**Islamic university of Science and Technology, Awantipora, Kashmir, 192122.**

**Department of computer science.**

**School of Engineering and technology**

**PROJECT REPORT ON**

**ONLINE VETERINARY CARE SYSTEM**

Project Report Submitted in partial fulfillment of requirements for the award of the degree of MASTER OF COMPUTER APPLICATIONS

Submitted by: Supervised by :

Zubair Ahmad Lone Mrs Shabana Nargis

Jasiyah Jabbar Assistant Professor

Humaira Parray

Islamic University Awantipora

**ISLAMIC UNIVERSITY AWANTIPORA**

**CERTIFICATE**

This is to certify that the project entitled **ONLINE VETERINARY CARE SYSTEM**

Has been carried out by:

Zubair Ahmad

Jasiya Jabbar

Humaira Parry

In partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTER APPLICATIONS

In the academic year 2020

Supervisor Academic Cordinator

Shabana Nargis Dr Syed Mohsin Saif

Department of computer Applications

Islamic university of Science and Technology, Awantipora, Kashmir, 192122.

Department of computer science.

School of Engineering and technology

**STUDENT DECLARATION / CERTIFICATE**

This is to certify that the above statements made by the candidates are correct to the best of my knowledge, which is being presented in the project entitled ” ONLINE VETERINARY CARE SYSTEM” in partial , we ZUBAIR ,JASIYA ,HUMAIRA declare the work , fulfilment of the requirement for the award of the degree of MATER OF COMPUTER APPLICATIONS (MCA) degree in the session of 2020 , is an authentic record of our own team work carried out under the supervision of MRS SHABANA NARGIS ,ASSISTANT PROFESSOR , DEPARTMENT OF COMPUTER APPLICATION , ISLAMIC UNIVERSITY AWANTIPORA

The matter embodied in this project has not been submitted by us for the award of the any other degree.

DATED:

ZUBAIR AHMAD LONE

JASIYAH JABBAR

HUMAIRA PARRY

**ACKNOWLEDGEMENT**

All the praise to the Almighty ALLAH our heads bow in humble towards ALLAH for blessing us with the strength and courage to accomplish this task.

We fall short of words to pen down anything for the co-operation and encouragement extended by our parents throughout on every step of our life and for the completion of this project. A mere “Thanks “is little for their love, encouragement and support .

In the completion of this project, there are countless debits incurred that can never be adequately repaid .To render an expression of simple thanks seems necessary .We are highly thankful to our project Guide SHABANA NARGIS, ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER APPLICATIONS ISLAMIC UNIVERSITY OF SCIENCE AND TECHNOLOGY, SCHOOL OF ENGINEERING AND TECHNOLOGY for without her support this project was impossible.

We are also thankful to the other assistant professors and other technical staff members of our department. They have been very helpful and kind to us throughout our MCA course.

At last we would like to thank all those who have given us support and encouragement and helped us in the execution of this project.

**INDEX**

1. INTRODUCTION …………………………………………………………….1

1.1) PROJECT OVERVIEW…………………………..……………………….1

1.2) PROJECT OBJECTIVES…………………………………………………..1

2. PROPOSED SYSTEM........................................................................................2

2.1) FEASIBILITY STUDY……………………………………………………..2

2.2) PROJECT PLAN………………………………………………...………....5

2.3 SOFTWARE ENGINEERING PARADIGM USED……………………….5

3. SYSTEM ANALYSIS…………………………………………………………..6

3.1) REQUIREMENTS ENGINEERING………………………………………6

3.2) ANALYSIS MODEL…....…………………………………………………9

3.3) SOFTWARE REQUIREMENT SPECIFICATION………………………11

4. SYSTEM DESIGN ……………………………………….………………….…16

4.1) MODULE DESIGN ……………………………………………………......21

4.2) DATA FLOW DAIGRAMS (DFD)……………………………………..….22

4.3) UML DIAGRAMS.…………………………………………………………26

4.4) INPUT /OUTPUT DESIGN……………………………………..………….27

5. CODING…………………………………………………………………………28

6. TESTING ………………………………………………………………………..48

7. IMPLEMENTATION OF PROJECT …………………………………………...57

8. SNAPSHOTS….…………………………………………..……………………..58

9. CONCLUSION………………………………………………….…..…………...70

10. FUTURE WORK/RECOMMENDATIONS……………………………………71

11. APPENDICES…………………………………………………………………..72

12. BIBLIOGRAPHY……………………………………………………………...75

**Chapter 1. INTRODUCTION**

* 1. **PROJECT OVERVIEW**

Online applications are playing an important role in our day to day life. From online shopping to online doctor booking, these applications are saving time and helping ineffective management of resources. AS far as veterinary care system is concerned, many applications for online veterinary systems like “vet coach live” are available, however these applications are limited in providing the information about vaccination schedule, no prompt response in case of emergency and less number of appointments. These applications charge fee when users want detailed information about vaccination, disease, treatment etc. Our application will overcome these limitations by introducing instant messaging where doctor can chat with user through instant messaging. Our application will also provide more information on vaccination schedule free of cost.

**1.2) PROJECT OBJECTIVES:**

Our goal is to develop a robust application which will help users to cure their pets instantly. Following are the objectives of our application:

To help users to book appointments with the veterinary doctors to cure their pets.

To help users to chat with doctors instantly in case of emergency.

To help users to get enough information regarding vaccination schedule, camps to be organized etc.

**Chapter 2. PROPOSED SYSTEM**

**Existing system:**

At present, there are many websites like https://www.petcoach.co/, http://vetcoachlive.com/ that provide information related to pets and provide answers to users’ questions related to the diseases of pets.

**Disadvantages:**

These websites do not provide much information and brief answers to users’ questions and require money when more information is needed.

In case of emergency, users do not get prompt response for treating their pets.

There are a limited number of appointments available.

**Proposed system:**

In proposed system website is developed by which users can book appointments with the doctors available for their pets. They can also use instant messaging service provided by the application in case of emergency. Enough information is also available to the users which can be beneficial to them regarding the diseases of their pets and the vaccines available.

**Advantages:**

Users can book appointments by selecting a specific doctor and the time slot available.

Users can instantly chat with the doctors in case of emergency and doctor can recommend them the immediate treatment.

A lot of information will be available to the users regarding the pets’ diseases and the vaccines.

Users can give feedback about doctors and the feedback will be evaluated by the Admin and if negative feedback is found, the doctor may be rejected.

**2.1 Feasibility Study:**

A feasibility study is undertaken to determine the possibility of either improving the existing system or developing a completely new system. This helps to obtain an overview of the problem and to get rough assessment of whether feasible solutions exist. The purpose of feasibility study is to determine whether the requested project is successfully realizable. There are three aspects of feasibility study namely:

Types of Feasibility Study: -

**2.1.1) Technical feasibility.**

**2.1.2) Economic feasibility.**

**2.1.3) Operational feasibility.**

**Technical feasibility:**

Technical feasibility is concerned with specifying equipment and software that will successfully support the required task. It centres on the required /existing computer system(hardware/software) and to what extent it can support the proposed application. For example, if the current computer is operating at 80% capacity, then running another application could overload the system or require additional hardware. This requires financial consideration to accommodate technical enhancement. This should answer the following questions:

Whether the project can be carried out with the existing equipment?

Whether the existing software is enough?

If a new technology is required, how best can it be implemented?

In case of our application, the following software that is required for developing the application is:

Microsoft Visual Studio 2017 or higher.

Microsoft SQL Server 2014 or higher.

For running the application, the following software is required:

Internet Information Services.

The following Operating System is required to develop and run the application:

Microsoft Windows 8 or higher with Internet Information Services installed. OR

Microsoft Windows Server 2016 or higher with Internet Information Services installed.

And the following hardware is required for

developing and running the application:

Intel Core i3 or higher.

8GB of RAM or higher.

100GB of Hard Disk Space or higher in case there is too much data stored in database.

**Economic feasibility:**

Economic feasibility study is the most frequently used method for evaluating the effectiveness of a new system. Cost-benefit analysis is performed to determine the benefits and savings that are expected from the new system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the new system. By conducting this study, the analyst can ascertain the following:

Whether the project is economically feasible?

If enough funds are not available, then what are the sources of funds?

Whether there are sufficient benefits when compared to the costs incurred?

In case of our application, the main resources are the developers developing the application. There will be one time cost of developing the application. Also there will be the cost of domain and hosting when the application is launched to the general public. But nowadays there are many hosting and domain providers which provide domain and hosting services at very low costs. And when the application is complete and launched to the general public, then we will earn from appointments which are booked by the users and we can also display ads on the website which will be a source of revenue.

**Operational Feasibility:**

Operational feasibility is concerned with human, organizational and political aspects. Operational feasibility covers two aspects. One is a technical performance aspect and the other is acceptance within the organization. Technical performance includes issues such as determining whether the system can provide the right information for the organization’s personnel, and whether the system can be organized so that it always delivers this information at the right place at the right time.

Our application will be technically sound because we will use RESTful API’s which are light weight and perform faster than simple server side based web applications. And it will also be accepted by public because it offers beneficial material to the users and also benefits them in making appointments and hence saving their time and serving their pets in case of emergency. And also this application does not harm any person’s social, political or economic interests.

**2.2. Project Planning:**

Software Project Management Begins with a set of activities that are collectively called Project Planning. Before the project begins, the manager and the software team must estimate the work to be done, the resources that will be required, and the time that will elapse from start to finish.

**Project Estimation:**

Project Estimation is an attempt to determine how much money, effort, resources, and time it will take to build a specific software-based system or product. Estimation begins with a description of the scope of the product. The problem is then decomposed into a set of smaller problems, and each of these is estimated using historical data and experience as guides. Problem complexity and risk are considered before a final estimate.

Our project will require the following amount of resources, money and time:

120 work hours x 100/hour = 12000

Core i3 Laptop/Desktop or higher configuration

4GB of RAM or higher (Recommended 8GB)

10GB of Disk Space or higher

Windows 8 or higher operating system

And to test on multiple systems, Internet Information Services is required and network sharing should be on in all the systems.

**Project Planning Objectives:**

**Software Scope:** The first activity in Software Project Planning is the determination of software scope. Software Scope describes the data and control to be processed, function, performance, constraints, interfaces and reliability. In case of our project, the data are the details of the users like name, email, phone no etc., the details of doctors, the details of booking slots, the details of appointments, messages to be shared between users and doctors and the vaccination schedule and other information available on the website. Our project will use RESTful api’s which are light weight and are very much responsive.

**Feasibility:** Once the scope has been identified, it is reasonable to ask: “Can we build software to meet this scope? Is the project feasible?”. Our project will meet the goals and objectives discussed above.

**Resources:** Another software planning task is the estimation of the resources required to accomplish the software development effort. We have estimated that our project will need almost 120 work hours of a developer.

**Decomposition:** Software Project estimation is a form of problem solving, and in most cases, the problem to be solved is too complex to be considered in one piece. For this we decompose the problem, re-characterizing it as a set of smaller problems. In our project, we have decomposed the whole software into the modules discussed above.

**2.3. Software Engineering Paradigm Used**

We have used Object Oriented Programming Paradigm in developing this application.

**Chapter 3. SYSTEM ANALYSIS**

**Introduction**

System analysis is the process of examining the situation with the intent of improving it through better procedures and methods. System design is the process of planning a new system to either replace or complement an existing system. But before any planning is done, the old system must be through understood and the requirements determined. System analysis is therefore, the process of gathering and interpreting facts, diagnosing problems and using the information to re-comment improvements in the system. Or in other words, system analysis means a detailed explanation or description. Before computerizing a system under consideration, it has to be analysed. We need to study how it functions currently, what are the problems, and what are the requirements that the proposed system should meet.

System analysis is conducted with the following objectives in mind:

1.Identify the customer’s need.

2.Evaluate the system concept for flexibility.

3.Allocate functions to hardware, software, people, database and other system elements.

4.Establish cost and schedule constraints.

5.Create a system definition that forms foundation for all the subsequent engineering work.

**3.1 Requirement Engineering:**

A requirement is defined as a condition or capability that must be met or fulfilled by a system to specify a contract, standard, specification. The requirements defined for a system should be correct, consistent, verifiable and traceable. Requirement Engineering is the process of eliciting, understanding, specifying and validating customer’s requirements. Requirements Engineering is the first technical step in the software process. Analysis must focus on information functional and behavioural domains of the problem. Analysis is about understanding situations, not solving problems. This is the first and important phase of software development because what will be developed on analysis. Only when a complete analysis is made, recommendations for design can be made.

