

IMEDSLIFE AVANTEL PROJECT WITH NEXTJS

PROJECT REPORT submitted in partial fulfillment of the requirements

Submitted by

Rizwanullah Mohammad (208W1A1299)

Under the guidance of

Dr. M. Suhasini

Asst Professor – V R Siddhartha Engineering college

For the award of the degree

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

V R SIDDHARTHA ENGINEERING COLLEGE

(AUTONOMOUS – AFFILIATED TO JNTU – K, KAKINADA)

Approved by AICTE & Accredited by NBA

KANURU, VIJAYAWADA – 7

ACADEMIC YEAR

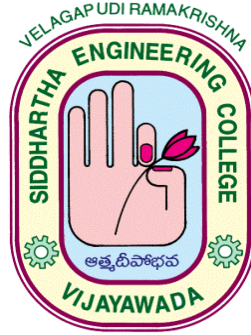
(2023 – 2024)

V.R. SIDDHARTHA ENGINEERING COLLEGE

(Affiliated to JNTUK: Kakinada, Approved by AICTE, Autonomous)

(An ISO certified and NBA accredited institution)

Kanuru, Vijayawada – 520007



CERTIFICATE

This is to certify that this project report titled “ **IMEDSLIFE PATIENT MODULE** ” is a bonafide record of work done by **RIZWANULLAH MOHAMMAD (208W1A1299)**, under my guidance and supervision is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, **V.R. Siddhartha Engineering College** (Autonomous under JNTUK) during the year **2023 - 2024**.

(**Dr . M .Suhasini**)

Asst Professor

Dept. of Information Technology

(**Dr. M. Suneetha**)

Professor & Head

Dept. of Information Technology

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, I sincerely salute our esteemed institution **V.R SIDDHARTHA ENGINEERING COLLEGE** for giving me this opportunity for fulfilling my project. I am grateful to our principal **Dr. A.V.RATNA PRASAD**, for his encouragement and support all through the way of my project.

On the submission of this Project report, I would like to extend my honour to **Dr. M.Suneetha**, Head of the Department, IT for her constant motivation and support during the course of my work.

I feel glad to express my deep sense of gratefulness to my project guide **Dr . M . Suhasini, Asst Professor ,Dept of Information Technology** for his guidance and assistance in completing this project successfully.

I would also like to convey my sincere indebtedness to all faculty members, including supporting staff of the Department, friends and family members who bestowed their great effort and guidance at appropriate times without which it would have been very difficulty on my part to finish the project work.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
LIST OF FIGURES	v
ABSTRACT	vi
CHAPTER-1 Introduction	7
1.1 Origion of the Problem	7
1.2 Basic definitions and Background	8
1.3 Problem Statement	9
1.4 Applications	9
CHAPTER-2 Review of Literature	10
2.1 Literature Survey	10
CHAPTER-3 Proposed Method	11
3.1 System Architecture Diagram (Description of all the modules)	11
3.2 Description of datasets and Tools	11
3.3 Flow Chart	12
3.4 Used technologies	13 - 16
CHAPTER-4 Implementation Steps	17
4.1 Database Schema	17
4.2 Backend Setup	19
4.3 Authentication	19
4.4 Frontend Creation	19

4.5 Frontend, Backend Connection	19
4.6 Application Deployment	19
CHAPTER-5 Results and Observations	20
5.1 Results	20 - 25
CHAPTER – 6 Conclusion and Future Study	26
6.1 Conclusions	26
6.2 Future Study	26
References	27

ABSTRACT

The project undertaken during the internship involved developing a patient module for the IMeds Life website to enhance patient-doctor interactions. The core development included creating a patient authentication system, a dashboard, and a symptom-selection interface. Upon symptom input, an algorithm, utilizing a dataset correlating symptoms with doctor specializations, was employed to match patients with relevant specialists. The algorithm's output, encapsulated as scores, directed patients to the most suited medical practitioners. This algorithm was converted into an API for seamless integration. Real-time consultations were facilitated by hosting Jitsi Meet on AWS, while a slot booking system was implemented for appointment scheduling. Payments were streamlined through Razorpay integration. MongoDB was utilized for backend data storage, ensuring a robust, scalable infrastructure. This comprehensive development significantly elevated the user experience on the IMeds Life website, paving the way for more efficient and user-centric online healthcare services.

Keywords : *IMeds Life Website ,Authentication System ,Symptom-Selection Interface ,API Conversion*

CHAPTER – 1 : INTRODUCTION

1.1 Origin Of Problem :

It is common for patients to face difficulties in finding a suitable doctor for their medical needs, and traditional healthcare systems may involve long waiting times and limited access to medical care. The origin of this project will be to address these issues and provide an efficient and convenient way for patients to access medical care online or offline, while also improving the overall patient experience.

1.2 Basic Defination :

React JS :

React, often referred to as React.js, is an open-source JavaScript library developed by Facebook. It is used for building user interfaces (UI) for web applications. React allows developers to create reusable UI components and efficiently update and render the user interface when data changes. It follows a component-based architecture and is commonly used for building single-page applications (SPAs) and interactive, dynamic web interfaces.

Mongo DB :

MongoDB is a NoSQL database management system known for its flexibility and scalability. It stores data in a document-oriented format, using BSON (Binary JSON) to represent data. MongoDB is designed to handle unstructured or semi-structured data and is suitable for a wide range of applications, including content management systems, e-commerce platforms, and real-time analytics. It's often used in conjunction with Node.js and Express.js for building full-stack applications.

NEXT JS :

Next JS is a powerful Node.js framework for building scalable, modular, and efficient server-side applications. It provides a solid architectural structure for developing back-end applications that are easy to maintain and test. In the healthcare management system project, Next JS can be used to build a robust and secure API that manages and stores sensitive patient data. With Next JS, developers can take advantage of its dependency injection, module system, and powerful error handling capabilities to build a maintainable and scalable application. Additionally, Next JS provides support for a wide range of databases and can be used with popular frontend frameworks like Angular and React. By using Next JS in the healthcare management system project, developers can ensure that the application is secure, scalable, and efficient, and can easily add new features and functionality in the future.

AMAZON S3 BUCKET :

Amazon S3 (Simple Storage Service) is a cloud-based object storage service provided by Amazon Web Services (AWS). It is designed for storing and retrieving any amount of data from anywhere on the internet. S3 provides a highly scalable, reliable, and secure storage solution for businesses of all sizes.

In a healthcare management system project, S3 can be used to securely store and manage large volumes of medical records, images, and other types of patient data. With S3, developers can create buckets, which are essentially containers for storing data, and define policies that control who has access to the data in those buckets. S3 also provides versioning capabilities, enabling developers to track changes to data over time, and lifecycle policies, which automate the process of moving data to different storage classes based on its age or access patterns.

1.3 Problem Statement ;

The project aims to develop a comprehensive healthcare management system through the development of a Web application and a Mobile Application. The system will streamline and improve the healthcare experience for users by providing various features such as appointment scheduling, symptom checker, and access to medical records.

The goal is to enhance the efficiency of healthcare delivery and make it more accessible and convenient for patients.

1.4 Applications :

A healthcare management system can have a wide range of applications across different healthcare settings, including hospitals, clinics, private practices, and nursing homes. Here are some of the potential applications of a healthcare management system:

Patient management: A healthcare management system can be used to manage patient records, including medical history, demographics, test results, and medications. It can also be used to schedule appointments, track patient visits, and manage billing and insurance information.

Electronic health records (EHRs): An EHR system can be integrated with a healthcare management system to provide a comprehensive view of a patient's health history. This can help healthcare providers make more informed decisions about diagnosis, treatment, and medication management.

