A

Project Report

On

HOSPITAL MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirement for the IV semester

Bachelor of Technology

By

Pratish Barthwal 2061891

Manas Chauhan 2061972

Aman Bisht 2061763

Under the Guidance of

Mr. Ravindra Koranga

Assistant Professor

Deptt. of CS&A

DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2021 – 2022

STUDENTS' DECLARATION

We, Pratish Barthwal, Manas Chauhan and Aman Bisht hereby declare the work, which is being presented in the project, entitled "HOSPITAL MANAGEMENT SYSTEM" in partial fulfillment of the requirement for the award of the degree Bachelor of Technology in the session **2021-2022**, is an authentic record of our own work carried out under the supervision of Mr. Ravindra Koranga, Assistant Professor Dept. of CSE, Graphic Era Hill University, Bhimtal.

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

Date: …… …….

(Full signature of student)

CERTIFICATE

The project report entitled "HOSPITAL MANAGEMENT SYSTEM" being submitted by Pratish Barthwal (2061891), Manas Chauhan (2061972) and Aman Bisht (2061763), **to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirements for the submission of report.**

(……) **(……)**

Project Guide **(HOD, CSE Dept.)**

ACKNOWLEDGEMENT

We take immense pleasure in thanking Honorable Mr. Ravindra Koranga (Assistant Professor**,** CS&A, GEHU Bhimtal Campus) to permit us and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that helped us to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering our thanks to GOD for providing us everything that we need. We again want to extend thanks to our President **"Prof. (Dr.) Kamal Ghanshala"** for providing us all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to Professor **"Dr. Manoj Chandra Lohani"** (Director, GEHU), **"Mr. Ankur Bisht" (HOD**, CS&A, GEHU) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

TABLE OF CONTENTS

1 - INTRODUCTION

1.1- Prologue

The project Hospital Management system includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically.

The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

The purpose of the project entitled as "HOSPITAL MANAGEMENT SYSTEM" is to computerize the Front Office Management of Hospital to develop software which is user friendly, simple, fast, and cost – effective. It deals with the collection of patient's information, diagnosis details, etc.

Traditionally, it was done manually. The main function of the system is to register and store patient details and doctor details and retrieve these details as and when required, and also to manipulate these details meaningfully System input contains patient details, diagnosis details; while system output is to get these details on to the CRT screen.

1.2- BACKGROUND AND MOTIVATION

We do not usually visit hospitals unless we need to but this visit does not always give us good experience. We face various problems there. There are automated hospital management systems but they do not provide any functionality for us.

Even the doctors do not have access to their patient's data while they are at home. It will be great for the patients to have an application that will keep them close to their personal data like: prescriptions, test reports and other important materials. Patients will be able to browse through all the doctors available and ask for appointment for their desired doctor. A doctor will also experience the similar benefits. This will reduce a lot of paper works and make things easy for everyone associated with it. I have some personal experience visiting the hospital and I alsohave gathered experience from other people by asking them and observing them for some time. People experience their worst nightmare while they need to visit the hospital. No body visits the hospital unless it is extremely necessary. People get mad but they do not have anything to do then. This gave me the perfect motivation to build something for them and offer them some help at their most needed time. I am confident that this product can and will help them.

1.3- PROBLEM STATEMENT

In this busy world we don't have the time to wait in infamously long hospital queues. The problem is, queuing at hospital is often managed manually by administrative staff, then take a token there and then wait for our turn then ask for the doctor and the most frustrating thing - we went there by traveling a long distance and then we come to know the doctor is on leave or the doctor can't take appointments.

HMS will help us overcome all these problems because now patients can book appointments at home, they can check whether the doctor they want to meet is available or not. Doctors can also confirm or decline appointments, this helps both patient and the doctor because if the doctor declines the appointment, then patient will know this in advance and patient will visit hospital only when the doctor confirms the appointment this will save time and money of the patient. Patients can also pay the doctor's consultant fee online to save their time.

HMS is essential for all healthcare establishments, be it hospitals, nursing homes, health clinics, rehabilitation centers, dispensaries, or clinics. The main goal is to computerize all the details regarding the patient and the hospital. The installation of this healthcare software results in improvement in administrative functions and hence better patient care, which is the prime focus of any healthcare unit.

1.4- OBJECTIVES AND RESEARCH METHODOLOGY

The system will be used as the application that serves hospitals, clinic, dispensaries or other health institutions. The intention of the system is to increase the number of patients that can be treated and managed properly.

If the hospital management system is file based, management of the hospital has to put much effort on securing the files. They can be easily damaged by fire, insects and natural disasters. Also, could be misplaced by losing data and information.

2 - PROPOSED SYSTEM

2.1- PROPOSED SYSTEM

The project Hospital Management system includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user friendly. The data are well protected for personal use and makes the data processing very fast. Hospital Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals. Hospital Management System is designed for multi-speciality hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting, in a seamless flow. Hospital Management System is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes.

3 - S/W AND H/W REQUIREMENTS

3.1- S/W AND H/W REQUIREMENTS

Software Requirements:

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS:

OPERATING SYSTEM: Windows XP/7/8/10/ 11
FRONTEND: Html, CSS, JavaScript, ReactJs.
BACKEND: ExpressJs
DATABASE: MySQL

Hardware Requirements:

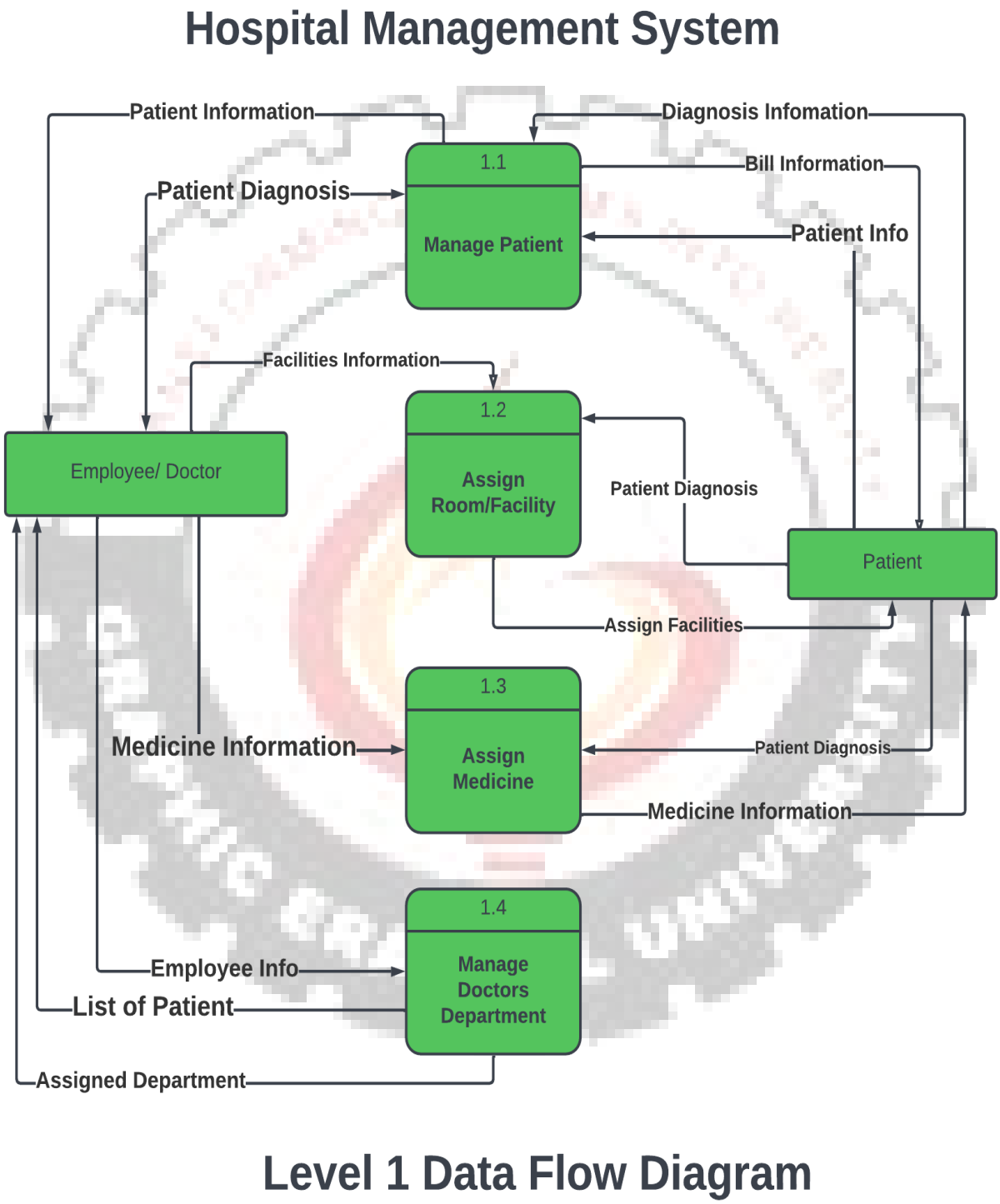
The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

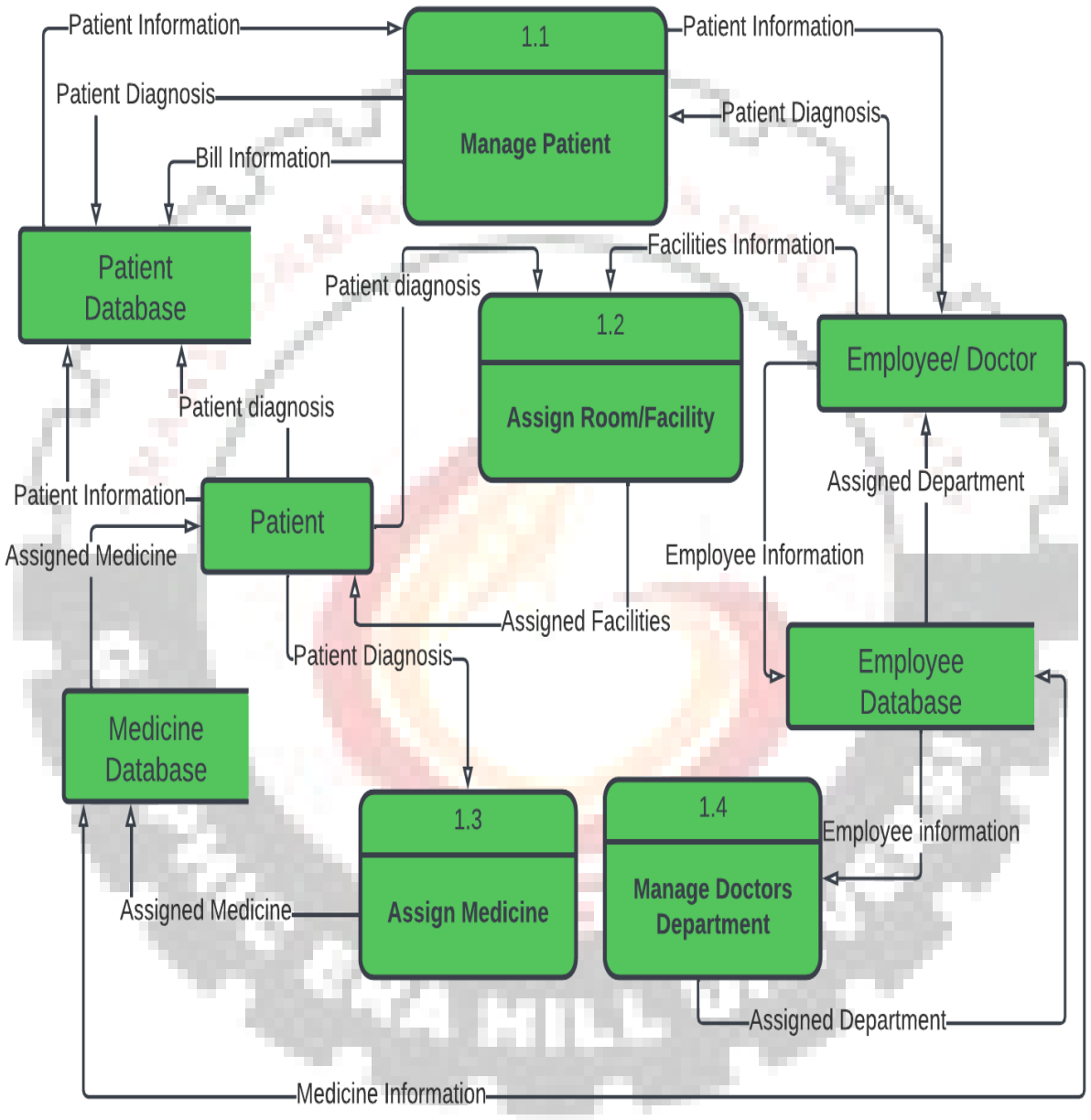
HARDWARE REQUIREMENTS:

PROCESSOR: Intel dual Core i3
RAM: 4 GB
HARD DISK: 50GB

4 - DATA FLOW DIAGRAMS

4.1- DATA FLOW DIAGRAMS





Level 2 Data Flow Diagram

5 - CODING OF FUNCTIONS

5.1- Client

//Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Map Hospital</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

//App.js

```
import React from 'react';
import { Route, BrowserRouter as Router } from 'react-router-dom';
import './App.css';
import Home from './component/Home/Home';
import About from './component/About/About';
import Gallery from './component/Gallery/Gallery';
import PatientLogin from './component/Login/PatientLogin';
import DoctorLogin from './component/Login/DoctorLogin';
import EmployeeLogin from './component/Login/EmployeeLogin';
import AdministratorLogin from './component/Login/AdministratorLogin';
import Contact from './component/Contact/Contact';
import PatientHome from './component/Patient/PatientHome'
import PatHome from './component/Patient/PatHome'
import PatientAbout from './component/Patient/PatientAbout'
import PatientGallery from './component/Patient/PatientGallery'
import PatientContact from './component/Patient/PatientContact'
import Doctor from './component/Doctor/Doctor'
import DocGallery from './component/Doctor/DocGallery'
import Employee from './component/Employee/Employee'
import Admin from './component/Admin/Admin'
import AdHome from './component/Admin/AdHome'
import AdminAbout from './component/Admin/AdminAbout'
```

```
import AdminGallery from './component/Admin/AdminGallery';
import AdminContact from './component/Admin/AdminContact';
import PatientReg from './component/PatientReg/PatientReg';
import DocHome from './component/Doctor/DocHome';
import EmpHome from './component/Employee/EmpHome';
import EmpAbout from './component/Employee/EmpAbout';
import EmpGallery from './component/Employee/EmpGallery';
import EmpContact from './component/Employee/EmpContact';
import DocAbout from './component/Doctor/DocAbout';
import DocContact from './component/Doctor/DocContact';

function App() {
  return (
    <div className="App">
    <Router>
      <Route exact path="/" component={Home} />
      <Route exact path="/about" component={About} />
      <Route exact path="/gallery" component={Gallery} />
      <Route exact path="/contact" component={Contact} />
      <Route exact path="/regPatient" component = {PatientReg}/>
      <Route exact path="/patient/login" component={PatientLogin} />
      <Route exact path="/doctors/login" component={DoctorLogin} />
      <Route exact path="/employee/login" component={EmployeeLogin} />
      <Route exact path="/administrator/login" component={AdministratorLogin} />
      <Route exact path="/patient/login/patient_home" component = {PatientHome} />
      <Route exact path="/patient/login/home" component = {PatHome} />
      <Route exact path="/patient/login/about" component = {PatientAbout} />
      <Route exact path="/patient/login/gallery" component = {PatientGallery} />
      <Route exact path="/patient/login/contact" component = {PatientContact} />
      <Route exact path="/doctors/login/doctor_home" component = {Doctor} />
      <Route exact path="/doctors/login/home" component = {DocHome} />
      <Route exact path="/doctors/login/about" component = {DocAbout}/>
      <Route exact path="/doctors/login/gallery" component = {DocGallery}/>
      <Route exact path="/doctors/login/contact" component = {DocContact}/>
      <Route exact path="/employee/login/employee_home" component = {Employee} />
      <Route exact path="/employee/login/home" component = {EmpHome}/>
      <Route exact path="/employee/login/about" component = {EmpAbout}/>
      <Route exact path="/employee/login/gallery" component = {EmpGallery}/>
      <Route exact path="/employee/login/contact" component = {EmpContact}/>
      <Route exact path="/administrator/login/admin_home" component = {Admin} />
      <Route exact path="/administrator/login/home" component = {AdHome}/>
      <Route exact path="/administrator/login/about" component = {AdminAbout}/>
      <Route exact path="/administrator/login/gallery" component = {AdminGallery}/>
      <Route exact path="/administrator/login/contact" component = {AdminContact}/>
    </Router>
    </div>
  );
}

export default App;
```

//Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
import 'bootstrap/dist/css/bootstrap.min.css';