Requirement Engineering provides the appropriate mechanism for understanding what the customer wants, analysing needs, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously validating the specification, and managing the requirements as they are transformed into an operational system.

The iterative process of requirement engineering consists of following steps: -

**Inception**

At this stage different sources of information including doctors, pet owners and shepherds were interviewed. The intent was to build a basic understanding of the problem, the people who want the nature solution that is desired and the effectiveness of preliminary communication and collaboration.

**Elicitation**

At this stage we asked the users (pet owners) what the objectives for the system are, what is to be accomplished, how the system fits into the needs and finally how the system is to be used on a day to day basis.

**Elaboration**

We expanded and refined the information obtained during inception and elaboration. We focused developing a refined technical model of software functions, features and constraints.

**Negotiation**

In this activity we reconciled the conflicting requirements proposed by the various pet owners, doctors and other health workers.

**Specification**

Here we produced a written document specifying all the captured requirements in a consistent and therefore more understandable manner. This document served as the foundation subsequent software engineering activities.

**Validation**

The product produced as a consequence of requirements engineering was assessed for quality during this step, in order to ensure that all software requirements have been stated unambiguously; that inconsistencies, omissions and errors have been detected and corrected; and that the work product conforms to the standard established for the process the project and the product.

In case of our software, we do not have to get requirements from any person or group of persons. Instead, we have to analyze existing system and see the limitations of these systems and try to overcome those limitations in the existing systems. For this purpose, we have analyzed some websites like https://www.petcoach.co/ and vetcoachlive.com. After analyzing these websites, we have noted some limitations and will overcome those limitations in our software.

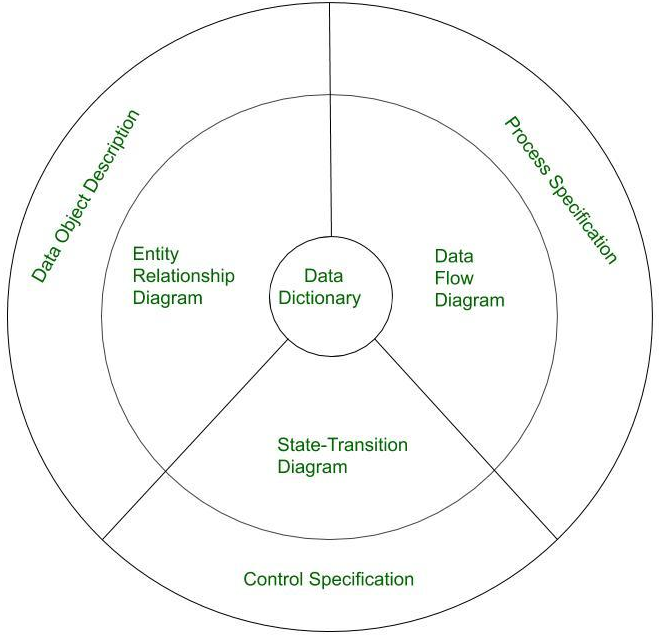
**3.2. Analysis Model**

**Analysis Model** is a technical representation of the system. It acts as a link between system description and design model. In Analysis Modelling, information, behaviour and functions of the system is defined and translated into the architecture, component and interface level design in the design modelling.

**Objectives of Analysis Modelling:**

1. It must establish a way of creation of software design.
2. It must describe requirements of customer.
3. It must define set of requirements which can be validated, once the software is built.

**Elements of Analysis Model:**



1. **Data Dictionary:**  
   It is a repository that consists of description of all data objects used or produced by software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modelling of data objects defined during software requirements.
2. **Entity Relationship Diagram (ERD):**  
   It depicts relationship between data objects and used in conducting of data modelling activity. The attributes of each object in the Entity Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.
3. **Data Flow Diagram (DFD):**  
   It depicts the functions that transform data flow and it also shows how data is transformed when moving from input to output. It provides the additional information which is used during the analysis of information domain and serves as a basis for the modelling of function. It also enables the engineer to develop models of functional and information domain at the same time.
4. **State Transition Diagram:**  
   It shows various modes of behaviour (states) of the system and also shows the transitions from one state to other state in the system. It also provides the details of how system behaves due to the consequences of external events. It represents the behaviour of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.
5. **Process Specification:**  
   It stores the description of each functions present in the data flow diagram. It describes the input to a function, the algorithm that is applied for transformation of input, and the output that is produced. It also shows regulations and barriers imposed on the performance characteristics that are applicable to the process, and layout constraints that could influence the way in which the process will be implemented.
6. **Control Specification:**  
   It stores the additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs and which processes are invoked due to the occurrence of the event. It also provides the details of the processes which are executed to manage events.
7. **Data Object Description:**  
   It stores and provides the complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in Entity Relationship Diagram. Hence, it incorporates all the data objects and its attributes.

**3.3. Software Requirement Specification**

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

* Correct
* Unambiguous
* Complete
* Consistent
* Ranked for importance and/or stability
* Verifiable
* Modifiable
* Traceable

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.



We have gathered all the requirements for our application and found out that our application should be able to perform the following activities.

Enable a user/doctor to register with the system.

Enable the admin to approve/reject a doctor, activate/deactivate a user.

Enable the doctor to add booking slots available.

Enable the user to book appointment against a selected date and booking slot.

Online veterinary care system is an online appointment booking platform which provides services of booking appointments, curing of pets, online consultation, vaccination schedule, information about production of hygienic milk and fertilization management.

**Document Conventions**

When writing the SRS for online veterinary care the following conventions are:

* To make the document more effective and readable.

- I used the normal font style, 11 font size and headings are underlined.

**Intended Audience and Reading Suggestions**

This Software Requirements document is intended for:

The intended audience of this document is all major stakeholders which include the developers, testers, the project user and anyone evaluating the project.

− Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes moreorganised for modifications.

-End users of this application who wish to read about what this project can do.

**Product Scope**

Online veterinary care is an online web based platform that will provide facilities to the users to care their pets in a better way. Our software will provide free consultation and would provide information about vaccination schedule.

The scope of our project is to design a safe and easy to use website for the services provided to users.

**Overall Description**

**Product Perspective**

Online veterinary care system is a new and innovative which has additional feature of instant messaging. The product aimed at the person who do not want to visit the doctor physically.

**Product Functions**

• Un- registered users may register to the system.

. Un-registered users may search to the system.

• Registered person may login to the system.

• Administrator may add/update/delete useful tips.

• Administrator may add/update/delete vaccination schedule.

• Administrator may get reports from the system.

User Classes and Characteristics

**Registered Users:** Any user that is registered with the system,. Registered users can book appointment and can get consultation. They can share vaccination schedule and can get their pets cured.

**Un-registered users**: Un-registered users cannot get the information.

2.4 Operating Environment

Online veterinary care system will operate on the latest versions of Google Chrome (Chrome 2018), Mozilla Firefox (version 59.0.1), and Internet Explorer (version 11). Users will be able to use the software using desktops, laptops or mobile devices.

2.5 Design and Implementation Constraints

4GB RAM (at least).

10GB of disk space.

Windows 8 operating system.

User Documentation

User should be familiar with the terms like registration and login . User can get the information through the web application.

2.7 Assumptions and Dependencies

• All Users need email id or phone number for registration.

• All Users must have internet connection and internet browser.

• There will be no server latencies.

• All governmental regulations will be considered by the system.

**External Interface Requirements**

3.1 User Interfaces

• Inputs will be entered via standard web controls such as combo box, check box, text

box, calendar, etc.

• Navigation and acceptance will be handled with buttons.

3.2 Hardware Interfaces

Web or mobile browsers will be used to access to the software.

3.3 Software Interfaces

Frameworks will be used for userinterfaces. Profile information will be gathered using web forms. ASP.NETwill be used to access the database, to validate the input, and to retrieve/display the results.

Communications Interfaces

The system will use TCPIP protocol for communication, SMTP protocol for email and HTTP

protocol for website. User form data will be transferred using HTTP-POST method and search

data will be transferred using HTTP-GET method. Password data will be encrypted.

**Registration**

**Description and Priority**

Non-registered users are required to register to perform the “login” function. This feature is of high priority.

**Stimulus/Response Sequences**

“Register” button/link click: Register data form will be displayed.

“Submit” button click: Register data will be validated, error messages will be displayed as labels or dialog boxes, successful registration will forward the user to user main page.

**Functional Requirements**

Un-registered user has to register with e-mail id or phone number and has to set a strong password through which he/she can then login into the system.

Login

Description and Priority

Registered Users are required to login in order to perform functions like “create a profile”. This feature is of high priority for registered Users and low priority for un-registered users.

Stimulus/Response Sequences

“Login” button/link click: Form will be displayed.

**Functional Requirements**

The various functional modules that can be implemented by the system will be

**Registration:-** the user must be registered.

**Login:-** customer login to the system by entering valid user id and password.

**Chapter 4. SYSTEM DESIGN**

**Overview:**

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer’s requirements and at the same time assessed for quality against a set of predefined criteria for “good” design. In the software engineering context, design focuses on four major areas of concern data, architecture, interfaces, and components.

**System Design Steps:**

Design begins with the requirements model. We work to transform this model into four levels of design detail: The Data Structure, the System Architecture, the Interface Representation, and the Component Level Detail. During each design activity, we apply basic concepts and principles that lead to high quality. The outcome of System Design is the Design Specification.

**Software Design and Software Engineering:**

Software Design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analysed and specified, software design is the first of three technical activities- design, code generation, and test- that are required to build and verify the software. Each activity transforms information in a manner that ultimately results in validated computer software.

Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design.

**Data Design:** The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity. Part of data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed.

**Architecture Design:** The architectural design defines the relationship between major structural elements of the software, the “design patterns” that can be used to achieve the requirements that have been defined for the system and the constraints that affect the way in which architectural design patterns can be applied.

**Interface Design:** The interface design describes how the software communicated within itself, with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and / or control) and a specific type of behaviour. Therefore, data and control flow diagrams provide much of the information required for interface design.

**Component Level Design:** The component level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC, and STD serve as the basis for component design.

**Design Web of Engineering:**

Jean Kaiser [KAI02] suggests the following design goals that are applicable to virtually every Web App regardless of application domain, size, or complexity:

**Simplicity:** Although it may seem old-fashioned, the aphorism “all things in moderation” applies to Web Apps. There is a tendency among some designers to provide the end-user with “too much” – exhaustive content, extreme visuals, intrusive animation, enormous Web pages, the list is long. Better to strive for moderation and simplicity.

**Consistency:** This design goal applies to virtually every element of the design model. Content should be the constructed consistently ( e.g., text formatting and font styles should be the same across all text documents, graphic art should have a consistent look, color scheme, and style).

**Identity:** The aesthetic, interface, and navigational design of a web App must be consistent with the application domain for which it is to be built. A Web site for a hip-hop group will undoubtedly have a different look and feel that a Web App designed for a financial services company.

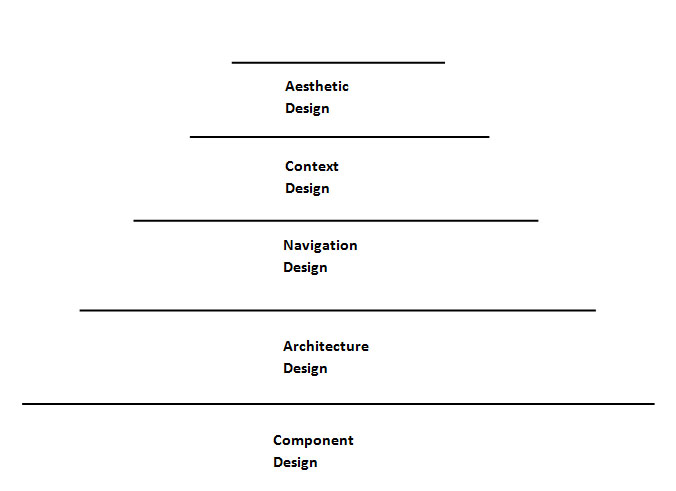
**Robustness:** Based on the identity that has been established, a Web App often makes an implicit “promise” to a user. The user expects robust content and functions that are relevant to the user’s needs. If these elements are missing or insufficient, it is likely that the Web App will fail.

**Visual Appeal:** Of all software categories, Web applications are unquestionably the most visual, the most unapologetically aesthetic.

**Compatibility:** A Web App will be used in a variety of environments (e.g., hardware, Internet connection types, operating systems, browsers) and must be designed to be compatible with each.

