

# **19Z612 - APPLICATION DEVELOPMENT LABORATORY**

**Adarsh G - 20Z204**

**Jeevan Krishna K V - 20Z220**

**Nirmal M - 20Z267**

**Ajay Deepak P M - 21Z431**

## **VEHICLE RENTAL SYSTEM**



**MARCH 2023**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PSG COLLEGE OF TECHNOLOGY**

**(Autonomous Institution)**

**COIMBATORE – 641004**

# **PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

COIMBATORE – 641004

## **19Z612 - APPLICATION DEVELOPMENT LABORATORY**

Bona fide record of work done by

Adarsh G - 20Z204

Jeevan Krishna K V - 20Z220

Nirmal M - 20Z267

Ajay Deepak P M - 21Z431

### **BACHELOR OF ENGINEERING**

**BRANCH: COMPUTER SCIENCE AND ENGINEERING**

Application Development Laboratory report submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering, Branch: Computer Science and Engineering of PSG College of Technology.

.....

(Faculty Guide)

.....

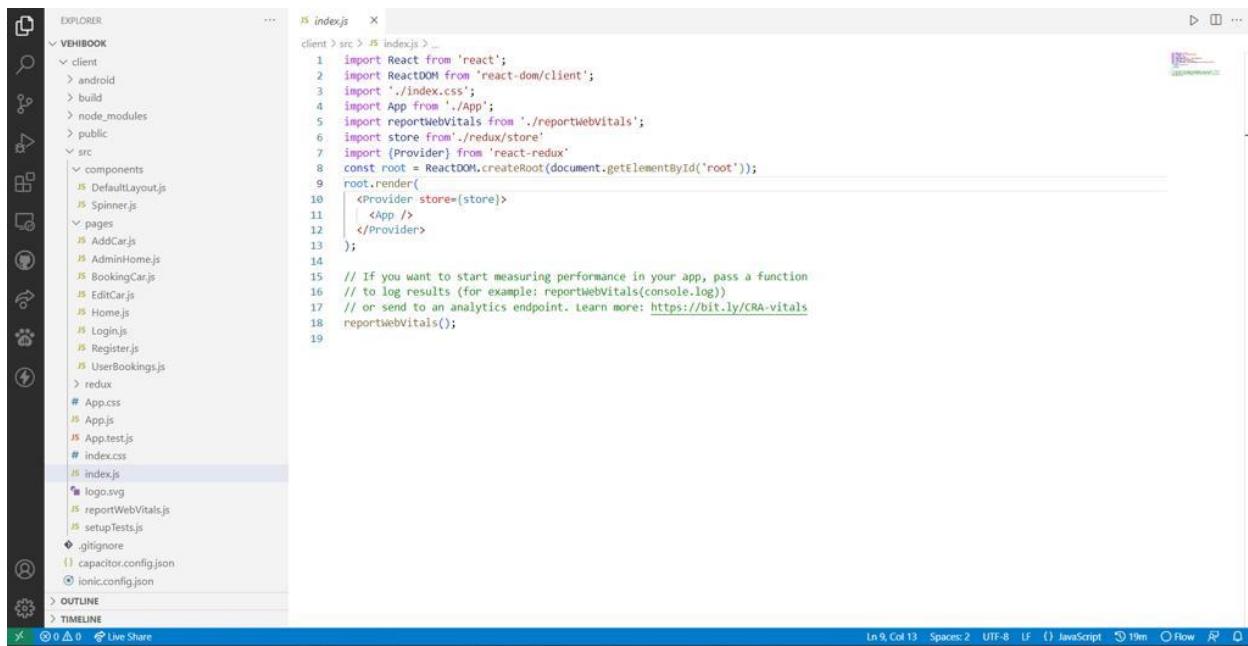
(External Examiner)

# OBJECTIVE

A vehicle rental system's goal is to give customers who need a car for a short period of time a convenient and adaptable mode of transportation. A vehicle rental system's main goals can be broadly divided into the following categories:

- Customer Satisfaction: The primary objective of any vehicle rental system is to provide a high level of customer satisfaction. This can be accomplished by offering customers a seamless and trouble-free rental experience, a variety of vehicles to suit their needs, and top-notch customer service.
- Vehicle Maintenance: The objective of a vehicle rental system is to maintain its vehicles in good condition. To keep the fleet's vehicles safe and roadworthy, this entails routine maintenance and servicing.
- Expansion: A vehicle rental system's expansion of operations and customer base is another goal. To provide it, add new rental locations, collaborate with tour operators, and offer new services to draw more clients.

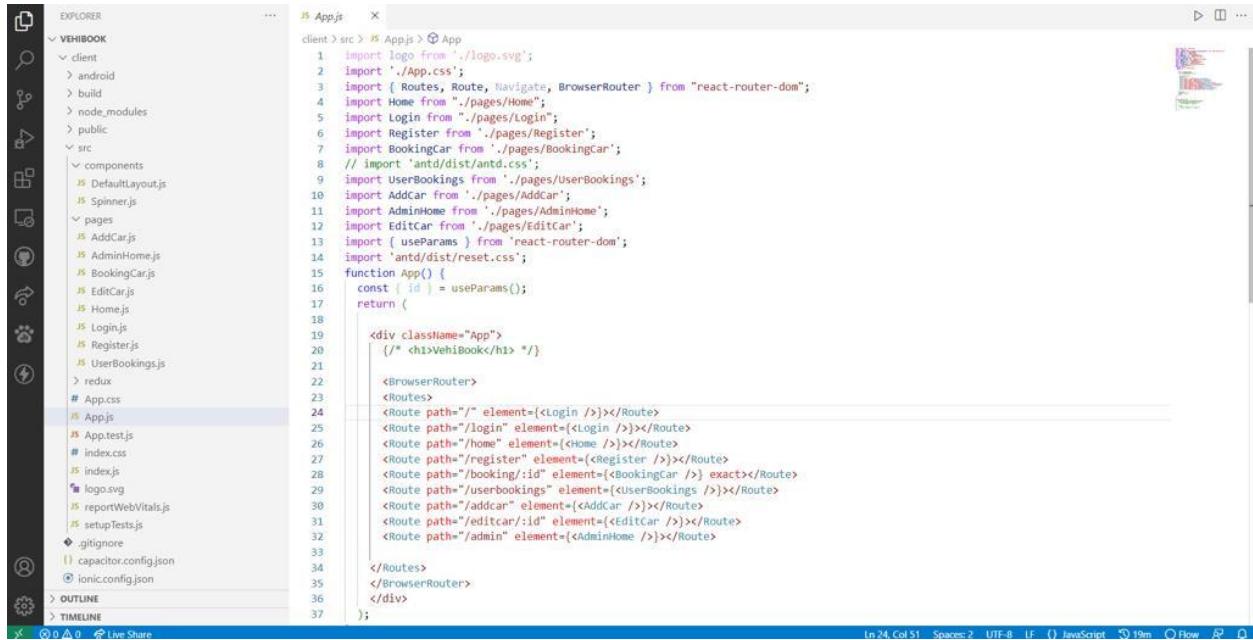
# CODE: WEB APPLICATION



The screenshot shows a code editor interface with the following details:

- EXPLORER:** On the left, the project structure is shown under the folder "VEHIBOOK". It includes a "client" folder containing "android", "build", "node\_modules", "public", and "src" (which further contains "components" and "pages"). The "src" folder also lists files like "DefaultLayout.js", "Spinner.js", "AddCar.js", "AdminHome.js", "BookingCar.js", "EditCar.js", "Home.js", "Login.js", "Register.js", and "UserBookings.js". Other files like "App.css", "App.js", "App.test.js", "index.css", "index.js", "logo.svg", "reportWebVitals.js", and "setupTests.js" are listed under the root and "redux" folder. There are also ".gitignore", "capacitor.config.json", and "ionic.config.json" files.
- CODE EDITOR:** The main area displays the content of the "index.js" file. The code imports React, ReactDOM, and CSS, and sets up a Provider for the Redux store. It then creates a root element and renders the App component within a Provider. A note at the bottom indicates how to start measuring performance.
- STATUS BAR:** At the bottom, the status bar shows "Ln 9, Col 13" and other developer tools like "Spaces: 2", "UTF-8", "LF", "JavaScript", "19m", "Flow", and "Live Share".

