Sri Lanka Institute of Information Technology

Travel Agency and Tour Planning

Project Report

Information Technology Project 2021

Project ID: ITP2021_S2_MTR_G05

Submitted by:

1. IT20172350– (M.P.O.M Gomes)
2. IT20255510– (Y.N Jayasekara)
3. IT20186906– (Tharuka Gayashan F.)
4. IT20046552– (Abeywardahange S.R.D)
5. IT20204648– (Nethu Nipuna M.)
6. IT20163204– (Manimendra N. H.)
7. IT20142964– (Dulakshi Hansini R.M)
8. IT20142728– (Ranawaka T.D)

Submitted to:

Chamari silva

Date of submission

Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	Travel Agency and Tour Planning
Project ID	ITP2021_S2_MTR_G05

Group Members

Reg. No	Name	Signature
IT20172350	M.P.O.M Gomes	*oshen*
IT20255510	Y.N Jayasekara	*yasiruJ*
IT20186906	Tharuka Gayashan F	*Gayshan*
IT20046552	Abeywardhanage S.R. D	*sachini*
IT20204648	Nethu Nipuna M	*nipuna*
IT20163204	Manimendra N. H.	*himanka*
IT20142964	Dulakshi Hansini R.M	*dulakshi*
IT20142728	Ranawaka T. D	*thenushka*

Abstract

Our client Uniwest wanted to create an attractive website for their travel Agency to make things easier to use for customers. Uniwest presented their needs to us, and our team created this website accordingly.

Therefore, thorough this website project is implemented under eight functions that cover the complete scope of the company. Those functions are accommodation reservation, ticket reservation, ground resource reservation, tour package reservation, travel document management, insurance reservation, event reservation. This website overcomes issues of a manual system such as less security, long access time, and difficulty to make changes.

To implement this project, we use the MERN stack. In the MERN stack, Mongo DB, React js, Express and Node js were used.

Acknowledgement

First of all, we extend our heartfelt gratitude to everyone those who supported and guided us in making our project a success one.

Especially, we need to give a big thanks and our uncommon much gratitude goes the instructor responsible for the Information Technology Project (ITP), the Ms. Chamari de silva for giving her significant time to explain our disarray and calling attention to our shortcomings. Also, we have to give our appreciate to her guidance to success our project.

We would also like to thank for our senior student Mr. Nimesh Madusanka who helps us by giving some instructions to solve our problems that emerged throughout the project and supported us to go ahead.

Also, we would like to give our kind express to Mr. Kasun Indika Kapugmage the owner of Uniwest travel agency for giving us such attention and choosing us as their developers to create and deploy the automated system.

We want to show thanks to our team members, who have actively supported and emancipated us in making and accomplishing such a great project.

Finally, we would not forget to remember and thank the staff of faculty of computing for guiding us step by step. And for giving this golden opportunity to us.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement

Client name is Uniwest travel agency which located in Nupe, Matara. It all started about ten years ago. This company which started by providing only the service of Student Recruitment for student visa. Now they are planning to provide all the services as a travel agency with that Student Recruitment for student visa. Therefore, they now entered into a complex business process. Therefore, they felt the need for a specific website as they wanted to provide a clear image for their organization to the customer while providing services as a more professional and reliable travel agency. In addition, they wanted to maintain a systematic database. So, they contacted us and explained their need. There were several requirements they put forward.

1. Creating a website that ensures customer attraction and trust.
2. Maintain a systematic database.
3. Give Various of categories for the customer to select for their accommodation reservations.
4. Issuing tickets for Airplanes, trains, buses, and events.
5. Facilitate customers to plan and reserve their tour packages.
6. To give the customer a complete understanding of the services provided by their company and their prices.
7. Facilitate customer to reserve ground resources for their needed.
8. Maintain travel documents of customers and give essential details about travel documents.
9. Issuing travel insurance for their travels with trusted insurance companies.
10. Facilitate customers to plan and reserve their events.

Our team worked tirelessly to meet these needs. As a result, Uniwest now has a well-maintained website and a dedicated database. Through this website, many of their basic needs were satisfied and they were able to save a lot of unnecessary expenses. This will enable Uniwest owner to promote their company while providing a more efficient and reliable service to their customers.

1.2 Product Scope

To ease the process of achieving the above-mentioned goals, the System for Uniwest travel agency was divided into eight functions.

1. User Account management
2. Accommodation reservation Management
3. Reservation of Tickets Management
4. Reservation of ground Management
5. Travel Document and Travel information Management
6. Tour Package Management
7. Travel Insurance Management
8. Event Management

1.2.1 User Account Management

This function provides hotel users to create their user profiles to get the service online. Users can log in using provided username and password, logout, change profile details anytime, and also they can delete their account from the system.
When the Employee login the system automatically redirects to the relevant user interfaces. Once a user account is created users can see the hotel dynamic web page**.**

1.2.2 Accommodation reservation Management

From this function manage accommodation reservations. Maintaining accommodations, prices and display to customer available accommodations. provide filters to advance search. Provide environment to book accommodations and payments. Manage the role of accommodation reservation manager (adding new accommodations to data base, update information, and manage reservation database)

1.2.3 Reservation of Tickets Management

In this function customer can select types of tickets easily. Bus tickets, Railway, airline tickets and events ticket available on. Customer can select each destination, starting points, fixed date they want. They can easily pay by Master and visa card to book their tickets. All the data save in our database as types of Ticket tables and Reservation table. Admin can add, Delete, and modify tickets and get a report about list of people who have booked tickets on daily basis

1.2.4 Reservation of ground Management

This function allows customers to book taxis and cars. In addition to booking a guide to make their destination easier. Also, the system admin can add new vehicles, guides, and judges to the system. And to change and remove their details. In addition, the system allows the admin to retrieve data as a report if desired.

1.2.5 Travel Document and Travel information Management

This function handling enquiries of travel destination details. Customer can know any doubts about travel information with this. And, customer want to apply a visa, they can send their travel document and set their visa from via through with our agency. That will make work easier for travelers.

As an admin side on this function, staff can communicate with their customer if they have any questions about travel information. And when customer send their travel document, staff can check their documents and process it to apply the visa.

1.2.6 Tour Package Management

This function allows to customer select and reserve a tour package at the discretion of the customer. As well customer can cancel his/her reservation with fill cancelation form.
From the admin side staff member can manage packages. Staff Member can add, update, and delete specific packages according to his/her opinion. And allows getting

1.2.7 Travel Insurance Management

This function allows the customer to choose the type of travel insurance they want. The types of travel insurance selected by the customer are stored in the travel insurance database. Also, on the admin side the administrator can manage the travel insurance type in travel insurance type. The administrator can add new travel insurance types, edit and delete existing ones. The administrator can also manage the travel insurance database and the administrator can obtain a report of the stored details if required

1.2.8 Event Management

This function allows to customers to be aware of entertaining events which including in their tour packages. Then customer can choose a tour package that including proper events. Also, the staff members and guides can be aware of their meetings and conferences.

Event manager can schedule meetings and conferences and he can add new events to the packages. Also, event manager updates the event and meeting schedules.

1.3 Project Report Structure

First Section:

As the first section it mainly focused on client's problems which they have while using manual system and client's requirements. Therefore, in this part gave the solutions for that problem and include the overall project plan and the scope of the project by us.

Second Section:

In this section focused on methodology part which contain the clarification of Requirement Analysis, Designs, Implementation, and the Testing. Use case Diagrams, Activity Diagrams and sequence diagrams are used for describing Requirement Analysis part. The Design part is classified utilizing the guide of the ER diagram and the snapshots of user Interfaces. For the Implementation part there are Snapshots of the Implementations and finally there are test cases for the test areas.

Third Section:

This section indicates the information of evaluation and accomplishment of the whole project and the new knowledge which are learned through this project.

Forth Section:

This section indicates all the references which used to full fill this project

2. Methodology

2.1 Requirements and Analysis

2.1.1 User Account management

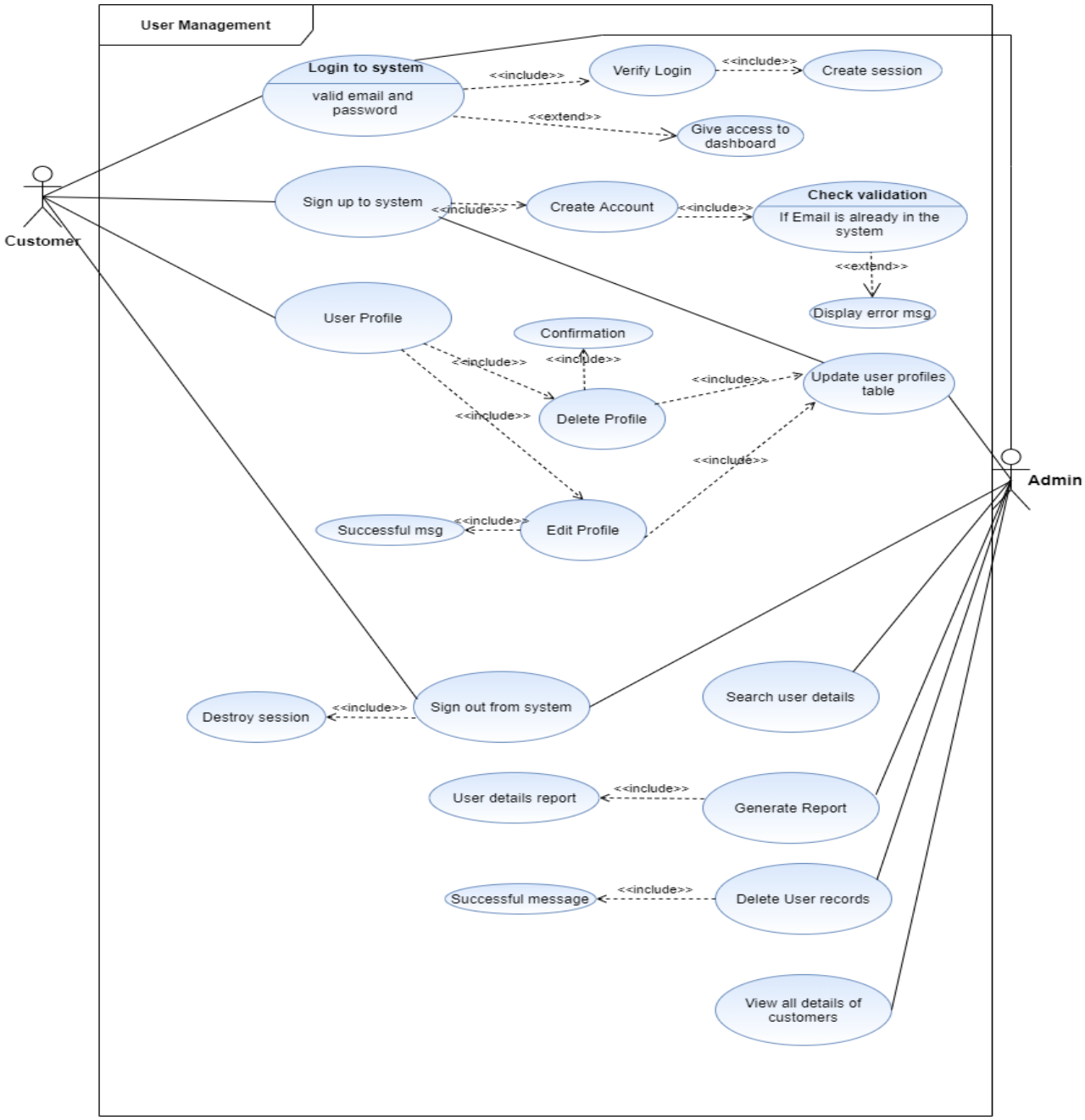


Figure 1

2.1.2 Accommodation reservation Management

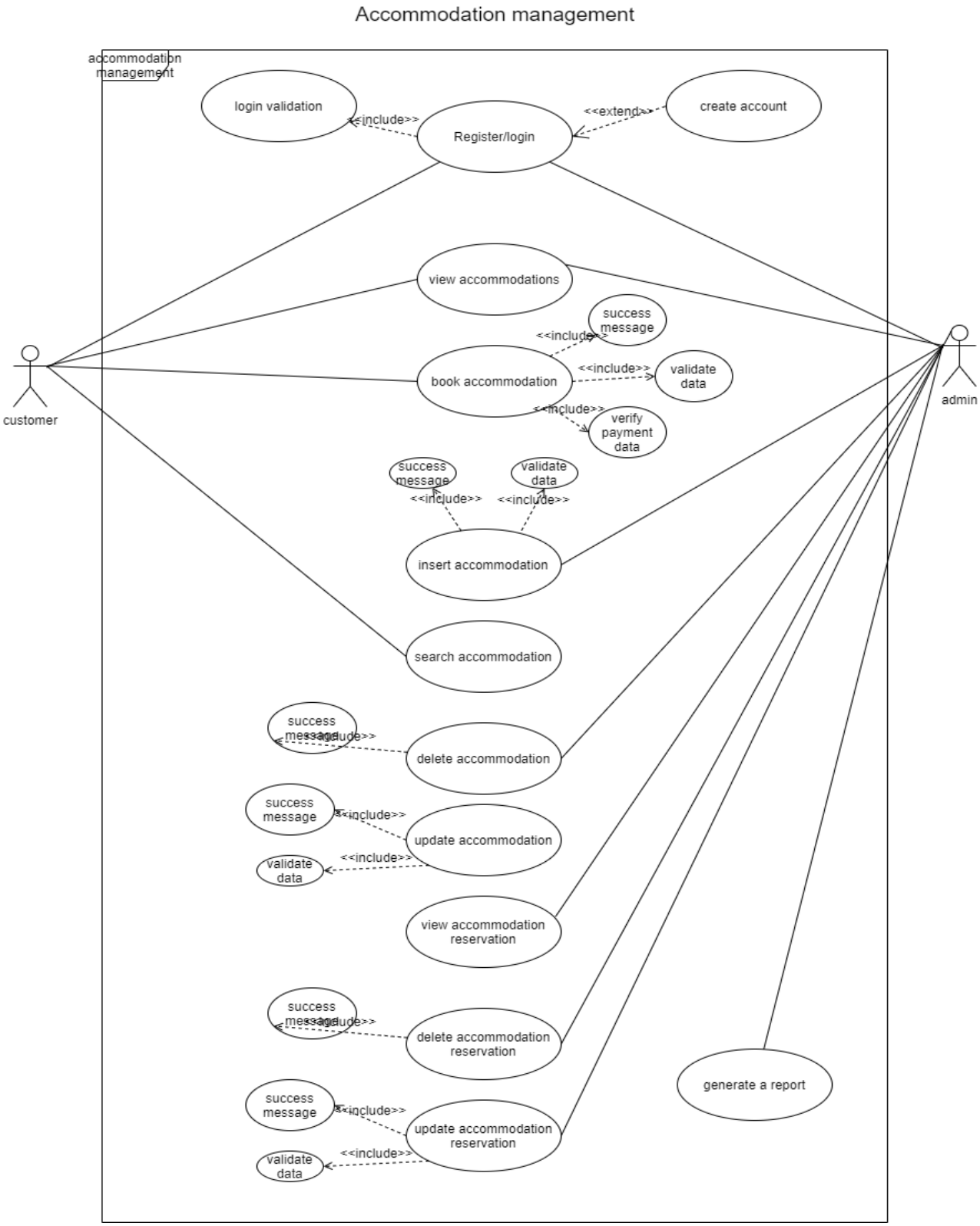


Figure 2

2.1.3 Reservation of Tickets Management

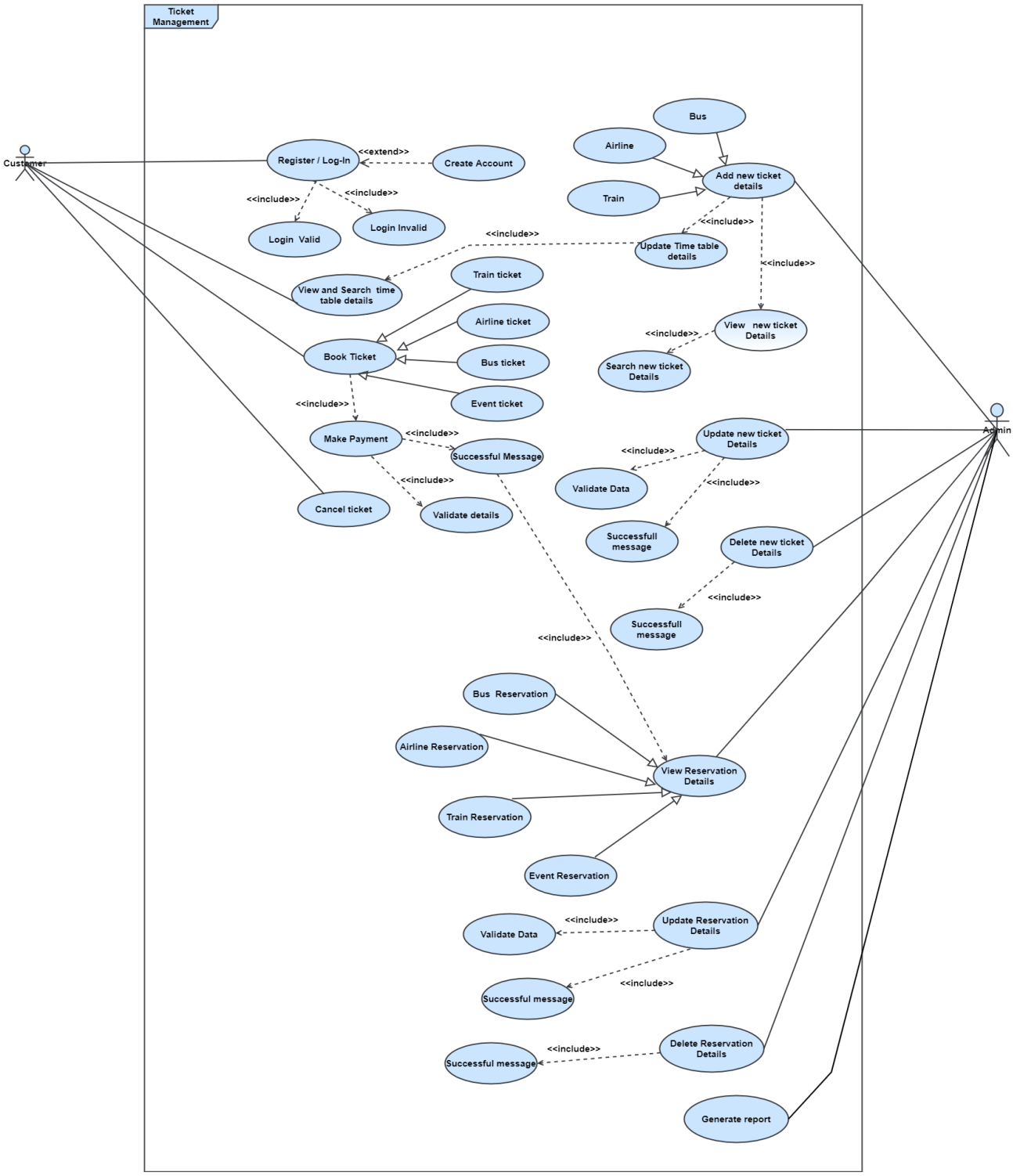


Figure 3

2.1.4 Reservation of ground Management

Figure 4

2.1.5 Travel Document and Travel information Management

Figure 5

2.1.6 Tour Package Management

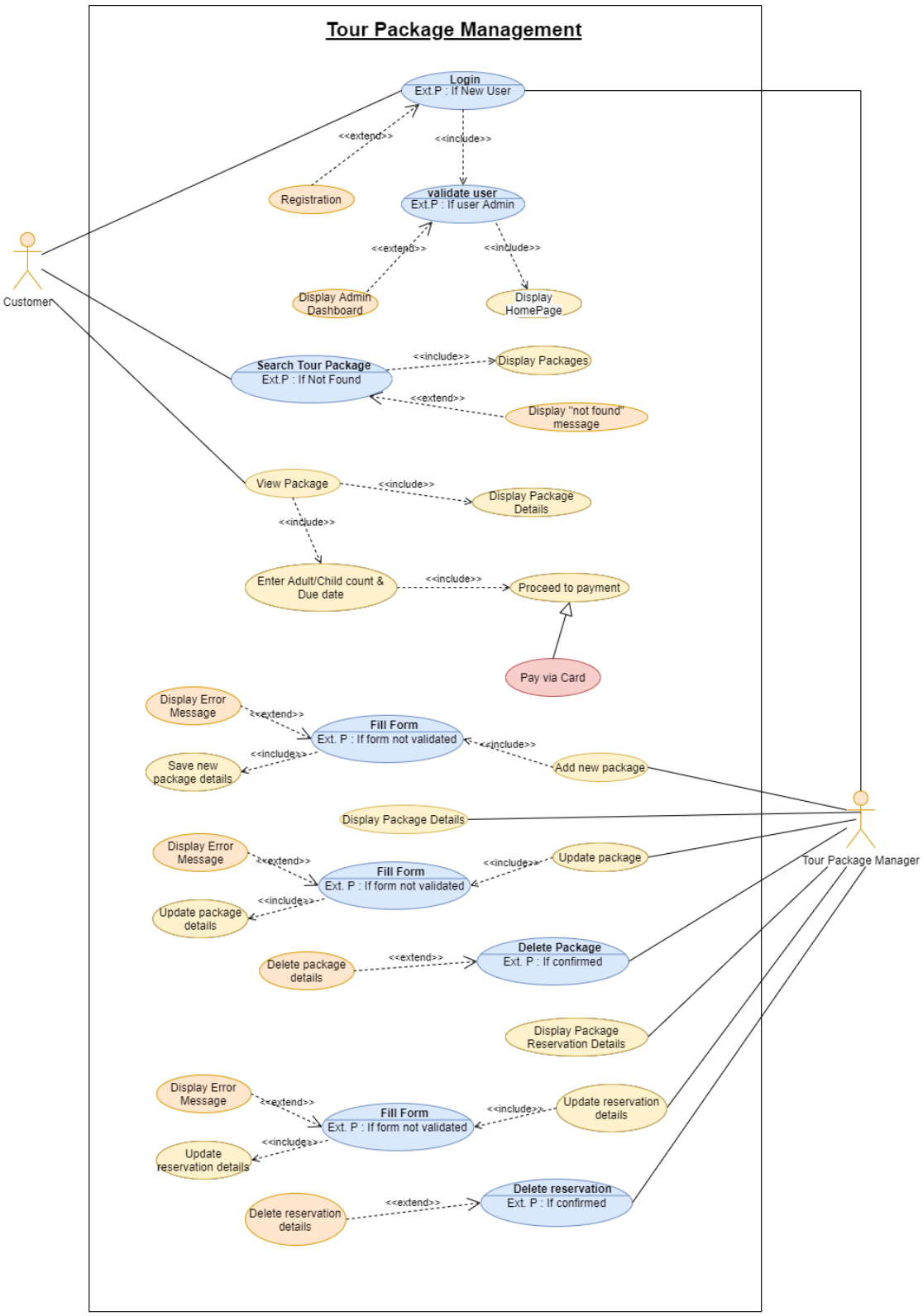


Figure 6

2.1.7 Travel Insurance Management

Figure 7

2.1.8 Event Management

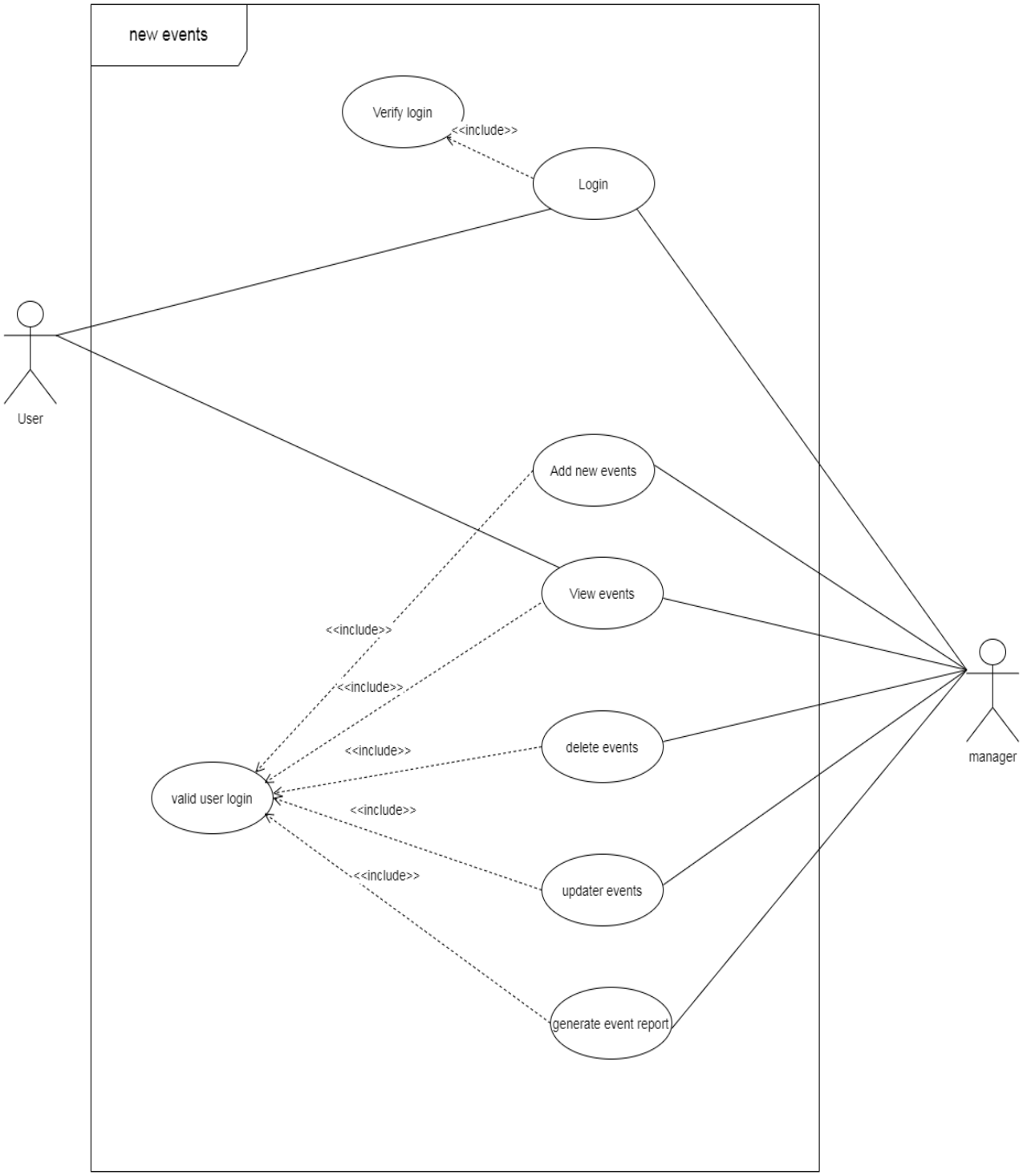
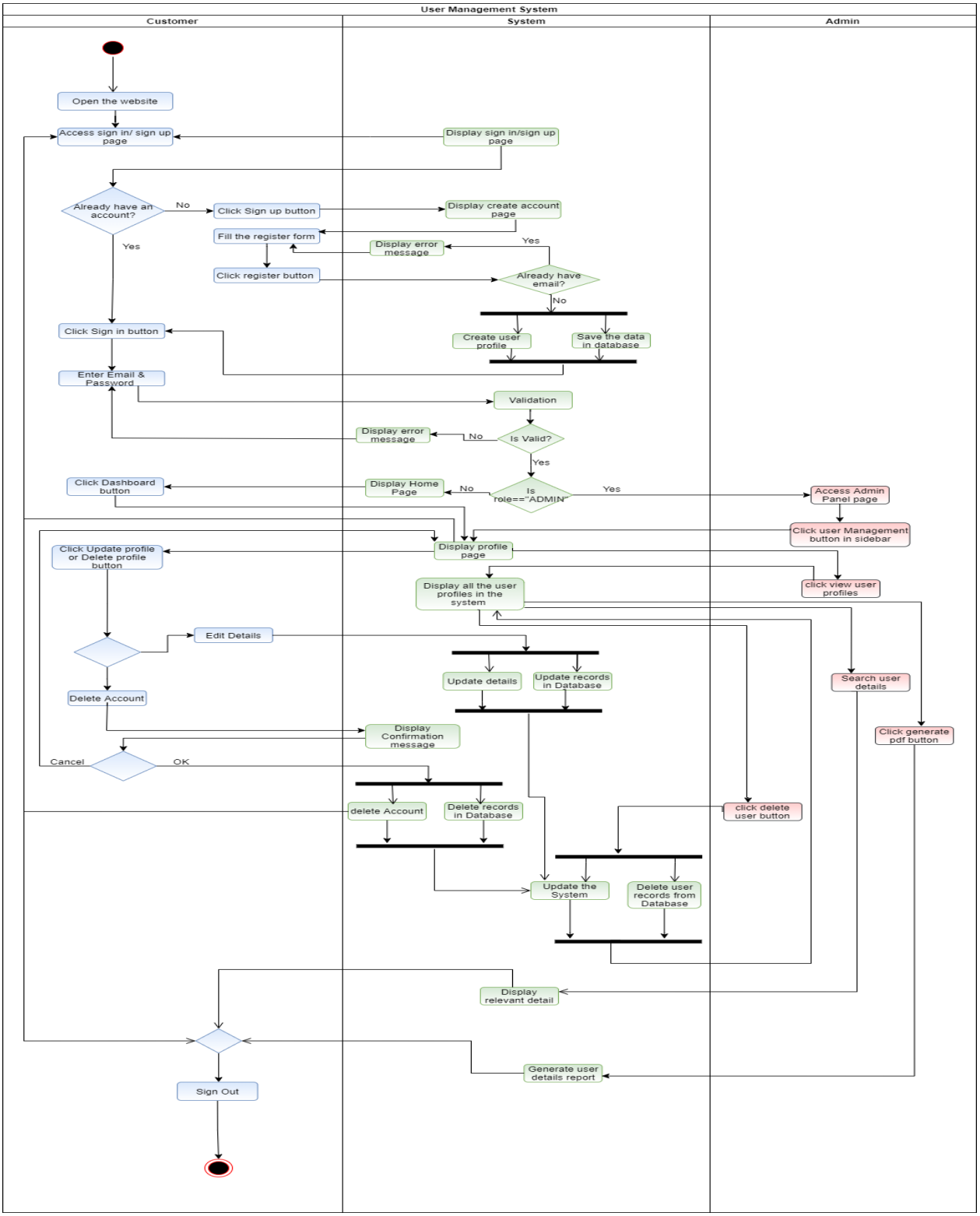


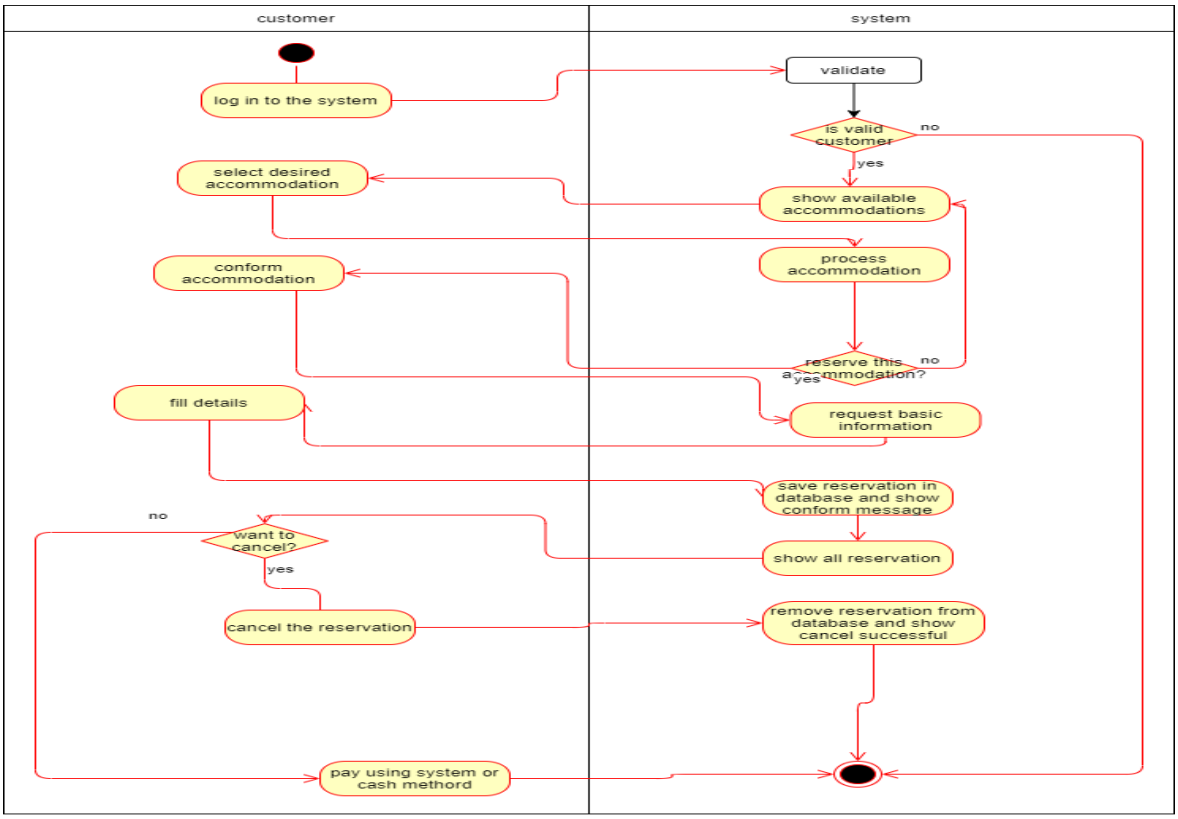
Figure 8

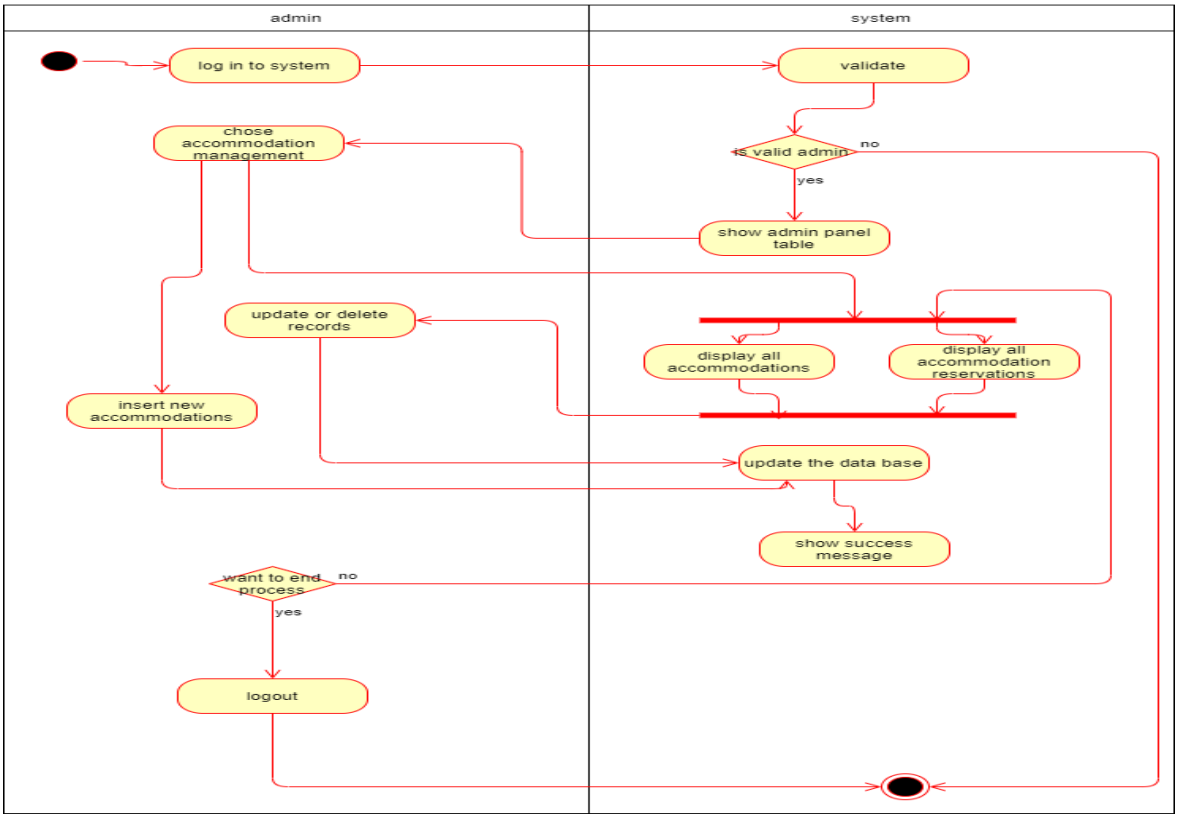
Activity Diagram



User Account Management

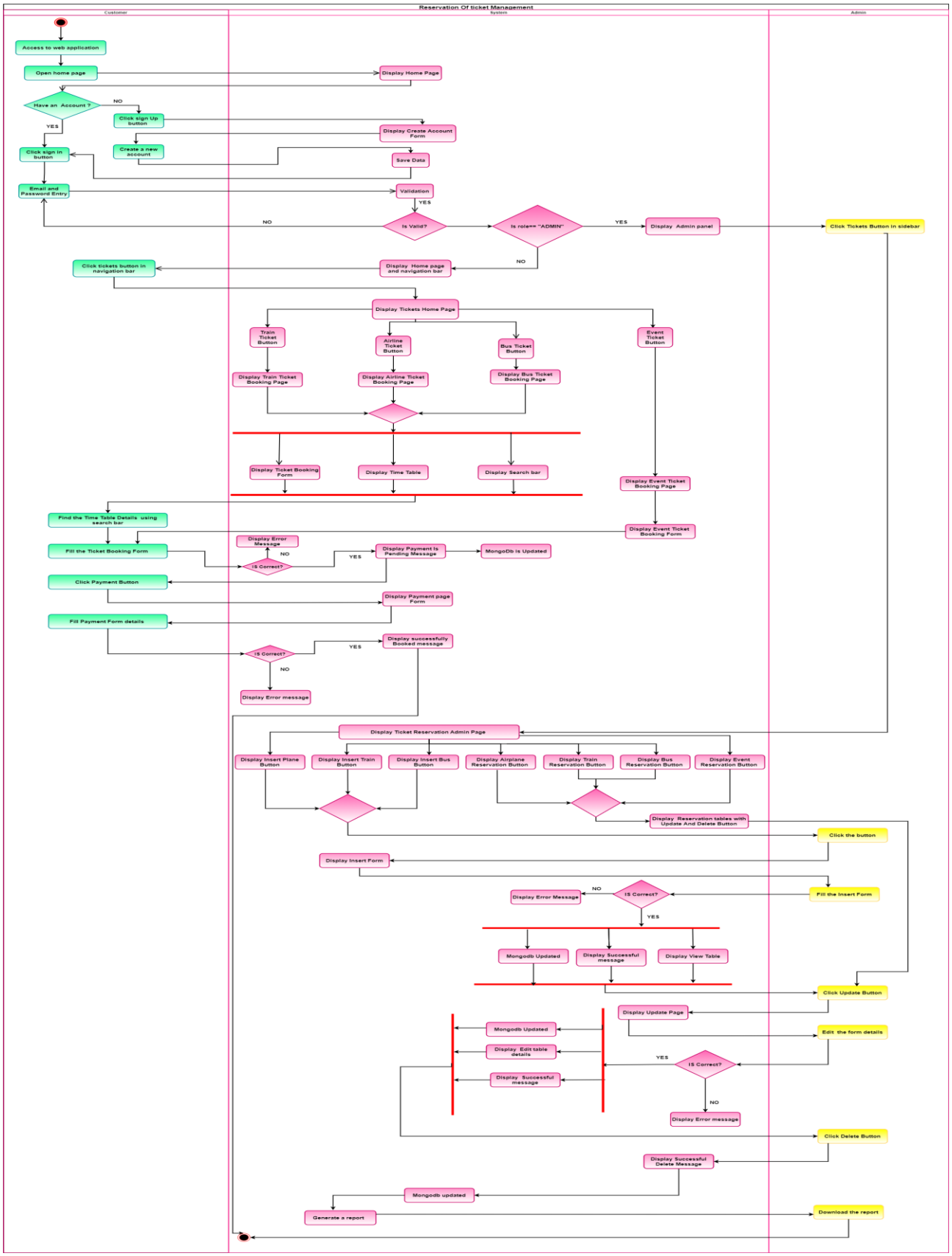
Figure 9





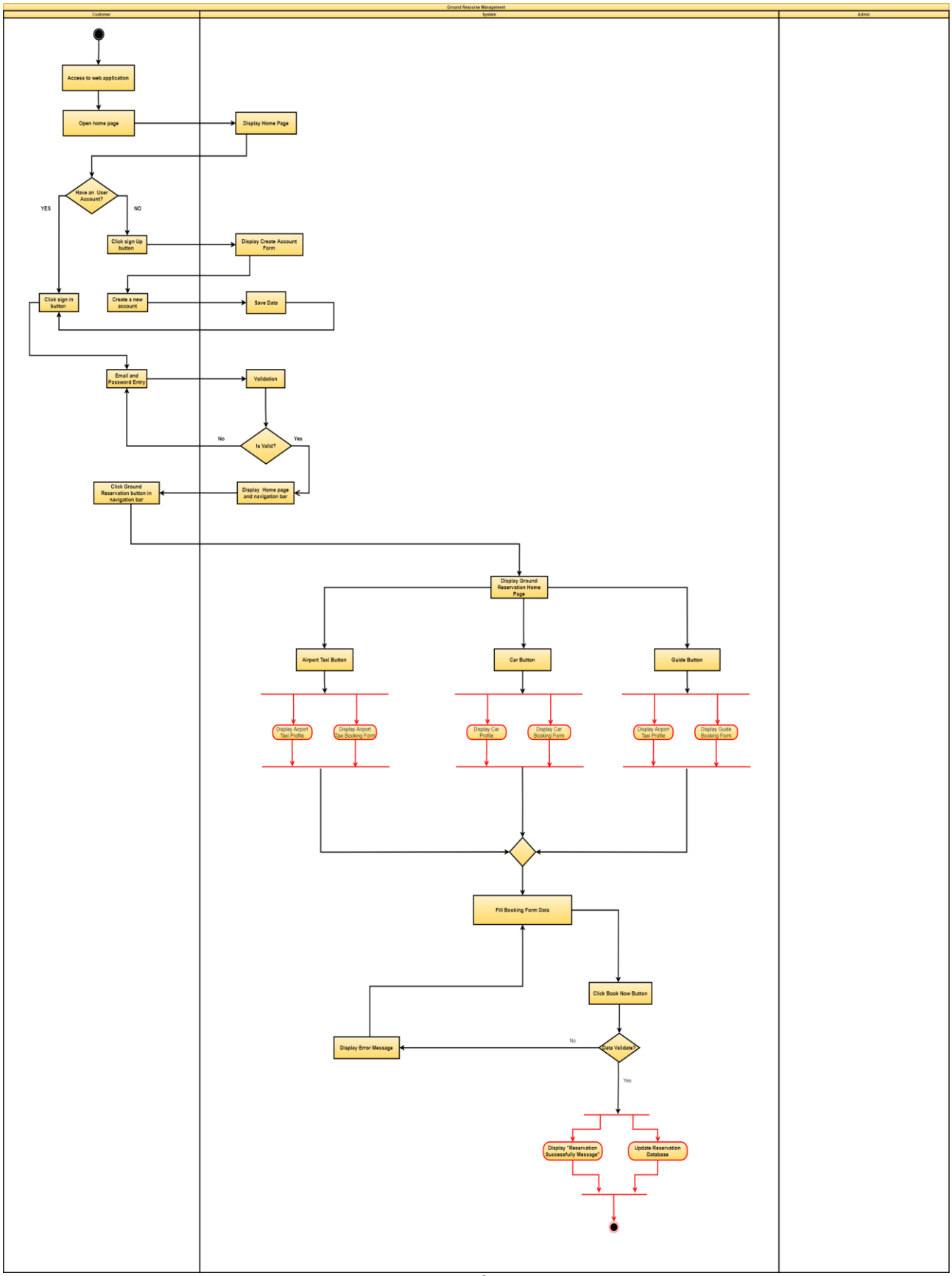
Accommodation reservation management

Figure 10



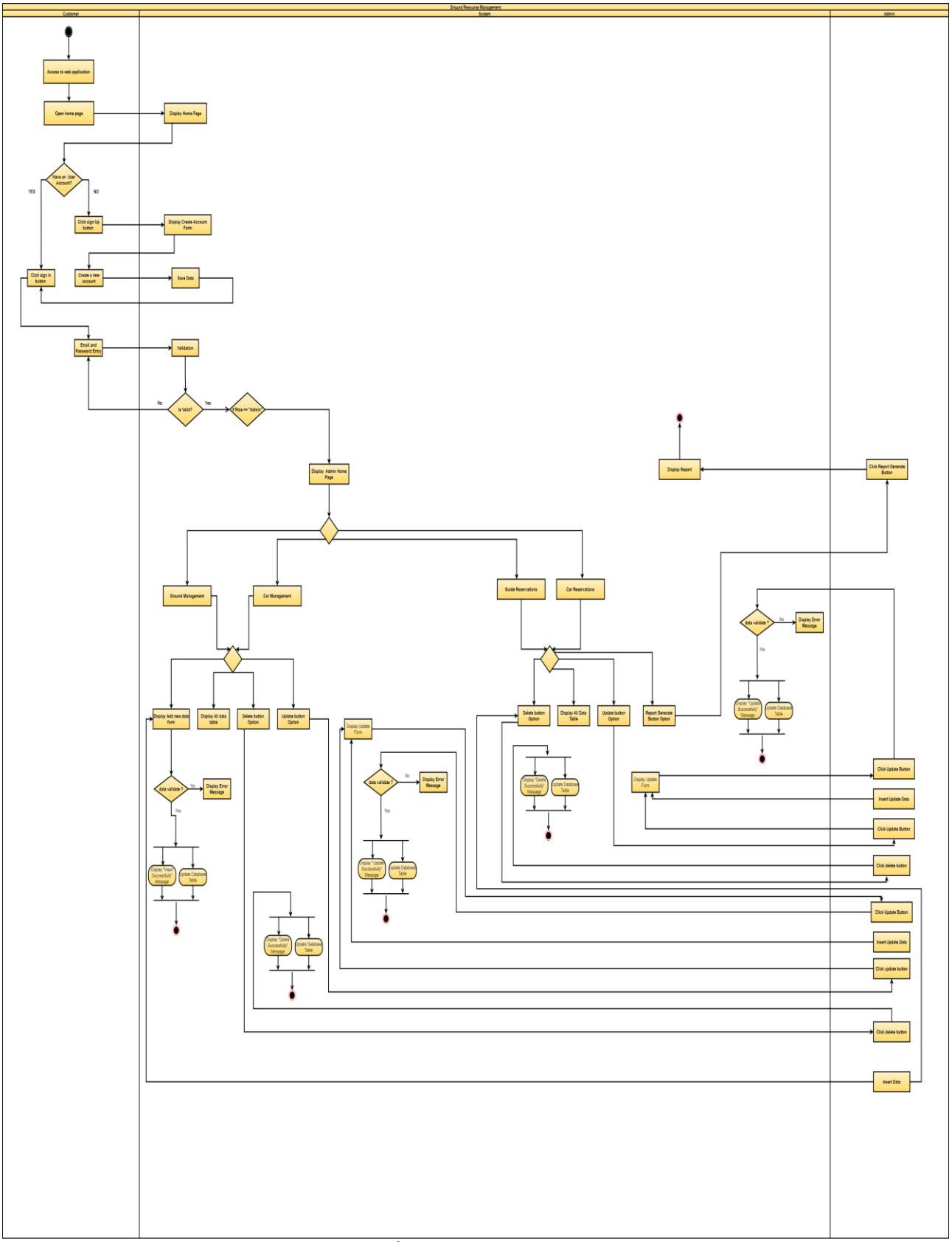
Reservation of Tickets Management

Figure 11



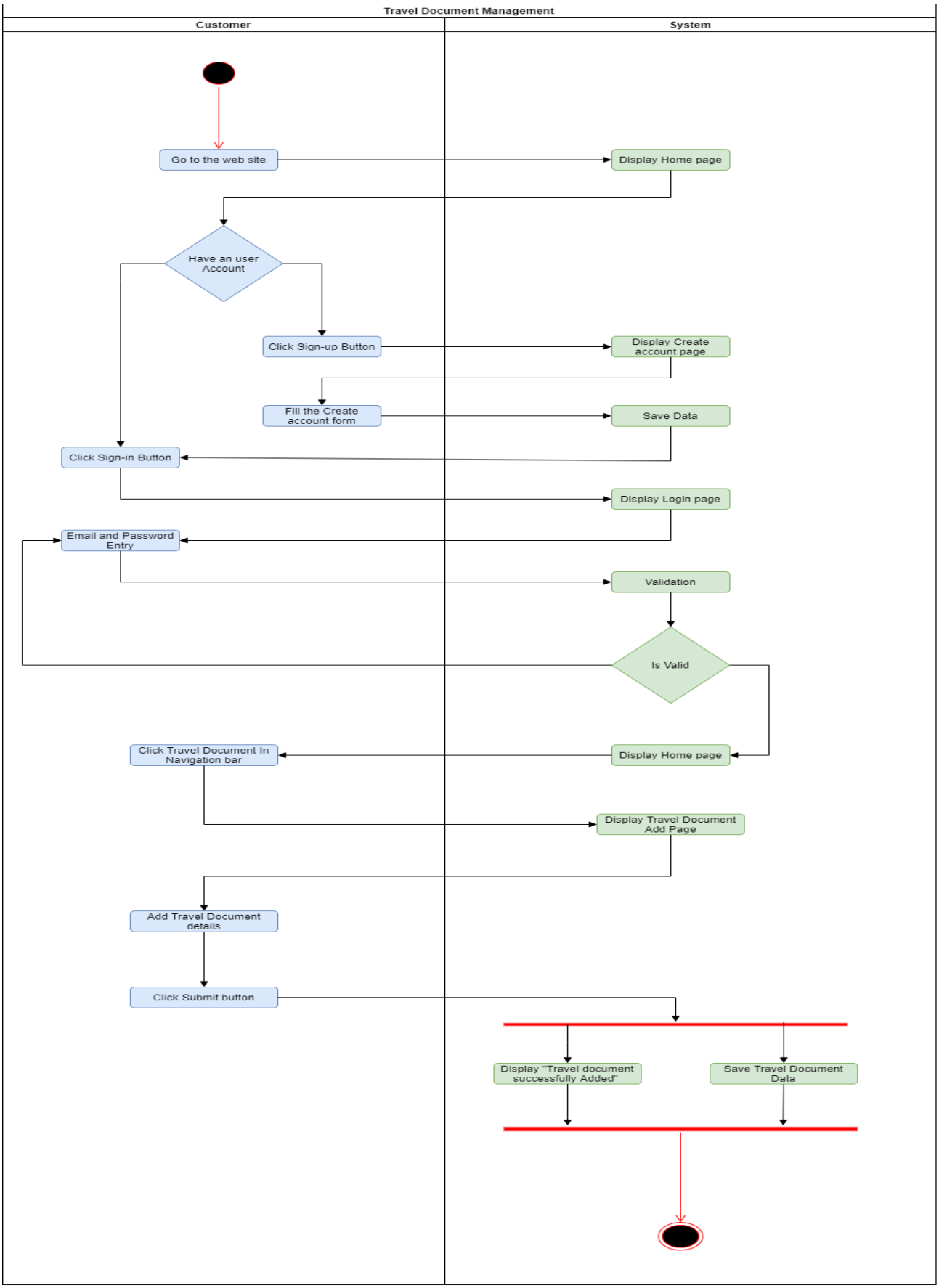
User Reservation of Ground Management

Figure 12



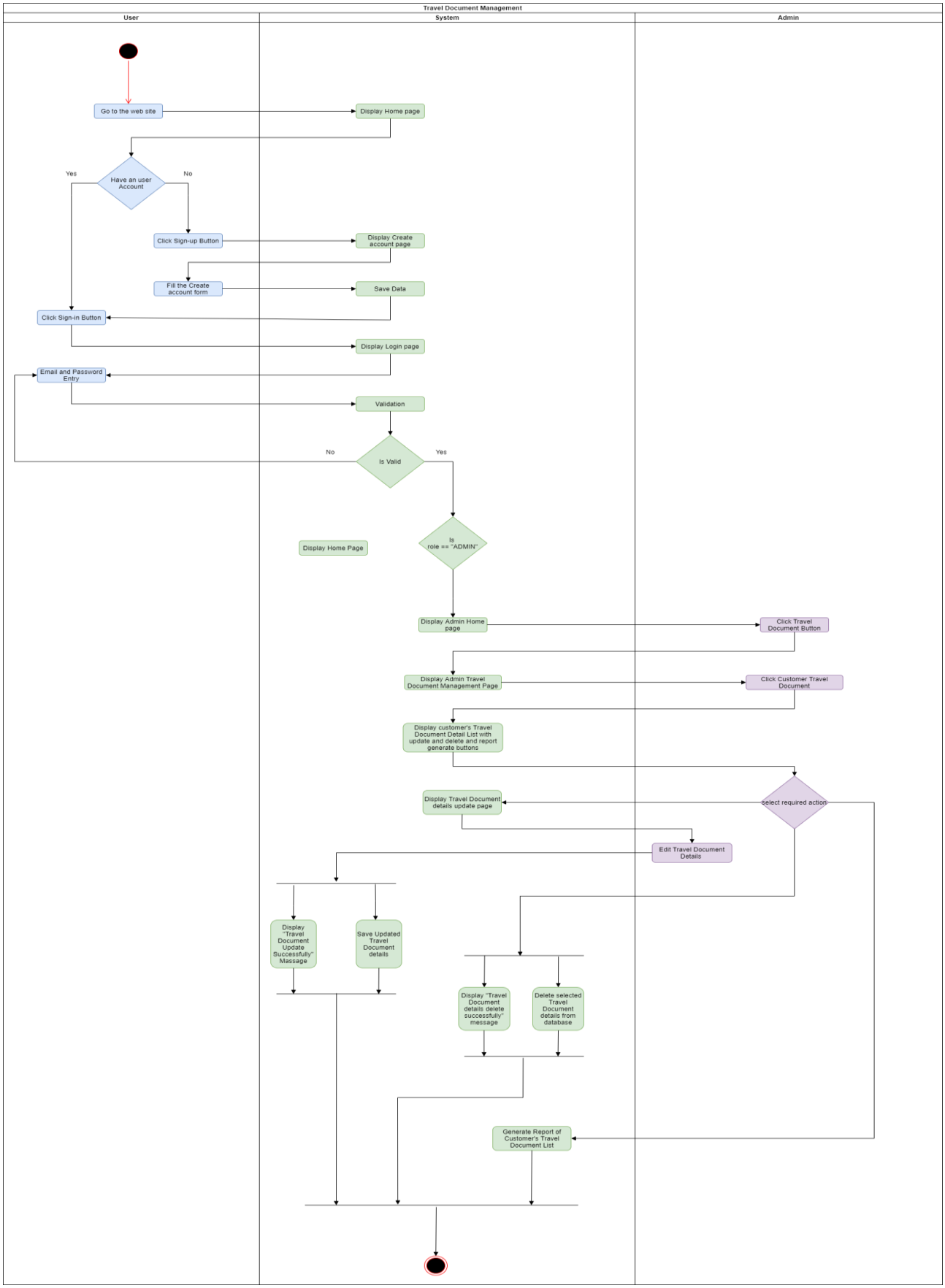
Admin Reservation of Ground Resources Management

Figure 13



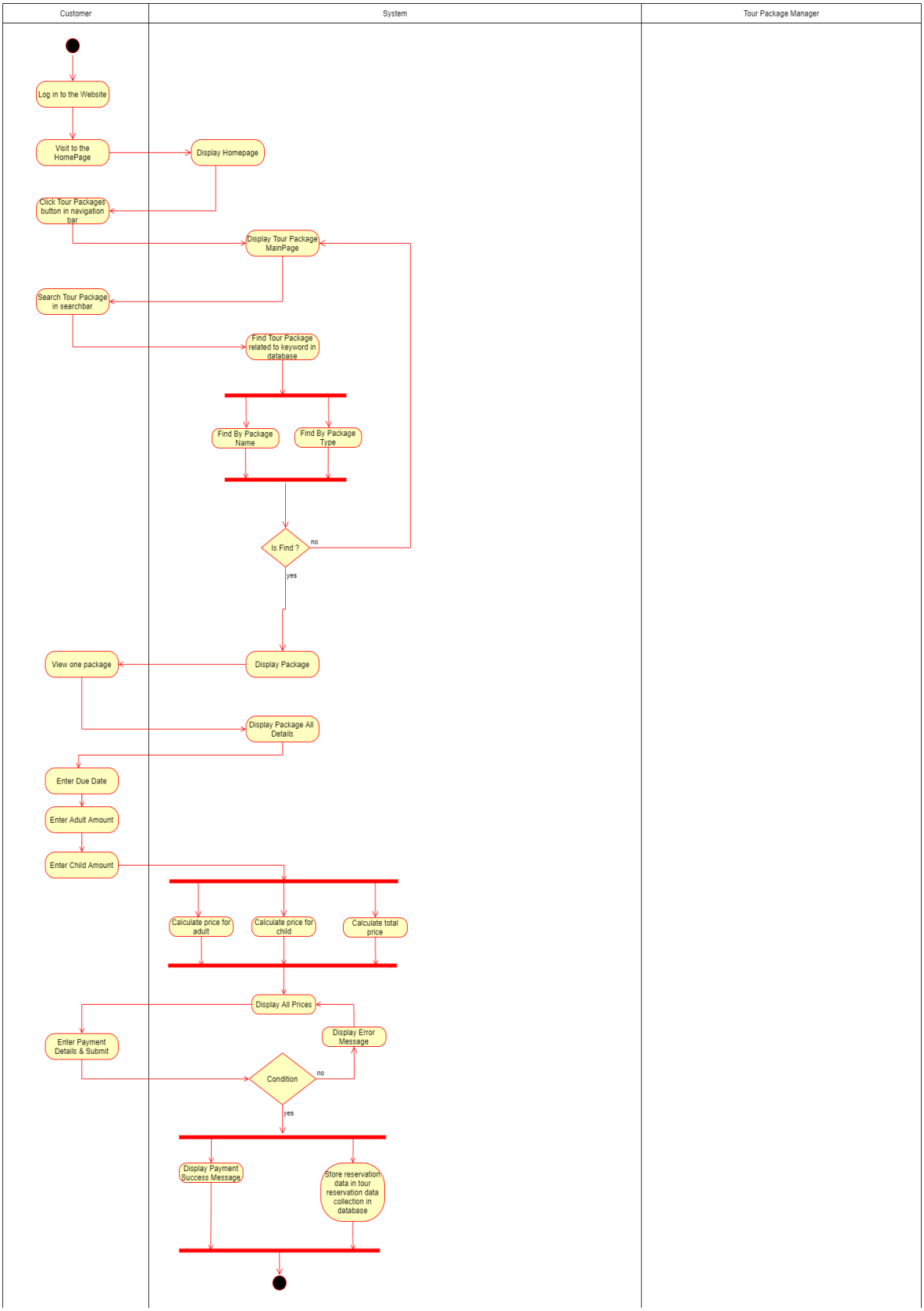
User Travel Document Management

Figure 14



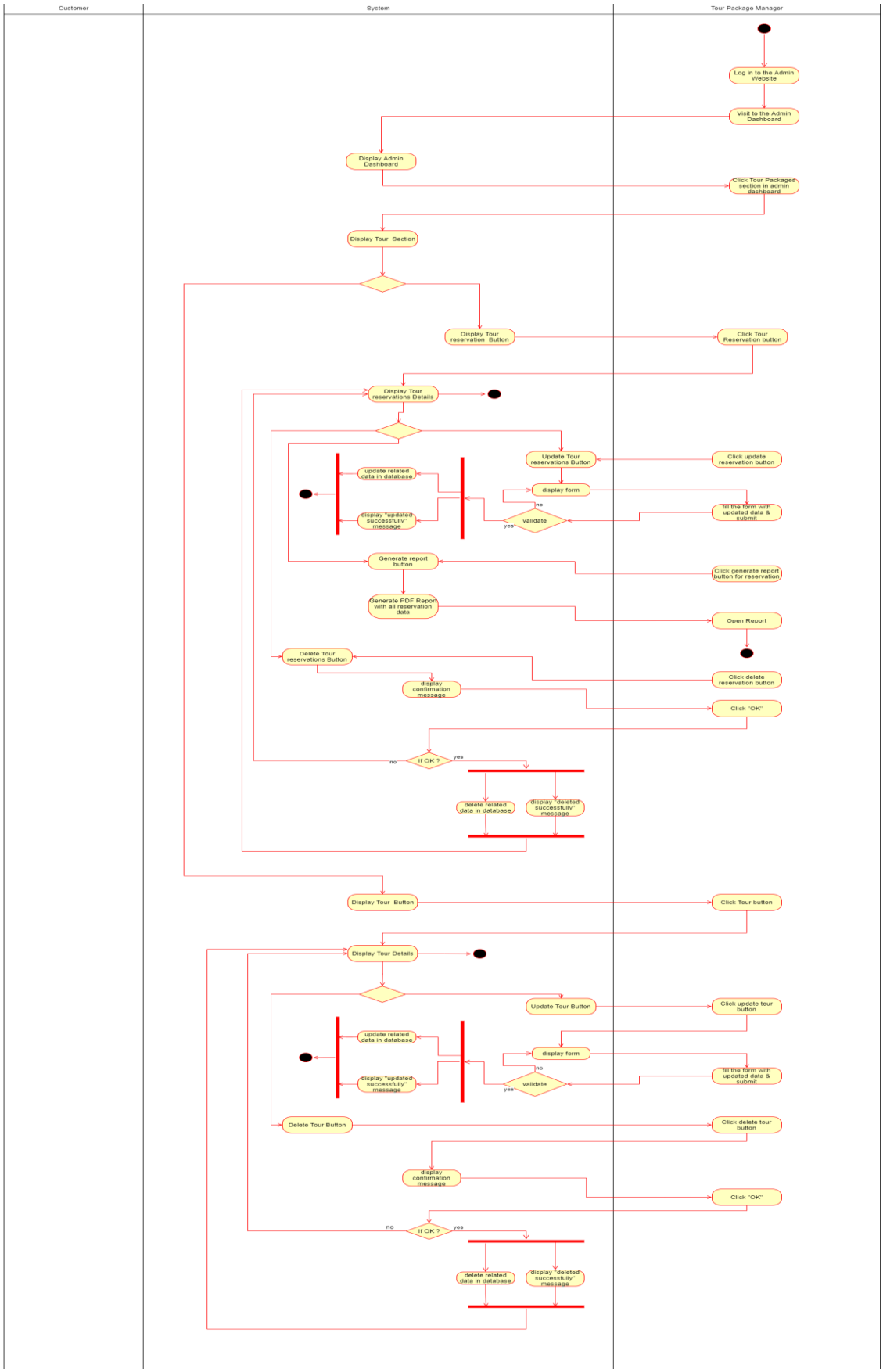
Admin Travel Document Management

Figure 15



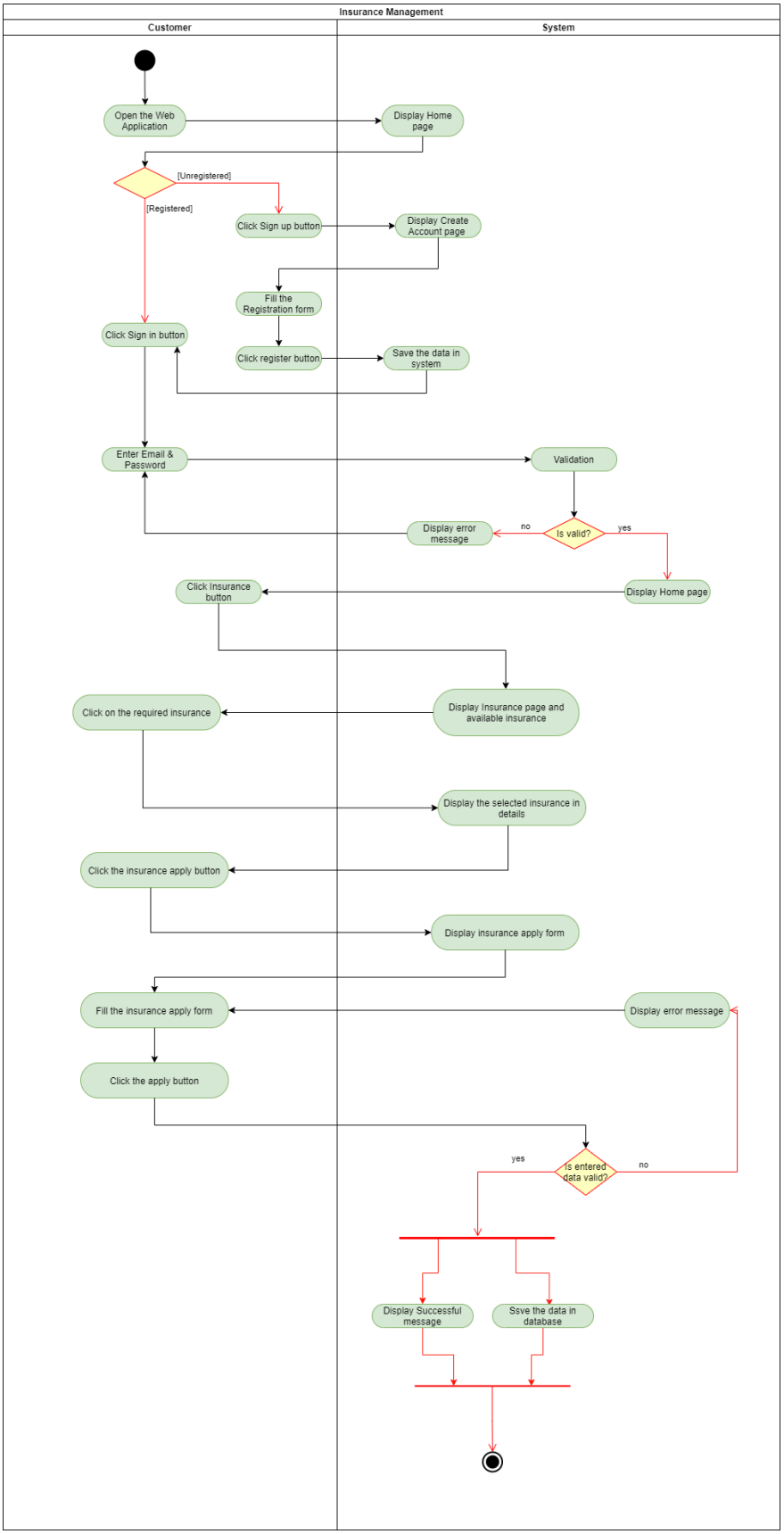
Tour Package Management (USER SIDE)