**The Web Design Pyramid:** What is design in the context of Web engineering? The simple question is more difficult to answer than one might believe. Design leads to a model that contains the appropriate mix of aesthetics, content and technology. The mix will vary depending upon the nature of the Web App, and as a consequence the design activities that are emphasized will also vary.

Figure (a) depicts a design pyramid for Web Engineering.



**Interface Design**

**Principles for designing User Interface:**

The fundamental principles used while designing user interface for the E-PAPER SETTING

The principle of user profiling (-- Know who your user is.)

The principle of metaphor (-- Borrow behaviors from systems familiar your Users.)

The principle of feature exposure (-- Let the user see clearly what functions are available )

The principle of coherence (-- The behavior of the program should be Internally and externally consistent.)

The principle of state visualization (-- Changes in behavior should be reflected in the appearance of the program.)

The principle of shortcuts (-- Provide both concrete and abstract ways of getting a task done. )

The principle of focus (-- Some aspects of the VI attract attention more than others do)

The principle of grammar (-- A user interface is a kind of language -- know what the rules are. )

The principle of help (-- understand the different kinds of help a user Needs.

The principle of safety (-- Let the user develop confidence by providing a safety net. )

The principle of context (-- Limit user activity to one well- defined context unless there's a good reason not to. )

The principle of aesthetics (-- Create a program of beauty)

The principle of user testing (-- Recruit help in spotting the Inevitable defects in your design. )

The principle of humility (-- Listen to what ordinary people have to say. )

User Interface Design Process

There are several phases and processes in the user interface design some of which are more demand upon than others depending on the project.

Functionality requirements gathering-- assembling a list of the functionality required of the system to accomplish the goals of the project and the potential needs of the users.

User analysis--analysis of the potential users of the system either through discussion with people who work with the users and/or the potential users themselves. Typical questions involve:

What would the user want the system to do?

How would the system fit in with the user's normal workflow or daily activities?

How technically savvy is the user and what similar systems does the user already use?

What interface look and feels appeal to the user?

**Information architecture** -- development of the process and / or information flow of the system.

**Prototyping** -- Development of wireframes, either in the form of paper prototypes or simple interactive screens. These prototypes are stripped of all look & feel elements and most content in order to concentrate on the interface.

**Graphic Interface design** --actual look & feel design of the final graphical user interface (GUI). It may be based on the findings developed during the usability testing if usability is predictable, or based on communication objectives and styles that would appeal to the user. In rare cases, the graphics may drive the prototyping, depending on the importance of visual form versus function. If the interface requires multiple skins, there may be multiple interface designs for one control panel, functional feature or widget.

This phase is often a collaborative effort between a graphic designer and a user interface designer, or handled by one who is proficient in both disciplines.

**Aesthetic Design:**

Aesthetic design, also called graphic design, is an artistic endeavor that complements the technical aspects of Web engineering. Without it, a Web App may be functional, but unappealing. With it, a Web App draws the users into a world that embraces them on a visceral, as well as an intellectual, level.

But what is aesthetic? There is an Old saying “beauty exists in the eye of the beholder.” This is particularly appropriate when aesthetic design for Web App is considered.

**Layout Issues:** Every Web page has a limited amount of “real estate” that can be used to support:

Non-functional aesthetics, navigation features, information content, and user directed functionality. The “development” of this real estate is planned during aesthetic design. Like all aesthetic issues, there are no absolute rules when screen layout is designed. However, a number of general layout guidelines are worth-considering.

1) Don’t be afraid of white spaces.

2) Emphasize content.

3) Organize layout elements from top-left to bottom-right

4) Group navigation, content and function geographically within the page.

5) Don’t extend your real estate with the scrolling bar.

6) Consider resolution and browser window size when designing layout.

Graphic Design Issues: Graphic design considers every aspect of the look and feel of a Web App. The graphics design process begins with layout and proceeds into a consideration of global colour schemes, typefaces, sizes, styles, the use of supplementary media (e.g., audio, video, animation), and all other aesthetic elements of an application.

A lot of care has been taken in this project to consider all these issues. Content has been emphasizing wherever required, lot of care taken in grouping menu items, elements have been organized from top-left to bottom-right, resolution and browser window size has been taken into consideration while designing layout. In addition, there is a global uniformity in colour schemes, typefaces, and styles.

**Content Design:**

Content design focuses on two different issues, each addressed by individuals with different skill set. Content design develops a design representation for content objects and represents the mechanism required to instantiate their relationships to one another.

In addition, content design is concerned with the representation of information within a specific content object- a design activity that is conducted by copywriters, graphic designers, and who generate the content to be used within a Web App.

In our application, we have used the following entities:

User,

BookingSlot,

Appointment,

Information,

InformationDissemination,

ConversationHistory,

Conversation

We have put the information of all the users i.e., Admin, Doctor or User all in User entity. When a doctor registers and is approved by Admin, he/she can add booking slots on which he/she is available into the system and that information is saved in BookingSlot entity. After that, a user can book appointment and that information is saved in Appointment entity. In addition, the Admin can add Vaccination schedule and other types of information like Fertility Management, Production of Hygienic Milk in his portal and that information is saved in Information entity. A user can chat with a doctor or an admin, a doctor can chat with a user or admin and admin can chat with both user and doctor and all this information is saved in Conversation and ConversationHistory entities.

As regards with UI, we have designed some master pages and child pages and we have followed this design pattern throughout the entire application. For example, we have used a master page for the main website and some child pages within this master page are home page, vaccination page, login/register page, doctor registration page and doctor verification page. Similarly, we have used a master page and child pages for user module, admin module and doctor module. We have used MVC design pattern in our application for designing the UI of the application.

**Modules:**

**Admin:**

When a doctor registers with the system, he will not be immediately available on the website. Instead, first he will be reviewed by Admin and if found genuine, then the Admin will approve the doctor. Admin can also add information like Vaccination Schedule, Fertility Management, Production of Hygienic Milk and various types of Information Dissemination into the system.

**User:**

A user who wants treatment for his pet(s) regularly or occasionally can register with the system. After registering, he can book appointment, chat with a doctor in case of emergency. A user can also see the information like vaccination schedule, camps to be organized on the website and for this he doesn’t need to register on the website.

**Doctor:**

A doctor can register with the system by entering his/her details and then he/she can be approved by admin and after approval, he/she can enter his/her booking slots available. He/she can also chat with the users and provide them necessary recommendations for treatment if they contact him/her in case of emergency.

**Data Flow Diagrams:**

**Figure 1: - Context level diagram (level 0)**

**Figure 2: - Level 1 DFD**

**Figure 3: - Entity Relationship Diagram**

Figure 1.

User

Admin

Login

Login

Acknowledge

Acknowledge

Login Acknowledge

Doctor

Figure 2.

Figure 1:-

Admin

User

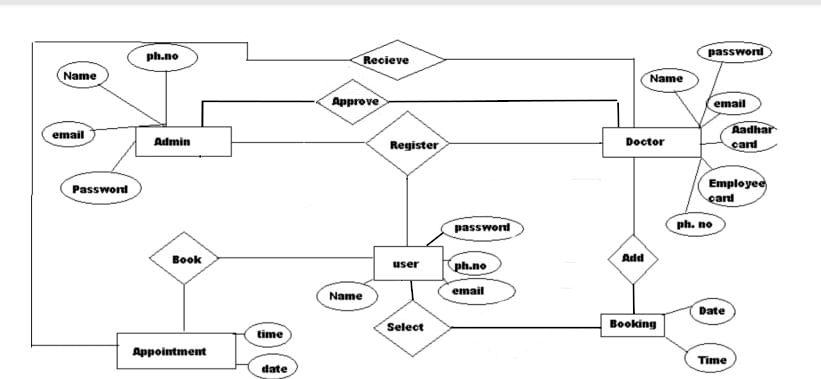
Doctor

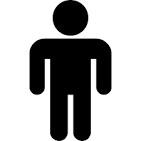
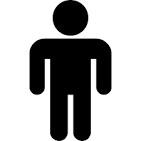
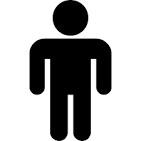
Request for booking

Apptt. (Booked/rejected)

Booking information status

Figure 3.



**Use case diagram:-**

Pet owners

Doctor

Admin

**Input /output design:**

In our project, the elements like textboxes and dropdowns have been used for input and tables and pages have been used for output.

**Chapter 5. Coding**

function doLogin() {

if ($('#username').val() == '') {

alert('Please enter username.');

$('#username').focus();

return;

}

if ($('#password').val() == '') {

alert('Please enter password.');

$('#password').focus();

return;

}

$.ajax({

url: "/api/token",

type: "POST",

data: $.param({ grant\_type: 'password', username: $("#username").val(), password: $("#password").val() }),

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

console.log(res);

localStorage.setItem('token', JSON.stringify(res));

if (res.role == 'Admin') {

window.location = '/admin';

} else if (res.role == 'Doctor') {

window.location = '/doctor';

} else {

window.location = '/user';

}

},

error: function () {

console.log('Error occured.');

alert('Invalid Username/ Password');

}

});

}

function doRegister() {

if (!/^[a-zA-Z ]+$/.test($('#name').val())) {

alert('Only letters and spaces are allowed in name.');

$('#name').focus();

return;

}

var reg = /^([A-Za-z0-9\_\-\.])+\@@([A-Za-z0-9\_\-\.])+\.([A-Za-z]{2,4})$/;

if (!reg.test($('#email').val())) {

alert('Please enter a valid email.');

$('#email').focus();

return;

}

if (!/[0-9]+/.test($('#mobile').val())) {

alert('Only digits allowed in mobile no.');

$('#mobile').focus();

return;

}

if ($('#mobile').val().length > 10 || $('#mobile').val().length < 10) {

alert('Exactly 10 digits allowed in Mobile Number.');

$('#mobile').focus();

return;

}

if ($('#passwordRegister').val() === '') {

alert('Password is required');

$('#passwordRegister').focus();

return;

}

if ($('#passwordRegister').val() != $('#confirmPassword').val()) {

alert('Passwords do not match.');

return;

}

$.ajax({

url: '/Login/GetCaptchaString',

method: 'GET',

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res !== $('#CaptchaText').val()) {

alert('Captcha not matched. Please try again...');

$('#CaptchaImage').removeAttr('src').attr('src', '/Login/ShowCaptchaImage?' + new Date().getTime());

return;

}

var userObj = {};

userObj.name = $('#name').val();

userObj.email = $('#email').val();

userObj.mobile = $('#mobile').val();

userObj.password = $('#passwordRegister').val();

userObj.isActive = true;

userObj.role = "User";

$.ajax({

url: '/api/User/Post',

data: JSON.stringify(userObj),

method: 'POST',

dataType: 'json',

contentType: "application/json; charset=utf-8",

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res.added) {

alert('Registration done successfully. Please login now to access your account.');

} else {

alert(res.message);

}

},

error: function (err) {

console.log('Error: ', err);

alert(err);

}

});

},

error: function (err) {

console.log('Error: ', err);

alert(err.statusText);

}

});

}

**Login/Register Page Frontend code**

[HttpPost]

public IHttpActionResult Post(User user)

{

user.ID = Guid.NewGuid();

user.Password = HashPassword(user.Password);

user.AddedOn = DateTime.Now;

return Ok(\_bal.RegisterUser(user));

}

UserBAL \_bal = new UserBAL();

[NonAction]

public string HashPassword(string password)

{

// Use input string to calculate MD5 hash

using (System.Security.Cryptography.MD5 md5 = System.Security.Cryptography.MD5.Create())

{

byte[] inputBytes = System.Text.Encoding.ASCII.GetBytes(password);

byte[] hashBytes = md5.ComputeHash(inputBytes);

// Convert the byte array to hexadecimal string

StringBuilder sb = new StringBuilder();

for (int i = 0; i < hashBytes.Length; i++)

{

sb.Append(hashBytes[i].ToString("X2"));

}

return sb.ToString();

}

}

**Register Backend coding**

function doRegister() {

if (!/^[a-zA-Z ]+$/.test($('#name').val())) {

alert('Only letters and spaces are allowed in name.');

$('#name').focus();

return;

}

var reg = /^([A-Za-z0-9\_\-\.])+\@@([A-Za-z0-9\_\-\.])+\.([A-Za-z]{2,4})$/;

if (!reg.test($('#email').val())) {

alert('Please enter a valid email.');

$('#email').focus();

return;