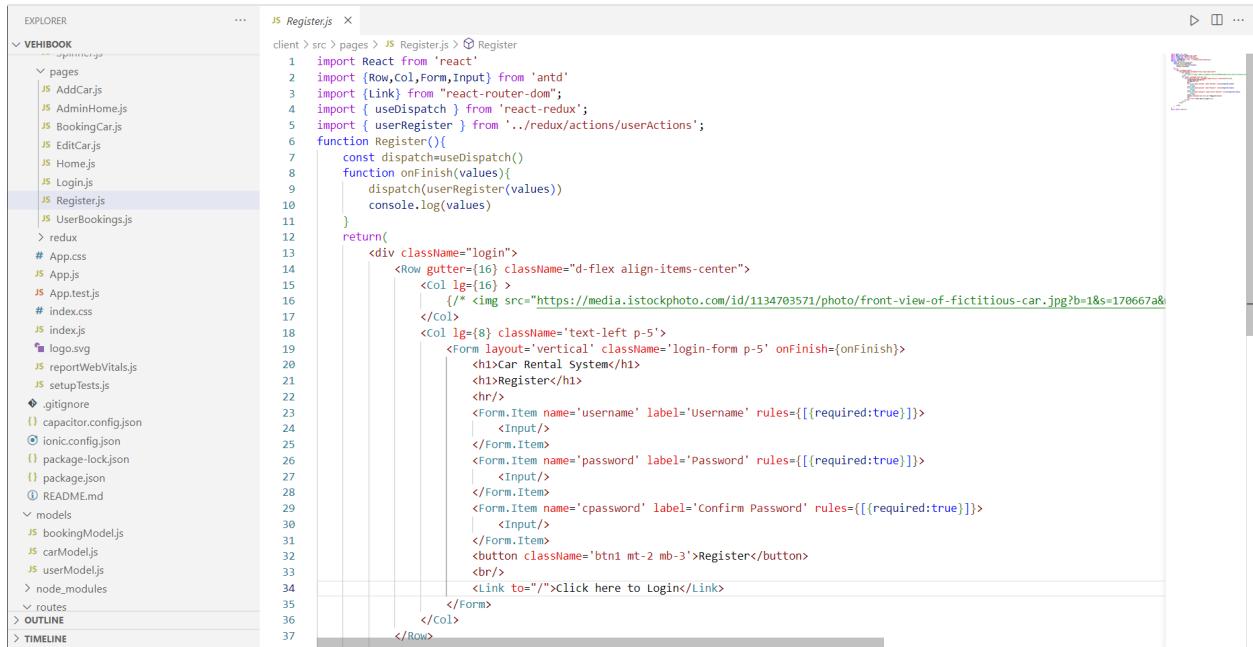
# FRONT END



The screenshot shows a code editor with the file `App.js` open. The code defines a `App` component that uses `react-router-dom` to handle different routes. It includes imports for `Logo`, `App.css`, and various pages like `Login`, `Home`, `BookingCar`, etc. The code is well-organized with proper indentation and comments.

```
client > src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3 import { Routes, Route, Navigate, BrowserRouter } from "react-router-dom";
4 import Home from "./pages/Home";
5 import Login from "./pages/Login";
6 import Register from "./pages/Register";
7 import BookingCar from "./pages/BookingCar";
8 // import 'antd/dist/antd.css';
9 import UserBookings from "./pages/UserBookings";
10 import AddCar from "./pages/AddCar";
11 import AdminHome from "./pages/AdminHome";
12 import EditCar from "./pages/EditCar";
13 import { useParams } from "react-router-dom";
14 import "antd/dist/reset.css";
15 function App() {
16   const [ id ] = useParams();
17   return (
18
19     <div className="App">
20       /* <h1>VehiBook</h1> */
21
22       <BrowserRouter>
23         <Routes>
24           <Route path="/" element={<Login />}></Route>
25           <Route path="/login" element={<Login />}></Route>
26           <Route path="/home" element={<Home />}></Route>
27           <Route path="/register" element={<Register />}></Route>
28           <Route path="/booking/:id" element={<BookingCar />} exact></Route>
29           <Route path="/userbookings" element={<UserBookings />}></Route>
30           <Route path="/addcar" element={<AddCar />}></Route>
31           <Route path="/editcar/:id" element={<EditCar />}></Route>
32           <Route path="/admin" element={<AdminHome />}></Route>
33
34         </Routes>
35       </BrowserRouter>
36     </div>
37   );
}
```

## Register



The screenshot shows a code editor with the file `Register.js` open. This file contains logic for user registration, using `react`, `antd`, and `react-router-dom`. It includes imports for `React`, `Row`, `Col`, `Form`, `Input`, and `Link`. The code defines a `Register` component that handles form submission and dispatches actions to the Redux store.

```
client > src > pages > JS Register.js > Register
1 import React from 'react'
2 import {Row, Col, Form, Input} from 'antd'
3 import {Link} from "react-router-dom";
4 import {useDispatch} from "react-redux";
5 import {userRegister} from '../redux/actions/userActions';
6 function Register(){
7   const dispatch=useDispatch()
8   function onFinish(values){
9     dispatch(userRegister(values))
10    console.log(values)
11  }
12  return(
13    <div className="login">
14      <Row gutter={16} className="d-flex align-items-center">
15        <Col lg={16}>
16          /*  */
17        </Col>
18        <Col lg={8} className='text-left p-5'>
19          <Form layout='vertical' className='login-form p-5' onFinish={onFinish}>
20            <h1>Car Rental System</h1>
21            <h1>Register</h1>
22            <br/>
23            <Form.Item name='username' label='Username' rules={[{required:true}]}>
24              <Input/>
25            </Form.Item>
26            <Form.Item name='password' label='Password' rules={[{required:true}]}>
27              <Input/>
28            </Form.Item>
29            <Form.Item name='password' label='Confirm Password' rules={[{required:true}]}>
30              <Input/>
31            </Form.Item>
32            <button className='btn1 mt-2 mb-3'>Register</button>
33            <br/>
34            <Link to="/">Click here to Login</Link>
35          </Form>
36        </Col>
37      </Row>

```

## Login

The screenshot shows a code editor interface with the file `Login.js` open. The left sidebar displays a project structure for a React application named "VEHIBOOK". The `pages` folder contains files like `AddCar.js`, `AdminHome.js`, `BookingCar.js`, `EditCar.js`, `Home.js`, `Login.js`, `Register.js`, and `UserBookings.js`. The `src` folder contains `App.css`, `App.js`, `App.test.js`, `index.css`, `index.js`, `logo.svg`, `reportWebVitals.js`, `setupTests.js`, and configuration files like `gitignore`, `capacitor.config.json`, `ionic.config.json`, `package-lock.json`, `package.json`, and `README.md`. The `models` folder includes `bookingModel.js`, `carModel.js`, and `userModel.js`. The `routes` folder is empty. The `OUTLINE` and `TIMELINE` sections are also visible.

```
client > src > pages > JS Login.js > Login
1 import React from 'react'
2 import { Row, Col, Form, Input } from 'antd'
3 import { Link } from 'react-router-dom';
4 import { useDispatch } from 'react-redux';
5 import { userLogin } from '../redux/actions/userActions';
6 const Login = () =>
7   const dispatch=useDispatch()
8   function onFinish(values){
9     dispatch(userLogin(values))
10    console.log(values)
11  }
12  return(
13    <div className="login">
14      <Row gutter={16} className="d-flex align-items-center">
15        <Col lg={16}>
16          /* 
17        </Col>
18        <Col lg={8} className="text-left p-5">
19          <Form layout="vertical" className="login-form p-5" onFinish={onFinish}>
20            <h1>Car Rental System</h1>
21            <h2>Login</h2>
22            <br/>
23            <Form.Item name="username" label="Username" rules={[{required:true}]}>
24              <Input/>
25            </Form.Item>
26            <Form.Item name="password" label="Password" rules={[{required:true}]}>
27              <Input/>
28            </Form.Item>
29            <button className="btn1 mt-2 mb-3">Login</button>
30            <br/>
31            <Link to="/register">Click here to Register</Link>
32          </Form>
33        </Col>
34      </Row>
35    </div>
36  )
37 )
```