Figure 16



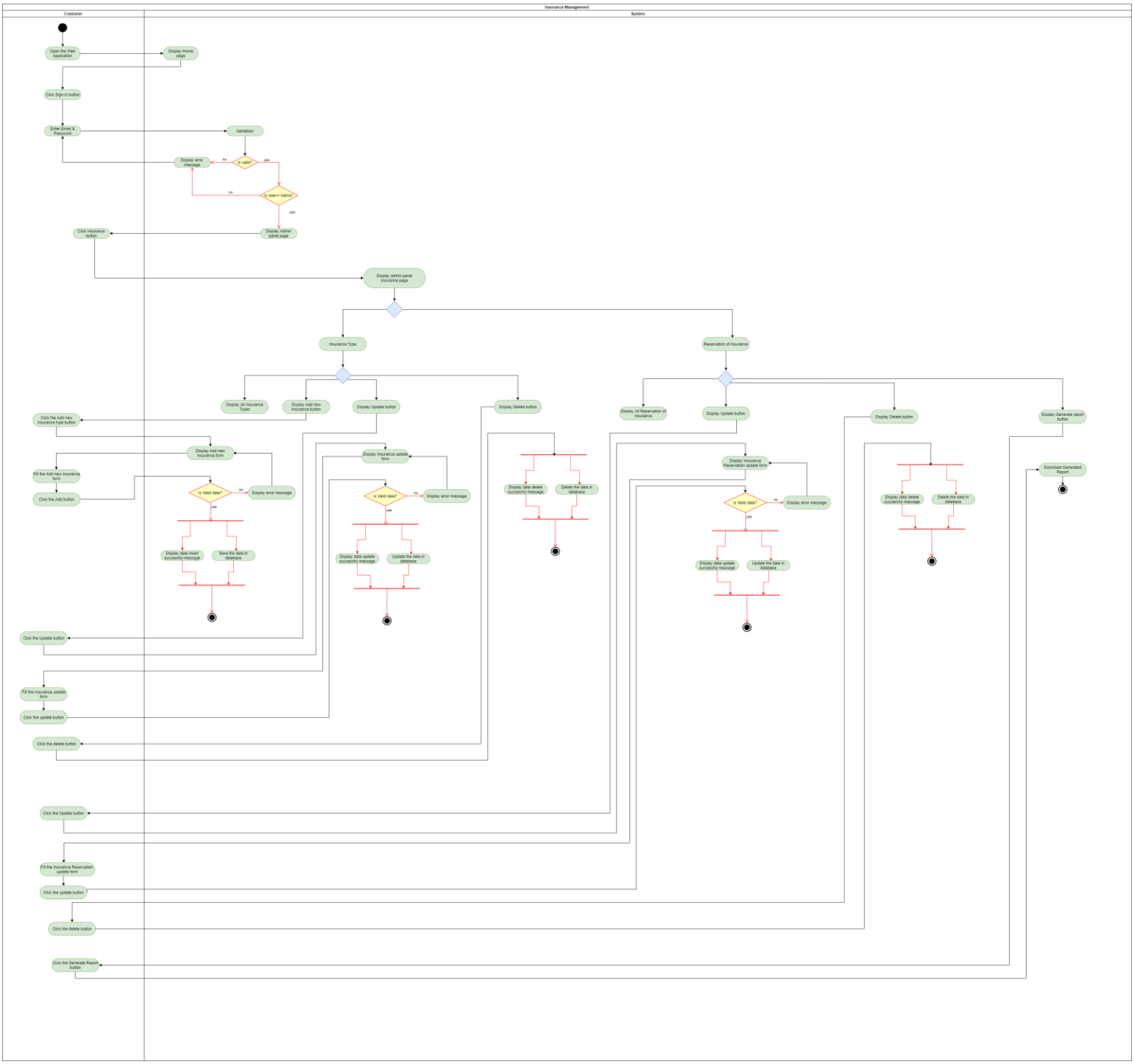
Tour Package Management (ADMIN SIDE)

Figure 17



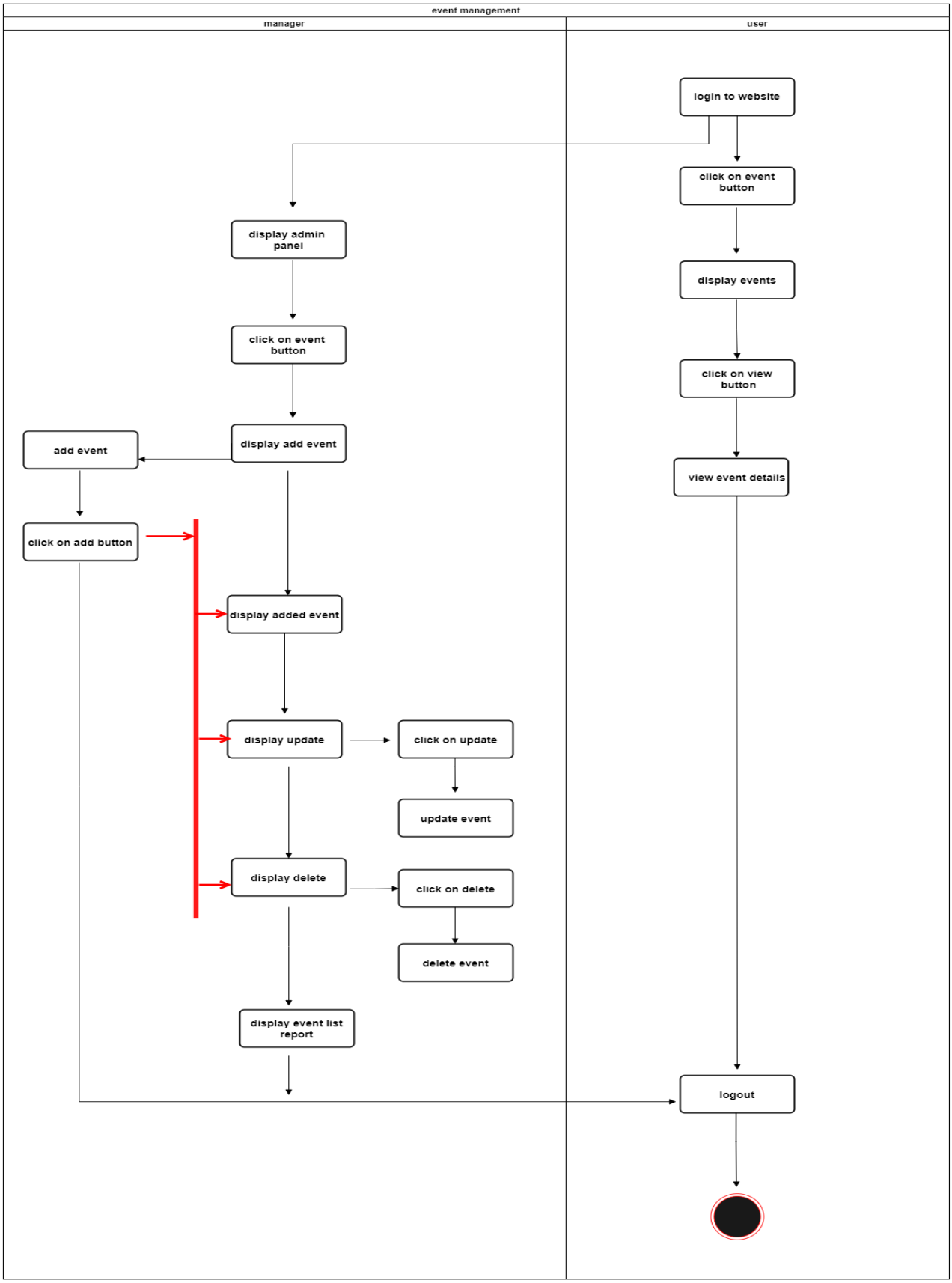
Travel Insurance Management (USER SIDE)

Figure 18



Travel Insurance Management (ADMIN SIDE)

Figure 19



Event Management

Figure 20

2.2 Design

2.2.1 ER Diagram

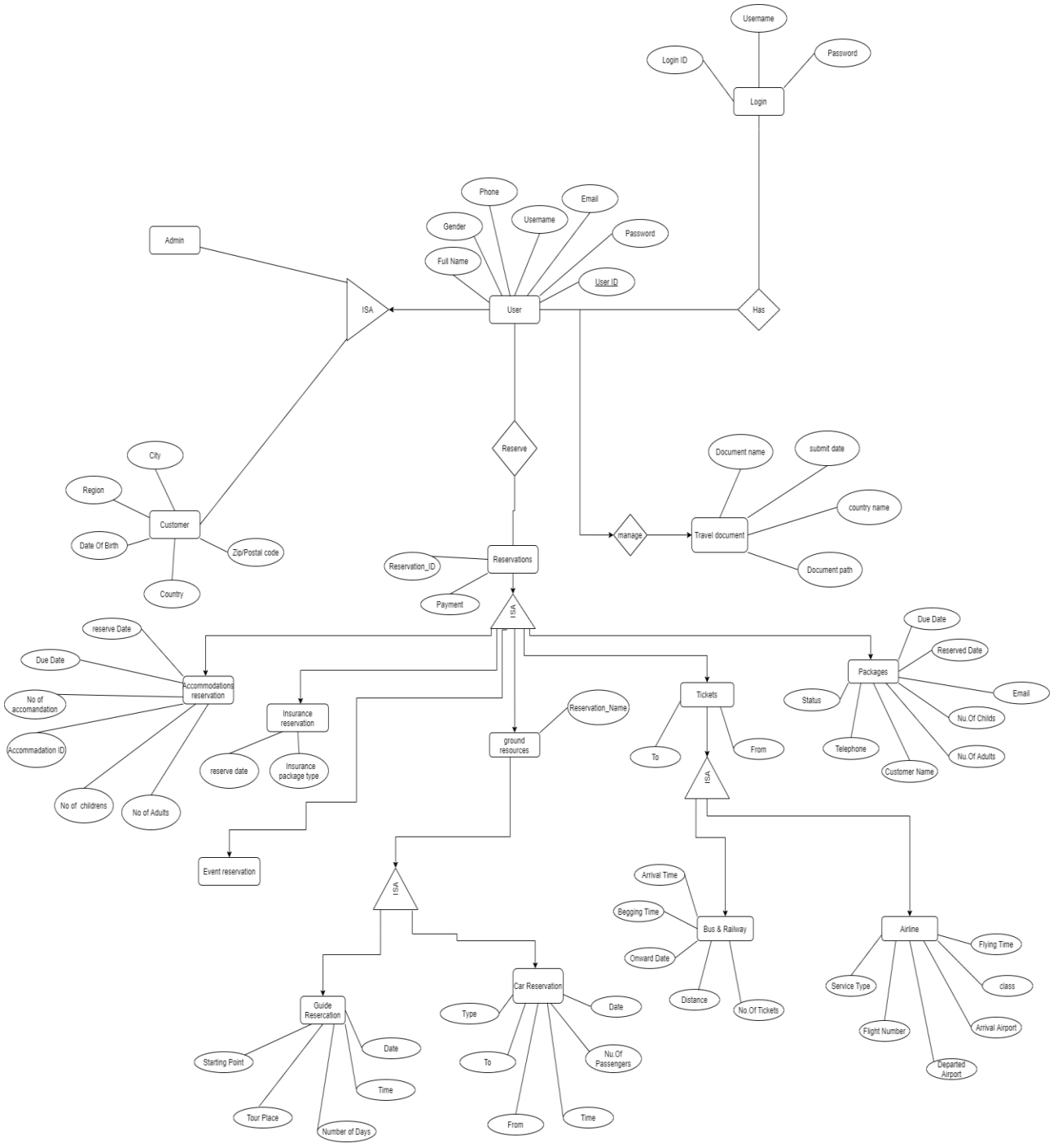
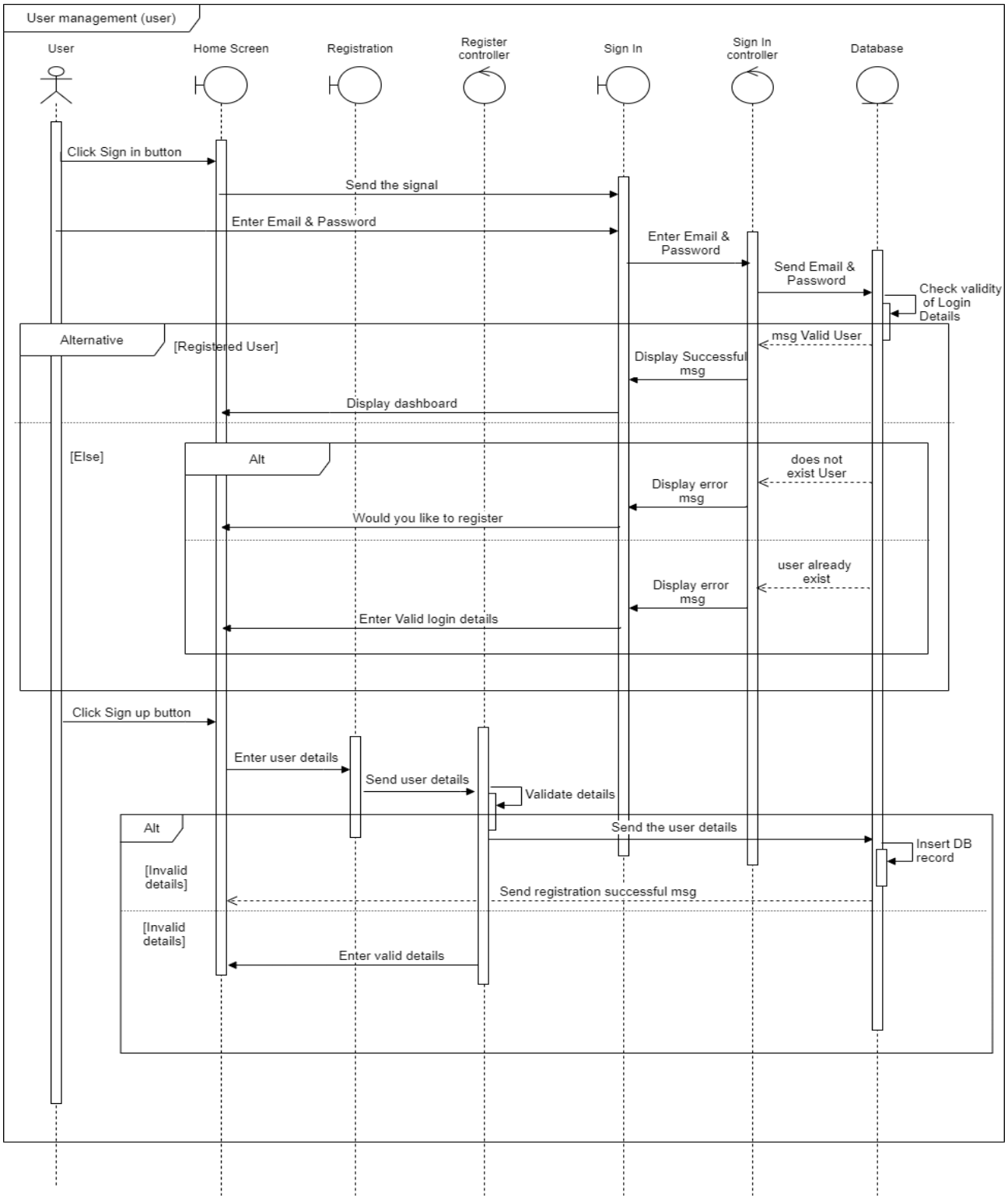


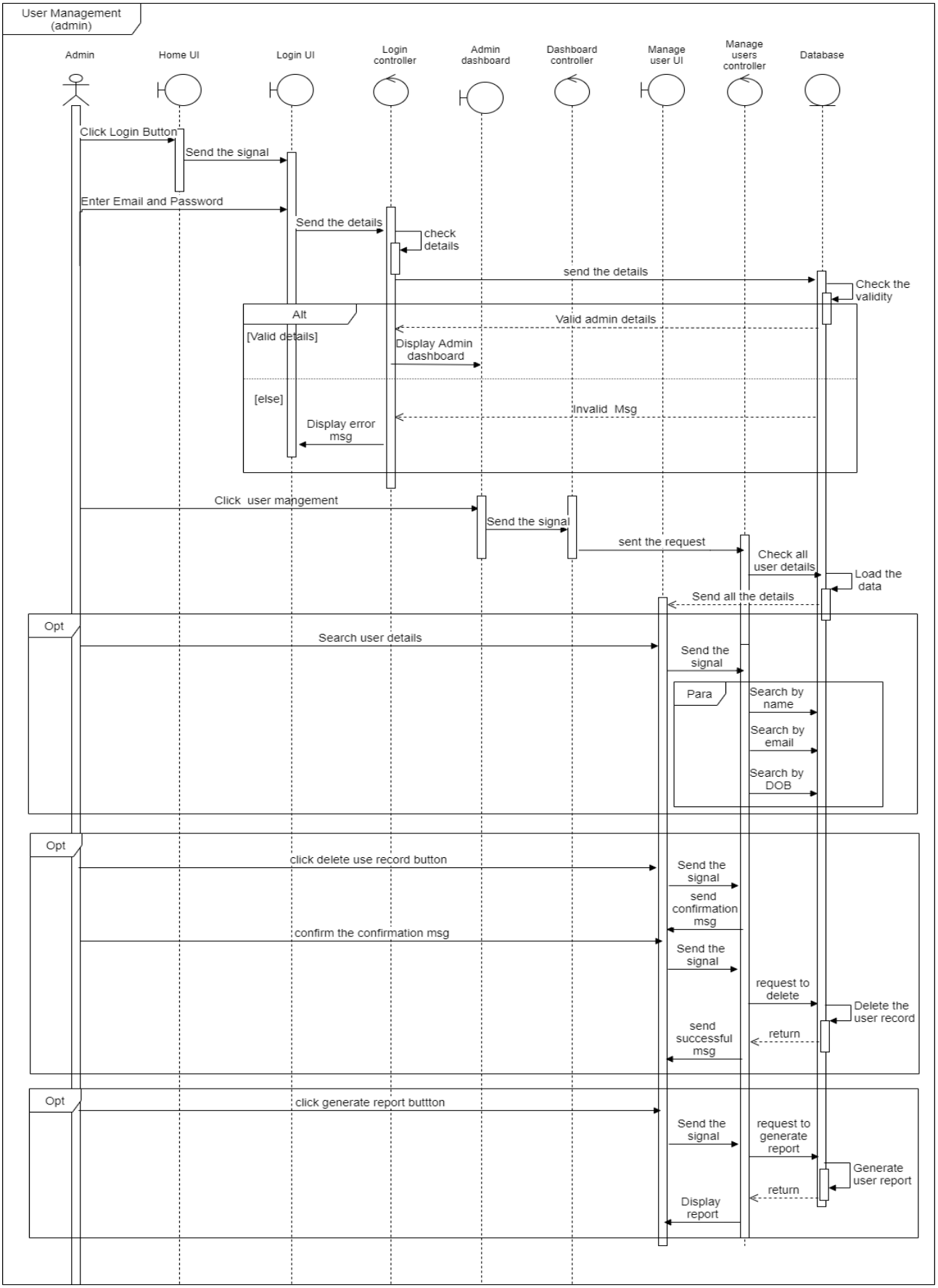
Figure 21

2.2.2 Sequence Diagram



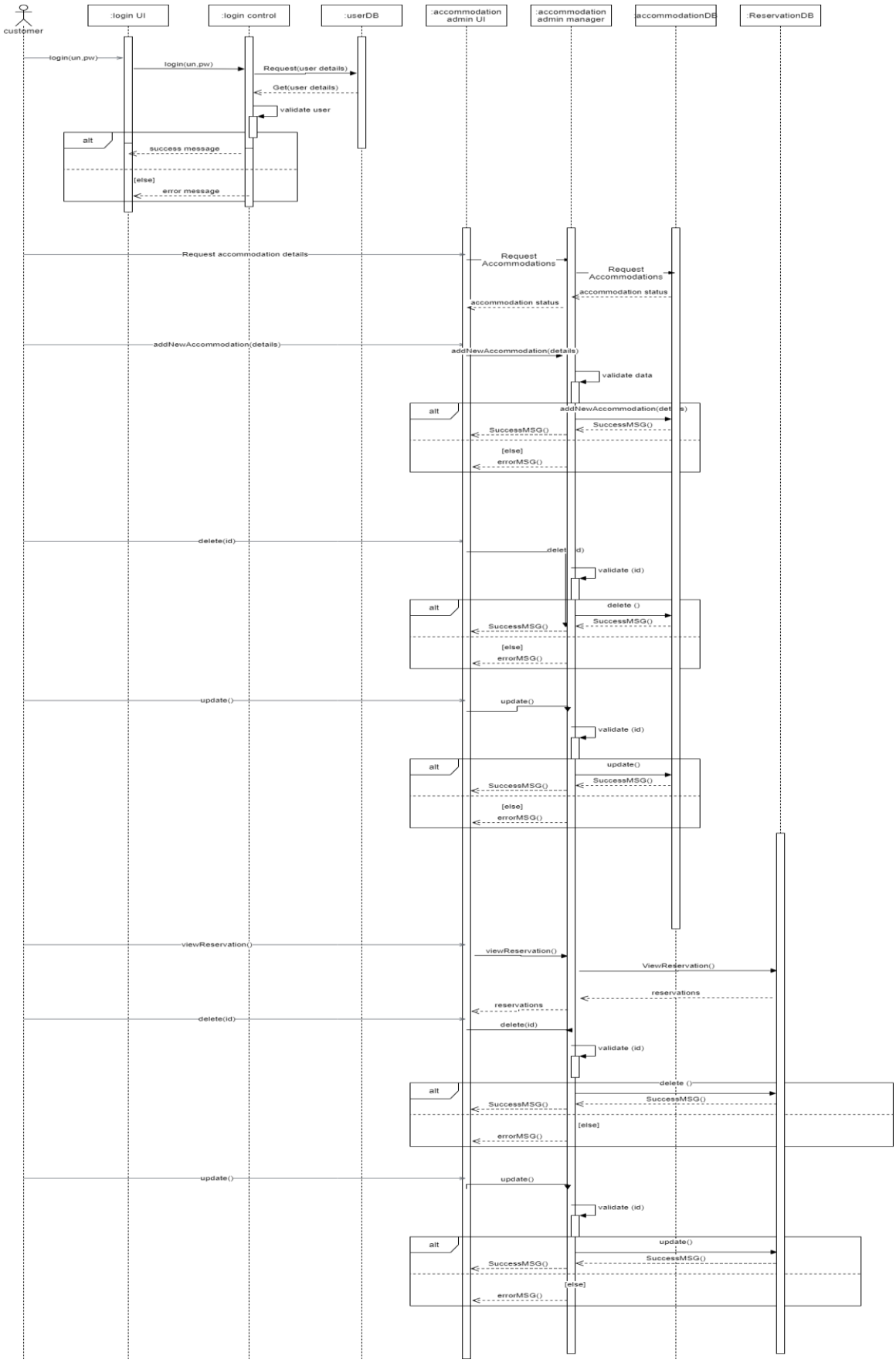
User Management (User side)

Figure 22



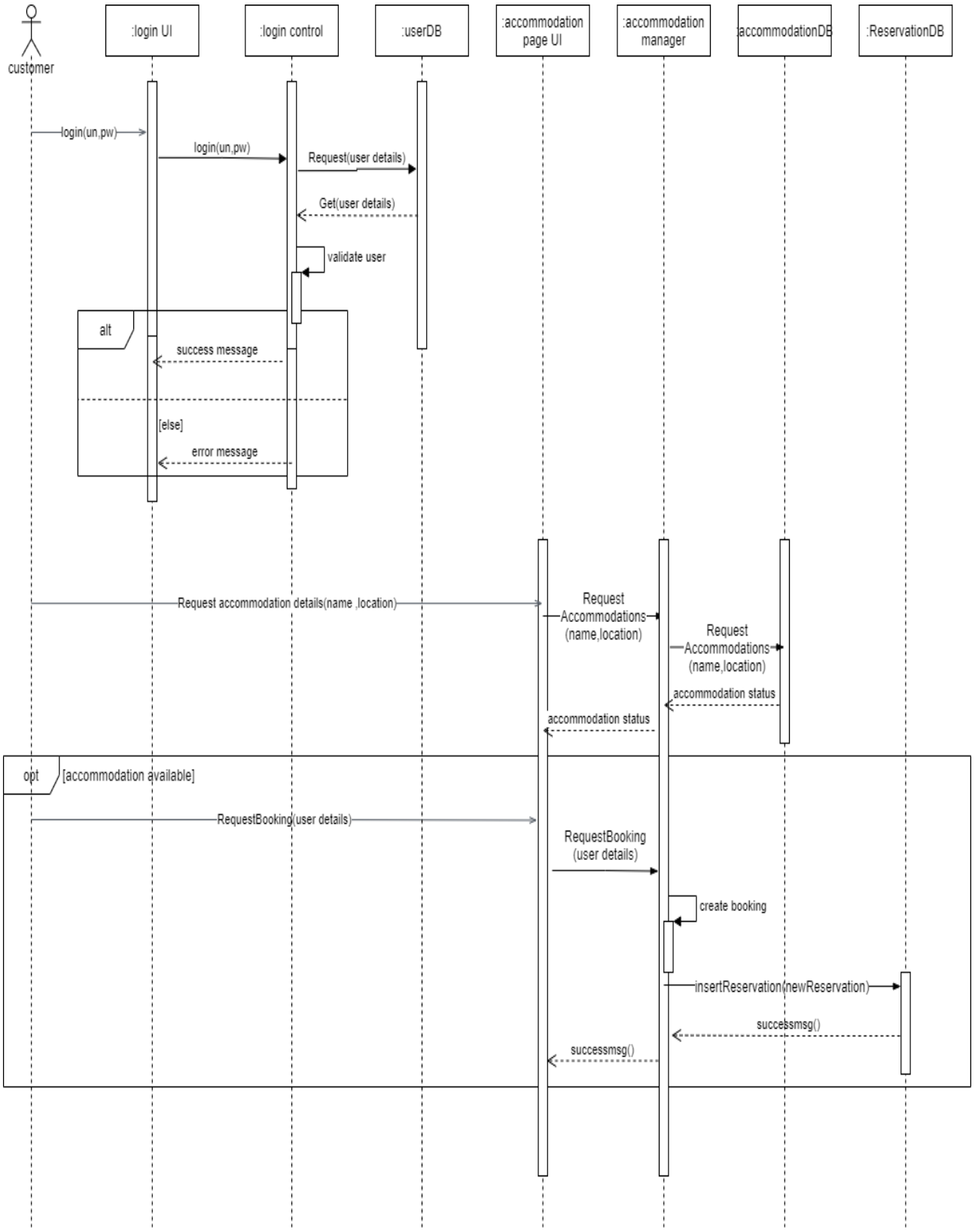
User Management (Admin side)

Figure 23



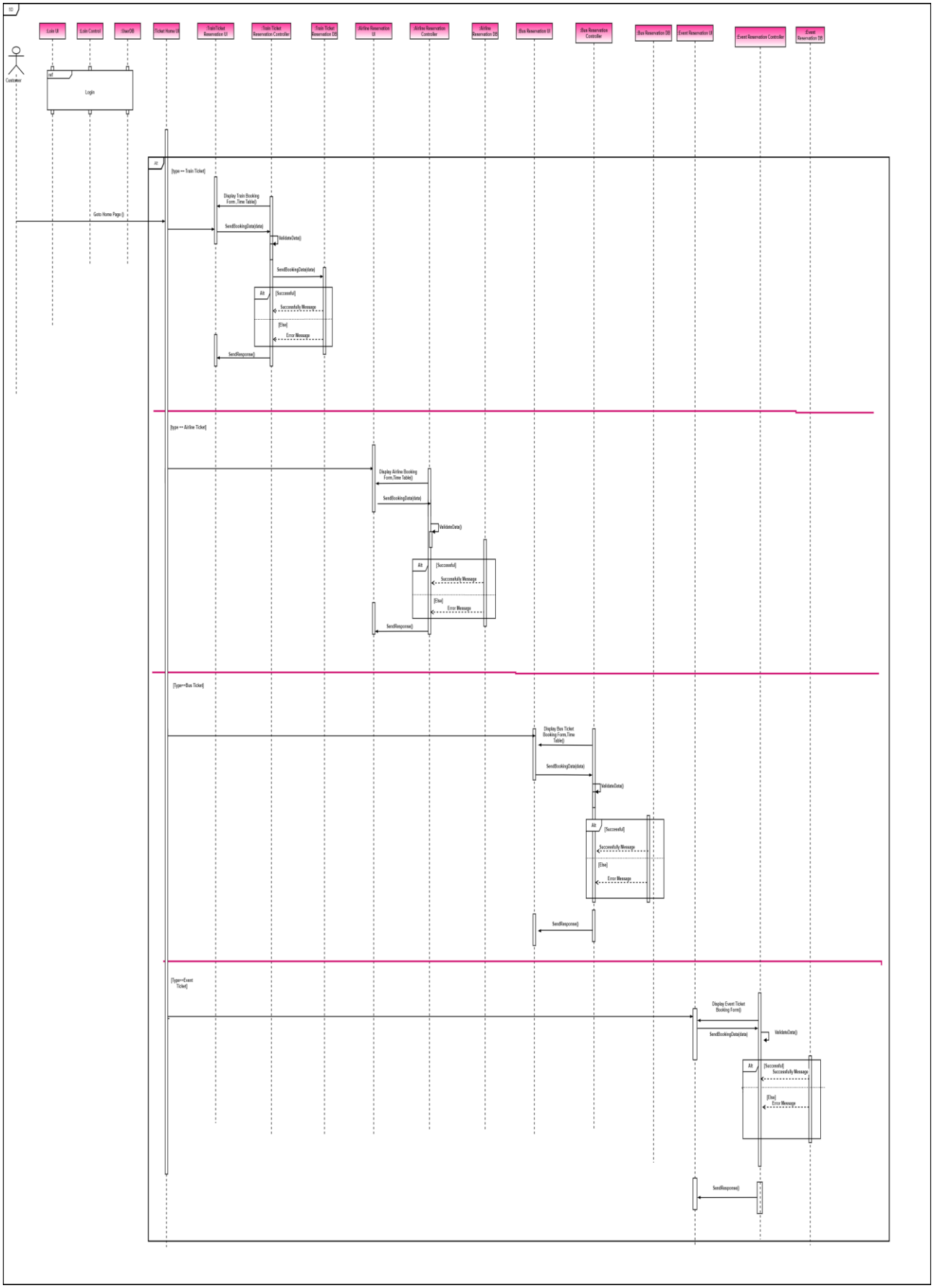
Accommodation reservation Management (User side)

Figure 24



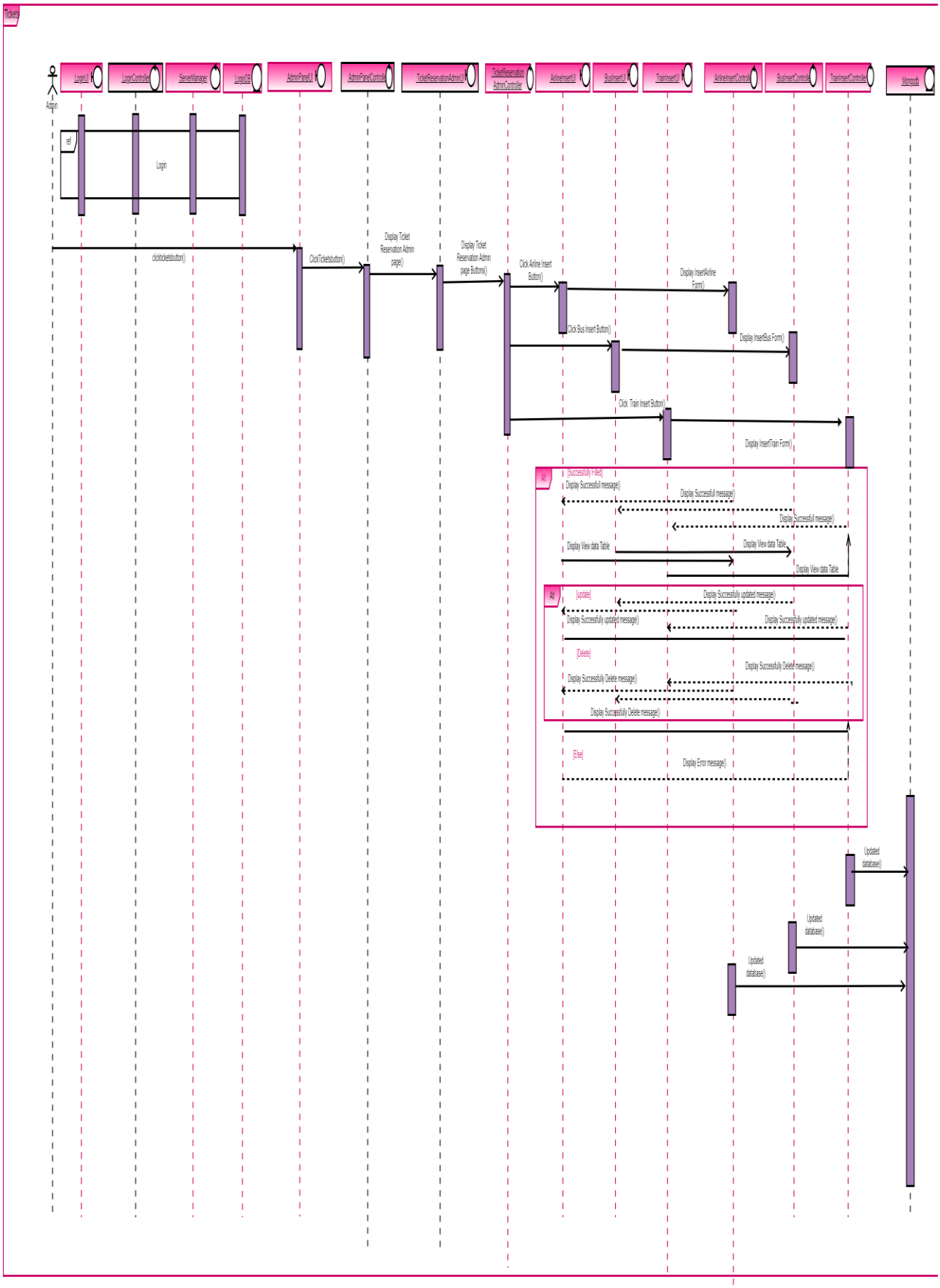
Accommodation reservation Management (Admin side)

Figure 25



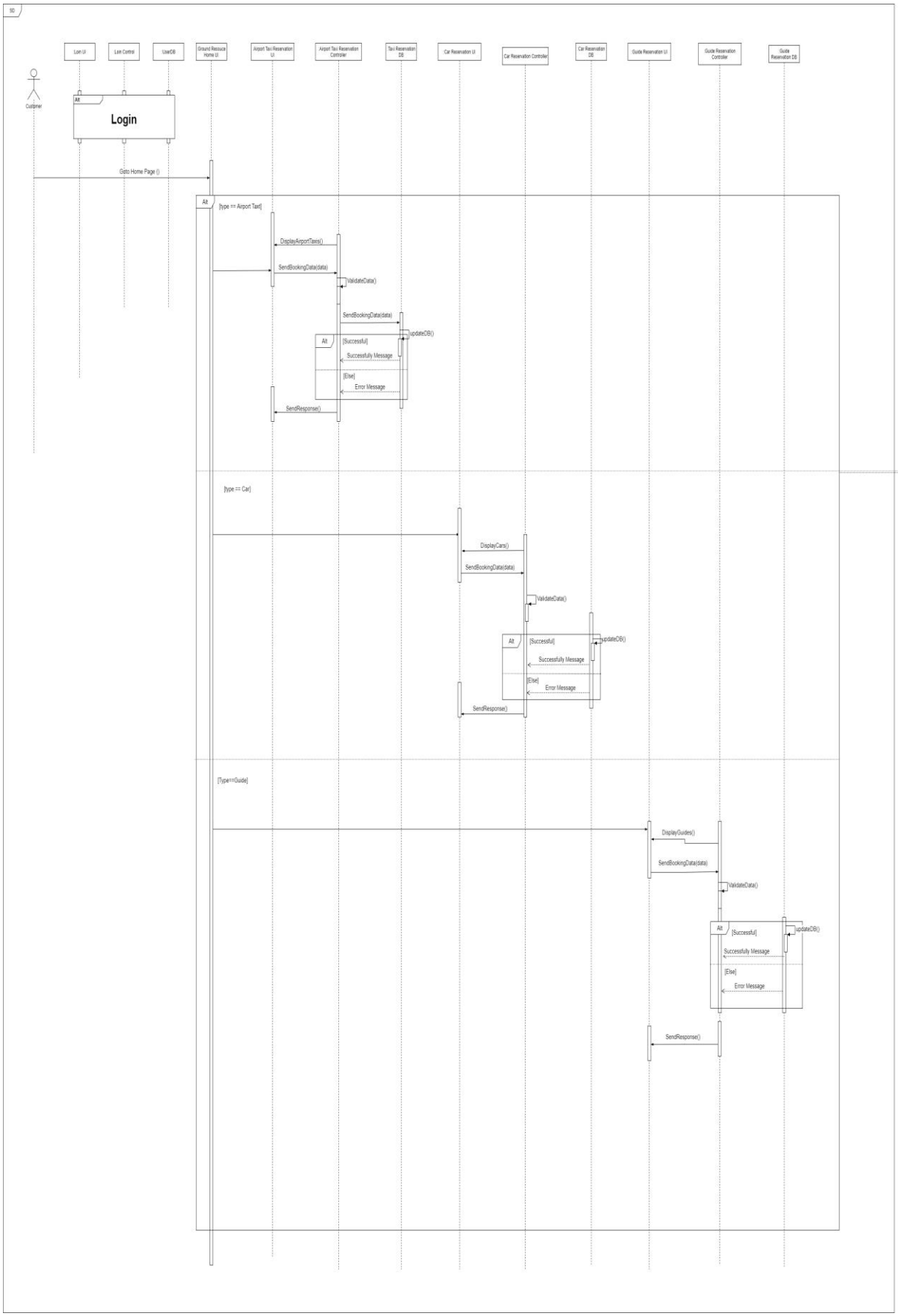
Reservation of Tickets Management (User side)

Figure 26



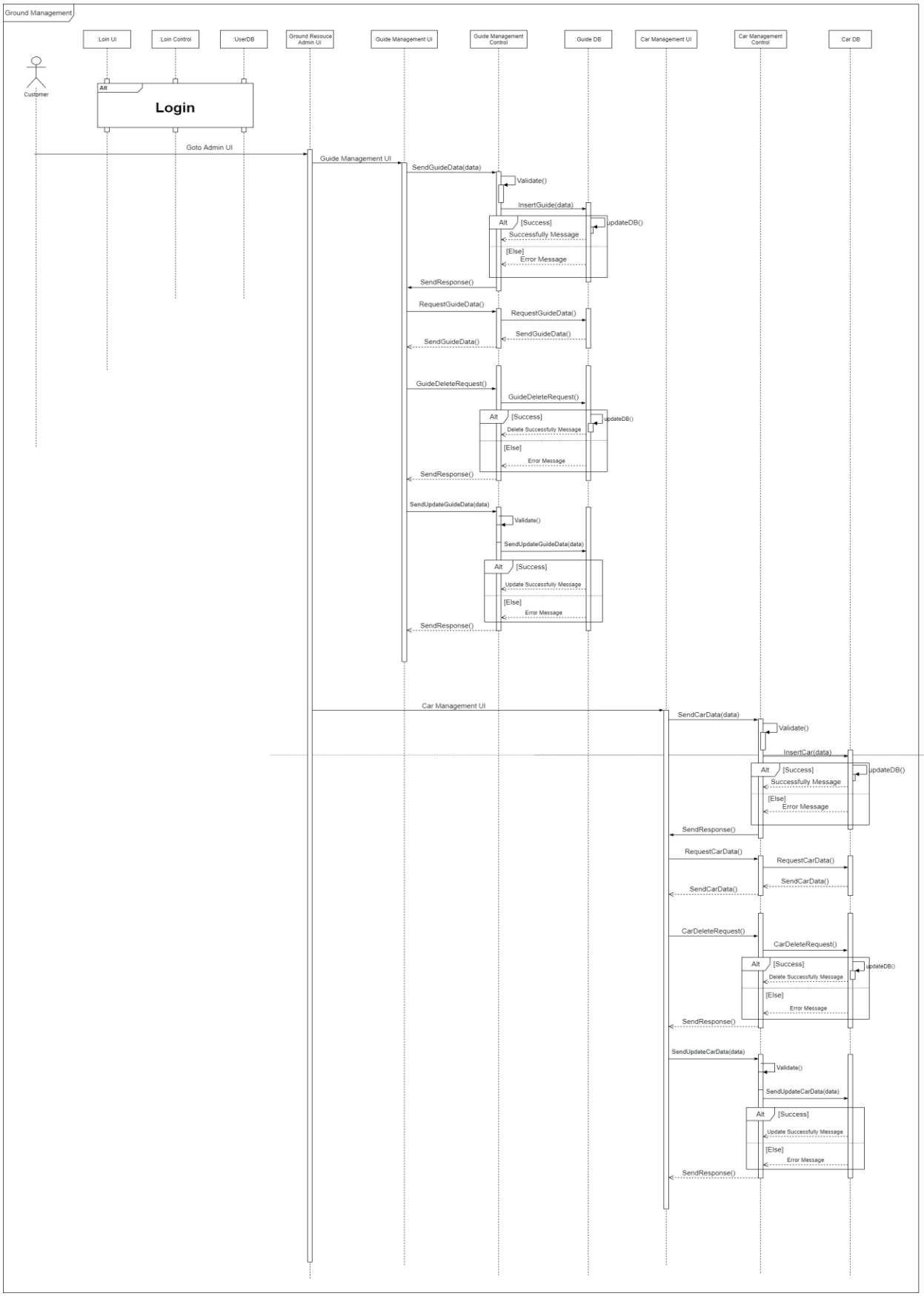
Reservation of Tickets Management (Admin side)

Figure 27



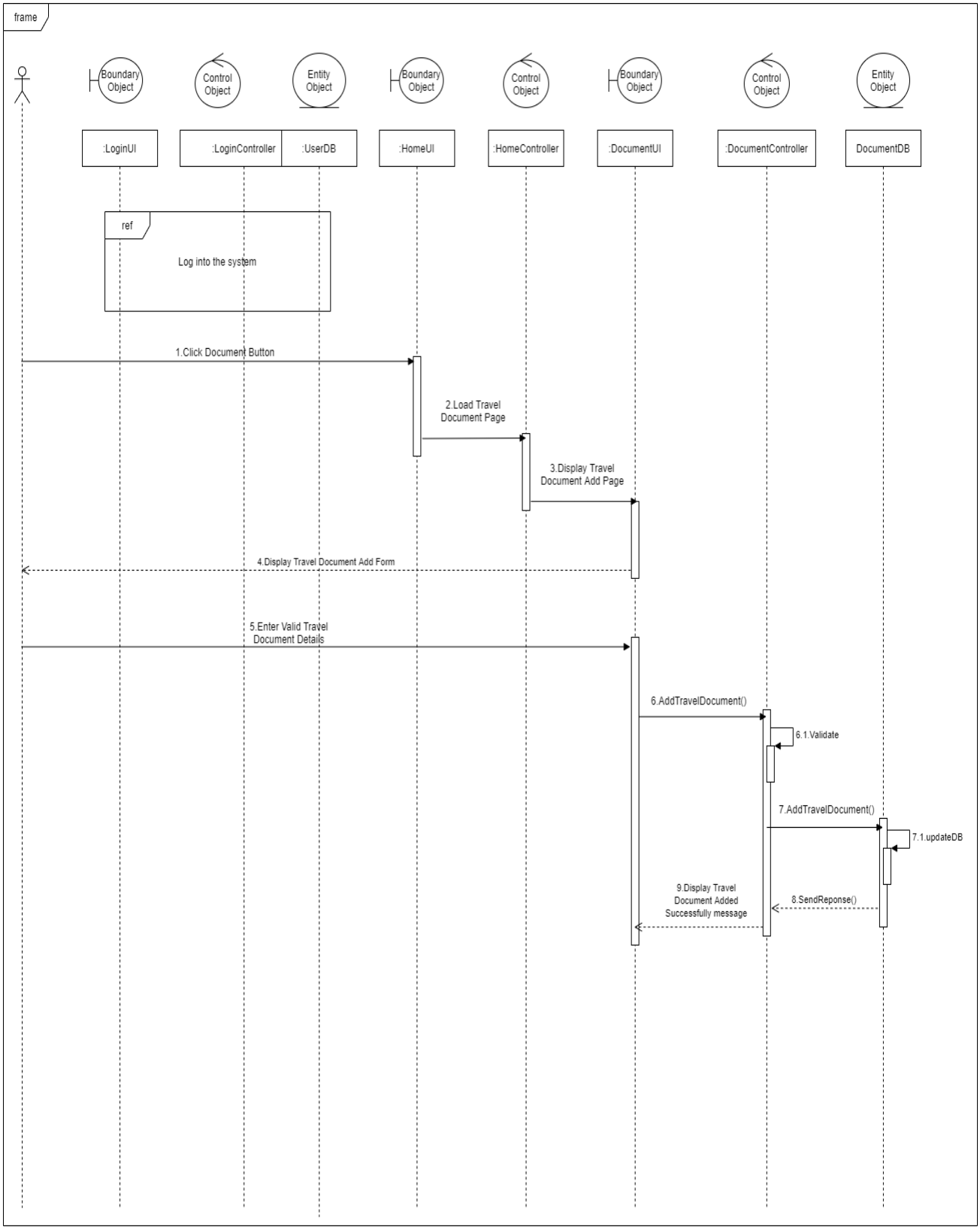
Reservation of Ground Resource Management

Figure 28



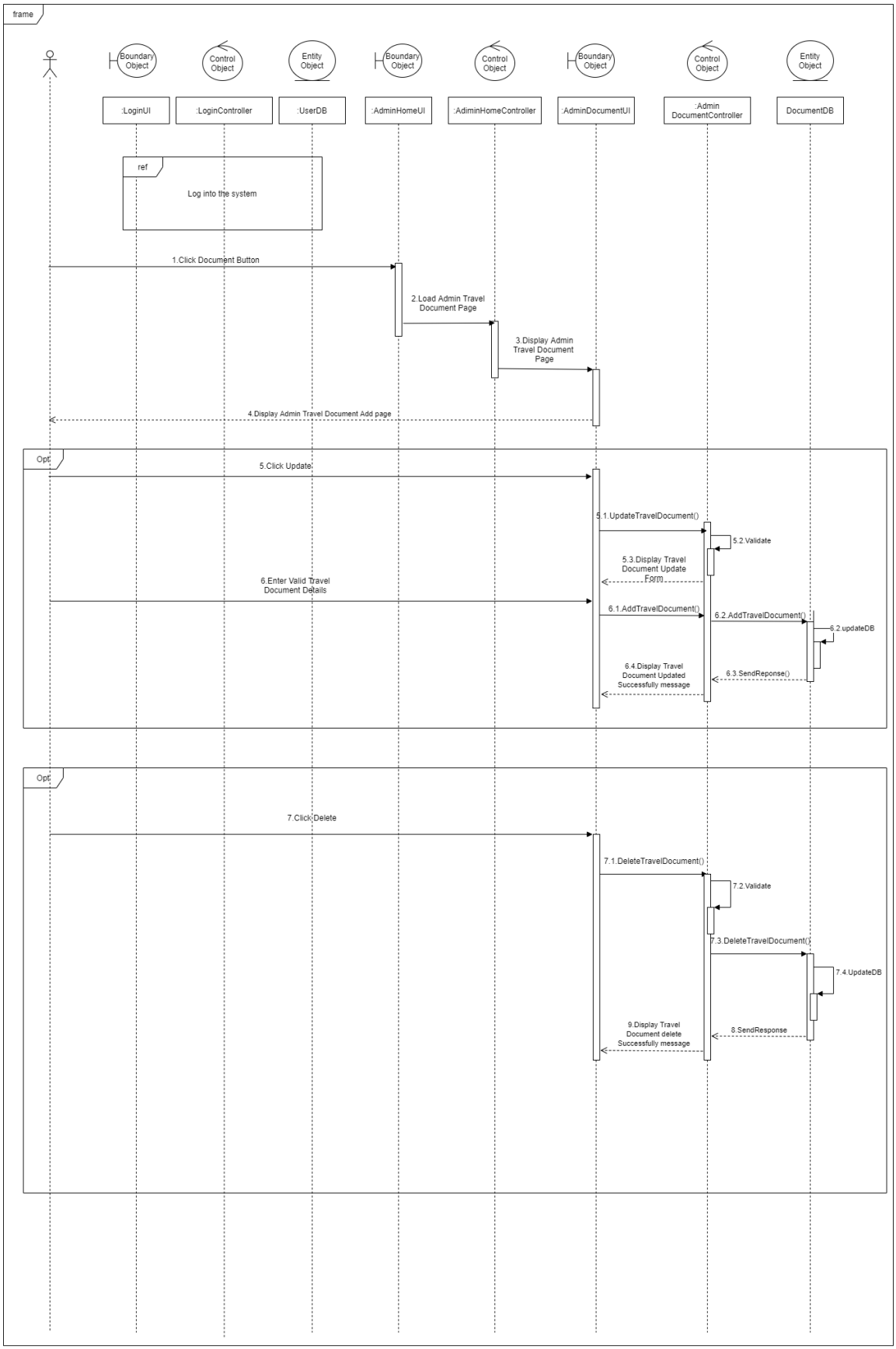
Reservation of Ground Resource Management

Figure 29



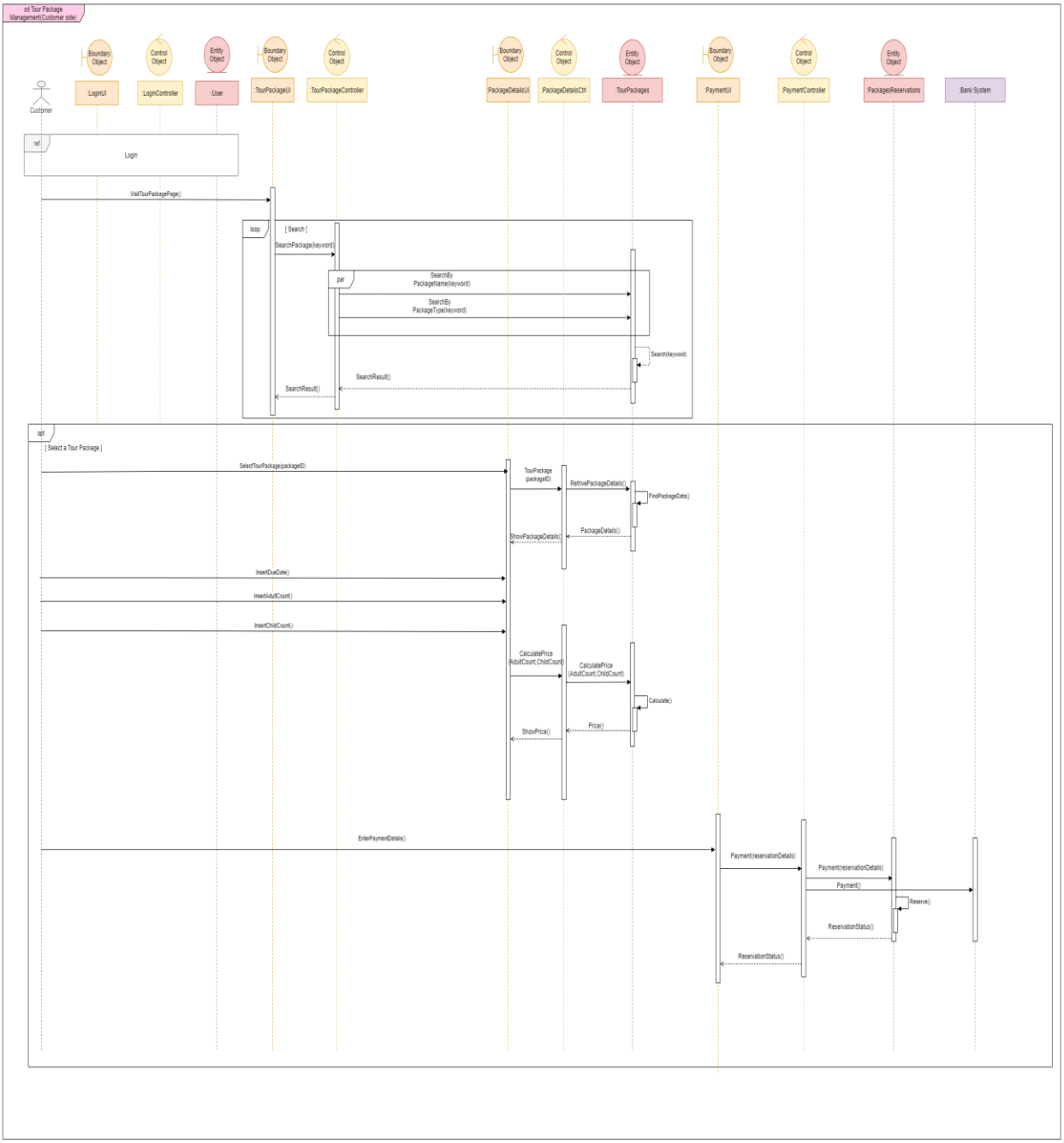
Travel Document Management (User side)

Figure 30



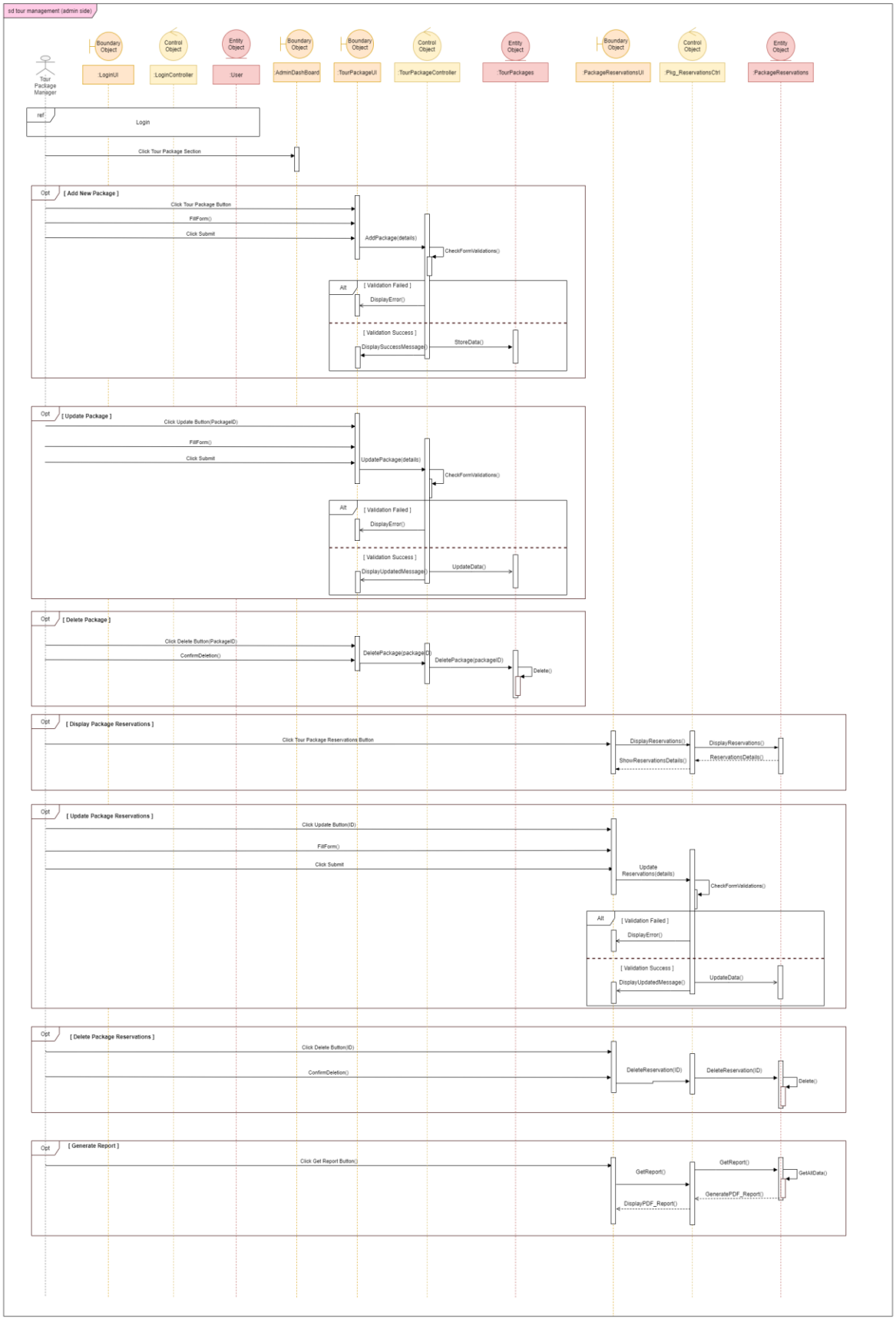
Travel Document Management (Admin side)

Figure 31



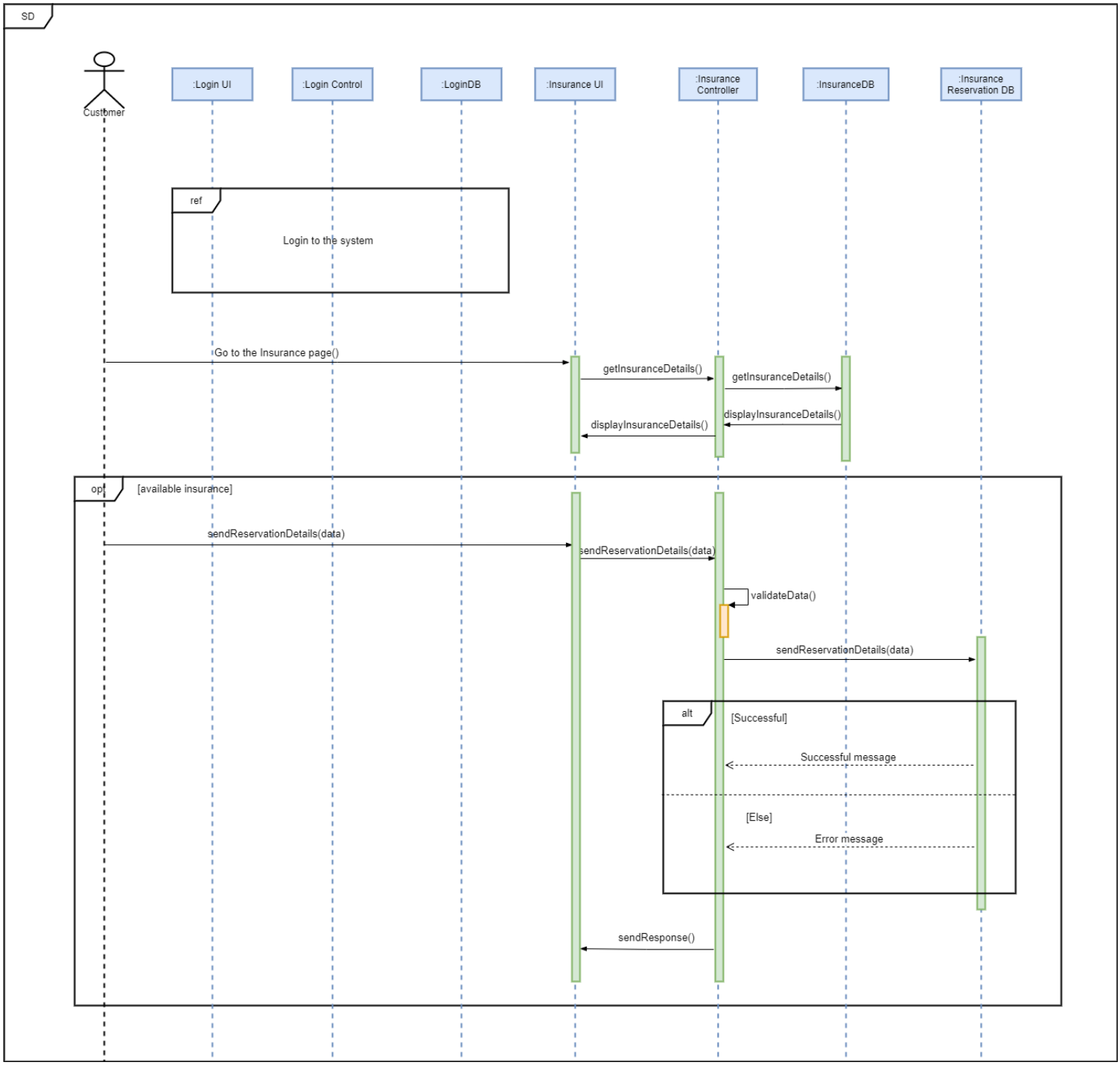
Tour Package (user side)

Figure 32



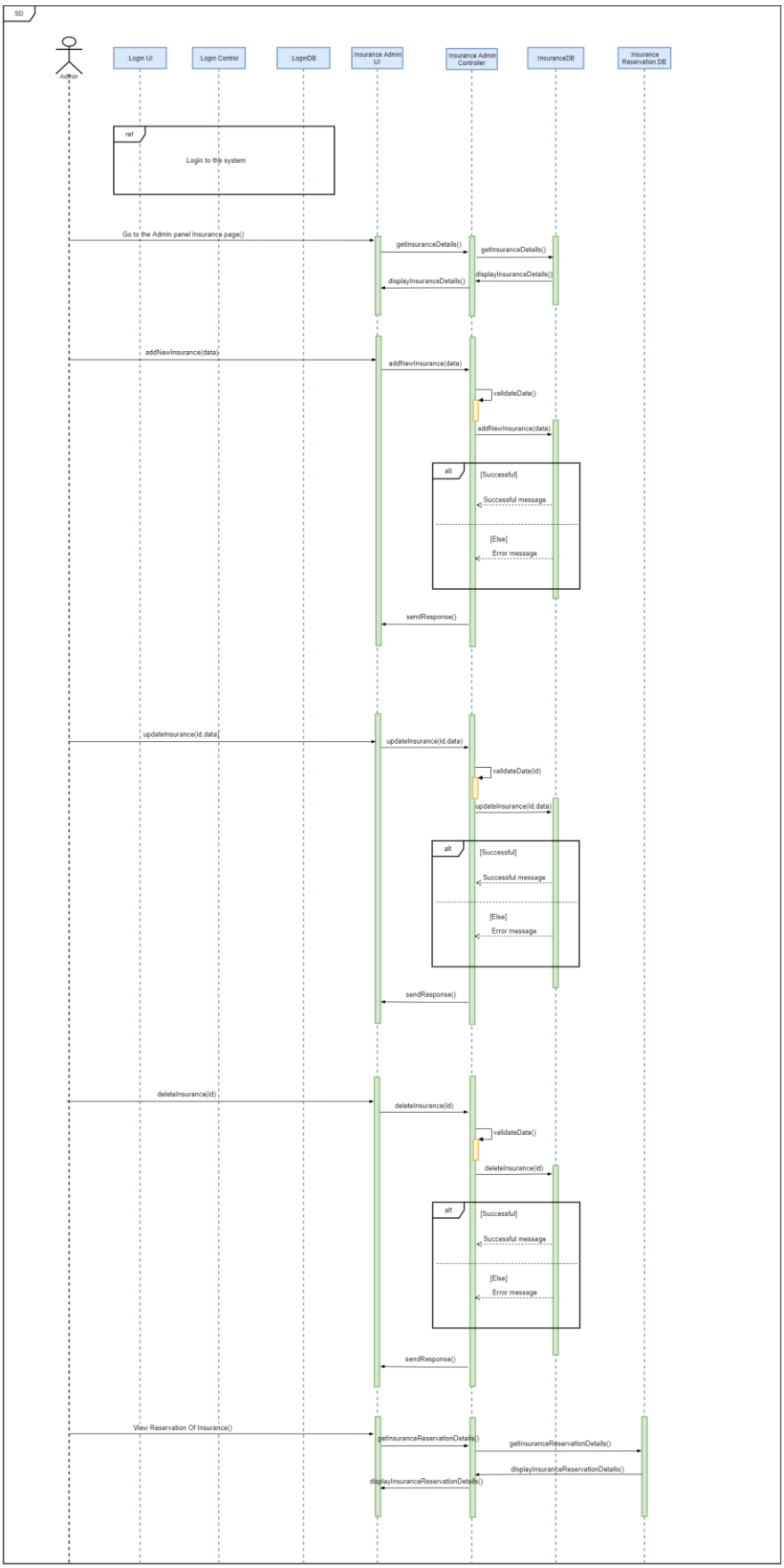
Tour Package Management (Admin side)