Telemedicine: With the rise of telemedicine, a healthcare management system can be used to facilitate remote consultations between healthcare providers and patients. This can be especially useful for patients in rural areas or those who have difficulty traveling to appointments.

Inventory management: A healthcare management system can be used to manage inventory of medical supplies and equipment, ensuring that there is always adequate stock on hand and minimizing waste.

Analytics and reporting: A healthcare management system can provide valuable insights into patient care and operational performance. It can be used to generate reports on patient outcomes, quality measures, and financial performance, helping healthcare providers identify areas for improvement.

CHAPTER – 2 : REVIEW OF LITERATURE

2.1 Literature Survey :

The purpose of this literature review is to explore the current state of the art in healthcare management systems and identify the key challenges and opportunities facing the industry

PROCTO	MY FAMILY DOCTOR	AVANTEL PROJECT
Virtual Consultation Available	Virtual Consultation Available	Virtual Consultation Available
Online Lab Tests Booking Available.	No Online Lab Tests	Online Lab Tests Booking Available.
Specialized Practitioners	General Purpose Practitioners	Specialized Practitioners
Multiple Countries	Only In India	Only For Hospital Patients
Online pharmacy	No Online pharmacy	Online pharmacy
Pay Per Service	Subscription Model	Pay Per Service
No Health Packages	Health Packages Available	Health Packages Available

PROCTO APP:

The Procto app [1] is a digital health platform designed to help patients with colorectal diseases. A search of the literature found limited research specifically related to the Procto app. However, there are studies on similar digital health platforms that provide insight into the potential benefits of such apps. For example, a study by Lai et al. (2020) found that a mobile health platform for patients with inflammatory bowel disease (IBD) led to improvements in disease management and patient satisfaction. Another study by Faleiro et al. (2020) found that a digital health platform for patients with chronic gastrointestinal disorders improved patient engagement and symptom monitoring. These findings suggest that the Procto app may also be beneficial for patients with colorectal diseases.

MY FAMILY DOCTOR APP:

The My Family Doctor app[2] is a telemedicine platform that allows patients to connect with doctors remotely. There is a growing body of research on telemedicine, which provides insights into the potential benefits and limitations of such platforms. For example, a systematic review by Flodgren et al. (2015) found that telemedicine can lead to improvements in patient outcomes, such as reduced

hospital admissions and improved clinical parameters. Another study by Whitten et al. (2018) found that telemedicine can improve patient satisfaction and reduce healthcare costs. However, there are also limitations to telemedicine, such as concerns about patient privacy and the potential for misdiagnosis. Overall, the literature suggests that the My Family Doctor app has the potential to improve access to healthcare and patient outcomes, but further research is needed to fully evaluate its effectiveness and safety.

CHAPTER – 3 : PROPOSED METHODOLOGY

3.1 DESIGN METHODOLOGY

The Healthcare Management system is a group of two applications, Web application. The web application will be installed on cloud and mobile application will be installed on smart phones with touch screen. The web application shall be able to process at least 100 requests per second. The web application shall not consume more than 40% of memory. The mobile application shall be supported on Android and iOS devices.

3.1.1 PRODUCT FUNCTIONS

The Healthcare Management System shall have following modules:

- Web application
- Database

Web application:

The web application is the main application and shall provide APIs to be consumed by mobile applications. The web application shall also provide all features of mobile application to be accessed in any browser in a laptop or a computer. The web application shall be developed in MVC architecture wherein there is a clear separation in presentation layer, business layer and data layer. The application shall be scalable to add more features in future based on requirement.

Database:

Database shall be used for storing user data, transactional data, reports and case studies. The data stored in database shall be used for report generation.

3.1.2 User Characteristics

- **Patient:** patient shall have access to schedule of appointments, scheduling lab tests, online consultations, consultation room, prescriptions, lab reports, reminders etc.

3.1.3 Constraints

Adaptability: The application shall be easy to use and adopt by users. The navigation from one screen to another screen shall be self explanatory and requires minimum user inputs required to access any feature.

Scalability: The application shall be scalable to add any new feature in future.

Accuracy: The suggestions based on symptoms shall be made at 100% accuracy as it is a healthcare application. **Reliability:** The application shall be reliable and free of errors.

3.1.4 Assumptions and Dependencies

- List of common symptoms available in the system. Admin users shall be able to add additional symptoms in the system.
- Symptom based specialization shall be predefined in the system. This list is expandable and the system shall allow adding or updating the mapping of symptom to specialization.

3.2 SYSTEM ARCHITECTURE DIAGRAM

The Architecture diagram of our work is displayed in figure below:

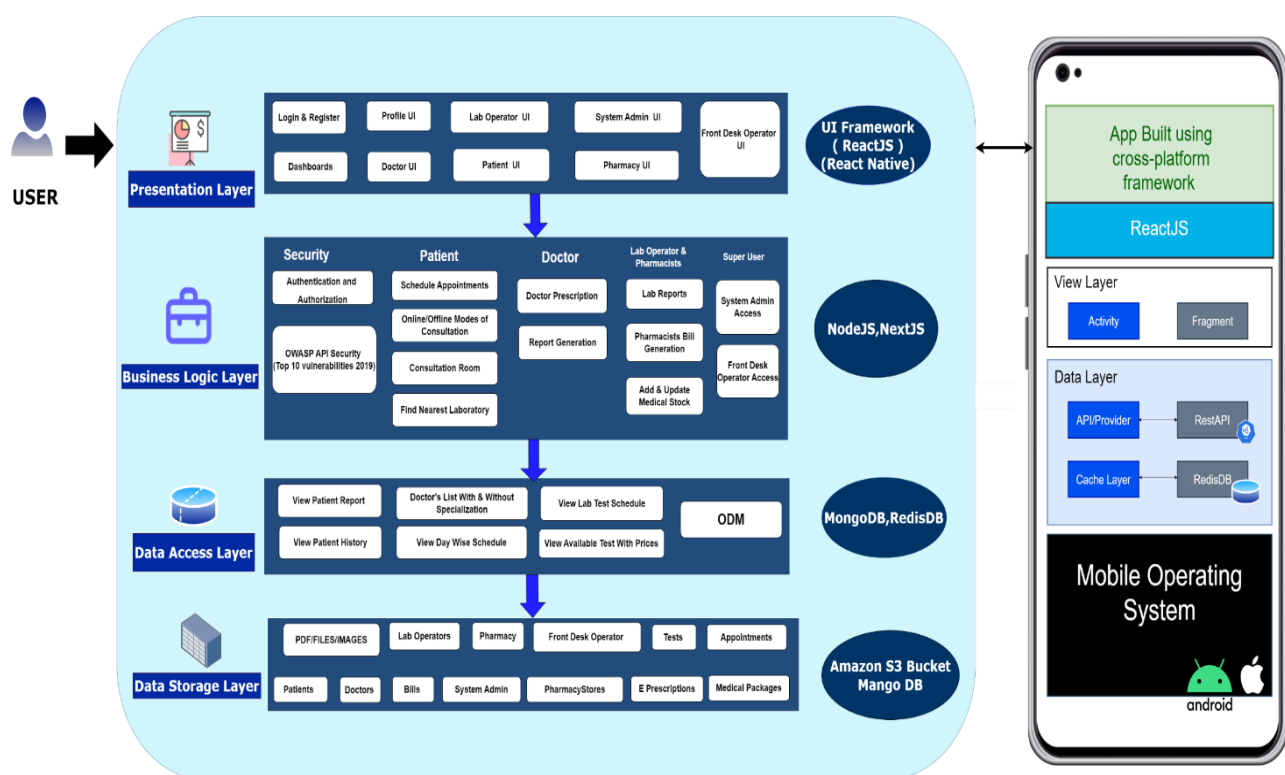


Figure 1. Architecture Diagram

3.2.1 Description - Layered Architecture Presentation Layer:

This layer deals with the user interface and interaction with the users. It includes components like User Registration, User Dashboards, User Profiles, Appointment / Schedule logic and other UI components that are need to be visible to the user.