ReactDOM.render(<App />, document.getElementById('root'));

serviceWorker.unregister();
```

//Components

//About.js

```
import React, { Component } from 'react';
import Navber from '../Navber/Navber';
import { MDBContainer } from 'mdbreact';

import './About.css';
import Homeimage from '../Homeimage';
import Footer from '../Footer';

class About extends Component {

   render() {
      return (
         <div className = "bg-dark">
           <Navber/>
           <Homeimage/><br/><br/>
           <h1 className="head text-white" align="center"> About Us </h1><br/><br/>
         <MDBContainer>
         <blockquote className="blockquote">
         <p className="text-white"> MAP Hospital Ltd is the premier private healthcare provider of
```

North India, with three super specialty hospitals at Bhimtal, Haldwani, and Dehradun, in Uttarakhand, a state-of-the-art daycare centre at Bhowali, and another super specialty hospital at New Delhi. The Group takes care of around 3.5 lakh patients annually, conducting more than 15,000 successful surgeries, with a roster of more than 5,000 healthcare professionals. An impressive roster of more than 600 doctors and a well-trained force of nursing staff work tirelessly across more than 1,000 beds across its four hospitals, backed by advanced technologies and latest equipment to treat people and save lives. With major changes and developments on its plate, the MAP Hospital Group is headed towards a path of steady growth. MAP Hospitals Ltd is all set to add around 700 beds in the near future.

MAP Hospital brings to the table a class of its own, making the healthcare group a major player in keeping North India ahead of the curve. The Group treats more than 3.5 lakh people every year and conducts around 15,000 surgeries annually, providing value-for-money services, backed by advanced equipment and latest technologies. The hallmark of MAP Hospitals is a committed team of doctors and caregivers, who take up the challenge of treating the most complicated cases, besides stepping up to handle Emergency and Critical Care with dedication, expertise and compassion. Over the last three

decades, the healthcare Group has made immense contributions to the lives of people, not just from Uttarakhand and the rest of North India, but also from other parts of South Asia and the world.

```
            </p>
            </blockquote>
            </MDBContainer> <br>  </br>
            <Footer/></div>
          );
    }
}

export default About;
```

//About.css

```
.default-color-dark{
    background-color: #00695c;
}
.default-color{
    background-color: #2BBBAD;
}

 .amber-darken-3{
    background-color: #ff8f00;
    background-size: contain;
}

 .amber-lighten-4{
    background-color:#ffecb3
 }

 .light-green-accent-3{
     background-color:  #76ff03;
 }
```

//Contact.js

```
import React, { Component } from "react";
import "./Contact.css";
import Navber from "../Navber/Navber";
import Footer from "../Footer";
import Homeimage from "../Homeimage";

class Contact extends Component {
  render() {
    return (
      <div className="bg-dark">
        <Navber />
        <div className="md-5">
          <Homeimage /></div><br /><br /><br /><br />
        <div className="mt-5">
          <Footer /></div></div>
```

```
    );
  }
}

export default Contact;
```

//Contact.css

```
 .amber-darken-3{
    background-color: #ff8f00;
    background-size: contain;
}
 .amber-lighten-4{
    background-color:#ffecb3
 }
 .orange-lighten-3{
    background-color: #ffcc80;
}
```

//Gallery.js

```
import React, { Component } from 'react';
import Navber from '../Navber/Navber';
import GalleryCarousel from '../GalleryCarousel';

class Gallery extends Component {
    render() {
        return (
        <div className="bg-dark">
            <Navber /><br></br>
            <GalleryCarousel/></div>
);
    }
}
export default Gallery;
```

//Gallery.css

```
.head{
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bolder;
  }
  .amber-darken-3{
    background-color: #ff8f00;
    background-size: contain;
}
.orange-lighten-3{
    background-color: #ffcc80;
}
.amber-darken-3{
    background-color: #ff8f00;
```

```
    background-size: contain;
}
 .amber-lighten-4{
    background-color:#ffecb3
 }
```

//Home.js

```
import React, { Component } from 'react';
import Homeimage from "../Homeimage"
import Navber from '../Navber/Navber';
import './Home.css';
import OurDoctors from '../OurDoctors';
import Footer from '../Footer';
import HomeQuote from '../HomeQuote';
import Mission from '../Mission';
class Home extends Component {

    render() {
        return (
            <div className = "bg-dark">
                <Navber />
                <Homeimage/>
                <Mission/><br></br>
                <h1 className="head text-white" align="center"> Our Doctors </h1>
                <br /><br/>
                <OurDoctors/><br></br><br></br>
                <h1 className="text-white" align="center">News and Achievements</h1>
                <br/><br/>
                <HomeQuote/><br/><br/>
                <Footer/>
            </div>
        );
    }
}
export default Home;
```

//Login.js

```
import React, { Component } from 'react'
import Navber from '../Navber/Navber';
import axios from 'axios';

class AdministratorLogin extends Component {
  constructor() {
    super()
    this.state = {
      email: '',
      password: '',
      errors: {}
    }
```

```
  this.onChange = this.onChange.bind(this)
  this.onSubmit = this.onSubmit.bind(this)
}

onChange(e) {
  this.setState({ [e.target.name]: e.target.value })
}
onSubmit(e) {
  e.preventDefault()

  const user = {
    email: this.state.email,
    password: this.state.password
  }

  axios.post('/admin/login', {
    email: user.email,
    password: user.password
  }).then(response => {
    if (response.data === "Email not found") return "Email not found";

    sessionStorage.setItem('usertoken', response.data)
    return response.data
  }).then(res => {
    if (res !== "Email not found") {
     sessionStorage.setItem('userData', JSON.stringify(user));
     this.props.history.push('/administrator/login/admin_home');
    }
  }).catch(err => {
    console.log(err)
  })
}
render() {
  return (
    <div className="body">
      <Navber />
      <div className="container my-5">
        <div className="row">
          <div className="col-md-6 mt-5 mx-auto">
            <form noValidate onSubmit={this.onSubmit} >
              <h1 className="h3 mb-3 mt-5 font-weight-normal btn-rg">Please sign in as Admin</h1>
              <div className="form-group btn-rg">
                <label htmlFor="email" >Email address</label>
                <input
                  type="email"
                  className="form-control"
                  name="email"
                  placeholder="Enter email"
                  value={this.state.email}
                  onChange={this.onChange}
```