Figure 33



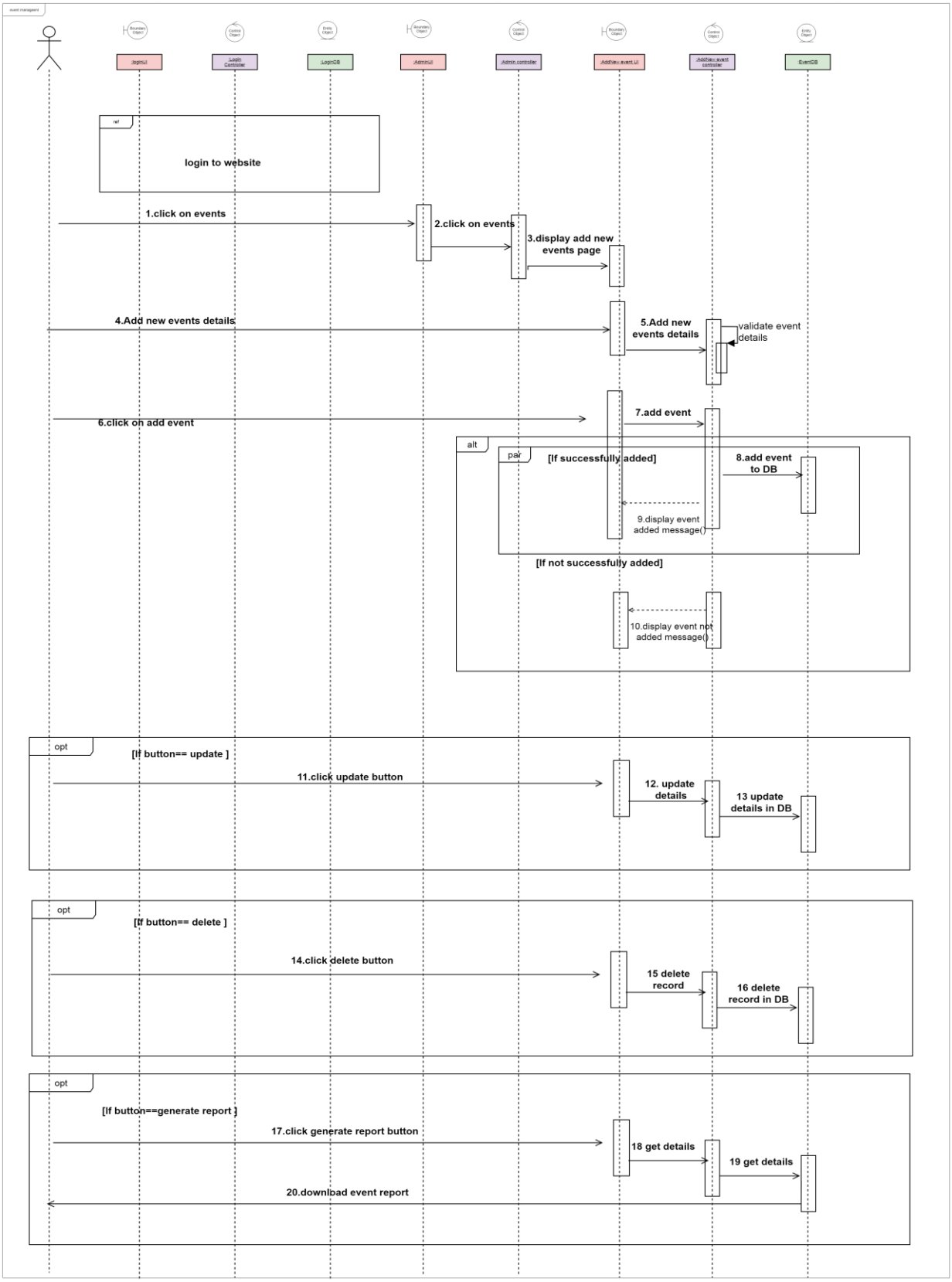
Travel Insurance Management (Admin side)

Figure 34



Travel Insurance Management (Admin side)

