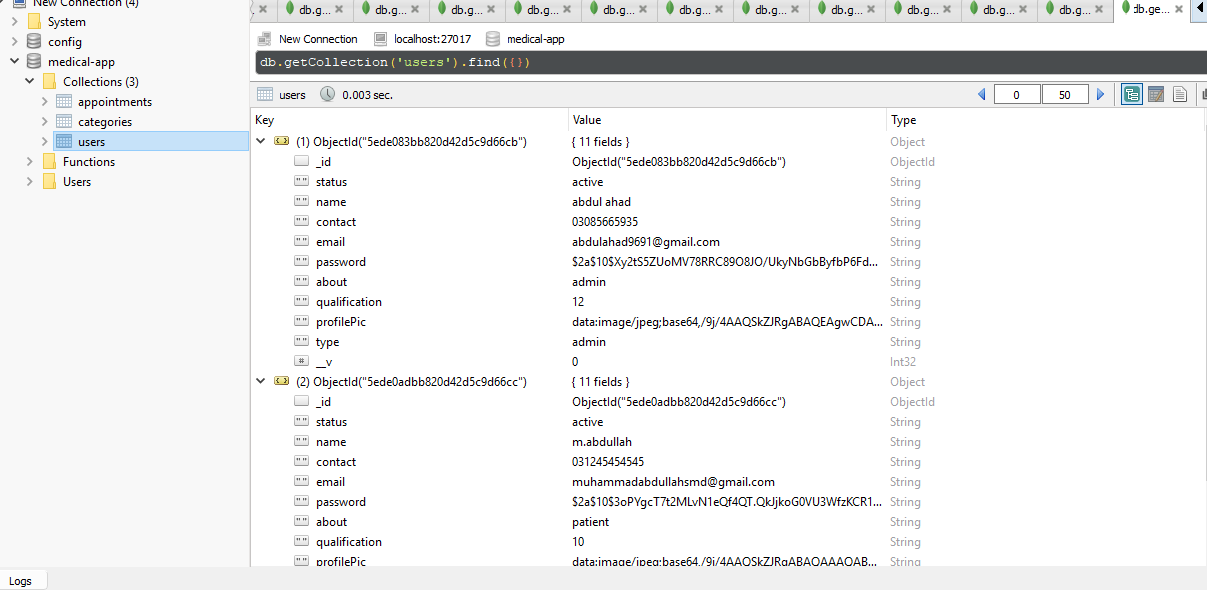
*Figure 4.9 Doctor Dashboard*

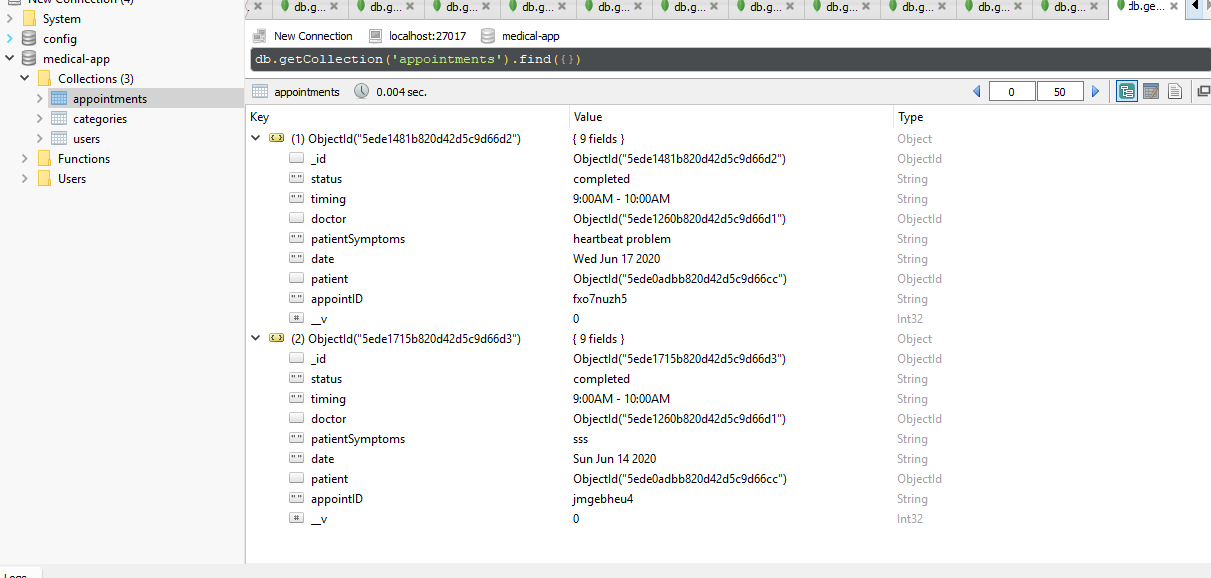
**10- Conformation Mail**

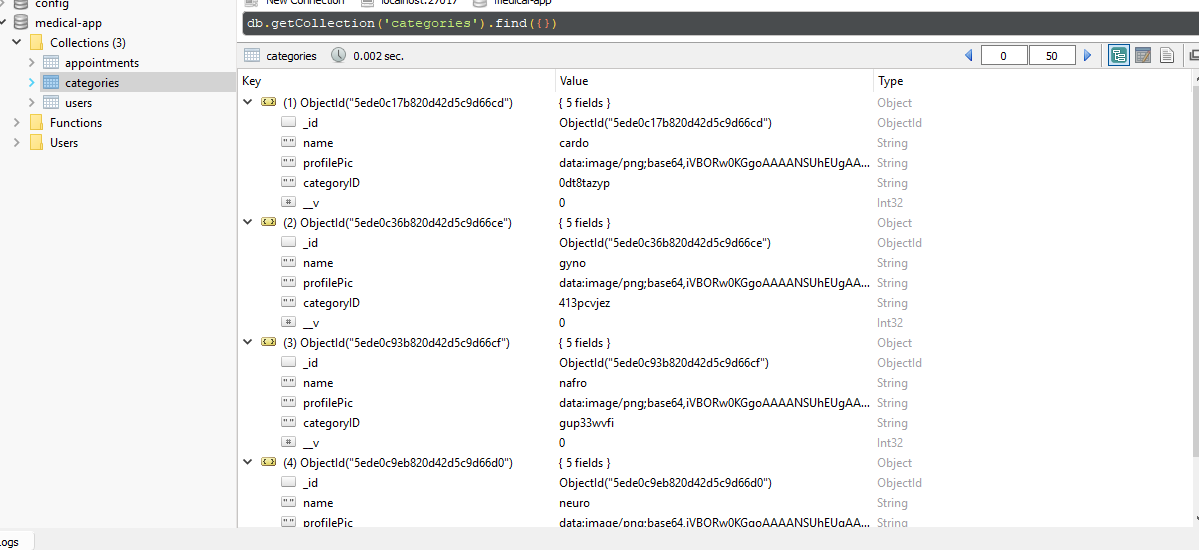
****

*Figure 4.10 Confirmation mail*

**11-Database**

****

****

****

*Figure 4.11 Database*