```
              />
            </div>
            <div className="form-group btn-rg">
              <label htmlFor="password">Password</label>
              <input
                type="password"
                className="form-control"
                name="password"
                placeholder="Password"
                value={this.state.password}
                onChange={this.onChange}
              />
            </div>
            <button
              type="submit"
              className="btn btn-lg btn-primary btn-block mb-5"
            >
              Sign in
            </button>
          </form>
        </div>
      </div>
    </div>
    <div className="mb-5 mt-5">v</div>
  </div>
 )
 }
}
export default AdministratorLogin;
```

//Navber.js

```
import React, { Component } from 'react';
import classnames from 'classnames'
import './Navber.css';
import Navbar from 'react-bootstrap/Navbar';
import Nav from 'react-bootstrap/Nav';
import NavDropdown from 'react-bootstrap/NavDropdown';

class Navber extends Component {
 constructor(props) {
  super(props);

  this.state = {
   prevScrollpos: window.pageYOffset,
   visible: true
  };
 }

 // Adds an event listener when the component is mount.
 componentDidMount() {
```

```
  window.addEventListener("scroll", this.handleScroll);
 }

 // Remove the event listener when the component is unmount.
 componentWillUnmount() {
  window.removeEventListener("scroll", this.handleScroll);
 }

 // Hide or show the menu.
 handleScroll = () => {
  const { prevScrollpos } = this.state;

  const currentScrollPos = window.pageYOffset;
  const visible = prevScrollpos > currentScrollPos;

  this.setState({
   prevScrollpos: currentScrollPos,
   visible
  });
 };

  render() {
     return (
       <div>
         <Navbar className={classnames("navbar", {
           "navbar--hidden": !this.state.visible
          })} bg="primary" text="white" var expand="lg">
         <Navbar.Brand style={{color:"white"}}>MAP Hospital Ltd</Navbar.Brand>
         <Navbar.Toggle aria-controls="basic-navbar-nav" />
         <Navbar.Collapse id="basic-navbar-nav">
           <Nav className="ml-auto">
           <Nav.Link className="text-white" href="/">Home</Nav.Link>
           <Nav.Link className="text-white" href="/about">About</Nav.Link>
           <Nav.Link className="text-white" href="/gallery">Gallery</Nav.Link>
           <Nav.Link className="text-white" href="/contact">Contact Us</Nav.Link>
           <Nav.Link className="text-white" href="/regPatient">Patient Register</Nav.Link>
           <NavDropdown title="Login" id="basic-nav-dropdown" className="mr-5">
             <NavDropdown.Item href="/patient/login">Patient Login</NavDropdown.Item>
             <NavDropdown.Item href="/doctors/login">Doctor Login</NavDropdown.Item>
             <NavDropdown.Item href="/employee/login">Employee Login</NavDropdown.Item>
             <NavDropdown.Item href="/administrator/login">Admin Login</NavDropdown.Item>
           </NavDropdown>
           </Nav>

         </Navbar.Collapse>
         </Navbar>
       </div>
     );
  }
}
```

```
export default Navber;
```

//Navber.css

```
.color_navber {
    color: #5162dd;
}

.light-green darken-1{
    color:#7cb342 ;
}

.navbar {
    width: 100%;
    padding: 10px;
    position: fixed;
    top: 0;
    transition: top 0.6s;
  }

  .navbar--hidden {
    top: -50px;
  }
```

//PatientReg.js

```
import React, { Component } from 'react';
import Navber from '../Navber/Navber';
import Footer from '../Footer';
import axios from 'axios';

class Register extends Component {
  constructor() {
    super()
    this.state = {
      first_name: '',
      last_name: '',
      email: '',
      password: '',
      address: '',
      phone_no: '',
      disease: '',
      errors: {}
    }

    this.onChange = this.onChange.bind(this)
    this.onSubmit = this.onSubmit.bind(this)
  }

  onChange(e) {
```

```
  this.setState({ [e.target.name]: e.target.value })
}
onSubmit(e) {
  e.preventDefault()

  const newUser = {
    first_name: this.state.first_name,
    last_name: this.state.last_name,
    email: this.state.email,
    password: this.state.password,
    address: this.state.address,
    phone_no: this.state.phone_no,
    disease: this.state.disease
  }

  axios.post('/patient/register', newUser)
    .then(responce => {
      console.log('Registered');
      return responce.data;
    })
    .then(res => {
      if(res === 'user already exist...') {
        this.setState({errors: res})
      }else {
        this.props.history.push(`/patient/login`)
      }
    })
}

render() {
  return (
      <div className="body">
      <Navber />
    <div className="container my-5">
      <div className="row">
        <div className="col-md-6 mt-5 mx-auto">
          <form noValidate onSubmit={this.onSubmit} >
            <h1 className="h3 mb-3 font-weight-normal btn-rg">Register</h1>
            <div className="form-group">
              <label htmlFor="name">First name</label>
              <input
                type="text"
                className="form-control"
                name="first_name"
                placeholder="Enter your first name"
                value={this.state.first_name}
                onChange={this.onChange}
              />
            </div>
            <div className="form-group">
              <label htmlFor="name">Last name</label>
```

```
  <input
    type="text"
    className="form-control"
    name="last_name"
    placeholder="Enter your last name"
    value={this.state.last_name}
    onChange={this.onChange}
  />
</div>
<div className="form-group">
  <label htmlFor="email">Email address</label>
  <input
    type="email"
    className="form-control"
    name="email"
    placeholder="Enter email"
    value={this.state.email}
    onChange={this.onChange}
  />
</div>
<div className="form-group">
  <label htmlFor="password">Password</label>
  <input
    type="password"
    className="form-control"
    name="password"
    placeholder="Password"
    value={this.state.password}
    onChange={this.onChange}
  />
</div>
<div className="form-group">
  <label htmlFor="name">Address</label>
  <input
    type="text"
    className="form-control"
    name="address"
    placeholder="Enter your Resident Address"
    value={this.state.address}
    onChange={this.onChange}
  />
</div>
<div className="form-group">
  <label htmlFor="name">Phone Number</label>
  <input
    type="text"
    className="form-control"
    name="phone_no"
    placeholder="Enter your Phone Number"
    value={this.state.phone_no}
    onChange={this.onChange}
```

