ONLINE BUS RESERVATION SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY

DANUSHNARAYAN S	**221701012**
HARRSHAVARDHANN S	**221701018**
JAYASABHAREESH N	**221701024**
KARTHICK N	**221701028**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND DESIGN

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM
CHENNAI – 602105

ANNA UNIVERSITY: CHENNAI 600625

BONAFIDE CERTIFICATE

Certified that this project report **"ONLINE BUS RESERVATION SYSTEM"** is the Bonafide work of **"DANUSHNARAYAN S(221701012), HARRSHAVARDHANN S (221701018), JAYASABHAREESH N(221701024),KARTHICK N(221701028)**" who carried out the project work under my supervision.

SIGNATURE

Mr.S.Uma Maheshwara Rao MA.,MFA.,
Professor and Head,
Computer Science and Design,
Rajalakshmi Engineering College,
Thndalam, Chennai – 602105.

SIGNATURE

Mr.R.Vijaykumar M.Tech.,
Asst. Professor (SS),
Computer Science and Design,
Rajalakshmi Engineering College,
Thandalam, Chennai – 602105.

EXTERNAL EXAMINER

INTERNAL EXAMINER

ACKNOWLEGEMENT

We are highly obliged in taking the opportunity to thank our Chairman **Mr. S. Meganathan**, Chairperson **Dr.Thangam Meganathan** and our Principal **Dr.S.N.Murugesan** for providing all the facilities which are required to carry out this project work.

We are ineffably indebted to our H.O.D **Mr.S.Uma Maheshwara Rao MA., MFA.,**for his conscientious guidance and encouragement to make this project a recognizable one.

We are extremely thankful to our faculty **Mr.R.Vijaykumar M.Tech.,** for his valuable guidance and indefatigable support and extend our heartfelt thanks to all the teaching and non-teaching staff of **Computer Science and Design department** who helped us directly or indirectly in the completion of this project successfully.

At last but not least gratitude goes to our friends who helped us compiling the project and finally to god who made all things possible.

Any omission in this brief acknowledgement doesn't mean lack of gratitude.

DANUSHNARAYAN S 221701012

HARRSHAVARDHANN S 221701018

JAYASABHAREESH N 221701024

KARTHICK N 221701028

ABSTRACT

The online bus reservation system is a web-based platform designed to simplify and enhance the process of booking bus tickets for passengers. This system provides an intuitive interface where users can search for available routes, compare prices, select preferred seats, and make secure online payments. By automating the ticketing process, it aims to reduce manual errors, minimize wait times, and offer a more efficient and user-friendly experience for both passengers and bus operators. Key features include user registration, route management, seat reservation, ticket cancellation, real-time seat availability updates, and mobile integration. The system also includes an administrative dashboard for operators to manage schedules, track bookings, and generate reports. The overall goal is to create a seamless, accessible, and reliable bus booking system that enhances operational efficiency and customer satisfaction.

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

1. INTRODUCTION

The online bus reservation system is designed to streamline the booking process for bus services by offering users a convenient, user-friendly platform to view schedules, book tickets, and manage reservations from anywhere. By digitizing this process, the system aims to replace traditional, time-consuming manual methods and eliminate inefficiencies. The project emphasizes ease of use, security, and reliability, improving customer satisfaction and service provider operations.

2. SCOPE OF THE WORK

This project encompasses the design, development, and implementation of a web-based bus reservation system. It includes features such as user registration, bus schedule management, seat selection, ticket booking, payment processing, and ticket cancellation. The system will also provide administrative capabilities for managing routes, schedules, and user data. The scope is limited to enhancing user experience and streamlining operations for mid-sized bus service companies.

3. PROBLEM STATEMENT

The traditional methods of bus ticket booking often involve long queues, limited operational hours, and human errors. These challenges result in inefficiencies, customer dissatisfaction, and loss of business. Existing digital solutions may be fragmented or lack a cohesive user experience. The problem is the need for a centralized, efficient, and reliable online system that meets modern users' expectations and supports both customers and operators effectively.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to develop an online bus reservation system that simplifies and automates the process of booking bus tickets, while enhancing user convenience and improving operational efficiency for service providers. The

project seeks to create a robust, secure, and feature-rich platform that facilitates seamless ticket booking and management.

CHAPTER – 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	React
Back - End	:	JavaScript
Language	:	React,JavaScript

CHAPTER - 3

MODULE DESCRIPTION

This application consists of two modules. When the program runs, it will ask for a confirmation to the login window. The person who interacts can login as an Administrator or as a User. The description of the modules are as follows:

1.Admin login

When the person who interacts tries to login as Admin then he needs to login with his username and password. The administrator only has the power to change and manipulate the data in the database.