}

if (!/[0-9]+/.test($('#mobile').val())) {

alert('Only digits allowed in mobile no.');

$('#mobile').focus();

return;

}

if ($('#mobile').val().length > 10 || $('#mobile').val().length < 10) {

alert('Exactly 10 digits allowed in Mobile Number.');

$('#mobile').focus();

return;

}

if ($('#address').val() === '') {

alert('Address is required.');

$('#address').focus();

return;

}

if ($('#password').val() === '') {

alert('Password is required');

$('#password').focus();

return;

}

if ($('#passwordRegister').val() != $('#confirmPassword').val()) {

alert('Passwords do not match.');

return;

}

$.ajax({

url: '/Login/GetCaptchaString',

method: 'GET',

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res !== $('#CaptchaText').val()) {

alert('Captcha not matched. Please try again...');

$('#CaptchaImage').removeAttr('src').attr('src', '/Login/ShowCaptchaImage?' + new Date().getTime());

return;

}

var userObj = {};

userObj.name = $('#name').val();

userObj.email = $('#email').val();

userObj.mobile = $('#mobile').val();

userObj.address = $('#address').val();

userObj.password = $('#passwordRegister').val();

userObj.isActive = false;

userObj.role = "Doctor";

$.ajax({

url: '/api/User/Post',

data: JSON.stringify(userObj),

method: 'POST',

dataType: 'json',

contentType: "application/json; charset=utf-8",

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res.added) {

alert('Registration done successfully. Please login now to access your account.');

} else {

alert(res.message);

}

},

error: function (err) {

console.log('Error: ', err);

alert(err.statusText);

}

});

},

error: function (err) {

console.log('Error: ', err);

alert(err.statusText);

}

});

}

**Doctor Registration frontend code**

var username = JSON.parse(localStorage.getItem('token')).username;

var identityCardImagePath = '';

var aadharCardImagePath = '';

var \_URL = window.URL || window.webkitURL;

$("#fupImageIdentityCard").on('change', function () {

var file, img;

if ((file = this.files[0])) {

img = new Image();

img.onload = function () {

sendFileIdentityCard(file);

};

img.onerror = function () {

alert("Not a valid file:" + file.type);

};

img.src = \_URL.createObjectURL(file);

}

});

function sendFileIdentityCard(file) {

if (!$('#fupImageIdentityCard').val().endsWith('.jpg') &&

!$('#fupImageIdentityCard').val().endsWith('.JPG') &&

!$('#fupImageIdentityCard').val().endsWith('.jpeg') &&

!$('#fupImageIdentityCard').val().endsWith('.JPEG') &&

!$('#fupImageIdentityCard').val().endsWith('.png') &&

!$('#fupImageIdentityCard').val().endsWith('.PNG')) {

alert('Only jpg, JPG, jpeg, JPEG, png and PNG file types allowed.');

$('#fupImageIdentityCard').focus();

return;

}

var formData = new FormData();

formData.append('file', $('#fupImageIdentityCard')[0].files[0]);

identityCardImagePath = '/DoctorDocuments/' + username + '/IdentityCard/' + $('#fupImageIdentityCard')[0].files[0].name;

formData.append('username', username);

$.ajax({

type: 'post',

url: '/api/User/UploadIdentityCardImage',

data: formData,

xhr: function () { // Custom XMLHttpRequest

var myXhr = $.ajaxSettings.xhr();

if (myXhr.upload) { // Check if upload property exists

//update progressbar percent complete

$('#statustxtIdentityCard').html('0%');

// For handling the progress of the upload

myXhr.upload.addEventListener('progress', progressHandlingFunctionIdentityCard, false);

}

return myXhr;

},

success: function (status) {

if (status != 'error') {

$('#statustxtIdentityCard').html('');

$('#progressPercentIdentityCard').css('width', '0%');

$('#photoIdentityCard').attr('src', identityCardImagePath);

}

},

processData: false,

contentType: false,

error: function (xhr) {

alert("Some error occured... " + xhr.statusText);

}

});

}

function progressHandlingFunctionIdentityCard(e) {

if (e.lengthComputable) {

var percentage = Math.floor((e.loaded / e.total) \* 100);

//update progressbar percent complete

$('#statustxtIdentityCard').html(percentage + '%');

$('#progressPercentIdentityCard').css('width', percentage + '%');

console.log("Value = " + e.loaded + " :: Max =" + e.total);

}

}

$("#fupImageAadharCard").on('change', function () {

var file, img;

if ((file = this.files[0])) {

img = new Image();

img.onload = function () {

sendFileAadharCard(file);

};

img.onerror = function () {

alert("Not a valid file:" + file.type);

};

img.src = \_URL.createObjectURL(file);

}

});

function sendFileAadharCard(file) {

if (!$('#fupImageAadharCard').val().endsWith('.jpg') &&

!$('#fupImageAadharCard').val().endsWith('.JPG') &&

!$('#fupImageAadharCard').val().endsWith('.jpeg') &&

!$('#fupImageAadharCard').val().endsWith('.JPEG') &&

!$('#fupImageAadharCard').val().endsWith('.png') &&

!$('#fupImageAadharCard').val().endsWith('.PNG')) {

alert('Only jpg, JPG, jpeg, JPEG, png and PNG file types allowed.');

$('#fupImageAadharCard').focus();

return;

}

var formData = new FormData();

formData.append('file', $('#fupImageAadharCard')[0].files[0]);

aadharCardImagePath = '/DoctorDocuments/' + username + '/AadharCard/' + $('#fupImageIdentityCard')[0].files[0].name;

formData.append('username', username);

$.ajax({

type: 'post',

url: '/api/User/UploadAadharCardImage',

data: formData,

xhr: function () { // Custom XMLHttpRequest

var myXhr = $.ajaxSettings.xhr();

if (myXhr.upload) { // Check if upload property exists

//update progressbar percent complete

$('#statustxtAadharCard').html('0%');

// For handling the progress of the upload

myXhr.upload.addEventListener('progress', progressHandlingFunctionAadharCard, false);

}

return myXhr;

},

success: function (status) {

if (status != 'error') {

$('#statustxtAadharCard').html('');

$('#progressPercentAadharCard').css('width', '0%');

$('#photoAadharCard').attr('src', aadharCardImagePath);

}

},

processData: false,

contentType: false,

error: function (xhr) {

alert("Some error occured... " + xhr.statusText);

}

});

}

function progressHandlingFunctionAadharCard(e) {

if (e.lengthComputable) {

var percentage = Math.floor((e.loaded / e.total) \* 100);

//update progressbar percent complete

$('#statustxtAadharCard').html(percentage + '%');

$('#progressPercentAadharCard').css('width', percentage + '%');

console.log("Value = " + e.loaded + " :: Max =" + e.total);

}

}

function saveDocuments() {

if ($('#fupImageIdentityCard').val() == '') {

alert('Please upload Identity Card image first.');

$('#fupImageIdentityCard').focus();

return;

}

if ($('#fupImageAadharCard').val() == '') {

alert('Please upload Aadhar Card image first.');

$('#fupImageAadharCard').focus();

return;

}

var obj = {};

obj.identityCardImagePath = '/DoctorDocuments/' + username + '/IdentityCard/' + $('#fupImageIdentityCard')[0].files[0].name;

obj.aadharCardImagePath = '/DoctorDocuments/' + username + '/AadharCard/' + $('#fupImageAadharCard')[0].files[0].name;

obj.mobile = username;

$.ajax({

type: "POST",

url: "/api/User/SaveDoctorVerificationImages",

data: JSON.stringify(obj),

contentType: "application/json; charset=utf-8",

dataType: "json",

beforeSend: function () {

$("#imageLoading").show();

},

complete: function () {

$("#imageLoading").hide();

},

success: function (r) {

alert(r.message);

},

error: function (x) {

swal('Oops!', x.statusText, 'error');

}

});

}

**Doctor Verification frontend code**

[HttpPost]

[Route("api/User/UploadIdentityCardImage")]

public IHttpActionResult UploadIdentityCardImage()

{

try

{

var httpRequest = HttpContext.Current.Request;

if (httpRequest.Files.Count > 0)

{

foreach (string file in httpRequest.Files)

{

var postedFile = httpRequest.Files[file];

var username = httpRequest.Form["username"];

var directory = new DirectoryInfo(HttpContext.Current.Server.MapPath("~/DoctorDocuments/" + username + "/IdentityCard/"));

if (!directory.Exists)

directory.Create();

var filePath = HttpContext.Current.Server.MapPath("~/DoctorDocuments/" + username + "/IdentityCard/" + postedFile.FileName);

postedFile.SaveAs(filePath);

}

}

return Ok(true);

}

catch (Exception ex)

{

return Ok(false);

}

}

[HttpPost]

[Route("api/User/UploadAadharCardImage")]

public IHttpActionResult UploadAadharCardImage()

{

try

{

var httpRequest = HttpContext.Current.Request;

if (httpRequest.Files.Count > 0)

{

foreach (string file in httpRequest.Files)

{

var postedFile = httpRequest.Files[file];

var username = httpRequest.Form["username"];

var directory = new DirectoryInfo(HttpContext.Current.Server.MapPath("~/DoctorDocuments/" + username + "/AadharCard/"));

if (!directory.Exists)

directory.Create();

var filePath = HttpContext.Current.Server.MapPath("~/DoctorDocuments/" + username + "/AadharCard/" + postedFile.FileName);

postedFile.SaveAs(filePath);

}

}

return Ok(true);

}

catch (Exception ex)

{

return Ok(false);

}

}

**Doctor Verification backend code**

function approveDoctor(mobile) {

$.ajax({

url: '/api/User/ApproveDoctor?mobile=' + mobile,

method: 'POST',

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res.added) {

alert(res.message);

location.reload();

} else {

alert(res.message);

}

},

error: function (err) {

console.log('Error: ', err);

alert(err);

}

});

}

function rejectDoctor(mobile) {

$.ajax({

url: '/api/User/RejectDoctor?mobile=' + mobile,

method: 'POST',

beforeSend: function () {

$('#loading').show();

},

complete: function () {

$('#loading').hide();

},

success: function (res) {

if (res.added) {

alert(res.message);

location.reload();

} else {

alert(res.message);

}

},

error: function (err) {

console.log('Error: ', err);

alert(err);

}

});

}

function showDocument(imagePath, documentTitle) {

$('#documentTitle').text(documentTitle);

$('#documentImage').attr('src', imagePath);

$('#documentModal').modal('show');

}

Approve, Reject doctor and see doctor verification documents coding frontend

[HttpPost]

[Route("api/User/ApproveDoctor")]

public IHttpActionResult ApproveDoctor(string mobile)

{

return Ok(\_bal.ApproveDoctor(mobile));

}

[HttpPost]

[Route("api/User/RejectDoctor")]

public IHttpActionResult RejectDoctor(string mobile)

{

return Ok(\_bal.RejectDoctor(mobile));

}

**Approve, reject doctor backend code**

**Chapter 6. Testing**

**Introduction**

Software testing is a critical element of the ultimate review of specification design and coding. Testing of software leads to the uncovering of errors in the software functional and performance requirements are met. Testing also provides a good indication of software reliability and software quality as a whole. The result of different phases of testing are evaluated and then compared with the expected results. If the errors are uncovered they are debugged and corrected. A strategy approach to software testing has the generic characteristics:

Testing begins at the module level and works “outwards” towards the integration of the entire computer based system.

Different testing techniques are appropriate at different points of time.

Testing and debugging are different activities, but debugging must be accommodating in the testing strategy.

**Goals & Objectives:**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error. Our Objective is to design test processes that systematically uncover different classes of errors and do so with minimum amount of time and effort.

**Statement of Scope:**

A description of the scope of the software testing is developed. All the features to be tested are noted as follows. The basic principles that guides software testing are:

All test cases should be traceable top customer requirements. The most severe defects from the customer’s point of view are those that cause the program to fail to meet its requirements.

Test case should be planned long before testing begins. Testing plan can begin as soon as the requirement model is complete. Detailed definition of the test cases can begin as soon as the design is solidified. Therefore, the entire test can be planned before any code has been generated.

Testing should begin “in the small” and progress towards “in the large”. The first test planned and executed generally focus on the individual modules. As testing progresses testing shifts focus in an attempt to find errors in integrating clusters of modules and ultimately in the entire system.

**Testing Principles:**

The basic principles that guide software testing are:

All the cases should be traceable top customer requirements. The most severe defects from the customer’s point of view are those that cause the program to fail to meet its requirements.