Business Logic Layer:

This layer contains the business logic of the proposed application. It includes components like controllers, services, and models to handle security, user operations and healthcare services.

Controllers :

Controllers in HCMS Proposed Layered Architecture are of two types :

- Authentication Controller
- Profile Controller

Authentication Controller is to manage every user authentication and authorization. Its operations include user login request, user logout, registering new users, resetting passwords, and controlling user access.

Profile controller is to maintain user information. Supporting methods need to be for retrieving user information, updating user information, and deleting user accounts.

Services

Services in HCMS Proposed Layered Architecture are of two types :

- Authentication Service
- Profile Service

Authentication Service:

This service is to handle user authentication and authorization at all times of user entry/exit of the developing system. Various necessary methods include authenticate user credentials, generating and validating access tokens and checking user permissions.

Profile Service:

This service deals with user profile information. Methods relevant here include retrieving user information, updating user information and deleting user accounts.

Models

User Model: This model represents all types of users in the system. Its characteristics include email, password, username, and profile picture.

Post Model: This model represents a post in the system. Its properties comprise title, content, author, and date. Data Access Layer : The data access layer is responsible for managing the interaction between the application and the underlying database. In this layer, it is included several functionalities.

The data storage layer uses AWS S3 BUCKET for storing PDF/files/images and Mongo DB technology for storing the rest of the data. AWS S3 BUCKET is a cloud-based storage service that provides secure, scalable, and durable storage for objects like files and images. Mongo DB is a NoSQL database technology that provides flexible and scalable data storage capabilities for complex data structures like JSON documents.

3.3 DATA BASE DESIGN :

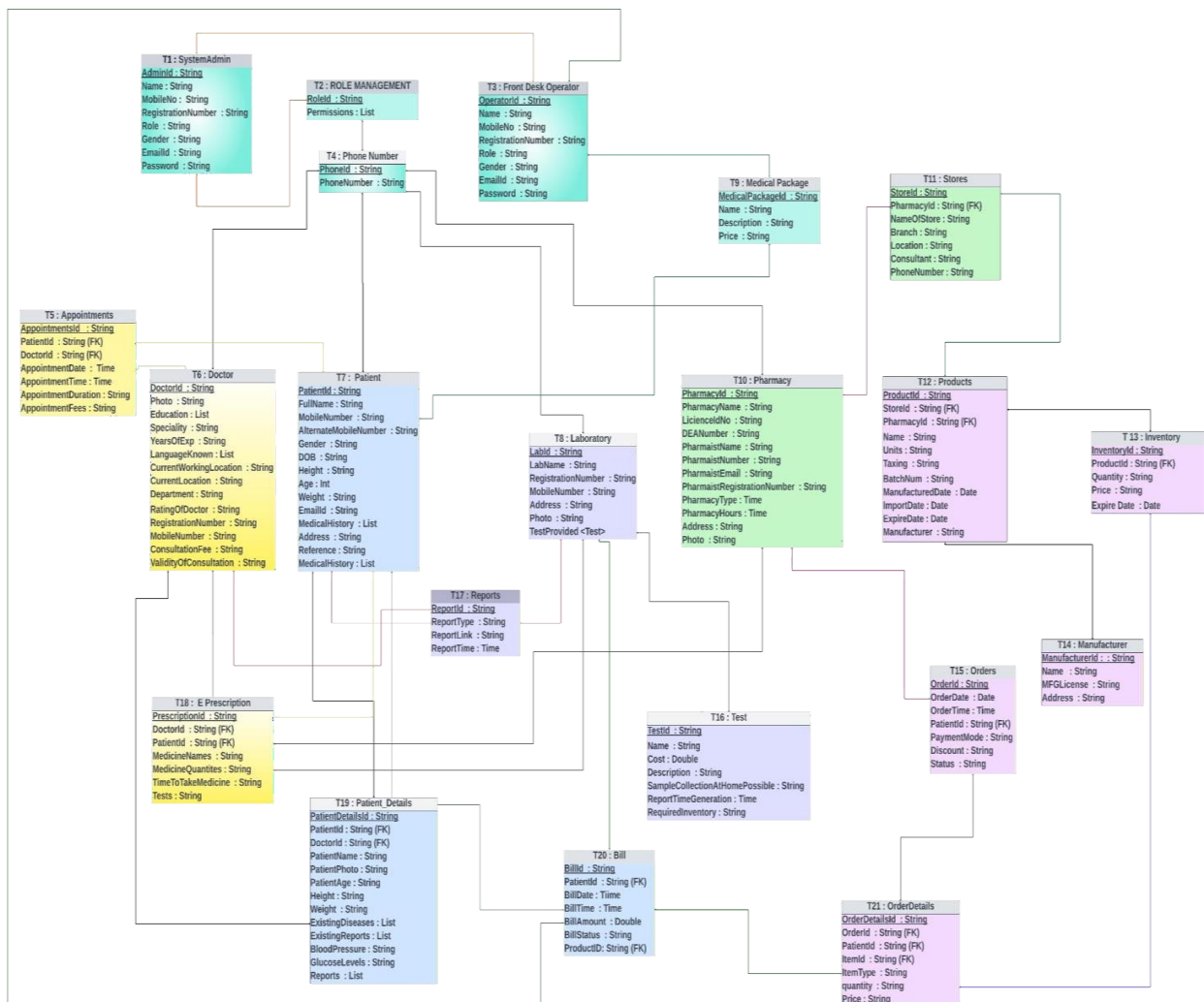


Figure 2 DATA BASE DESIGN

3.3.1 DATABASE DESIGN DESCRIPTION :

- **T1 : SystemAdmin** : AdminId (PK) ,Name , MobileNo , RegistrationNumber , Role , Gender , EmailId, Password : This is the super user in the proposed system. Admin will have access to all the Modules
- **T2 : ROLE MANAGEMENT** : RoleId (PK), Permissions : This table is used for permissions