Figure 35



Event Management

Figure 36

2.2.3 User Interfaces

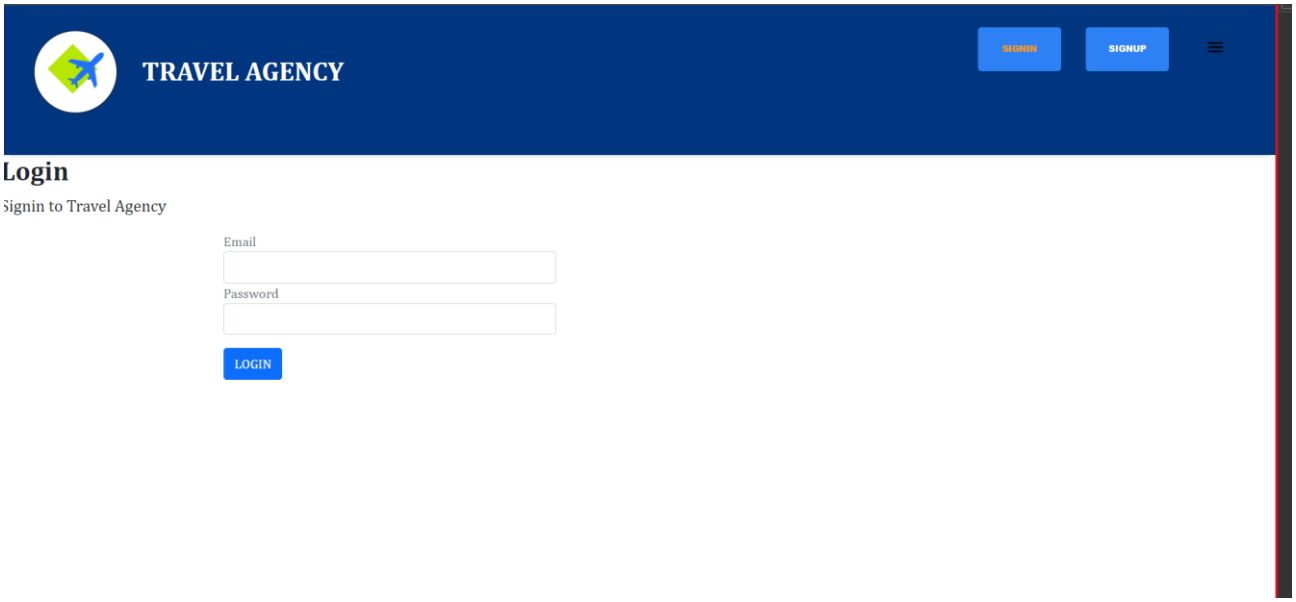


Figure 37 – Login page

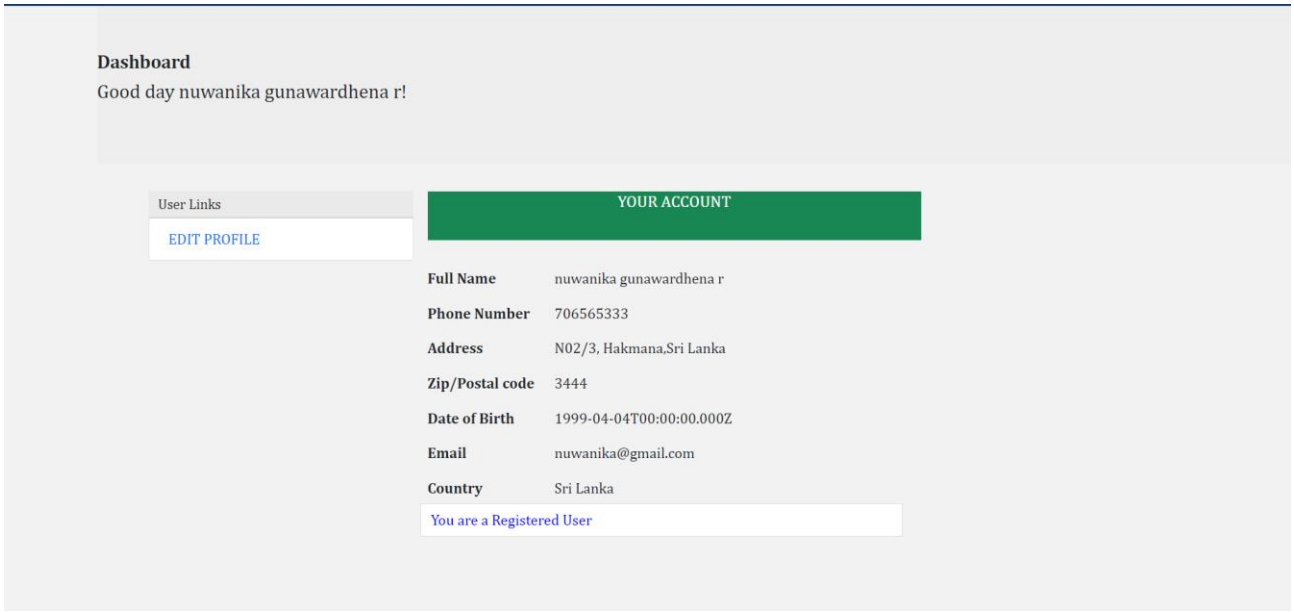


Figure 38 – User Details page

Figure 39 – User Details Edit page

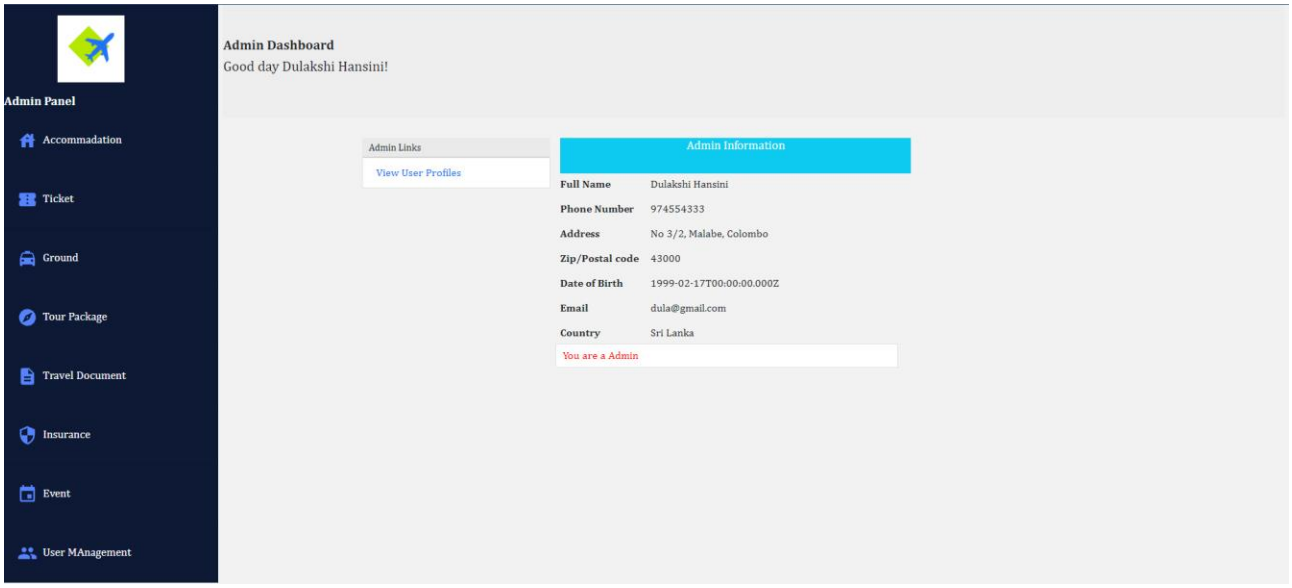


Figure 40– Admin Dashboard page

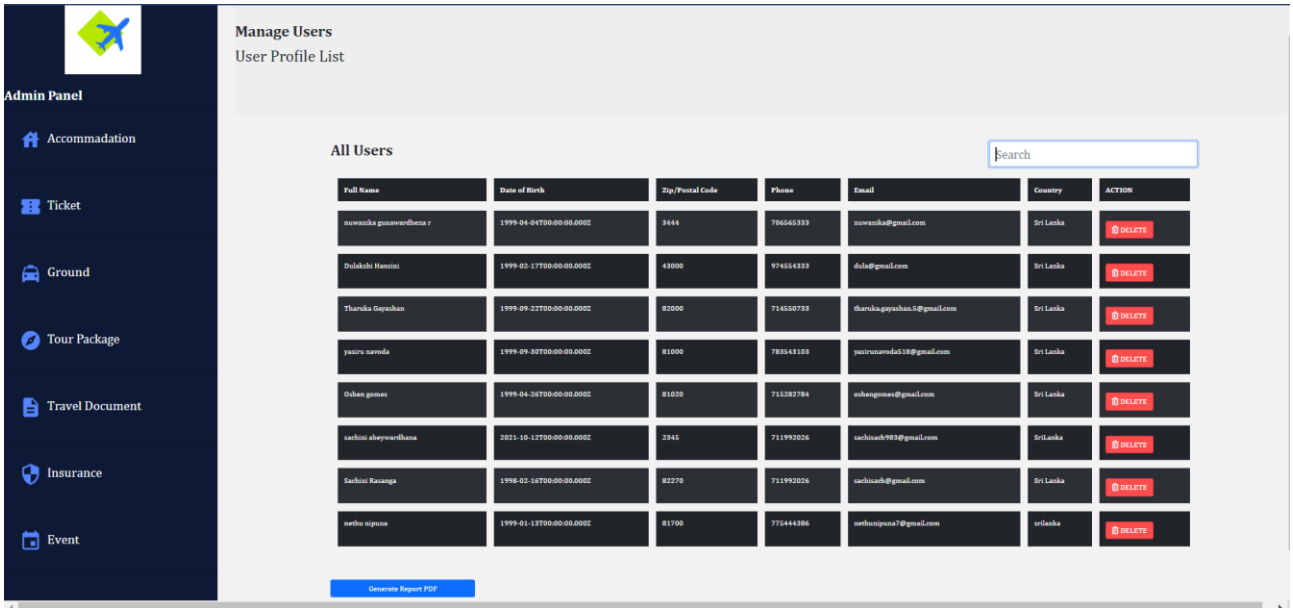


Figure 41 – Admin User Management page

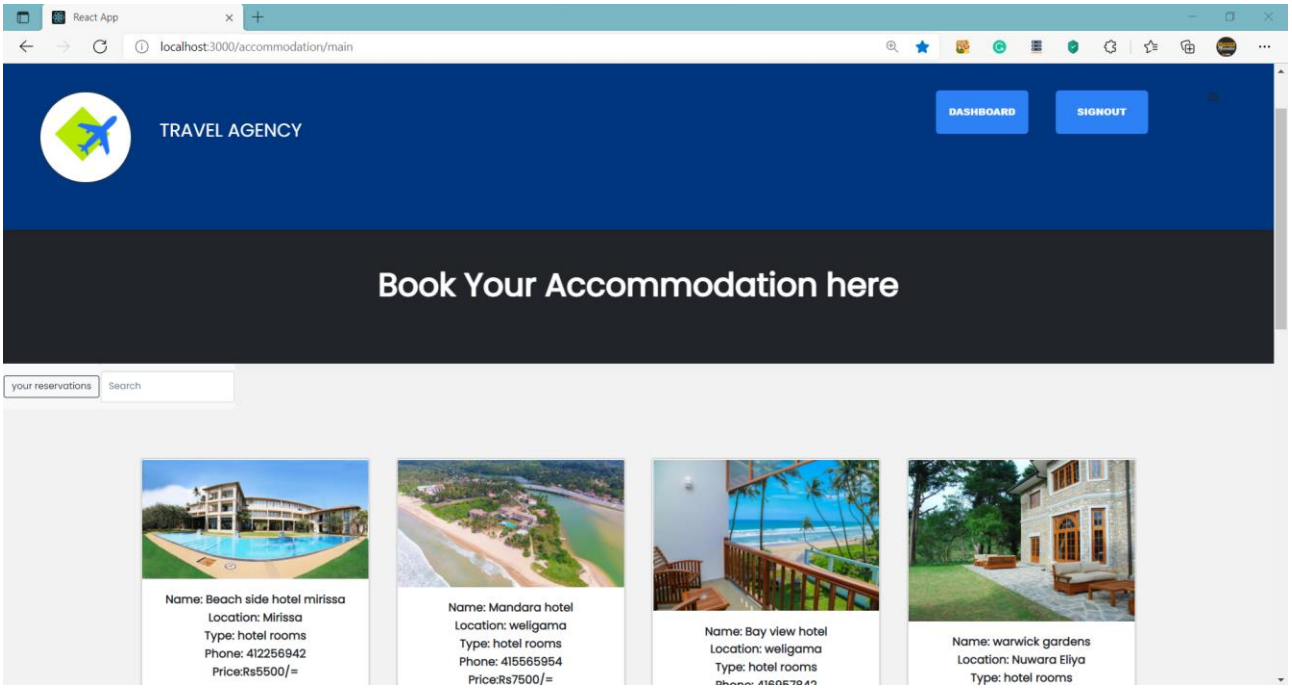


Figure 42 – Accommodation view page

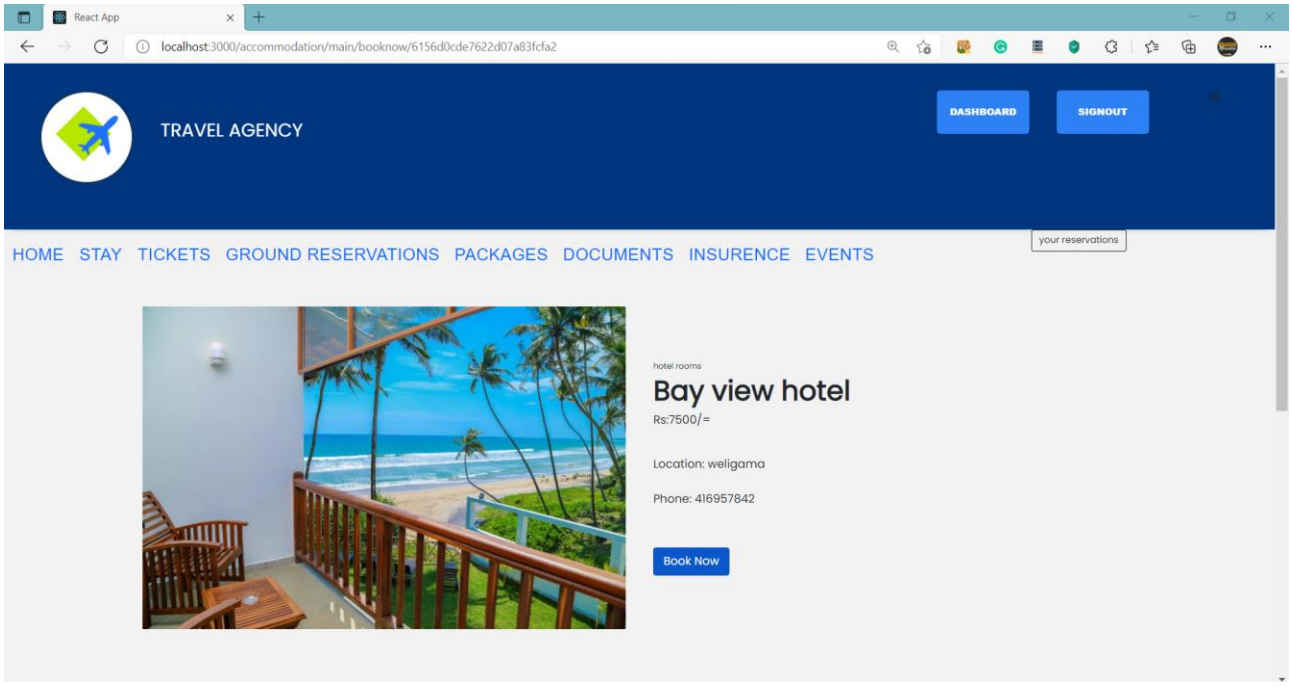


Figure 43 – Selected Accommodation details view page

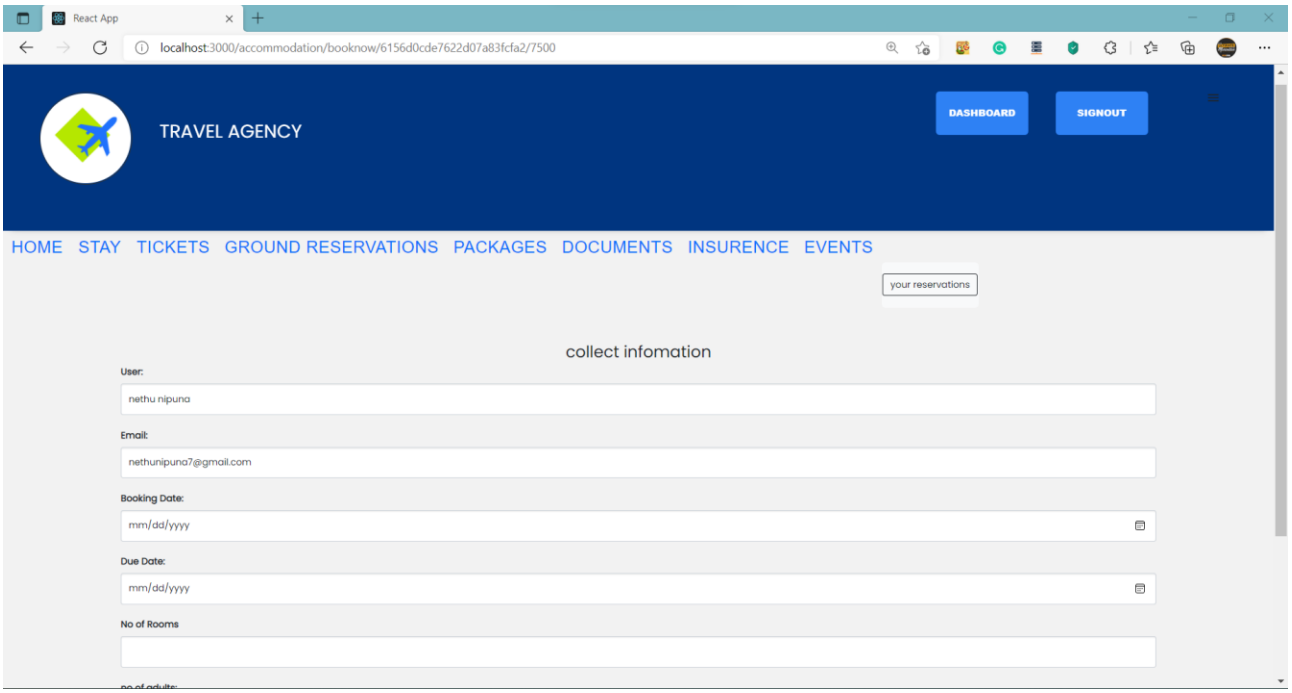


Figure 44 – User details for Accommodation page

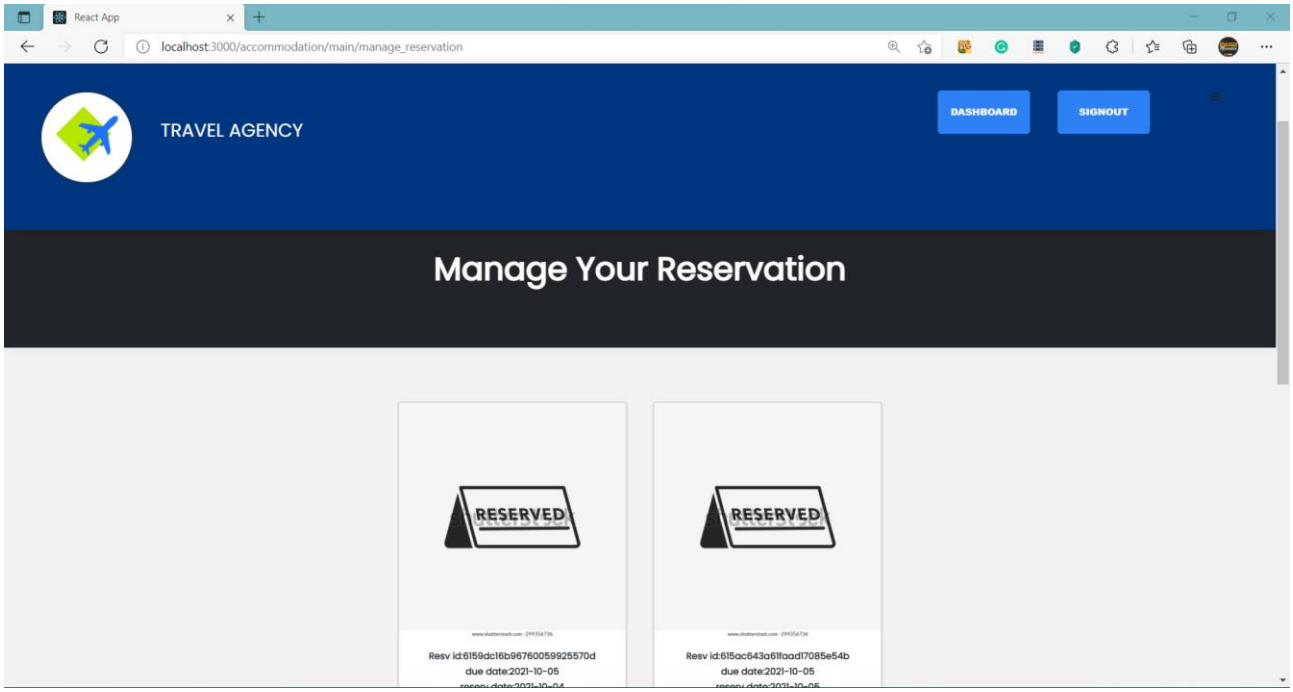


Figure 45 – User's Reservation of Accommodation page

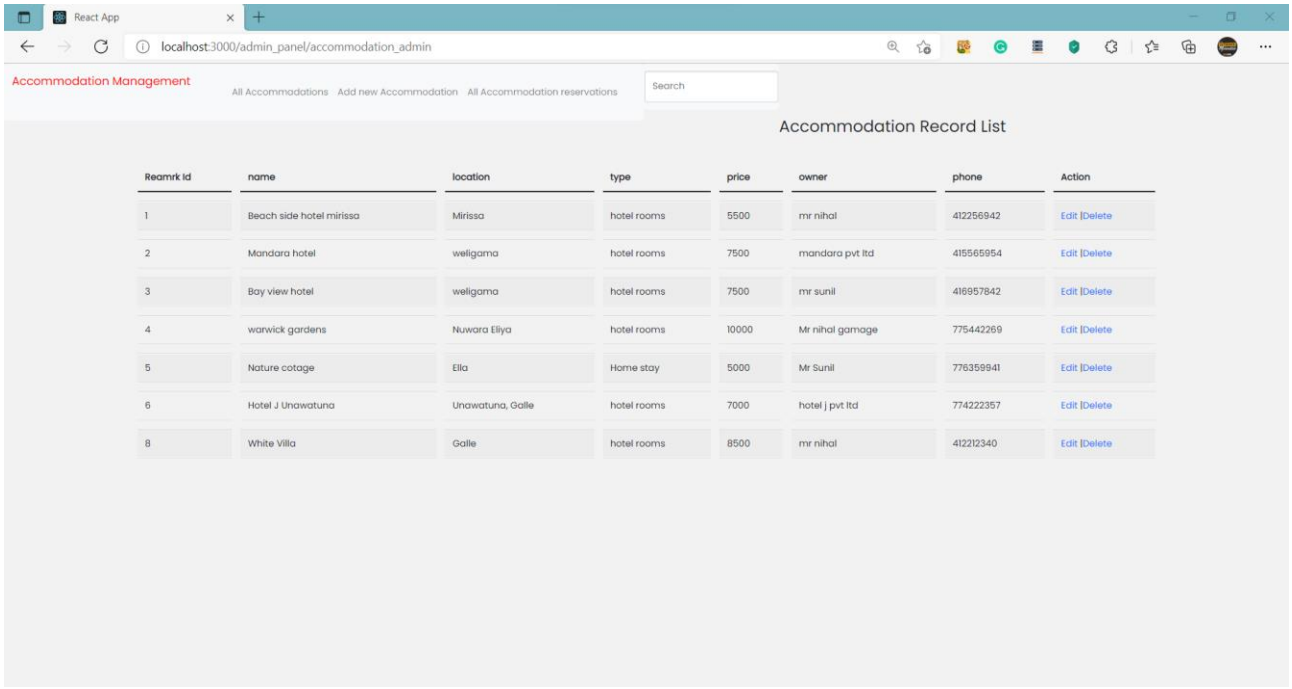


Figure 46 – List of Accommodation page

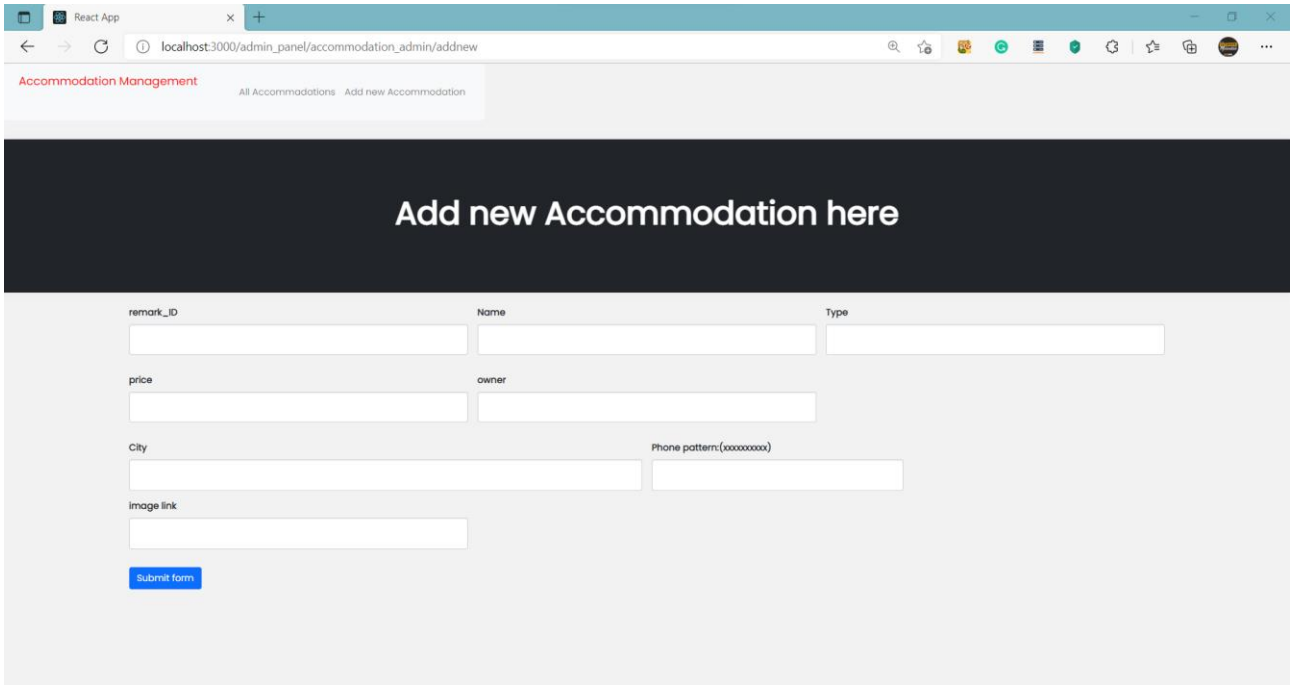


Figure 47 – Insert new Accommodation page

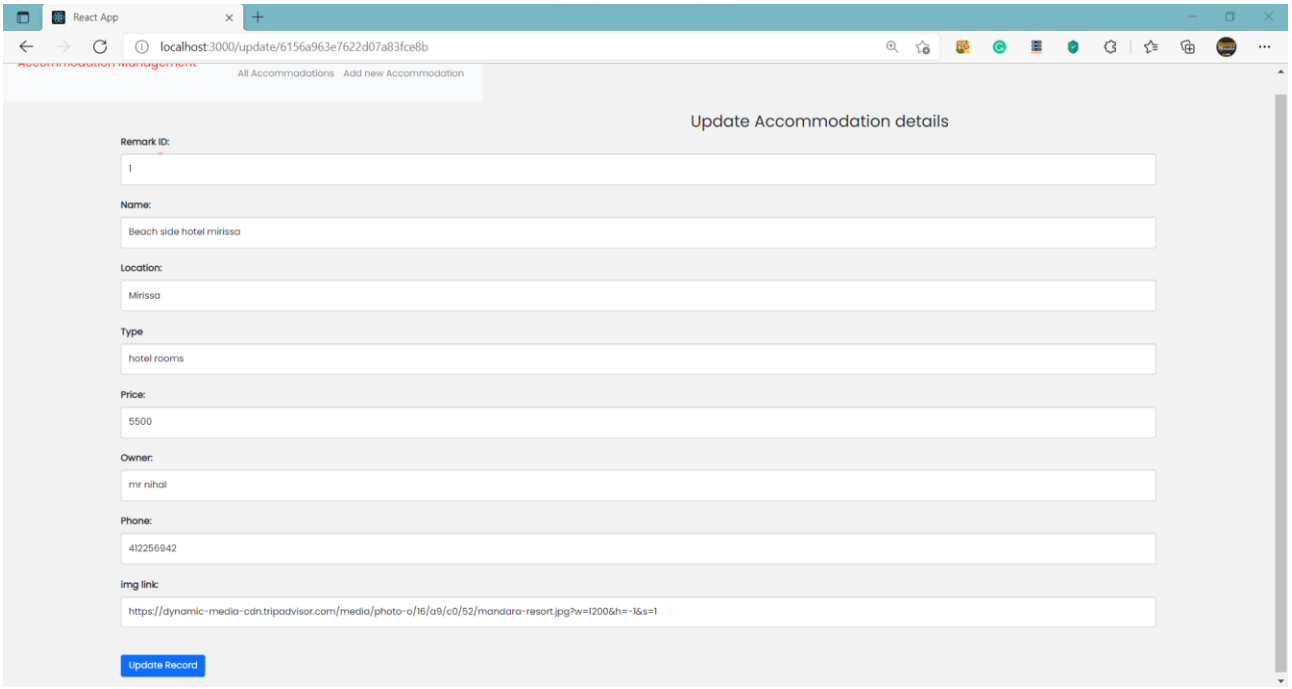


Figure 48 – Update Accommodation page

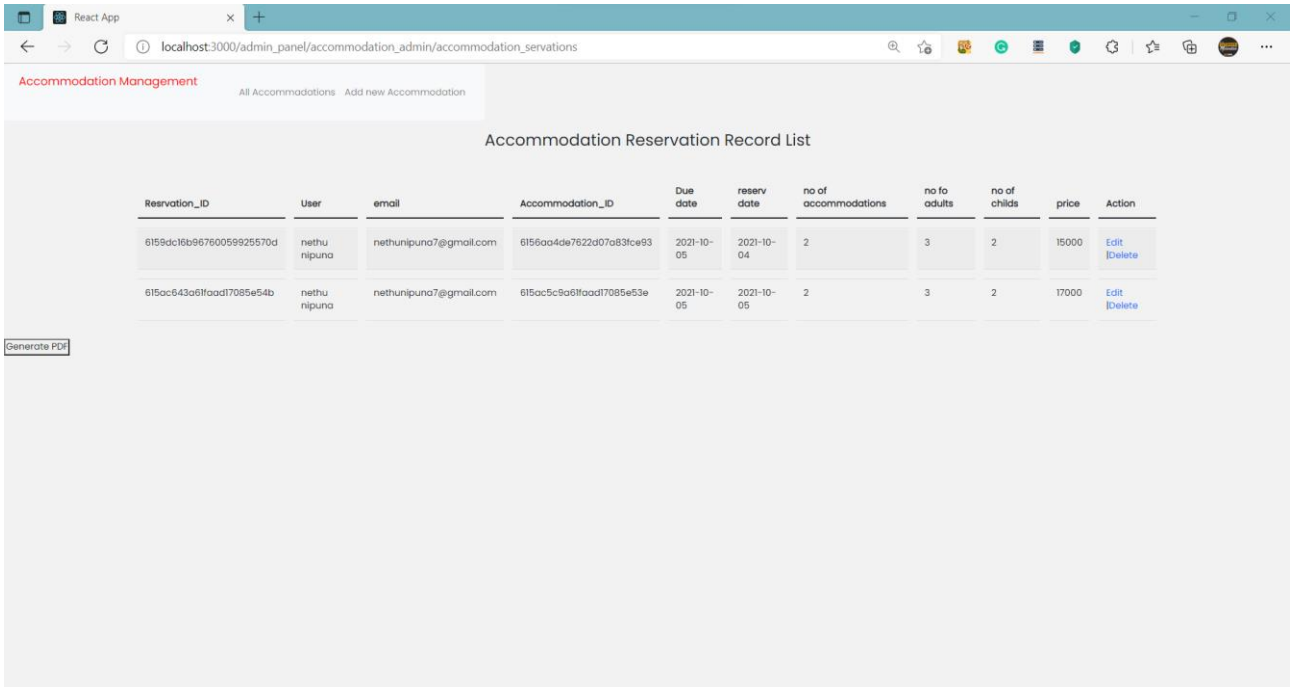


Figure 49 – Reservation List of Accommodation page

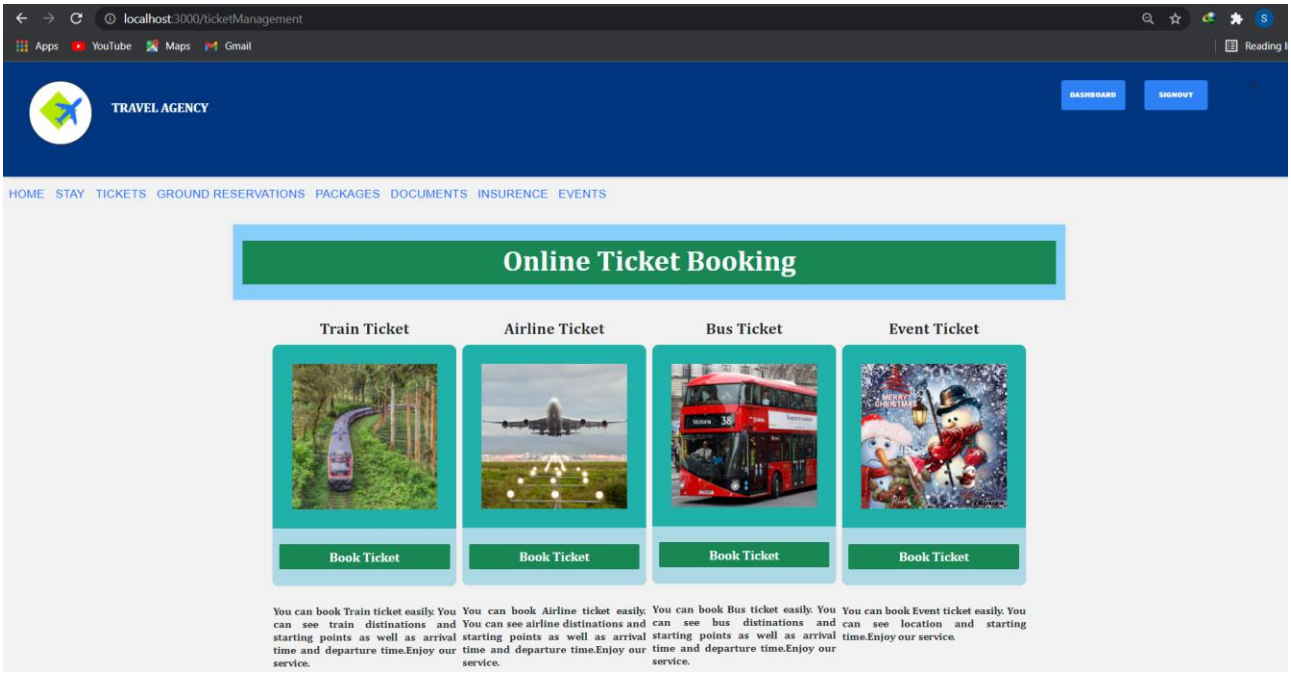


Figure 50 – Ticket Category page

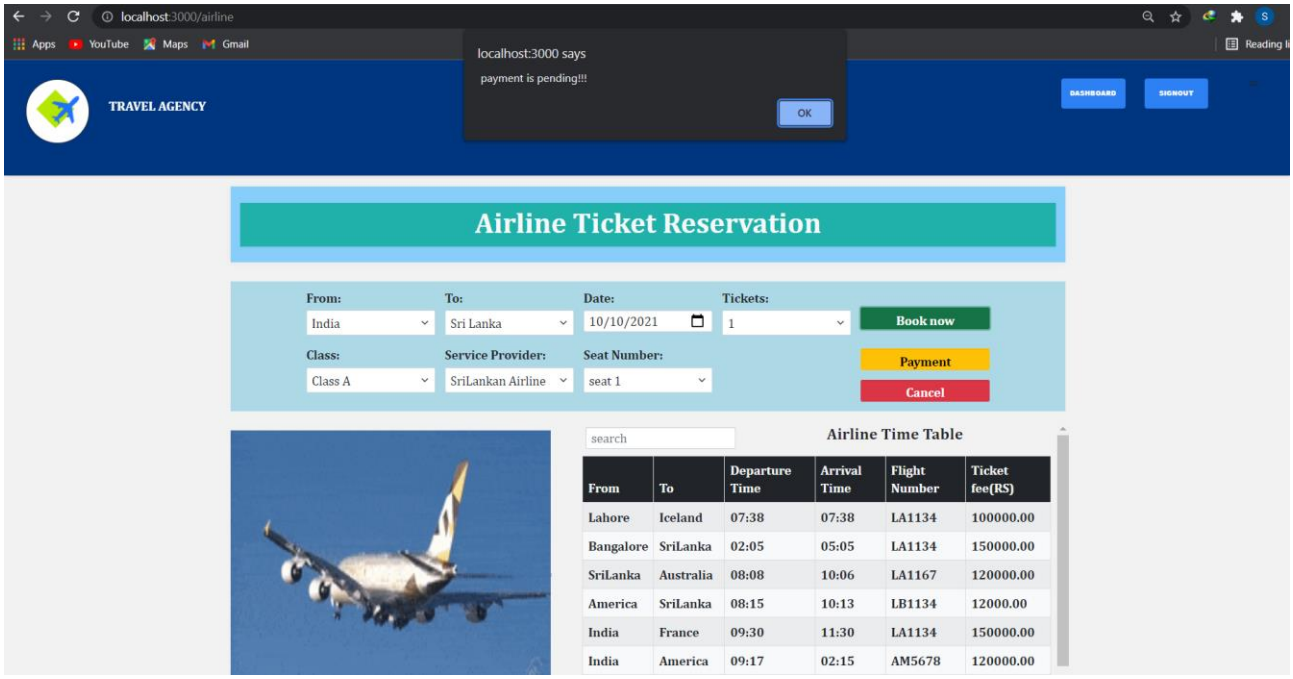


Figure 51 – Airplane Ticket Reservation page

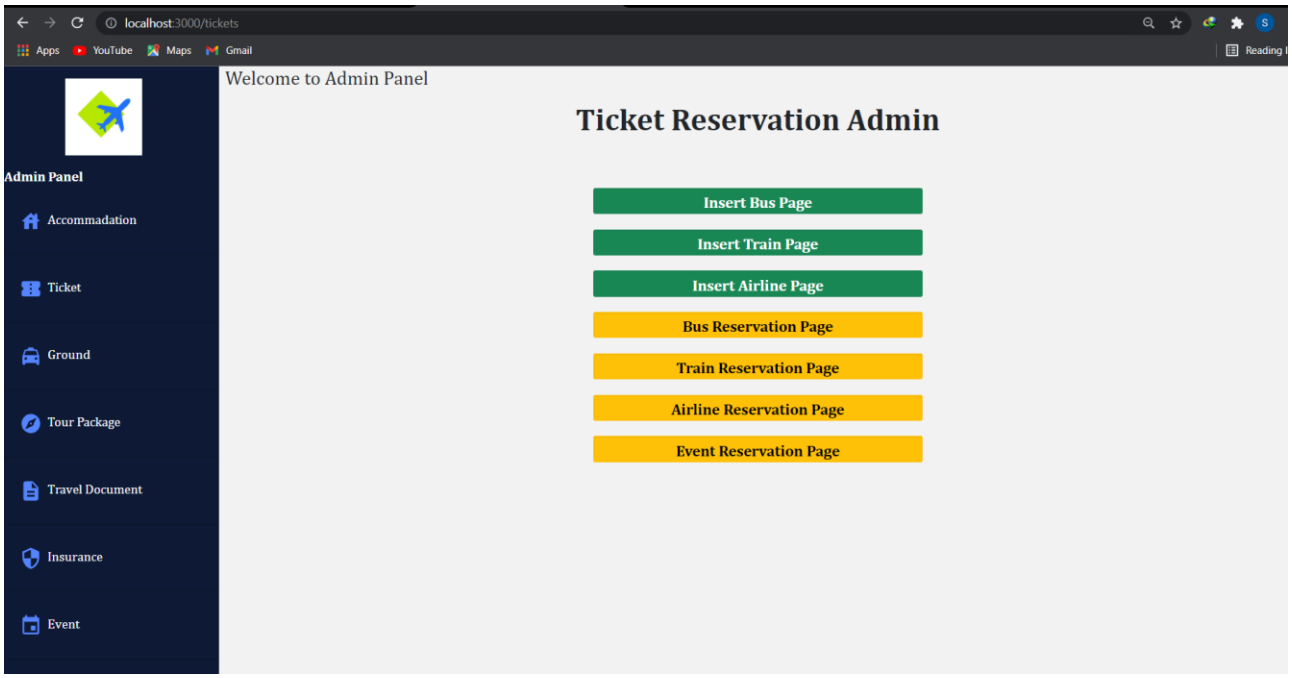


Figure 52 – Ticket Reservation Admin page

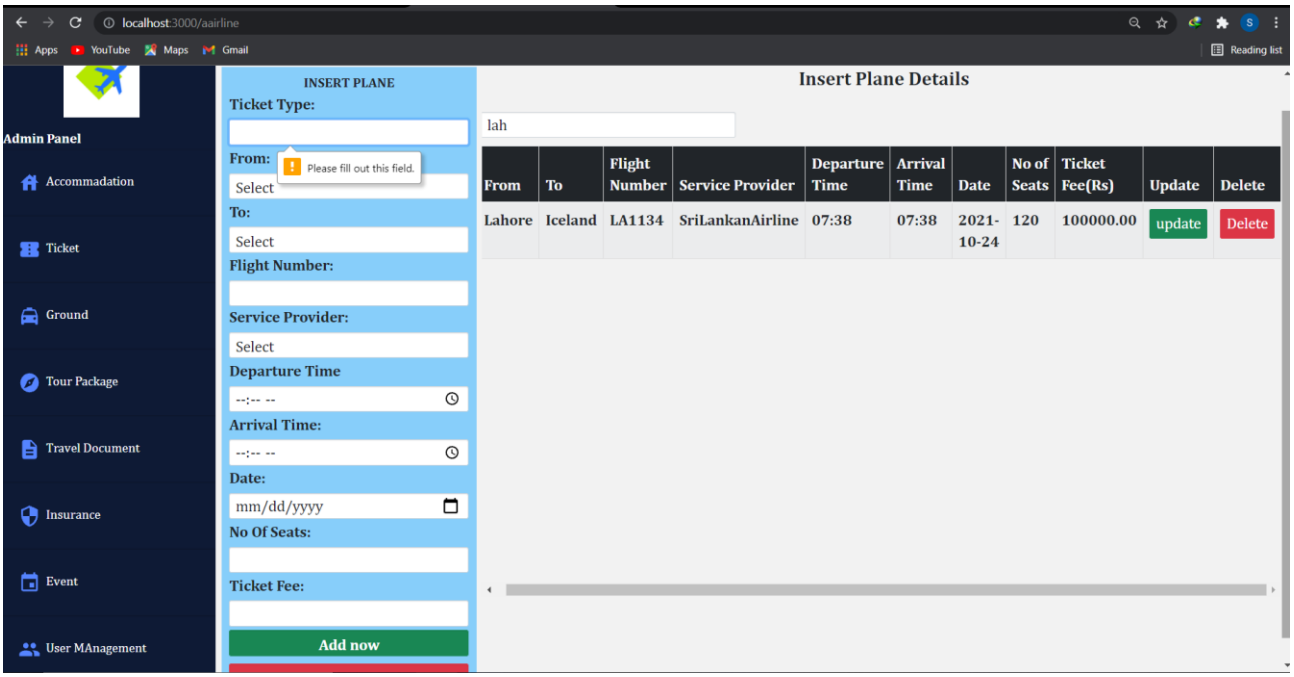


Figure 53 – Airplane Ticket insert Admin page

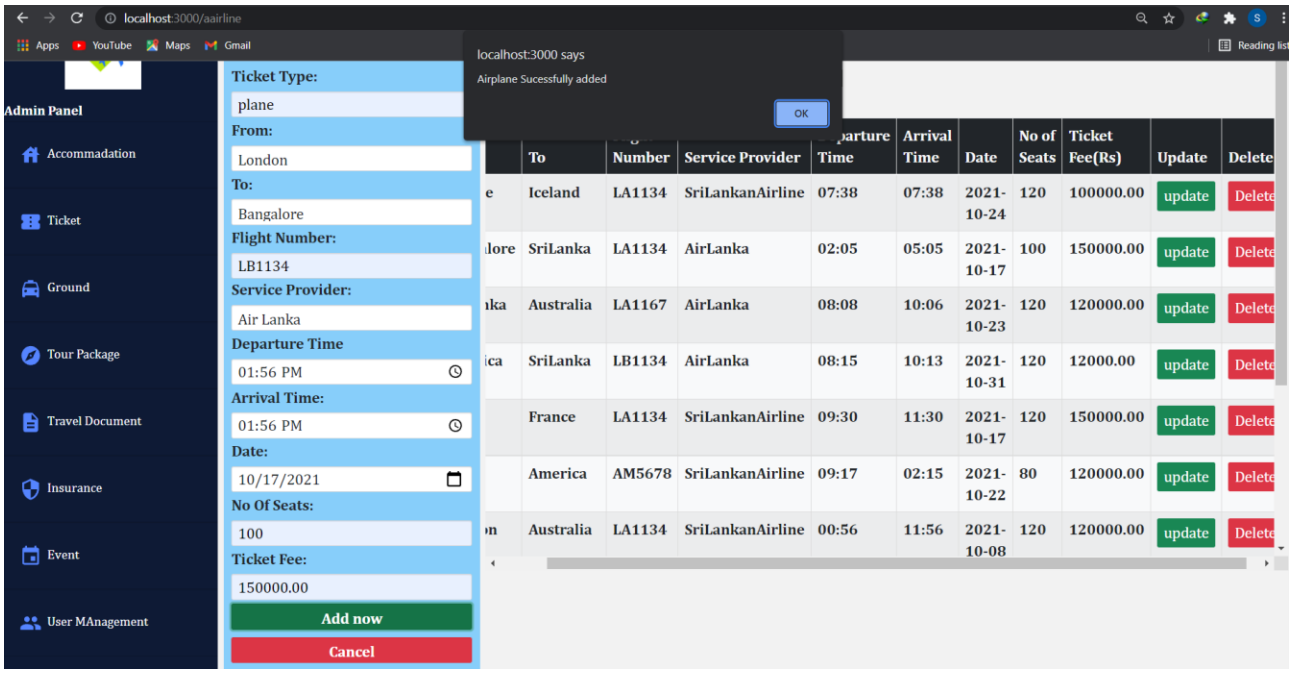


Figure 54 – Airplane Tickets view Admin page

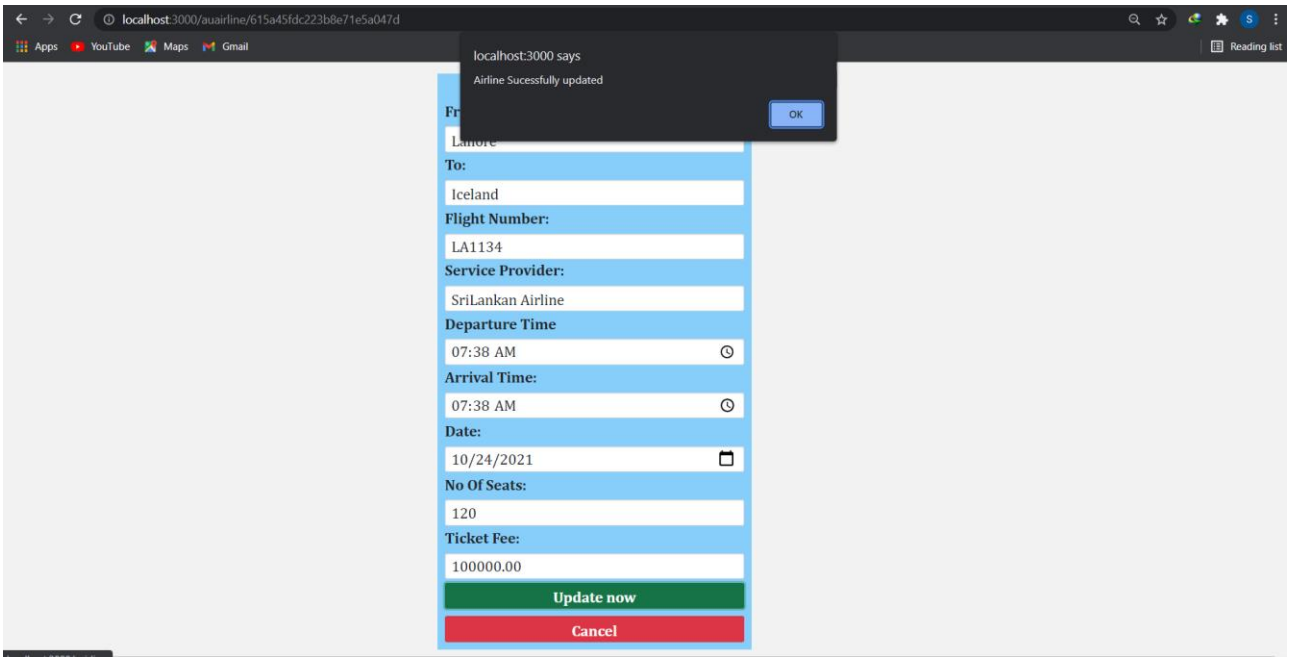


Figure 55 – Airplane Tickets Update Admin page

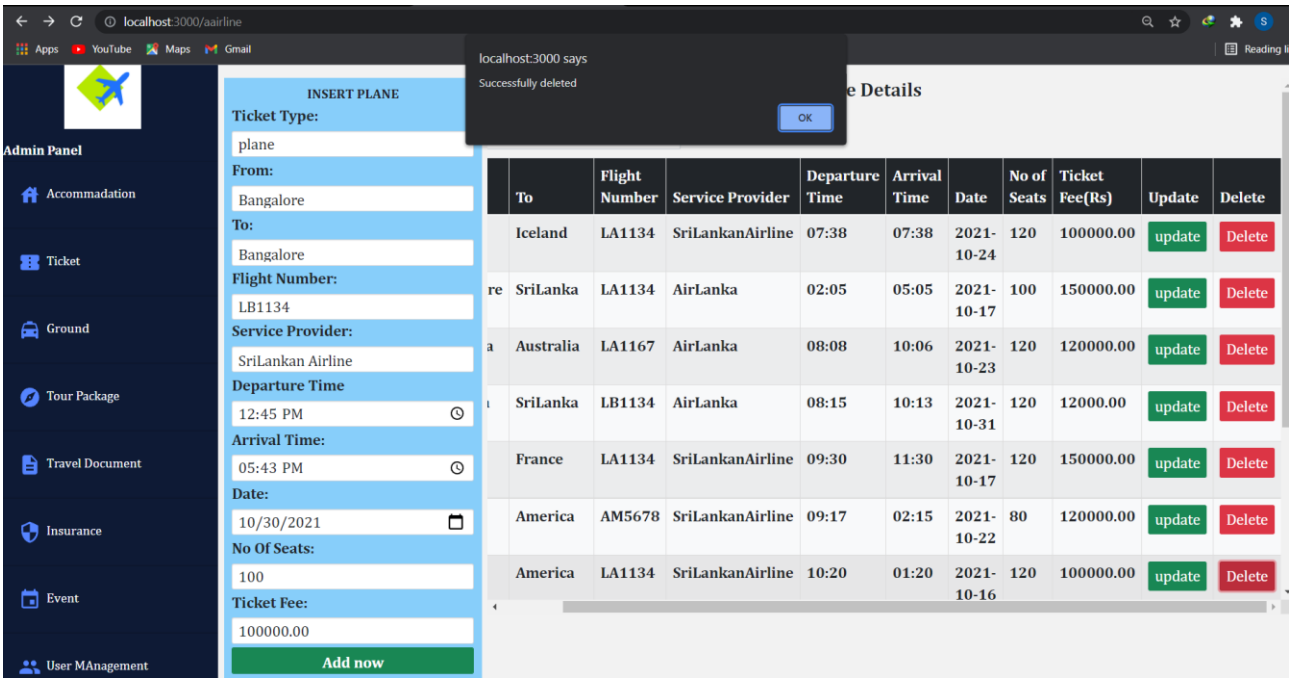


Figure 56 – Airplane Tickets Delete Admin page

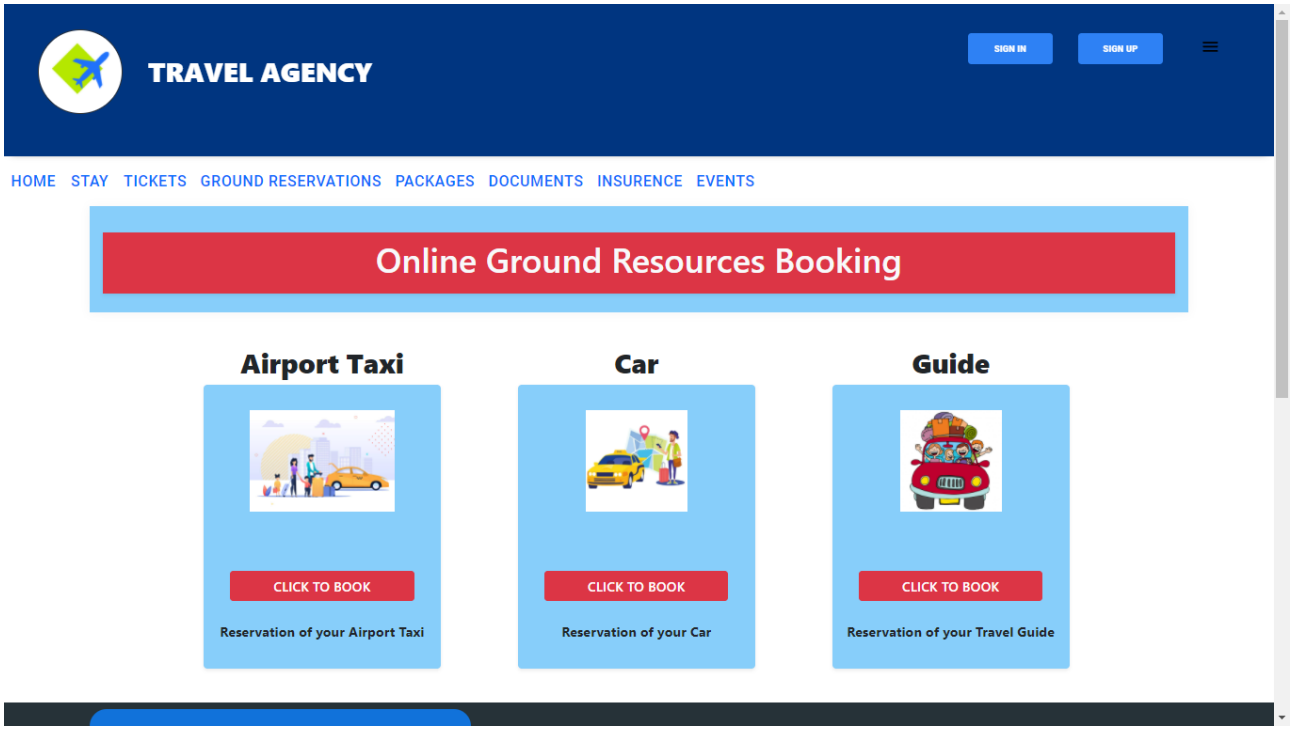


Figure 57 – Ground Resource category page

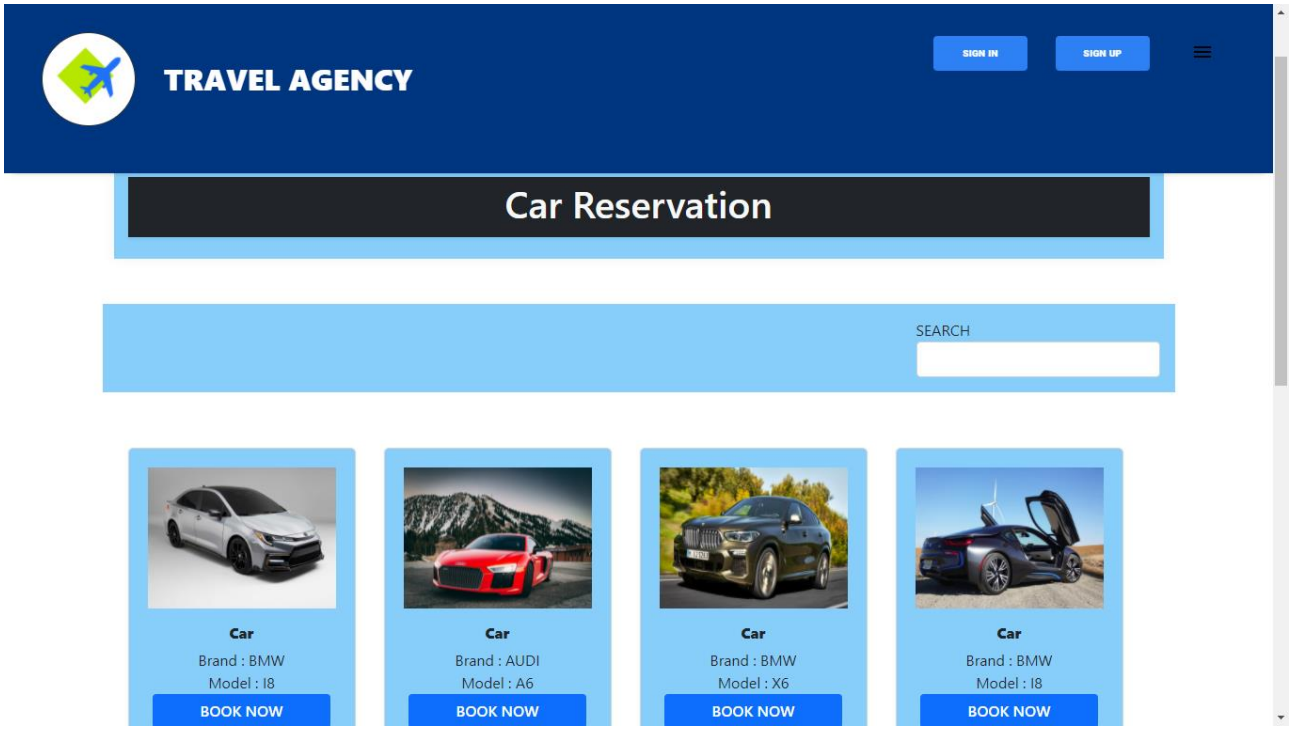


Figure 58 – (Ground Resource) Car reservation category page

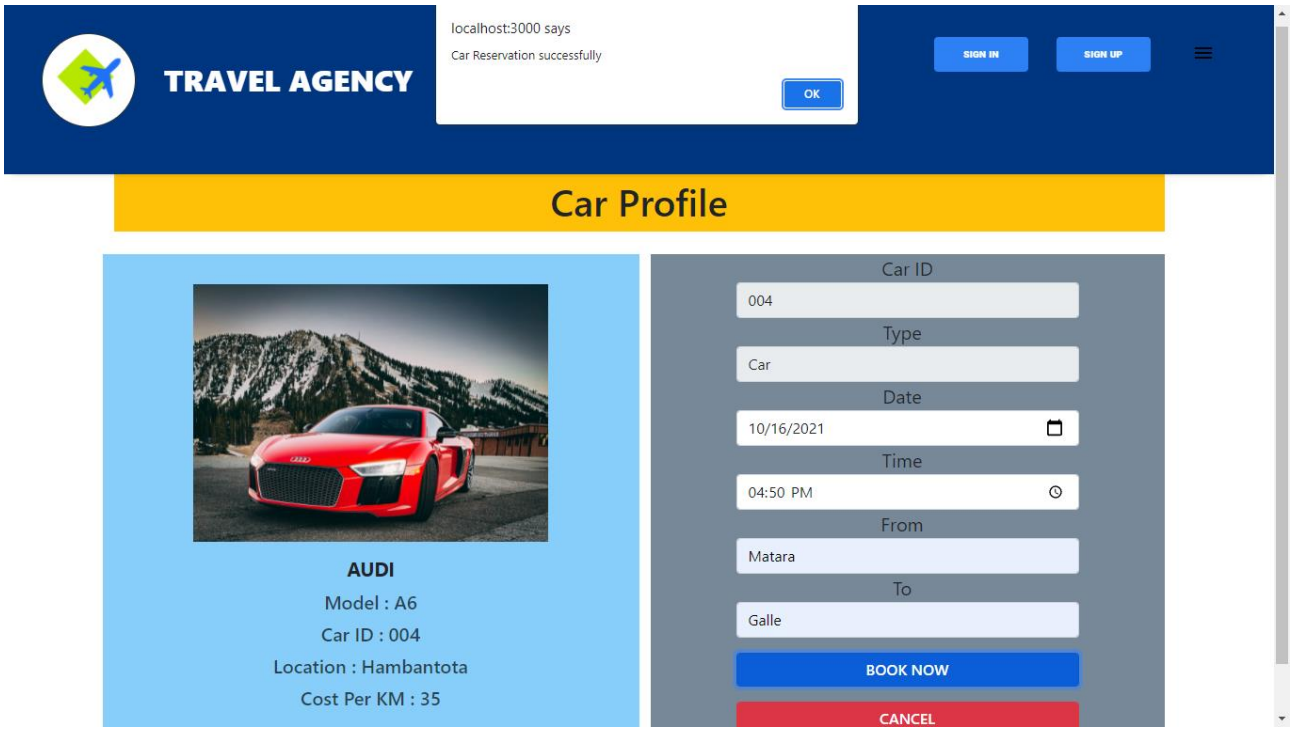


Figure 59 – (Ground Resource) Car reservation page

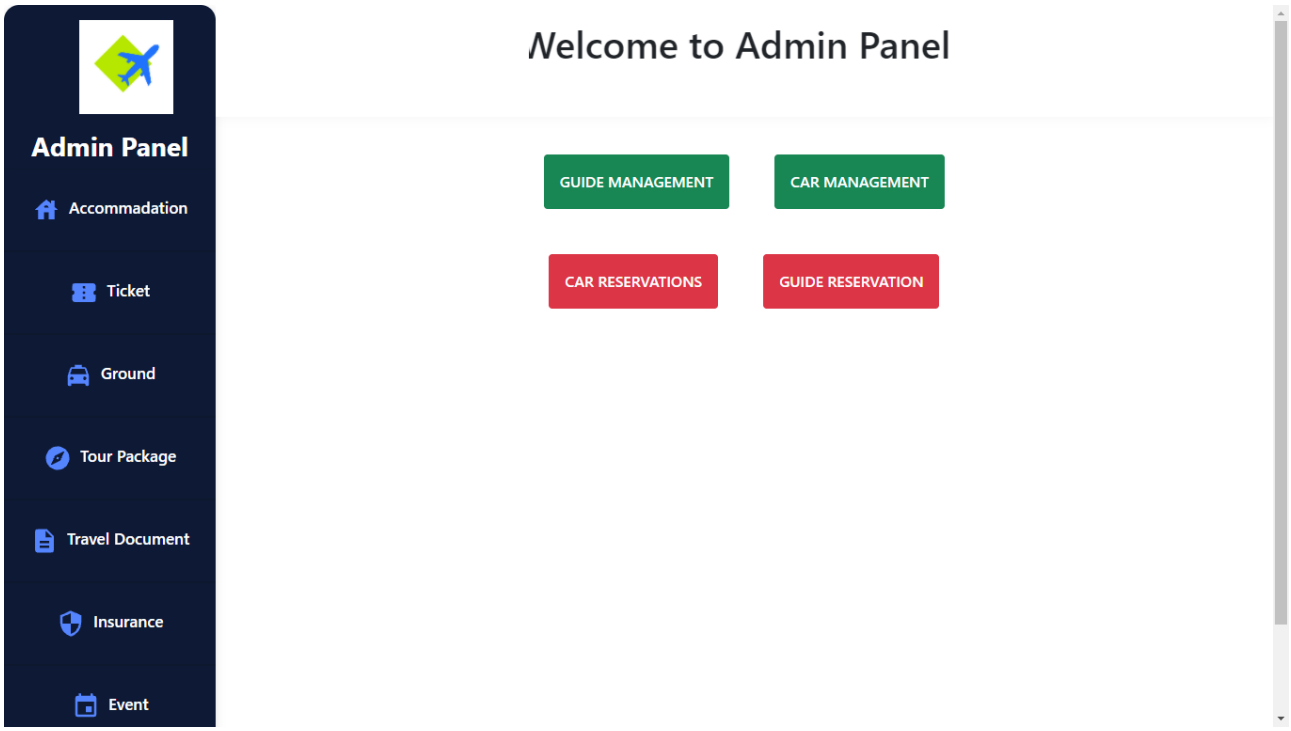


Figure 60 – Ground Resource Admin page

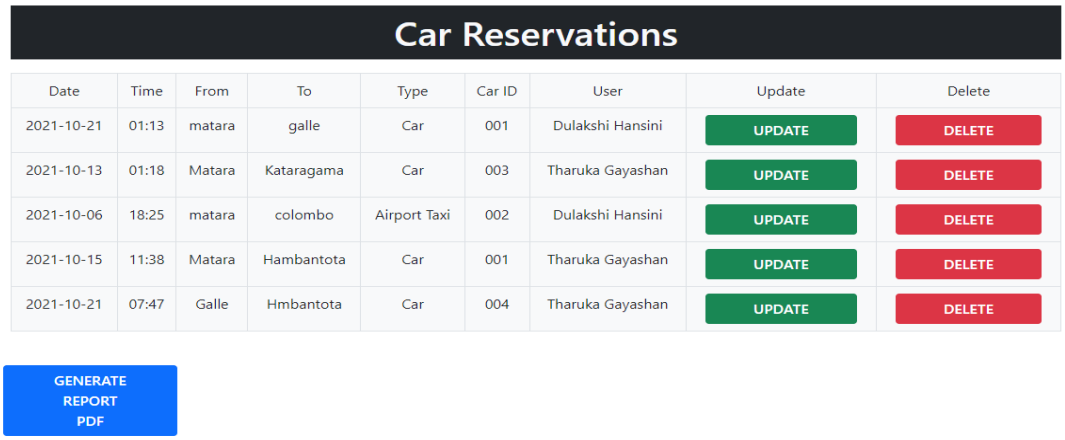


Figure 61 – (Ground Resource) Car reservation list view Admin page

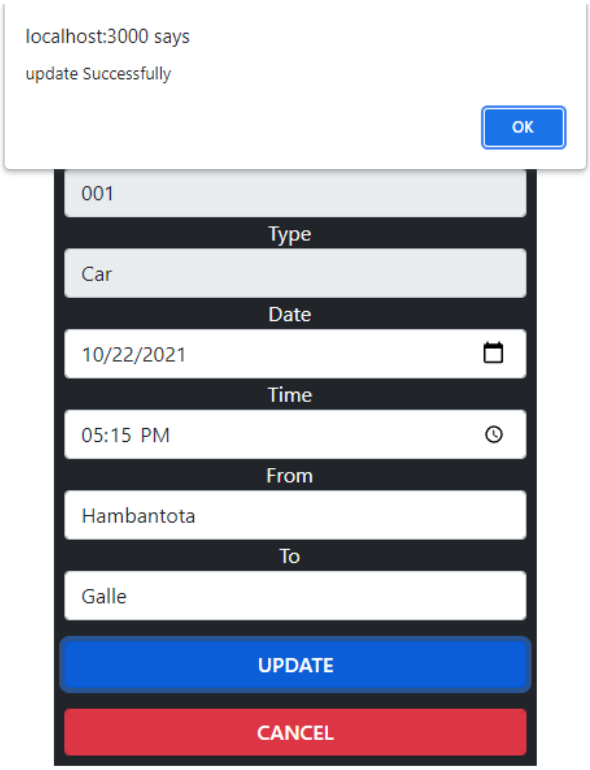


Figure 62 – (Ground Resource) Car reservation Update Admin page

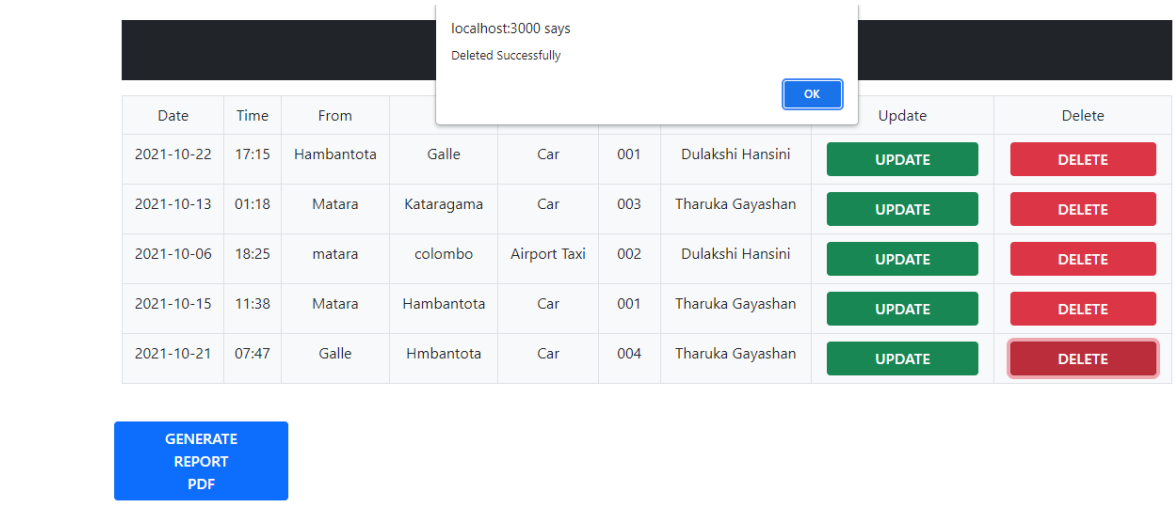


Figure 63 – (Ground Resource) Car reservation Delete Admin page

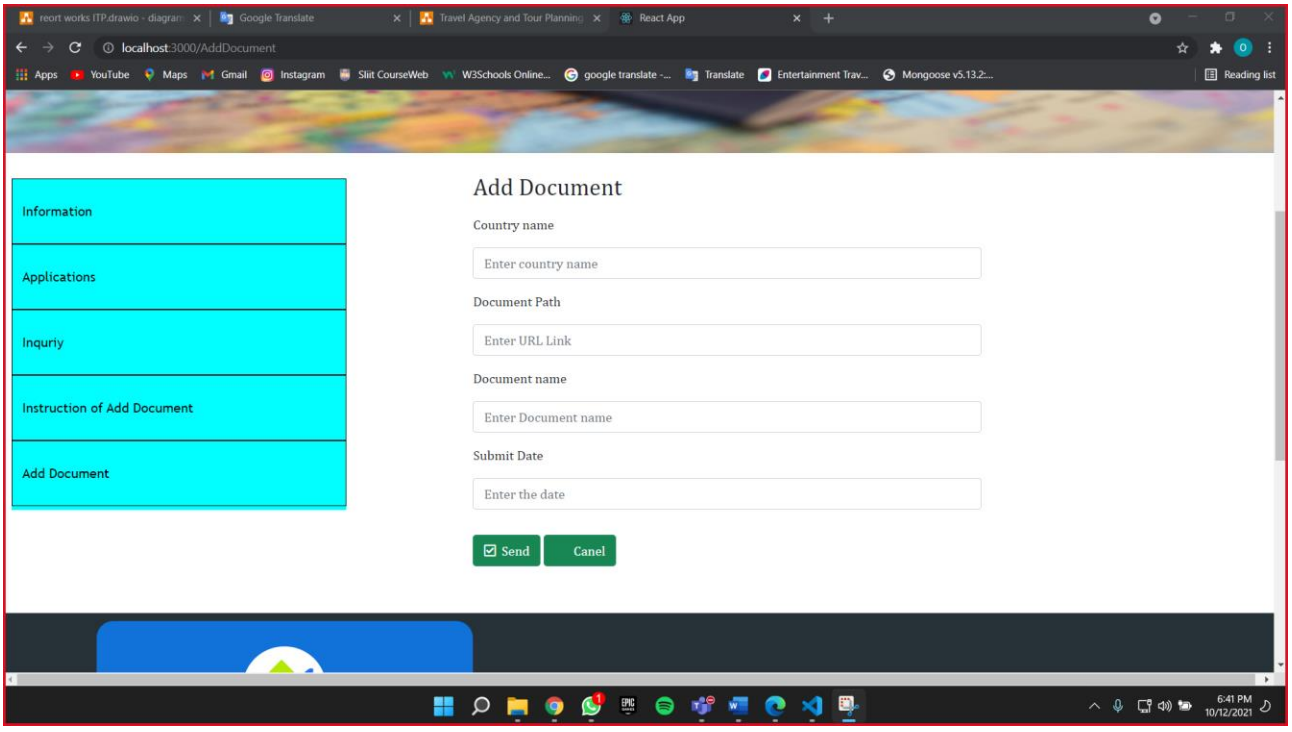


Figure 64 –Travel Document add page

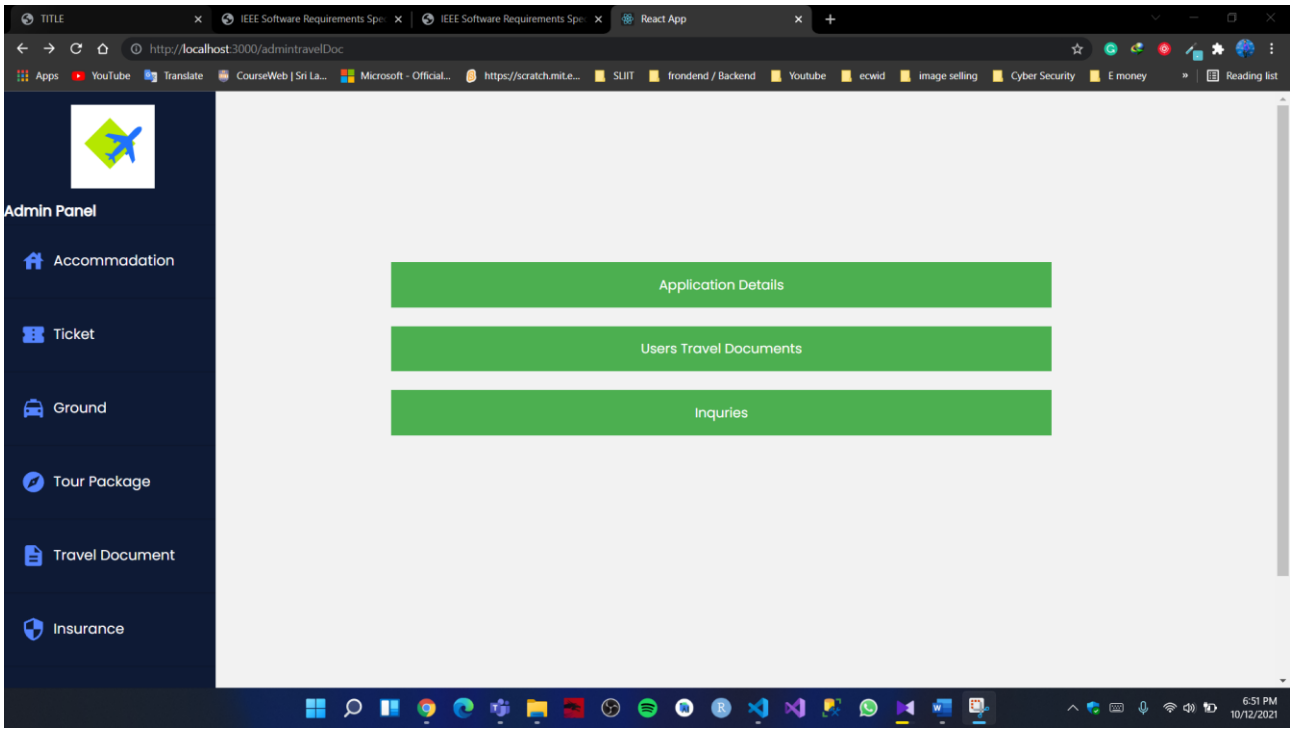


Figure 65 –Travel Document Admin page

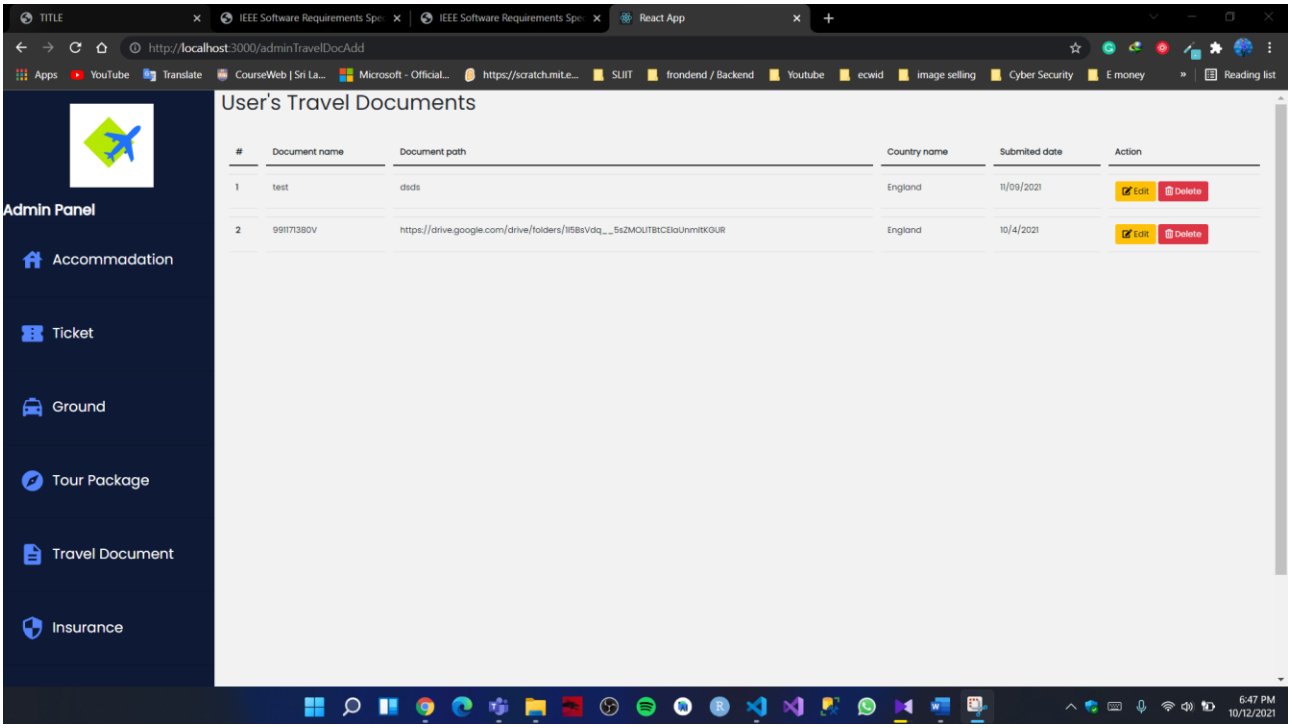


Figure 66 – Customer's Travel Document added list view page

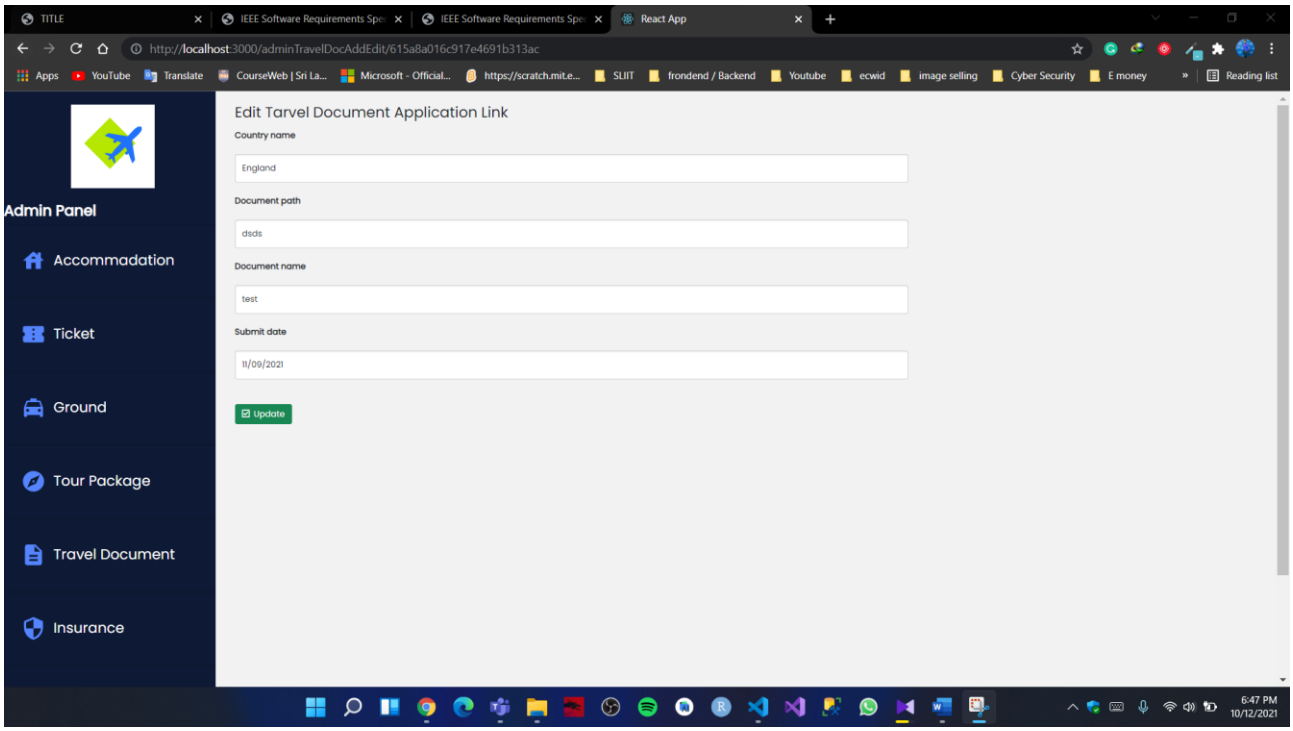


Figure 67 – Customer's Travel Document update page

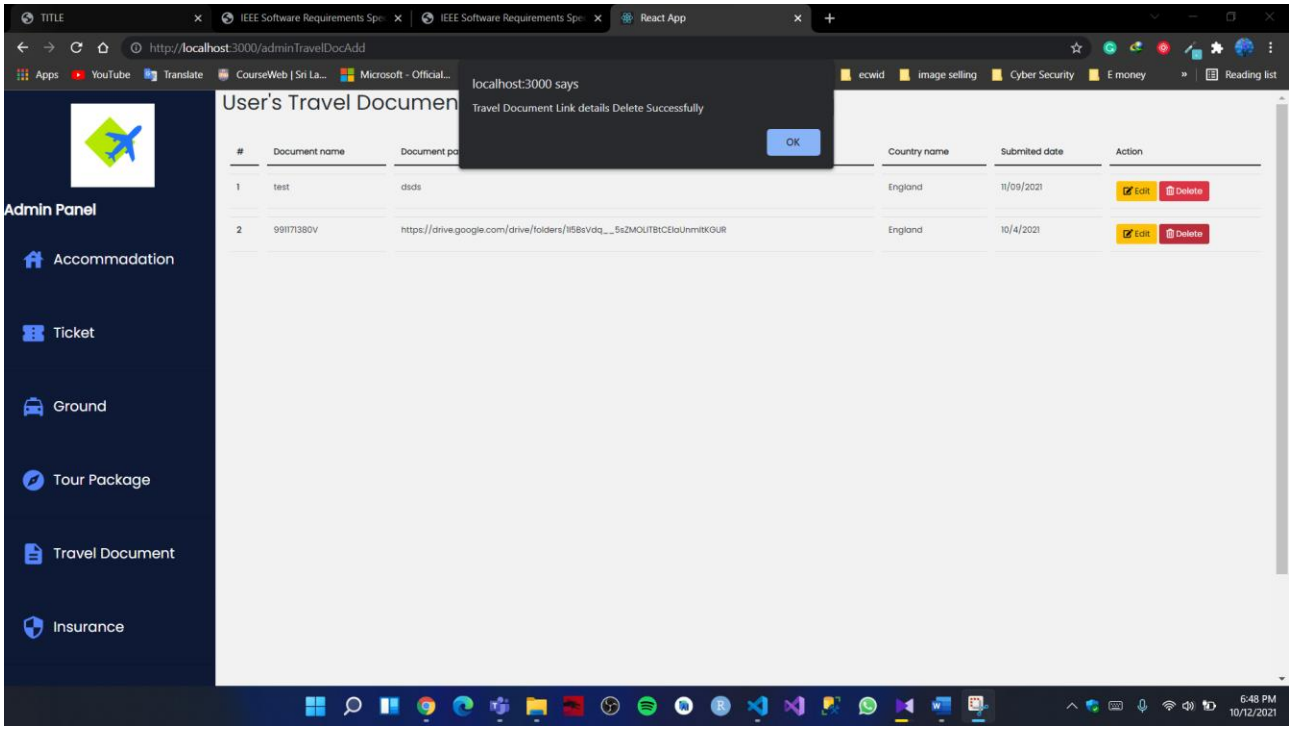


Figure 68 – Customer's Travel Document Delete page

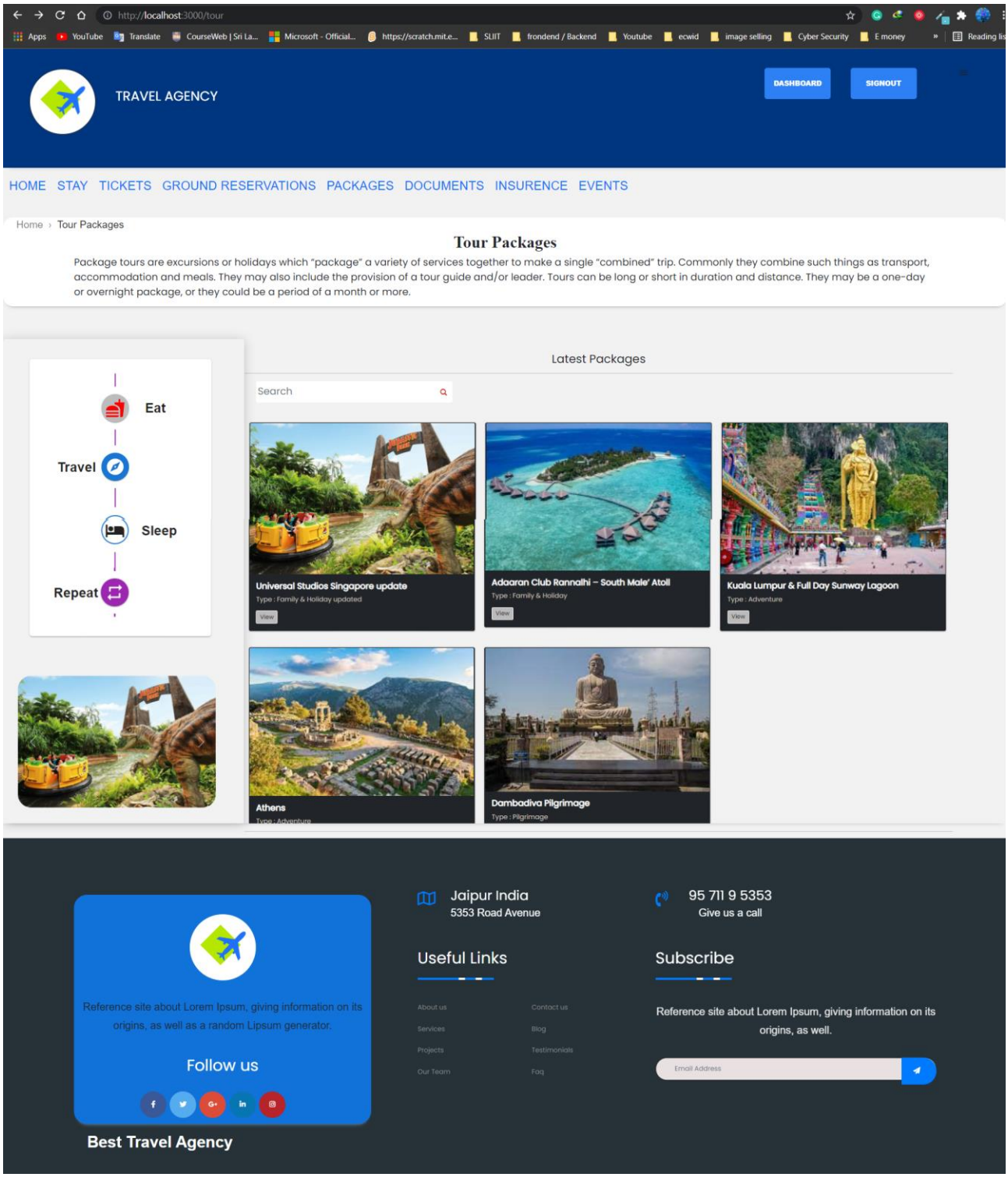


Figure 69 – Tour Package Category view page

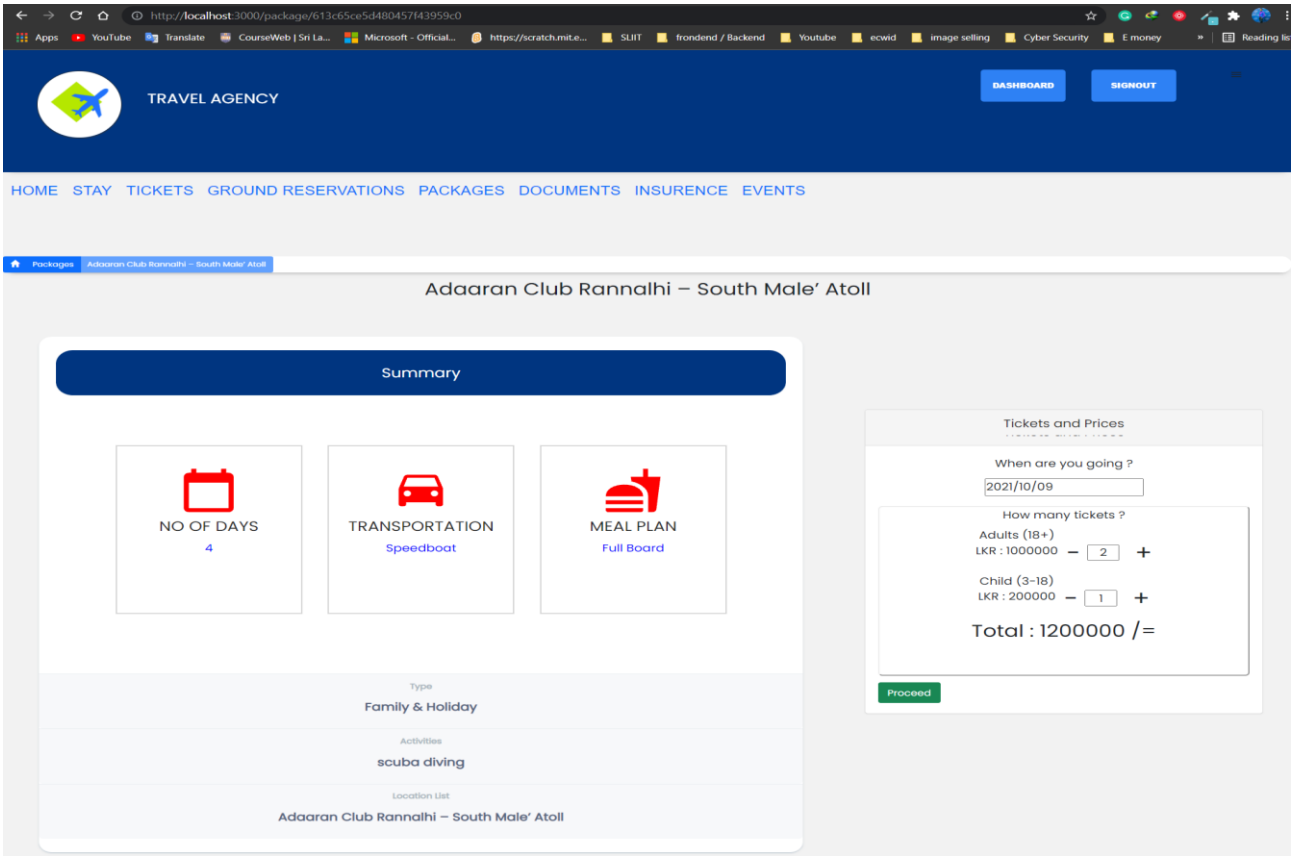


Figure 70 – Selected Tour Package Detail view page

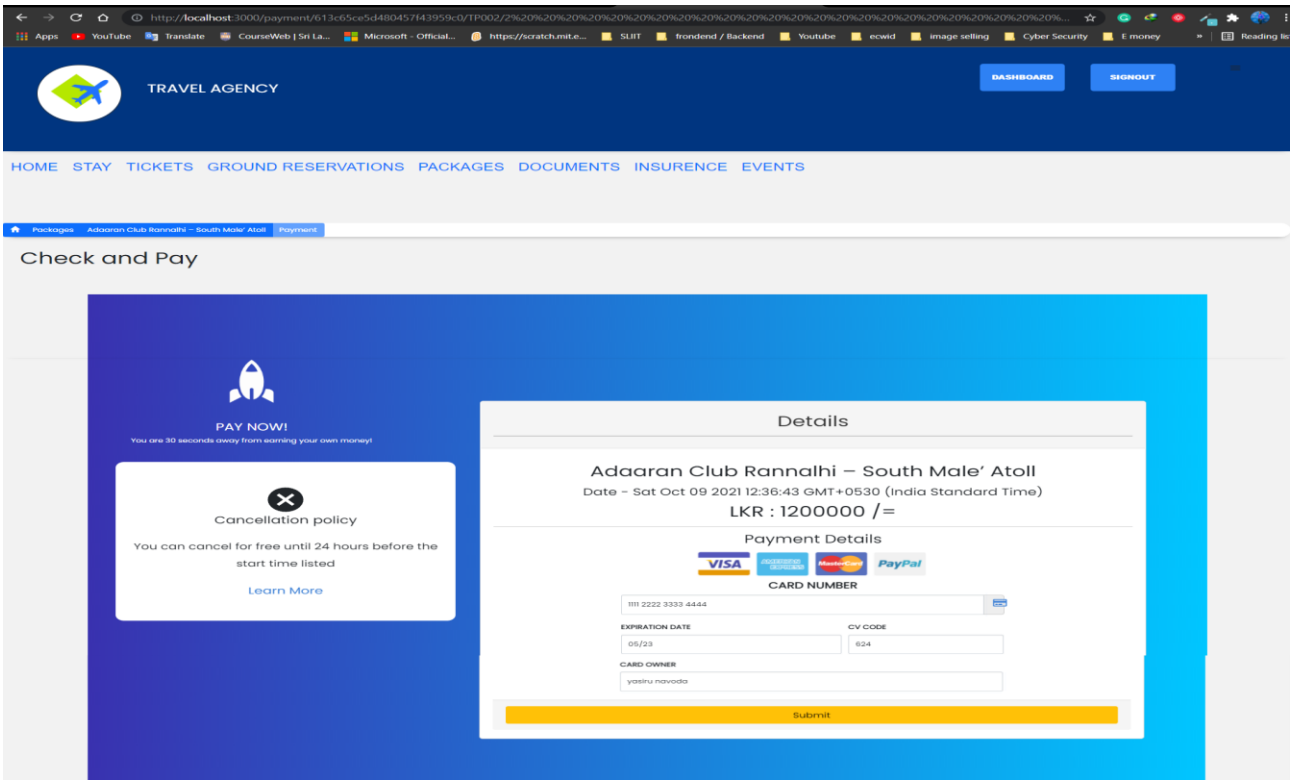


Figure 71 – Selected Tour Package Payment page

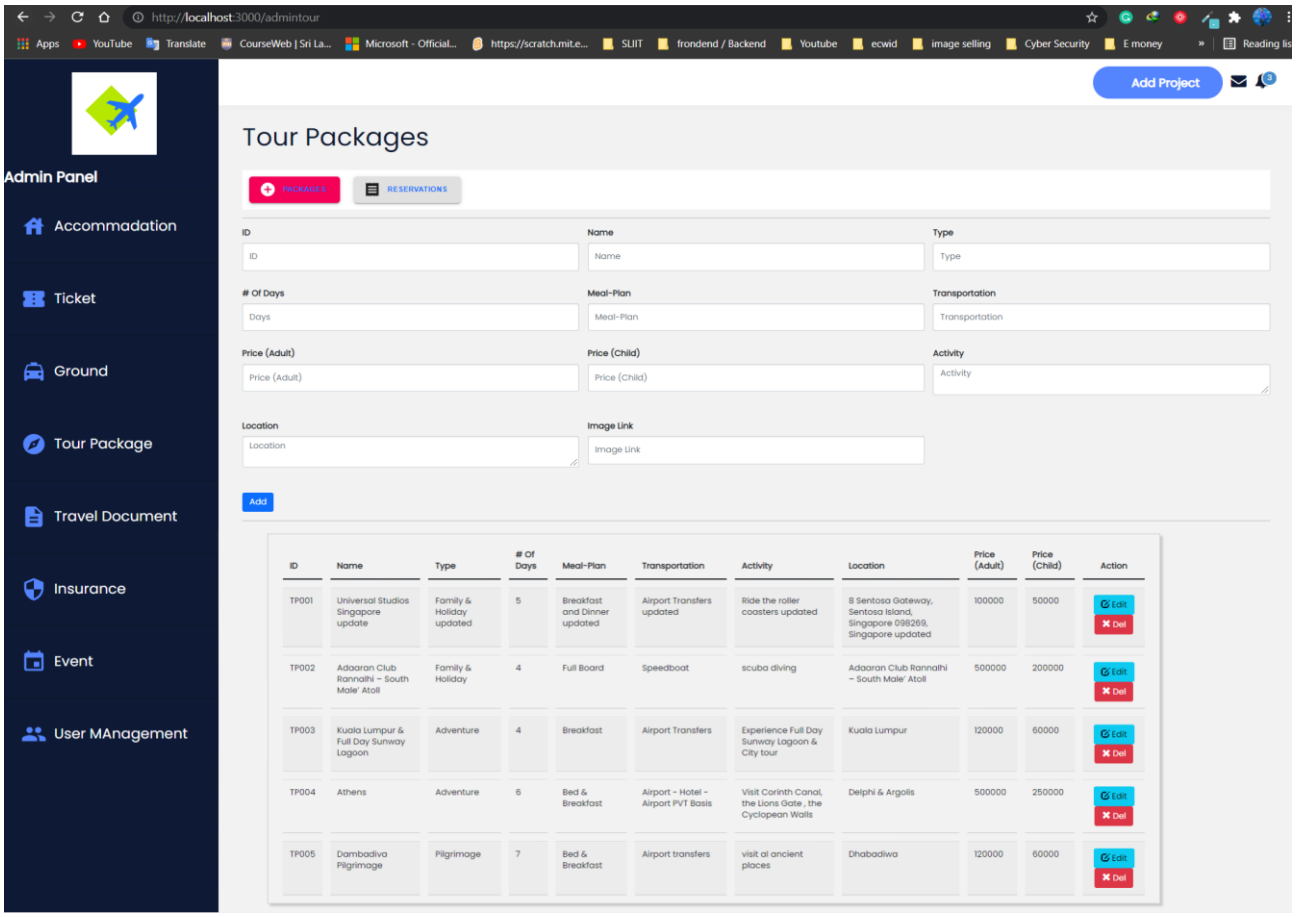


Figure 72 – Tour Package Admin page

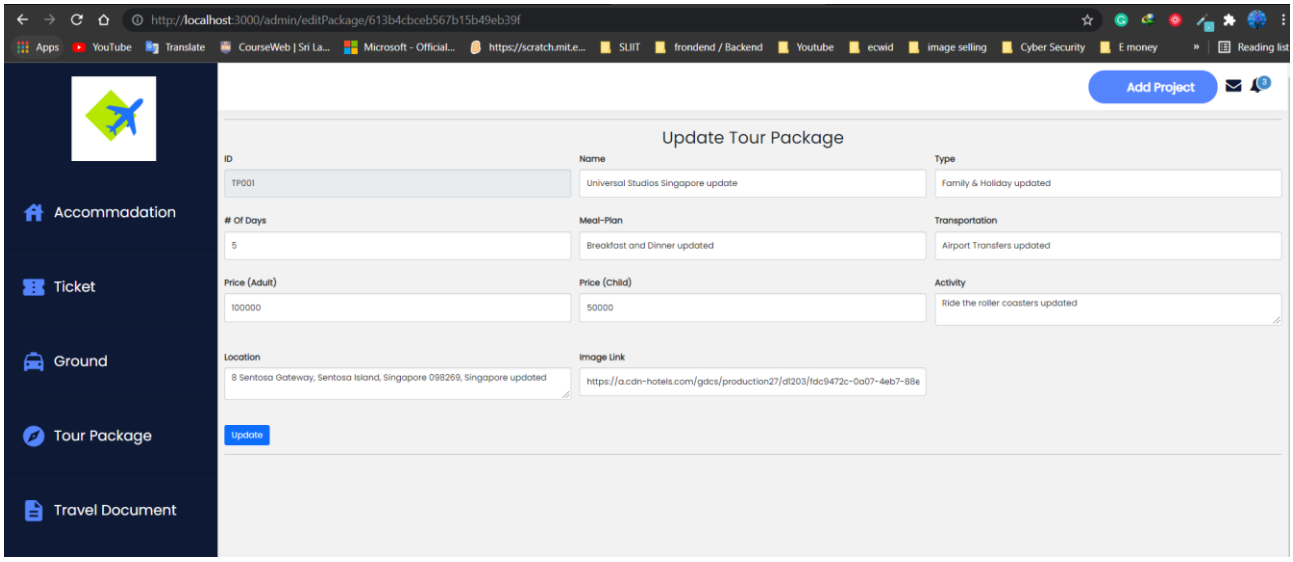


Figure 73 – Tour Package Update Admin page

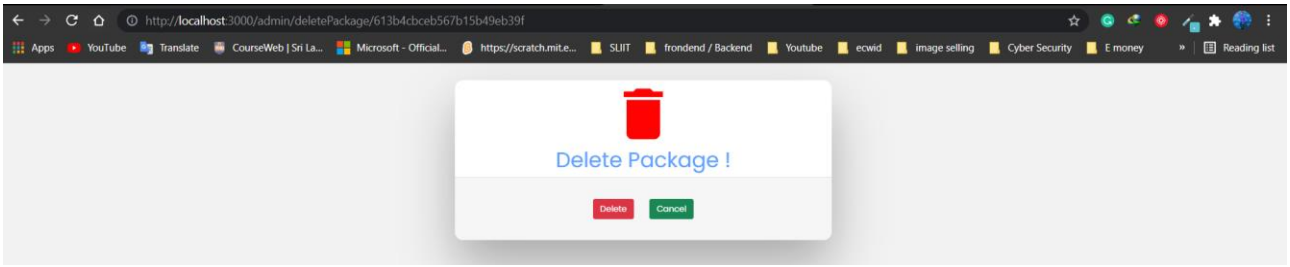


Figure 74 – Tour Package Delete Admin page

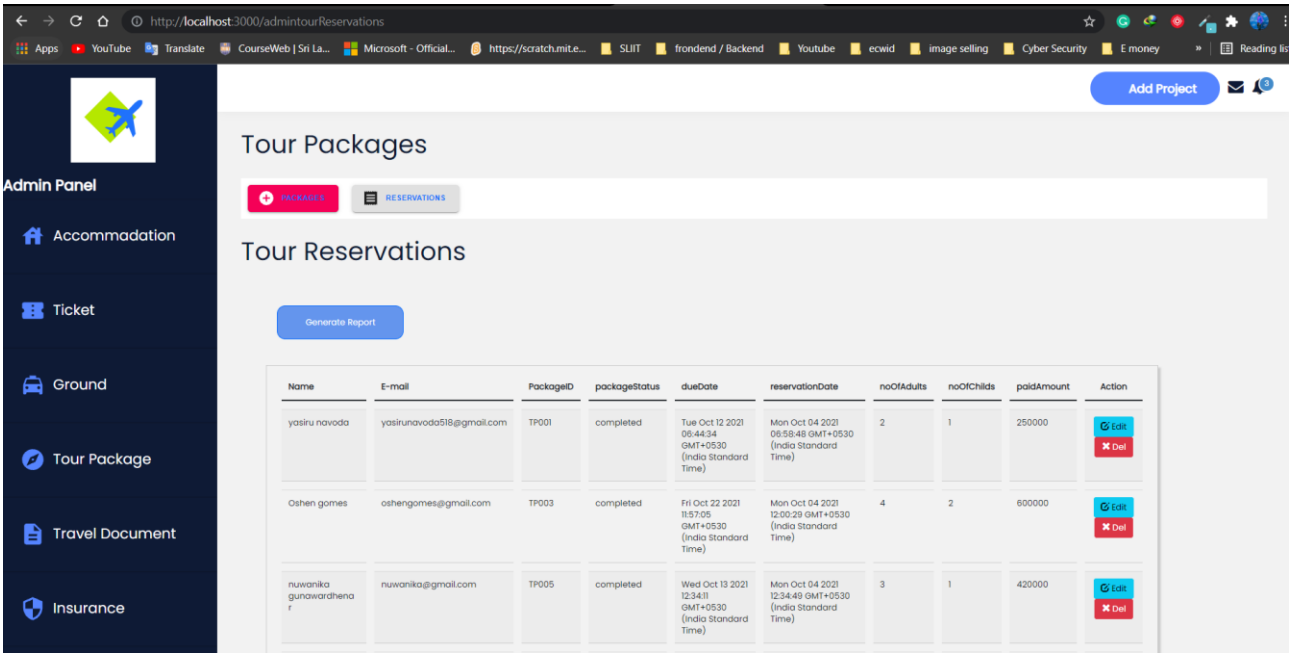


Figure 75 – Tour Package reservation list view Admin page

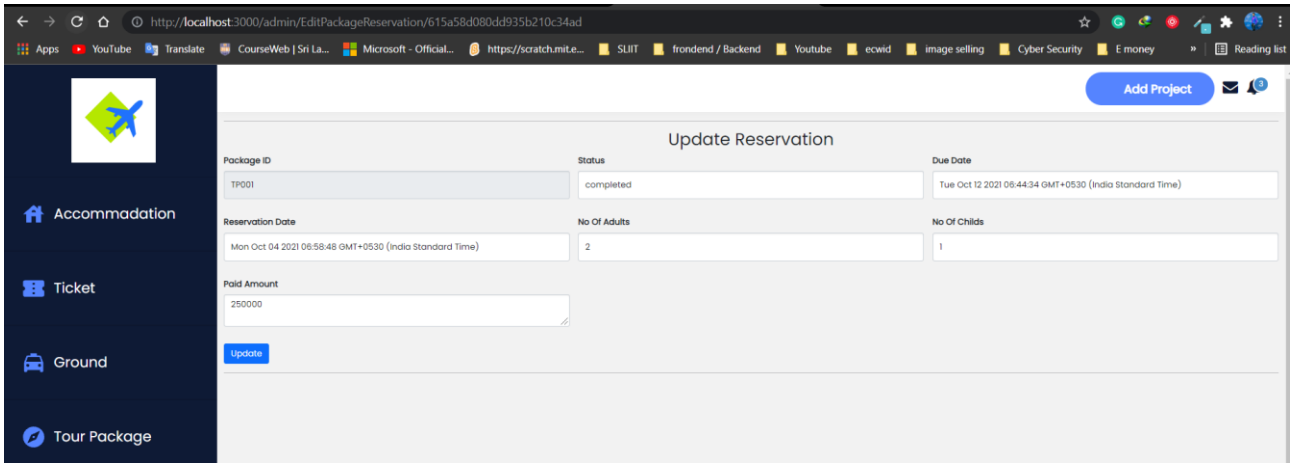


Figure 76 – Tour Package reservation Update Admin page

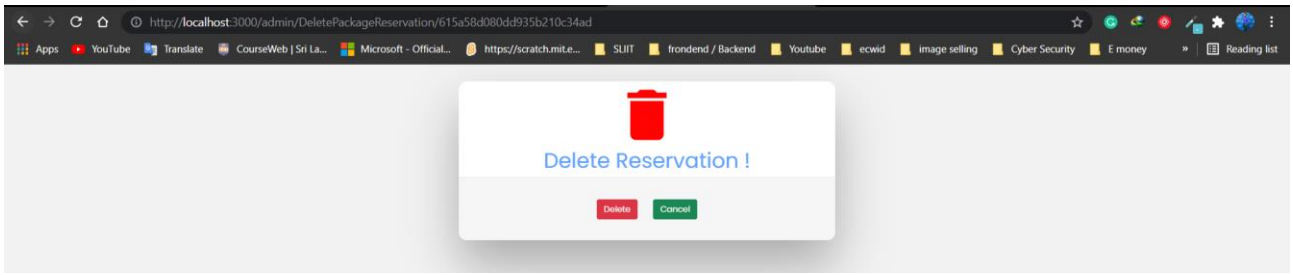


Figure 77 – Tour Package reservation Delete Admin page

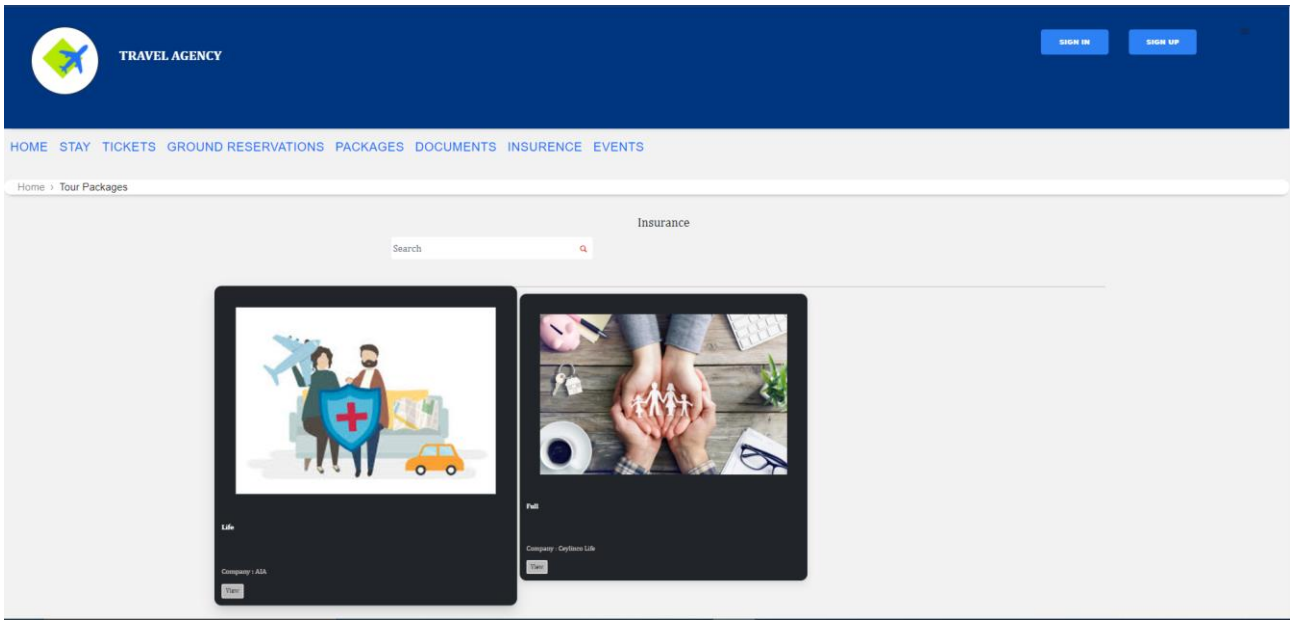


Figure 78 – Insurance Category view page

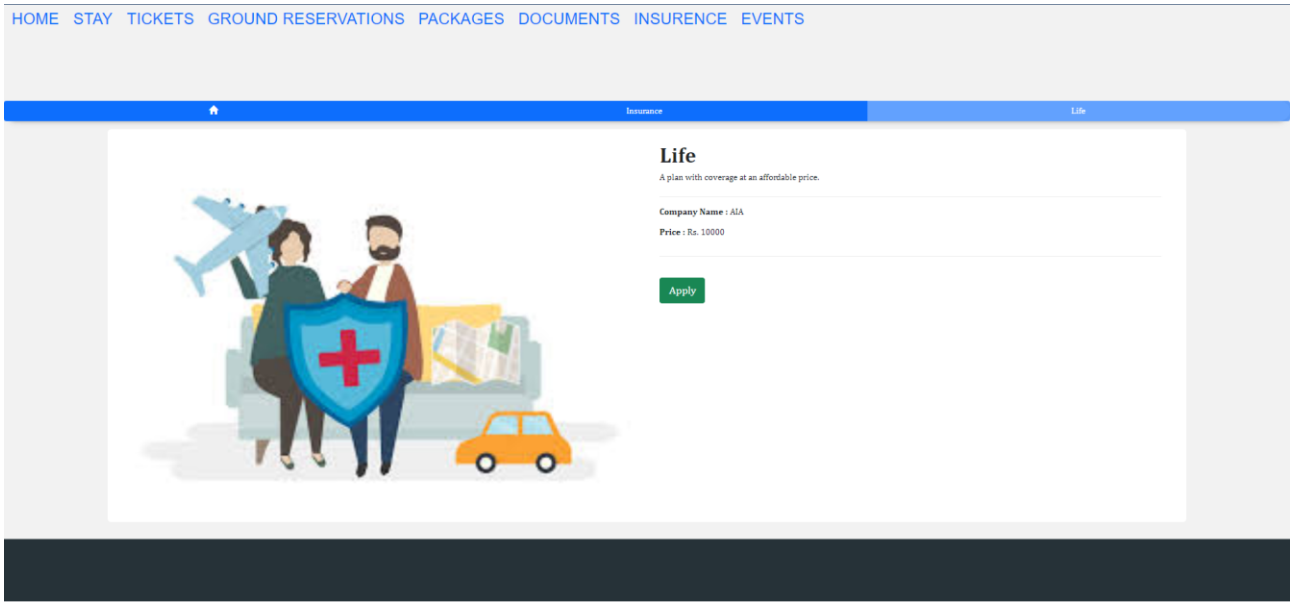


Figure 79 – Selected Insurance Details view page

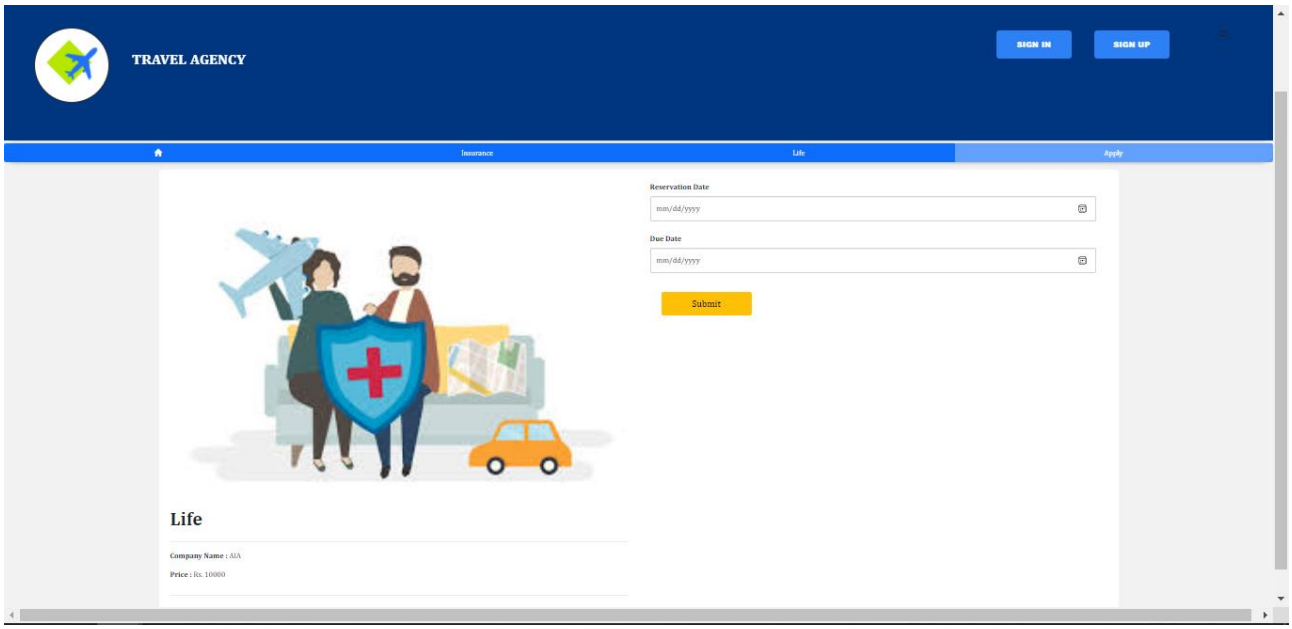


Figure 80 – Selected Insurance Details Resevartion page

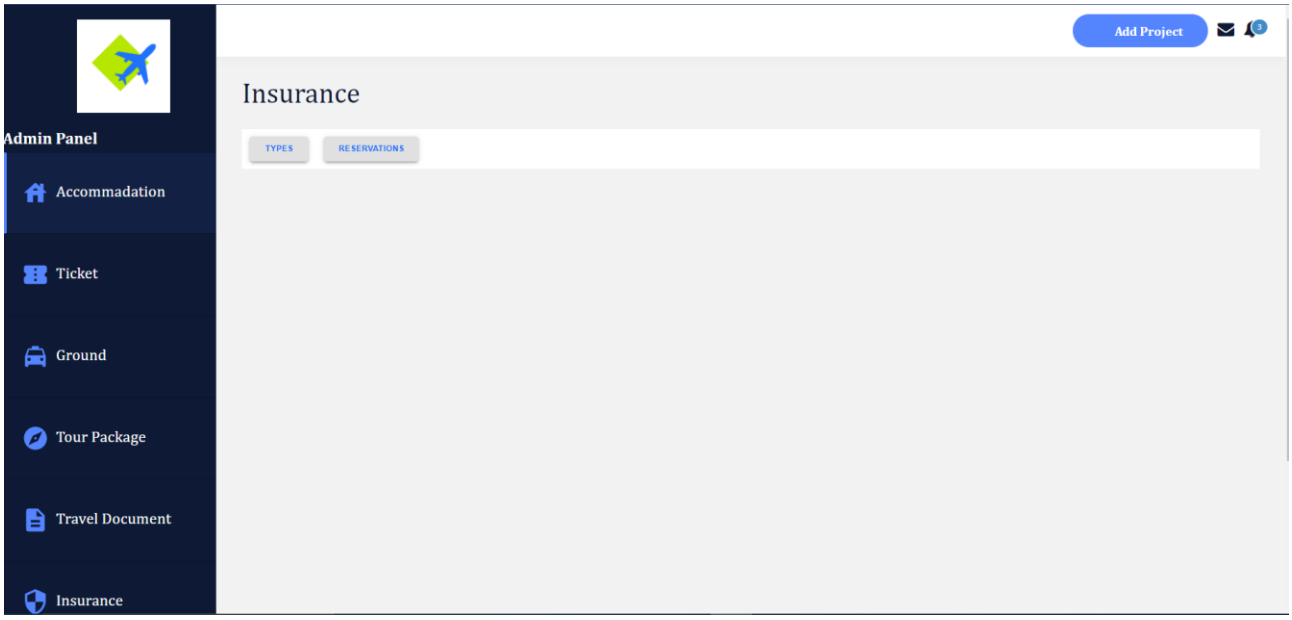


Figure 81 – Insurance Admin page

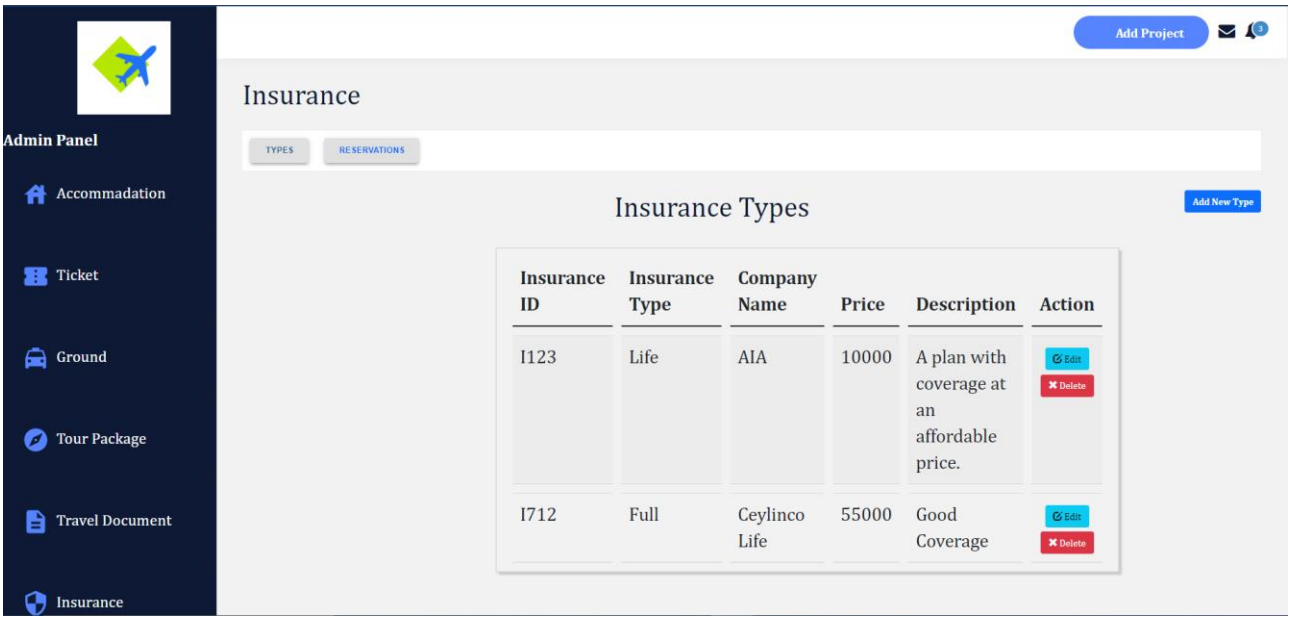


Figure 82 – Insurance type list view Admin page

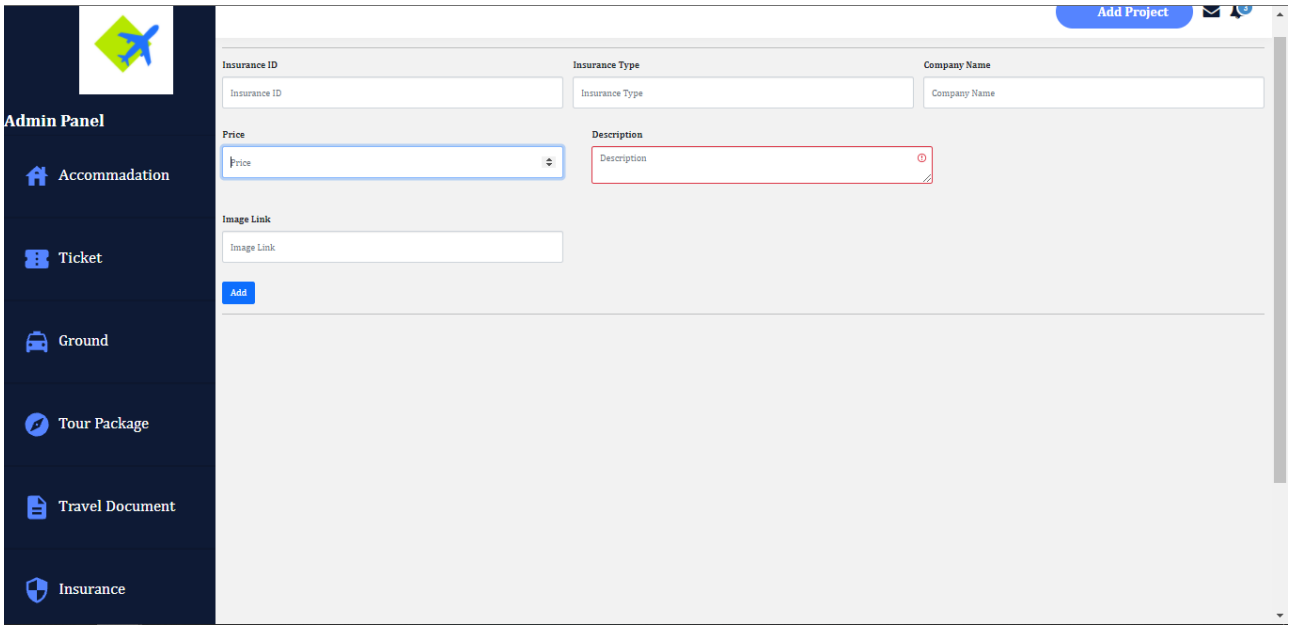


Figure 83 – Insurance type Add Admin page

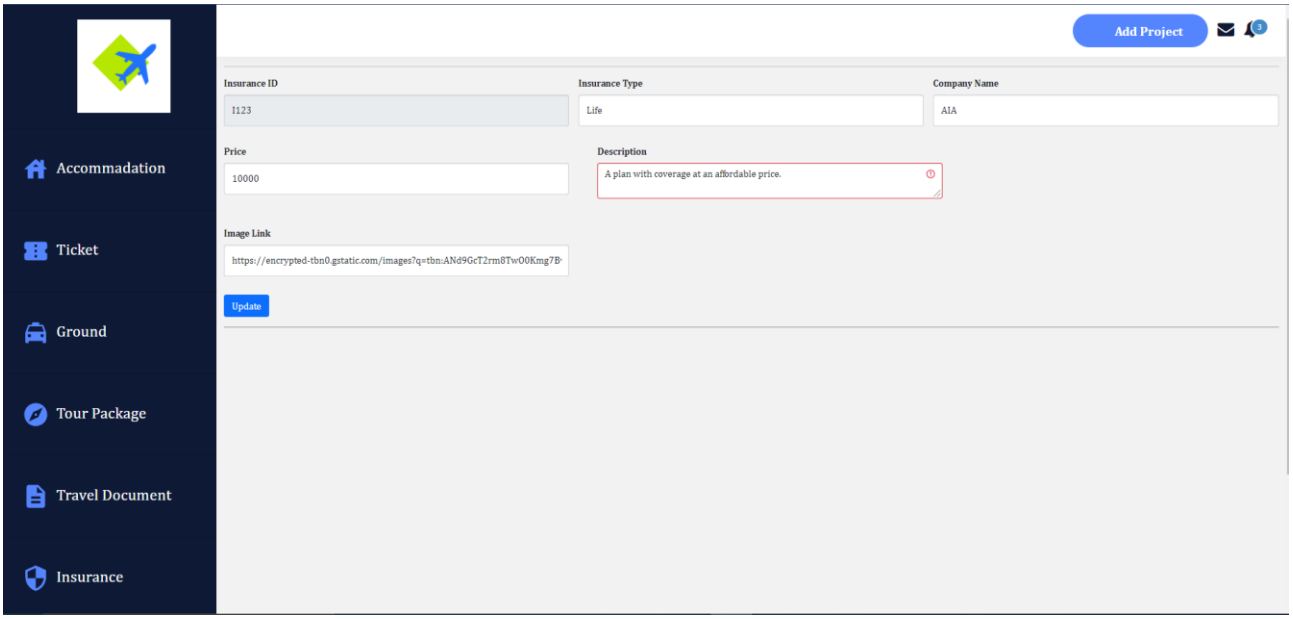


Figure 84 –Insurance type Update Admin page

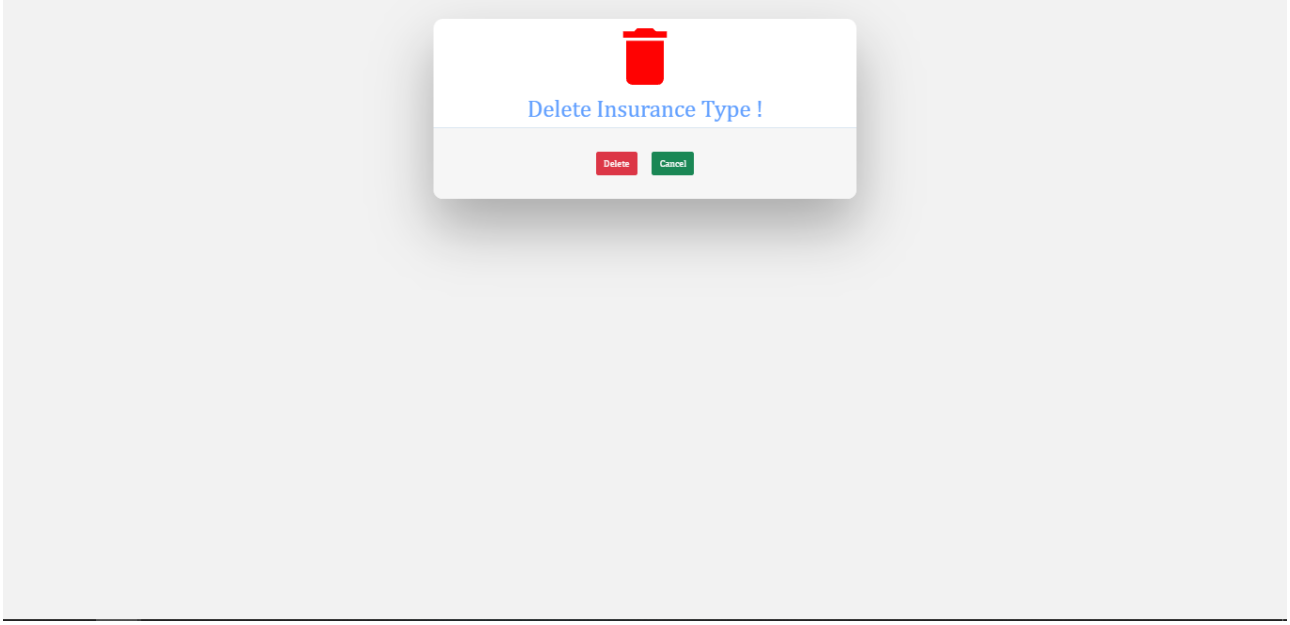


Figure 85 –Insurance type Delete Admin page

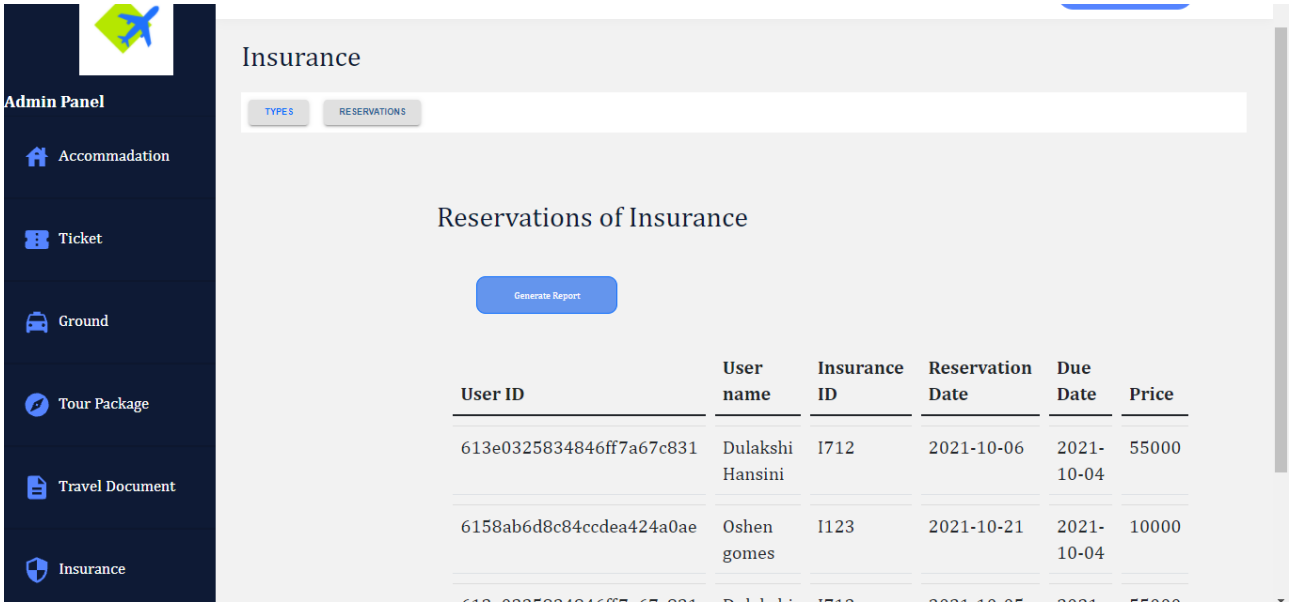


Figure 86 –Insurance Resevartion list Admin page

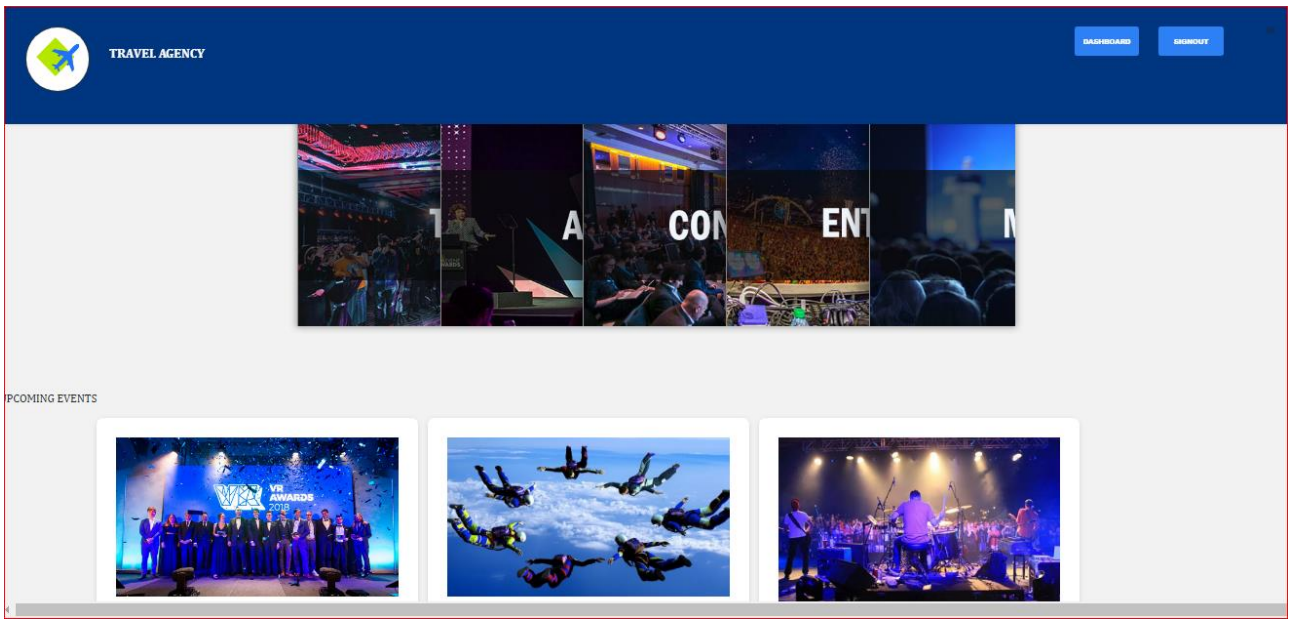


Figure 87 – Event Category view page

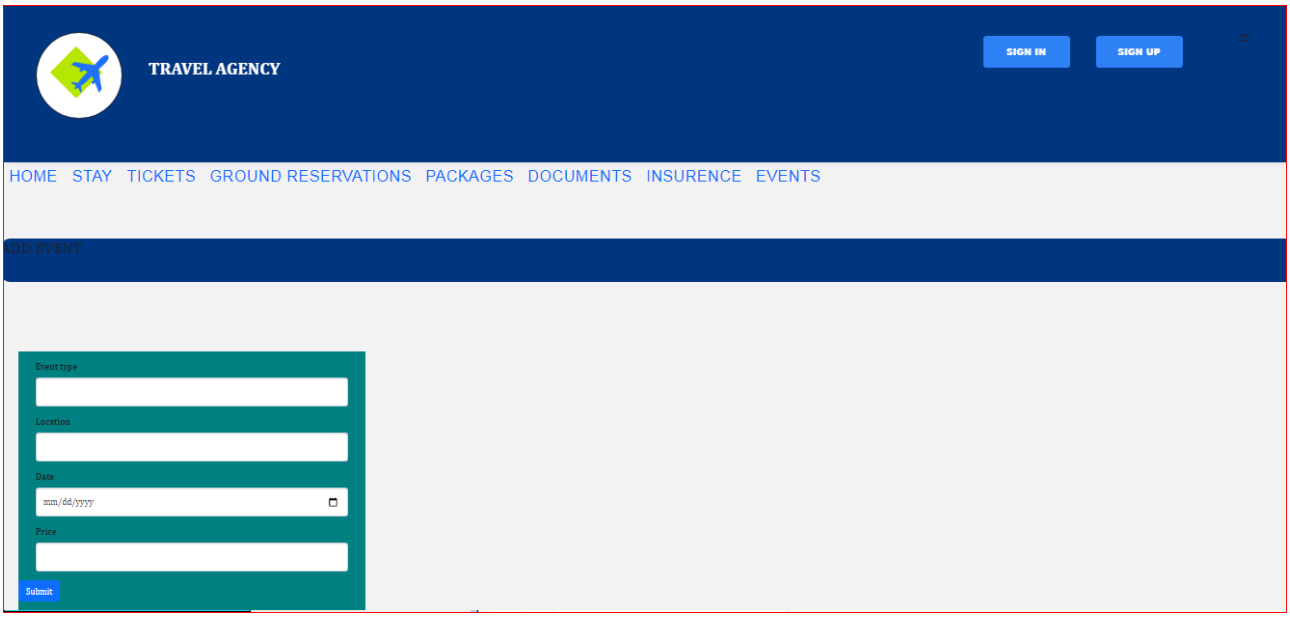


Figure 88 – Event Add page

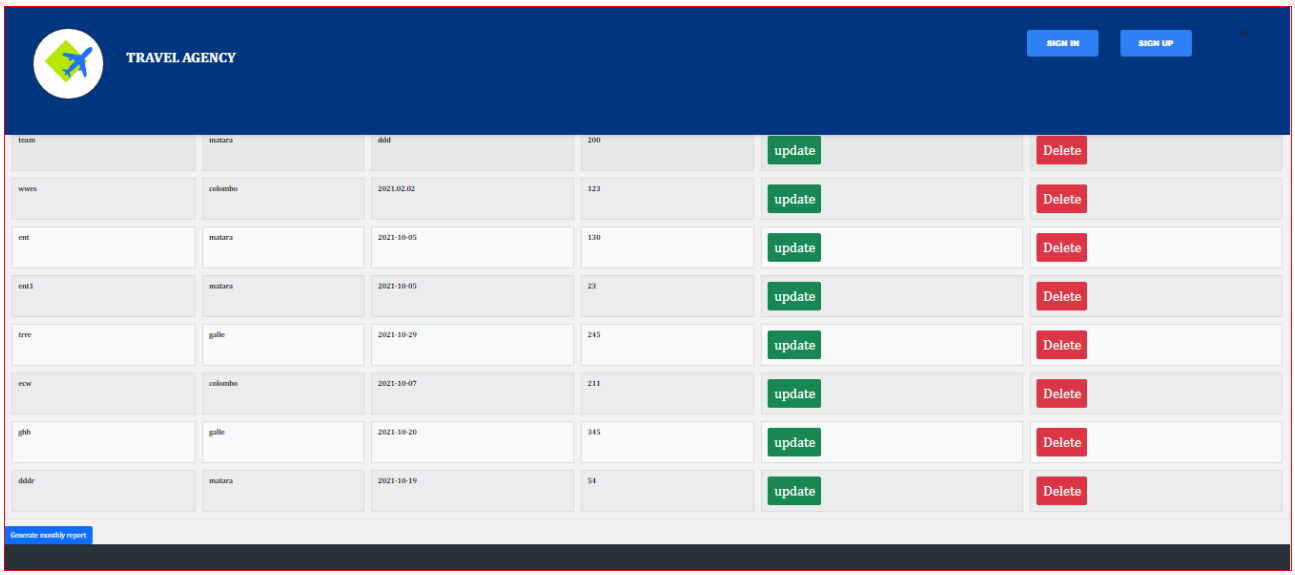


Figure 89 – Event Add list view page

2.3 Implementation

2.3.1 Software

- Mongo DB
- Express
- React js
- Node js
- Postman

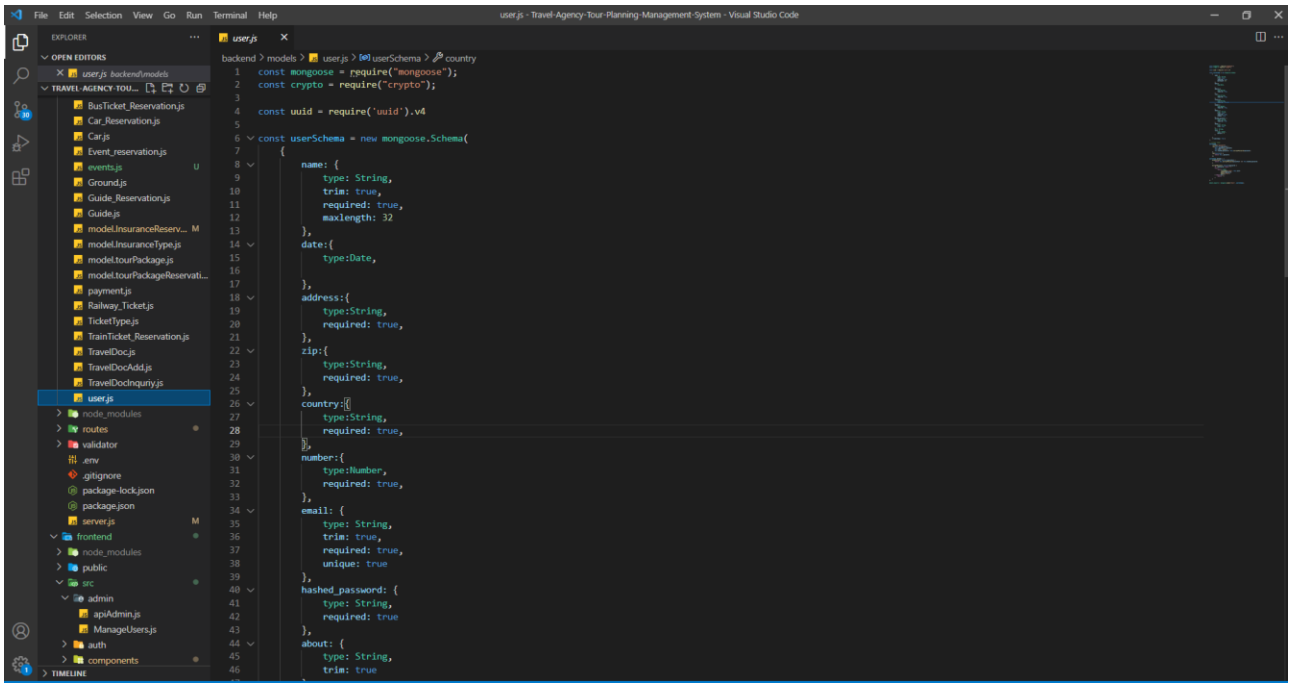
2.3.2 Technology

- Java Script
- CSS
- HTML
- Bootstrap

2.3.3 Codes

2.3.3.1 User Management

Models



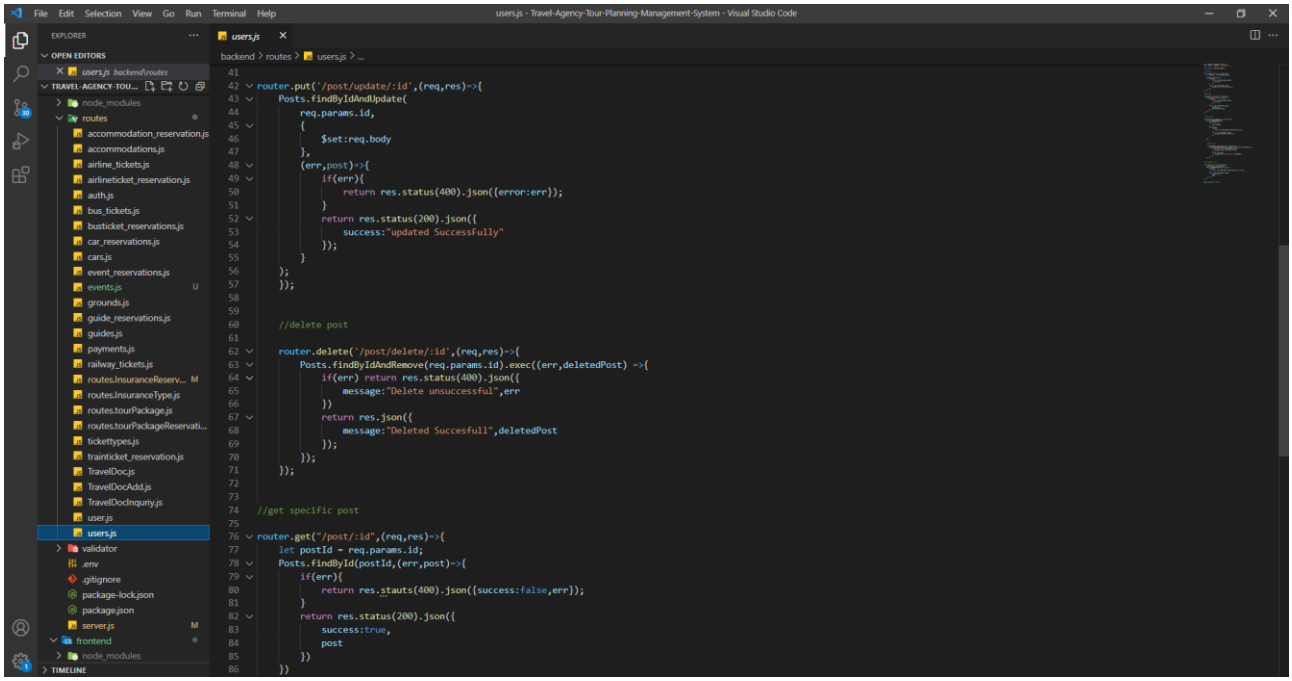
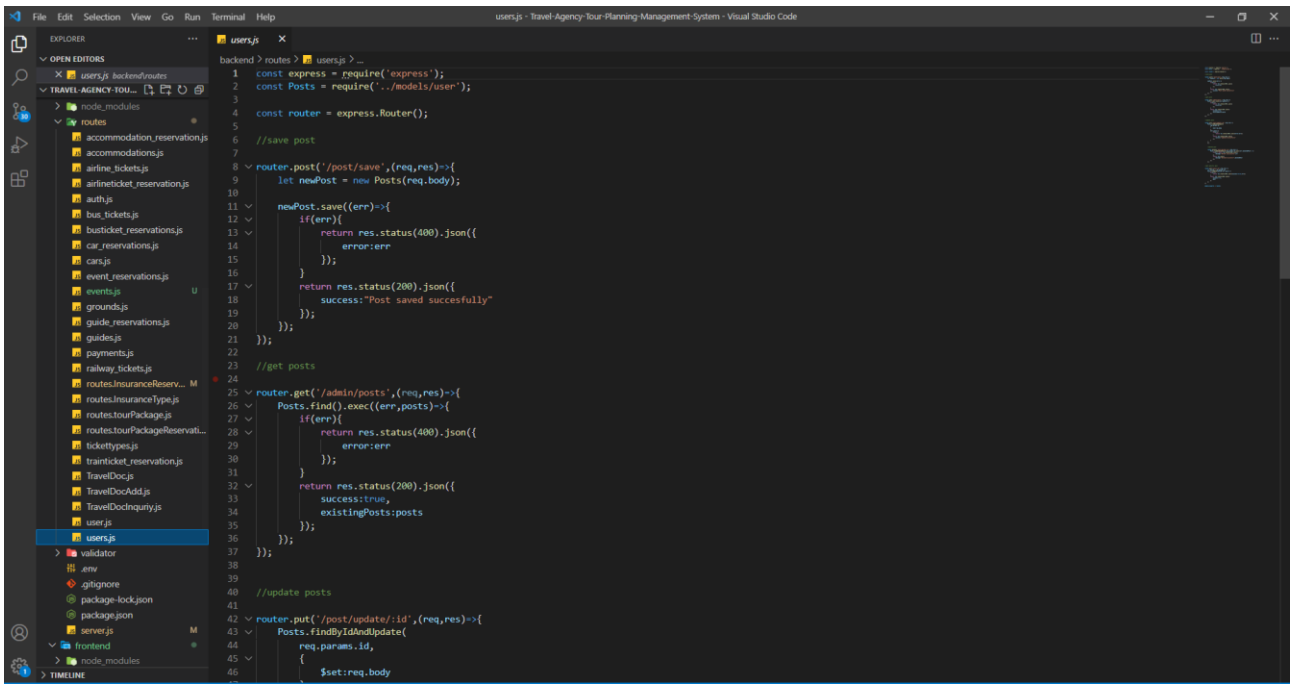


Figure 90

Routes



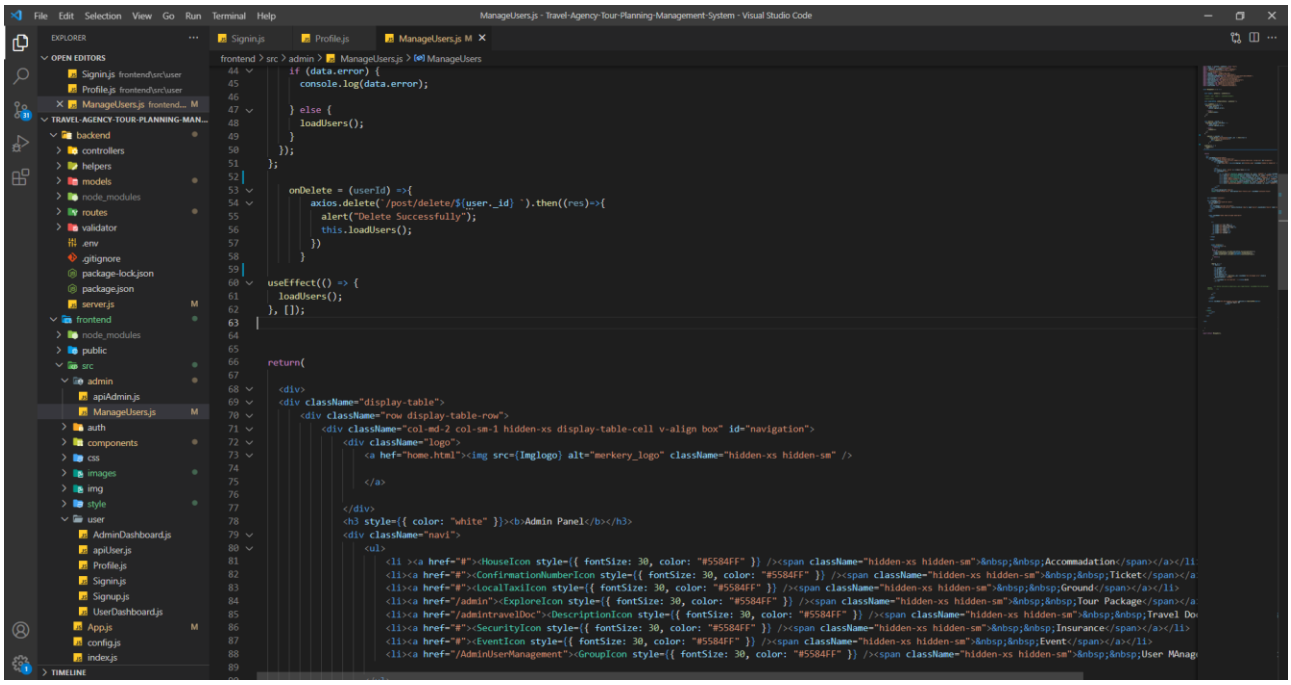
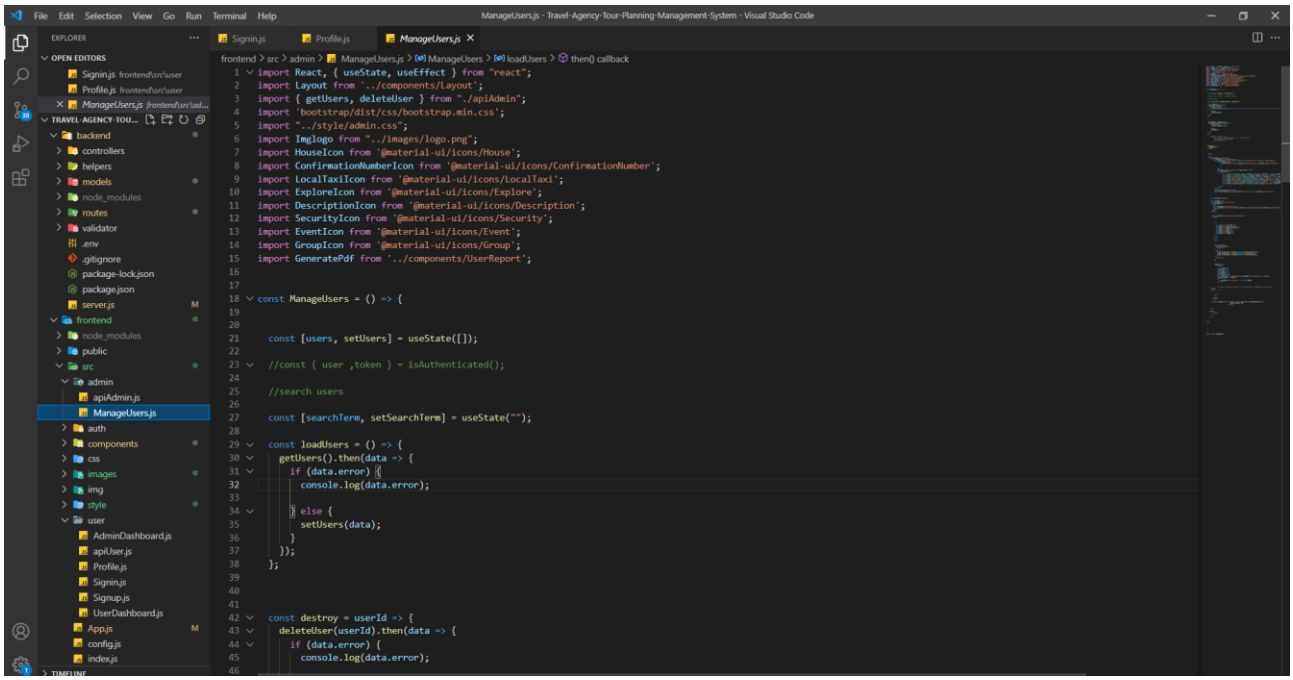
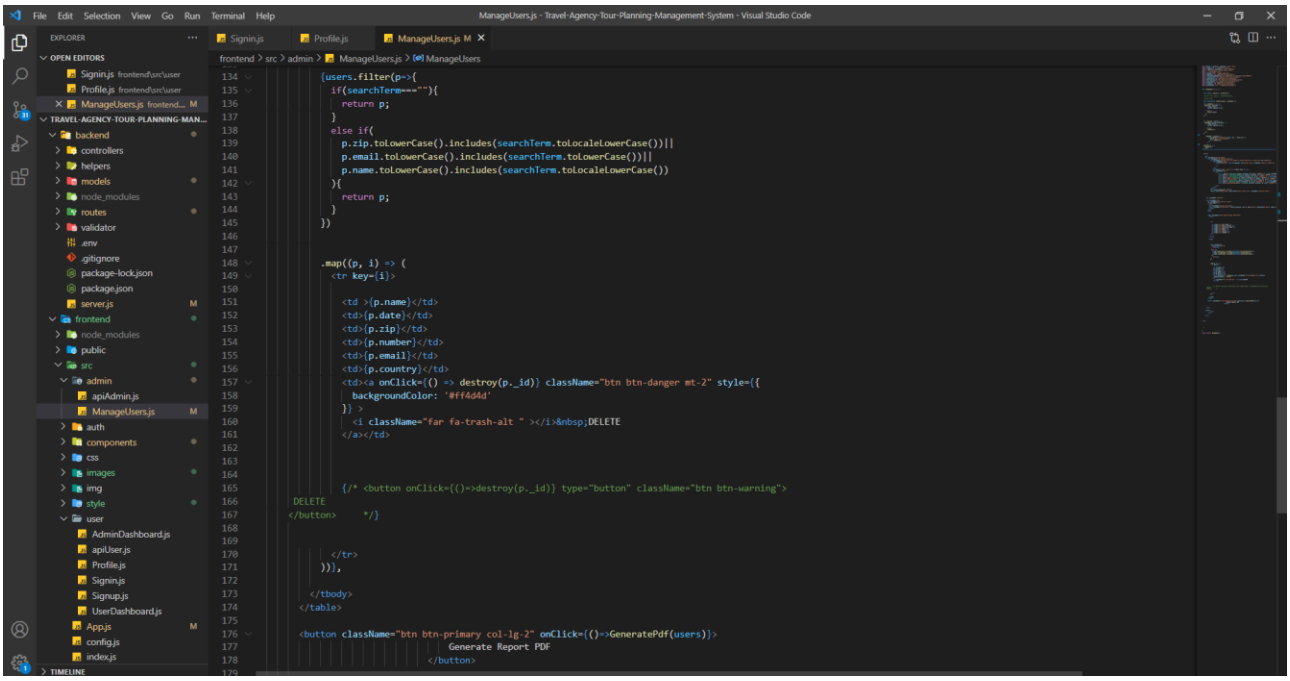
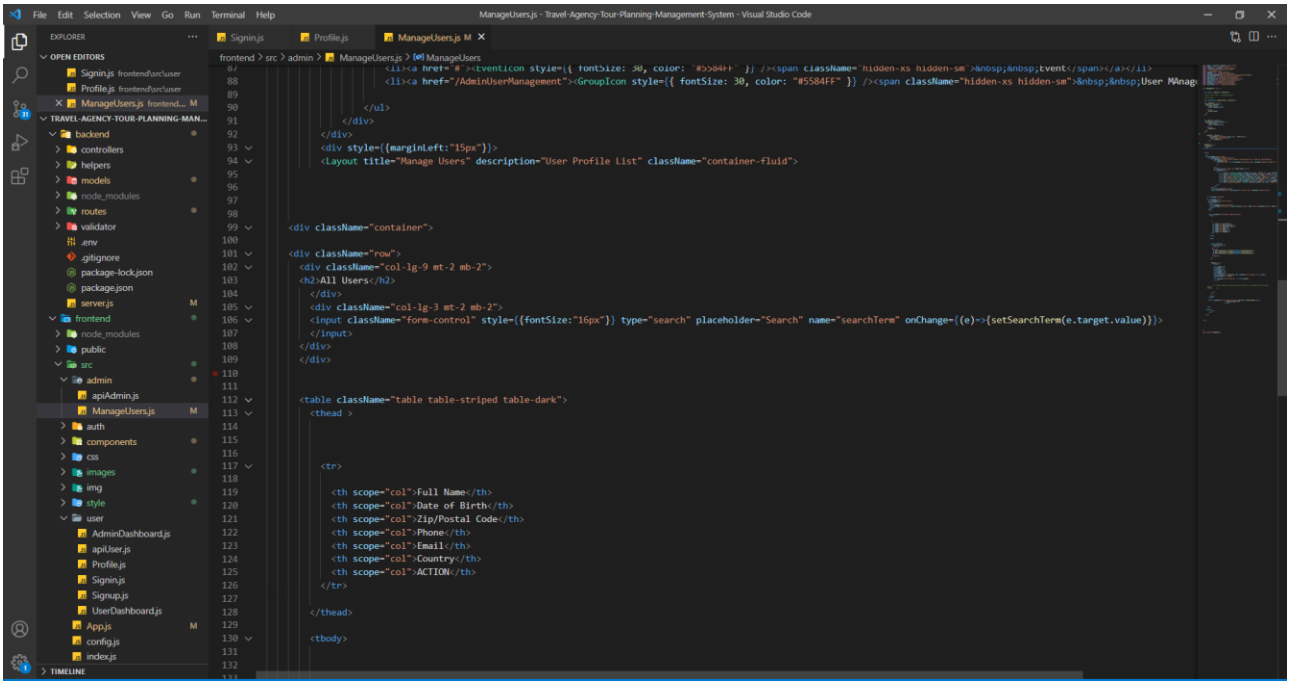


Figure 91

Frontend




File Edit Selection View Go Run Terminal Help
ManageUsers.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
Signin.js
Profile.js
ManageUsers.js M
frontend > src > admin > ManageUsers.js > ManageUsers
OPEN EDITORS
TRAVEL-AGENCY-TOUR-PLANNING-MAN...
backend
controllers
helpers
models
node_modules
routes
validator
.env
.gitignore
package-lock.json
package.json
server.js
frontend
node_modules
public
src
admin
apiAdmin.js
ManageUsers.js
auth
components
css
images
img
style
user
AdminDashboard.js
apiUser.js
Profile.js
Signin.js
Signup.js
UserDashboard.js
App.js
config.js
index.js
TIMELINE
{users.filter(p=>{
if(searchTerm===""){
return p;
}
else if(
p.zip.toLowerCase().includes(searchTerm.toLocaleLowerCase())||
p.email.toLowerCase().includes(searchTerm.toLowerCase())||
p.name.toLowerCase().includes(searchTerm.toLocaleLowerCase())
){
return p;
}
})
.map((p, i) => (
<tr key={i}>
<td >{p.name}</td>
<td>{p.date}</td>
<td>{p.zip}</td>
<td>{p.number}</td>
<td>{p.email}</td>
<td>{p.country}</td>
<td><a onClick={() => destroy(p._id)} className="btn btn-danger mt-2" style={{
backgroundColor: '#ff4d4d'
}} >
 DELETE
</td>
{/* <button onClick={()=>destroy(p._id)} type="button" className="btn btn-warning">
DELETE
</button> */}
</tr>
))}
</tbody>
</table>
<button className="btn btn-primary col-lg-2" onClick={()=>GeneratePdf(users)}>
Generate Report PDF
</button>


File Edit Selection View Go Run Terminal Help
ManageUsers.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
Signin.js
Profile.js
ManageUsers.js M
frontend > src > admin > ManageUsers.js > ManageUsers
OPEN EDITORS
TRAVEL-AGENCY-TOUR-PLANNING-MAN...
backend
controllers
helpers
models
node_modules
routes
validator
.env
.gitignore
package-lock.json
package.json
server.js
frontend
node_modules
public
src
admin
apiAdmin.js
ManageUsers.js
auth
components
css
images
img
style
user
AdminDashboard.js
apiUser.js
Profile.js
Signin.js
Signup.js
UserDashboard.js
App.js
config.js
index.js
TIMELINE
<a href="/AdminUserManagement"><GroupIcon style={{ fontSize: 30, color: "#5584FF" }} /> User MAnag

<div style={{marginLeft:"15px"}}>
<Layout title="Manage Users" description="User Profile List" className="container-fluid">

<h2>All Users</h2>

<input className="form-control" style={{fontSize:"16px"}} type="search" placeholder="Search" name="searchTerm" onChange={(e)=>{setSearchTerm(e.target.value)}}>
</input>

<table className="table table-striped table-dark">
<thead >
<tr>
<th scope="col">Full Name</th>
<th scope="col">Date of Birth</th>
<th scope="col">Zip/Postal Code</th>
<th scope="col">Phone</th>
<th scope="col">Email</th>
<th scope="col">Country</th>
<th scope="col">ACTION</th>
</tr>
</thead>
<tbody>

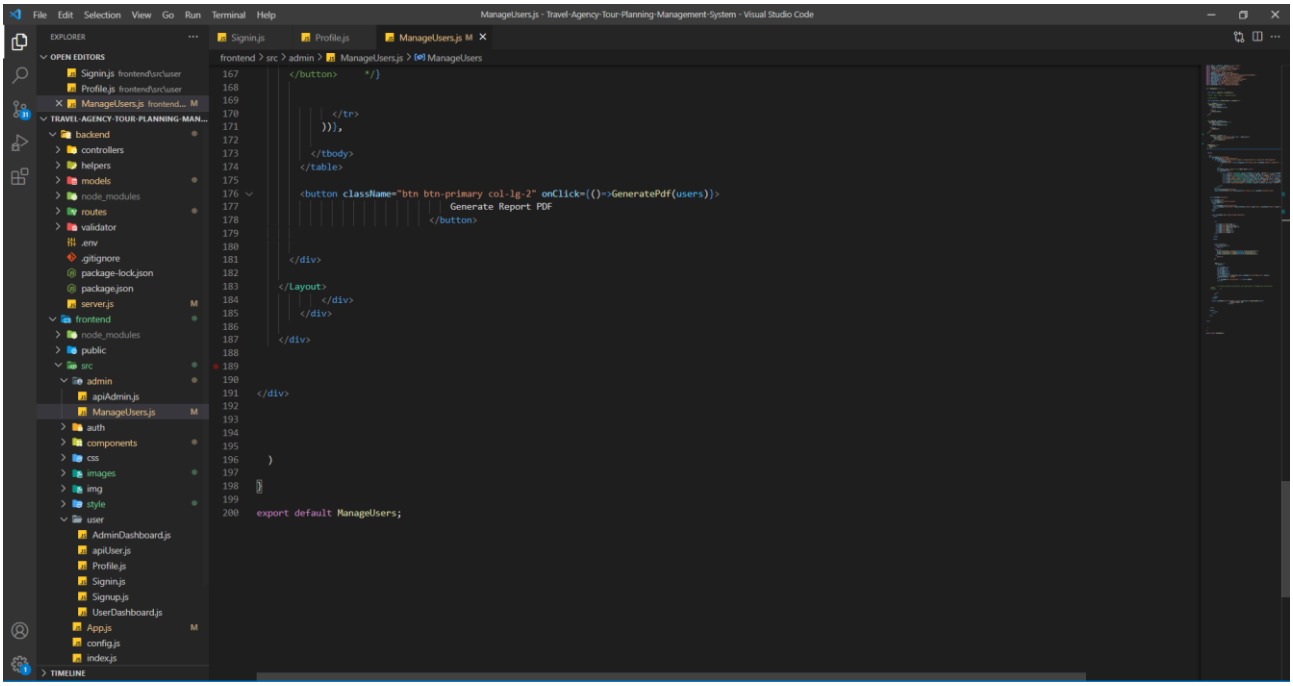


Figure 92

2.3.3.2 Accommodation reservation Management

Models

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

const Schema = mongoose.Schema;

const accommodationSchema = new Schema({
    accommodation_ID : {
        type : Number,
        required: true
    },
    name : {
        type : String,
        required: true
    },

    location : {
        type : String,
        required: true

    },
    type : {
        type : String,
        required: true
    },
    price : {
        type : Number,
        required: true
    },
    owner : {
        type : String,
        required: true
    },
    phone : {
        type : Number,
        required: true
    },
    link : {
        type : String,
```

```
    },
    type : {
        type : String,
        required: true
    },
    price : {
        type : Number,
        required: true
    },
    owner : {
        type : String,
        required: true
    },
    phone : {
        type : Number,
        required: true
    },
    link : {
        type : String,
        required: true
    }

})
const Accommodation = mongoose.model("Accommodation",accommodationSchema);
module.exports=Accommodation;
```

Figure 93

Routes

```
1   const router= require("express").Router();
2   let Accommodation =require("../models/accommodation");
3
4   router.route("/add").post((req,res)=>{
5       const accommodation_ID= Number(req.body.accommodation_ID);
6       const name= req.body.name;
7       const location= req.body.location;
8       const type= req.body.type;
9       const phone= Number(req.body.phone);
10      const price= Number(req.body.price);
11      const owner = req.body.owner;
12      const link = req.body.link;
13
14      const newAccommodation = new Accommodation({
15          accommodation_ID,
16          name,
17          location,
18          type,
19          price,
20          owner,
21          phone,
22          link
23      })
24      newAccommodation.save().then(()=>{
25          res.json("Accommodation Added")
26      }).catch((err)=>{
27          console.log(err);
28      })
29  })
30
31  router.route("/").get((req,res)=>{
32      Accommodation.find().then((accommodations)=>{
33          res.json(accommodations)
34      }).catch((err)=>{
35          console.log(err)
36      })
37  })
```

```
router.route('/update/:id').put((req, res, next) => {
    Accommodation.findByIdAndUpdate(req.params.id, {
      $set: req.body
    }, (error, data) => {
      if (error) {
        return next(error);
        console.log(error)
      } else {
        res.json(data)
        console.log('Accommodation updated successfully !')
      }
    })
  })

router.route("/delete/:id").delete(async(req, res)=>{
    let accId=req.params.id;

    await Accommodation.findByIdAndDelete(accId)
    .then(()=>{
        res.status(200).send({status:" accommodation deleted"});
    }).catch((err)=>{
        console.log(err.message);
        res.status(500).send({status:"Error with delete",error:err.message});
    })
})


router.route('/get/:id').get((req, res) => {
    Accommodation.findById(req.params.id, (error, data) => {
      if (error) {
        return next(error)
      } else {
        res.json(data)
      }
    })
  })
module.exports=router;
```

Figure 94

Frontend

```
import React,{useState} from "react"
import axios from "axios";


export default function AddAccommodation(){

    const[accommodation_ID, setAccommodation_ID]= useState("");
    const[name, setname]= useState("");
    const[location, setlocation]= useState("");
    const[type, setType]= useState("");
    const[price, setprice]= useState("");
    const[owner, setowner]= useState("");
    const[phone, setphone]= useState("");
    const[link, setlink]= useState("");

    function sendData(e){
        e.preventDefault();
        const newAccommodation={
          accommodation_ID,
          name,
          location,
          type,
          price,
          owner,
          phone,
          link
        }
        //console.log(newAccommodation);
        axios.post("http://localhost:8000/accommodation/add",newAccommodation).then(()=>{
          alert("Accommodation added")
        }).catch((err)=>{
          alert(err)
        })
    }

    return(

```

```
      <div>
        <nav className="navbar navbar-expand-lg navbar-light bg-light">
  <a className="navbar-brand" href="/admin_panel/home"style={{color:"red "}}>Accommodation Management</a>
  <button className="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
    <span className="navbar-toggler-icon"></span>
  </button>

  <div className="collapse navbar-collapse" id="navbarSupportedContent">
    <ul className="navbar-nav mr-auto">