```
            />
          </div>
          <div className="form-group">
            <label htmlFor="name">Disease</label>
            <input
              type="text"
              className="form-control"
              name="disease"
              placeholder="Enter your Phone Number"
              value={this.state.disease}
              onChange={this.onChange}
            />
          </div>
          <button
            type="submit"
            className="btn btn-lg btn-primary btn-block"
          >
            Register!
          </button>
        </form>
      </div>
    </div>
  </div>

  <Footer />
  </div>
 )
 }
}

export default Register;
```

5.2- SERVER

//server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

const Users = require('./routes/users');
const Patient = require('./routes/users/Patient/patient');
const Employee = require('./routes/users/Employee/employee');
const Doctor = require('./routes/users/Doctor/docter');
const Admin = require('./routes/users/Administrator/admin');
const superAdmin = require('./routes/users/Administrator/superAdmin');
const api = require('./routes/api/api');


const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

app.use('/users', Users);
app.use('/patient', Patient);
app.use('/employee', Employee);
app.use('/doctor' , Doctor);
app.use('/admin', Admin);
app.use('/super', superAdmin);
app.use('/api', api);

app.use(express.static(path.join(__dirname + '/client/build')));

app.use('/*', (req, res) => {
   res.sendFile(path.join(__dirname + '/client/build/index.html'));
});


const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server started at port ${PORT}`));
```

//db.js

```
const mysql = require('mysql');

const db = mysql.createConnection({
   host    : 'localhost',
   user    : 'root',
   password: 'pratish@Mysql04',
   database: 'hospitalmanagementsystem'
});
```

```
db.connect((err) => {
   if(err) throw err;
   console.log("MySQL connected");
});

module.exports = db;
```

//users.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const users = express.Router();


const db = require('../utils/db');

process.env.SECRET_KEY = 'map';

users.post('/register', (req, res) => {

   const userData = {
      first_name  : req.body.first_name,
      last_name   : req.body.last_name,
      email       : req.body.email,
      password    : req.body.password
   }

   let find = `SELECT * FROM users WHERE email = "${userData.email}"`;

   db.query(find, (err1, result1) => {
      if(err1) console.log(err1);
      console.log(result1[0]);

      if(result1[0] == undefined) {
         bcrypt.hash(req.body.password, 10, (err, hash) => {
            userData.password = hash;

            let create = `INSERT INTO users (firstname, lastname, email, password)
                        VALUES ( "${userData.first_name}",
                              "${userData.last_name}",
                              "${userData.email}",
                              "${userData.password}")`;

            db.query(create, (err2, result2) => {
               if(err2) console.log(err2);
               res.send("Created Database ooooooooooooohhhhhh");
            })
         });
      }else {
         res.send("user already exist...");
```

```
        }
    });
});

users.get('/login', (req, res) => {
    let find = `SELECT password, user_id FROM users WHERE email = "${req.body.email}"`;

    db.query(find, (err, result) => {
        if(err) console.log(err);
        console.log(result);

        if(result[0] != undefined) {
            if(bcrypt.compareSync(req.body.password, result[0].password)) {
                let token = jwt.sign(result[0].user_id, process.env.SECRET_KEY);
                res.send(token);
            } else {
                res.send('Password incorrect');
            }
        } else {
            res.send("Email not found");
        }
    });
});

users.get('/profile', (req, res) => {
    let user_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let user = `SELECT * FROM users WHERE user_id = ${user_id}`;
    db.query(user, (err, result) => {
        if (err) console.log(err);
        res.send(result);
    });
});


module.exports = users;
```

//admin.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const admin = express.Router();

const db = require('../../../utils/db');

process.env.SECRET_KEY = 'map';

admin.post('/register', (req, res) => {

    const adminData = {
```

```
        first_name  : req.body.first_name,
        last_name   : req.body.last_name,
        email       : req.body.email,
        phone_no    : req.body.phone_no,
        designation : req.body.designation,
        password    : req.body.password,
        address     : req.body.address,
        salary      : req.body.salary
    }

    console.log(adminData)

    let find = `SELECT * FROM admin WHERE email = "${adminData.email}"`;

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);
        console.log(result1[0]);

        if(result1[0] == undefined) {
            bcrypt.hash(req.body.password, 10, (err, hash) => {
             adminData.password = hash;

              let create = `INSERT INTO admin (first_name, last_name, email, phone_no, designation,
password, salary, address)
                        VALUES ( "${adminData.first_name}",
                                 "${adminData.last_name}",
                                 "${adminData.email}",
                                 "${adminData.phone_no}",
                                 "${adminData.designation}",
                                 "${adminData.password}",
                                 ${adminData.salary},
                                 "${adminData.address}"
                                 )`;

                db.query(create, (err2, result2) => {
                    if(err2) console.log(err2);
                    res.send("admin registered");
                })
            });
        }else {
            res.send("admin already exist...");
        }
    });
});

admin.post('/login', (req, res) => {
    let find = `SELECT password, admin_id FROM admin WHERE email = "${req.body.email}"`;

    db.query(find, (err, result) => {
        if(err) console.log(err);
        // console.log(result);
```

```
        if(result[0] != undefined) {
            if(bcrypt.compareSync(req.body.password, result[0].password)) {
                let token = jwt.sign(result[0].admin_id, process.env.SECRET_KEY);
                res.send(token);
            } else {
                res.send('Password incorrect');
            }
        } else {
            res.send("Email not found");
        }
    });
});

admin.get('/details', (req, res) => {
    let user_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let user = `SELECT * FROM admin WHERE admin_id = ${user_id}`;
    db.query(user, (err, result) => {
        if (err) console.log(err);
        res.send(result);
    });
});

admin.post('/delete', (req, res) => {
    const find = `SELECT * FROM admin WHERE admin_id = ${req.body.admin_id}`;
    let del =  `DELETE FROM admin WHERE admin_id = ${req.body.admin_id}`

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);

        if(result1[0] != undefined) {
            db.query(del, (err2, result2) => {
                res.send('DELETED');
            })
        }
    })
})

admin.post('/assign_doctor', (req, res) => {
    const data = {
        patient_id: req.body.patient_id,
        doctor_id: req.body.doctor_id
    };

    const sql = `SELECT * FROM assign_doctor WHERE patient_id = "${data.patient_id}"`;

    db.query(sql, (err1, result1) => {
        if(err1) console.log(err1);

        if(result1[0] == undefined) {
```

```
            const create = `INSERT INTO assign_doctor (patient_id, doctor_id)
                        VALUES ( "${data.patient_id}",
                              "${data.doctor_id}"
                        )`
            db.query(create, (err2, result2) =>{
               if(err2) console.log(err2);
               res.send("Yes   ")
            })
         }else {
            res.send("already exist...");
         }
      })
});

admin.post('/bill', (req, res) => {
   const data = {
      patient_email: req.body.patient_email,
      medicine_cost: req.body.medicine_cost,
      room_charge: req.body.room_charge,
      misc_charge: req.body.misc_charge,
      operation_charge: req.body.operation_charge,
   }

   const sql = `SELECT * FROM patient WHERE email = "${data.patient_email}"`;


   db.query(sql, (err1, result1) => {
      if(err1) console.log(err1);


      if(result1[0] !== undefined) {

         const update = `
                        UPDATE bill
                        SET
                           medicine_cost = medicine_cost + ${data.medicine_cost},
                           operation_charge = operation_charge + ${data.operation_charge},
                           room_charge = room_charge + ${data.room_charge},
                           misc_charge = misc_charge + ${data.misc_charge}
                        WHERE patient_id = ${result1[0].patient_id}
                        `
         console.log(update);

         db.query(update, (err2, result2) =>{
            if(err2) console.log(err2);
            res.send("Yes   ")
         })
      }else {
         res.send("already exist...");
      }
   })
```

```
});


module.exports = admin;
```