- T3 : Front Desk Operator : OperatorId (PK), Name , MobileNo , RegistrationNumber , Role , Gender, EmailId , Password : Bills and payments can also be accessed by the Front desk operator. The front desk operator will have access to all and medical packages and discounts will be managed by front desk operator .
- T4 : Phone Number : PhoneId (PK), PhoneNumber : It will store all the phone numbers.and with accordingly give access to the particular user
- T5 : Appointments : AppointmentsId (PK), PatientId (FK), DoctorId (FK) ,AppointmentDate, AppointmentTime , AppointmentDuration , AppointmentFees : The appointments booking will be stored in this table with reference to the patientsid and doctorsID .
- T6 : Doctor : DoctorId (PK), Photo , Education ,Speciality , YearsOfExp ,LanguageKnown, CurrentWorkingLocation ,CurrentLocation , Department ,RatingOfDoctor RegistrationNumber , MobileNumber ,ConsultationFee ,ValidityOfConsultation : The individual doctor details will be stored here and Doctorid is the Foreign key. It has a relationship with E Prescription table.
- T7 : Patient : PatientId , FullName , MobileNumber , AlternateMobileNumber , Gender , DOB , Height , Age : Int Weight , EmailId , MedicalHistory , Address , Reference , MedicalHistory ,
- T8 : Laboratory : LabId , LabName , RegistrationNumber , MobileNumber , Address , Photo , TestProvided <TEST> : Apart from laboratory details, this table is connected to Test table
- T9 : Medical Package : MedicalPackageId , Name , Description , Price : The medical packages which are dialy updated by the front desk operator will be stored in this and will be accessed by the patients and they can use the packages .
- T10 : Pharmacy : PharmacyId , PharmacyName , LicienceIdNo , DEANumber , PharmaistName , PharmaistNumber , PharmaistEmail , PharmaistRegistrationNumber , PharmacyType PharmacyHours ,Address , Photo : Maintains pharmacy operations. It has connections to the Store table.
- T11 : Stores : StoreId , PharmacyId , (FK) NameOfStore , Branch , Location , Consultant , PhoneNumber : The stores contains all the stores which are registred under the pharmacy table. The store table is connected to the products table
- T12 : Products : ProductId , StoreId , (FK) PharmacyId , (FK) Name , Units , Taxing , BatchNum , ManufacturedDate , ImportDate , ExpireDate , Manufacturer : Contains all the Products like medicines etc., available in the stores. This table has relationship with inventory table.
- T13 : Inventory : InventoryId , ProductId , (FK) Quantity , Price , Expire Date : Comprises all medicines details. Like how many tables from a company and its details . This has relationship with ordered details table.
- T14 : Manufacturer : ManufacturerId : , Name , MFGLicense , Address : The manufacturer details of the medicines will be stored In this .
- T15 : Orders : OrderId , OrderDate , OrderTime ,PatientId , (FK) PaymentMode , Discount , Status , : Ther orders which are received from the patient will be stored in this .

- T16 : Test : TestId , Patient ID , (FK) Name , Cost : Double Description , SampleCollectionAtHomePossible , ReportTimeGeneration ,Required Inventory : The test table will contain all the tests provided by the laboratory.
- T17 : Reports : ReportId ,ReportType ,ReportLink,ReportTime : will be accessed by Doctors, lab operator, patient
- T18 : E Prescription : PrescriptionId (PK), DoctorId (FK) ,PatientId (FK), MedicineNames MedicineQuantites , TimeToTakeMedicine , Tests : Contains the prescription given by the doctors which will be accessible by the Patient, Pharmacy and Lab operator.
- T19 : Patient_Details : PatientDetailsId ,PatientId (FK), DoctorId (FK), PatientName , PatientPhoto , PatientAge , Height , Weight , ExistingDiseases, ExistingReports , BloodPressure , GlucoseLevels ,Reports : It has relationship with Doctor Table. By this the doctor can view the health conditions of the patients before consultation.
- T20 : Bill : BillId (PK), PatientId (FK), BillDate ,BillTime , BillAmount , BillStatus , ProductID, (FK) : Contains all the billing information. The bills belong to patient, pharmacy, laboratory and consultation Fees. It is connected to front desk operator table.
- T21 : OrderDetails : OrderDetailsId (PK), OrderId (FK) ,PatientId (FK), ItemId (FK) ,ItemType , quantity,Price : This contains all the orders received from the patient.

CHAPTER – 4 : IMPLEMENTATION STEPS

4.1 Database Schema :

Determine the data structure for storing information about voters(users) and for votes(polls). Use MongoDB as your database to store this information.

```
pollifyback > models > JS pollschema.js > ...
1  const mongoose = require('mongoose');
2
3  const pollSchema = new mongoose.Schema({
4    qno: { type: Number, required: true },
5    question: { type: String, required: true },
6    option1: { type: String, required: true },
7    option2: { type: String, required: true },
8    option3: { type: String, required: true },
9    option4: { type: String, required: true },
10  }, {
11    timestamps: true,
12  });
13
14  const Poll = mongoose.model("Poll", pollSchema);
15
16  module.exports = { Poll };
```

Fig – 1 : Database Patient

```
pollifyback > models > JS schema.js > ...
1  const mongoose = require('mongoose');
2
3  const userSchema = new mongoose.Schema({
4    firstname: { type: String, required: true },
5    lastname: { type: String, required: true },
6    username: { type: String, required: true },
7    password: { type: String, required: true },
8    email: { type: String, required: true },
9    mobile: { type: String, required: true },
10   city: { type: String, required: true },
11   pincode: { type: String, required: true },
12 }, {
13   timestamps: true,
14 });
15
16 const User = mongoose.model("User", userSchema);
17
18 module.exports = { User };
```

Fig – 2 : Database User Schema

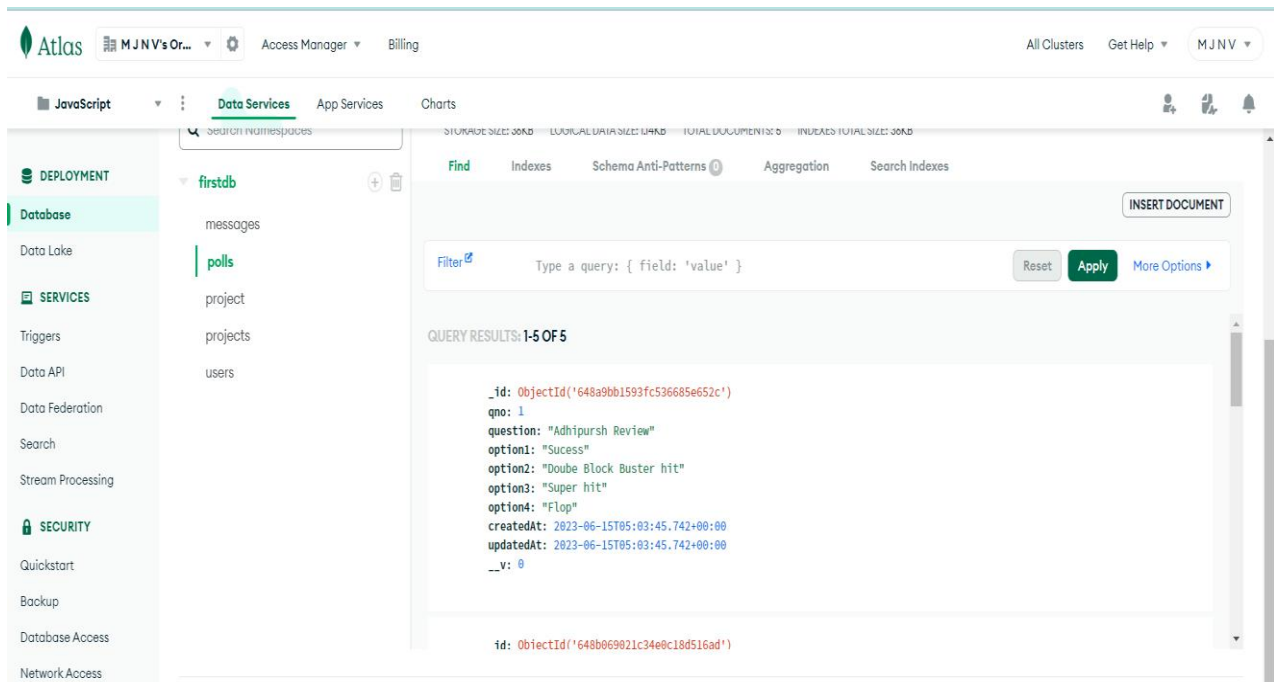


Fig – 3 : Data in Vote Poll Schema

4.2 Backend Setup :

Create a backend server using Next js. This server will handle API requests and interact with the database. Set up routes for authentication, voter registration, candidate creation, election management, and vote submission.