      <li className="nav-item active">
        <a className="nav-link" href="/admin_panel/accommodation_admin">All Accommadations <span className="sr-only">(current)</span></a>
      </li>
      <li className="nav-item">
        <a className="nav-link" href="/admin_panel/accommodation_admin/addnew">Add new Accommodation</a>
      </li>




    </ul>

  </div>
</nav>

<header className="bg-dark py-5">
            <div className="container px-4 px-lg-5 my-5">
                <div className="text-center text-white">
                    <h1 className="display-4 fw-bolder">Add new Accommodation here</h1>
                    <p className="lead fw-normal text-white-50 mb-0"></p>
                </div>
            </div>

        </header>
        <br/><br/>
     <div className="container">
        <form className="row g-3 needs-validation" novalidate onSubmit={sendData}>
          <div classNam="col-md-4">
```

```
        <br/><br/>
     <div className="container">
        <form className="row g-3 needs-validation" novalidate onSubmit={sendData}>
          <div className="col-md-4">
            <label for="accommodation_ID" className="form-label">remark_ID</label>
            <input type="number" className="form-control " id="validationServer01" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
             onChange={(e)=>{
                setAccommodation_ID(e.target.value)

            }}/>

            <div className="valid-feedback">
              Looks good!
            </div>

          </div>
          <div className="col-md-4">
            <label for="name" className="form-label">Name</label>
            <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
             onChange={(e)=>{
                setname(e.target.value)

            }}/>

            <div className="valid-feedback">
              Looks good!
            </div>

          </div>
          <div className="col-md-4">
            <label for="type_ID" className="form-label">Type</label>
            <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
            onChange={(e)=>{
                setType(e.target.value)

            }}/>

```

```
<div className="col-md-4">
  <label for="type_ID" className="form-label">Type</label>
  <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
  onChange={(e)=>{
    setType(e.target.value)

  }}/>

  <div className="valid-feedback">

  </div>

</div>
<div className="col-md-4">
  <label for="price" className="form-label">price</label>
  <input type="number" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
   onChange={(e)=>{
    setprice(e.target.value)

  }}/>

  <div className="valid-feedback">

  </div>

</div>
<div className="col-md-4">
  <label for="owner" className="form-label">owner</label>
   <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
   onChange={(e)=>{
    setowner(e.target.value)
  }}/>
    <div id="validationServerUsernameFeedback" className="invalid-feedback">
     Enter owner's name
    </div>
</div>

```

```
     Enter owner's name
    </div>
</div>

<div className="col-md-6">
  <label for="location" className="form-label">City</label>
  <input type="text" className="form-control " id="validationServer03" aria-describedby="validationServer03Feedback" required
   onChange={(e)=>{
    setlocation(e.target.value)

  }}/>

  <div id="validationServer03Feedback" className="invalid-feedback">
   Please provide a valid city.
  </div>

</div>

<div className="col-md-3">
  <label for="phone" className="form-label">Phone   pattern:(xxxxxxxxxx)</label>
  <input type="tel" className="form-control " id="validationServer05" pattern="^\d{3}\d{7}$" aria-describedby="validationServer05Feedback"  required
   onChange={(e)=>{
    setphone(e.target.value)

  }}/>

  <div id="validationServer05Feedback" className="invalid-feedback">
   Please provide a valid phone number.
  </div>

</div>
<div className="col-md-4">
  <label for="owner" className="form-label">image link</label>
   <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
   onChange={(e)=>{
    setlink(e.target.value)
  }}/>
```

```
159
160                 <div id="validationServer05Feedback" className="invalid-feedback">
161                   Please provide a valid phone number.
162                 </div>
163
164               </div>
165               <div className="col-md-4">
166                 <label for="owner" className="form-label">image link</label>
167                   <input type="text" className="form-control " id="validationServer02" aria-describedby="inputGroupPrepend3 validationServerUsernameFeedback" required
168                   onChange={(e)=>{
169                     setlink(e.target.value)
170                   }}/>
171                     <div id="validationServerUsernameFeedback" className="invalid-feedback">
172                       Enter img link name
173                     </div>
174               </div>
175              <br/>
176               <div className="col-12">
177                 <button className="btn btn-primary" type="submit">Submit form</button>
178               </div>
179             </form>
180           </div>
181         </div>
182           )
183       }
```

```
1   import React, { Component } from "react";
2   // This will require to npm install axios
3   import axios from 'axios';
4   import { Link } from "react-router-dom";
5
6   const Record = (props) => (
7     <tr>
8
9       <td>{props.record.accommodation_ID}</td>
10      <td>{props.record.name}</td>
11      <td>{props.record.location}</td>
12      <td>{props.record.type}</td>
13      <td>{props.record.price}</td>
14      <td>{props.record.owner}</td>
15      <td>{props.record.phone}</td>
16      <td>
17        <Link to={"/update/" + props.record._id}>Edit</Link> |
18        <a
19          href="/admin_panel/accommodation_admin"
20          onClick={() => {
21            props.deleteRecord(props.record._id);
22            this.props.history.push("/admin_panel/accommodation_admin");
23          }}
24        >
25          Delete
26        </a>
27      </td>
28    </tr>
29  );
30
31  export default class RecordList extends Component {
32    // This is the constructor that shall store our data retrieved from the database
33    constructor(props) {
34      super(props);
35      this.deleteRecord = this.deleteRecord.bind(this);
36      this.state = { records: [] };
37    }
```

```
// This method will get the data from the database.
componentDidMount() {
  axios
    .get("http://localhost:8000/accommodation/")
    .then((response) => {
      this.setState({ records: response.data });
    })
    .catch(function (error) {
      console.log(error);
    });
}

// This method will delete a record based on the method
deleteRecord(id) {
  axios.delete("http://localhost:8000/accommodation/delete/" + id).then((response) => {
    console.log(response.data);
  });

  this.setState({
    record: this.state.records.filter((el) => el._id !== id),
  });
}

// This method will map out the users on the table
recordList() {
  return this.state.records.map((currentrecord) => {
    return (
      <Record
        record={currentrecord}
        deleteRecord={this.deleteRecord}
        key={currentrecord._id}
      />
    );
  });
}

filterData(records,searchKey){
```

```
filterData(records,searchKey){
  const result=records.filter((accommodations)=>
  accommodations.name.toLowerCase().includes(searchKey)||
  accommodations.location.toLowerCase().includes(searchKey)||
  accommodations.type.toLowerCase().includes(searchKey)


  )
  this.setState({records:result})
}

handleSerchArea=(e) =>{

  const searchKey=e.currentTarget.value;
  console.log(searchKey);
  axios
  .get("http://localhost:8000/accommodation/")
  .then((response) => {
    this.filterData(response.data,searchKey);
    console.log("successfull");
  });


}

// This following section will display the table with the records of individuals.
render() {
  return (
    <div>
      <div><nav className="navbar navbar-expand-lg navbar-light bg-light">
        <div className= "container-fluid">
        <a className="navbar-brand" href="/admin_panel/home"style={{color:"red "}}>Accommodation Management</a>
<button className="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="
  <span className="navbar-toggler-icon"></span>
</button>

<div className="collapse navbar-collapse" id="navbarSupportedContent">
```

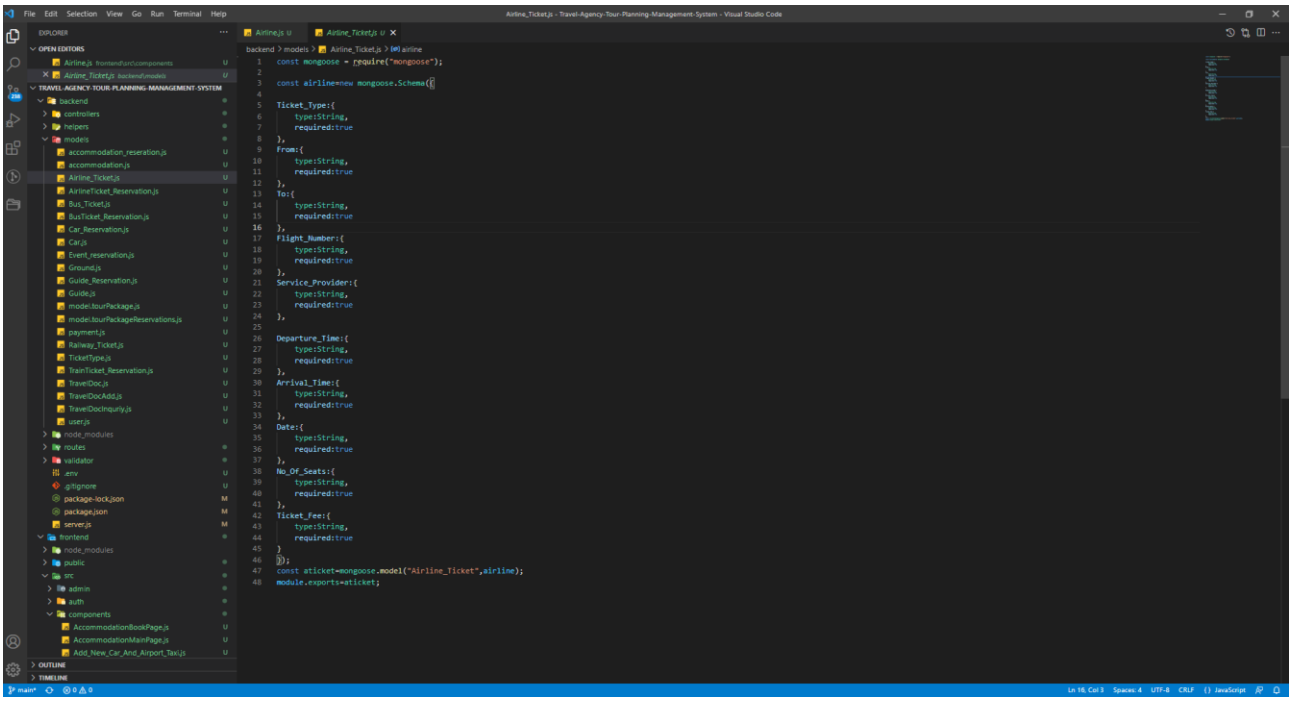
```
110
111     <div className="collapse navbar-collapse" id="navbarSupportedContent">
112       <ul className="navbar-nav mr-auto">
113         <li className="nav-item active">
114           <a className="nav-link" href="/admin_panel/accommodation_admin">All Accommadations <span className="sr-only">(current)
115         </li>
116         <li className="nav-item">
117           <a className="nav-link" href="/admin_panel/accommodation_admin/addnew">Add new Accommodation</a>
118         </li>
119         <li className="nav-item">
120           <a className="nav-link" href="/admin_panel/accommodation_admin/accommodation_servations">All Accommodation reservation
121         </li>
122       </ul>
123     </div>
124             </div>
125
126   </nav></div>
127   <nav class="navbar navbar-light bg-light">
128                         <form class="form-inline">
129                           <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search" onChang
130
131                         </form>
132                         </nav>
133
134           <br/><br/><br/><br/>
135         <div className="container">
136
137         <div><h3> Accommodation Record List</h3></div>
138          <table className="table table-striped" style={{ marginTop: 20 }}>
139            <thead>
140              <tr>
141
142                <th>Reamrk Id</th>
143                <th>name</th>
144                <th>location</th>
145                <th>type</th>
146                <th>price</th>
```

```
126   </nav></div>
127   <nav class="navbar navbar-light bg-light">
128                         <form class="form-inline">
129                           <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search" onChange={this.handleSerchArea} />
130
131                         </form>
132                         </nav>
133
134           <br/><br/><br/><br/>
135         <div className="container">
136
137         <div><h3> Accommodation Record List</h3></div>
138          <table className="table table-striped" style={{ marginTop: 20 }}>
139            <thead>
140              <tr>
141
142                <th>Reamrk Id</th>
143                <th>name</th>
144                <th>location</th>
145                <th>type</th>
146                <th>price</th>
147                <th>owner</th>
148                <th>phone</th>
149                <th>Action</th>
150              </tr>
151            </thead>
152            <tbody>{this.recordList()}</tbody>
153          </table>
154        </div>
155        </div>
156      );
157    }
158  }
159
```

Figure 95

2.3.3.3 Reservation of Ticket Management

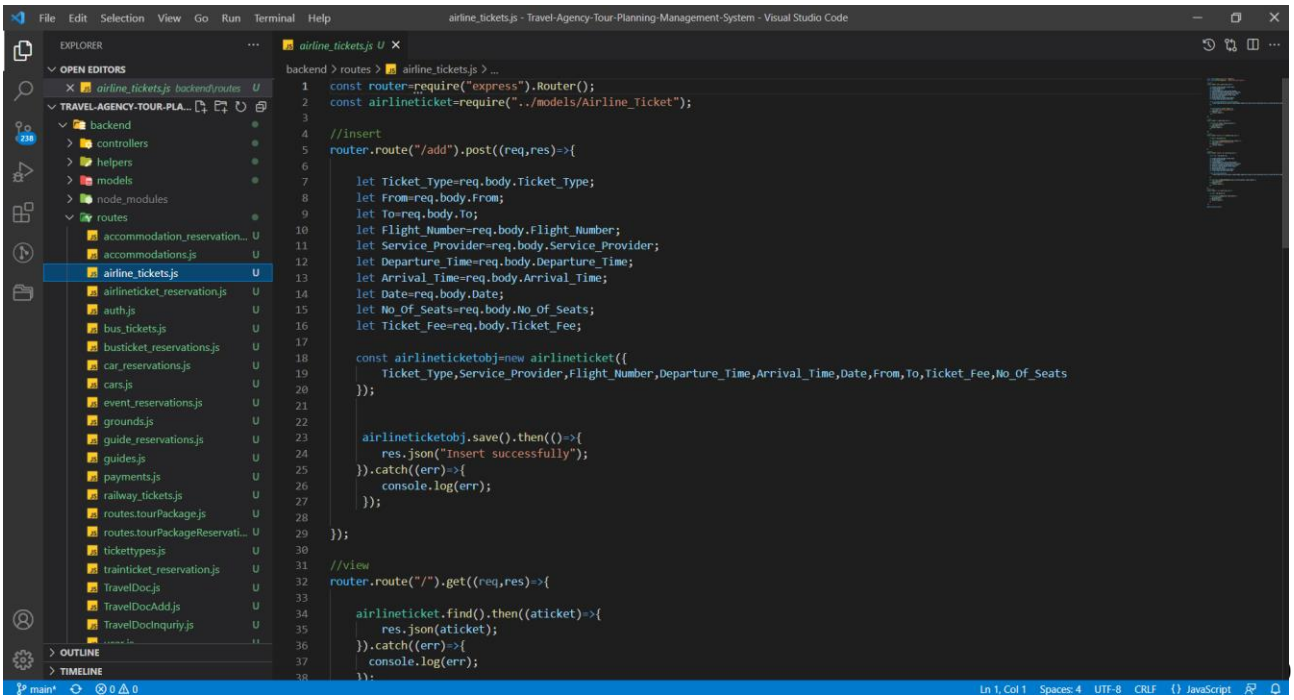
Models



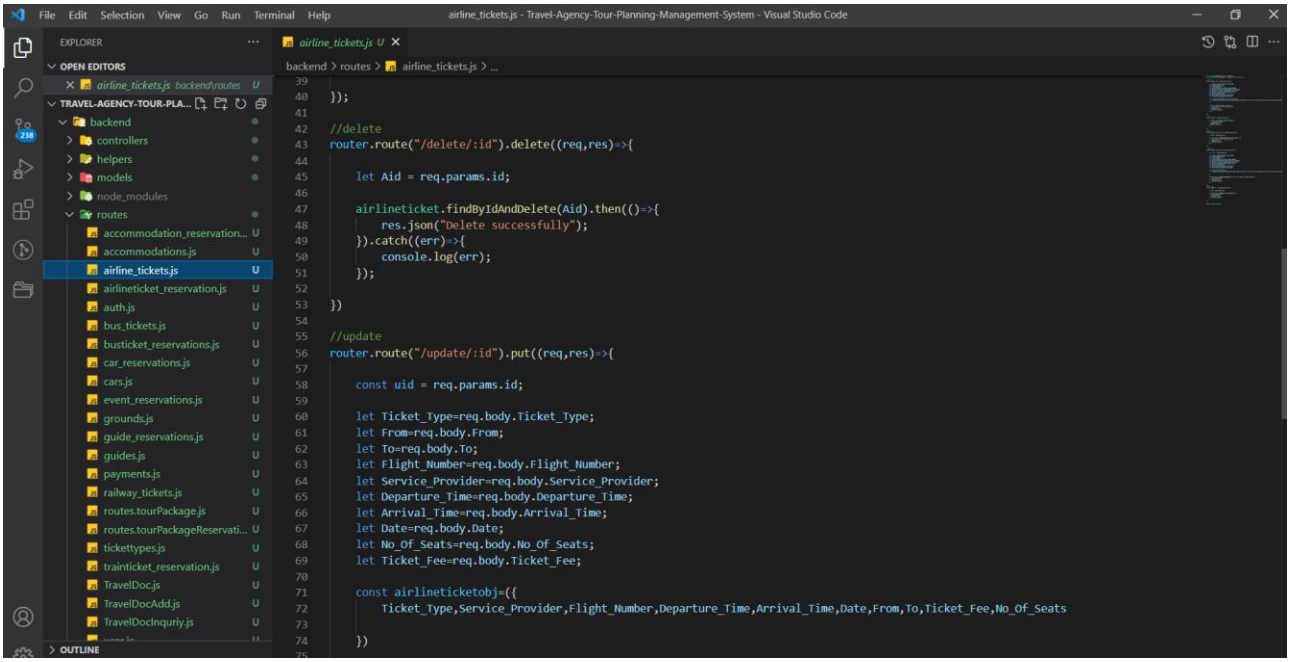
```
const mongoose = require("mongoose");

const airline=new mongoose.Schema({

Ticket_Type:{
    type:String,
    required:true
},
From:{
    type:String,
    required:true
},
To:{
    type:String,
    required:true
},
Flight_Number:{
    type:String,
    required:true
},
Service_Provider:{
    type:String,
    required:true
},

Departure_Time:{
    type:String,
    required:true
},
Arrival_Time:{
    type:String,
    required:true
},
Date:{
    type:String,
    required:true
},
No_Of_Seats:{
    type:String,
    required:true
},
Ticket_Fee:{
    type:String,
    required:true
}
});
const aticket=mongoose.model("Airline_Ticket",airline);
module.exports=aticket;
```

Figure 96

Routes



```
const router=require("express").Router();
const airlineticket=require("../models/Airline_Ticket");

//insert
router.route("/add").post((req,res)=>{

    let Ticket_Type=req.body.Ticket_Type;
    let From=req.body.From;
    let To=req.body.To;
    let Flight_Number=req.body.Flight_Number;
    let Service_Provider=req.body.Service_Provider;
    let Departure_Time=req.body.Departure_Time;
    let Arrival_Time=req.body.Arrival_Time;
    let Date=req.body.Date;
    let No_Of_Seats=req.body.No_Of_Seats;
    let Ticket_Fee=req.body.Ticket_Fee;

    const airlineticketobj=new airlineticket({
        Ticket_Type,Service_Provider,Flight_Number,Departure_Time,Arrival_Time,Date,From,To,Ticket_Fee,No_Of_Seats
    });


    airlineticketobj.save().then(()=>{
        res.json("Insert successfully");
    }).catch((err)=>{
        console.log(err);
    });

});