//doctor.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const doctor = express.Router();

const db = require('../../../utils/db');

process.env.SECRET_KEY = 'map';

doctor.post('/register', (req, res) => {

    const doctorData = {
        first_name      : req.body.first_name,
        last_name       : req.body.last_name,
        address         : req.body.address,
        email           : req.body.email,
        salary          : req.body.salary,
        specialisation  : req.body.specialisation,
        shift_time      : req.body.shift_time,
        password        : req.body.password
    }

    let find = `SELECT * FROM doctors WHERE email = "${doctorData.email}"`;

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);
        console.log(result1[0]);

        if(result1[0] == undefined) {
            bcrypt.hash(req.body.password, 10, (err, hash) => {
                doctorData.password = hash;

                let create = `INSERT INTO doctors (first_name, last_name, address, email, salary,
specialisation, shift_time, password)
                        VALUES ( "${doctorData.first_name}",
                                "${doctorData.last_name}",
                                "${doctorData.address}",
                                "${doctorData.email}",
                                "${doctorData.salary}",
                                "${doctorData.specialisation}",
                                "${doctorData.shift_time}",
                                "${doctorData.password}")`;

                db.query(create, (err2, result2) => {
```

```
                if(err2) console.log(err2);
                res.send("doctor registered");
            })
        });
    }else {
        res.send("doctor already exist...");
    }
  });
});

doctor.post('/login', (req, res) => {
  let find = `SELECT password, doctor_id FROM doctors WHERE email = "${req.body.email}"`;

  db.query(find, (err, result) => {
    if(err) console.log(err);
    console.log(result);

    if(result[0] != undefined) {
      if(bcrypt.compareSync(req.body.password, result[0].password)) {
        let token = jwt.sign(result[0].doctor_id, process.env.SECRET_KEY);
        res.send(token);
      } else {
        res.send('Password incorrect');
      }
    } else {
      res.send("Email not found");
    }
  });
});

doctor.get('/patient', (req,res) => {
  let doctor_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

  const sql = `SELECT
                p.patient_id,
                p.first_name,
                p.last_name
              FROM assign_doctor ad
                JOIN patient p ON p.patient_id = ad.patient_id
                JOIN doctors d ON d.doctor_id = ad.doctor_id
              WHERE ad.doctor_id = ${doctor_id}

              `
  console.log(sql);
  db.query(sql, (err, result) => {
    if (err) console.log(err);
    res.send(result);

  });
})
```

```
doctor.get('/profile', (req, res) => {
    let doctor_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let user = `SELECT * FROM doctors WHERE doctor_id = ${doctor_id}`;
    db.query(user, (err, result) => {
        if (err) console.log(err);
        res.send(result);

    });
});

doctor.post('/delete', (req, res) => {
    const find = `SELECT * FROM doctors WHERE doctor_id = ${req.body.doctor_id}`;
    let del =  `DELETE FROM doctors WHERE doctor_id = ${req.body.doctor_id}`

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);

        if(result1[0] != undefined) {
            db.query(del, (err2, result2) => {
                res.send('DELETED');
            })
        }
    })
});

doctor.post('/update_sal', (req, res) => {
    const find = `SELECT * FROM doctors WHERE doctor_id = ${req.body.doctor_id}`;
    const upd = `UPDATE doctors
                SET salary =" ${req.body.salary}"
                WHERE doctor_id = ${req.body.doctor_id}`;