Test case should be planned long before testing begins. Testing plan can begin as soon as the requirement model is complete. Detailed definition of the test cases can begin as soon as the design is solidified. Therefore, all the test can be planned before code has been generated.

The Pareto Principle applies to software testing. Stated simply the Pareto Principle implies that 80% of all errors uncovered during testing will likely to be traceable to 20% of all program modules. The program of course is to isolate these suspect modules and to thoroughly test them.

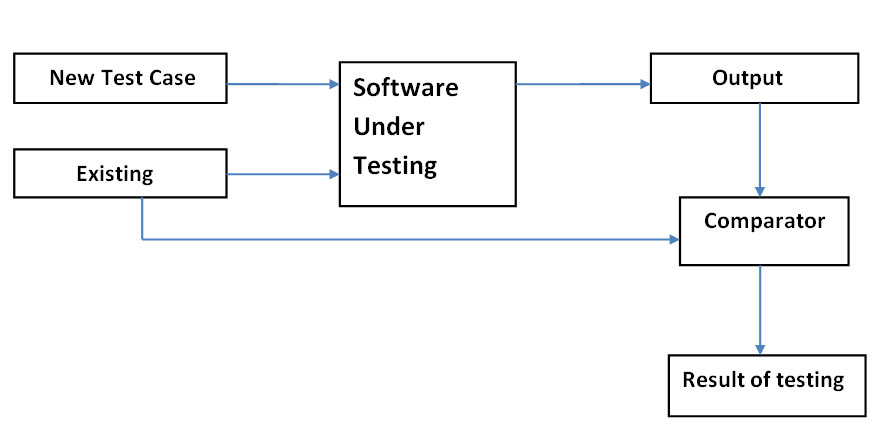
Testing should begin “in the small” and “in the large”. The first test planned

and executed generally focus on the individual modules. As testing progresses testing shifts focus in an attempt to find errors in integration clusters of modules and ultimately in the entire system.

Exhaustive testing is not possible. The number of paths permutations for even a moderately sized program is exceptionally large. For this reason, it is impossible to execute every combination of path during testing. It is possible however to ensure that all conditions in the procedural design have been exercised. To be most effective an independent third party should conduct testing. The third party has the highest probability of finding the errors.

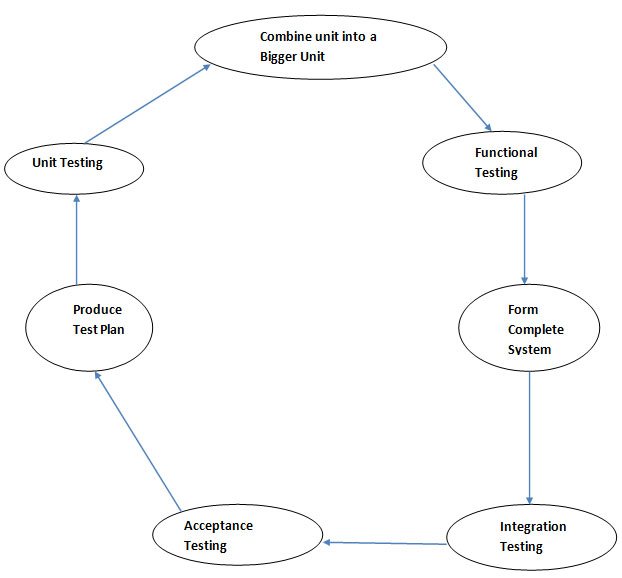
Test case: Before the project is released, it has to pass through a test cases suit, so that the required functionality is met and previous functionality of the system is also not broken to do this, there is an existing test cases which checks the previous functionality. New test cases are prepared and added to this existing test suit to check for the added functionality.

A pictorial representation of this can be shown as follows:



**Testing Process:**

The testing process can be shown as:



**6.1 Testing Approaches Used in Project:**

The module interface is tested to ensure that information properly flows into and out of the program unit under test. The unit testing is normally considered as an adjunct step to coding step. Because modules are not a standalone program, drivers and/or stubs software must be developed for each unit. A driver is nothing more than a “main program” that accepts test cases data and passes it to the module. A stub serves to replace the modules that subordinate to the modules to be tested. A stub may do minimal data manipulation, prints verification of entry and returns.

**Functional Test:** Each part of the code was tested individually and the pages were tested individually on all platforms to see if they are working properly.

**Performance Test:** These determines the amount of execution time spent on various parts of units and the resulting throughput, response time given by the module. Our project takes less time in executing any action. It also works on slow networks like 2g.

**Stress Test:** A lot of test files were made to work at the same time in order to check how much workloads can the unit bear.

**Structure Test:** These tests were made to check the internal logic of the program and traversing particular execution paths.

**Integration Test:**

“If they all work individually; they should work when we put them together.” The problem of course is “putting them together “. This can be done in two ways:

**Top Down Integration:** Modules are integrated by moving downwards through the control hierarchy, beginning with main control module are incorporated into the structure in either a depth first or breadth first manner.

**Bottom Up Integration:** It begins with construction and testing with atomic modules i.e modules at the lowest level of the program structure. Because modules are integrated from the bottom up, processing required for the modules subordinate to a given level is always available and the need of stubs is eliminated.

**Validation Test:** Validation succeeds when software functions in a manner that can be reasonably expected by the customer. It covers the following:

**Validation Test Criteria:** Performance, functional characteristics and uncovered deviation from specification.

**Configuration Review:** Ensures that all the elements of software configuration have been properly developed catalogued and have support for the maintenance phase of software life cycle.

**Alpha-Beta Testing:** Alpha test is conducted by developer’s site by customer. Beta test is conducted at one or more customer site by software end user.

**Modular Integration Testing:** Modular integration testing is done to ensure that the module is working independently. The inputs as required by the module are given as required and the output is tested as per the specifications.

We tested the application for all kinds of bugs and errors. We tested each page individually. We also tested the application as a whole. The validation testing was also done and all the validation criteria was met. The load balancing testing was also done to make sure that the application does not hang or crash when there are large number of users. User Interface testing was also done to make sure that UI was consistent throughout the application.

**6.2. Test Cases**

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S. No | Test Case Name | Description | Test Case Date | Input | Output | Results | Debugging |
| 1 | Login | Login using mobile/email and password | 10-10-2020 | Email/Mobile, Password | If email/mobile and/or password incorrect, displays incorrect username/password | If successful, redirects to appropriate module. |  |
| 2 | Register | Register as a user | 10-10-2020 | Name, Email, Mobile, Password, Confirm Password | If registration successful, displays the message “Registration done successfully” | If registration unsuccessful, displays message “Error occurred. Please try again.” |  |
| 3 | Doctor Register | Register as a doctor | 11-10-2020 | Name, Email, Mobile, Address, Official Email, Hospital Address | If registration successful, displays message “Registration done successfully” | If error occurs, displays message “Error occurred. Please try again…” |  |
| 4 | Approve/Reject Doctor | Admin approve/rejects a doctor | 11-10-2020 | List of doctors, button against each doctor | If action was successful, displays appropriate message | If approved, doctor can login and add booking slots. |  |

**6.3. Testing Techniques**

**Functional vs. Non-functional Testing**

The goal of utilizing numerous testing methodologies in your development process is to make sure your software can successfully operate in multiple environments and across different platforms. These can typically be broken down between functional and non-functional testing. Functional testing involves testing the application against the business requirements. It incorporates all test types designed to guarantee each part of a piece of software behaves as expected by using uses cases provided by the design team or business analyst. These testing methods are usually conducted in order and include:

* Unit testing
* Integration testing
* System testing
* Acceptance testing

Non-functional testing methods incorporate all test types focused on the operational aspects of a piece of software. These include:

* Performance testing
* Security testing
* Usability testing
* Compatibility testing

The key to releasing high quality software that can be easily adopted by your end users is to build a robust testing framework that implements both functional and non-functional software testing methodologies.

**Unit Testing**

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. TestLeft is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

**Integration Testing**

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

**System Testing**

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

**Acceptance Testing**

Acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance with all of the original business criteria and that it meets the end user’s needs. This requires the product be tested both internally and externally, meaning you’ll need to get it into the hands of your end users for beta testing along with those of your QA team. Beta testing is key to getting real feedback from potential customers and can address any final usability concerns.

**Performance Testing**

Performance testing is a non-functional testing technique used to determine how an application will behave under various conditions. The goal is to test its responsiveness and stability in real user situations. Performance testing can be broken down into four types:

* **Load testing** is the process of putting increasing amounts of simulated demand on your software, application, or website to verify whether or not it can handle what it’s designed to handle.
* **Stress testing** takes this a step further and is used to gauge how your software will respond at or beyond its peak load. The goal of stress testing is to overload the application on purpose until it breaks by applying both realistic and unrealistic load scenarios. With stress testing, you’ll be able to find the failure point of your piece of software.
* **Endurance testing,** also known as soak testing, is used to analyze the behavior of an application under a specific amount of simulated load over longer amounts of time. The goal is to understand how your system will behave under sustained use, making it a longer process than load or stress testing (which are designed to end after a few hours). A critical piece of endurance testing is that it helps uncover memory leaks.
* **Spike testing** is a type of load test used to determine how your software will respond to substantially larger bursts of concurrent user or system activity over varying amounts of time. Ideally, this will help you understand what will happen when the load is suddenly and drastically increased.

**Security Testing**

With the rise of cloud-based testing platforms and cyber attacks, there is a growing concern and need for the security of data being used and stored in software. Security testing is a non-functional software testing technique used to determine if the information and data in a system is protected. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses. There are multiple types of this testing method, each of which aimed at verifying six basic principles of security:

1. Integrity
2. Confidentiality
3. Authentication
4. Authorization
5. Availability
6. Non-repudiation

**Usability Testing**

Usability testing is a testing method that measures an application’s ease-of-use from the end-user perspective and is often performed during the system or acceptance testing stages. The goal is to determine whether or not the visible design and aesthetics of an application meet the intended workflow for various processes, such as logging into an application. Usability testing is a great way for teams to review separate functions, or the system as a whole, is intuitive to use.

**Compatibility Testing**

Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that your software’s functionality is consistently supported across any environment you expect your end users to be using.

**Testing with TestComplete**

TestComplete is our robust automated GUI testing tool that excels in compatibility and integration testing. It helps QA teams create and run tests across desktop, mobile, and web applications – enabling testing professionals to speed up delivery cycles and improve software quality. Testcomplete comes with built-in support for various test environments, integrations to performance testing tools, as well as support for developer friendly SCMs, allowing you to seamlessness integrate it into your development process. Using TestComplete will enable you to build a robust testing framework that utilizes the broad spectrum of available software testing methodologies.

**Chapter 7. IMPLEMENTATION OF PROJECT**

**Introduction to ASP.NET:**

ASP.NET is a framework for developing Web Applications using either C# or VISUAL Basic as the backend programming language .it was developed by Microsoft when they developed the .NET framework in 2003 .it uses the concept if web forms which means that the web pages are designed in a way that the developer who has already developed Windows application with windows form can easily develop Web Application with Web Forms provided by ASP.NET

**Introduction to C#:**

C# is a high level programming language which was developed by Microsoft in early 1990’s .it uses OOPS concept like abstraction, encapsulation, overriding, overloading etc. this application is largely used in web development for the application written in ASP.NET .it is also used to develop Windows Application using either the windows forms or the Windows Presentation Technology .it can also be used to create services, API’S etc.

**Introduction to Visual Studio:**

It is an integrated Development Environment which can be used to develop Windows Applications, Web Applications, and Mobile Applications.