//view
router.route("/").get((req,res)=>{

    airlineticket.find().then((aticket)=>{
        res.json(aticket);
    }).catch((err)=>{
      console.log(err);
    });
```


File Edit Selection View Go Run Terminal Help
airline_tickets.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
airline_tickets.js backend\routes U
TRAVEL-AGENCY-TOUR-PLA...
backend
controllers
helpers
models
node_modules
routes
accommodation_reservation... U
accommodations.js U
airline_tickets.js U
airlineticket_reservation.js U
auth.js U
bus_tickets.js U
busticket_reservations.js U
car_reservations.js U
cars.js U
event_reservations.js U
grounds.js U
guide_reservations.js U
guides.js U
payments.js U
railway_tickets.js U
routes.tourPackage.js U
routes.tourPackageReservati... U
tickettypes.js U
trainticket_reservation.js U
TravelDoc.js U
TravelDocAdd.js U
TravelDocInquriy.js U
OUTLINE
airline_tickets.js U
backend > routes > airline_tickets.js > ...
39
40 });
41
42 //delete
43 router.route("/delete/:id").delete((req,res)=>{
44
45 let Aid = req.params.id;
46
47 airlineticket.findByIdAndDelete(Aid).then(()=>{
48 res.json("Delete successfully");
49 }).catch((err)=>{
50 console.log(err);
51 });
52
53 })
54
55 //update
56 router.route("/update/:id").put((req,res)=>{
57
58 const uid = req.params.id;
59
60 let Ticket_Type=req.body.Ticket_Type;
61 let From=req.body.From;
62 let To=req.body.To;
63 let Flight_Number=req.body.Flight_Number;
64 let Service_Provider=req.body.Service_Provider;
65 let Departure_Time=req.body.Departure_Time;
66 let Arrival_Time=req.body.Arrival_Time;
67 let Date=req.body.Date;
68 let No_Of_Seats=req.body.No_Of_Seats;
69 let Ticket_Fee=req.body.Ticket_Fee;
70
71 const airlineticketobj=({
72 Ticket_Type,Service_Provider,Flight_Number,Departure_Time,Arrival_Time,Date,From,To,Ticket_Fee,No_Of_Seats
73
74 })

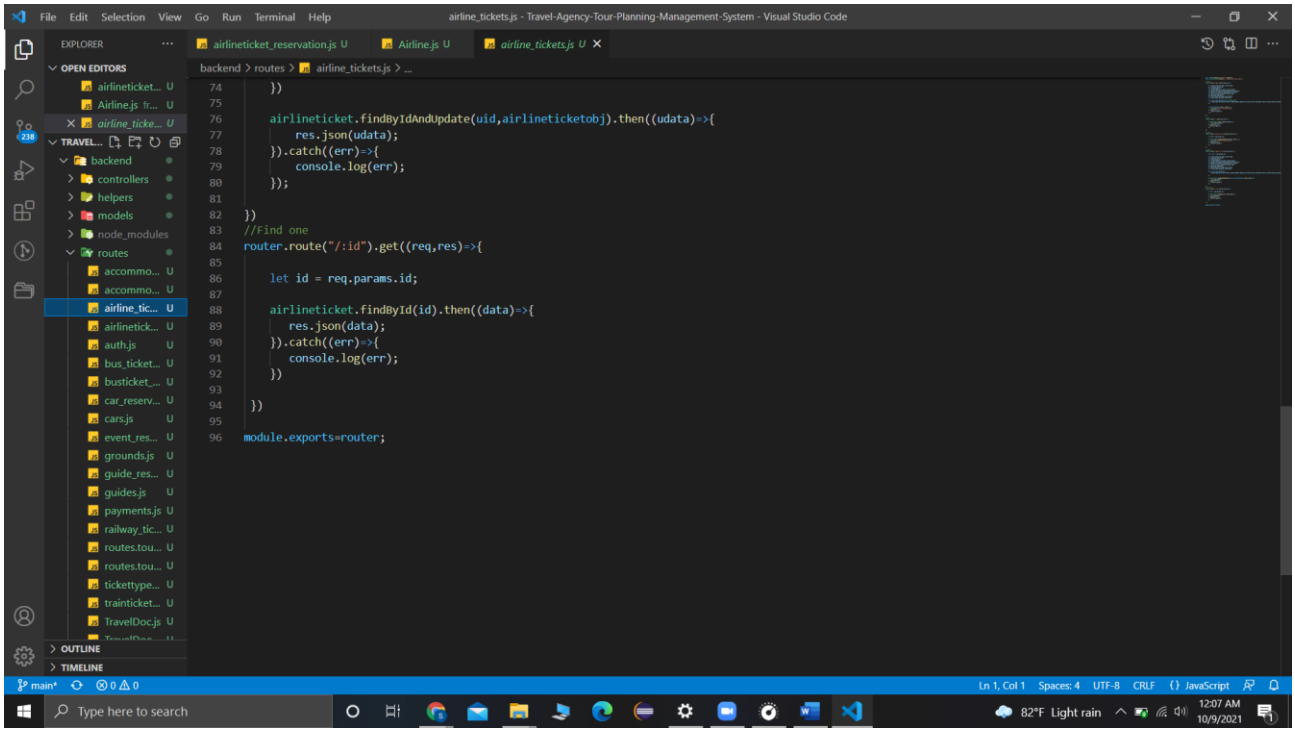
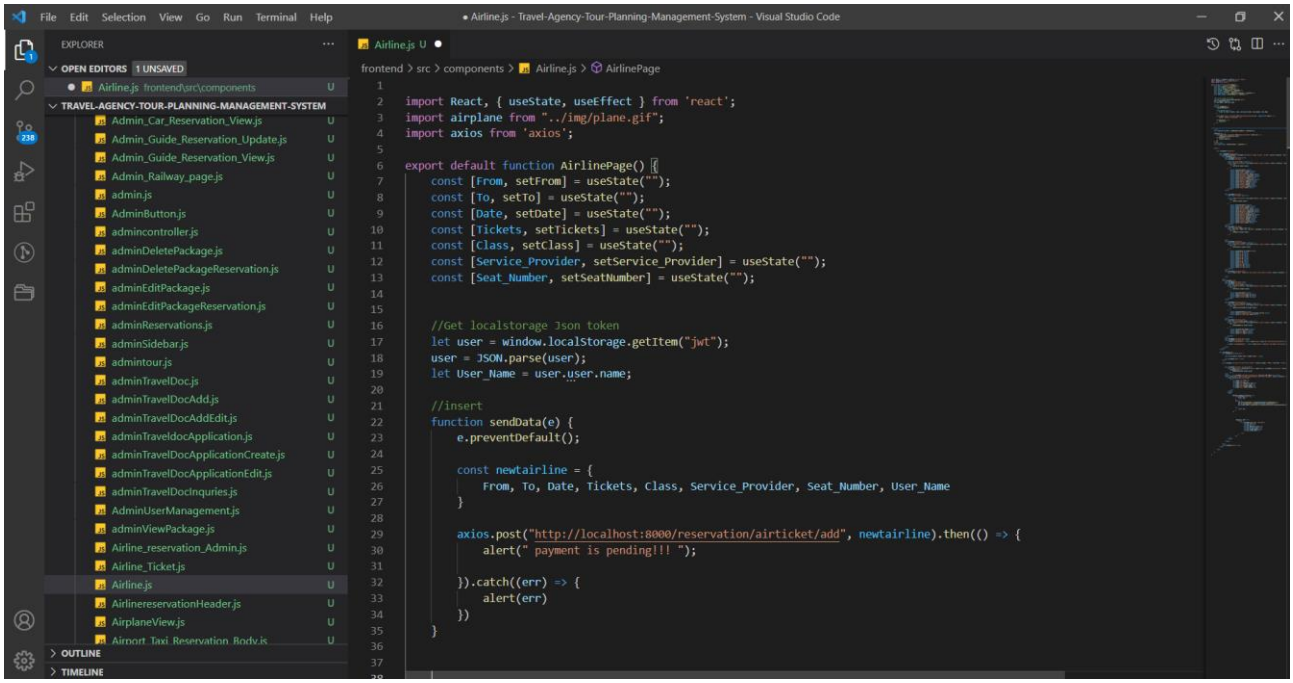
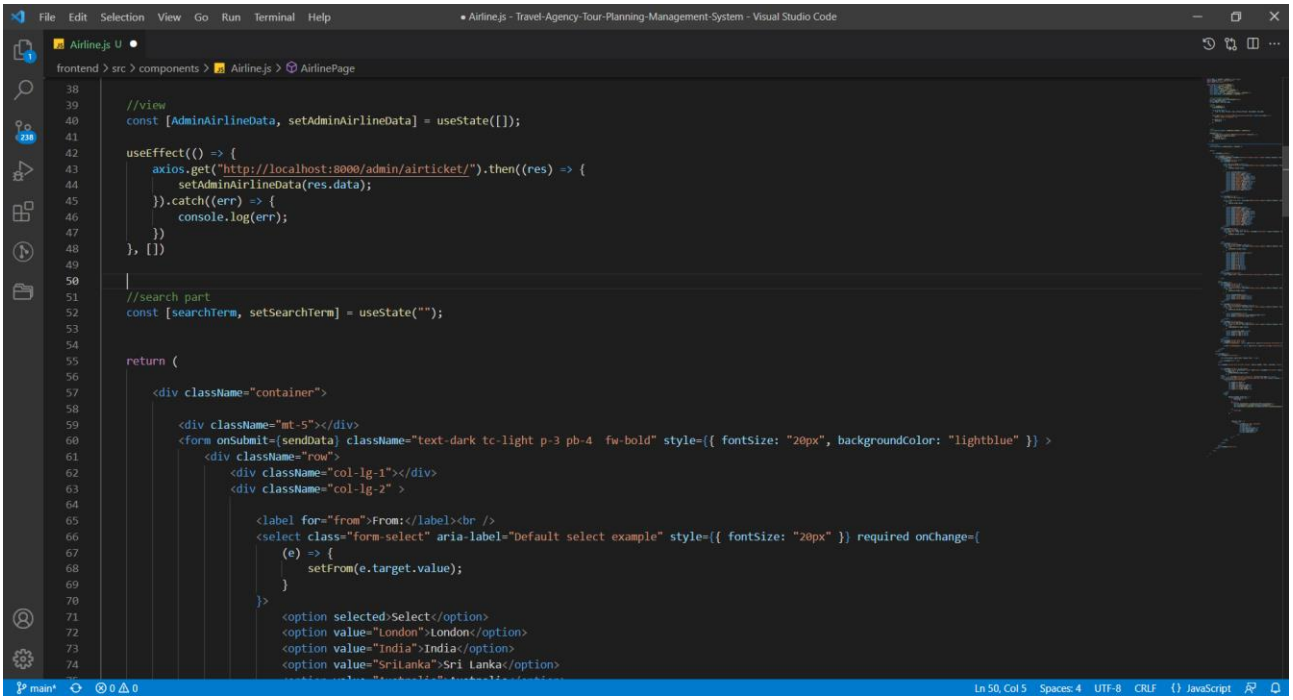

File Edit Selection View Go Run Terminal Help
airline_tickets.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
airlineticket_reservation.js U
Airline.js U
airline_tickets.js U
backend > routes > airline_tickets.js > ...
74 })
75
76 airlineticket.findByIdAndUpdate(uid,airlineticketobj).then((udata)=>{
77 res.json(udata);
78 }).catch((err)=>{
79 console.log(err);
80 });
81
82 })
83 //Find one
84 router.route("/:id").get((req,res)=>{
85
86 let id = req.params.id;
87
88 airlineticket.findById(id).then((data)=>{
89 res.json(data);
90 }).catch((err)=>{
91 console.log(err);
92 })
93
94 })
95
96 module.exports=router;
OUTLINE
TIMELINE
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Type here to search
82°F Light rain
12:07 AM
10/9/2021

Figure 97

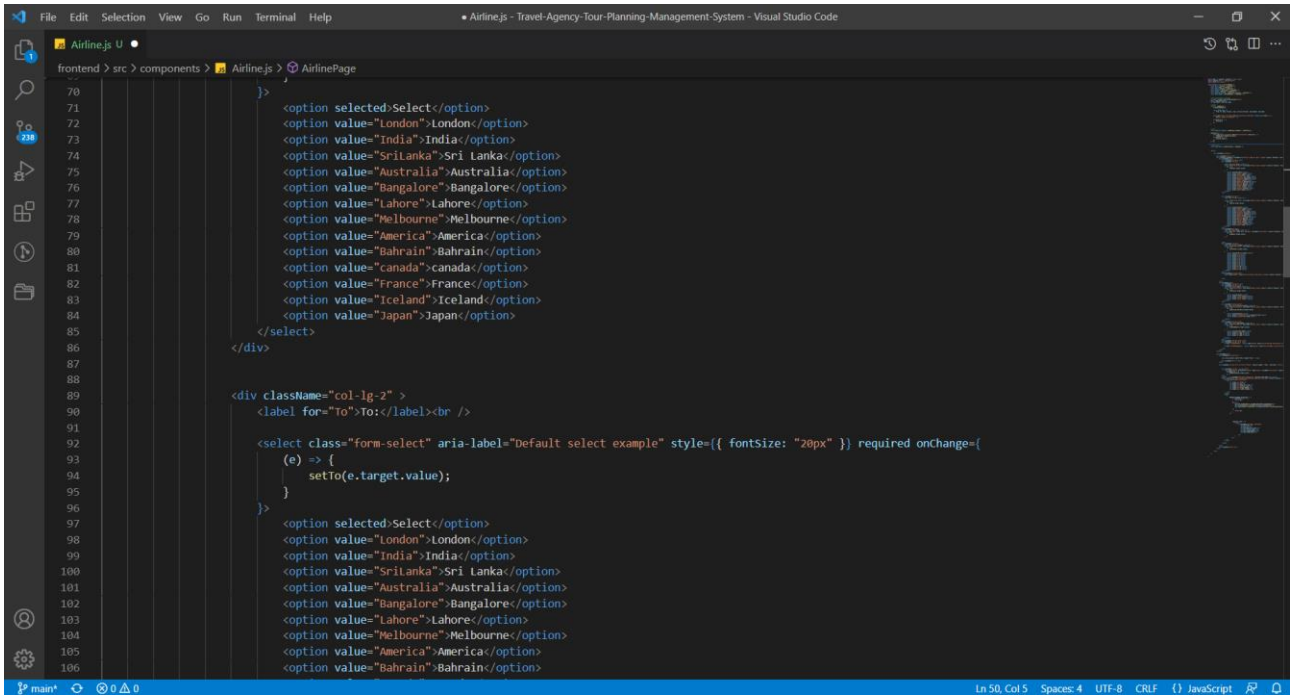
Frontend


File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
Airline.js frontend\src\components U
TRAVEL-AGENCY-TOUR-PLANNING-MANAGEMENT-SYSTEM
Admin_Car_Reservation_View.js U
Admin_Guide_Reservation_Update.js U
Admin_Guide_Reservation_View.js U
Admin_Railway_page.js U
admin.js U
AdminButton.js U
admincontroller.js U
adminDeletePackage.js U
adminDeletePackageReservation.js U
adminEditPackage.js U
adminEditPackageReservation.js U
adminReservations.js U
adminSidebar.js U
admintour.js U
adminTravelDoc.js U
adminTravelDocAdd.js U
adminTravelDocAddEdit.js U
adminTraveldocApplication.js U
adminTravelDocApplicationCreate.js U
adminTravelDocApplicationEdit.js U
adminTravelDocInquries.js U
AdminUserManagement.js U
adminViewPackage.js U
Airline_reservation_Admin.js U
Airline_Ticket.js U
Airline.js U
AirlinereservationHeader.js U
AirplaneView.js U
Airport_Taxi_Reservation_Body.js U
OUTLINE
TIMELINE
Airline.js U
frontend > src > components > Airline.js > AirlinePage
import React, { useState, useEffect } from 'react';
import airplane from "../img/plane.gif";
import axios from 'axios';
export default function AirlinePage() {
const [From, setFrom] = useState("");
const [To, setTo] = useState("");
const [Date, setDate] = useState("");
const [Tickets, setTickets] = useState("");
const [Class, setClass] = useState("");
const [Service_Provider, setService_Provider] = useState("");
const [Seat_Number, setSeatNumber] = useState("");
//Get localstorage Json token
let user = window.localStorage.getItem("jwt");
user = JSON.parse(user);
let User_Name = user.user.name;
//insert
function sendData(e) {
e.preventDefault();
const newtairline = {
From, To, Date, Tickets, Class, Service_Provider, Seat_Number, User_Name
}
axios.post("http://localhost:8000/reservation/airticket/add", newtairline).then(() => {
alert(" payment is pending!!! ");
}).catch((err) => {
alert(err)
})
}

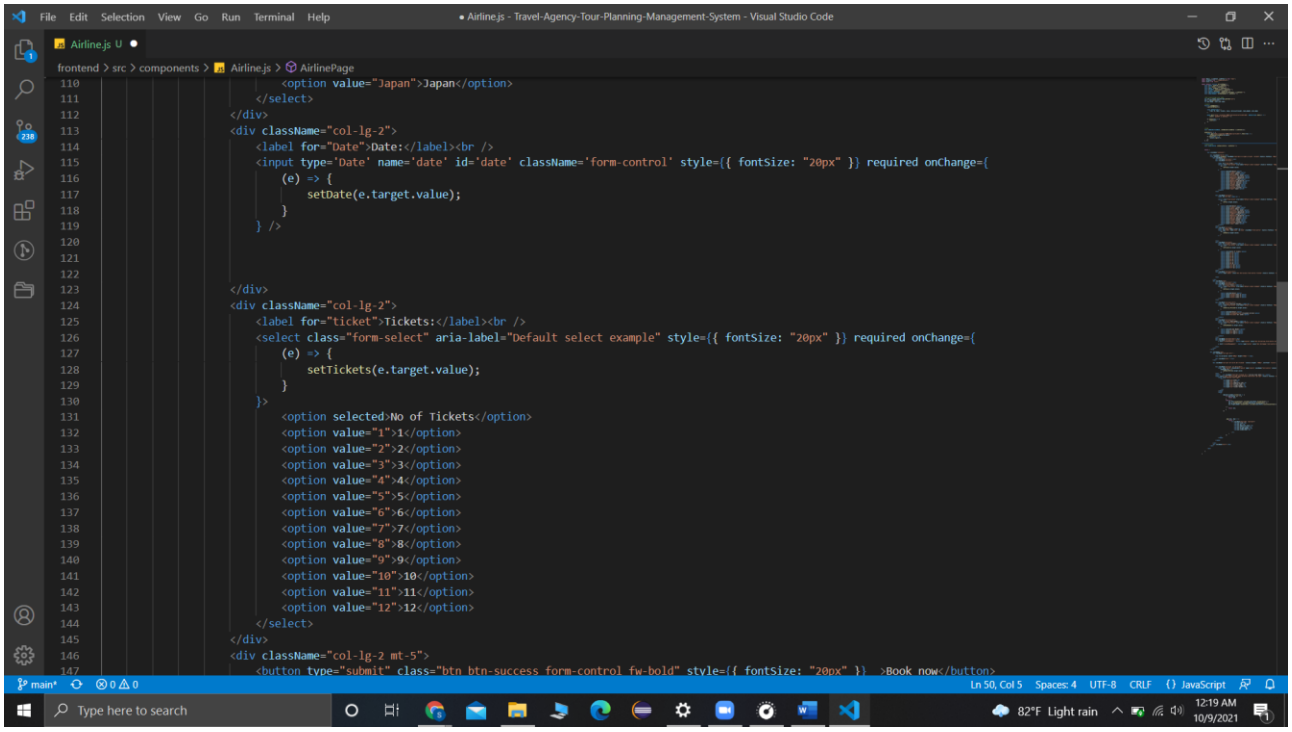

File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage
//view
const [AdminAirlineData, setAdminAirlineData] = useState([]);
useEffect(() => {
axios.get("http://localhost:8000/admin/airticket/").then((res) => {
setAdminAirlineData(res.data);
}).catch((err) => {
console.log(err);
})
}, [])
//search part
const [searchTerm, setSearchTerm] = useState("");
return (

<form onSubmit={sendData} className="text-dark tc-light p-3 pb-4 fw-bold" style={{ fontSize: "20px", backgroundColor: "lightblue" }} >

<label for="from">From:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setFrom(e.target.value);
}
}>
<option selected>Select</option>
<option value="London">London</option>
<option value="India">India</option>
<option value="SriLanka">Sri Lanka</option>
main*
Ln 50, Col 5 Spaces: 4 UTF-8 CRLF {} JavaScript


File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage
}>
<option selected>Select</option>
<option value="London">London</option>
<option value="India">India</option>
<option value="SriLanka">Sri Lanka</option>
<option value="Australia">Australia</option>
<option value="Bangalore">Bangalore</option>
<option value="Lahore">Lahore</option>
<option value="Melbourne">Melbourne</option>
<option value="America">America</option>
<option value="Bahrain">Bahrain</option>
<option value="canada">canada</option>
<option value="France">France</option>
<option value="Iceland">Iceland</option>
<option value="Japan">Japan</option>
</select>

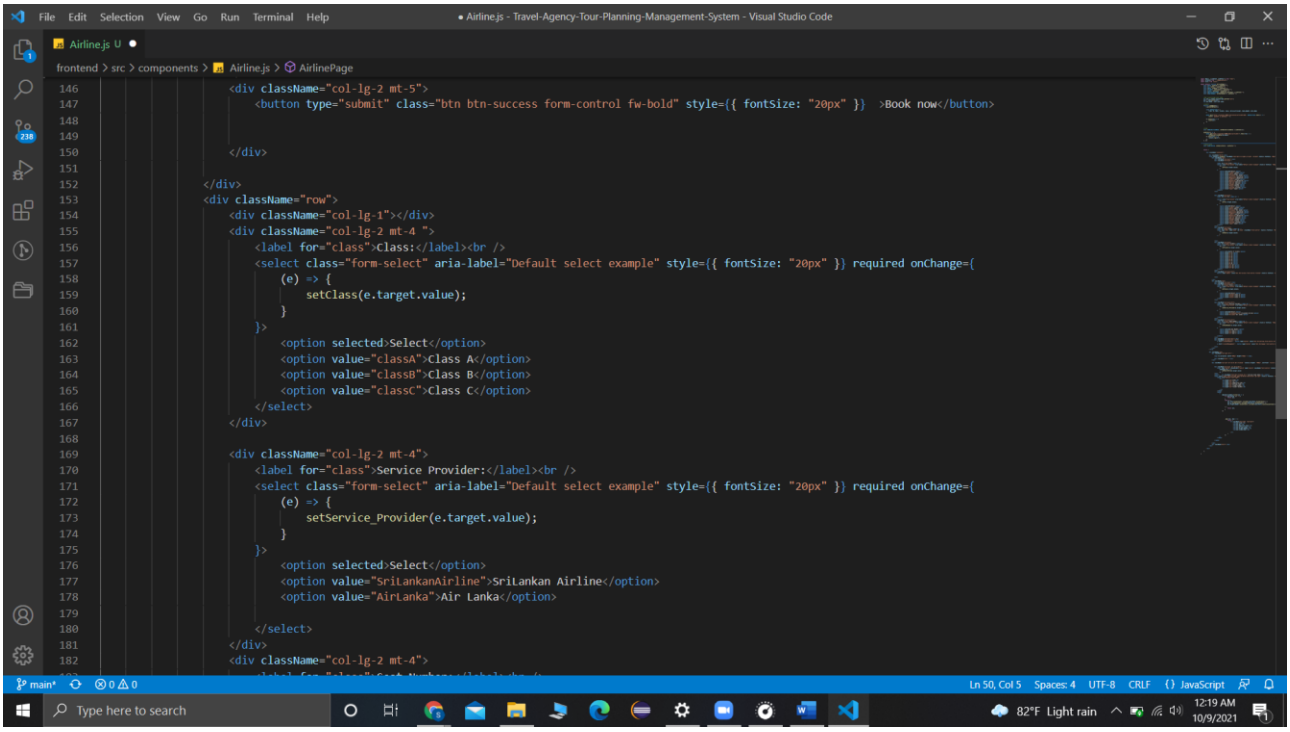
<label for="To">To:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setTo(e.target.value);
}
}>
<option selected>Select</option>
<option value="London">London</option>
<option value="India">India</option>
<option value="SriLanka">Sri Lanka</option>
<option value="Australia">Australia</option>
<option value="Bangalore">Bangalore</option>
<option value="Lahore">Lahore</option>
<option value="Melbourne">Melbourne</option>
<option value="America">America</option>
<option value="Bahrain">Bahrain</option>
main*
Ln 50, Col 5 Spaces: 4 UTF-8 CRLF {} JavaScript


File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage
<option value="Japan">Japan</option>
</select>

<label for="Date">Date:</label>
<input type='Date' name='date' id='date' className='form-control' style={{ fontSize: "20px" }} required onChange={
(e) => {
setDate(e.target.value);
}
} />

<label for="ticket">Tickets:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setTickets(e.target.value);
}
}>
<option selected>No of Tickets</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
</select>

<button type="submit" class="btn btn-success form-control fw-bold" style={{ fontSize: "20px" }} >Book now</button>
main*
Ln 50, Col 5 Spaces: 4 UTF-8 CRLF {} JavaScript
Type here to search
82°F Light rain
12:19 AM
10/9/2021

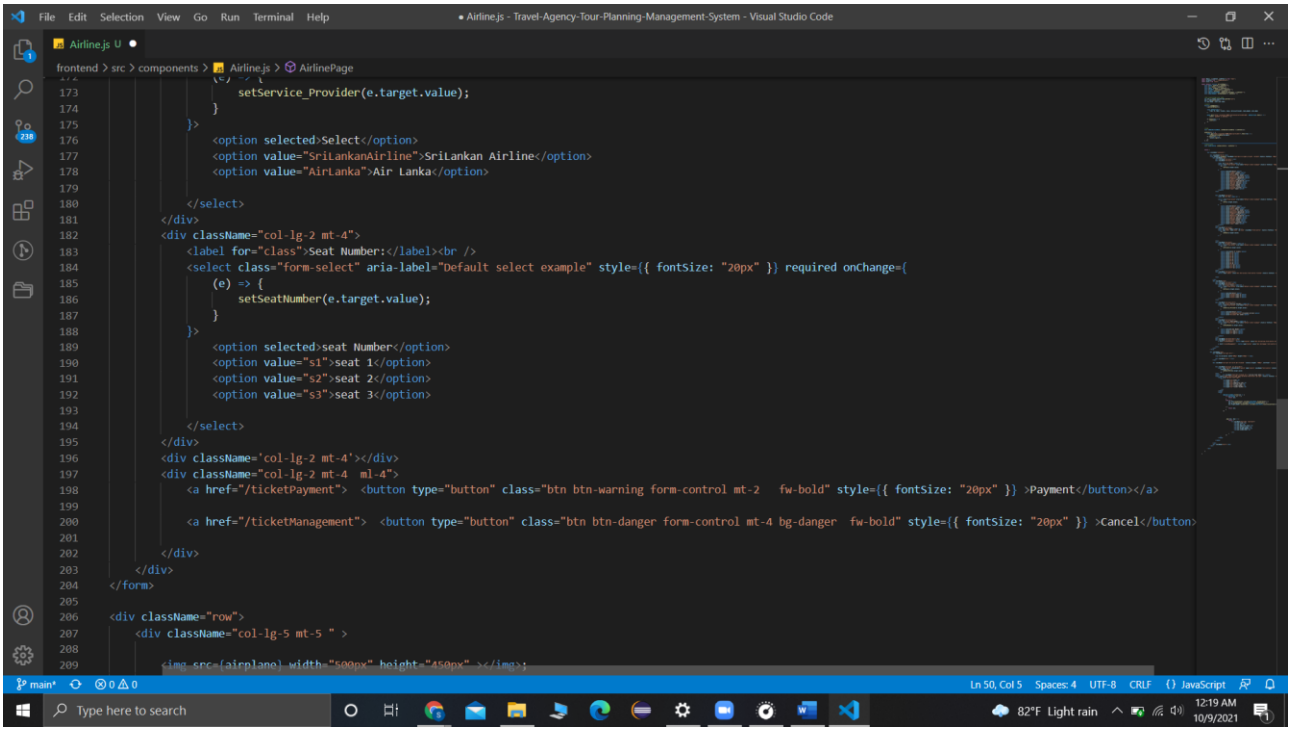

File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage

<button type="submit" class="btn btn-success form-control fw-bold" style={{ fontSize: "20px" }} >Book now</button>

<label for="class">Class:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setClass(e.target.value);
}
}>
<option selected>Select</option>
<option value="classA">Class A</option>
<option value="classB">Class B</option>
<option value="classC">Class C</option>
</select>

<label for="class">Service Provider:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setService_Provider(e.target.value);
}
}>
<option selected>Select</option>
<option value="SriLankanAirline">SriLankan Airline</option>
<option value="AirLanka">Air Lanka</option>
</select>

main*
Ln 50, Col 5 Spaces: 4 UTF-8 CRLF JavaScript
Type here to search
82°F Light rain
12:19 AM
10/9/2021

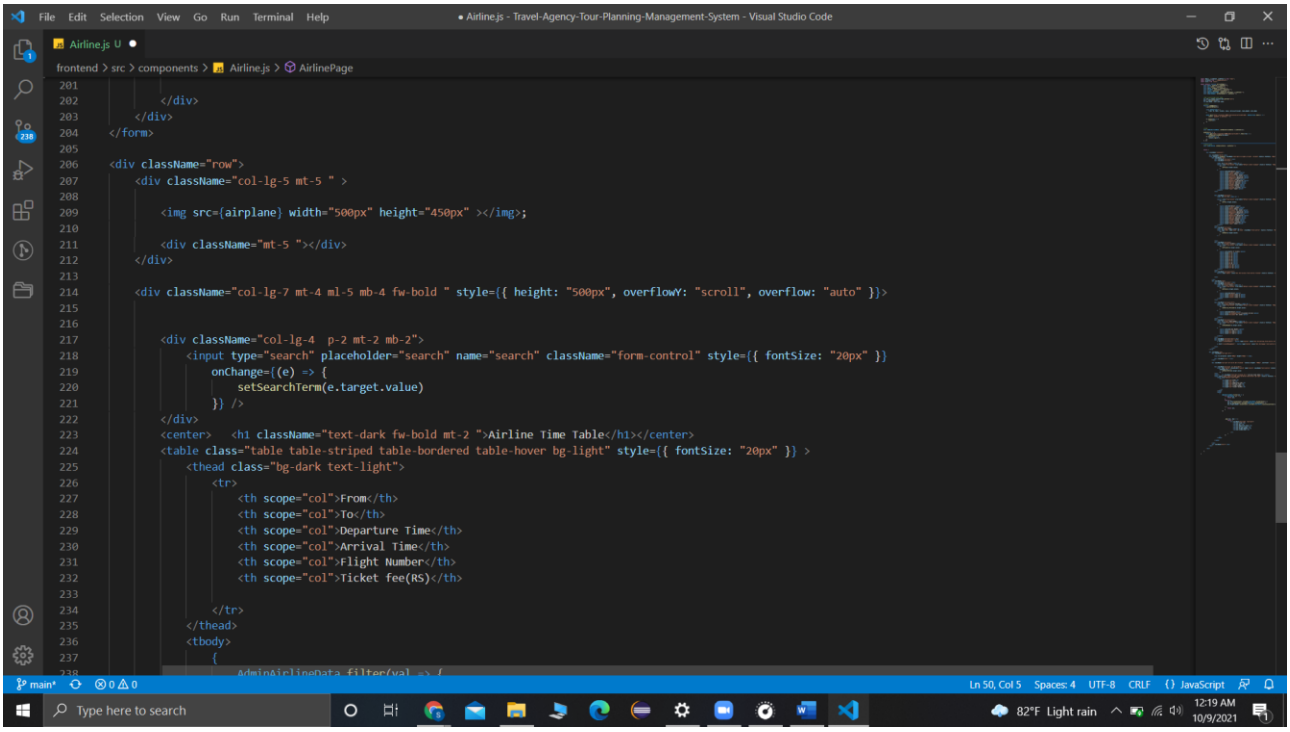

File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage
setService_Provider(e.target.value);
}
}>
<option selected>Select</option>
<option value="SriLankanAirline">SriLankan Airline</option>
<option value="AirLanka">Air Lanka</option>
</select>

<label for="class">Seat Number:</label>
<select class="form-select" aria-label="Default select example" style={{ fontSize: "20px" }} required onChange={
(e) => {
setSeatNumber(e.target.value);
}
}>
<option selected>seat Number</option>
<option value="s1">seat 1</option>
<option value="s2">seat 2</option>
<option value="s3">seat 3</option>
</select>

 <button type="button" class="btn btn-warning form-control mt-2 fw-bold" style={{ fontSize: "20px" }} >Payment</button>
<a href="/ticketManagement"> <button type="button" class="btn btn-danger form-control mt-4 bg-danger fw-bold" style={{ fontSize: "20px" }} >Cancel</button>

</form>

<img src={airplane} width="500px" height="450px" ></img>
main*
Ln 50, Col 5 Spaces: 4 UTF-8 CRLF JavaScript
Type here to search
82°F Light rain
12:19 AM
10/9/2021


File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage

</form>

<img src={airplane} width="500px" height="450px" ></img>;

<div className="col-lg-7 mt-4 ml-5 mb-4 fw-bold " style={{ height: "500px", overflowY: "scroll", overflow: "auto" }}>

<input type="search" placeholder="search" name="search" className="form-control" style={{ fontSize: "20px" }}
onChange={(e) => {
setSearchTerm(e.target.value)
}} />

 <h1 className="text-dark fw-bold mt-2 ">Airline Time Table</h1>
<table class="table table-striped table-bordered table-hover bg-light" style={{ fontSize: "20px" }} >
<thead class="bg-dark text-light">
<tr>
<th scope="col">From</th>
<th scope="col">To</th>
<th scope="col">Departure Time</th>
<th scope="col">Arrival Time</th>
<th scope="col">Flight Number</th>
<th scope="col">Ticket fee(RS)</th>
</tr>
</thead>
<tbody>
{
main*
Ln 50, Col 5
Spaces: 4
UTF-8
CRLF
JavaScript
Type here to search
82°F Light rain
12:19 AM
10/9/2021

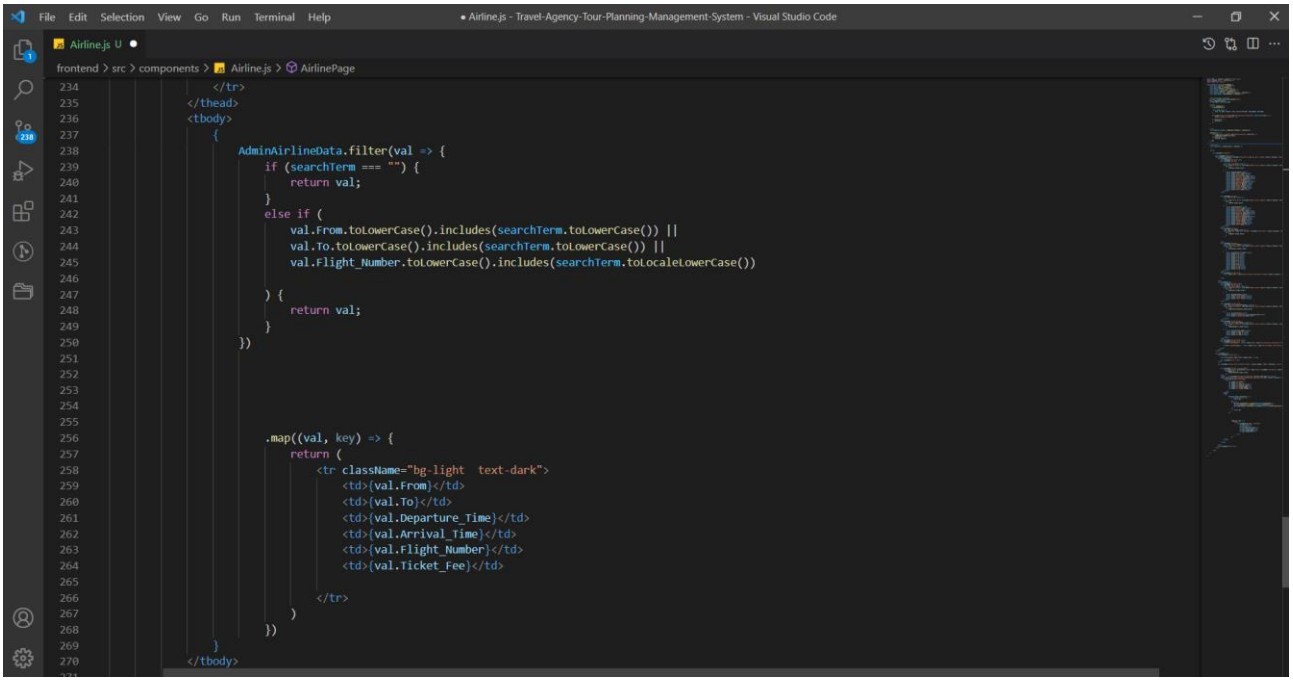

File Edit Selection View Go Run Terminal Help
• Airline.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
Airline.js U
frontend > src > components > Airline.js > AirlinePage
</tr>
</thead>
<tbody>
{
AdminAirlineData.filter(val => {
if (searchTerm === "") {
return val;
}
else if (
val.From.toLowerCase().includes(searchTerm.toLowerCase()) ||
val.To.toLowerCase().includes(searchTerm.toLowerCase()) ||
val.Flight_Number.toLowerCase().includes(searchTerm.toLocaleLowerCase())
) {
return val;
}
})
.map((val, key) => {
return (
<tr className="bg-light text-dark">
<td>{val.From}</td>
<td>{val.To}</td>
<td>{val.Departure_Time}</td>
<td>{val.Arrival_Time}</td>
<td>{val.Flight_Number}</td>
<td>{val.Ticket_Fee}</td>
</tr>
)
})
}
</tbody>

Figure 98

2.3.3.4 Reservation of Ground resource Management

Models

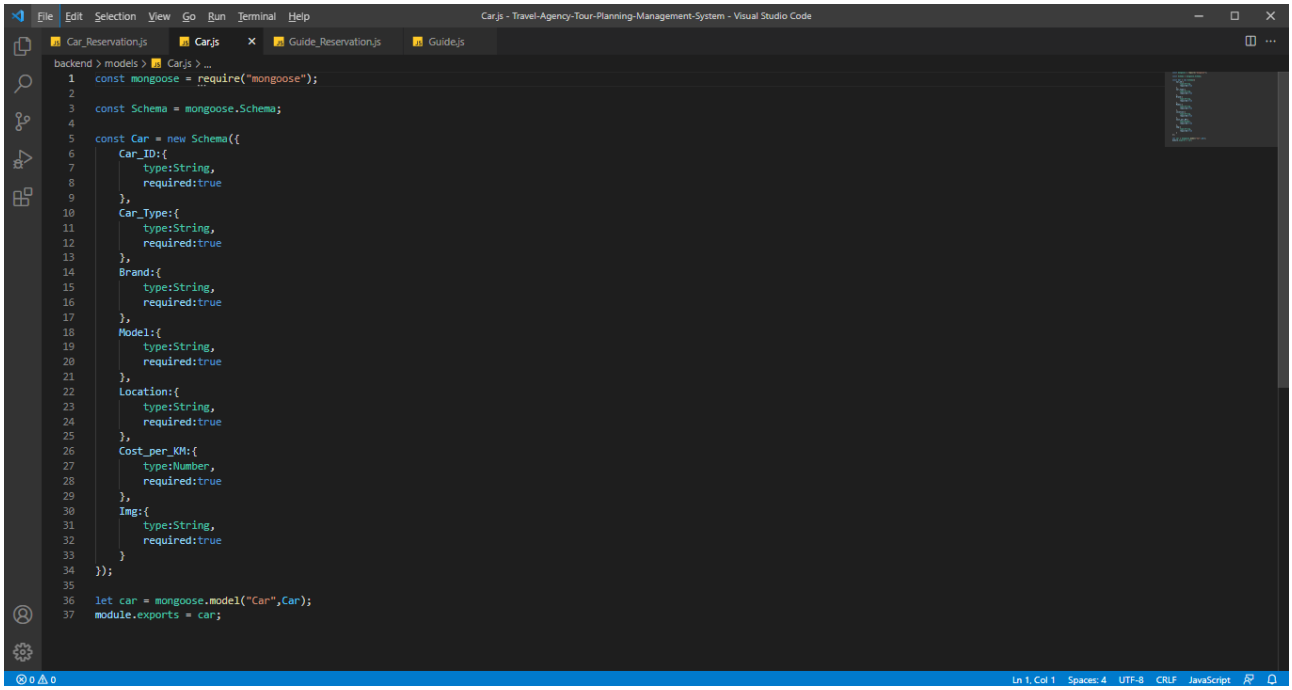
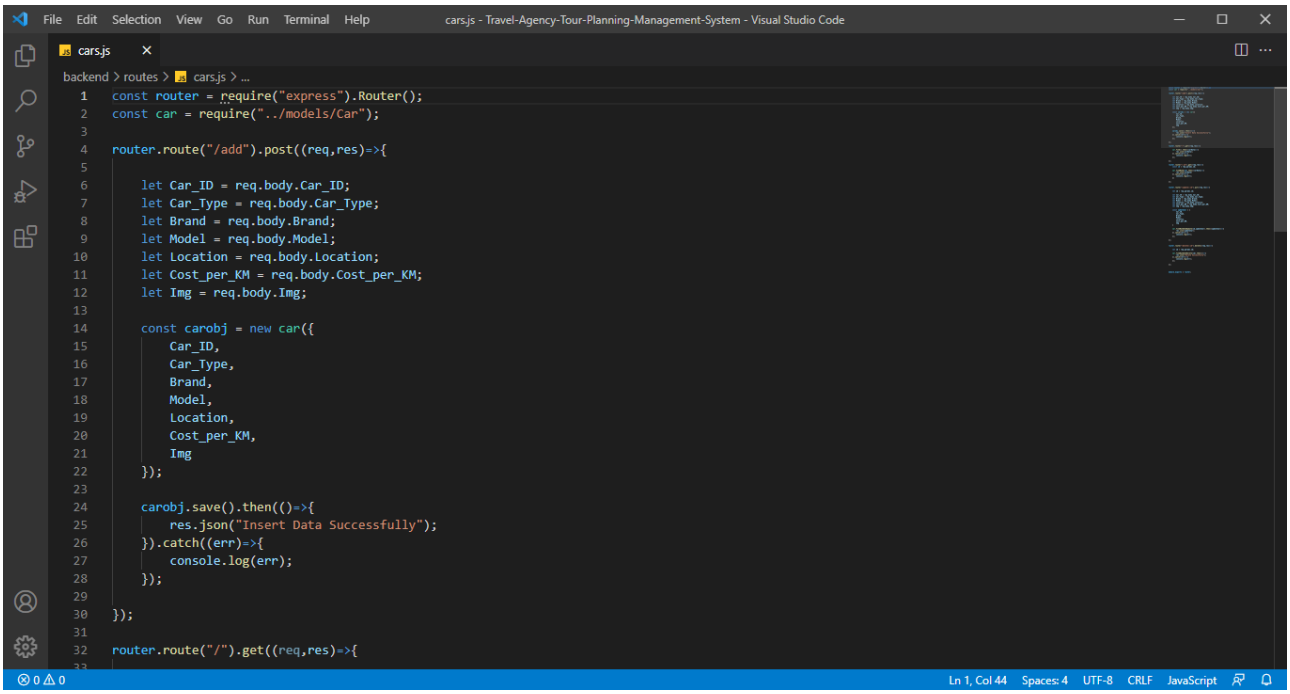
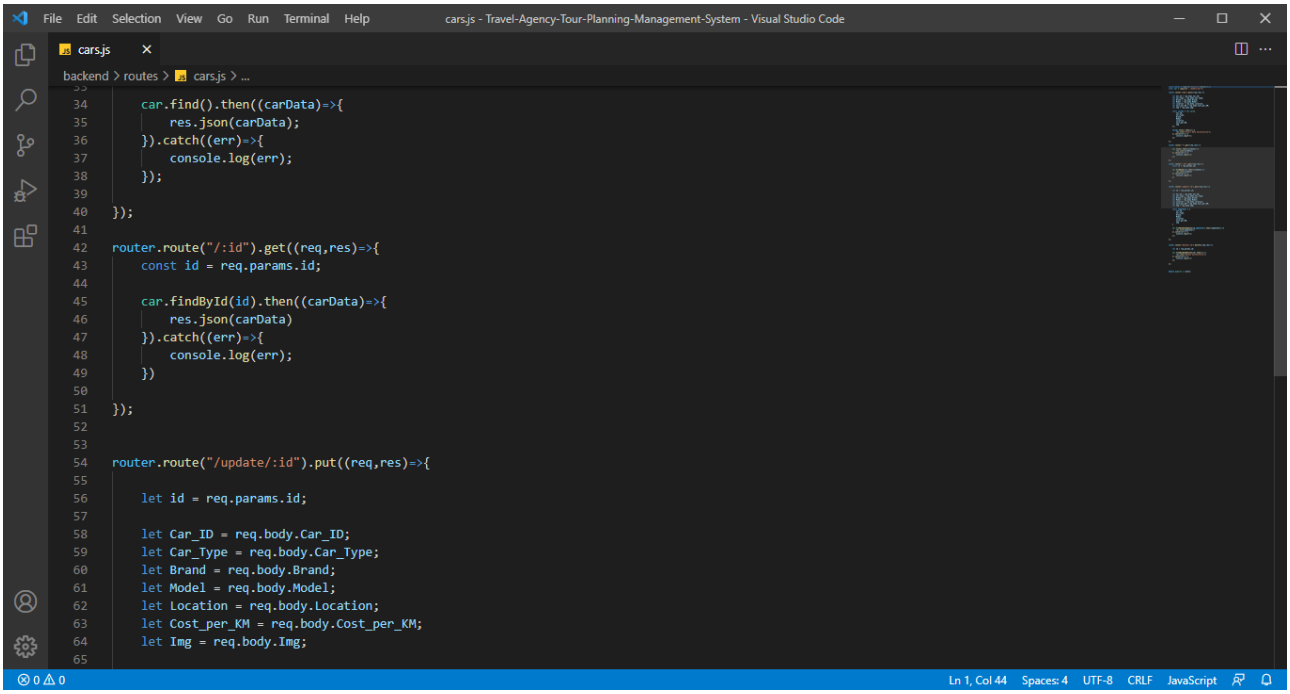


Figure 99

Routes




File Edit Selection View Go Run Terminal Help
cars.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
backend > routes > cars.js > ...
car.find().then((carData)=>{
res.json(carData);
}).catch((err)=>{
console.log(err);
});
});
router.route("/:id").get((req,res)=>{
const id = req.params.id;
car.findById(id).then((carData)=>{
res.json(carData)
}).catch((err)=>{
console.log(err);
})
});
router.route("/update/:id").put((req,res)=>{
let id = req.params.id;
let Car_ID = req.body.Car_ID;
let Car_Type = req.body.Car_Type;
let Brand = req.body.Brand;
let Model = req.body.Model;
let Location = req.body.Location;
let Cost_per_KM = req.body.Cost_per_KM;
let Img = req.body.Img;
Ln 1, Col 44 Spaces: 4 UTF-8 CRLF JavaScript

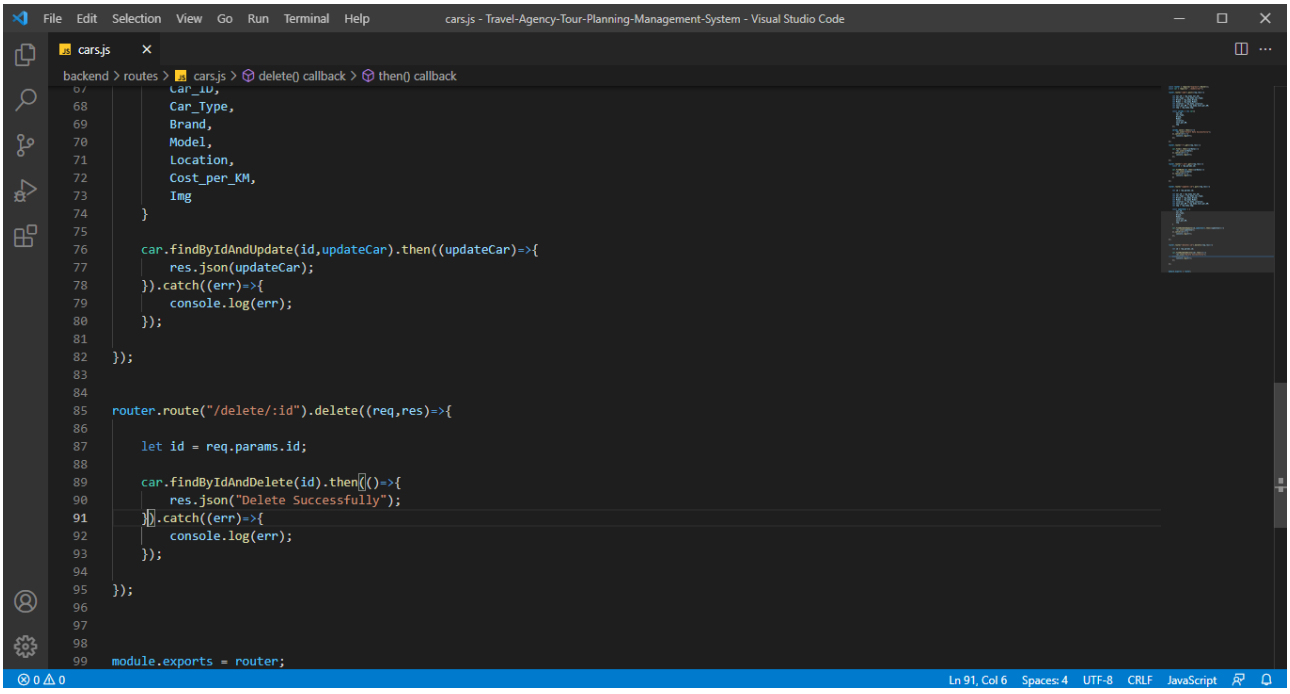
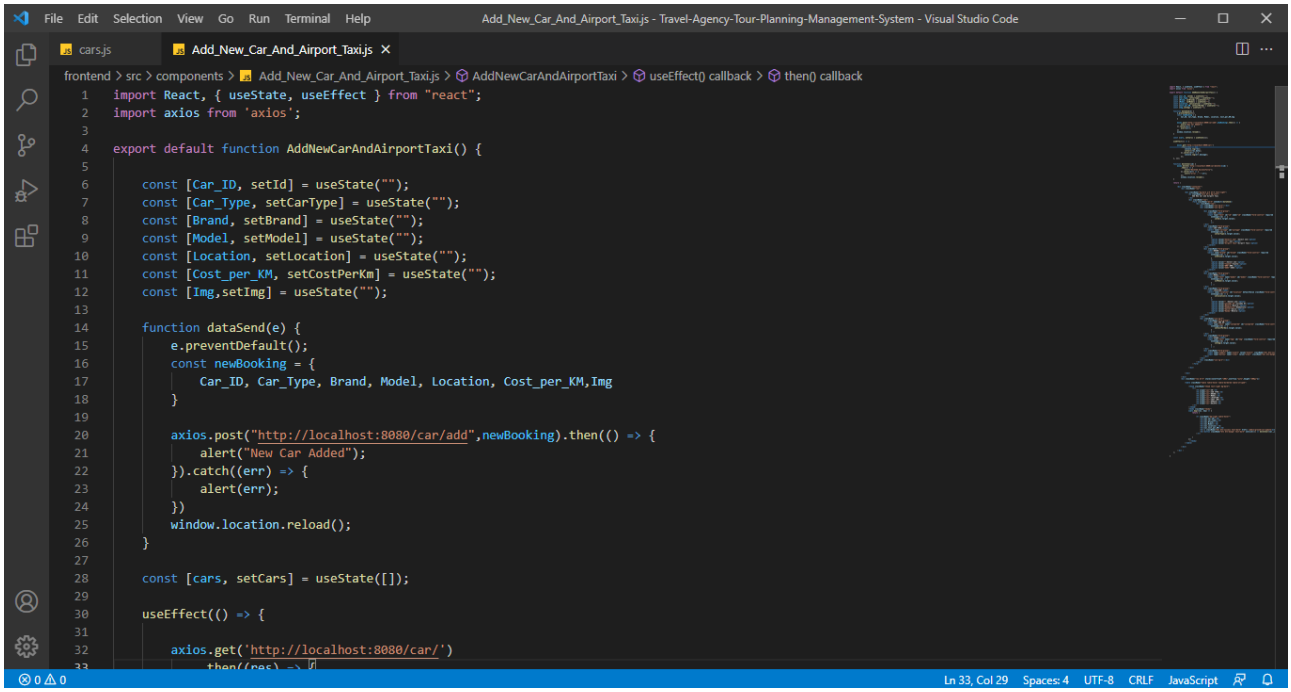
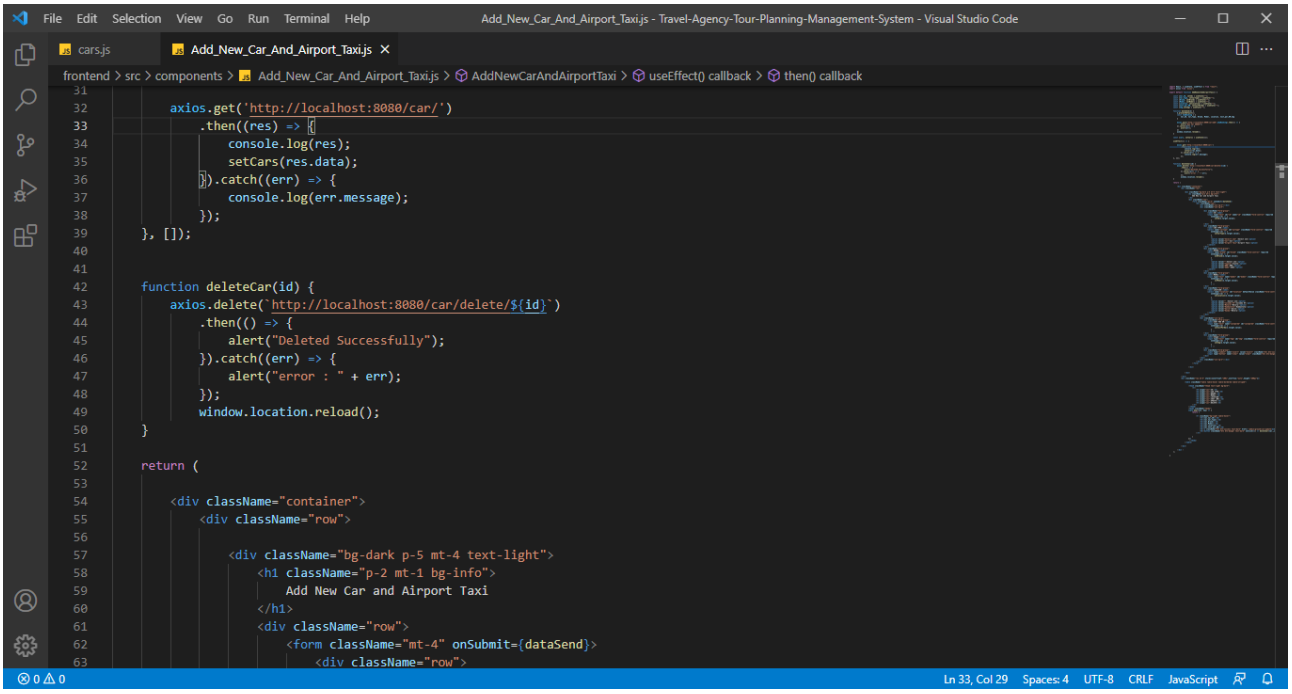

File Edit Selection View Go Run Terminal Help
cars.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
backend > routes > cars.js > delete() callback > then() callback
Car_ID,
Car_Type,
Brand,
Model,
Location,
Cost_per_KM,
Img
}
car.findByIdAndUpdate(id,updateCar).then((updateCar)=>{
res.json(updateCar);
}).catch((err)=>{
console.log(err);
});
});
router.route("/delete/:id").delete((req,res)=>{
let id = req.params.id;
car.findByIdAndDelete(id).then(()=>{
res.json("Delete Successfully");
}).catch((err)=>{
console.log(err);
});
});
module.exports = router;
Ln 91, Col 6 Spaces: 4 UTF-8 CRLF JavaScript

Figure 100

Frontend

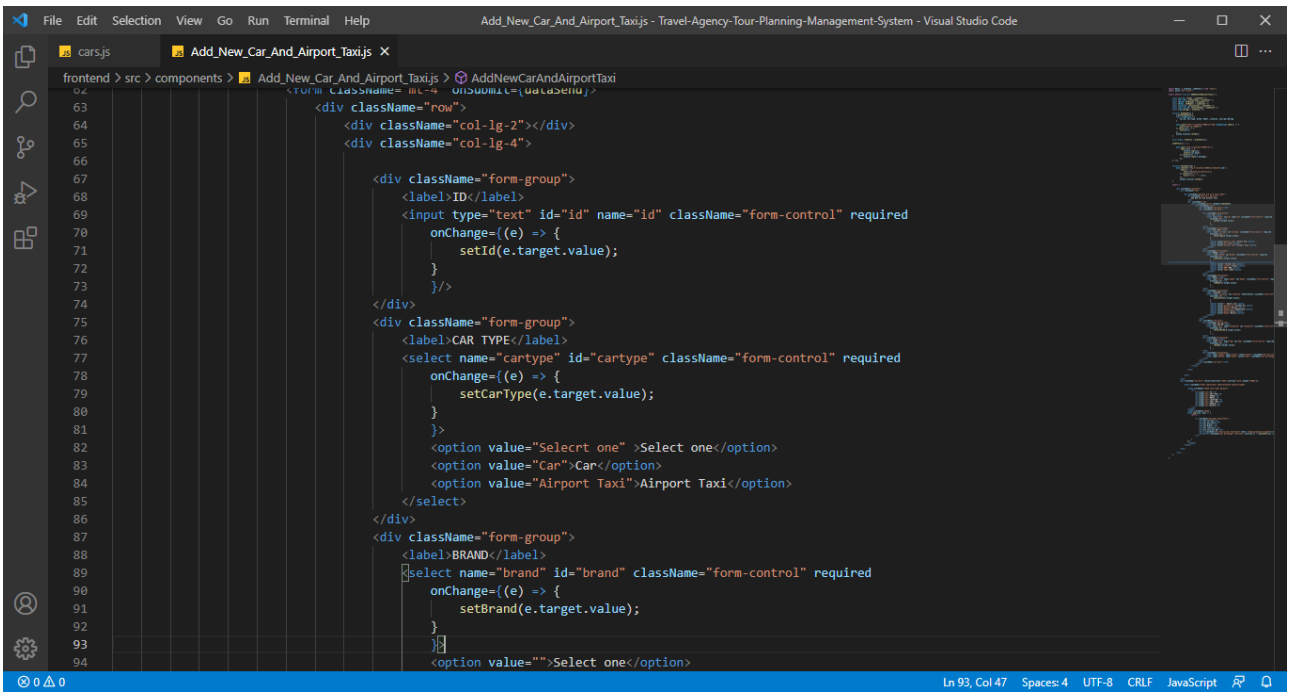

File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi > useEffect() callback > then() callback
import React, { useState, useEffect } from "react";
import axios from 'axios';
export default function AddNewCarAndAirportTaxi() {
const [Car_ID, setId] = useState("");
const [Car_Type, setCarType] = useState("");
const [Brand, setBrand] = useState("");
const [Model, setModel] = useState("");
const [Location, setLocation] = useState("");
const [Cost_per_KM, setCostPerKm] = useState("");
const [Img,setImg] = useState("");
function dataSend(e) {
e.preventDefault();
const newBooking = {
Car_ID, Car_Type, Brand, Model, Location, Cost_per_KM,Img
}
axios.post("http://localhost:8080/car/add",newBooking).then(() => {
alert("New Car Added");
}).catch((err) => {
alert(err);
})
window.location.reload();
}
const [cars, setCars] = useState([]);
useEffect(() => {
axios.get('http://localhost:8080/car/')
Ln 33, Col 29 Spaces: 4 UTF-8 CRLF JavaScript


File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi > useEffect() callback > then() callback
axios.get('http://localhost:8080/car/')
.then((res) => {
console.log(res);
setCars(res.data);
}).catch((err) => {
console.log(err.message);
});
}, []);
function deleteCar(id) {
axios.delete(`http://localhost:8080/car/delete/${id}`)
.then(() => {
alert("Deleted Successfully");
}).catch((err) => {
alert("error : " + err);
});
window.location.reload();
}
return (

<h1 className="p-2 mt-1 bg-info">
Add New Car and Airport Taxi
</h1>

<form className="mt-4" onSubmit={dataSend}>

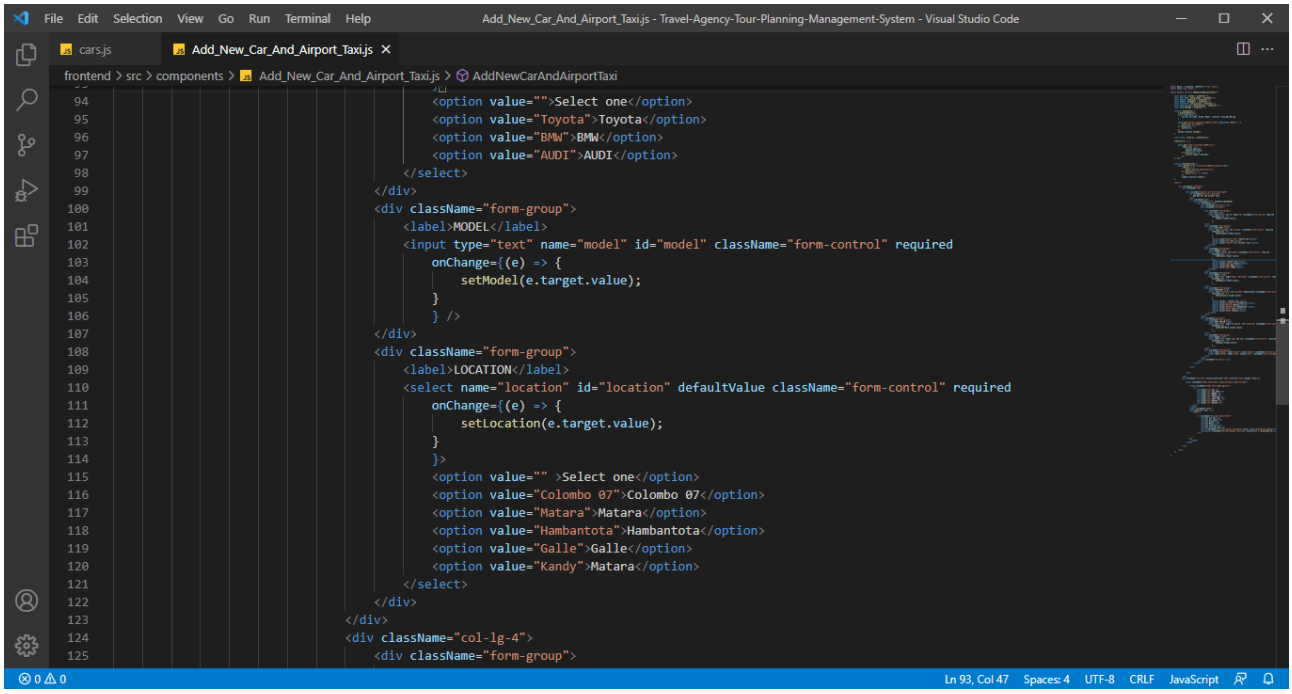
Ln 33, Col 29 Spaces: 4 UTF-8 CRLF JavaScript


File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi
<form className="mt-4" onSubmit={dataSend}>

<label>ID</label>
<input type="text" id="id" name="id" className="form-control" required
onChange={(e) => {
setId(e.target.value);
}
}/>

<label>CAR TYPE</label>
<select name="cartype" id="cartype" className="form-control" required
onChange={(e) => {
setCarType(e.target.value);
}
}>
<option value="Selecrt one" >Select one</option>
<option value="Car">Car</option>
<option value="Airport Taxi">Airport Taxi</option>
</select>

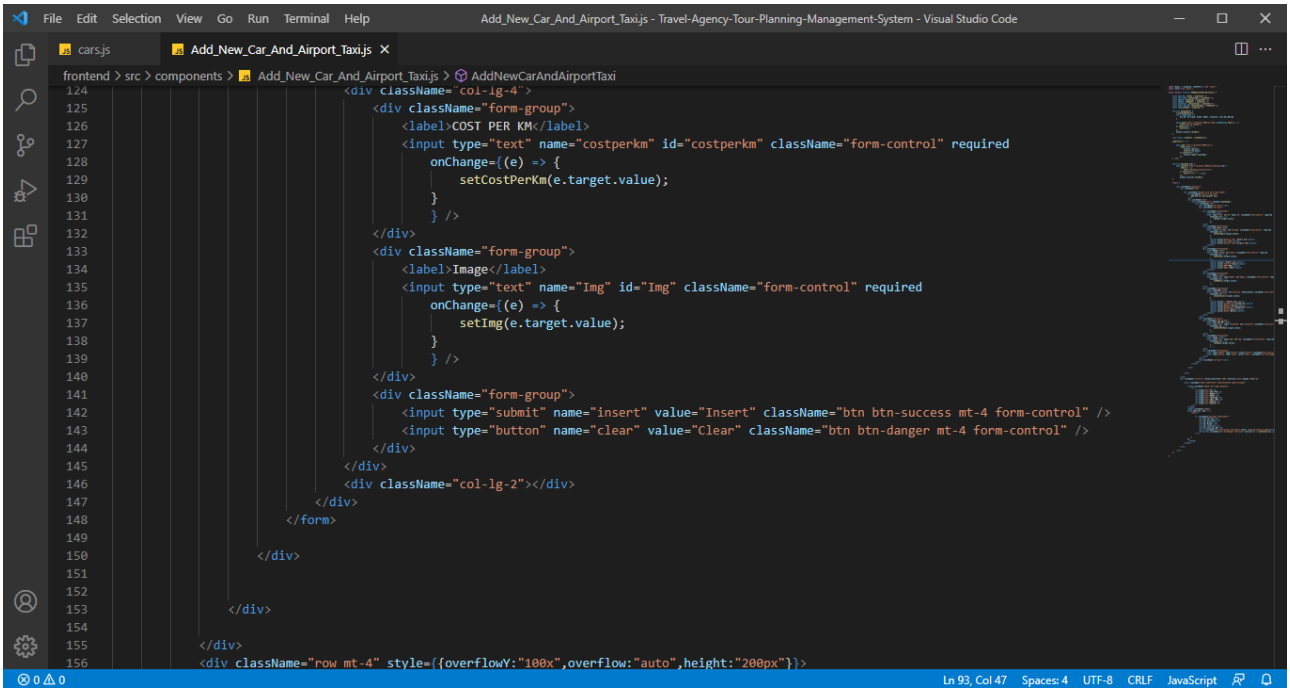
<label>BRAND</label>
<select name="brand" id="brand" className="form-control" required
onChange={(e) => {
setBrand(e.target.value);
}
}>
<option value="">Select one</option>
Ln 93, Col 47 Spaces: 4 UTF-8 CRLF JavaScript


File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi
<option value="">Select one</option>
<option value="Toyota">Toyota</option>
<option value="BMW">BMW</option>
<option value="AUDI">AUDI</option>
</select>

<label>MODEL</label>
<input type="text" name="model" id="model" className="form-control" required
onChange={(e) => {
setModel(e.target.value);
}
} />

<label>LOCATION</label>
<select name="location" id="location" defaultValue className="form-control" required
onChange={(e) => {
setLocation(e.target.value);
}
}>
<option value="" >Select one</option>
<option value="Colombo 07">Colombo 07</option>
<option value="Matara">Matara</option>
<option value="Hambantota">Hambantota</option>
<option value="Galle">Galle</option>
<option value="Kandy">Matara</option>
</select>

Ln 93, Col 47 Spaces: 4 UTF-8 CRLF JavaScript


File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi

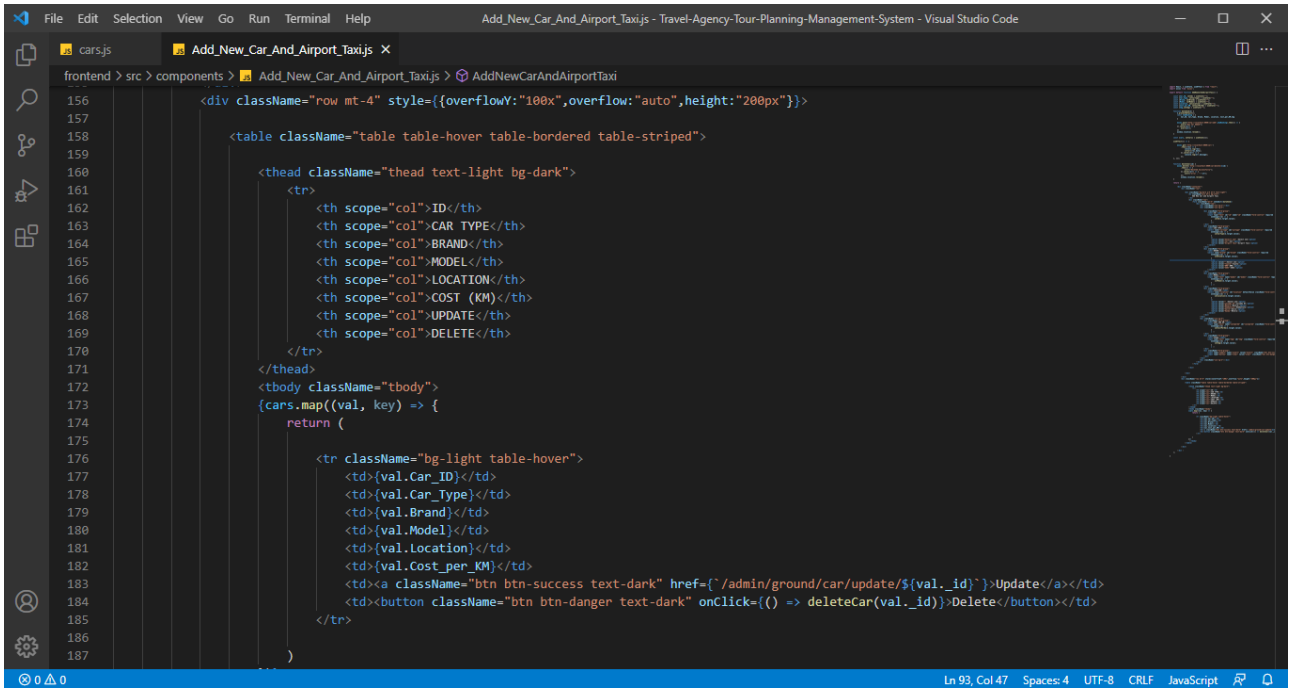
<label>COST PER KM</label>
<input type="text" name="costperkm" id="costperkm" className="form-control" required
onChange={(e) => {
setCostPerKm(e.target.value);
}
} />

<label>Image</label>
<input type="text" name="Img" id="Img" className="form-control" required
onChange={(e) => {
setImg(e.target.value);
}
} />

<input type="submit" name="insert" value="Insert" className="btn btn-success mt-4 form-control" />
<input type="button" name="clear" value="Clear" className="btn btn-danger mt-4 form-control" />

</form>

<div className="row mt-4" style={{overflowY:"100x",overflow:"auto",height:"200px"}}>
Ln 93, Col 47 Spaces: 4 UTF-8 CRLF JavaScript


File Edit Selection View Go Run Terminal Help
Add_New_Car_And_Airport_Taxi.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
cars.js
Add_New_Car_And_Airport_Taxi.js
frontend > src > components > Add_New_Car_And_Airport_Taxi.js > AddNewCarAndAirportTaxi
<div className="row mt-4" style={{overflowY:"100x",overflow:"auto",height:"200px"}}>
<table className="table table-hover table-bordered table-striped">
<thead className="thead text-light bg-dark">
<tr>
<th scope="col">ID</th>
<th scope="col">CAR TYPE</th>
<th scope="col">BRAND</th>
<th scope="col">MODEL</th>
<th scope="col">LOCATION</th>
<th scope="col">COST (KM)</th>
<th scope="col">UPDATE</th>
<th scope="col">DELETE</th>
</tr>
</thead>
<tbody className="tbody">
{cars.map((val, key) => {
return (
<tr className="bg-light table-hover">
<td>{val.Car_ID}</td>
<td>{val.Car_Type}</td>
<td>{val.Brand}</td>
<td>{val.Model}</td>
<td>{val.Location}</td>
<td>{val.Cost_per_KM}</td>
<td><a className="btn btn-success text-dark" href={`/admin/ground/car/update/${val._id}`}>Update</td>
<td><button className="btn btn-danger text-dark" onClick={() => deleteCar(val._id)}>Delete</button></td>
</tr>
)
Ln 93, Col 47 Spaces: 4 UTF-8 CRLF JavaScript

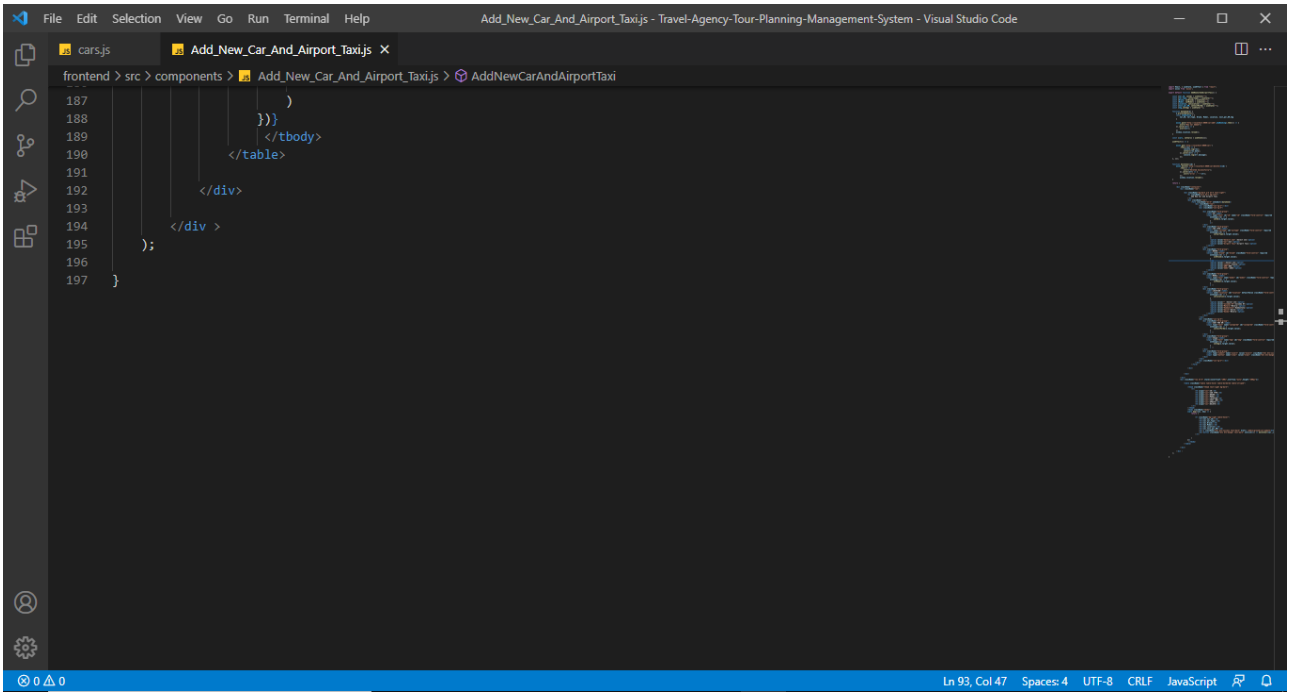


Figure 101

2.3.3.5 Travel Document Management

Models

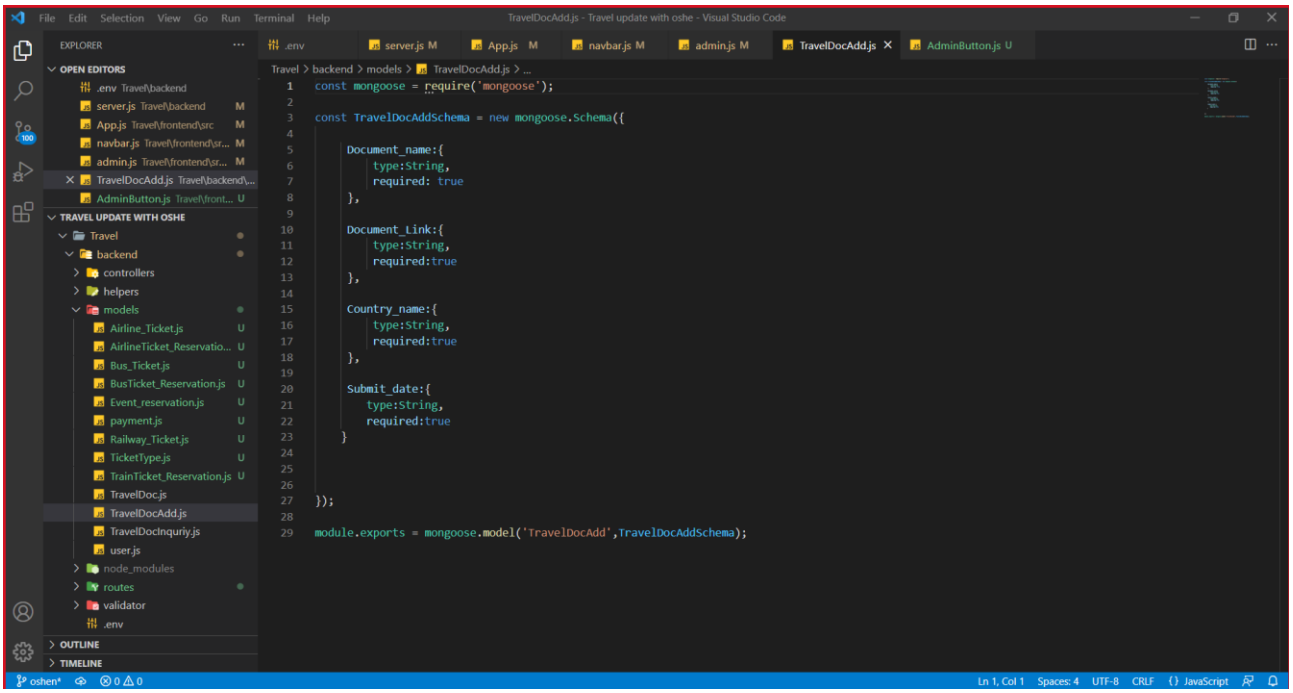
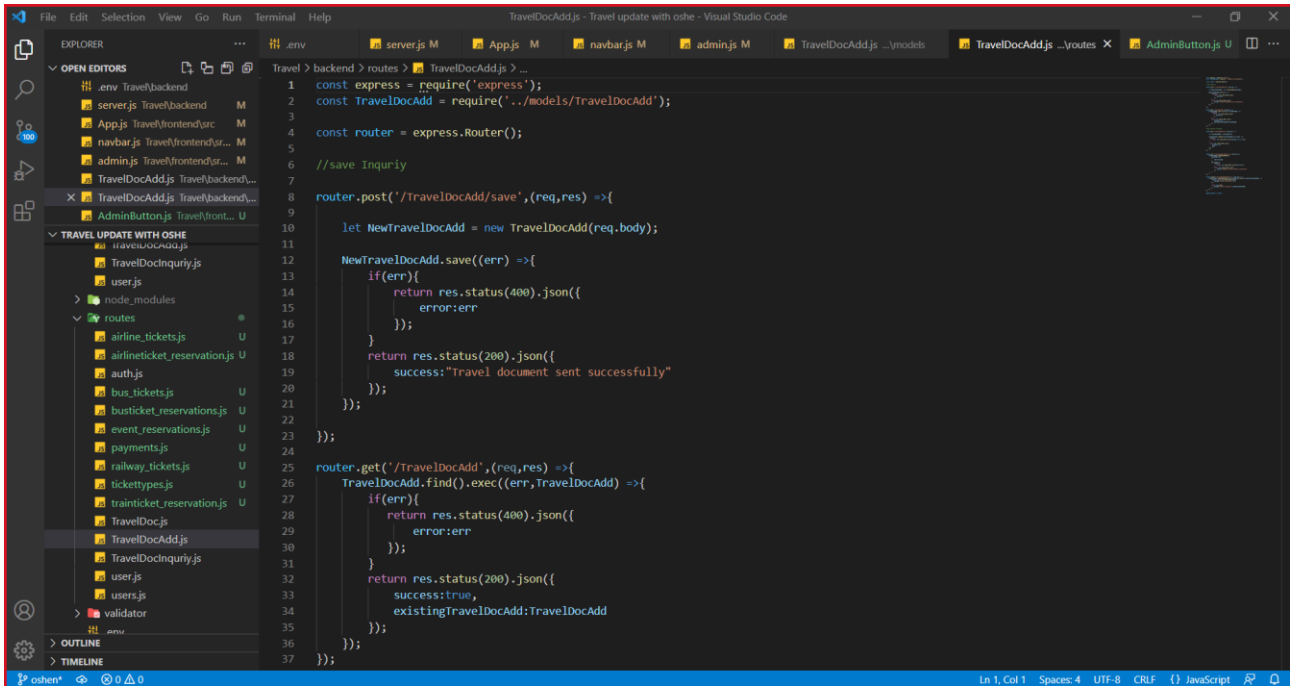
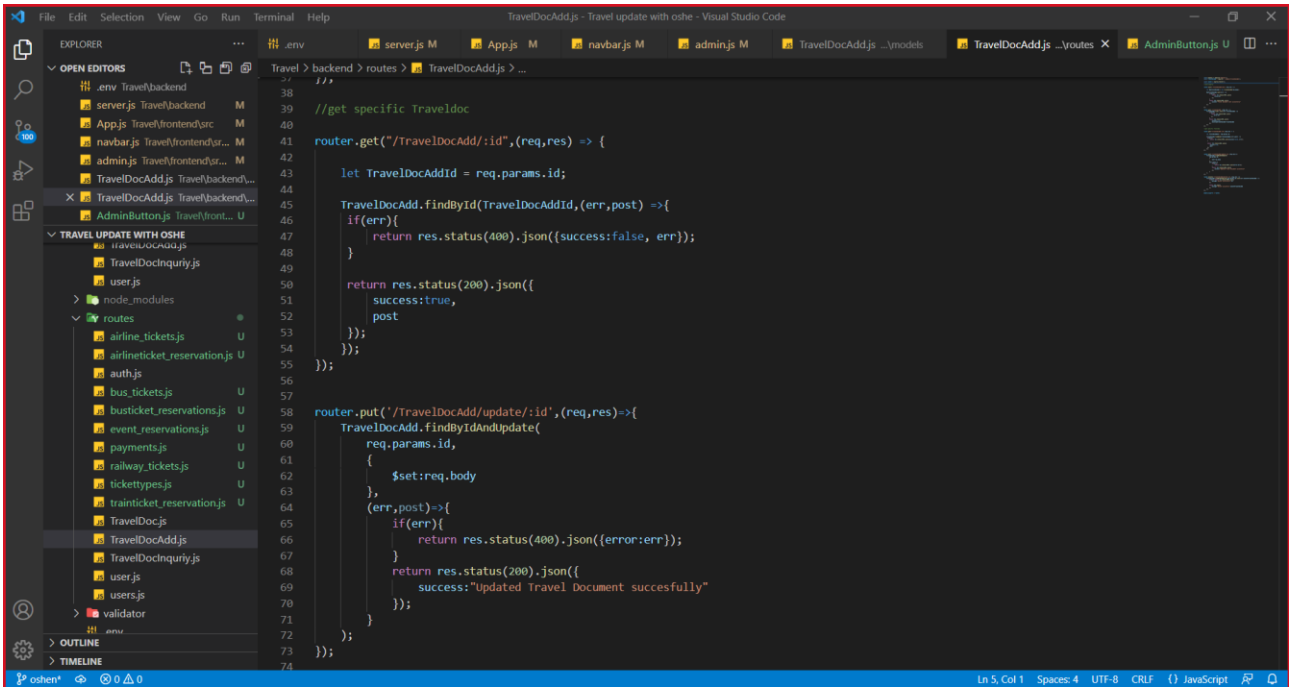


Figure 102

Routes


TravelDocAdd.js - Travel update with oshe - Visual Studio Code
Travel > backend > routes > TravelDocAdd.js > ...
const express = require('express');
const TravelDocAdd = require('../models/TravelDocAdd');
const router = express.Router();
//save Inquriy
router.post('/TravelDocAdd/save',(req,res) =>{
let NewTravelDocAdd = new TravelDocAdd(req.body);
NewTravelDocAdd.save((err) =>{
if(err){
return res.status(400).json({
error:err
});
}
return res.status(200).json({
success:"Travel document sent successfully"
});
});
});
router.get('/TravelDocAdd',(req,res) =>{
TravelDocAdd.find().exec((err,TravelDocAdd) =>{
if(err){
return res.status(400).json({
error:err
});
}
return res.status(200).json({
success:true,
existingTravelDocAdd:TravelDocAdd
});
});
});


TravelDocAdd.js - Travel update with oshe - Visual Studio Code
Travel > backend > routes > TravelDocAdd.js > ...
//get specific Traveldoc
router.get("/TravelDocAdd/:id",(req,res) => {
let TravelDocAddId = req.params.id;
TravelDocAdd.findById(TravelDocAddId,(err,post) =>{
if(err){
return res.status(400).json({success:false, err});
}
return res.status(200).json({
success:true,
post
});
});
});
router.put('/TravelDocAdd/update/:id',(req,res)=>{
TravelDocAdd.findByIdAndUpdate(
req.params.id,
{
$set:req.body
},
(err,post)=>{
if(err){
return res.status(400).json({error:err});
}
return res.status(200).json({
success:"Updated Travel Document succesfully"
});
}
);
});

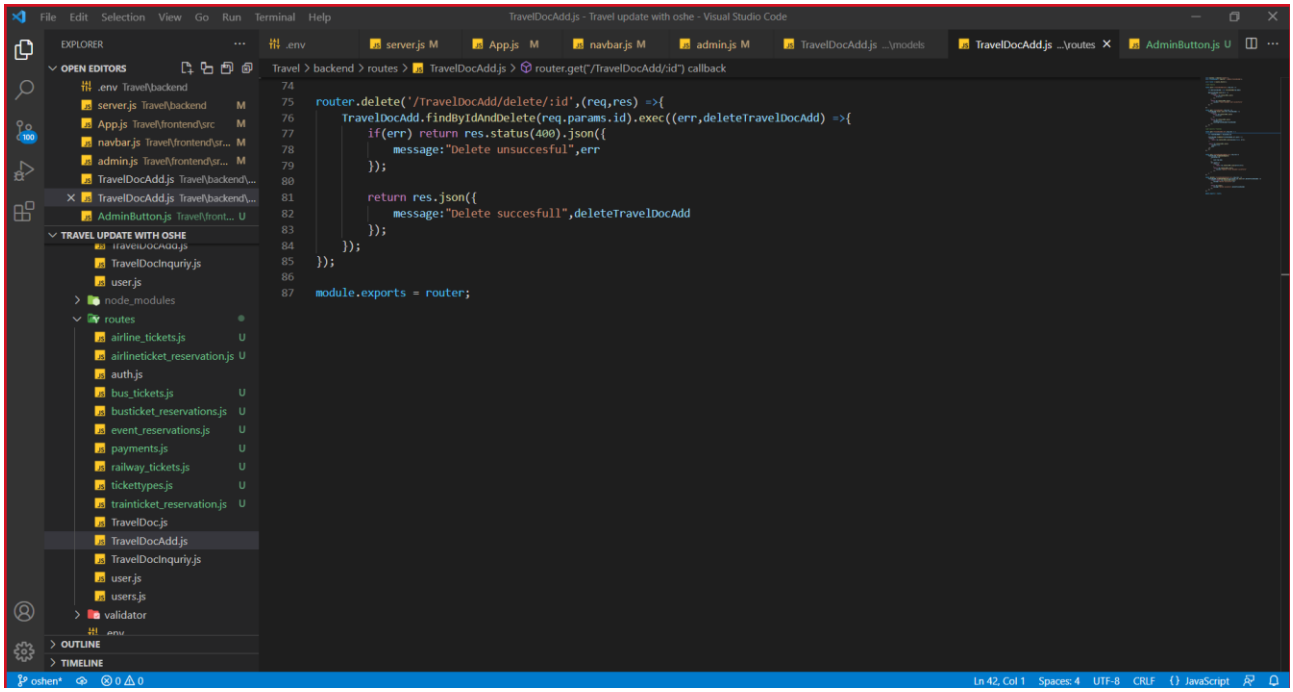
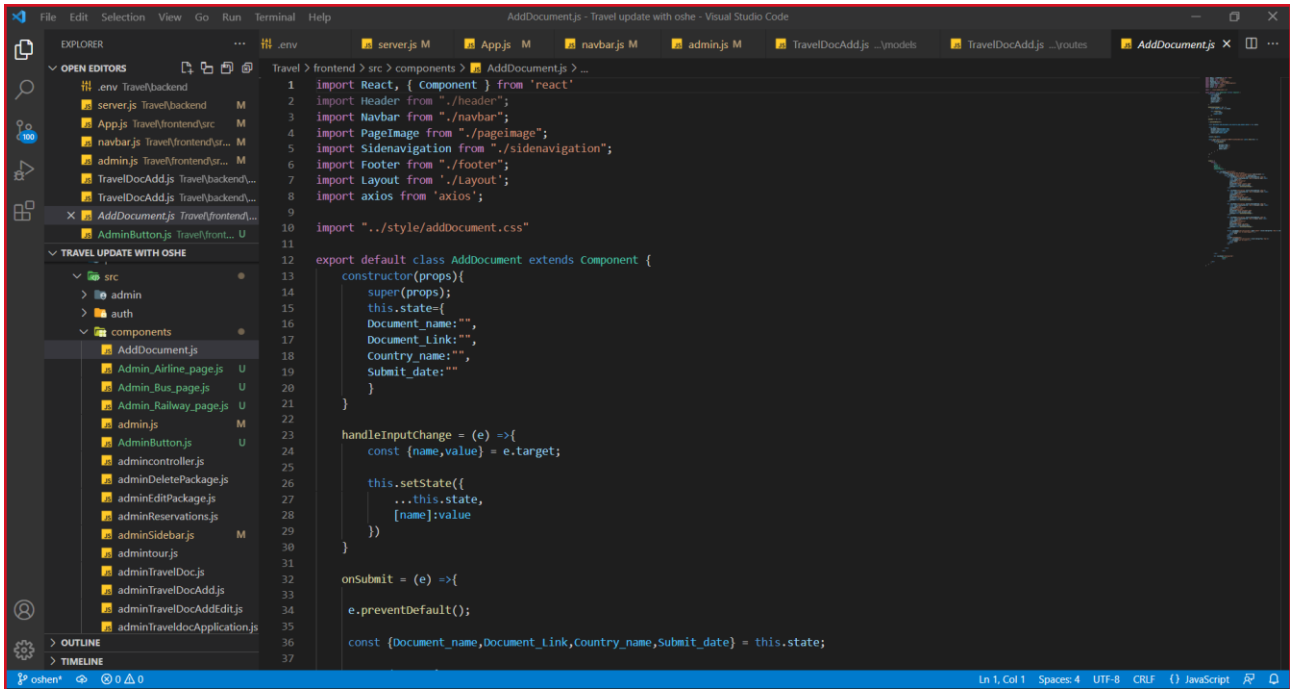
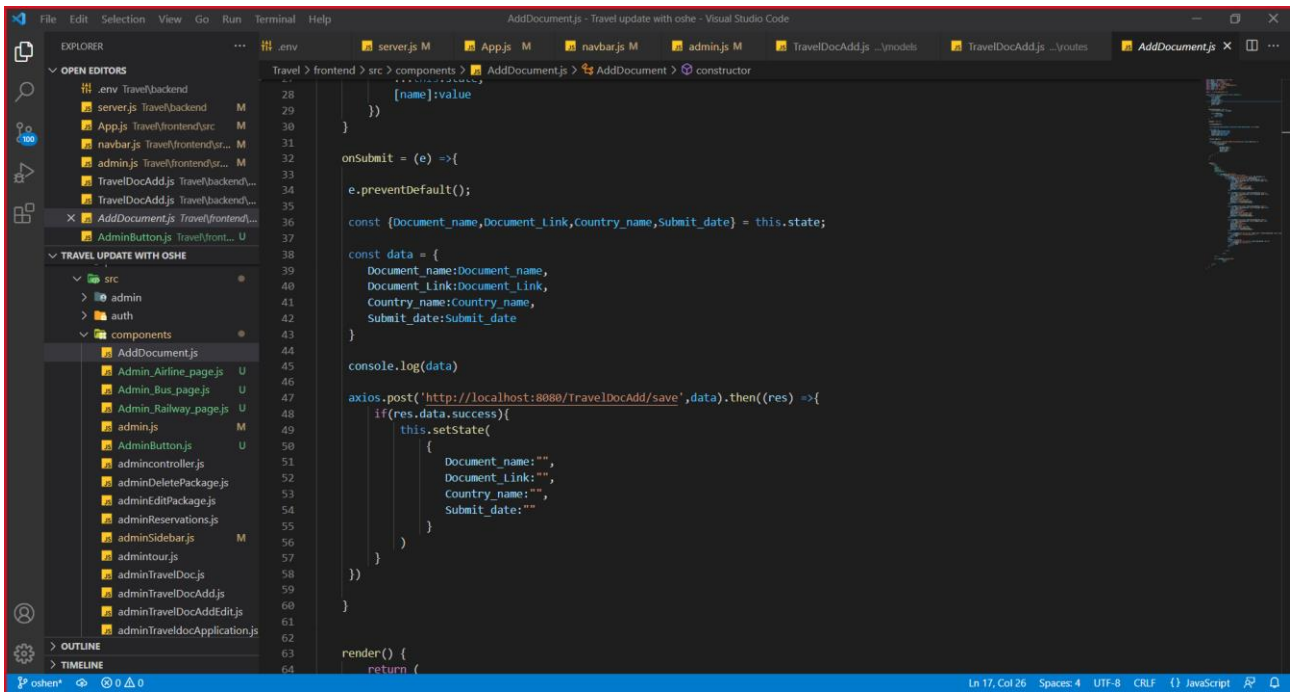


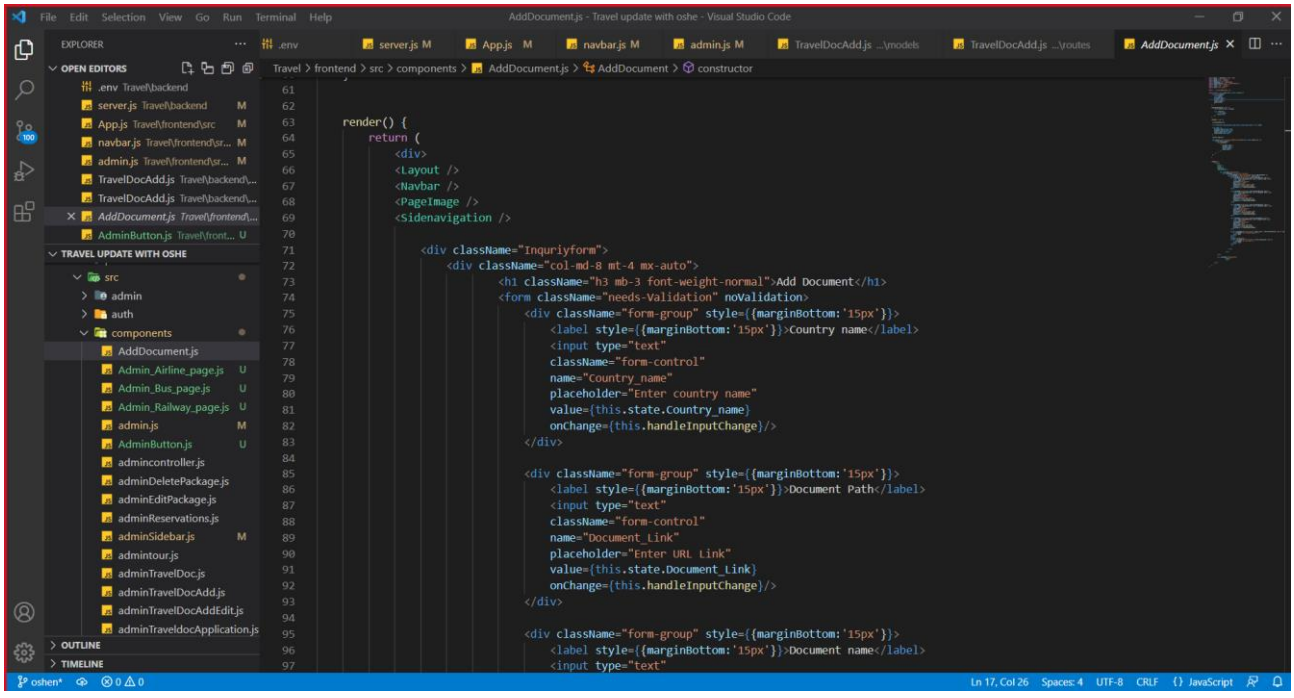
Figure 103

Frontend

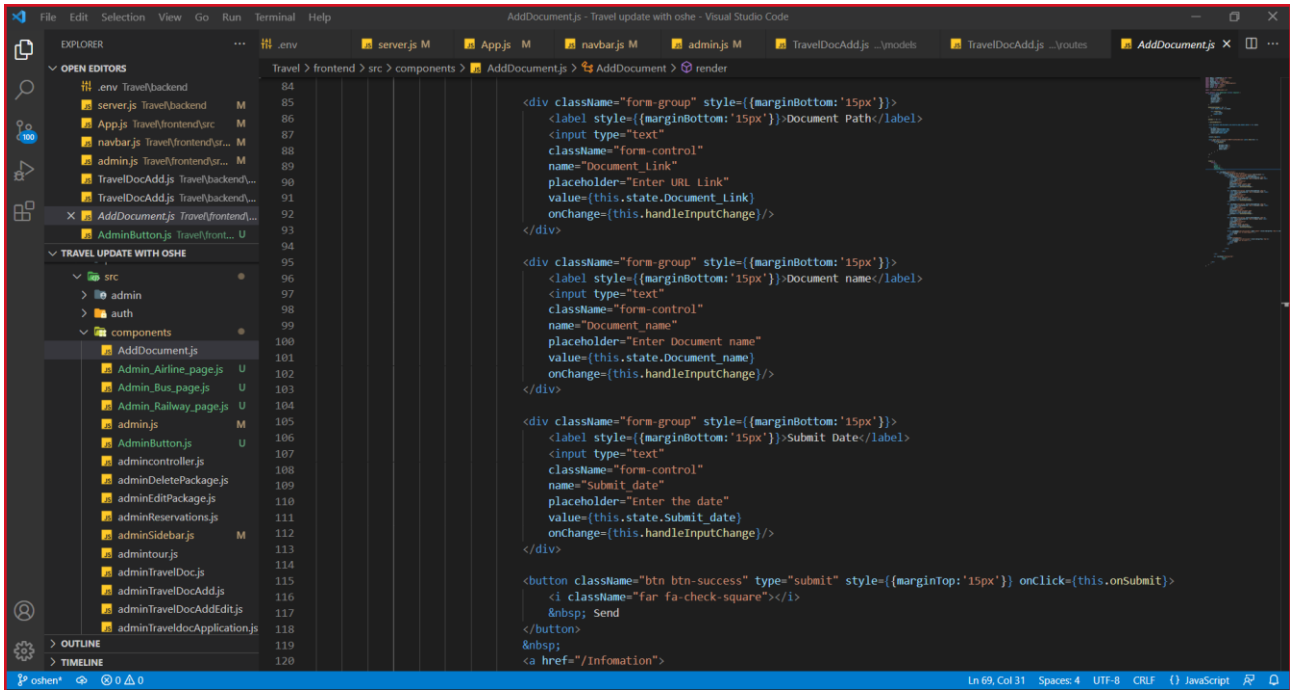