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);

        if(result1[0] != undefined) {
            db.query(upd, (err2, result2) => {
                res.send('UPDATED');
            })
        }
    })
})

module.exports = doctor;
```

//employee.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const users = express.Router();
```

```
const db = require('../../../utils/db');

process.env.SECRET_KEY = 'map';

users.post('/register', (req, res) => {

    const userData = {
        first_name  : req.body.first_name,
        last_name   : req.body.last_name,
        email       : req.body.email,
        password    : req.body.password
    }

    let find = `SELECT * FROM users WHERE email = "${userData.email}"`;

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);
        console.log(result1[0]);

        if(result1[0] == undefined) {
            bcrypt.hash(req.body.password, 10, (err, hash) => {
                userData.password = hash;

                let create = `INSERT INTO users (firstname, lastname, email, password)
                            VALUES ( "${userData.first_name}",
                                    "${userData.last_name}",
                                    "${userData.email}",
                                    "${userData.password}")`;

                db.query(create, (err2, result2) => {
                    if(err2) console.log(err2);
                    res.send("Registration successful");
                })
            });
        }else {
            res.send("user already exist...");
        }
    });
});

users.get('/login', (req, res) => {
    let find = `SELECT password, user_id FROM users WHERE email = "${req.body.email}"`;

    db.query(find, (err, result) => {
        if(err) console.log(err);
        console.log(result);

        if(result[0] != undefined) {
            if(bcrypt.compareSync(req.body.password, result[0].password)) {
                let token = jwt.sign(result[0].user_id, process.env.SECRET_KEY);
```

```
            res.send(token);
          } else {
            res.send('Password incorrect');
          }
        } else {
          res.send("Email not found");
        }
    });
});

users.get('/profile', (req, res) => {
    let user_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let user = `SELECT * FROM users WHERE user_id = ${user_id}`;
    db.query(user, (err, result) => {
        if (err) console.log(err);
        res.send(result);
    });
});


module.exports = users;
```

//patient.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const patient = express.Router();

const db = require('../../../utils/db');

process.env.SECRET_KEY = 'map';

patient.post('/register', (req, res) => {

    const patientData = {
        first_name  : req.body.first_name,
        last_name   : req.body.last_name,
        address     : req.body.address,
        email       : req.body.email,
        phone_no    : req.body.phone_no,
        password    : req.body.password,
        disease     : req.body.disease
    }

    let find = `SELECT * FROM patient WHERE email = "${patientData.email}"`;

    db.query(find, (err1, result1) => {
        if(err1) console.log(err1);
        //console.log(result1[0]);
```

```
        if(result1[0] == undefined) {
            bcrypt.hash(req.body.password, 10, (err, hash) => {
                patientData.password = hash;

                let create = `INSERT INTO patient (first_name, last_name, address, email, phone_no,
password, disease)
                            VALUES ( "${patientData.first_name}",
                                    "${patientData.last_name}",
                                    "${patientData.address}",
                                    "${patientData.email}",
                                    "${patientData.phone_no}",
                                    "${patientData.password}",
                                    "${patientData.disease}")`;

                db.query(create, (err2, result2) => {

                    db.query(find, (err3, result3)=> {
                        const patient_id = result3[0].patient_id;

                        let bill = `INSERT INTO bill (patient_id) VALUES ("${patient_id}")`;

                        db.query(bill, (err4, result4)=>{
                            console.log("OK");
                        })
                    })


                    if(err2) console.log(err2);
                    res.send("Created Database ooooooooooooohhhhhh");
                })
            });
        }else {
            res.send("user already exist...");
        }
    });
});

patient.post('/login', (req, res) => {
    let find = `SELECT password, patient_id FROM patient WHERE email = "${req.body.email}"`;

    db.query(find, (err, result) => {
        if(err) console.log(err);
        // console.log(result);

        if(result[0] != undefined) {
            if(bcrypt.compareSync(req.body.password, result[0].password)) {
                let token = jwt.sign(result[0].patient_id, process.env.SECRET_KEY);
                res.send(token);
            } else {
                res.status(400).json({ error: 'User does not exist' })
```

```
        }
      } else {
        res.send("Email not found");
      }
    });
});

patient.get('/profile', (req, res) => {
    let patient_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let patient = `SELECT * FROM patient WHERE patient_id = ${patient_id}`;
    db.query(patient, (err, result) => {
      if (err) console.log(err);
      res.send(result);
    });
});


patient.get('/details', (req, res) => {
    let patient_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let patient =  `SELECT
                    *
                FROM patient
                WHERE patient_id = ${patient_id}`;
    db.query(patient, (err, result) => {
      if (err) console.log(err);
      // console.log(patient_id, result);
      res.send(result);
    });
});

patient.get('/doctor', (req, res) => {
    let patient_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    let patient =  `SELECT
                    d.first_name as doctor_firstname,
                    d.last_name doctor_lastname,
                    d.specialisation
                FROM assign_doctor ad
                JOIN patient p
                    ON p.patient_id = ad.patient_id
                JOIN doctors d
                    ON ad.doctor_id = d.doctor_id
                WHERE p.patient_id = ${patient_id}`;
    db.query(patient, (err, result) => {
      if (err) console.log(err);
      // console.log(patient_id, result);
      res.send(result);
    });
})
```

```
patient.get('/bill', (req, res) => {
    let patient_id = jwt.verify(req.headers['authorization'], process.env.SECRET_KEY);

    const bill = `SELECT * FROM bill WHERE patient_id = ${patient_id}`;

    db.query(bill, (err, result) => {
        res.send(result);
    })
})


module.exports = patient;
```

6 - LIMITATIONS AND ENHANCEMENTS

- The system currently lacks an appointment booking module. It appears to be an essential feature of a hospital management system. Look forward to equip this system with this functionality in the future.
- The system also lacks an email verification module which is quite essential as it could lead to frauds if users are registered with non-existing or fraud emails.

7 - CONCLUSION

Since we are entering details of the patients electronically in the" Hospital Management System", data will be secured. Using this application, we can retrieve patient's history with a single click. Thus, processing information will be faster. It guarantees accurate maintenance of Patients' details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

REFERENCES

- https://www.reactjs.org/
- https://www.expressjs.com/
- https://www.axios-http.com/
- https://www.geeksforgeeks.org/
- https://www.javatpoint.com/
- https://www.w3schools.com/