2. **User login**

When the person tries to login as a user then he/she will be prompted to enter the number of symptoms and the final result will be printed in the form of table.

CHAPTER - 4

CODING

Sample code for React:

```
import axios from "axios";
import React, { useEffect } from "react";
import { useState } from "react";
import { toast } from "react-toastify";

const AppointmentForm = () => {
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [email, setEmail] = useState("");
  const [phone, setPhone] = useState("");
  const [nic, setNic] = useState("");
  const [dob, setDob] = useState("");
  const [gender, setGender] = useState("");
  const [appointmentDate, setAppointmentDate] = useState("");
  const [department, setDepartment] = useState("snacks");
  const [doctorFirstName, setDoctorFirstName] = useState("");
  const [doctorLastName, setDoctorLastName] = useState("");
  const [address, setAddress] = useState("");
  const [hasVisited, setHasVisited] = useState(false);

  const departmentsArray = [
    "Ac- sleeper",
    "Non-Ac seater",
    "Non-Ac sleeper",
    "Ac-seater",
    "Tour-Buses",
  ];

  const [doctors, setDoctors] = useState([]);
  useEffect(() => {
    const fetchDoctors = async () => {
      try {
        const { data } = await axios.get(
          "http://localhost:4000/api/v1/user/doctors",
          { withCredentials: true }
        );
        console.log("Fetched Doctors:", data.doctors); // Check the structure of the data
        setDoctors(data.doctors || []); // Ensure doctors is an array
      } catch (error) {
        console.error("Error fetching doctors:", error);
      }
    };
    fetchDoctors();
```

```
}, []);

const handleAppointment = async (e) => {
  e.preventDefault();
  try {
    const hasVisitedBool = Boolean(hasVisited);
    const { data } = await axios.post(
      "http://localhost:4000/api/v1/appointment/post",
      {
        firstName,
        lastName,
        email,
        phone,
        nic,
        dob,
        gender,
        appointment_date: appointmentDate,
        department,
        doctor_firstName: doctorFirstName,
        doctor_lastName: doctorLastName,
        hasVisited: hasVisitedBool,
        address,
      },
      {
        withCredentials: true,
        headers: { "Content-Type": "application/json" },
      }
    );
    toast.success(data.message);
    setFirstName(""),
      setLastName(""),
      setEmail(""),
      setPhone(""),
      setNic(""),
      setDob(""),
      setGender(""),
      setAppointmentDate(""),
      setDepartment(""),
      setDoctorFirstName(""),
      setDoctorLastName(""),
      setHasVisited(""),
      setAddress("");
  } catch (error) {
    toast.error(error.response.data.message);
  }
};

return (
  <>
    <div className="container form-component appointment-form">
```

```
<h2>Booking</h2>
<form onSubmit={handleAppointment}>
  <div>
    <input
      type="text"
      placeholder="First Name"
      value={firstName}
      onChange={(e) => setFirstName(e.target.value)}
    />
    <input
      type="text"
      placeholder="Last Name"
      value={lastName}
      onChange={(e) => setLastName(e.target.value)}
    />
  </div>
  <div>
    <input
      type="text"
      placeholder="Email"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
    />
    <input
      type="number"
      placeholder="Mobile Number"
      value={phone}
      onChange={(e) => setPhone(e.target.value)}
    />
  </div>
  <div>
    <input
      type="number"
      placeholder="Number of seats"
      value={nic}
      onChange={(e) => setNic(e.target.value)}
    />
    <input
      type="date"
      placeholder="Date of Birth"
      value={dob}
      onChange={(e) => setDob(e.target.value)}
    />
  </div>
  <div>
    <select value={gender} onChange={(e) => setGender(e.target.value)}>
      <option value="">Select Gender</option>
      <option value="Male">Male</option>
      <option value="Female">Female</option>
    </select>
```

```
          <input
            type="date"
            placeholder="Journey Date"
            value={appointmentDate}
            onChange={(e) => setAppointmentDate(e.target.value)}
          />
        </div>
        <div>
          <select
            value={department}
            onChange={(e) => {
              setDepartment(e.target.value);
              setDoctorFirstName("");
              setDoctorLastName("");
            }}
          >
            {departmentsArray.map((depart, index) => {
              return (
                <option value={depart} key={index}>
                  {depart}
                </option>
              );
            })}
          </select>
          <select
  value={${doctorFirstName} ${doctorLastName}}
  onChange={(e) => {
    const [firstName, lastName] = e.target.value.split(" ");
    setDoctorFirstName(firstName || "");
    setDoctorLastName(lastName || "");
  }}
  disabled={!department}
>
  <option value="">Select Bus </option>
  {doctors.length > 0 ? (
    doctors
      .filter((doctor) => doctor.doctorDepartment === department)
      .map((doctor, index) => (
        <option
          value={${doctor.firstName} ${doctor.lastName}}
          key={index}
        >
          {doctor.firstName} {doctor.lastName}
        </option>
      ))
  ) : (
    <option disabled>No doctors available for the selected department</option>
  )}
</select>
```

```
          </div>
          <textarea
            rows="10"
            value={address}
            onChange={(e) => setAddress(e.target.value)}
            placeholder="Address"
          />
          <div
            style={{
              gap: "10px",
              justifyContent: "flex-end",
              flexDirection: "row",
            }}
          >
            <p style={{ marginBottom: 0 }}>Have you visited before?</p>
            <input
              type="checkbox"
              checked={hasVisited}
              onChange={(e) => setHasVisited(e.target.checked)}
              style={{ flex: "none", width: "25px" }}
            />
          </div>
          <button style={{ margin: "0 auto" }}>GET Booking</button>
        </form>
      </div>
    </>
  );
};


export default AppointmentForm;
```