**Software Requirements**

**Operating System:** Windows 7/8/10.

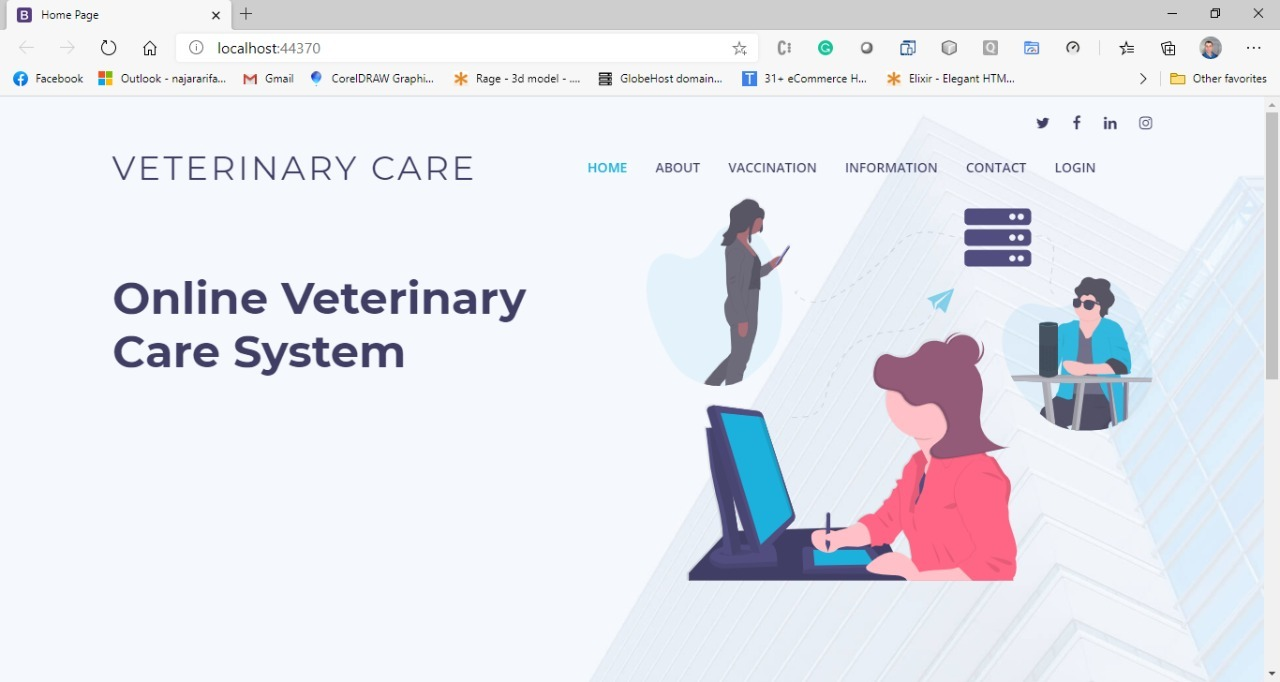
**Tools:** SQL Server 2012 or Later. Microsoft Visual Studio 2019

**Hardware Requirements:** Intel Core i3 or higher.

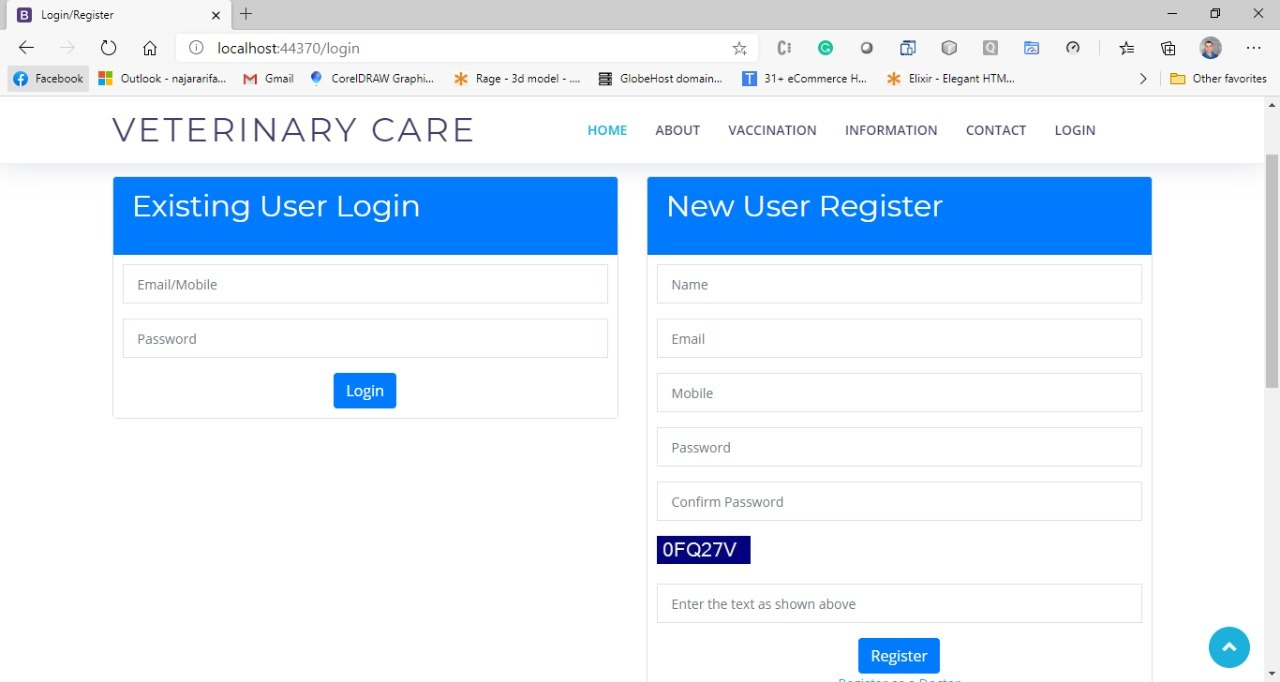
8GB of RAM or higher.

100GB of Hard Disk Space or higher in case there is too much data stored in database.