## Admin Home

The screenshot shows a code editor interface with the file `AdminHome.js` open. The left sidebar displays a project structure for a React application named "VEHIBOOK". The `pages` folder contains files like `AddCar.js`, `AdminHome.js`, `BookingCar.js`, `EditCar.js`, `Home.js`, `Login.js`, `Register.js`, and `UserBookings.js`. The `src` folder contains `App.css`, `App.js`, `App.test.js`, `index.css`, `index.js`, `logo.svg`, `reportWebVitals.js`, `setupTests.js`, and configuration files like `gitignore`, `capacitor.config.json`, `ionic.config.json`, `package-lock.json`, `package.json`, and `README.md`. The `models` folder includes `bookingModel.js`, `carModel.js`, and `userModel.js`. The `routes` folder is empty. The `OUTLINE` and `TIMELINE` sections are also visible.

```
client > src > pages > JS AdminHome.js > AdminHome > totalCars.map() callback > <function> > card
1 import React, { useState, useEffect } from "react";
2 import { useSelector, useDispatch } from "react-redux";
3 import DefaultLayout from "../components/DefaultLayout";
4 import { deleteCar, getAllCars } from "../redux/actions/carsAction";
5 import { Col, Row, Divider, DatePicker, Checkbox, Edit } from "antd";
6 import { Link } from "react-router-dom";
7 import Spinner from "../components/spinner";
8 import moment from "moment";
9 import { DeleteOutlined, EditOutlined } from "@ant-design/icons";
10 import { Popconfirm, message } from "antd";
11 const { RangePicker } = DatePicker;
12 function AdminHome() {
13   const [ cars ] = useSelector((state) => state.carsReducer);
14   const [ loading ] = useSelector((state) => state.alertsReducer);
15   const [ totalCars, setTotalCars ] = useState([]);
16   const dispatch = useDispatch();
17   useEffect(() => {
18     dispatch(getAllCars());
19   }, []);
20   useEffect(() => {
21     setTotalCars(cars);
22   }, [cars]);
23   return (
24     <DefaultLayout>
25       <Row justify="center" gutter={16} className="mt-2">
26         <Col lg={20} sm={24}>
27           <div className="d-flex justify-content-between align-items-center">
28             <h3 className="mt-1 mr-2">Admin Panel</h3>
29             <button className="btn1">
30               <a href="/addcar">ADD CAR</a>
31             </button>
32           </div>
33         </Col>
34       </Row>
35     )
36   )
37 )
```

## Add Car

EXPLORER

VEHIBOOK

pages

- AddCar.js
- AdminHome.js
- BookingCar.js
- EditCar.js
- Home.js
- Login.js
- Register.js
- UserBookings.js

> redux

# App.css

App.js

App.test.js

# index.css

index.js

logo.svg

reportWebVitals.js

setupTests.js

.gitignore

capacitor.config.json

ionic.config.json

package-lock.json

package.json

README.md

models

- bookingModel.js
- carModel.js
- userModel.js

routes

OUTLINE

TIMELINE

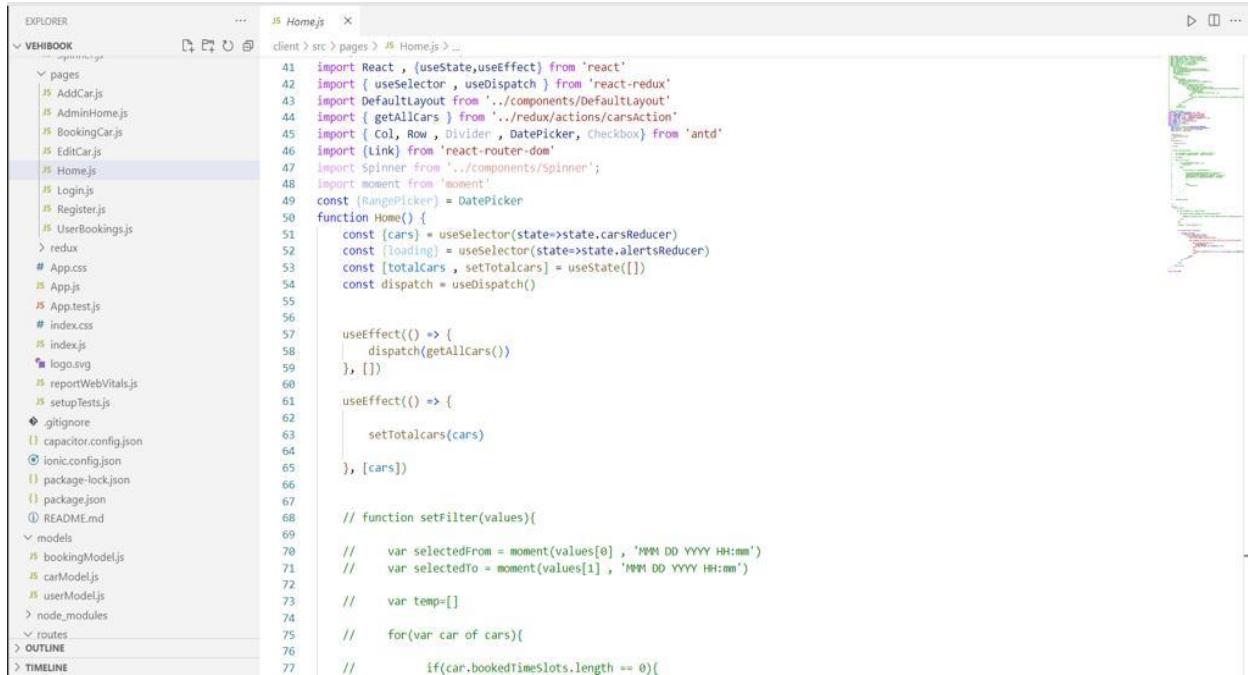
JS AddCar.js ×

```
client > src > pages > JS AddCar.js > AddCar > onFinish
1 import { Col , Row , Form , Input} from 'antd'
2 import React from 'react'
3 import { useDispatch , useSelector } from 'react-redux'
4 import DefaultLayout from '../components/DefaultLayout'
5 import Spinner from '../components/Spinner'
6 import { addCar } from '../redux/actions/carsAction'
7 function AddCar() {
8
9     const dispatch = useDispatch()
10    const {loading} = useSelector(state=>state.alertsReducer)
11
12    function onFinish(values){
13
14        values.bookedTimeSlots=[]
15
16        dispatch(addCar(values))
17        console.log(values)
18    }
19
20    return (
21        <DefaultLayout>
22            {loading && <Spinner />}
23            <Row justify='center mt-5'>
24                <Col lg={12} sm={24} xs={24} className='p-2'>
25                    <Form className='bs1 p-2' layout='vertical' onFinish={onFinish}>
26                        <h3>Add New Car</h3>
27                        <br />
28                        <Form.Item name='name' label='Car name' rules={[{required: true}]}>
29                            <Input/>
30                        </Form.Item>
31                        <Form.Item name='image' label='Image url' rules={[{required: true}]}>
32                            <Input/>
33                        </Form.Item>
34                        <Form.Item name='rentPerHour' label='Rent per hour' rules={[{required: true}]}>
35                            <Input/>
36                        </Form.Item>
37                        <Form.Item name='capacity' label='Capacity' rules={[{required: true}]}>
```