Sample Code for JavaScript:

```
import { catchAsyncErrors } from "../middleware/catchAsyncErrors.js";
import ErrorHandler from "../middleware/errorMiddleware.js";
import { Appointment } from "../models/appointmentSchema.js";
import { User } from "../models/userSchema.js";

export const postAppointment = catchAsyncErrors(async (req, res, next) => {
  const {
    firstName,
    lastName,
    email,
    phone,
    nic,
    dob,
```

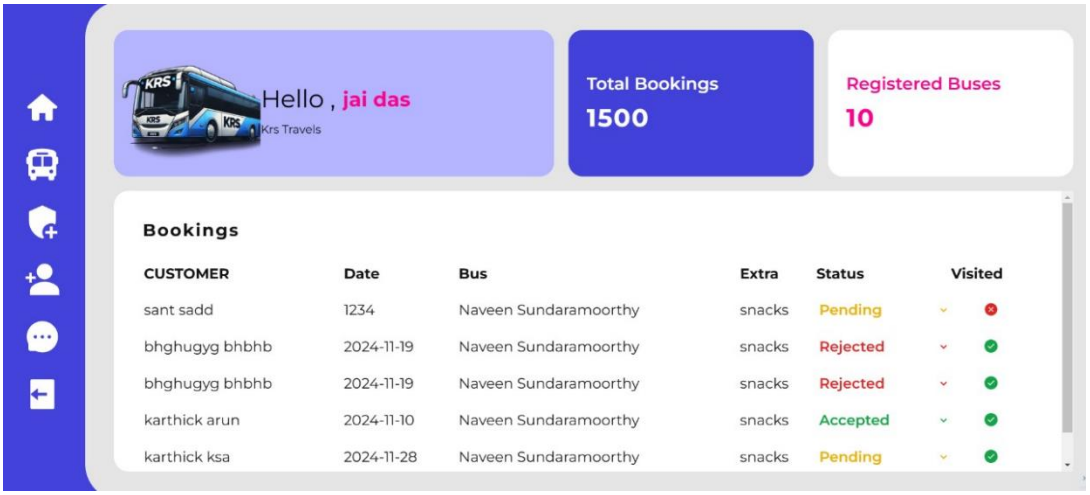
```
  gender,
  appointment_date,
  department,
  doctor_firstName,
  doctor_lastName,
  hasVisited,
  address,
} = req.body;
if (
  !firstName ||
  !lastName ||
  !email ||
  !phone ||
  !nic ||
  !dob ||
  !gender ||
  !appointment_date ||
  !department ||
  !doctor_firstName ||
  !doctor_lastName ||
  !address
) {
  return next(new ErrorHandler("Please Fill Full Form!", 400));
}
const isConflict = await User.find({
  firstName: doctor_firstName,
  lastName: doctor_lastName,
  role: "Doctor",
  doctorDepartment: department,
});
if (isConflict.length === 0) {
  return next(new ErrorHandler("Doctor not found", 404));
}

if (isConflict.length > 1) {
  return next(
    new ErrorHandler(
      "Doctors Conflict! Please Contact Through Email Or Phone!",
      404
    )
  );
}
const doctorId = isConflict[0]._id;
```