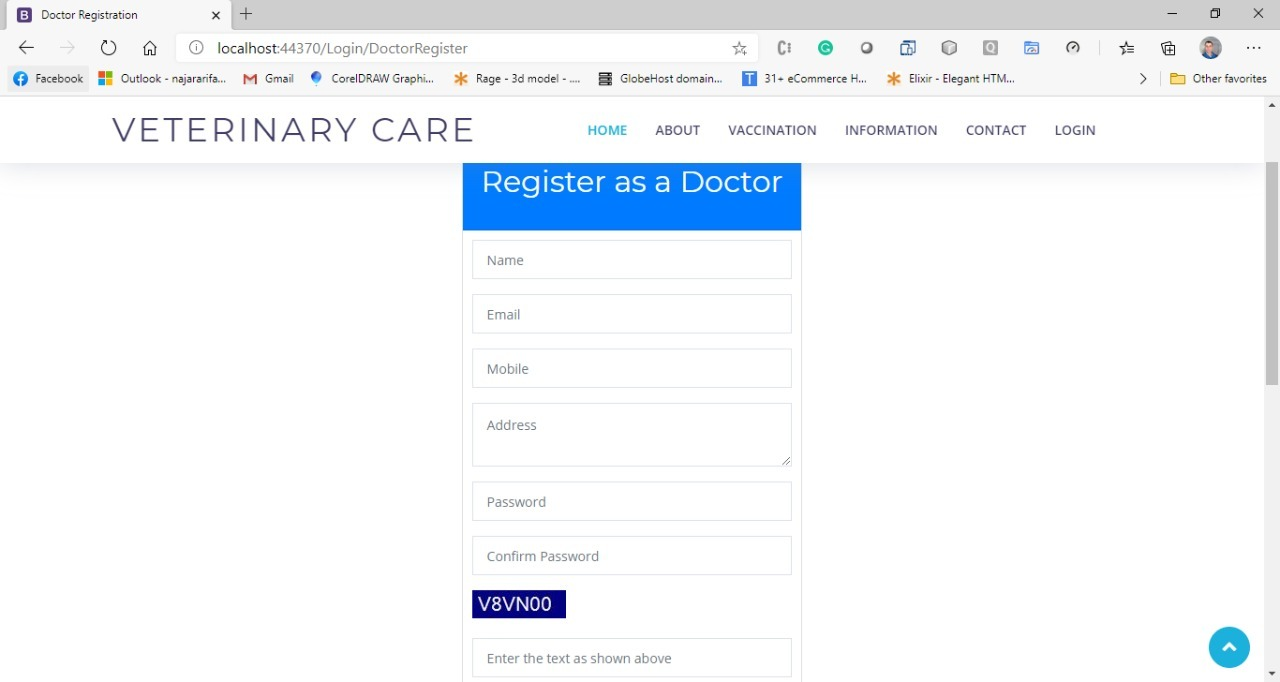
**Chapter 8. SNAPSHOTS**



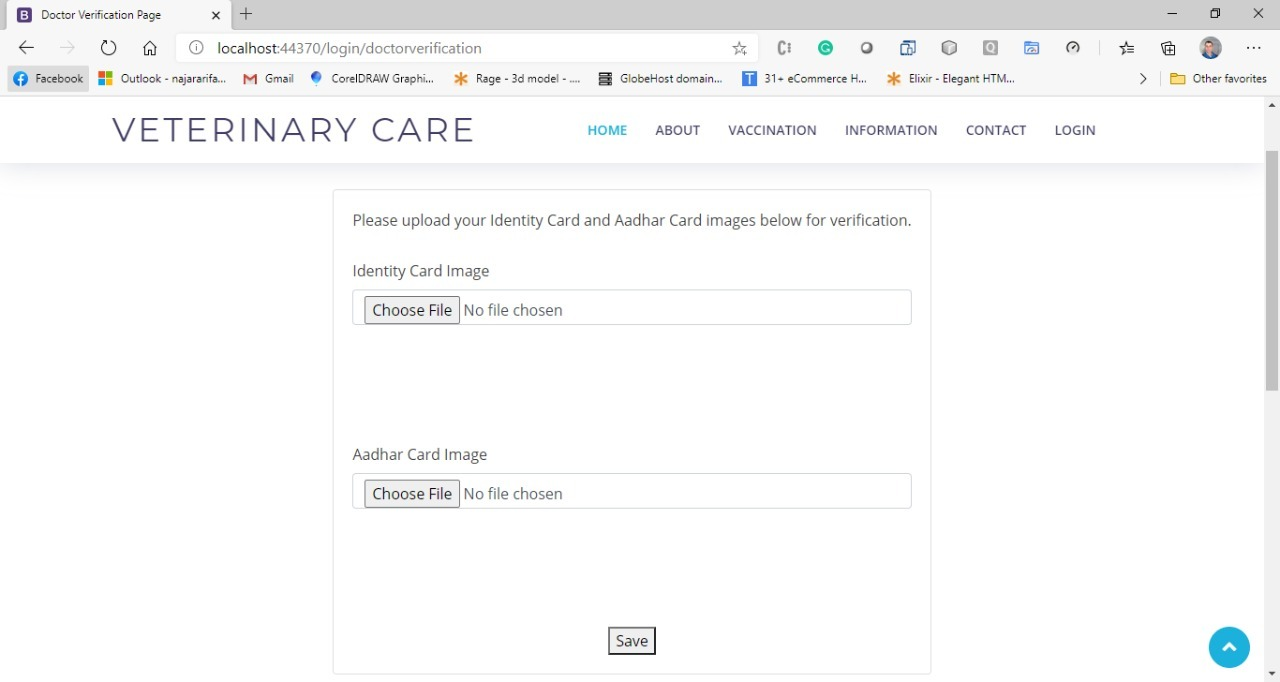
**Home Page**



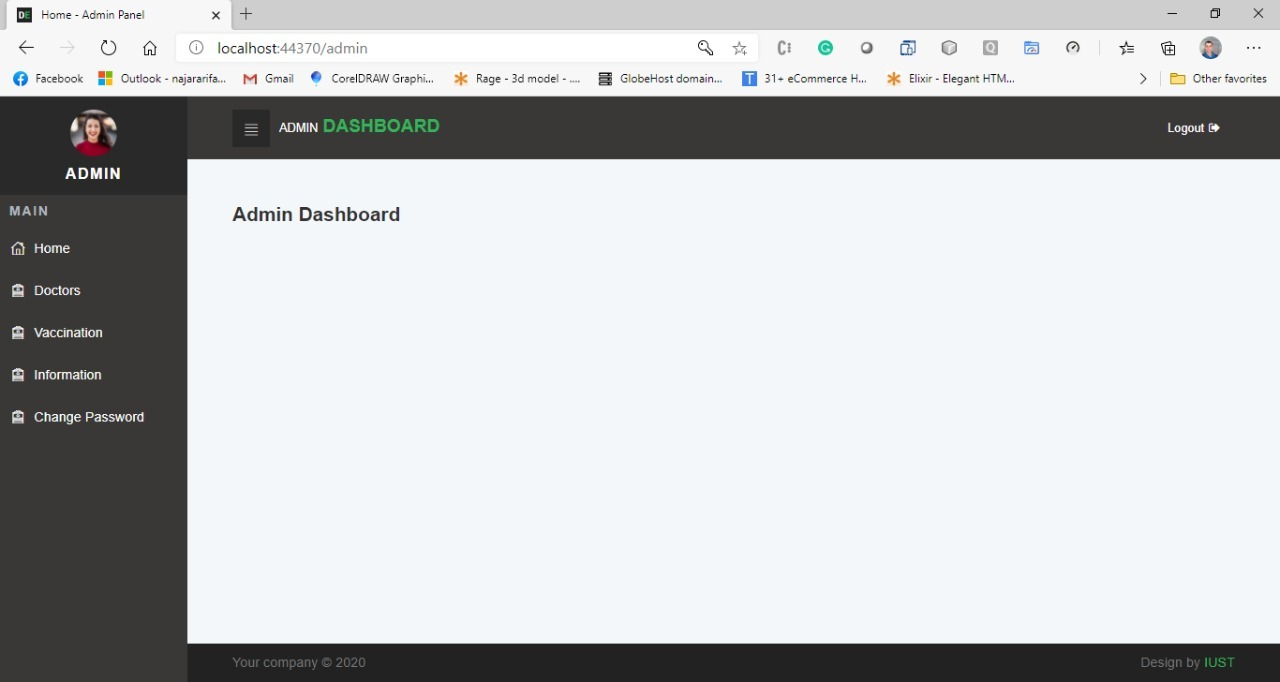
**Login/Register Page**



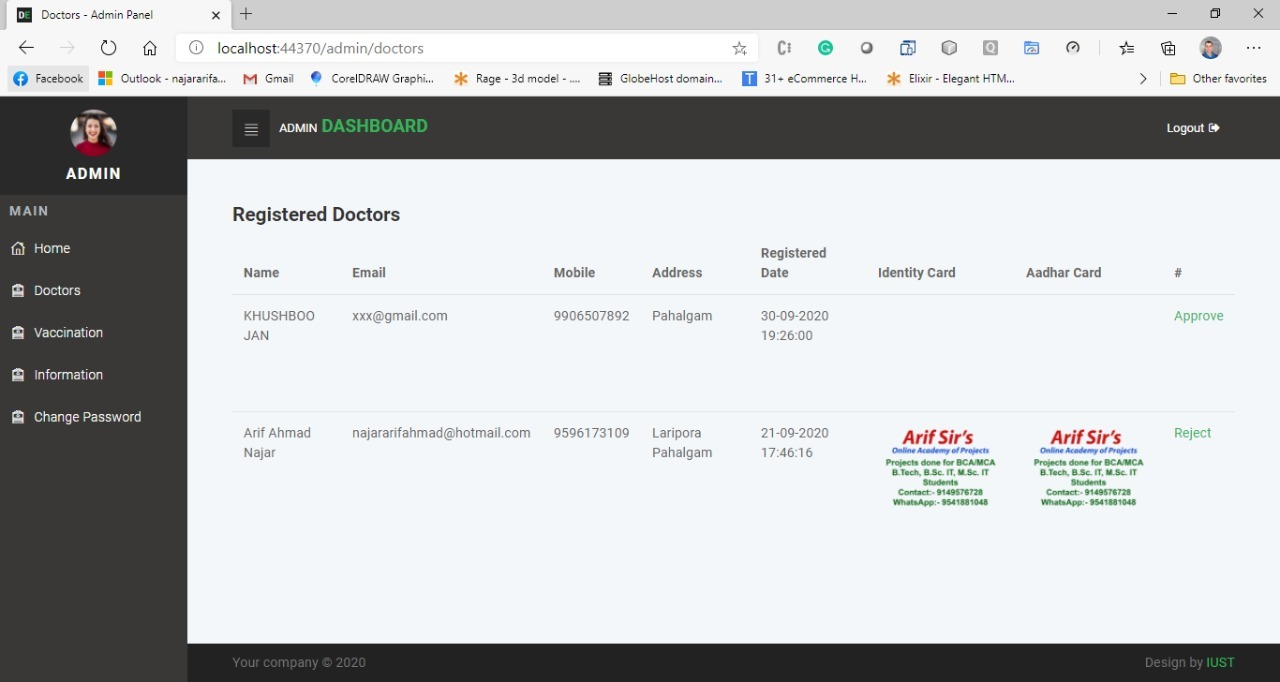
**Doctor Registration Page**



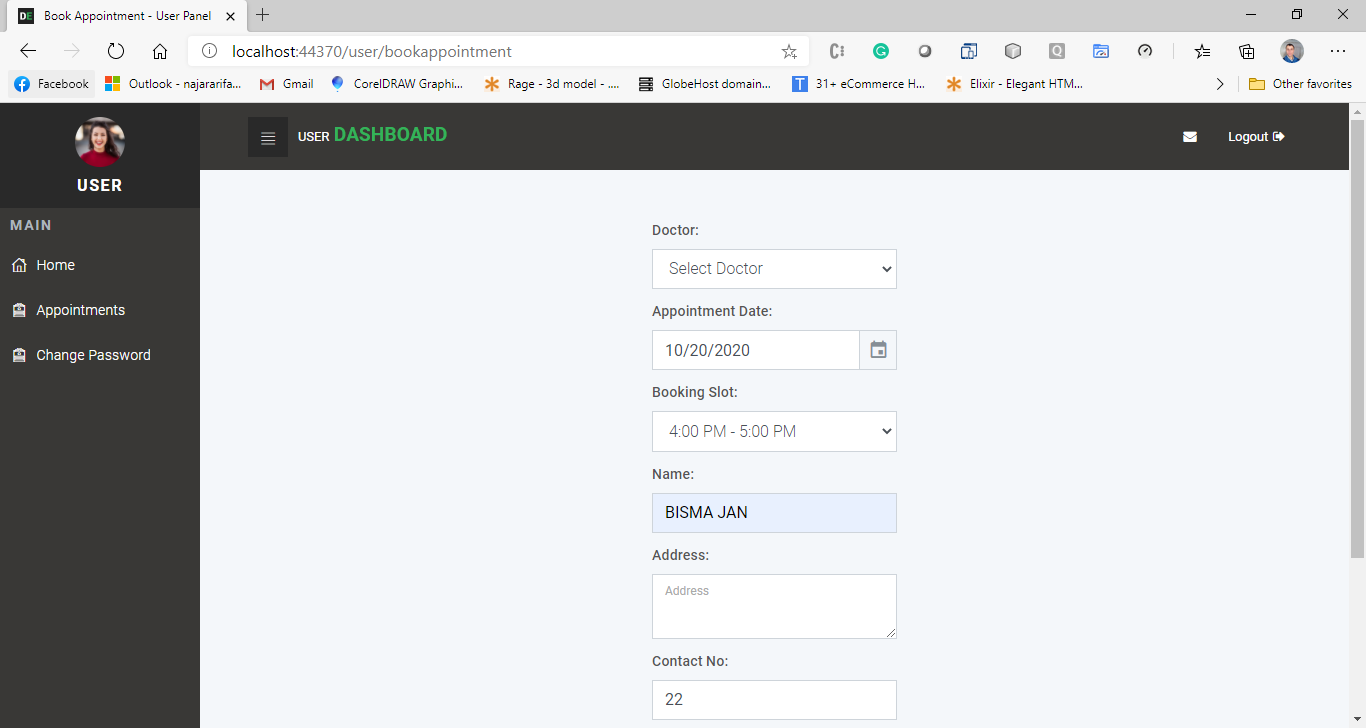
**Doctor Verification Page**



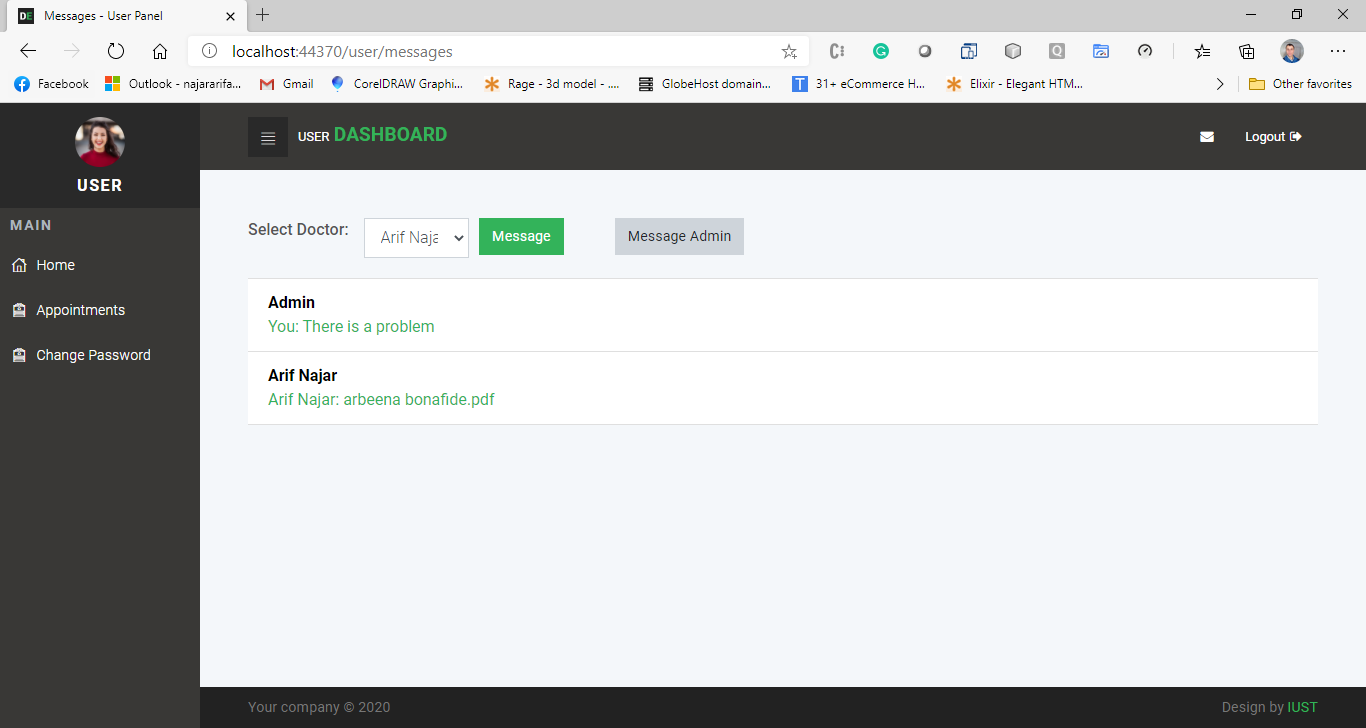
**Admin Dashboard**



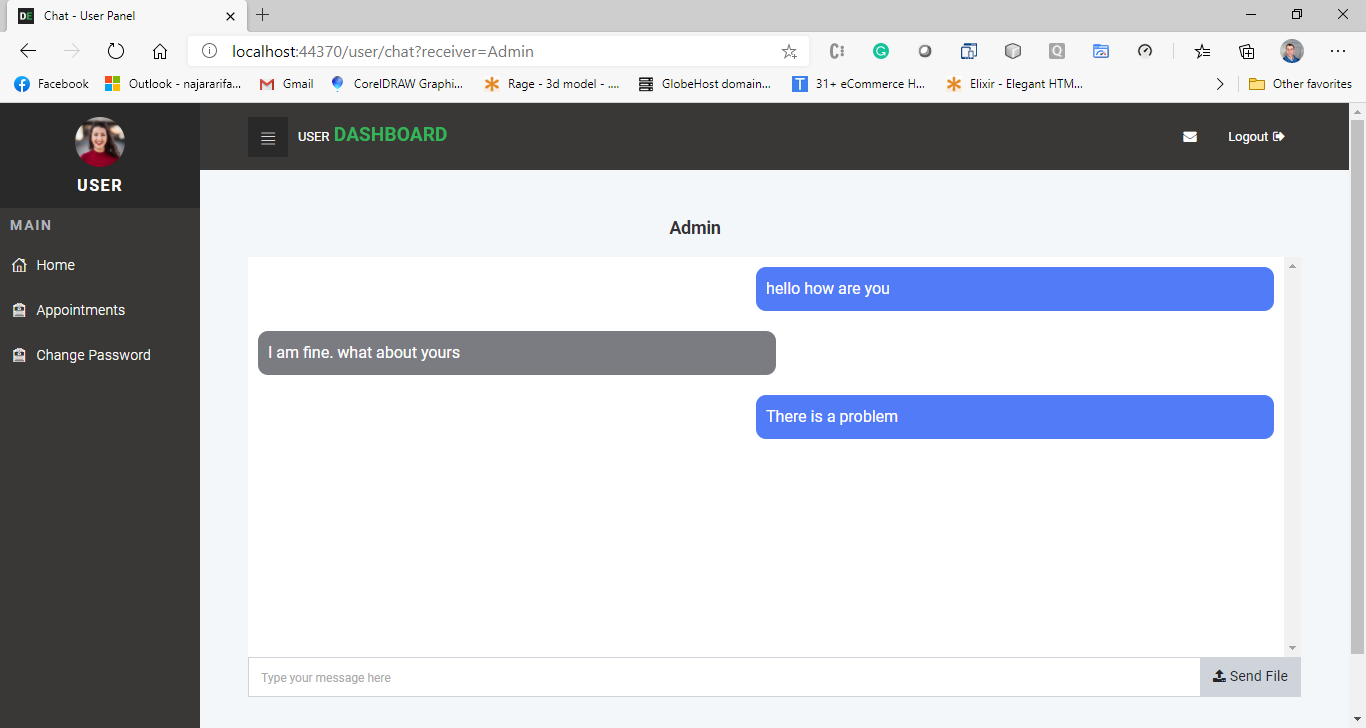
**Registered Doctors as seen by Admin**



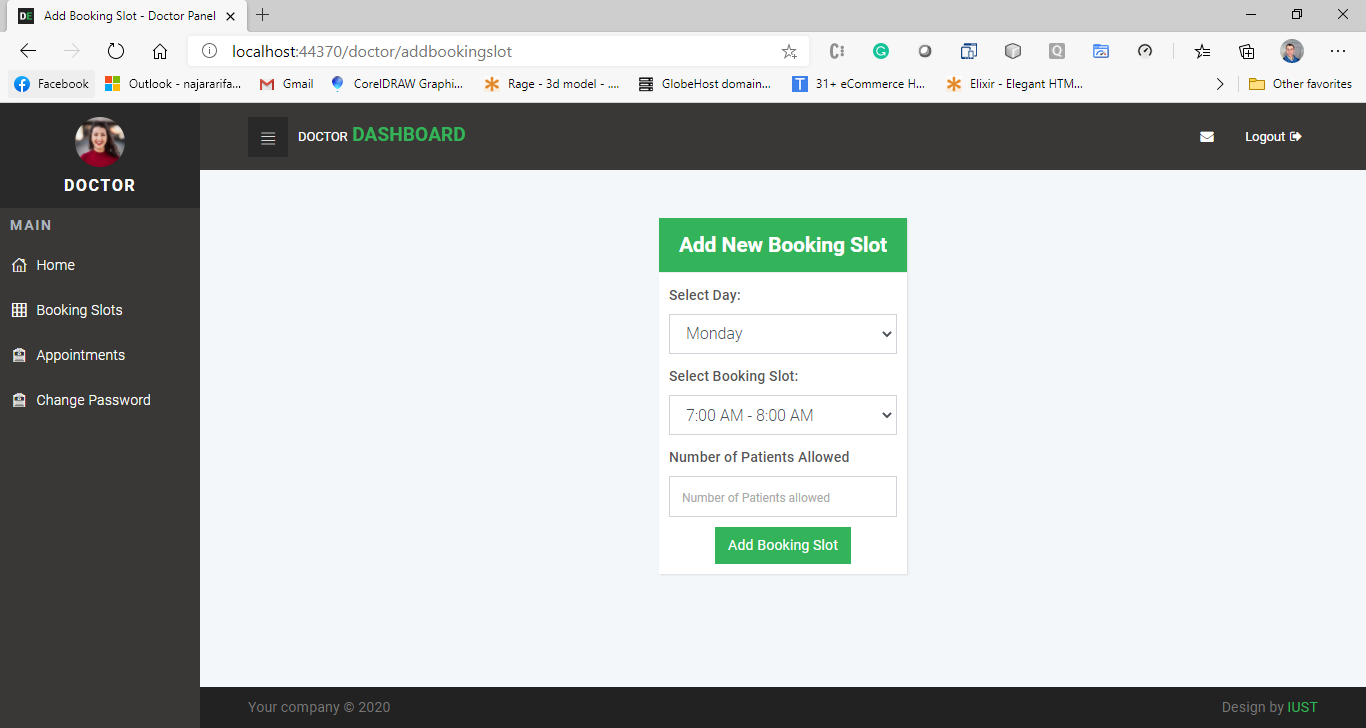
**Book Appointment Page**



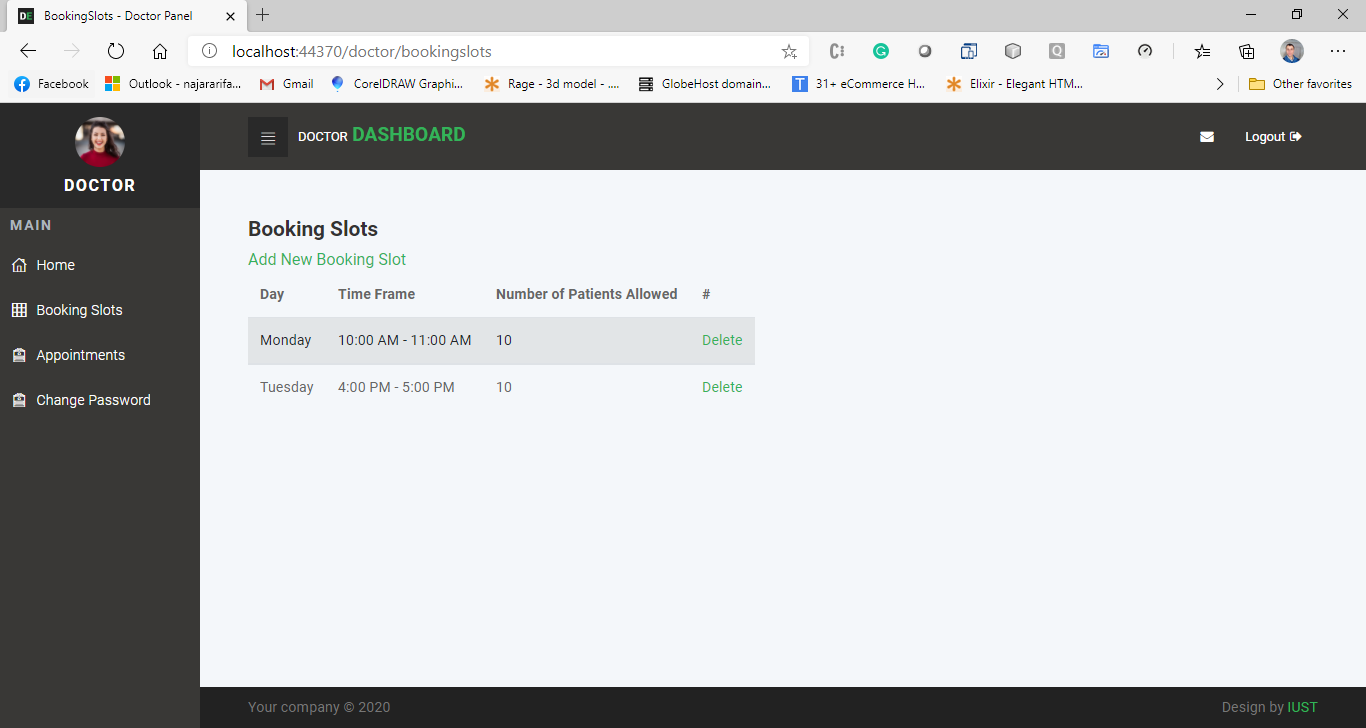
**Messages Page**



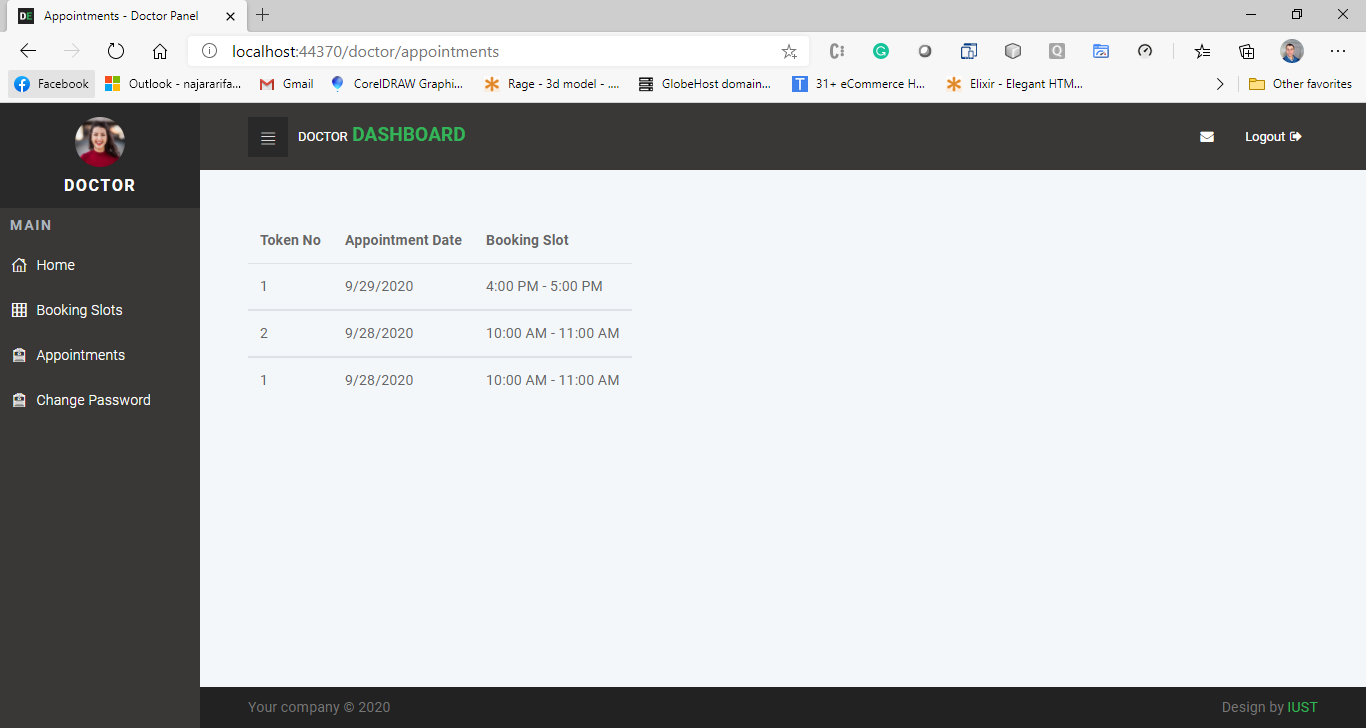
**Chat Page**



**Add Booking Slot Page**



**View Booking Slots Page**



**Doctor View Appointments Page**

**Chapter 9. CONCLUSION**

The main purpose of this project is:

To enable the user to book the appointments with the veterinary doctors to cure their pets.

To help users to chat with doctors instantly in case of emergency.

To help users to get enough info regarding vaccination schedule, camps to be organised etc.

**Chapter 10. FUTURE WORK/RECOMMENDATIONS**

In future we will add some more features like enabling the doctor to make his/her booking slots inactive if the doctor is not available. User will also be able to cancel or reschedule appointments. Chat system will also be made good.