## Edit Car

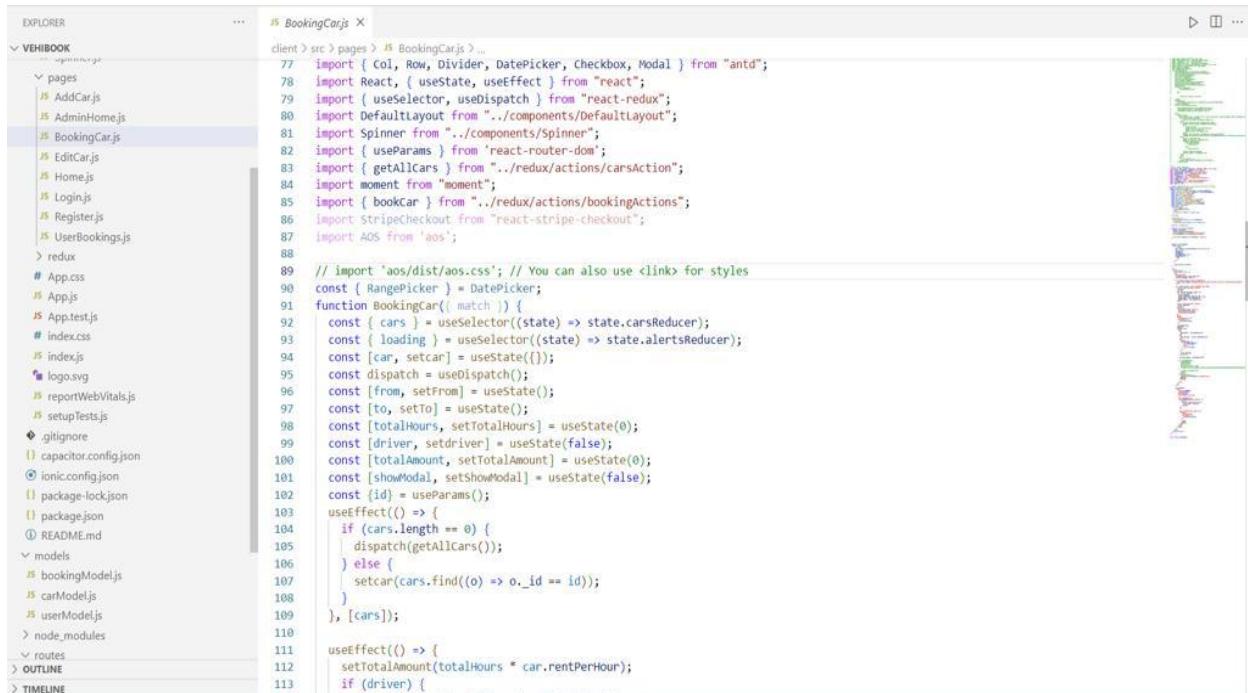
```
client > src > pages > EditCar.js > EditCar
1 import { Col, Row, Form, Input } from "antd";
2 import React, { useEffect, useState } from "react";
3 import { useDispatch, useSelector } from "react-redux";
4 import DefaultLayout from "../components/DefaultLayout";
5 import { useParams } from "react-router-dom";
6 import Spinner from "../components/Spinner";
7 import { addcar, editcar, getAllCars } from "../redux/actions/carsAction";
8 function EditCar({ match }) {
9   const { cars } = useSelector((state) => state.carsReducer);
10  const dispatch = useDispatch();
11  const { loading } = useSelector((state) => state.alertsReducer);
12  const [car, setcar] = useState();
13  const [totalcars, settotalcars] = useState([]);
14  const [id] = useParams();
15  useEffect(() => {
16    if (cars.length == 0) {
17      dispatch(getAllCars());
18    } else {
19      settotalcars(cars);
20      setcar(cars.find((o) => o._id == id));
21      console.log(car);
22    }
23  }, [cars]);
24
25  function onFinish(values) {
26    values._id = car._id;
27
28    dispatch(editcar(values));
29    console.log(values);
30  }
31
32  return (
33    <DefaultLayout>
34      {loading && <Spinner />}
35      <Row justify="center mt-5">
36        <Col lg={12} sm={24} xs={24} className='p-2'>
37          {totalcars.length > 0 && (
```

## User Home



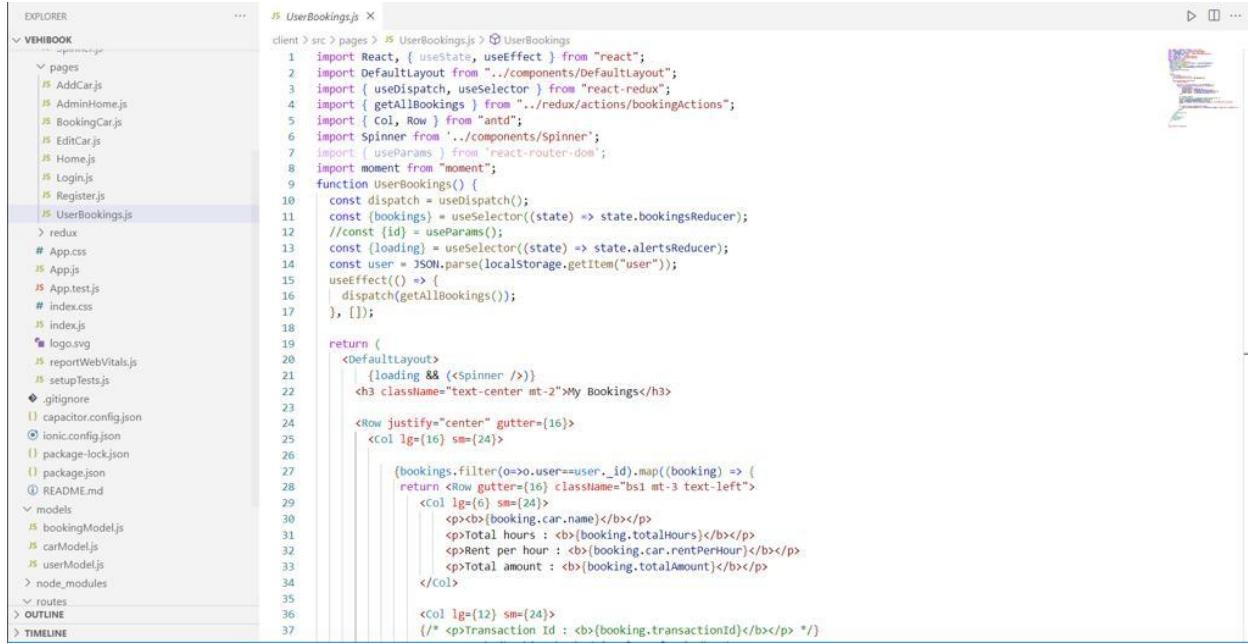
```
client > src > pages > JS Home.js > ...  
41 import React, {useState,useEffect} from 'react'  
42 import {useSelector, useDispatch} from 'react-redux'  
43 import DefaultLayout from './components/DefaultLayout'  
44 import { getAllCars } from '../redux/actions/carsAction'  
45 import { Col, Row, Divider, DatePicker, Checkbox } from 'antd'  
46 import { Link } from 'react-router-dom'  
47 import Spinner from './components/Spinner';  
48 import moment from 'moment'  
49 const RangePicker = DatePicker  
function Home() {  
  const [cars] = useSelector(state=>state.carsReducer)  
  const [loading] = useSelector(state=>state.alertsReducer)  
  const [totalCars, setTotalCars] = useState([])  
  const dispatch = useDispatch()  
  useEffect(() => {  
    dispatch(getAllCars())  
  }, [])  
  useEffect(() => {  
    setTotalCars(cars)  
  }, [cars])  
  
  // function setFilter(values){  
  //   var selectedFrom = moment(values[0], 'MM DD YYYY HH:mm')  
  //   var selectedTo = moment(values[1], 'MM DD YYYY HH:mm')  
  //   var temp=[]  
  //   for(var car of cars){  
  //     if(car.bookedTimeSlots.length == 0){  
  //       temp.push(car)  
  //     }  
  //   }  
  //   setTotalCars(temp)  
  // }  
}  
export default Home
```