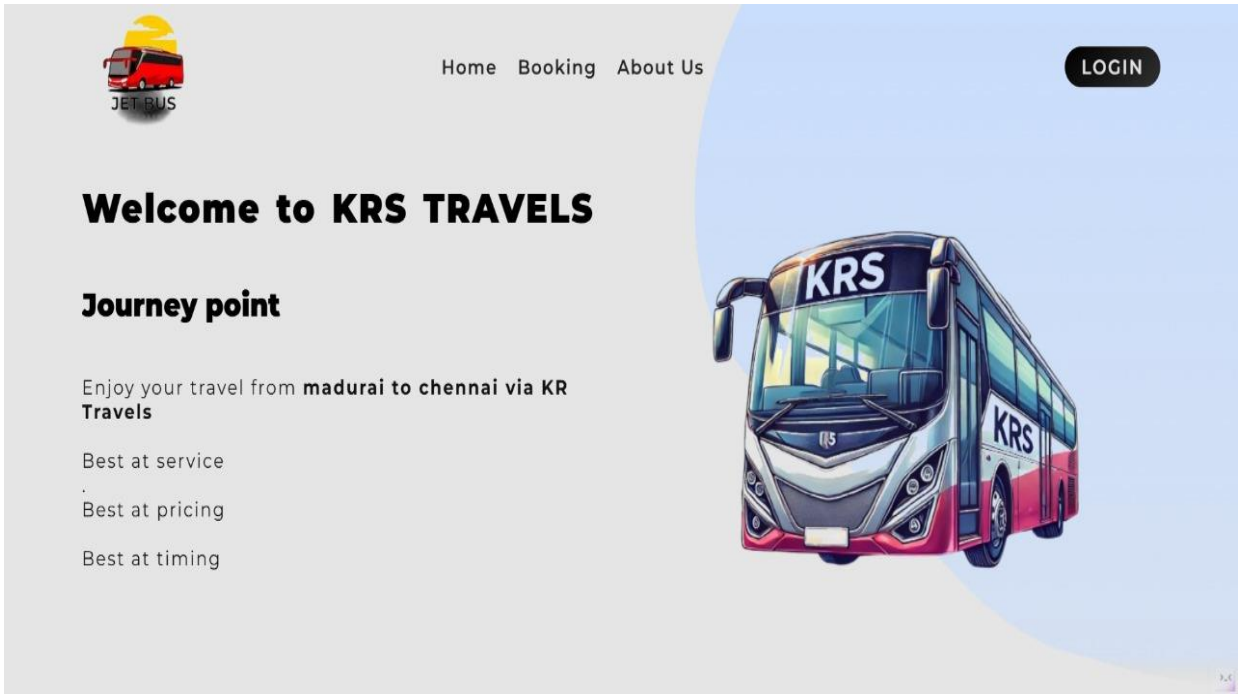
```
  const patientId = req.user._id;
  const appointment = await Appointment.create({
    firstName,
    lastName,
    email,
    phone,
    nic,
    dob,
    gender,
    appointment_date,
    department,
    doctor: {
      firstName: doctor_firstName,
      lastName: doctor_lastName,
    },
    hasVisited,
    address,
    doctorId,
    patientId,
  });
  res.status(200).json({
    success: true,
    appointment,
    message: "Booking Send!",
    appointment,
  });
});
export default postAppointment;


export const getAllAppointments = catchAsyncErrors(async (req, res, next) =>
{
  const appointments = await Appointment.find();
  res.status(200).json({
    success: true,
    appointments,
  });
});
export const updateAppointmentStatus = catchAsyncErrors(
  async (req, res, next) => {
    const { id } = req.params;
    let appointment = await Appointment.findById(id);
    if (!appointment) {
```

```
      return next(new ErrorHandler("Booking not found!", 404));
    }
    appointment = await Appointment.findByIdAndUpdate(id, req.body, {
      new: true,
      runValidators: true,
      useFindAndModify: false,
    });
    res.status(200).json({
      success: true,
      message: "Booking Status Updated!",
    });
  }
);
export const deleteAppointment = catchAsyncErrors(async (req, res, next) => {
  const { id } = req.params;
  const appointment = await Appointment.findById(id);
  if (!appointment) {
    return next(new ErrorHandler("Appointment Not Found!", 404));
  }
  await appointment.deleteOne();
  res.status(200).json({
    success: true,
    message: "Booking  Deleted!",
  });
});
```

CHAPTER - 5

SCREEN SHOTS




JET BUS
Home
Booking
About Us
LOGIN
Welcome to KRS TRAVELS
Journey point
Enjoy your travel from madurai to chennai via KR Travels
Best at service
.
Best at pricing
Best at timing
KRS
KRS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, an online bus reservation system streamlines the booking process by offering a user-friendly platform for customers to plan and purchase tickets efficiently. This system enhances customer satisfaction through real-time seat availability, automated payments, and notifications. For future enhancements, incorporating AI-driven predictive analytics for dynamic pricing, integrating multilingual support, and ensuring better security with advanced encryption can further improve user experience and accessibility. Adding features like loyalty programs, offline booking capabilities, and mobile app integration can help widen the user base and enhance system reliability.

CHAPTER – 7

REFERENCES

1. https://www.w3schools.com/REACT/DEFAULT.ASP
2. https://reactnative.dev/
3. https://www.w3schools.com/js/
4. https://www.codecademy.com/catalog/language/javascript