Here in this project, we created a backend folder named “ **Imedislife** ”.

In pollifyback, Our main file for the backend is **server.js**, in which will be establishing the backend server, import all the routes, and also we will establish the Mongodb (database) connection here itself.

```

JS server.js x
pollifyback > JS server.js > ...
1  const express = require('express');
2  const cors = require('cors');
3  const mongoose = require('mongoose');
4  const auth = require('./routes/auth');
5  const user = require('./routes/user');
6  const { Poll } = require('./models/pollschema');
7
8  //initilze express.js
9  const app = express();
10
11 //to receive json data
12 app.use(express.json());
13
14 //initilze cors
15 app.use(cors({
16   origin: '*'
17 }));
18
19 //connect mongobd
20 mongoose.connect('mongodb+srv://mjnvsai:mjnvsai@cluster0.kc6nhff.mongodb.net/firstdb?retryWrites=true&w=majority').then(
21   console.log("Mongo Db Atlas is connected Sucessfully")
22 );
23
24
25 //auth api's
26 app.use('/api/auth', auth);
27
28 //users api's
29 app.use('/api/user', user);
30
31 app.get("/retrive", (req, res) => {
32   Poll.find().then(found => res.json(found) )
33 } )
34
35 //run server
36 app.listen(5000, () => console.log('server is running on port number 5000'));

```

Fig – 5 : Main Server File

In imedsife, we will be writing all the routes files i.e.; jwt token, authentication, get and delete routes.

4.3 Authentication :

Add user authentication to ensure that only registered voters can participate in the voting process. For authentication and session management, you can use popular authentication libraries such as Passport.js or JSON Web Tokens (JWT). This authentication code has been written in the routes → auth.js file in the backend and also we have used an authentication library like JWT. JSON Web Tokens (JWT) are a widely used authentication and authorization mechanism on the server side. They are compact, self-contained tokens that carry user information and are digitally signed for security. Servers can validate and extract information from JWTs, enabling secure access control and user authentication. JWTs are often used in stateless applications and APIs for maintaining user sessions. On the server side, JWTs are typically generated when a user logs in or accesses a protected resource. They contain user data and a signature, ensuring their authenticity. Servers can quickly verify and decode JWTs to make access control decisions without needing to store session information, making them a scalable and efficient solution.

```

pollifyback > routes > JS auth.js > ...
1 | //authentication code
2 | const router = require('express').Router();
3 | const bcrypt = require('bcrypt');
4 | const { User } = require('../models/schema');
5 | const { Poll } = require('../models/polls/schema');
6 | const jwt = require('jsonwebtoken');
7 |
8 | //register route
9 | router.post("/register", async (req, res) => {
10 |   let { firstname, lastname, username, password, email, mobile, city, pincode } = req.body;
11 |
12 |   //check the user already exist with this email
13 |   const takenEmail = await User.findOne({ username: username });
14 |
15 |   if (takenEmail)
16 |   {
17 |     return res.status(405).json("voter already exists");
18 |   }
19 |   else

```

Fig – 6 : Register Route

```

38 | //login user
39 | router.post("/login", async (req, res) => {
40 |   try
41 |   {
42 |     const { username, password } = req.body;
43 |
44 |     //confirm the user is register or not
45 |     const userexist = await User.findOne({ username: username });
46 |
47 |     if (!userexist)
48 |     {
49 |       return res.status(404).json('user not found');
50 |     }
51 |
52 |     bcrypt.compare(password, userexist.password).then( (isCorrect) => {
53 |       if (isCorrect)
54 |       {
55 |         let payload = {
56 |           user: {
57 |             id: userexist.id
58 |           }
59 |         }
60 |
61 |         jwt.sign(payload, 'newsecrete', { expiresIn: 360000000 }, (err, token) => {
62 |           if (err) throw err;
63 |           return res.status(200).json({ token: token, name: userexist.name });
64 |         });

```

Fig – 7 : Login Route

We have included the jwt token in routes → middleware.js file.

In server-side development, a middleware.js file is commonly used to manage middleware functions in an application. These functions intercept and process incoming requests before they reach the final route handler. Middleware can be used for tasks like authentication, logging, request parsing, and more. By chaining middleware functions, developers can create a modular and flexible approach to handling HTTP requests in their server-side applications.

```

pollifyback > routes > JS middleware.js > ...
1  const jwt = require('jsonwebtoken');
2
3  module.exports = function (req, res, next)
4  {
5      try
6      {
7          let token = req.header('x-token');
8          if (!token)
9          {
10             return res.status(400).send('Token Not found');
11          }
12
13          let decode = jwt.verify(token, 'newsecreate');
14          req.user = decode.user
15          next();
16      }
17      catch (err)
18      {
19          console.log(err);
20          return res.status(500).send('Invalid token')
21      }
22  }

```

Fig – 8 : Middleware

Use React Router to manage different routes and navigation within the application.

```

pollifyback > routes > JS user.js > ...
1  const router = require('express').Router();
2  const { User } = require('../models/schema');
3
4  router.get('/', async(req, res) => {
5      return res.status(200).json(await User.find());
6  });
7
8  router.delete('/:id', async(req, res) => {
9      await User.findByIdAndDelete(req.params.id);
10     return res.status(200).json("deleted successful")
11 });
12
13 module.exports = router;

```

Fig – 9 : User Authentication

4.4 Frontend Creation :

Create the front end with React.js. Create a user interface that allows voters to register, browse candidates, vote in elections, and submit their ballots. In this project, we titled the frontend folder "pollifyfront".

In pollifyfront, our primary frontend file is App.js, in which we will import all the components such as Home, Register, Login, CreatePolls, Logout, Visitpolls, and so on, and we will navigate into those components using BrowserRouter, Route, Routes from "react-router-dom."

```

pollifyfront > src > JS App.js > ...
1  import React, { Component } from 'react'
2  import { BrowserRouter, Route, Routes } from "react-router-dom";
3  import Navigation from './components/Navigation'
4  import Home from './components/Home';
5  import Register from './components/Register';
6  import Login from './components/Login';
7  import Logout from './components/Logout';
8  import CreatePoll from './components/CreatePoll';
9  import Visitpolls from './components/visitpolls';
10 // import SurveyPoll from './components/SurveyPoll';
11
12 export default class App extends React.Component {
13   render() {
14     return (
15       <div>
16         <BrowserRouter>
17           <Navigation />
18
19           <Routes>
20             <Route path="/" element = {<Home />} />
21             <Route path="/visitpolls" element = {<Visitpolls />} />
22             <Route path="/createpoll" element = {<CreatePoll />} />
23             <Route path="/register" element = {<Register />} />
24             <Route path="/login" element = {<Login />} />
25             <Route path="/logout" element = {<Logout />} />
26           </Routes>
27         </BrowserRouter>
28       </div>
29     )
30   }
31 }
32

```

Fig – 10 : All Routes related to the application

We will be writing all those component files in a separate folder called components.

```

pollifyfront > src > components > JS CreatePoll.js > CreatePoll
1  import React, {useEffect, useState} from 'react'
2  import { useNavigate } from 'react-router-dom'
3  import swal from 'sweetalert'
4  import './App.css'
5  import axios from 'axios'
6  export default function CreatePoll()
7  {
8    const navigate = useNavigate()
9    useEffect(() => {
10      const localStorageItem = localStorage.getItem('token');
11      if(!localStorageItem)
12      {
13        swal({
14          title: "ERROR",
15          text: "User Login Required",
16          icon: "error",
17        });
18        navigate('/login')
19      }
20    })
21    }, [navigate] )