## Booking



```
client > src > pages > JS BookingCar.js > ...  
77 import { Col, Row, Divider, DatePicker, Checkbox, Modal } from "antd";  
78 import React, { useState, useEffect } from "react";  
79 import { useSelector, useDispatch } from "react-redux";  
80 import DefaultLayout from "./components/DefaultLayout";  
81 import Spinner from "./components/Spinner";  
82 import { useParams } from "react-router-dom";  
83 import { getAllCars } from "../redux/actions/carsAction";  
84 import moment from "moment";  
85 import { bookCar } from "../redux/actions/bookingActions";  
86 import StripeCheckout from "react-stripe-checkout";  
87 import AOS from "aos";  
88 // import 'aos/dist/aos.css'; // You can also use <link> for styles  
89 const RangePicker = DatePicker;  
90 function BookingCar({ match }) {  
  const [cars] = useSelector((state) => state.carsReducer);  
  const [loading] = useSelector((state) => state.alertsReducer);  
  const [car, setCar] = useState();  
  const dispatch = useDispatch();  
  const [from, setFrom] = useState();  
  const [to, setTo] = useState();  
  const [totalHours, setTotalHours] = useState(0);  
  const [driver, setDriver] = useState(false);  
  const [totalAmount, setTotalAmount] = useState(0);  
  const [showModal, setShowModal] = useState(false);  
  const [id] = useParams();  
  useEffect(() => {  
    if (cars.length == 0) {  
      dispatch(getAllCars());  
    } else {  
      setCar(cars.find((o) => o._id == id));  
    }  
  }, [cars]);  
  
  useEffect(() => {  
    setTotalAmount(totalHours * car.rentPerHour);  
    if (driver) {  
      setShowModal(true);  
    }  
  }, [totalHours, driver, showModal]);  
  return (  
    <Formik  
      initialValues={{  
        from:  
      }}  
      validationSchema={...}  
      onSubmit={...}  
    >
```

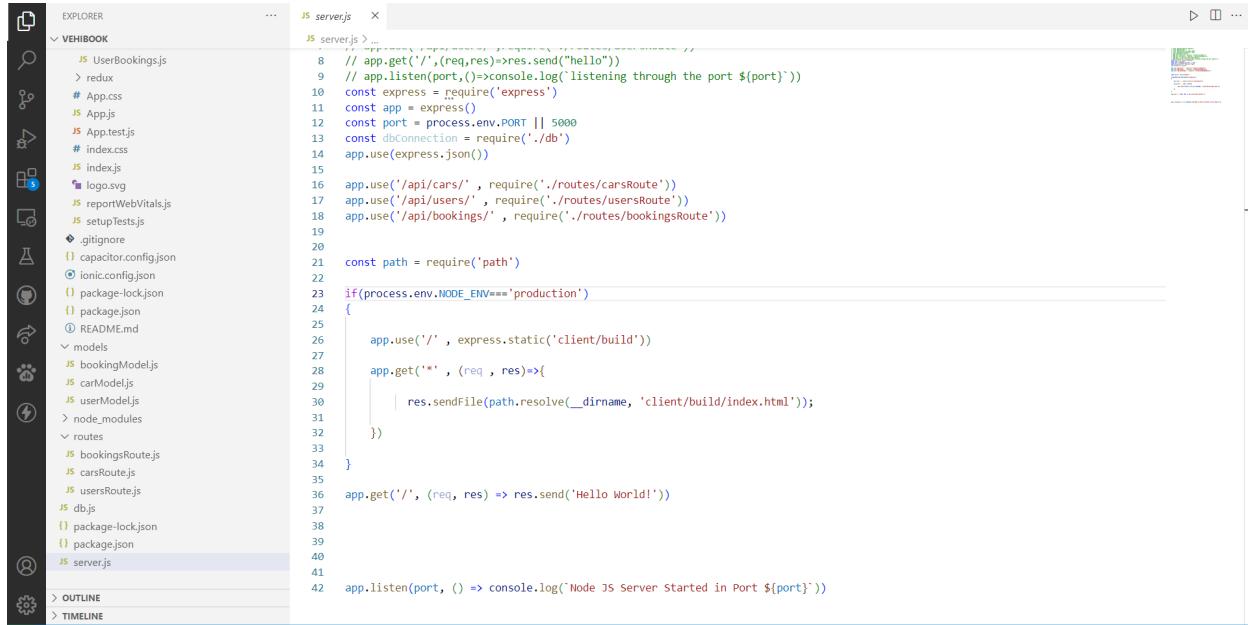
## User Booking



The screenshot shows the UserBookings.js file in a code editor. The file contains React code for displaying user bookings. It imports React, useState, useEffect, DefaultLayout, useDispatch, useSelector, moment, and spinner components. It also imports getAllBookings from redux/actions/bookingActions. The code uses useState and useSelector to manage state, and useEffect to dispatch actions. The component displays a loading spinner and a list of bookings for the current user.

```
client > src > pages > JS UserBookings.js > UserBookings
1 import React, { useState, useEffect } from "react";
2 import DefaultLayout from "../components/DefaultLayout";
3 import { useDispatch, useSelector } from "react-redux";
4 import { getAllBookings } from "../redux/actions/bookingActions";
5 import { Col, Row } from "antd";
6 import Spinner from "../components/Spinner";
7 import { useParams } from "react-router-dom";
8 import moment from "moment";
9 function UserBookings() {
10   const dispatch = useDispatch();
11   const [bookings] = useState([]);
12   const [loading] = useSelector((state) => state.alertsReducer);
13   const user = JSON.parse(localStorage.getItem("user"));
14   useEffect(() => {
15     dispatch(getAllBookings());
16   }, []);
17
18   return (
19     <DefaultLayout>
20       {loading && <Spinner />}
21       <h3 className="text-center mt-2">My Bookings</h3>
22
23       <Row justify="center" gutter={16}>
24         <Col lg={16} sm={24}>
25           {bookings.filter(o=>o.user==user._id).map((booking) => (
26             return <Row gutter={16} className="bs1 mt-3 text-left">
27               <Col lg={6} sm={24}>
28                 <p><b>{booking.car.name}</b></p>
29                 <p>Total hours : <b>{booking.totalHours}</b></p>
30                 <p>Rent per hour : <b>{booking.car.rentPerHour}</b></p>
31                 <p>Total amount : <b>{booking.totalAmount}</b></p>
32               </Col>
33             </Row>
34           ))
35         </Col>
36       </Row>
37       /* <p>Transaction Id : <b>{booking.transactionId}</b></p> */
38     </DefaultLayout>
39   );
40 }
```

## BACK END



The screenshot shows the server.js file in a code editor. The file sets up a Node.js application using Express. It defines routes for cars, users, and bookings. It uses static files for the client build. The app listens on port 5000 and logs a message upon startup.

```
client > src > server.js > server
1 // app.get('/',(req,res)=>res.send("Hello"))
2 // app.listen(port,()=>console.log(`listening through the port ${port}`))
3 const express = require('express')
4 const app = express()
5 const port = process.env.PORT || 5000
6 const dbConnection = require('./db')
7 app.use(express.json())
8
9 app.use('/api/cars/' , require('./routes/carsRoute'))
10 app.use('/api/users/' , require('./routes/usersRoute'))
11 app.use('/api/bookings/' , require('./routes/bookingsRoute'))
12
13 const path = require('path')
14
15 if(process.env.NODE_ENV==='production')
16 {
17   app.use('/*' , express.static('client/build'))
18
19   app.get('*', (req , res)=>{
20     | res.sendFile(path.resolve(__dirname, 'client/build/index.html'));
21   })
22
23   app.get('/', (req , res) => res.send('Hello World!'))
24
25   app.listen(port, () => console.log(`Node JS Server Started in Port ${port}`))
26 }
```

# Schema

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure for a Node.js application named 'VEHIBOOK'. The 'models' folder contains three files: bookingModel.js, carModel.js, and userModel.js. The userModel.js file is currently selected and shown in the main editor area. The code defines a mongoose schema for users:

```
JS userModel.js x
models > JS userModel.js > userSchema > username
1 const mongoose=require("mongoose")
2 const userSchema=new mongoose.Schema({
3   username:{type:String,required:true},
4   password:{type:String,required:true}
5 })
6
7 const userModel=mongoose.model('users',userSchema)
8 module.exports=userModel
```

The status bar at the bottom indicates the code has 3 lines, 13 columns, and 8 selected characters, and is saved in JavaScript with UTF-8 encoding.