```
            [name]:value
        })
    }

    onSubmit = (e) =>{

    e.preventDefault();

    const {Document_name,Document_Link,Country_name,Submit_date} = this.state;

    const data = {
       Document_name:Document_name,
       Document_Link:Document_Link,
       Country_name:Country_name,
       Submit_date:Submit_date
    }

    console.log(data)

    axios.post('http://localhost:8080/TravelDocAdd/save',data).then((res) =>{
        if(res.data.success){
            this.setState(
                {
                    Document_name:"",
                    Document_Link:"",
                    Country_name:"",
                    Submit_date:""
                }
            )
        }
    })

    }


    render() {
        return (
```



```
    render() {
        return (
            <div>
            <Layout />
            <Navbar />
            <PageImage />
            <Sidenavigation />

                <div className="Inquriyform">
                    <div className="col-md-8 mt-4 mx-auto">
                        <h1 className="h3 mb-3 font-weight-normal">Add Document</h1>
                        <form className="needs-Validation" noValidation>
                            <div className="form-group" style={{marginBottom:'15px'}}>
                                <label style={{marginBottom:'15px'}}>Country name</label>
                                <input type="text"
                                className="form-control"
                                name="Country_name"
                                placeholder="Enter country name"
                                value={this.state.Country_name}
                                onChange={this.handleInputChange}/>
                            </div>

                            <div className="form-group" style={{marginBottom:'15px'}}>
                                <label style={{marginBottom:'15px'}}>Document Path</label>
                                <input type="text"
                                className="form-control"
                                name="Document_Link"
                                placeholder="Enter URL Link"
                                value={this.state.Document_Link}
                                onChange={this.handleInputChange}/>
                            </div>