```

Fig – 11 : patient loginCreation

```

pollifyfront > src > components > JS Navigation.js > Navigation > render
1  import React, { Component } from 'react'
2  import '../App.css'
3  import { Link } from 'react-router-dom'
4
5  export default class Navigation extends Component
6  {
7      state = {
8          logo : "https://img.freepik.com/free-vector/hand-with-voting-sign-election_1017-184"
9          title : "Pollify Voting"
10     }
11     render()
12     {
13         return (
14             <nav>
15                 <ul className = "navbar-list"><li style={{marginRight : '50%' }} >
16                     <img style={{paddingTop : '12px',borderRadius : '50%',width : '50px'}}src="" alt="Voting Sign Election" />
17                     <li><Link className = "nav-link" to = "/" > Home </Link></li>
18                     <li><Link className = "nav-link" to = "/visitpolls" > Visit Polls </Link></li>
19                     <li><Link className = "nav-link" to = "/createpoll" > Create Poll </Link></li>

```

Fig – 12 : Application Navigation

```

pollifyfront > src > components > JS Login.js > Login
1  import React, { useState } from 'react'
2  import '../App.css'
3  import swal from 'sweetalert'
4  import { useNavigate } from 'react-router-dom';
5  import axios from 'axios';
6
7  export default function Login()
8  {
9      const navigate = useNavigate();
10
11      const [login, setlogin] = useState({ username: '', password: '' });
12      const changeHandler = (e) => {
13          setlogin( { ...login, [e.target.name]: e.target.value } );
14      }
15
16      const submitHandler = (e) => {
17          e.preventDefault();
18          axios.post('http://localhost:5000/api/auth/login', login)
19              .then( res => {
20                  localStorage.setItem('token', res.data.token);
21                  navigate('/')
22              } )
23              .catch( err => {
24                  // alert(err.response.data)
25                  swal({
26                      title: 'ERROR',
27                      text: `${err.response.data}`,
28                      icon: 'error',
29                      button: 'OK'
30                  });
31              } );
32      }

```

Fig – 13 : Login Component

The login component will interact with the express and node servers to determine whether or not data exists in the MongoDB Cloud Atlas platform. If the user is not registered with the program and no records are identified for that user, a pop-up message will appear stating that you have not registered with us. as a result, please register with the application database.


```

pollifyfront > src > components > JS Register.js > Register > changeHandler
1  import React, { useState } from 'react'
2  import '../App.css'
3  import swal from 'sweetalert'
4  import { useNavigate } from 'react-router-dom';
5  import axios from 'axios';
6
7  export default function Register()
8  {
9    const navigate = useNavigate();
10   const [register, setregister] = useState({ firstname: '', lastname: '', username: '', password:
11   const changeHandler = (e) => {
12     setregister({ ...register, [e.target.name]: e.target.value });
13   }
14   function validation()
15   {
16     const namecheck = /^[a-zA-Z ]+$/
17     const phonecheck = /^[0-9]{10}$/
18     const mailcheck = /^[a-z0-9.#]+@[a-z]+\.[a-z]{2,3}$/
19     const passcheck = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}$/
20     const pincheck = /^[1-9][0-9]{5}$/
21
22     if(register.firstname === "" || register.firstname.length < 3)
23     {
24       swal({
25         title: "ERROR",
26         text: "Please Enter Your First Name Correctly",
27         icon: "error",
28       });
29       return false
30     }

```

Fig – 14 : Registration Component

4.5 Connection Between Frontend and Backend :

Establish communication between the frontend and backend by making API requests. Use libraries like Axios or the built-in fetch API to send HTTP requests from the frontend to the backend server.

```

    axios.post(`http://localhost:5000/api/auth/createpoll`, polldata)
    .then( res => {
      // alert(res.data);
      swal({
        title: 'SUCCESS',
        text: `${res.data}`,
        icon: 'success',
        button: 'OK'
      });
      navigate('/visitpolls')
    } )
    .catch( err => {
      // alert(err.response.data)
      swal({
        title: 'ERROR',
        text: `${err.response.data}`,
        icon: 'error',
        button: 'OK'
      });
    } );
  }
}

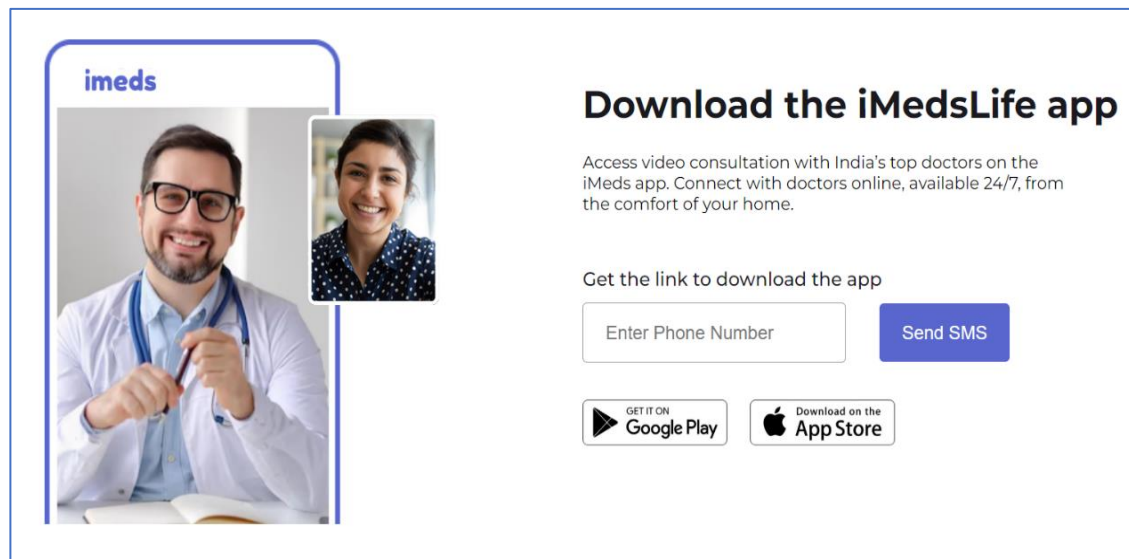
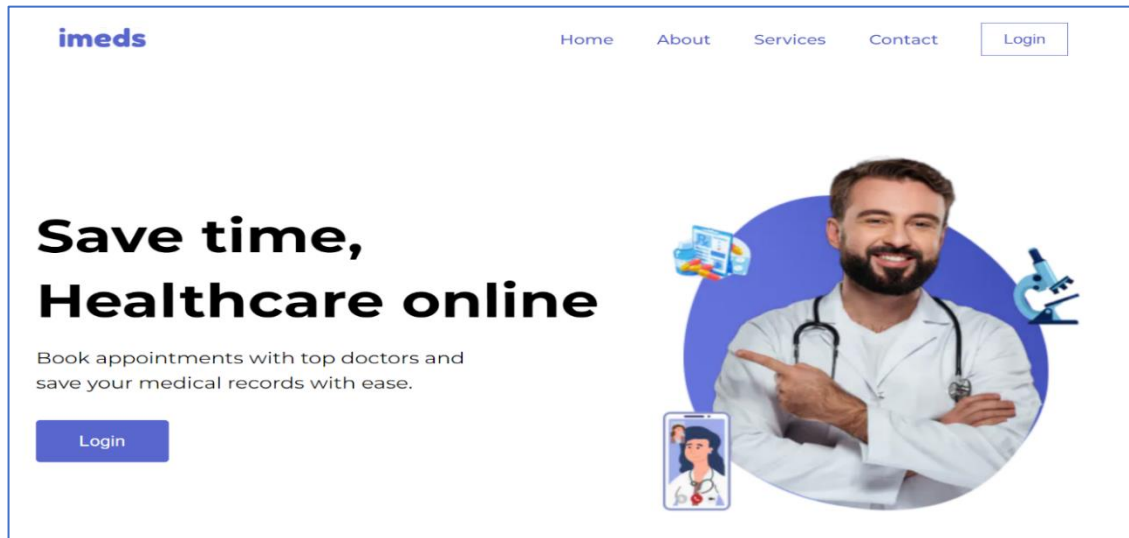
```