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure for a Node.js application named 'VEHIBOOK'. The 'models' folder contains three files: bookingModel.js, carModel.js, and userModel.js. The carModel.js file is currently selected and shown in the main editor area. The code defines a mongoose schema for cars:

```
JS carModel.js x
models > JS carModel.js > carModel
1 const mongoose=require("mongoose")
2 const carSchema=new mongoose.Schema({
3   name:{type:String,required:true},
4   image:{type:String,required:true},
5   capacity:{type:Number,required:true},
6   fuelType:{type:String,required:true},
7   bookedTimeSlots:[
8     {
9       from:{type:String,required:true},
10      to:{type:String,required:true}
11    }
12  ],
13  rentPerHour:{type:Number,required:true}
14 },{timestamps:true})
15
16 const carModel=mongoose.model('cars',carSchema)
17 module.exports=carModel
```

The status bar at the bottom indicates the code has 17 lines, 36 columns, and 4 selected characters, and is saved in JavaScript with UTF-8 encoding.

The screenshot shows the VS Code interface with the 'models' folder selected in the Explorer sidebar. The 'bookingModel.js' file is open in the main editor area. The code defines a mongoose schema for bookings:

```

1 const mongoose = require("mongoose");
2
3 const bookingsSchema = new mongoose.Schema({
4
5   car : {type : mongoose.Schema.Types.ObjectId , ref:'cars'},
6   user : {type : mongoose.Schema.Types.ObjectId , ref:'users'},
7   bookedTimeSlots : [
8     {
9       from : {type : String} ,
10      to : {type : String}
11    },
12    totalHours : {type : Number},
13    totalAmount : {type : Number},
14    // transactionId : {type : String},
15    driverRequired : {type : Boolean}
16
17  },
18  {timestamps : true}
19
20
21
22 const bookingModel = mongoose.model('bookings' , bookingsSchema)
23
24 module.exports = bookingModel

```

The status bar at the bottom indicates: In 24, Col 30 (12 selected) Spaces:2 UTF-8 CRLF (JavaScript) 19m ⚡ Flow R ⚡

## Routes

The screenshot shows the VS Code interface with the 'routes' folder selected in the Explorer sidebar. The 'usersRoute.js' file is open in the main editor area. The code handles a POST request for registration:

```

10  }
11  } catch (error) {
12   return res.status(400).json(error);
13 }
14
15 });
16 router.post("/register",body('password').isLength({min:3}), async(req, res) => {
17   const errors=validationResult(req)
18   if(!errors.isEmpty()){
19     return res.status(400).json("number of characters used for password should be greater");
20   }
21   const {username , password} = req.body
22   try {
23     const user = await User.findOne({username , password})
24     if(user) {
25       return res.status(400).json("user already exists");
26     }
27     const newuser = new User(req.body)
28     await newuser.save()
29     res.send('User registered successfully')
30   } catch (error) {
31     return res.status(400).json(error);
32   }
33 }
34
35 module.exports = router
36
37
38
39
40
41 });
42
43
44
45
46

```

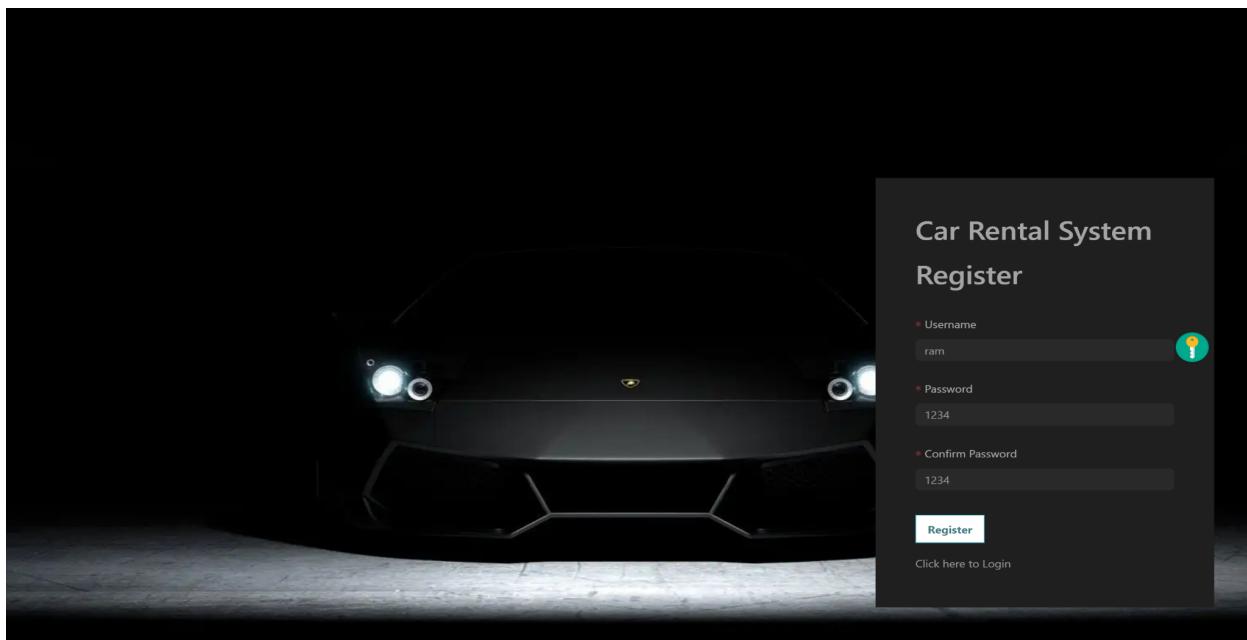
The status bar at the bottom indicates: In 24, Col 30 (12 selected) Spaces:2 UTF-8 CRLF (JavaScript) 19m ⚡ Flow R ⚡

```
routes > JS carsRoute.js > router.get("/getcar") callback
1 const express = require("express");
2 const router = express.Router();
3 const Car = require("../models/carModel");
4 router.get("/getallcars", async (req, res) => {
5   try {
6     const cars = await Car.find();
7     res.send(cars);
8   } catch (error) {
9     return res.status(400).json(error);
10  }
11 });
12 router.get("/getcar", async (req, res) => {           //not needed
13   try {
14     const cc = await Car.find({ _id: req.body._id });
15     if(cc.length==0){
16       return res.status(400).json("no car exists with the ID");
17     }
18     res.send(cc);
19   } catch (error) {
20     return res.status(400).json(error);
21   }
22 });
23 router.post("/addcar", async (req, res) => {
24   try {
25     const newcar = new Car(req.body);
26     await newcar.save();
27     res.send("Car added successfully");
28   } catch (error) {
29     return res.status(400).json(error);
30   }
31 });
32
33 router.patch("/editcar", async (req, res) => {      //post
34   try {
35     const car = await Car.findOne({ _id: req.body._id });
36     car.name = req.body.name;
37     car.image = req.body.image;

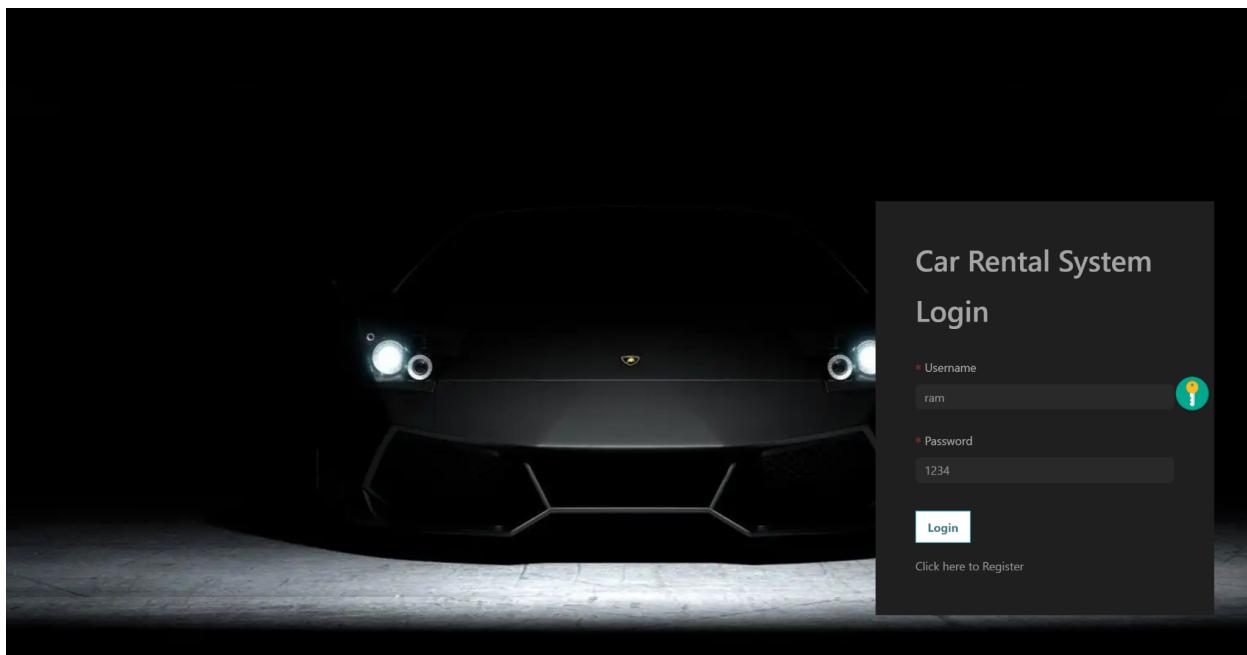
```