**Chapter 11. Appendices**

**Appendix A: Goals of the system**

The main goals of this system are to provide users with an easy to use system to contact doctors to cure their pets either through booking appointments or chatting with the doctors. In case there is some serious problem, the user can contact with the admin also. It will allow only authentic doctors and users to enter the system. Users will get very much information about curing their pets rather than searching for a doctor offline.

**Appendix B: Technologies used in the system**

This system uses ASP.NET MVC and RESTful Web Api’s provided by ASP.NET MVC architecture to achieve the goals described above. It uses C# as the server side language and Microsoft SQL Server as the DBMS for database.

**Appendix C: Limitations of the system**

This application has the limitation that the user cannot rebook or cancel an appointment which will be addressed in the future releases. Any person can also access the register api with the help of POSTMAN or any other REST api tool to register the user without the verification. This issue can be resolved by using an OTP provided in the SMS to the user’s mobile and that will also be addressed in future releases.

**Chapter 12. Bibliography**

<https://smartbear.com/learn/automated-testing/software-testing-methodologies/>

<https://www.guru99.com/test-case.html>

<https://www.tutorialspoint.com/software_testing_dictionary/software_requirement_specification.htm>

<https://www.geeksforgeeks.org/analysis-modelling-in-software-engineering/#:~:text=Analysis%20Model%20is%20a%20technical,design%20in%20the%20design%20modeling>.