Fig – 15 : Connecting with Backend Server

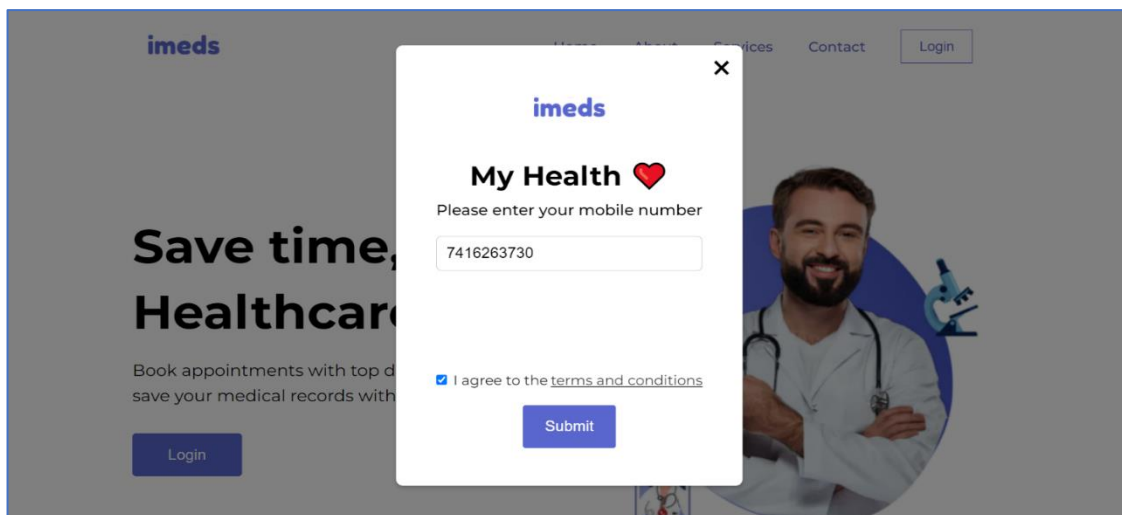
CHAPTER – 5 : RESULTS AND OBSERVATIONS

5.1 Results :

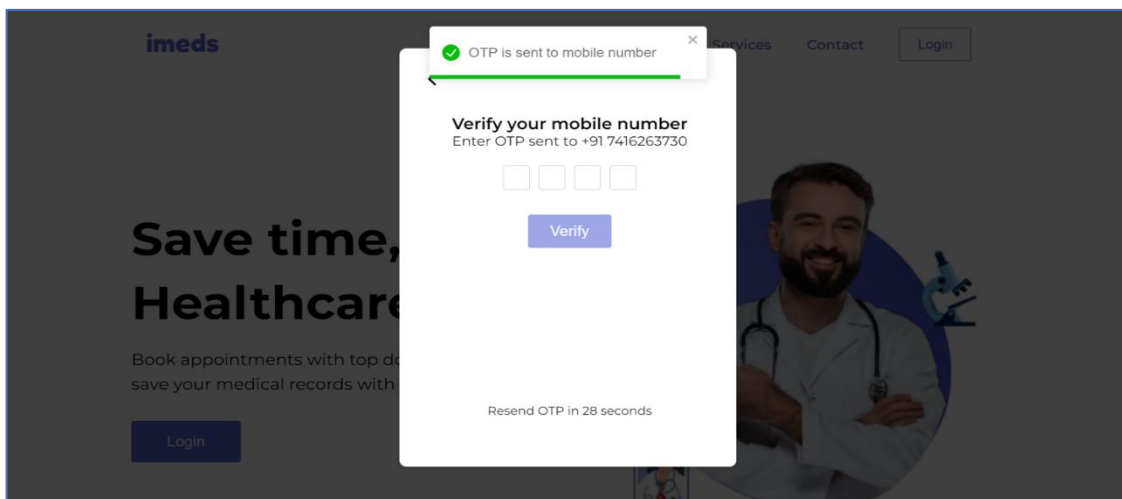
STEP 1 : Created a doctor landing page with sections for Home, About, Service, Contact, and login authentication.



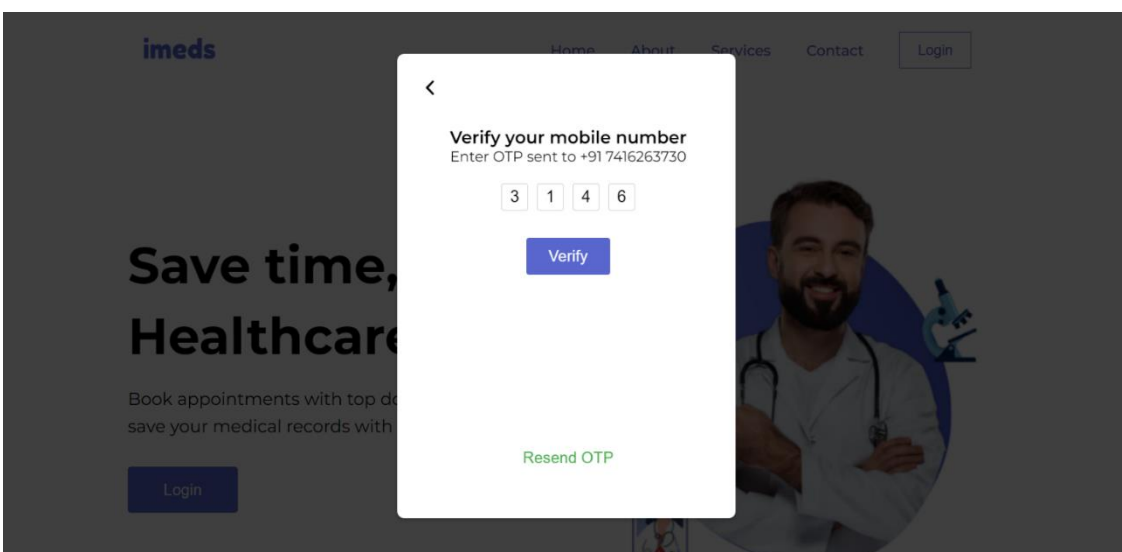
STEP 2 : For authentication, users enter their phone number, receive an OTP on their mobile, and then confirm through OTP verification



The screenshot shows the 'My Health' screen of the imeds app. A modal dialog is open with the title 'My Health' and a red heart icon. Below the title, it says 'Please enter your mobile number'. There is a text input field containing the number '7416263730'. Below the input field, there is a checkbox labeled 'I agree to the terms and conditions' which is checked. At the bottom of the modal is a blue 'Submit' button. The background of the app shows a doctor's profile and a 'Login' button.




The screenshot shows the 'Verify your mobile number' screen of the imeds app. A modal dialog is open with the title 'Verify your mobile number' and the text 'Enter OTP sent to +91 7416263730'. Above the input field, there is a green progress bar and a green checkmark icon with the text 'OTP is sent to mobile number'. The input field consists of four empty boxes. Below the input field is a blue 'Verify' button. At the bottom of the modal, it says 'Resend OTP in 28 seconds'. The background of the app shows a doctor's profile and a 'Login' button.