# IMAGES: WEB APPLICATION

## Register



## Login



## Admin Panel

Car Rental System

Admin Panel

[ADD CAR](#)

Mahindra XUV700  
Rent Per Hour 800/-

Designed and Developed by  
Team 7

## Edit Car

Car Rental System

[Edit CAR](#)

**Edit Car**

Car name: Mahindra XUV700

Image url: <https://stimg.carcdekho.com/images/carexteriorimages/630x420/Mahindra/XUV700/8620/1633939302512/front->

Rent per hour: 800

Capacity: 5

Fuel Type: petrol

Designed and Developed by  
Team 7

## User Panel

Car Rental System

[Book Now](#)

Designed and Developed by  
Team 7

# Booking

Car Rental System

ram



Car Info

Huundai i20  
500 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

Select Time Slots

Start date  End date

[See Booked Slots](#)

Designed and Developed by  
Team 7

Car Rental System

ram



Car Info

Huundai i20  
500 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

Select Time Slots

Mar 30 2023 06:25  Mar 31 2023 06:25

[See Booked Slots](#)

Total Hours : 24  
Rent Per Hour : 500  
 Driver Required

**Total Amount : 12000**

[Book Now](#)

Designed and Developed by  
Team 7

Car Rental System

ram



Car Info

Huundai i20  
500 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

Select Time Slots

Mar 21 2023 06:25  Mar 22 2023 06:25

[See Booked Slots](#)

Total Hours : 24  
Rent Per Hour : 500  
 Driver Required

**Total Amount : 12720**

[Book Now](#)

Designed and Developed by  
Team 7

## User Bookings

Car Rental System

ram

### My Bookings

<b>Huindai i20</b>	From: Mar 21 2023 06:25	To: Mar 22 2023 06:25
Total hours : 24	Rent per hour : 500	Date of booking: Mar 31 2023
Total amount : 12720		

Designed and Developed by  
Team 7

## Booked Time slots

Car Rental System

ram









Tata Nexon  
Rent Per Hour 700 /-

[Book Now](#)





Designed and Developed by  
Team 7

Car Rental System

ram



**Car Info**

Tata Nexon  
700 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

**Select Time Slots**

Mar 23 2023 06:27 → Mar 24 2023 06:27

[See Booked Slots](#)

Total Hours : 24  
Rent Per Hour : 700  
 Driver Required

**Total Amount : 17520**

[Book Now](#)

Designed and Developed by  
Team 7

# Car Rental System

ram

## My Bookings

Tata Nexon  
Total hours : **24**  
Rent per hour : **700**  
Total amount : **17520**

From: **Mar 23 2023 06:27**  
To: **Mar 24 2023 06:27**  
Date of booking: **Mar 31 2023**



Designed and Developed by  
Team 7

# Car Rental System

ram

### Booked time slots

Mar 30 2023 01:21 - Mar 31 2023 01:21  
Mar 23 2023 06:27 - Mar 24 2023 06:27

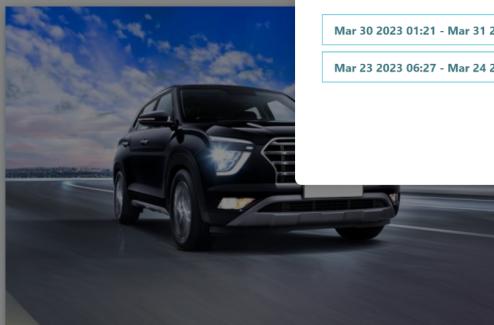
CLOSE

### Car Info

Tata Nexon  
700 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

### Select Time Slots

Start date  End date   
[See Booked Slots](#)



Designed and Developed by  
Team 7

# Database

## Cars

The screenshot shows the MongoDB Compass interface with the database 'atlascluster.p9a...' selected. The left sidebar lists databases: admin, config, local, vehi (which is expanded to show bookings, cars, users), and vrs. The main area displays the 'vehi.cars' collection with 4 documents and 1 index. A document is selected, showing fields like \_id, name, image, rentPerHour, fuelType, bookedTimeSlots, capacity, and updatedAt.

```
_id: ObjectId("6423a8886a3fffa8a6f3f14b")
name: "MG Astor"
image: "https://stimg.cardekho.com/images/carexteriorimages/630x420/MG/Astor/7_"
rentPerHour: 500
fuelType: "petrol"
bookedTimeSlots: []
capacity: 5
__v: 1

_id: ObjectId("6423a8886a3fffa8a6f3f14a")
name: "Hyundai i20"
image: "https://stimg.cardekho.com/images/carexteriorimages/630x420/Hyundai/i2_"
rentPerHour: 500
fuelType: "petrol"
bookedTimeSlots: []
capacity: 5
__v: 3
updatedAt: 2023-03-31T01:04:46.781+00:00

_id: ObjectId("6423a8886a3fffa8a6f3f147")
name: "Tata Altroz"
image: "https://stimg.cardekho.com/images/carexteriorimages/630x420/Tata/Altroz/7_"


```

## Bookings

The screenshot shows the MongoDB Compass interface with the database 'atlascluster.p9a...' selected. The left sidebar lists databases: admin, config, local, vehi (expanded to bookings, cars, users), and vrs. The main area displays the 'vehi.bookings' collection with 2 documents and 1 index. A document is selected, showing fields like \_id, car, user, bookedTimeSlots, totalHours, and createdAt.

```
_id: ObjectId('6426301ca8627e56a8a38fa')
car: ObjectId("6423a8886a3fffa8a6f3f149")
user: ObjectId("64262f2ccaa8627e56a8a38da")
bookedTimeSlots: Object
totalHours: 24
totalAmount: 17520
doNotRevert: true
createdAt: 2023-03-31T00:57:58.453+00:00
updatedAt: 2023-03-31T00:57:58.453+00:00
__v: 0

_id: ObjectId("642631eca8627e56a8a3172")
car: ObjectId("6423a8886a3fffa8a6f3f14a")
user: ObjectId("6425ed7809fb42d42d703cf47e")
bookedTimeSlots: Object
totalHours: 24
totalAmount: 12000
doNotRevert: false
createdAt: 2023-03-31T01:04:46.671+00:00
updatedAt: 2023-03-31T01:04:46.671+00:00
__v: 0


```

## Users

The screenshot shows the MongoDB Compass interface with the database 'atlascluster.p9a...' selected. The left sidebar lists databases: admin, config, local, vehi (expanded to bookings, cars, users), and vrs. The main area displays the 'vehi.users' collection with 10 documents and 1 index. A document is selected, showing fields like \_id, username, password, and \_\_v.

```
_id: ObjectId("640a22c84460caa7133678de")
username: "ajay"
password: "1234"
__v: 0

_id: ObjectId("640a2402585f9512b9af9e13")
username: "ajaydeepak"
password: "12345"
__v: 0

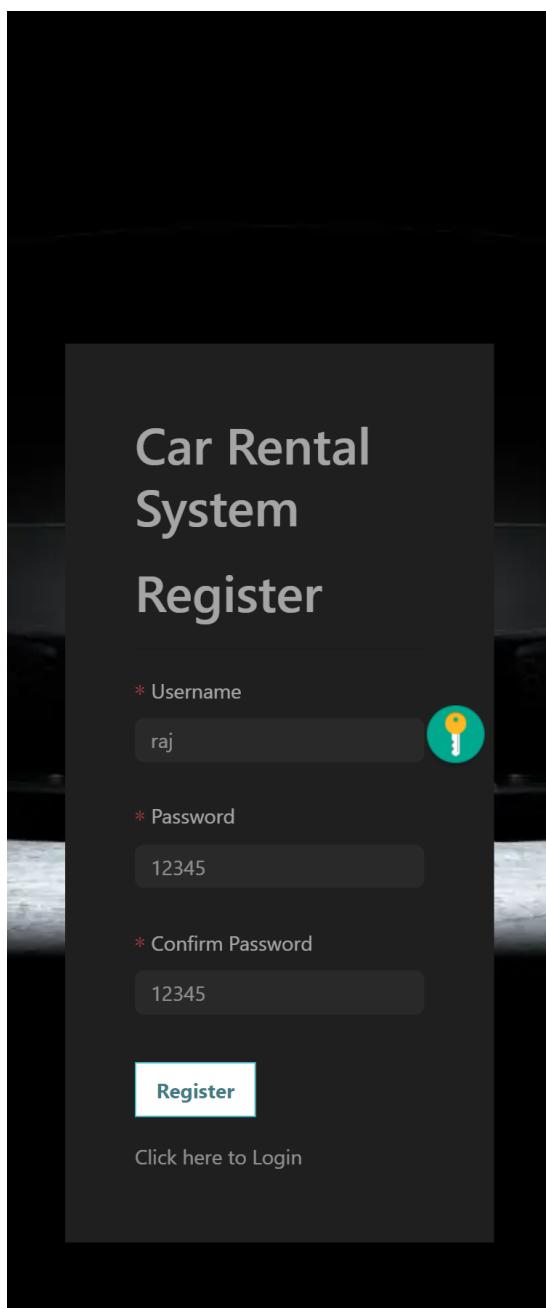
_id: ObjectId("6423a00324220c60c86d3625")
username: "hhhhh"
password: "123456666"
__v: 0

_id: ObjectId("6423a0f312b30863759f21f2f")
username: "hhhhh"
password: "1"
__v: 0

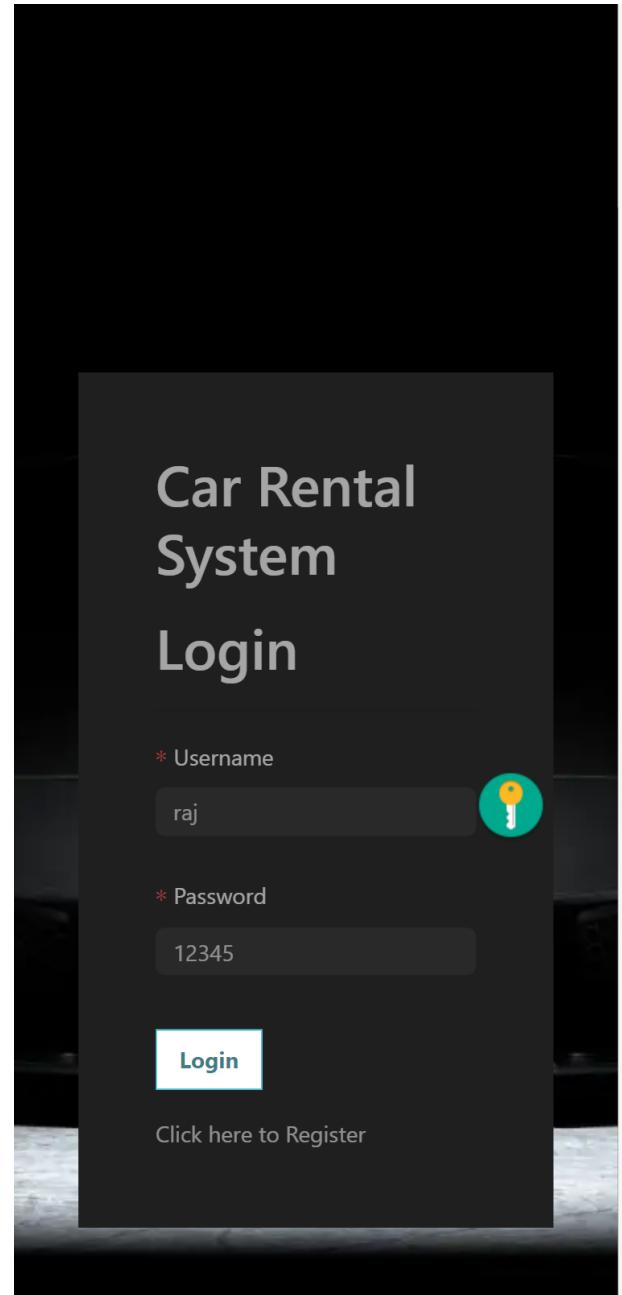

```

## IMAGES: MOBILE APPLICATION

Register



Login



## User Panel

### Car Rental System

raj



Designed and Developed by  
Team 7

### Car Rental System

raj



MG Astor  
Rent Per Hour 500/-

Book Now



Designed and Developed by  
Team 7

## Booking

### Car Rental System

raj



#### Car Info

MG Astor  
500 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

#### Select Time Slots

Mar 23 2023 06:42  Mar 25 2023 06:42

[See Booked Slots](#)

Total Hours : **48**  
Rent Per Hour : **500**  
 Driver Required

**Total Amount : 24000**

[Book Now](#)

Designed and Developed by  
Team 7

## My Bookings

### Car Rental System

raj

#### My Bookings

**MG Astor** From: **Mar 23 2023 06:42**  
Total hours : **48** To: **Mar 25 2023 06:42**  
Rent per hour : **500** Date of booking: **Mar 31 2023**  
Total amount : **24000**



Designed and Developed by  
Team 7

## Booked Time Slots

Car Rental System raj

Booked time slots

Mar 23 2023 06:42 - Mar 25 2023  
06:42

CLOSE

Car Info

MG Astor  
500 Rent Per hour /-  
Fuel Type : petrol  
Max Persons : 5

Select Time Slots

Start date  End date

See Booked Slots

Designed and Developed by  
Team 7