The screenshot shows the 'Verify your mobile number' screen of the imeds app. A modal dialog is open with the title 'Verify your mobile number' and the text 'Enter OTP sent to +91 7416263730'. The input field consists of four boxes containing the digits '3', '1', '4', and '6'. Below the input field is a blue 'Verify' button. At the bottom of the modal, there is a green 'Resend OTP' link. The background of the app shows a doctor's profile and a 'Login' button.

STEP 3 : After confirming the OTP, the user is logged into their dashboard ,Here, they can book an online consultation by clicking the 'Find Doctor' button and also view their consultation schedule. OR if the user is new he will be navigated into create account page



Create account

Name


DOB

Gender

cm kg

STEP 4 : Upon clicking 'Find Doctor', the user is directed to a page where they can select who the consultation is for (e.g., myself or family members). They can also create a new profile for a family member by clicking the 'Add Profile' button

imeds

Home My Lab Tests My Consultations Logout Rizwanullah 

Rizwanullah

Find Doctor

Schedule lab test

Health Tracker


My Consultations

Fazulunnisa's appointment

Dr. Rizwanullah Mohammad 10:30 13/10/2023


Join Video Call

imeds


Home My Lab Tests My Consultations Logout Rizwanullah 

Rizwanullah


User Profile



MySelf



father



mother

STEP 5 : After selecting a patient profile, the user is directed to the symptoms page. Here, they must choose at least three symptoms. Based on their selections, an appropriate doctor will be automatically recommended.

imeds Home My Lab Tests My Consultations Logout Rizwanullah

Rizwanullah

Find a Doctor Schedule a Test Health Tracker

What is your concern?

Search Symptoms or Specialist Choose Doctor

Most Selected Issues

Fever	Gas	Loose Motion/Diarrhea	Blocked Nose
Sneezing	Acne	Rashes	Period Related Issues
Spots On Skin	Dark Circles	Vomiting	Headache
Constipation	Runny Nose	Abdominal Pain	Hairfall
Obesity	Dry Skin	Heartburn	Throat Pain
Itching			

STEP 6: After selecting the symptoms, the patient will see a list of doctors specialized in treating those symptoms.

imeds Home My Lab Tests My Consultations Logout Rizwanullah

Rizwanullah

Video consult Availability All Filters Sort By Relevance

Doctors Found

Showing earliest available doctors

Pune
Rizwanullah Mohammad
Surgery, Dentistry, Surgical Super-Specialty, Dental Specialized Field, Research, Psychiatry
Hindi, English, Bengali, Telugu, Marathi
13 years Exp ₹ 500 Know More Book Appointment

Pune
Dharmapuri Mahith Paul
Surgery, Specialized Medicine, Dentistry
Hindi, English, Telugu, Marathi
5 years Exp ₹ 500 Know More Book Appointment

STEP 7: After choosing a doctor, the patient needs to schedule a time for their consultation. The available consultation modes are video call, audio call, and text-based.

imeds Home My Lab Tests My Consultations Logout Rizwanullah

Rizwanullah

Book Appointment

Consultation type

Video call Audio call Text based

Today Tomorrow In 2 days

October 2023

Time Slots

10:00 10:10 10:40 10:50

11:00 11:10 11:20 11:30

11:40 12:00 12:10 12:20

12:30 12:40 12:50 13:00

13:10 13:20 13:30 13:40

13:50 14:00 14:10 14:20

14:30 14:40 14:50 15:00

15:10 15:20 15:30 15:40

15:50 16:00 16:10 16:20

16:30 16:40 16:50 17:00

Continue

STEP 8 : After choosing a time slot, the patient will be directed to the payment section to make a payment for the consultation with the doctor

imeds Home My Lab Tests My Consultations Logout Rizwanullah

Rizwanullah

Consultation Details

Type Of Consultation : Video

Date Of Consultation : 13-10-2023

Time Of Consultation : 11:10

Appointment For : Rahamatullah

Relation : Father

Cancel Pay Now

imeds Home My Lab Tests My Consultations Logout Rizwanullah

Rizwanullah

Consultation confirmed

Your consultation is confirmed.
A confirmation message has been sent

Appointment Details

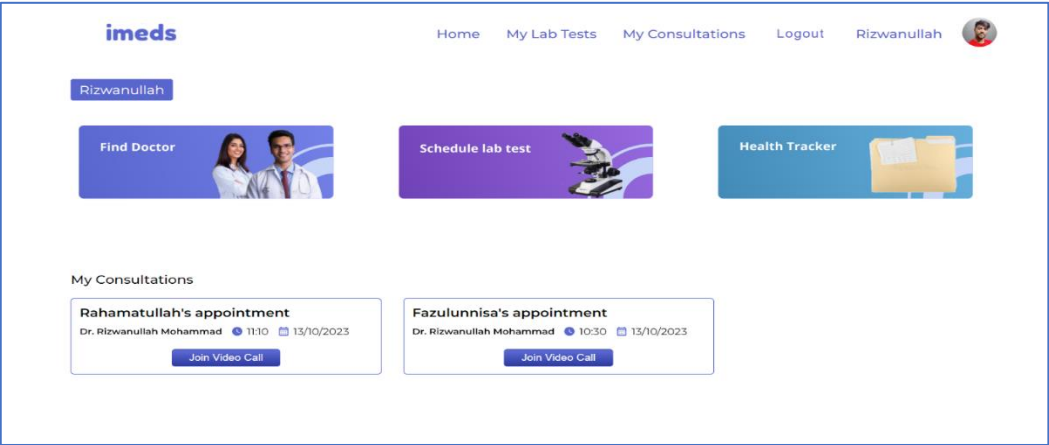
Dr. Rizwanullah Mohammad
Doctor Speciality - Surgery,
Dentistry, Surgical Super-
Specialty, Dental Specialized Field,
Research, Psychiatry
Consultation Type - video

13-10-2023 11:10

☐ Send me a reminder 10min before the call

Done

STEP 9 : The scheduled time slot will be displayed on the patient's main dashboard



CHAPTER – 6 : CONCLUSION AND FUTURE STUDY

6.1 Conclusion :

The development of an online consultation website is a valuable and effective solution for providing convenient and accessible healthcare services. This website offers numerous benefits for both healthcare professionals and patients as well and the online consultation website prioritizes privacy and security. By implementing robust data encryption and secure server systems, it ensures the confidentiality of patient information and medical records. Patients can confidently share sensitive details with healthcare providers, fostering trust and confidentiality.

Lastly, the online consultation platform promotes efficiency and productivity. Healthcare professionals can manage their schedules more effectively, reducing waiting times and optimizing their workflow. Patients experience reduced waiting times and enjoy the convenience of accessing healthcare services from the comfort of their homes.

6.2 Future Study :

Developing Mobile Application for the Health Care Management System for Doctor, Laboratory , pharmacy , admin

Apply Authentication & Security in Web application and Mobile application. Identify and handle OWASP API Security the Top 10 Vulnerabilities 2019 in APIs exposed by the web application. Implement TLS v1.2 in APIs for securing data transfer between web server and mobile/web application. Implementation of logging and registration for all users in the proposed system. Design registrations, schedules, calendars, e-prescription through dashboard

REFERENCES

<https://nextjs.org/>

<https://react.dev/>

<https://www.mongodb.com/>

<https://www.practo.com/>

<https://www.clickittech.com/devops/web-application-architecture/>

<https://soft-builder.com/bank-management-system-database-model/>