                            <div className="form-group" style={{marginBottom:'15px'}}>
                                <label style={{marginBottom:'15px'}}>Document name</label>
                                <input type="text"
```


File Edit Selection View Go Run Terminal Help
AddDocument.js - Travel update with oshe - Visual Studio Code
EXPLORER
OPEN EDITORS
TRAVEL UPDATE WITH OSHE
src
admin
auth
components
AddDocument.js
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Railway_page.js
admin.js
AdminButton.js
admincontroller.js
adminDeletePackage.js
adminEditPackage.js
adminReservations.js
adminSidebar.js
admintour.js
adminTravelDoc.js
adminTravelDocAdd.js
adminTravelDocAddEdit.js
adminTraveldocApplication.js
OUTLINE
TIMELINE
Travel > frontend > src > components > AddDocument.js > AddDocument > render
<div className="form-group" style={{marginBottom:'15px'}}>
<label style={{marginBottom:'15px'}}>Document Path</label>
<input type="text"
className="form-control"
name="Document_Link"
placeholder="Enter URL Link"
value={this.state.Document_Link}
onChange={this.handleInputChange}/>

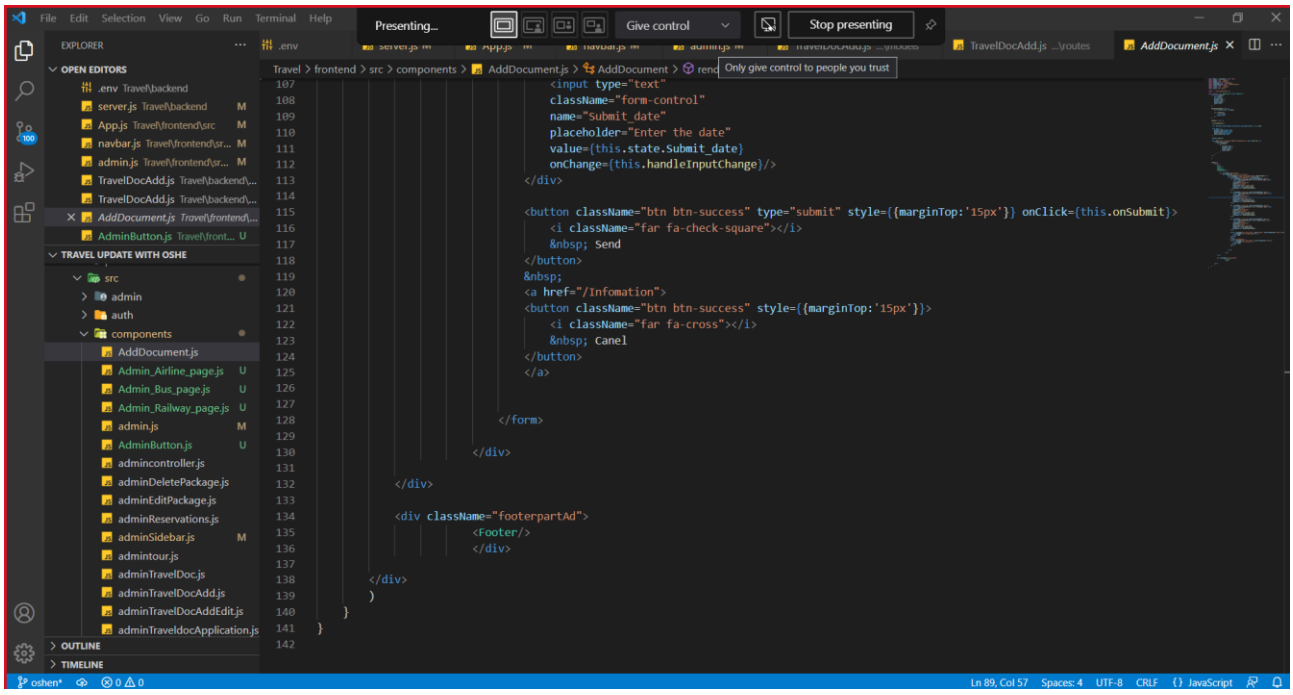
<div className="form-group" style={{marginBottom:'15px'}}>
<label style={{marginBottom:'15px'}}>Document name</label>
<input type="text"
className="form-control"
name="Document_name"
placeholder="Enter Document name"
value={this.state.Document_name}
onChange={this.handleInputChange}/>

<div className="form-group" style={{marginBottom:'15px'}}>
<label style={{marginBottom:'15px'}}>Submit Date</label>
<input type="text"
className="form-control"
name="Submit_date"
placeholder="Enter the date"
value={this.state.Submit_date}
onChange={this.handleInputChange}/>

<button className="btn btn-success" type="submit" style={{marginTop:'15px'}} onClick={this.onSubmit}>

  Send
</button>

oshen*
Ln 69, Col 31
Spaces: 4
UTF-8
CRLF
JavaScript


Presenting...
Give control
Stop presenting
Only give control to people you trust
<input type="text"
className="form-control"
name="Submit_date"
placeholder="Enter the date"
value={this.state.Submit_date}
onChange={this.handleInputChange}/>

<button className="btn btn-success" type="submit" style={{marginTop:'15px'}} onClick={this.onSubmit}>

  Send
</button>

<button className="btn btn-success" style={{marginTop:'15px'}}>

  Canel
</button>

</form>

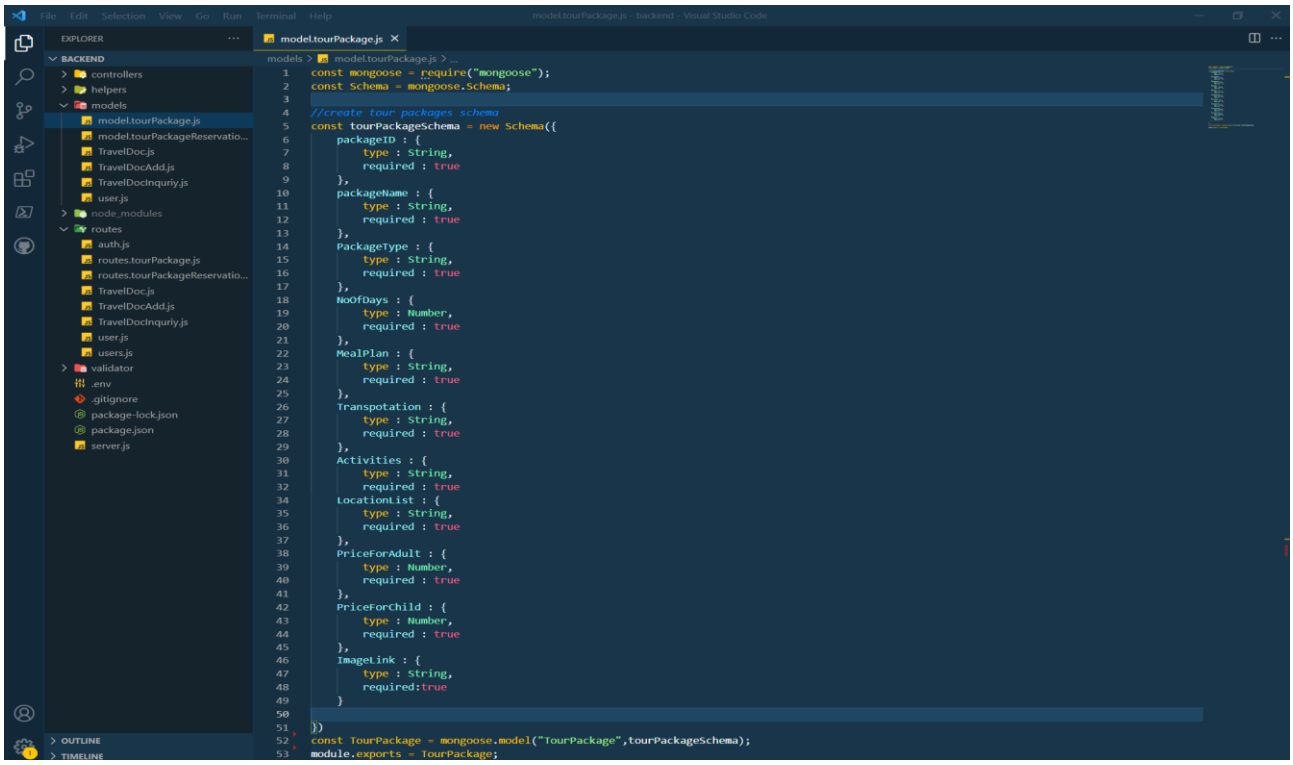
<Footer/>

)
}
}
oshen*
Ln 89, Col 57
Spaces: 4
UTF-8
CRLF
JavaScript

Figure 104

2.3.3.6 Tour Package Management

Models



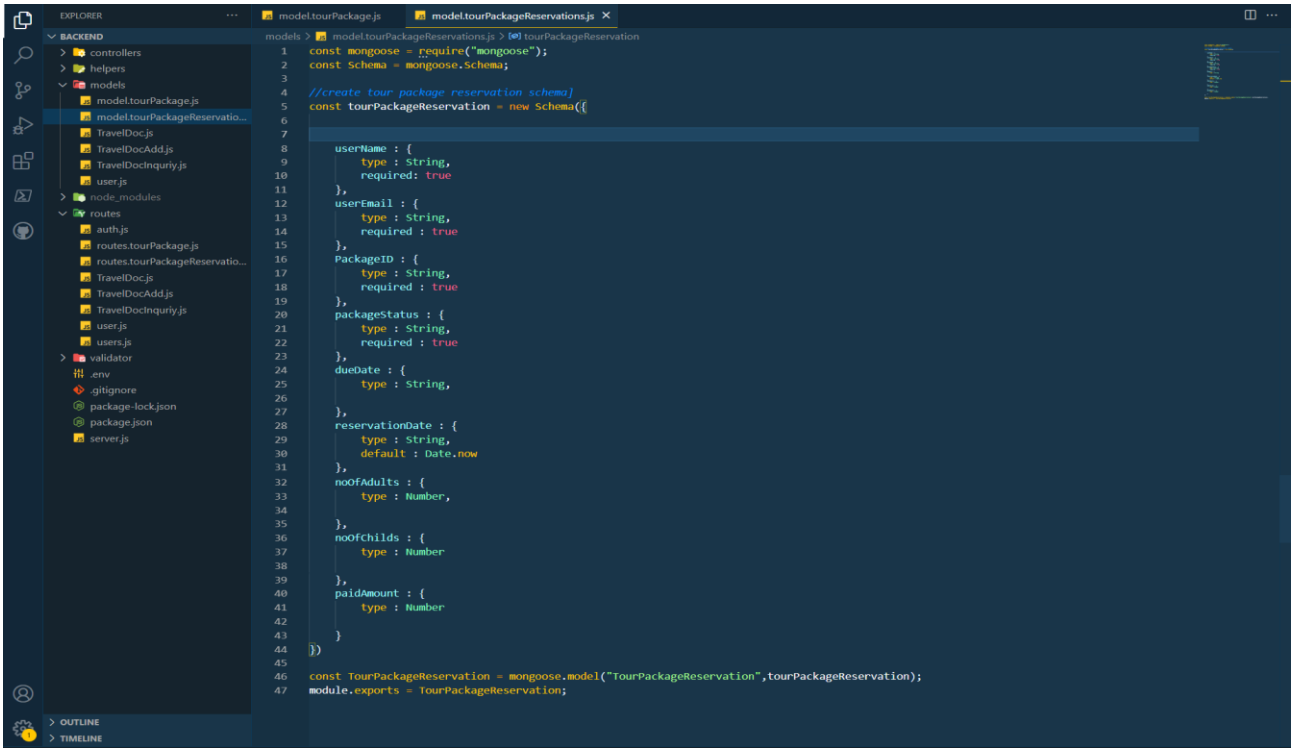
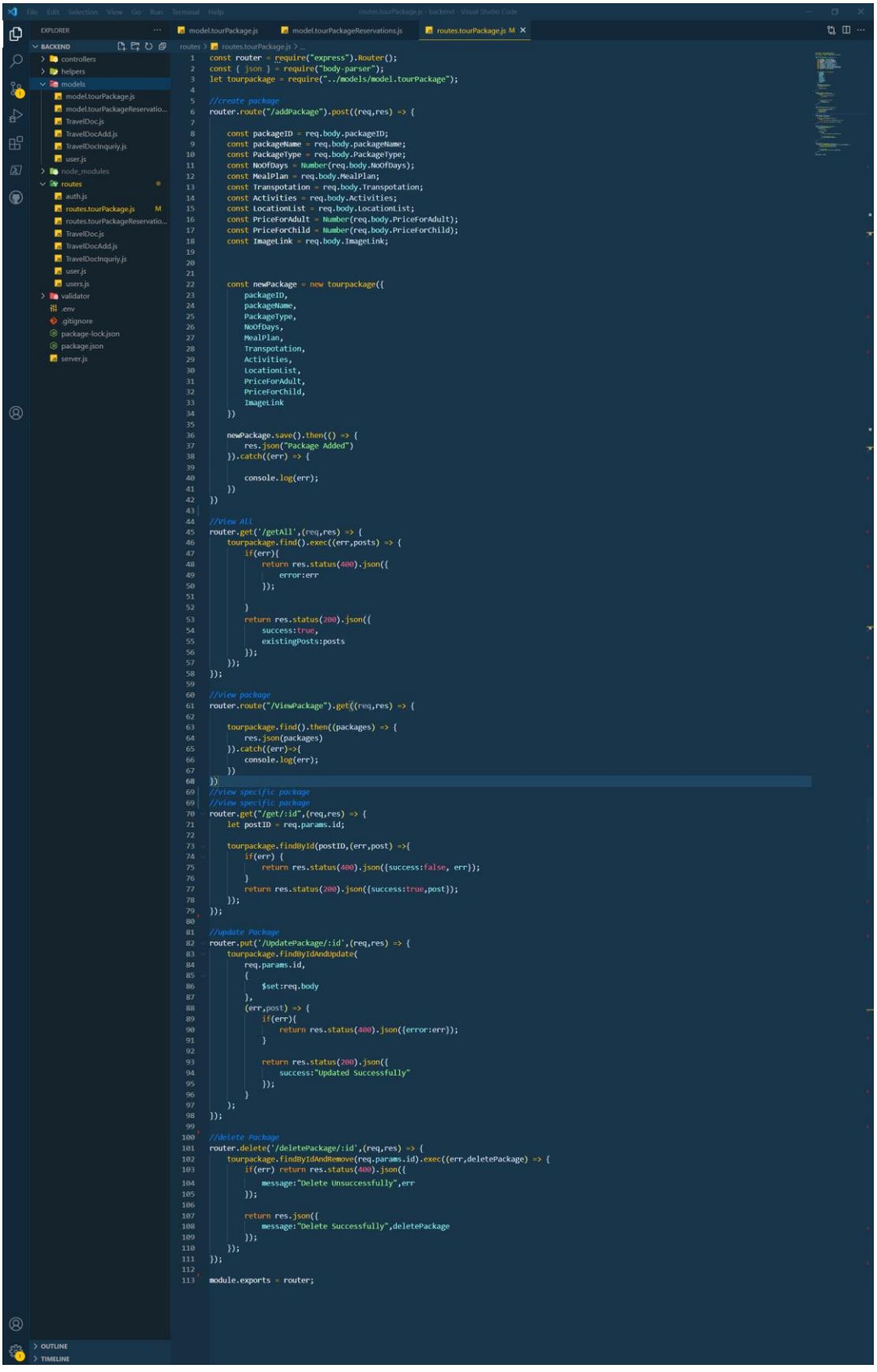


Figure 105

Routes


routes.tourPackage.js - backend - Visual Studio Code
EXPLORER
BACKEND
controllers
helpers
models
model.tourPackage.js
model.tourPackageReservatio...
TravelDoc.js
TravelDocAdd.js
TravelDocInquriy.js
user.js
node_modules
routes
auth.js
routes.tourPackage.js M
routes.tourPackageReservatio...
TravelDoc.js
TravelDocAdd.js
TravelDocInquriy.js
user.js
users.js
validator
.env
.gitignore
package-lock.json
package.json
server.js
OUTLINE
TIMELINE
model.tourPackage.js
model.tourPackageReservations.js
routes.tourPackage.js M
routes > routes.tourPackage.js > ...
1 const router = require("express").Router();
2 const { json } = require("body-parser");
3 let tourpackage = require("../models/model.tourPackage");
4
5 //create package
6 router.route("/addPackage").post((req,res) => {
7
8 const packageID = req.body.packageID;
9 const packageName = req.body.packageName;
10 const PackageType = req.body.PackageType;
11 const NoOfDays = Number(req.body.NoOfDays);
12 const MealPlan = req.body.MealPlan;
13 const Transpotation = req.body.Transpotation;
14 const Activities = req.body.Activities;
15 const LocationList = req.body.LocationList;
16 const PriceForAdult = Number(req.body.PriceForAdult);
17 const PriceForChild = Number(req.body.PriceForChild);
18 const ImageLink = req.body.ImageLink;
19
20
21
22 const newPackage = new tourpackage({
23 packageID,
24 packageName,
25 PackageType,
26 NoOfDays,
27 MealPlan,
28 Transpotation,
29 Activities,
30 LocationList,
31 PriceForAdult,
32 PriceForChild,
33 ImageLink
34 })
35
36 newPackage.save().then(() => {
37 res.json("Package Added")
38 }).catch((err) => {
39
40 console.log(err);
41 })
42 })
43
44 //View All
45 router.get('/getAll',(req,res) => {
46 tourpackage.find().exec((err,posts) => {
47 if(err){
48 return res.status(400).json({
49 error:err
50 });
51
52 }
53 return res.status(200).json({
54 success:true,
55 existingPosts:posts
56 });
57 });
58 });
59
60 //view package
61 router.route("/ViewPackage").get((req,res) => {
62
63 tourpackage.find().then((packages) => {
64 res.json(packages)
65 }).catch((err)=>{
66 console.log(err);
67 })
68 })
69 //view specific package
70 //view specific package
71 router.get("/get/:id",(req,res) => {
72 let postID = req.params.id;
73
74 tourpackage.findById(postID,(err,post) =>{
75 if(err) {
76 return res.status(400).json({success:false, err});
77 }
78 return res.status(200).json({success:true,post});
79 });
80 });
81
82 //update Package
83 router.put('/UpdatePackage/:id',(req,res) => {
84 tourpackage.findByIdAndUpdate(
85 req.params.id,
86 {
87 $set:req.body
88 },
89 (err,post) => {
90 if(err){
91 return res.status(400).json({error:err});
92 }
93
94 return res.status(200).json({
95 success:"Updated Successfully"
96 });
97 }
98);
99 });
100
101 //delete Package
102 router.delete('/deletePackage/:id',(req,res) => {
103 tourpackage.findByIdAndRemove(req.params.id).exec((err,deletePackage) => {
104 if(err) return res.status(400).json({
105 message:"Delete Unsuccessfully",err
106 });
107
108 return res.json({
109 message:"Delete Successfully",deletePackage
110 });
111 });
112 });
113
114 module.exports = router;

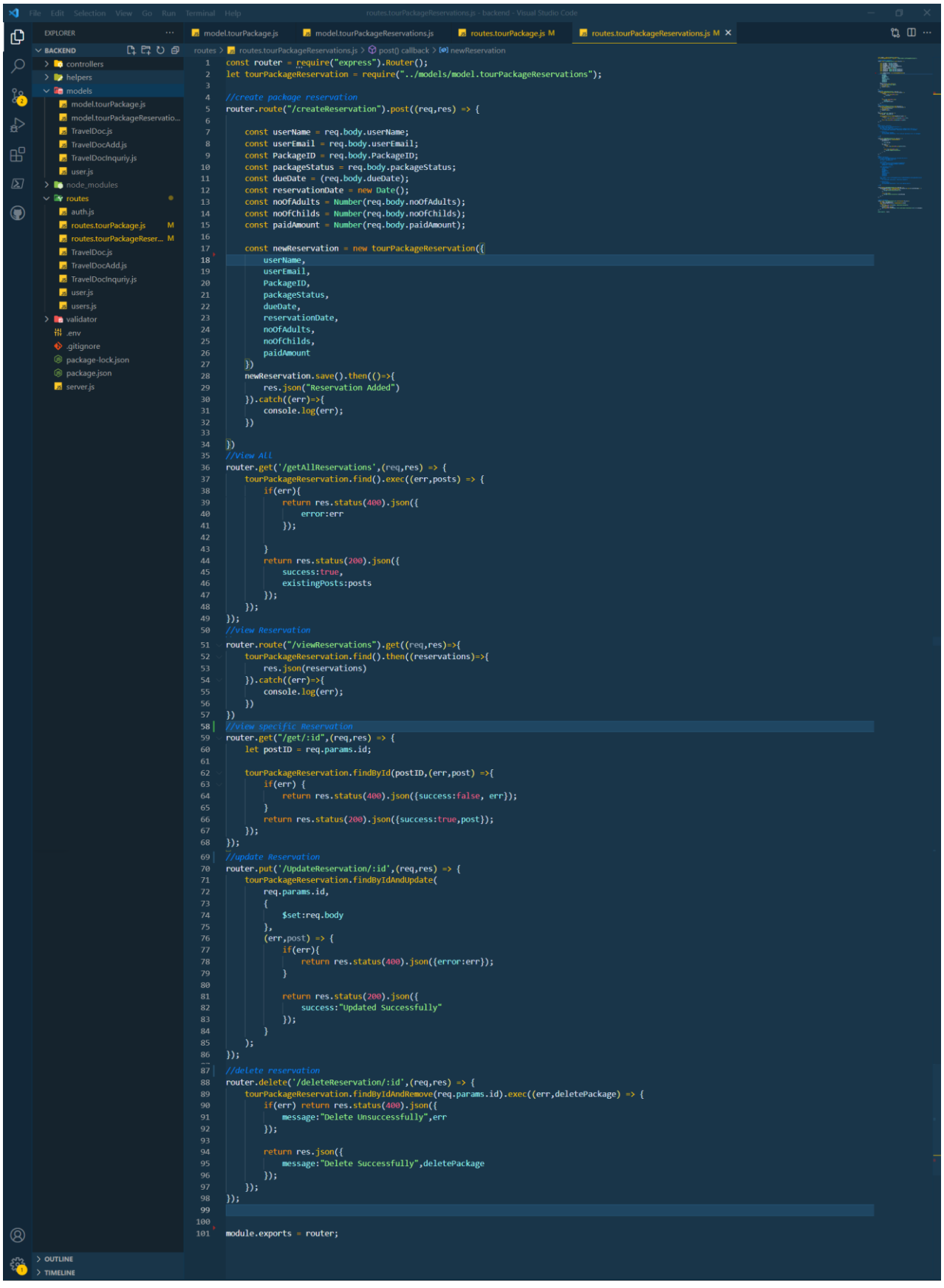
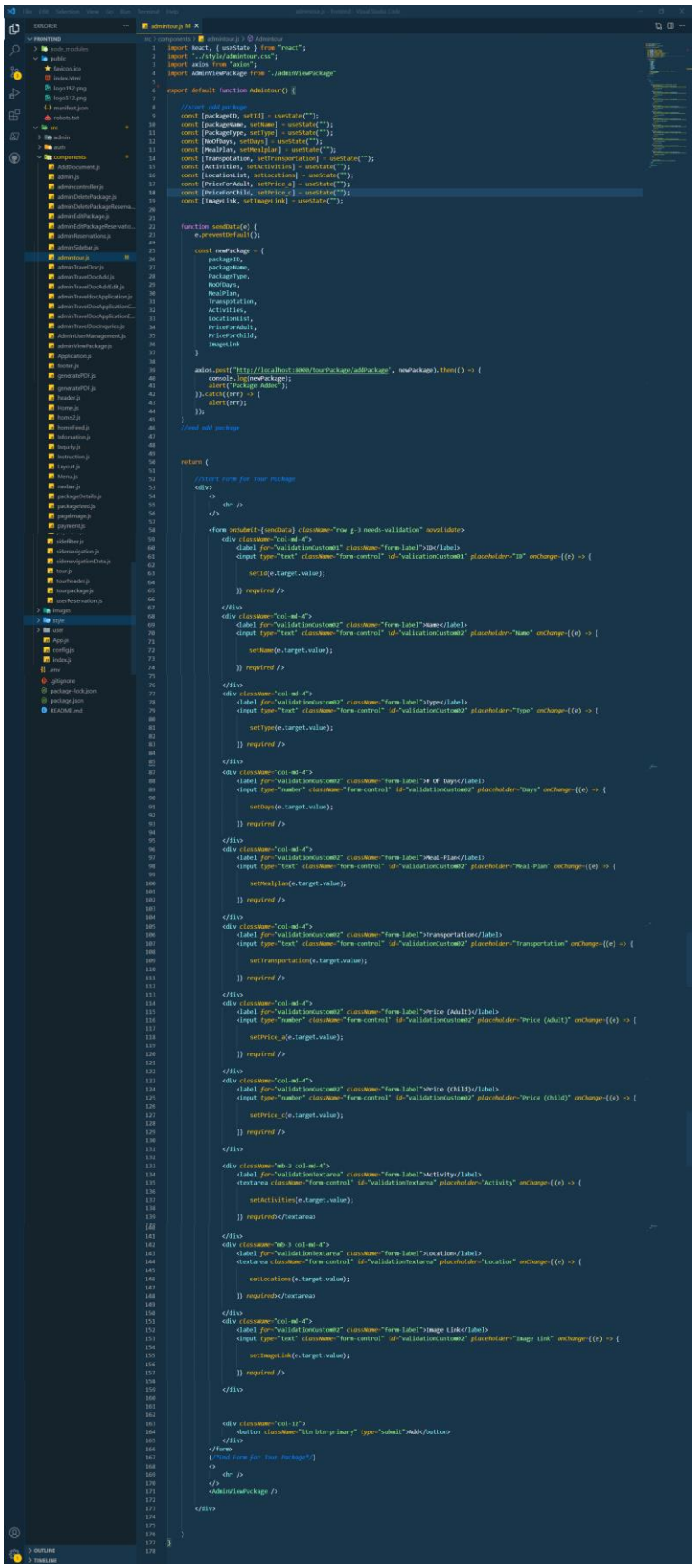
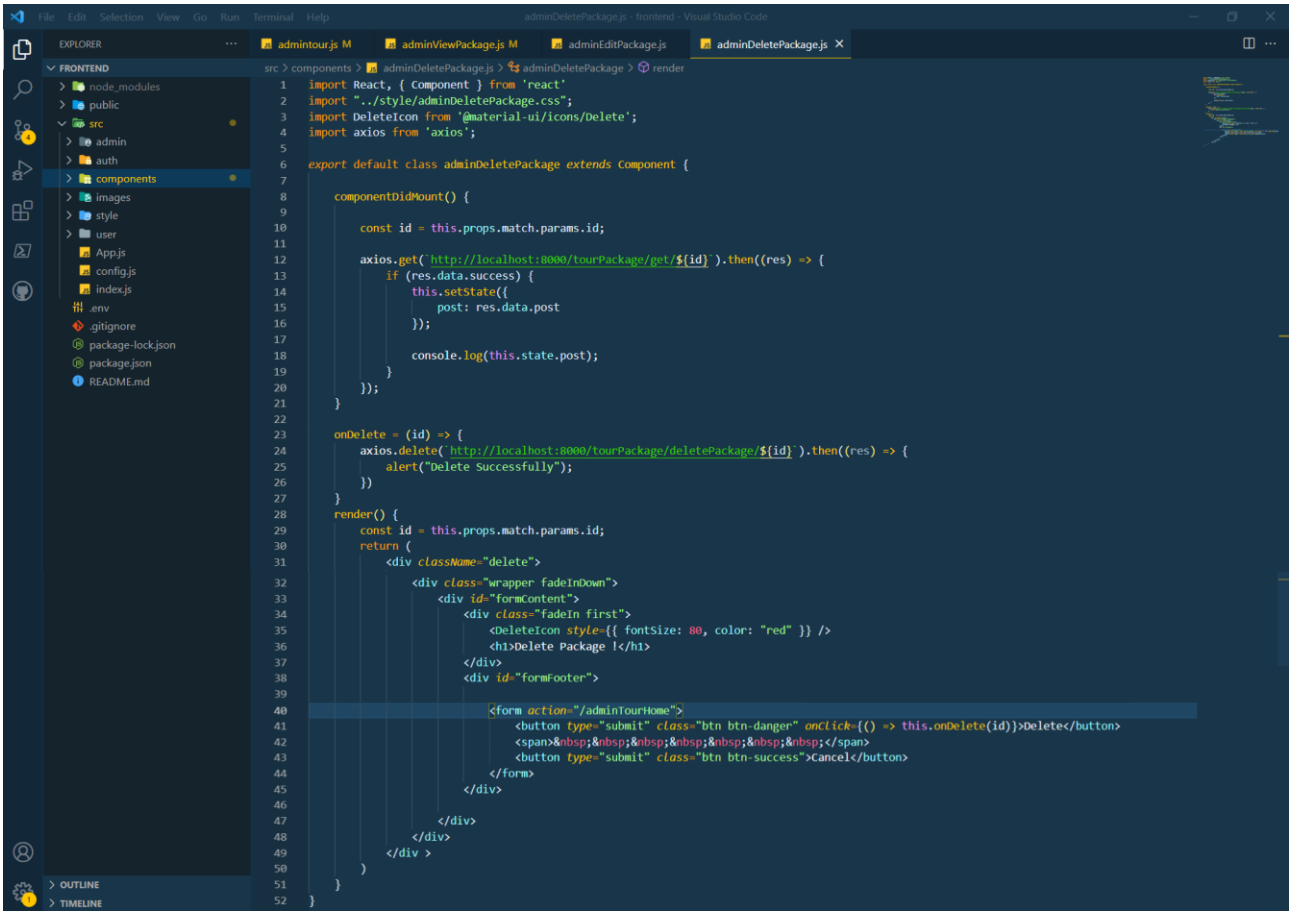

routes.tourPackageReservations.js - backend - Visual Studio Code
EXPLORER
BACKEND
controllers
helpers
models
model.tourPackage.js
model.tourPackageReservatio...
TravelDoc.js
TravelDocAdd.js
TravelDocInquriy.js
user.js
node_modules
routes
auth.js
routes.tourPackage.js M
routes.tourPackageReser... M
TravelDoc.js
TravelDocAdd.js
TravelDocInquriy.js
user.js
users.js
validator
.env
.gitignore
package-lock.json
package.json
server.js
OUTLINE
TIMELINE
model.tourPackage.js
model.tourPackageReservations.js
routes.tourPackage.js M
routes.tourPackageReservations.js M
routes > routes.tourPackageReservations.js > post() callback > newReservation
const router = require("express").Router();
let tourPackageReservation = require("../models/model.tourPackageReservations");
//create package reservation
router.route("/createReservation").post((req,res) => {
const userName = req.body.userName;
const userEmail = req.body.userEmail;
const PackageID = req.body.PackageID;
const packageStatus = req.body.packageStatus;
const dueDate = (req.body.dueDate);
const reservationDate = new Date();
const noOfAdults = Number(req.body.noOfAdults);
const noOfchilds = Number(req.body.noOfchilds);
const paidAmount = Number(req.body.paidAmount);
const newReservation = new tourPackageReservation({
userName,
userEmail,
PackageID,
packageStatus,
dueDate,
reservationDate,
noOfAdults,
noOfchilds,
paidAmount
})
newReservation.save().then(()=>{
res.json("Reservation Added")
}).catch((err)=>{
console.log(err);
})
})
//View All
router.get('/getAllReservations',(req,res) => {
tourPackageReservation.find().exec((err,posts) => {
if(err){
return res.status(400).json({
error:err
});
}
return res.status(200).json({
success:true,
existingPosts:posts
});
});
});
//view Reservation
router.route("/viewReservations").get((req,res)=>{
tourPackageReservation.find().then((reservations)=>{
res.json(reservations)
}).catch((err)=>{
console.log(err);
})
})
//view specific Reservation
router.get("/get/:id",(req,res) => {
let postID = req.params.id;
tourPackageReservation.findById(postID,(err,post) =>{
if(err) {
return res.status(400).json({success:false, err});
}
return res.status(200).json({success:true,post});
});
});
//update Reservation
router.put('/UpdateReservation/:id',(req,res) => {
tourPackageReservation.findByIdAndUpdate(
req.params.id,
{
$set:req.body
},
(err,post) => {
if(err){
return res.status(400).json({error:err});
}
return res.status(200).json({
success:"Updated Successfully"
});
}
);
});
//delete reservation
router.delete('/deleteReservation/:id',(req,res) => {
tourPackageReservation.findByIdAndRemove(req.params.id).exec((err,deletePackage) => {
if(err) return res.status(400).json({
message:"Delete Unsuccessfully",err
});
return res.json({
message:"Delete Successfully",deletePackage
});
});
});
module.exports = router;

Figure 106

Frontend



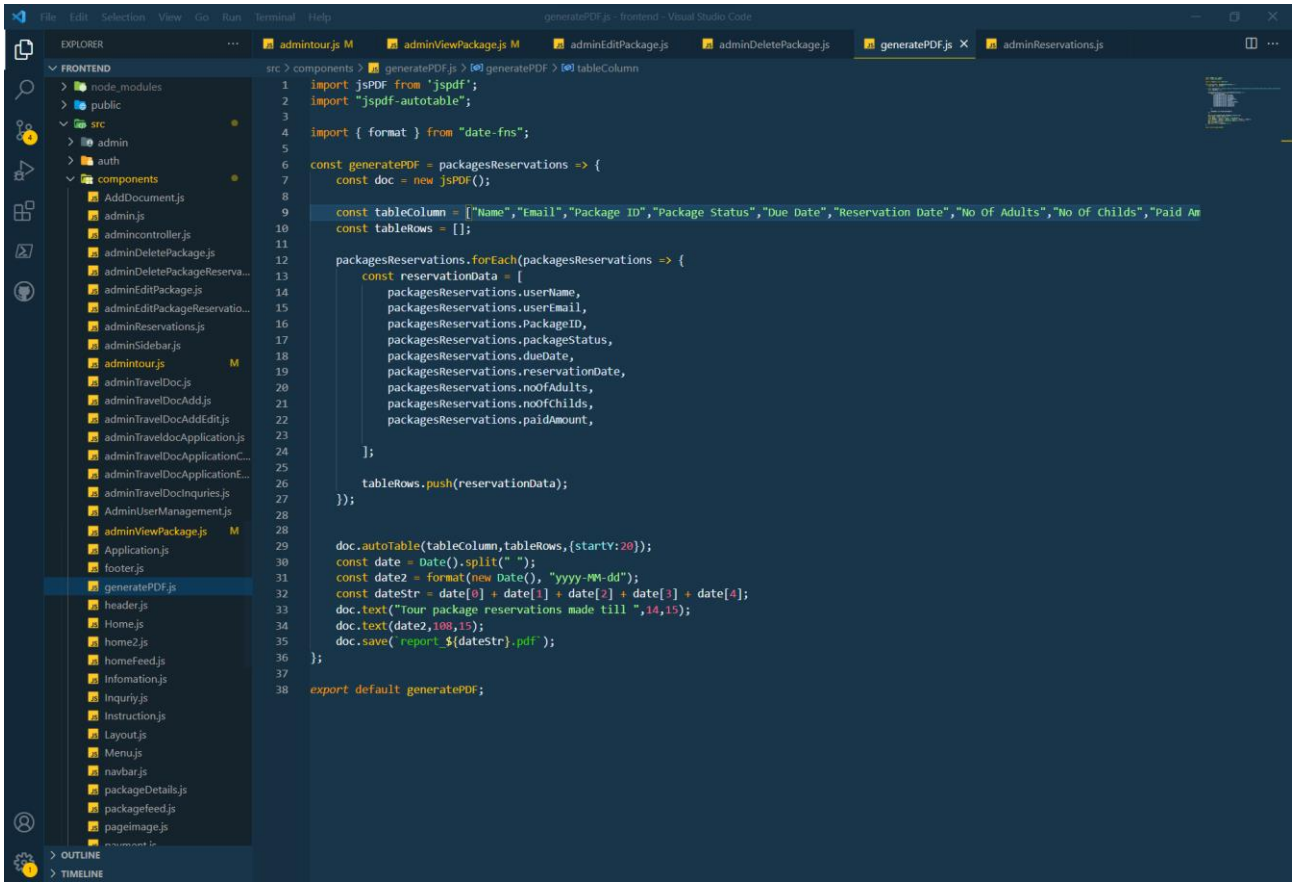

adminDeletePackage.js - frontend - Visual Studio Code
admintour.js M
adminViewPackage.js M
adminEditPackage.js
adminDeletePackage.js
src > components > adminDeletePackage.js > adminDeletePackage > render
import React, { Component } from 'react'
import "../style/adminDeletePackage.css";
import DeleteIcon from '@material-ui/icons/Delete';
import axios from 'axios';
export default class adminDeletePackage extends Component {
componentDidMount() {
const id = this.props.match.params.id;
axios.get(`http://localhost:8000/tourPackage/get/${id}`).then((res) => {
if (res.data.success) {
this.setState({
post: res.data.post
});
console.log(this.state.post);
}
});
}
onDelete = (id) => {
axios.delete(`http://localhost:8000/tourPackage/deletePackage/${id}`).then((res) => {
alert("Delete Successfully");
})
}
render() {
const id = this.props.match.params.id;
return (

<DeleteIcon style={{ fontSize: 80, color: "red" }} />
<h1>Delete Package !</h1>

<form action="/adminTourHome">
<button type="submit" class="btn btn-danger" onClick={() => this.onDelete(id)}>Delete</button>

<button type="submit" class="btn btn-success">Cancel</button>
</form>

)
}
}


generatePDF.js - frontend - Visual Studio Code
admintour.js M
adminViewPackage.js M
adminEditPackage.js
adminDeletePackage.js
generatePDF.js
adminReservations.js
src > components > generatePDF.js > generatePDF > tableColumn
import jsPDF from 'jspdf';
import "jspdf-autotable";
import { format } from "date-fns";
const generatePDF = packagesReservations => {
const doc = new jsPDF();
const tableColumn = ["Name","Email","Package ID","Package Status","Due Date","Reservation Date","No Of Adults","No Of Childs","Paid Am
const tableRows = [];
packagesReservations.forEach(packagesReservations => {
const reservationData = [
packagesReservations.userName,
packagesReservations.userEmail,
packagesReservations.PackageID,
packagesReservations.packageStatus,
packagesReservations.dueDate,
packagesReservations.reservationDate,
packagesReservations.noOfAdults,
packagesReservations.noOfChilds,
packagesReservations.paidAmount,
];
tableRows.push(reservationData);
});
doc.autoTable(tableColumn,tableRows,{startY:20});
const date = Date().split(" ");
const date2 = format(new Date(), "yyyy-MM-dd");
const dateStr = date[0] + date[1] + date[2] + date[3] + date[4];
doc.text("Tour package reservations made till ",14,15);
doc.text(date2,108,15);
doc.save(`report_${dateStr}.pdf`);
};
export default generatePDF;


File Edit Selection View Go Run Terminal Help
packagefeed.js - frontend - Visual Studio Code
EXPLORER
FRONTEND
admintour.js M
adminViewPackage.js M
adminEditPackage.js
adminDeletePackage.js
generatePDF.js
sidefilter.js
packagefeed.js M
src > components > packagefeed.js > ...
adminTravelDocApplicationC...
adminTravelDocApplicationE...
adminTravelDocInquries.js
AdminUserManagement.js
adminViewPackage.js M
Application.js
footer.js
generatePDF.js
header.js
Home.js
home2.js
homeFeed.js
Infomation.js
Inquriy.js
Instruction.js
Layout.js
Menu.js
navbar.js
packageDetails.js
packagefeed.js M
pageimage.js
payment.js
sidefilter.js
sidenavigation.js
sidenavigationData.js
tour.js
tourheader.js
tourpackage.js
userReservation.js
images
style
user
App.js
config.js
index.js
.env
.gitignore
package-lock.json
package.json
OUTLINE
TIMELINE
import "../style/packagefeed.css";
import axios from "axios";
import React, { Component } from 'react'
export default class packagefeed extends Component {
constructor(props) {
super(props);
this.state = {
posts: []
};
}
componentDidMount() {
this.retrievePackages();
}
retrievePackages() {
axios.get("http://localhost:8000/tourPackage/getAll").then(res => {
if (res.data.success) {
this.setState({
posts: res.data.existingPosts
});
}
});
console.log(this.state.posts);
}
filterData(posts, searchKey) {
const result = posts.filter((post) =>
post.packageName.toLowerCase().includes(searchKey) || post.packageName.includes(searchKey)
)
this.setState({ posts: result })
}
handleSearchArea = (e) => {
const searchKey = e.currentTarget.value;
console.log(e.currentTarget.value);
axios.get("http://localhost:8000/tourPackage/getAll").then(res => {
if (res.data.success) {
this.filterData(res.data.existingPosts, searchKey)
}
});
}
render() {
return (

<h3 style={{textAlign:"center",marginTop:20}}>
Latest Packages
</h3>
<hr />

<input
type="text"
class=" search-query form-control"
placeholder="Search"
name="searchQueue"
onchange={this.handleSearchArea}
/>

<button class="btn btn-danger" type="button" style={{marginLeft:5,marginTop:10}}>

</button>

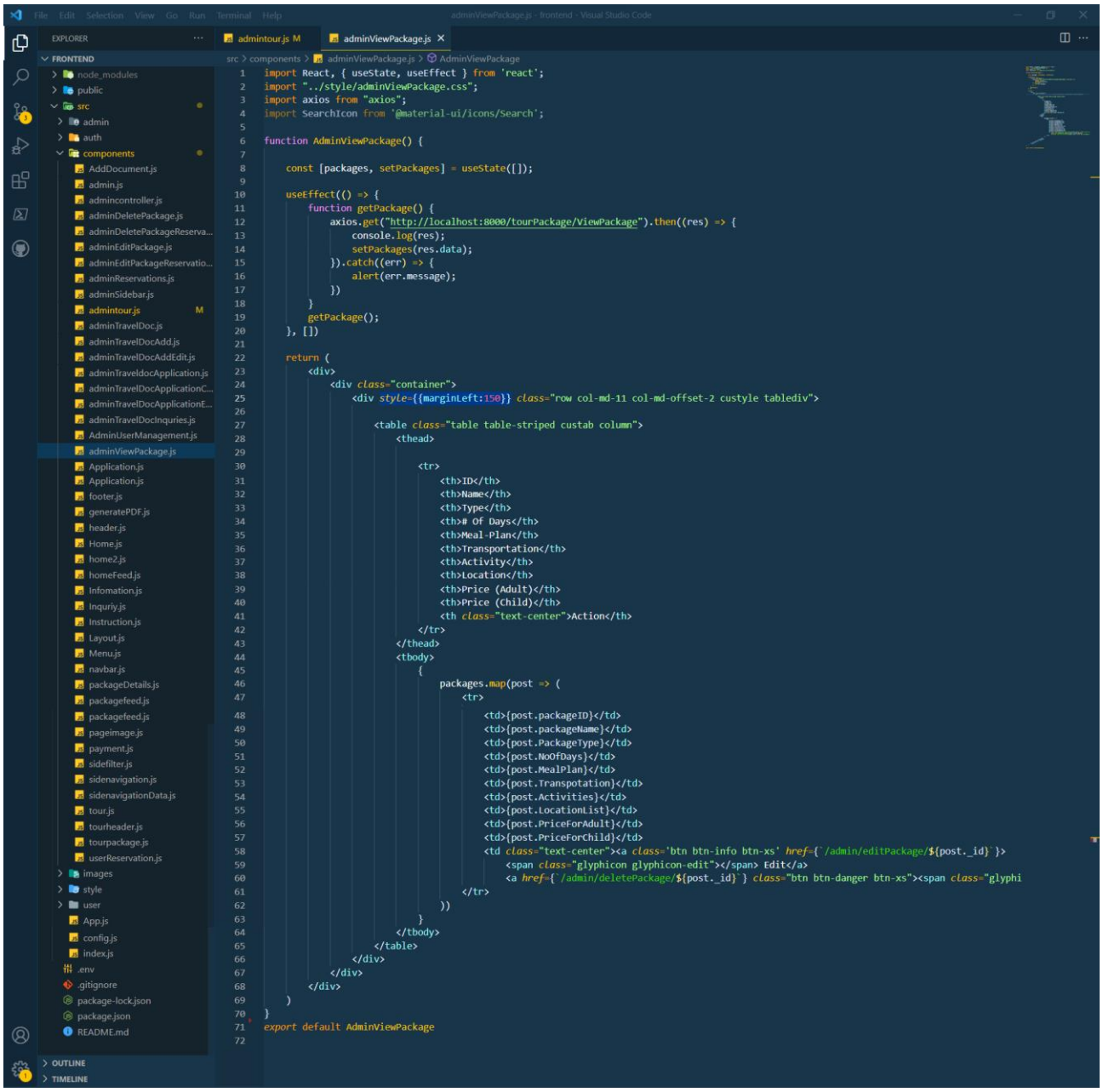
{this.state.posts.map(post => (

<h5 class="card-title">{post.packageName}</h5>
<p style={{ color: "#C8C0BE" }} class="card-text">Type : {post.PackageType} </p>
<a href={`/package/${post._id}`} style={{ backgroundColor: "#BCBBBB", color: "black" }} class

))
}

<hr />

)
}


File Edit Selection View Go Run Terminal Help
adminViewPackage.js - frontend - Visual Studio Code
EXPLORER
admintour.js M
adminViewPackage.js
src > components > adminViewPackage.js > AdminViewPackage
FRONTEND
node_modules
public
src
admin
auth
components
AddDocument.js
admin.js
admincontroller.js
adminDeletePackage.js
adminDeletePackageReserva...
adminEditPackage.js
adminEditPackageReservatio...
adminReservations.js
adminSidebar.js
admintour.js M
adminTravelDoc.js
adminTravelDocAdd.js
adminTravelDocAddEdit.js
adminTraveldocApplication.js
adminTravelDocApplicationC...
adminTravelDocApplicationE...
adminTravelDocInquries.js
AdminUserManagement.js
adminViewPackage.js
Application.js
Application.js
footer.js
generatePDF.js
header.js
Home.js
home2.js
homeFeed.js
Infomation.js
Inquriy.js
Instruction.js
Layout.js
Menu.js
navbar.js
packageDetails.js
packagefeed.js
packagefeed.js
pageimage.js
payment.js
sidefilter.js
sidenavigation.js
sidenavigationData.js
tour.js
tourheader.js
tourpackage.js
userReservation.js
images
style
user
App.js
config.js
index.js
.env
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
import React, { useState, useEffect } from 'react';
import "../style/adminViewPackage.css";
import axios from "axios";
import SearchIcon from '@material-ui/icons/Search';
function AdminViewPackage() {
const [packages, setPackages] = useState([]);
useEffect(() => {
function getPackage() {
axios.get("http://localhost:8000/tourPackage/ViewPackage").then((res) => {
console.log(res);
setPackages(res.data);
}).catch((err) => {
alert(err.message);
})
}
getPackage();
}, [])
return (

<table class="table table-striped custab column">
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th># Of Days</th>
<th>Meal-Plan</th>
<th>Transportation</th>
<th>Activity</th>
<th>Location</th>
<th>Price (Adult)</th>
<th>Price (Child)</th>
<th class="text-center">Action</th>
</tr>
</thead>
<tbody>
{
packages.map(post => (
<tr>
<td>{post.packageID}</td>
<td>{post.packageName}</td>
<td>{post.PackageType}</td>
<td>{post.NoOfDays}</td>
<td>{post.MealPlan}</td>
<td>{post.Transpotation}</td>
<td>{post.Activities}</td>
<td>{post.LocationList}</td>
<td>{post.PriceForAdult}</td>
<td>{post.PriceForChild}</td>
<td class="text-center">
 Edit
<a href={ /admin/deletePackage/${post._id} } class="btn btn-danger btn-xs"><span class="glyphi
</tr>
))
}
</tbody>
</table>

)
}
export default AdminViewPackage

Figure 107

2.3.3.7 Insurance Management

Models

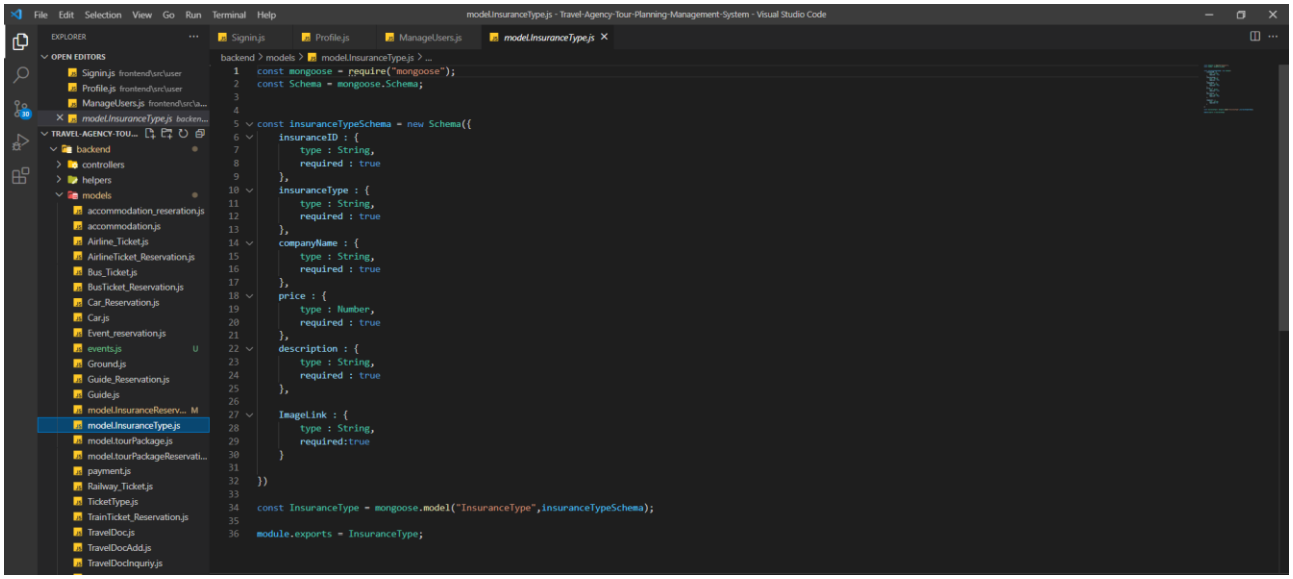
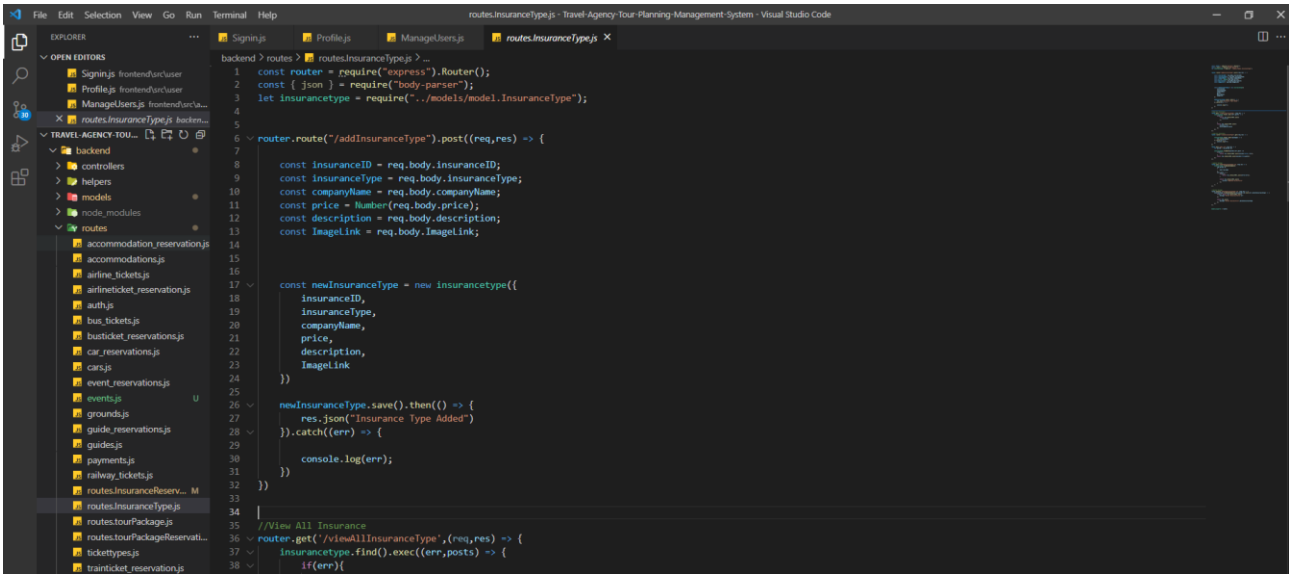
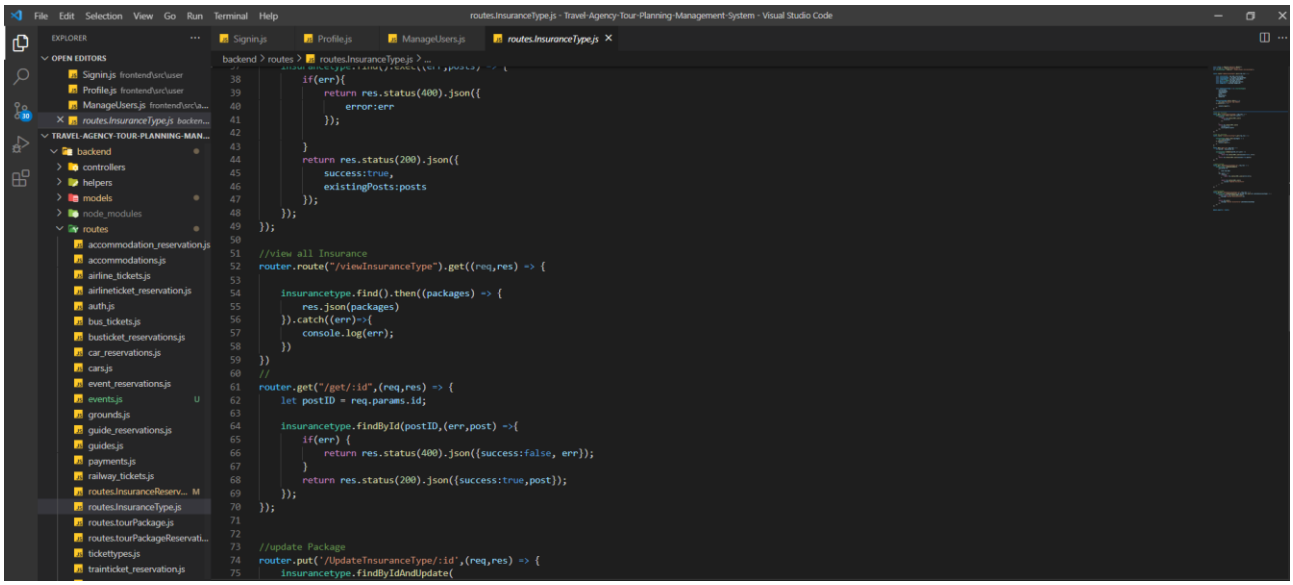


Figure 108

Routes




File Edit Selection View Go Run Terminal Help
routes.InsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
Signin.js frontend\src\user
Profile.js frontend\src\user
ManageUsers.js frontend\src\a...
routes.InsuranceType.js backen...
TRAVEL-AGENCY-TOUR-PLANNING-MAN...
backend
controllers
helpers
models
node_modules
routes
accommodation_reservation.js
accommodations.js
airline_tickets.js
airlineticket_reservation.js
auth.js
bus_tickets.js
busticket_reservations.js
car_reservations.js
cars.js
event_reservations.js
events.js
grounds.js
guide_reservations.js
guides.js
payments.js
railway_tickets.js
routes.InsuranceReserv... M
routes.InsuranceType.js
routes.tourPackage.js
routes.tourPackageReservati...
tickettypes.js
trainticket_reservation.js
Signin.js
Profile.js
ManageUsers.js
routes.InsuranceType.js
backend > routes > routes.InsuranceType.js > ...
if(err){
return res.status(400).json({
error:err
});
}
return res.status(200).json({
success:true,
existingPosts:posts
});
});
});
//view all Insurance
router.route("/viewInsuranceType").get((req,res) => {
insurancetype.find().then((packages) => {
res.json(packages)
}).catch((err)=>{
console.log(err);
})
})
//
router.get("/get/:id",(req,res) => {
let postID = req.params.id;
insurancetype.findById(postID,(err,post) =>{
if(err) {
return res.status(400).json({success:false, err});
}
return res.status(200).json({success:true,post});
});
});
//update Package
router.put('/UpdateInsuranceType/:id',(req,res) => {
insurancetype.findByIdAndUpdate(

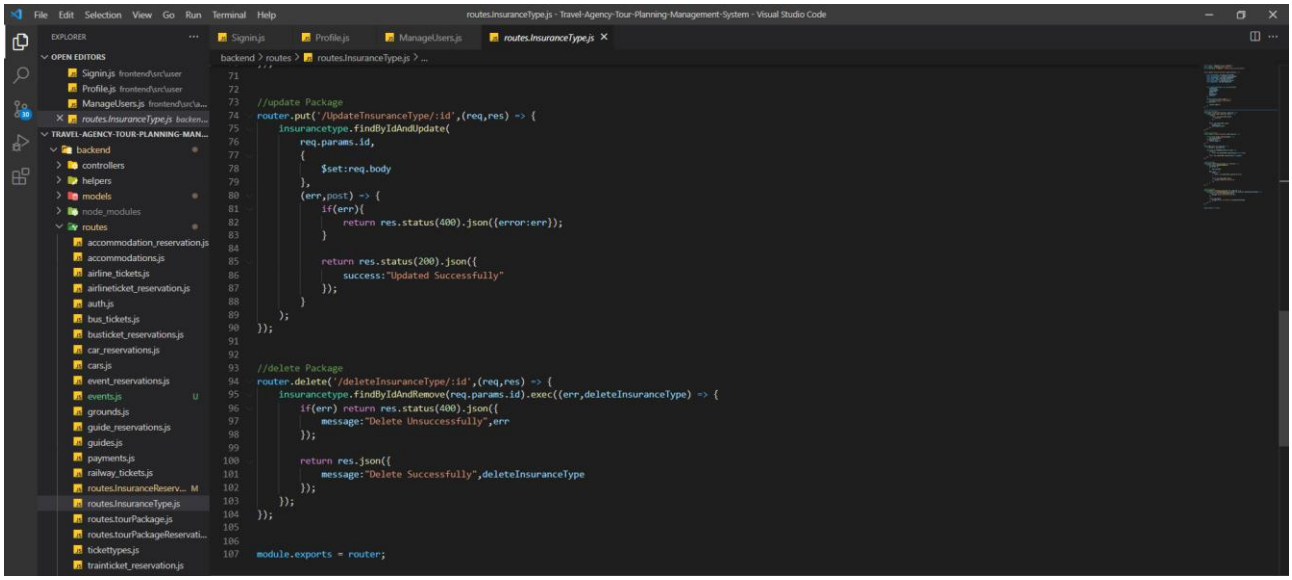

File Edit Selection View Go Run Terminal Help
routes.InsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
Signin.js frontend\src\user
Profile.js frontend\src\user
ManageUsers.js frontend\src\a...
routes.InsuranceType.js backen...
TRAVEL-AGENCY-TOUR-PLANNING-MAN...
backend
controllers
helpers
models
node_modules
routes
accommodation_reservation.js
accommodations.js
airline_tickets.js
airlineticket_reservation.js
auth.js
bus_tickets.js
busticket_reservations.js
car_reservations.js
cars.js
event_reservations.js
events.js
grounds.js
guide_reservations.js
guides.js
payments.js
railway_tickets.js
routes.InsuranceReserv... M
routes.InsuranceType.js
routes.tourPackage.js
routes.tourPackageReservati...
tickettypes.js
trainticket_reservation.js
Signin.js
Profile.js
ManageUsers.js
routes.InsuranceType.js
backend > routes > routes.InsuranceType.js > ...
//update Package
router.put('/UpdateInsuranceType/:id',(req,res) => {
insurancetype.findByIdAndUpdate(
req.params.id,
{
$set:req.body
},
(err,post) => {
if(err){
return res.status(400).json({error:err});
}
return res.status(200).json({
success:"Updated Successfully"
});
}
);
});
//delete Package
router.delete('/deleteInsuranceType/:id',(req,res) => {
insurancetype.findByIdAndRemove(req.params.id).exec((err,deleteInsuranceType) => {
if(err) return res.status(400).json({
message:"Delete Unsuccessfully",err
});
return res.json({
message:"Delete Successfully",deleteInsuranceType
});
});
});
module.exports = router;

Figure 109

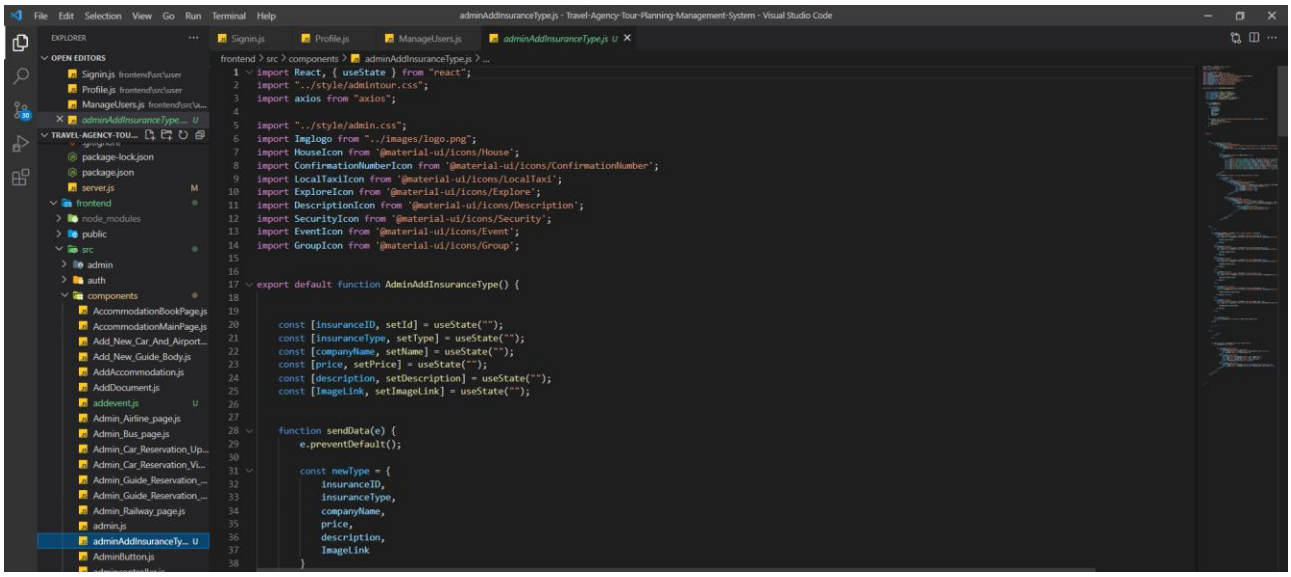
Frontend

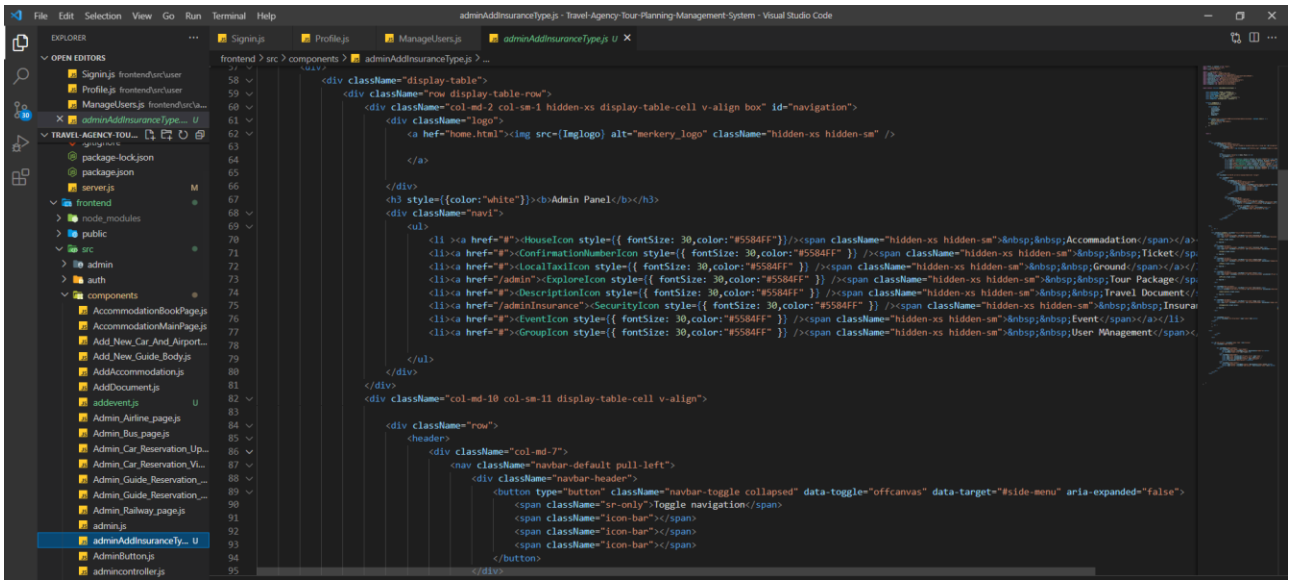

frontend > src > components > adminViewInsuranceType.js > AdminViewInsuranceType > insurancetype.map() callback
import React, { useState, useEffect } from 'react';
import "../style/adminViewPackage.css";
import axios from "axios";
import SearchIcon from '@material-ui/icons/Search';
function AdminViewInsuranceType() {
const [insurancetype, setInsurancetype] = useState([]);
useEffect(() => {
function getInsurancetype() {
axios.get("http://localhost:8000/InsuranceType/viewInsuranceType").then((res) => {
console.log(res);
setInsurancetype(res.data);
}).catch((err) => {
alert(err.message);
})
}
getInsurancetype();
}, [])
return (
<div style={{marginLeft:"400px",fontSize:"20px"}}>

<h1>Insurance Types
<button className="btn btn-primary" type="submit" style={{float:'righ
Add New Type
</button></h1>

<table class="table table-striped custab column">
<thead>
<tr>
<th>Insurance ID</th>
<th>Insurance Type</th>
<th>Company Name</th>
<th>Price</th>
<th>Description</th>
<th class="text-center">Action</th>
</tr>
</thead>
<tbody>
{
insurancetype.map(post => (
<tr>
<td>{post.insuranceID}</td>
<td>{post.insuranceType}</td>
<td>{post.companyName}</td>
<td>{post.price}</td>
<td>{post.description}</td>
<td class="text-center">
 Edit
<a href={`/adminInsurance/deleteType/${post._id}`} class="btn btn-danger btn-xs"><span class="glyphic
</tr>
))
}
</tbody>
</table>

)
}
export default AdminViewInsuranceType


File Edit Selection View Go Run Terminal Help
adminAddInsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
Signin.js frontend\src\user
Profile.js frontend\src\user
ManageUsers.js frontend\src\a...
adminAddInsuranceType.... U
TRAVEL-AGENCY-TOU...
package-lock.json
package.json
server.js M
frontend
node_modules
public
src
admin
auth
components
AccommodationBookPage.js
AccommodationMainPage.js
Add_New_Car_And_Airport...
Add_New_Guide_Body.js
AddAccommodation.js
AddDocument.js
addevent.js U
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Car_Reservation_Up...
Admin_Car_Reservation_Vi...
Admin_Guide_Reservation_...
Admin_Guide_Reservation_...
Admin_Railway_page.js
admin.js
adminAddInsuranceTy... U
AdminButton.js
Signin.js
Profile.js
ManageUsers.js
adminAddInsuranceType.js U
frontend > src > components > adminAddInsuranceType.js > ...
import React, { useState } from "react";
import "../style/admintour.css";
import axios from "axios";
import "../style/admin.css";
import Imglogo from "../images/logo.png";
import HouseIcon from '@material-ui/icons/House';
import ConfirmationNumberIcon from '@material-ui/icons/ConfirmationNumber';
import LocalTaxiIcon from '@material-ui/icons/LocalTaxi';
import ExploreIcon from '@material-ui/icons/Explore';
import DescriptionIcon from '@material-ui/icons/Description';
import SecurityIcon from '@material-ui/icons/Security';
import EventIcon from '@material-ui/icons/Event';
import GroupIcon from '@material-ui/icons/Group';
export default function AdminAddInsuranceType() {
const [insuranceID, setId] = useState("");
const [insuranceType, setType] = useState("");
const [companyName, setName] = useState("");
const [price, setPrice] = useState("");
const [description, setDescription] = useState("");
const [ImageLink, setImageLink] = useState("");
function sendData(e) {
e.preventDefault();
const newType = {
insuranceID,
insuranceType,
companyName,
price,
description,
ImageLink
}


File Edit Selection View Go Run Terminal Help
adminAddInsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
TRAVEL-AGENCY-TOU...
frontend
src
components
admin
auth
Signin.js
Profile.js
ManageUsers.js
adminAddInsuranceType.js U
frontend > src > components > adminAddInsuranceType.js > ...

<h3 style={{color:"white"}}>Admin Panel</h3>

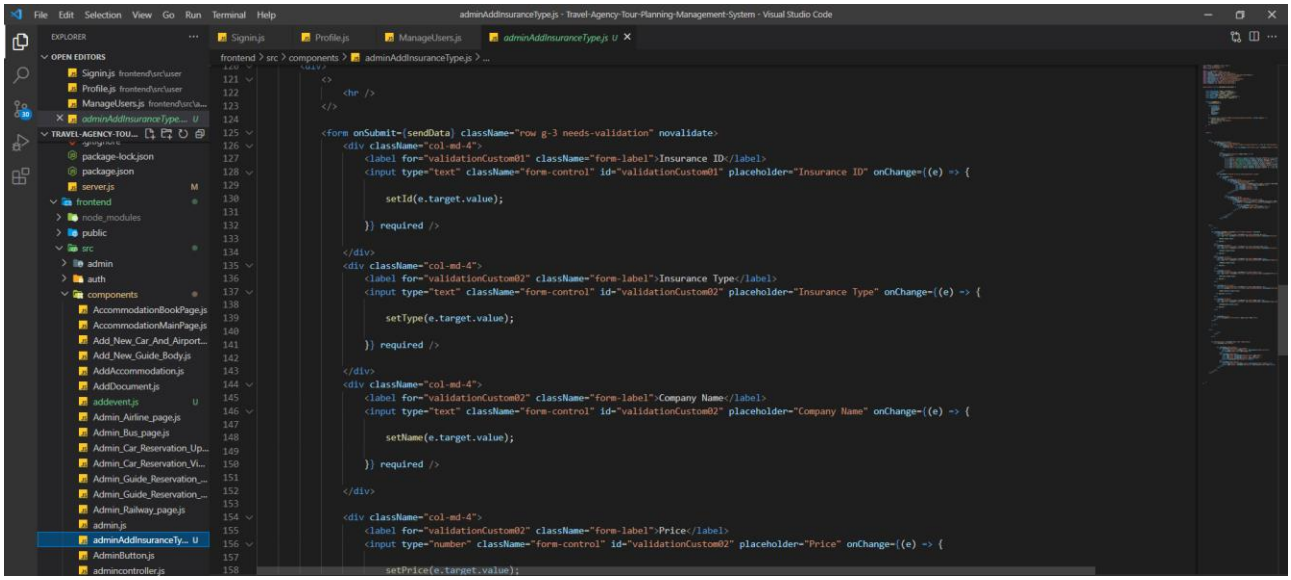
<HouseIcon style={{ fontSize: 30,color:"#5584FF"}}/> Accommadation
<ConfirmationNumberIcon style={{ fontSize: 30,color:"#5584FF" }} /> Ticket</sp
<LocalTaxiIcon style={{ fontSize: 30,color:"#5584FF" }} /> Ground</
<ExploreIcon style={{ fontSize: 30,color:"#5584FF" }} /> Tour Package</sp
<DescriptionIcon style={{ fontSize: 30,color:"#5584FF" }} /> Travel Document</
<SecurityIcon style={{ fontSize: 30,color:"#5584FF" }} /> Insura
<EventIcon style={{ fontSize: 30,color:"#5584FF" }} /> Event
<a href="#"><GroupIcon style={{ fontSize: 30,color:"#5584FF" }} /> User MAnagement<.

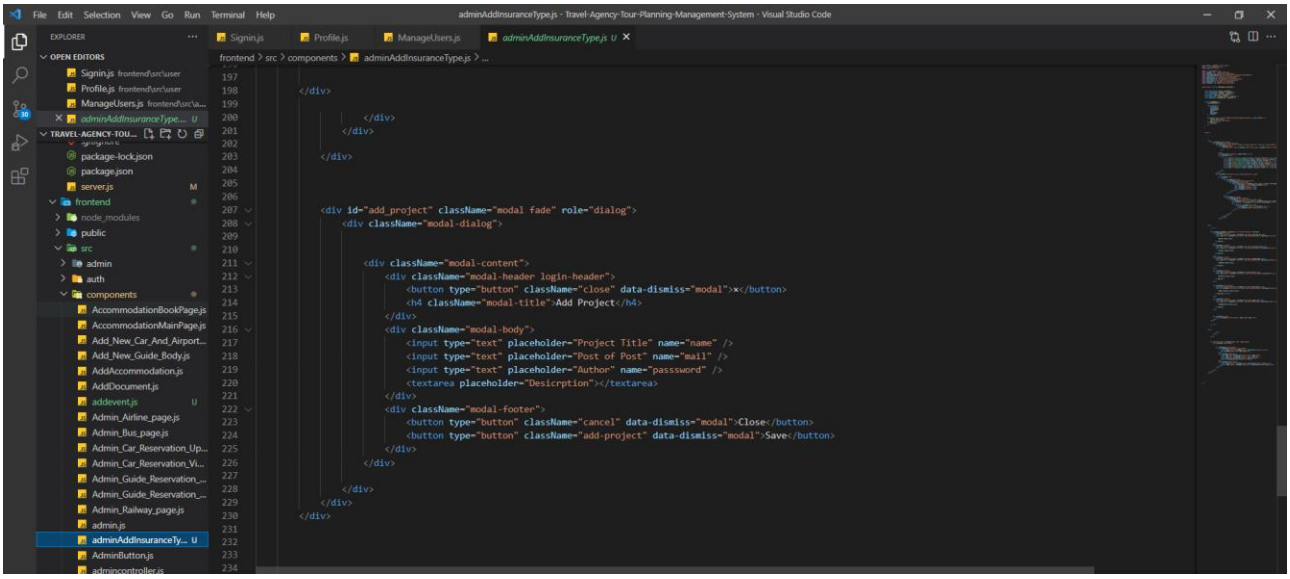
<header>

<nav className="navbar-default pull-left">

<button type="button" className="navbar-toggle collapsed" data-toggle="offcanvas" data-target="#side-menu" aria-expanded="false">
Toggle navigation

</button>


File Edit Selection View Go Run Terminal Help
adminAddInsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
Signin.js frontend\src\user
Profile.js frontend\src\user
ManageUsers.js frontend\src\a...
adminAddInsuranceType.... U
TRAVEL-AGENCY-TOU...
package-lock.json
package.json
server.js M
frontend
node_modules
public
src
admin
auth
components
AccommodationBookPage.js
AccommodationMainPage.js
Add_New_Car_And_Airport...
Add_New_Guide_Body.js
AddAccommodation.js
AddDocument.js
addevent.js U
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Car_Reservation_Up...
Admin_Car_Reservation_Vi...
Admin_Guide_Reservation_...
Admin_Guide_Reservation_...
Admin_Railway_page.js
admin.js
adminAddInsuranceTy... U
AdminButton.js
admincontroller.js
Signin.js
Profile.js
ManageUsers.js
adminAddInsuranceType.js U
frontend > src > components > adminAddInsuranceType.js > ...
<hr />
<form onSubmit={sendData} className="row g-3 needs-validation" novalidate>
<label for="validationCustom01" className="form-label">Insurance ID</label>
<input type="text" className="form-control" id="validationCustom01" placeholder="Insurance ID" onChange={(e) => {
setId(e.target.value);
}} required />
<label for="validationCustom02" className="form-label">Insurance Type</label>
<input type="text" className="form-control" id="validationCustom02" placeholder="Insurance Type" onChange={(e) => {
setType(e.target.value);
}} required />
<label for="validationCustom02" className="form-label">Company Name</label>
<input type="text" className="form-control" id="validationCustom02" placeholder="Company Name" onChange={(e) => {
setName(e.target.value);
}} required />
<label for="validationCustom02" className="form-label">Price</label>
<input type="number" className="form-control" id="validationCustom02" placeholder="Price" onChange={(e) => {
setPrice(e.target.value);


File Edit Selection View Go Run Terminal Help
adminAddInsuranceType.js - Travel-Agency-Tour-Planning-Management-System - Visual Studio Code
EXPLORER
OPEN EDITORS
Signin.js frontend\src\user
Profile.js frontend\src\user
ManageUsers.js frontend\src\a...
adminAddInsuranceType.... U
TRAVEL-AGENCY-TOU...
package-lock.json
package.json
server.js M
frontend
node_modules
public
src
admin
auth
components
AccommodationBookPage.js
AccommodationMainPage.js
Add_New_Car_And_Airport...
Add_New_Guide_Body.js
AddAccommodation.js
AddDocument.js
addevent.js U
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Car_Reservation_Up...
Admin_Car_Reservation_Vi...
Admin_Guide_Reservation_...
Admin_Guide_Reservation_...
Admin_Railway_page.js
admin.js
adminAddInsuranceTy... U
AdminButton.js
admincontroller.js
Signin.js
Profile.js
ManageUsers.js
adminAddInsuranceType.js U
frontend > src > components > adminAddInsuranceType.js > ...

<button type="button" className="close" data-dismiss="modal">x</button>
<h4 className="modal-title">Add Project</h4>

<input type="text" placeholder="Project Title" name="name" />
<input type="text" placeholder="Post of Post" name="mail" />
<input type="text" placeholder="Author" name="passsword" />
<textarea placeholder="Desicrption"></textarea>

<button type="button" className="cancel" data-dismiss="modal">Close</button>
<button type="button" className="add-project" data-dismiss="modal">Save</button>

frontend > src > components > JS adminDeleteInsuranceType.js > ...

```
import React, { Component } from 'react'
import "../style/adminDeletePackage.css";
import DeleteIcon from '@material-ui/icons/Delete';
import axios from 'axios';

export default class AdminDeleteInsuranceType extends Component {

    componentDidMount() {

        const id = this.props.match.params.id;

        axios.get(`http://localhost:8000/InsuranceType/get/${id}`).then((res) => {
            if (res.data.success) {
                this.setState({
                    post: res.data.post
                });

                console.log(this.state.post);
            }
        });
    }

    onDelete = (id) => {
        axios.delete(`http://localhost:8000/InsuranceType/deleteInsuranceType/${id}`).then((res) => {
            alert("Delete Successfully");
        });
    }
    render() {
        const id = this.props.match.params.id;
        return (
            <div className="delete">
                <div class="wrapper fadeInDown">
                    <div id="formContent">
                        <div class="fadeIn first">
                            <DeleteIcon style={{ fontSize: 80, color: "red" }} />
                            <h1>Delete Insurance Type !</h1>
                        </div>
                        <div id="formFooter">

                            <form action="/adminInsurance">
                                <button class="btn btn-danger" onClick={() => this.onDelete(id)}>Delete</button>
                                <span>      </span>
                                <button class="btn btn-success">Cancel</button>
                            </form>
                        </div>

                    </div>
                </div>
            </div >
        )
    }
}

```


frontend > src > components > JS InsuranceFeed.js > ...
import React, { Component } from "react";
//import React, { useState, useEffect } from 'react'
import "../style/packagefeed.css";
import axios from "axios";
export default class InsuranceFeed extends Component {
constructor(props) {
super(props);
this.state = {
posts: []
};
}
componentDidMount() {
this.retrieveInsurance();
}
retrieveInsurance() {
axios.get("http://localhost:8000/InsuranceType/viewAllInsuranceType").then(res => {
if (res.data.success) {
this.setState({
posts: res.data.existingPosts
});
}
});
console.log(this.state.posts);
}
filterData(posts, searchKey) {
const result = posts.filter((post) =>
post.insuranceType.toLowerCase().includes(searchKey) || post.insuranceType.includes(searchKey) ||
post.companyName.toLowerCase().includes(searchKey) || post.companyName.includes(searchKey)
)
this.setState({ posts: result })
}
handleSearchArea = (e) => {
const searchKey = e.currentTarget.value;
console.log(e.currentTarget.value);
axios.get("http://localhost:8000/InsuranceType/viewAllInsuranceType").then(res => {
if (res.data.success) {
this.filterData(res.data.existingPosts, searchKey)
}
});
}
render() {
return (
<div style={{fontSize:"20px"}}>

<h3 style={{fontSize:"20px",textAlign:"center",marginTop:20}}>
Insurance
</h3>

<input
type="text"
class=" search-query form-control"
placeholder="Search"
name="searchQueue"
onChange={this.handleSearchArea}
/>

<button class="btn btn-danger" type="button" style={{marginLeft:5,marginTop:10}}>

</button>

<hr />

{this.state.posts.map(post => (

<h5 style={{fontSize:"20px"}} className="card-title">{post.insuranceType}</h5>
<p style={{ fontSize:"20px",color: "#C8C0BE" }} className="card-text">Company : {post.companyName} </
<a href={`/Insurance/${post._id}`} style={{fontSize:"20px", backgroundColor: "#BCBBBB", color: "black

))
}

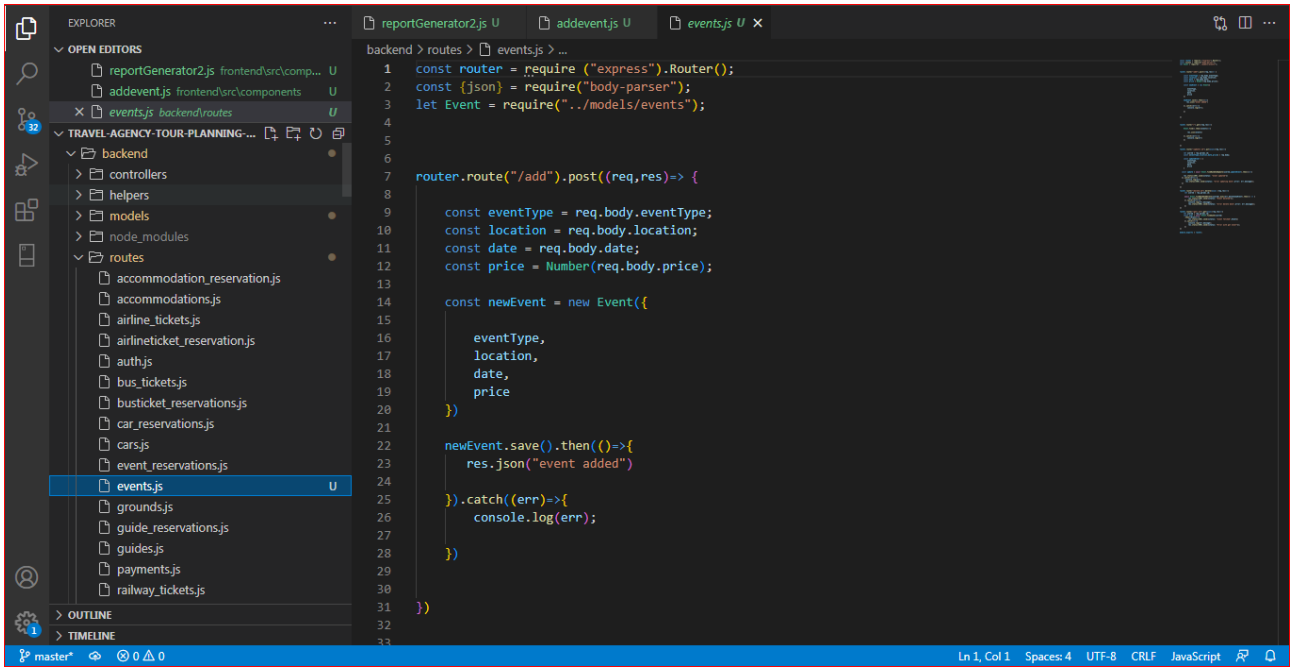
<hr />

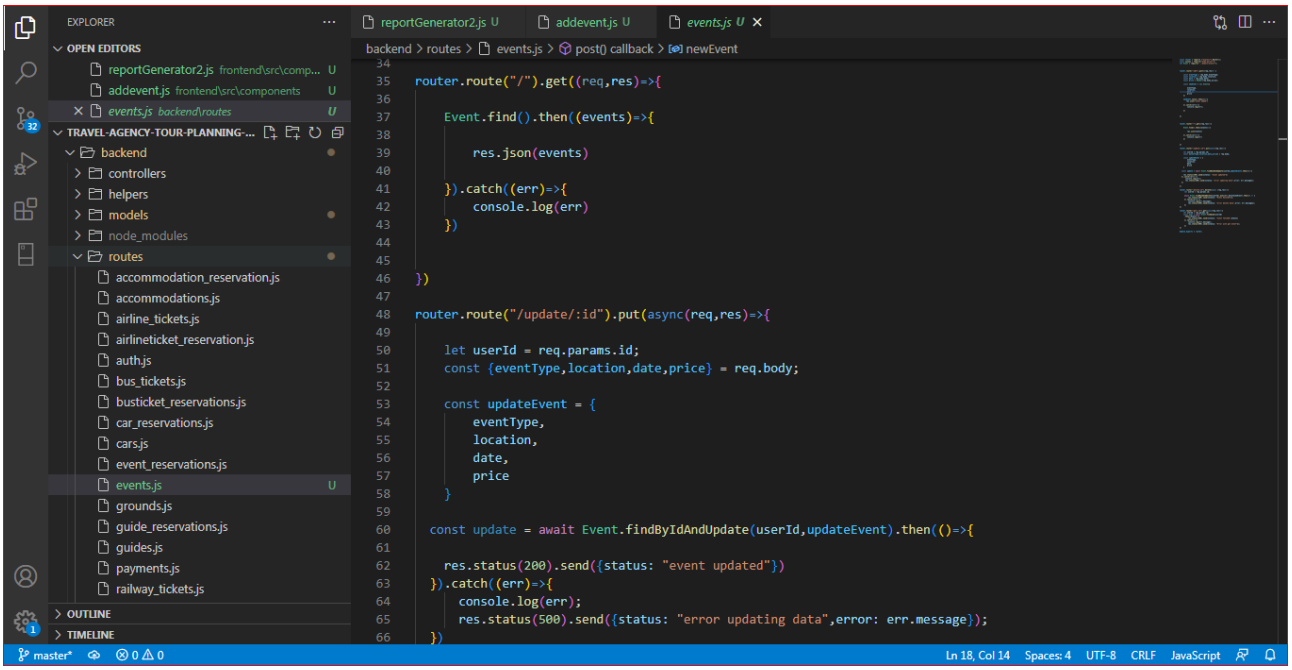
)
}
}

Figure 110

2.3.3.8 Event Management

Routes





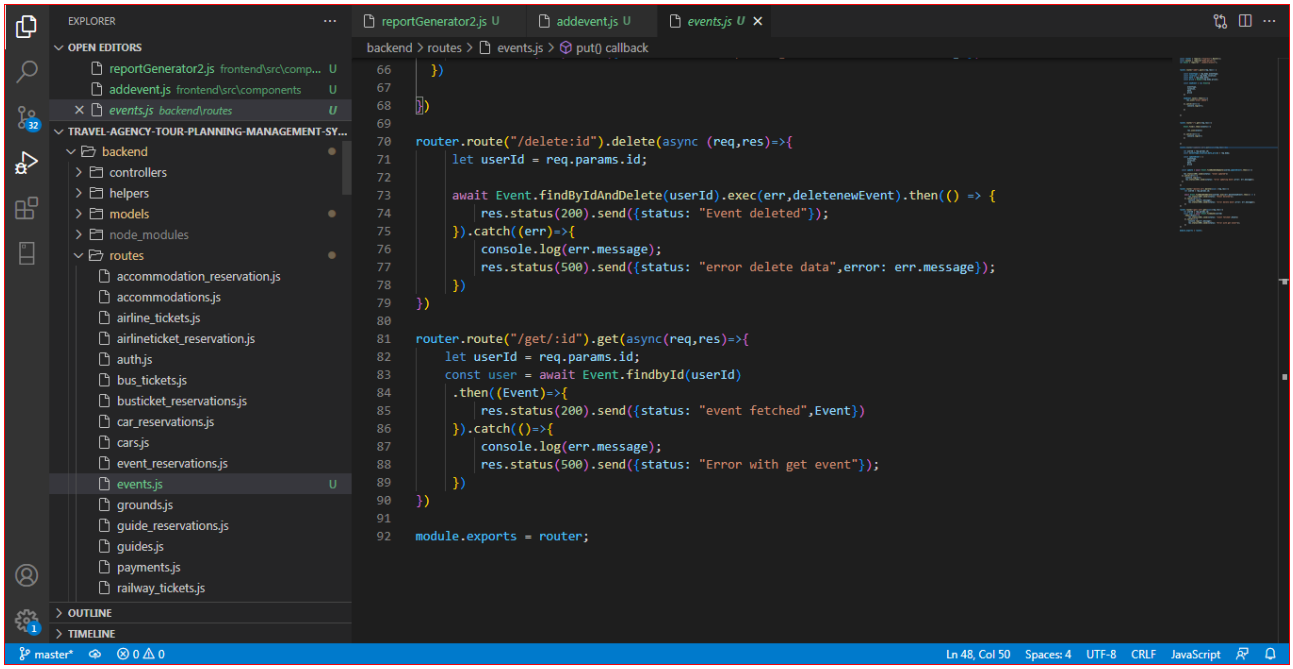
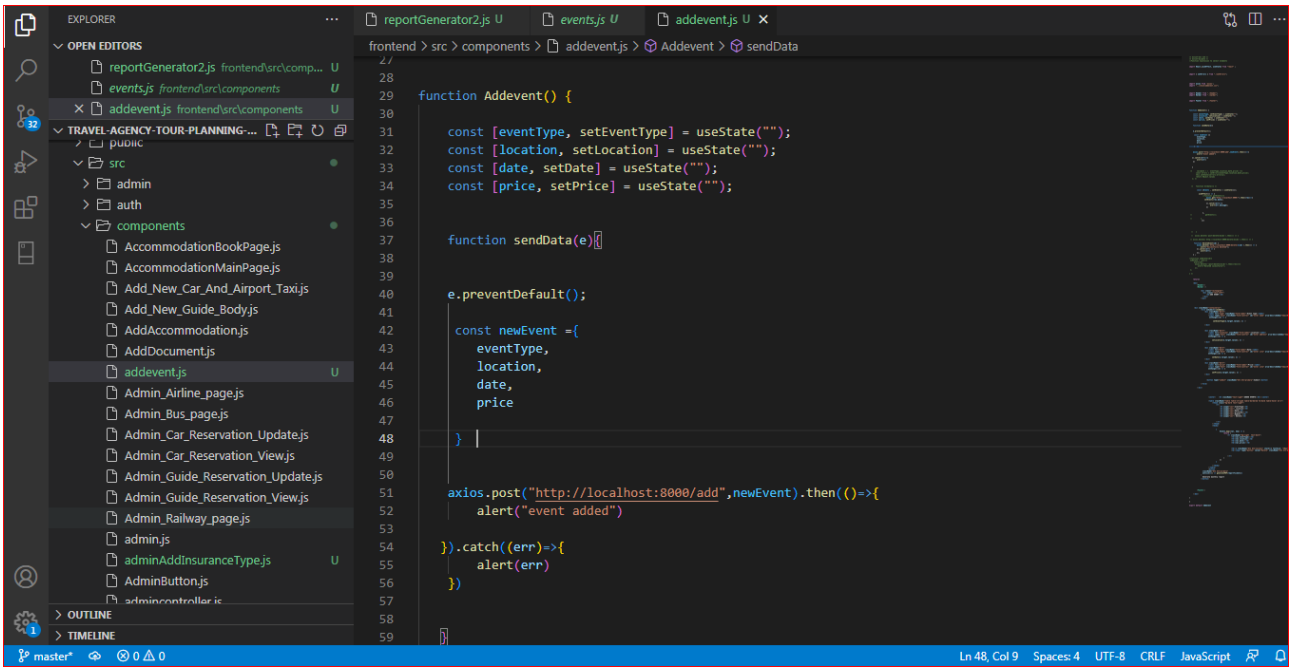
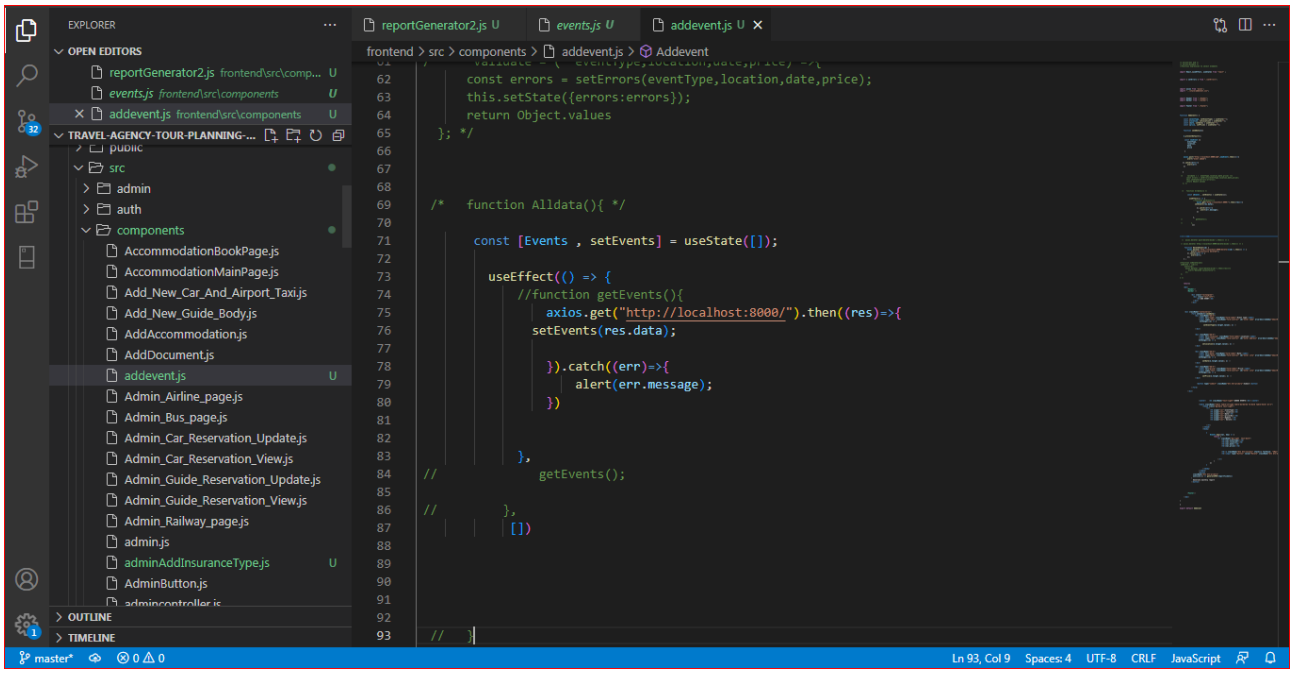


Figure 111

Frontend




EXPLORER
OPEN EDITORS
reportGenerator2.js frontend\src\comp... U
events.js frontend\src\components U
addevent.js frontend\src\components U
TRAVEL-AGENCY-TOUR-PLANNING-...
public
src
admin
auth
components
AccommodationBookPage.js
AccommodationMainPage.js
Add_New_Car_And_Airport_Taxi.js
Add_New_Guide_Body.js
AddAccommodation.js
AddDocument.js
addevent.js U
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Car_Reservation_Update.js
Admin_Car_Reservation_View.js
Admin_Guide_Reservation_Update.js
Admin_Guide_Reservation_View.js
Admin_Railway_page.js
admin.js
adminAddInsuranceType.js U
AdminButton.js
OUTLINE
TIMELINE
reportGenerator2.js U
events.js U
addevent.js U
frontend > src > components > addevent.js > Addevent
const errors = setErrors(eventType,location,date,price);
this.setState({errors:errors});
return Object.values
}; */
/* function Alldata(){ */
const [Events , setEvents] = useState([]);
useEffect(() => {
//function getEvents(){
axios.get("http://localhost:8000/").then((res)=>{
setEvents(res.data);
}).catch((err)=>{
alert(err.message);
})
},
// getEvents();
// },
[])
// }
master*
Ln 93, Col 9
Spaces: 4
UTF-8
CRLF
JavaScript

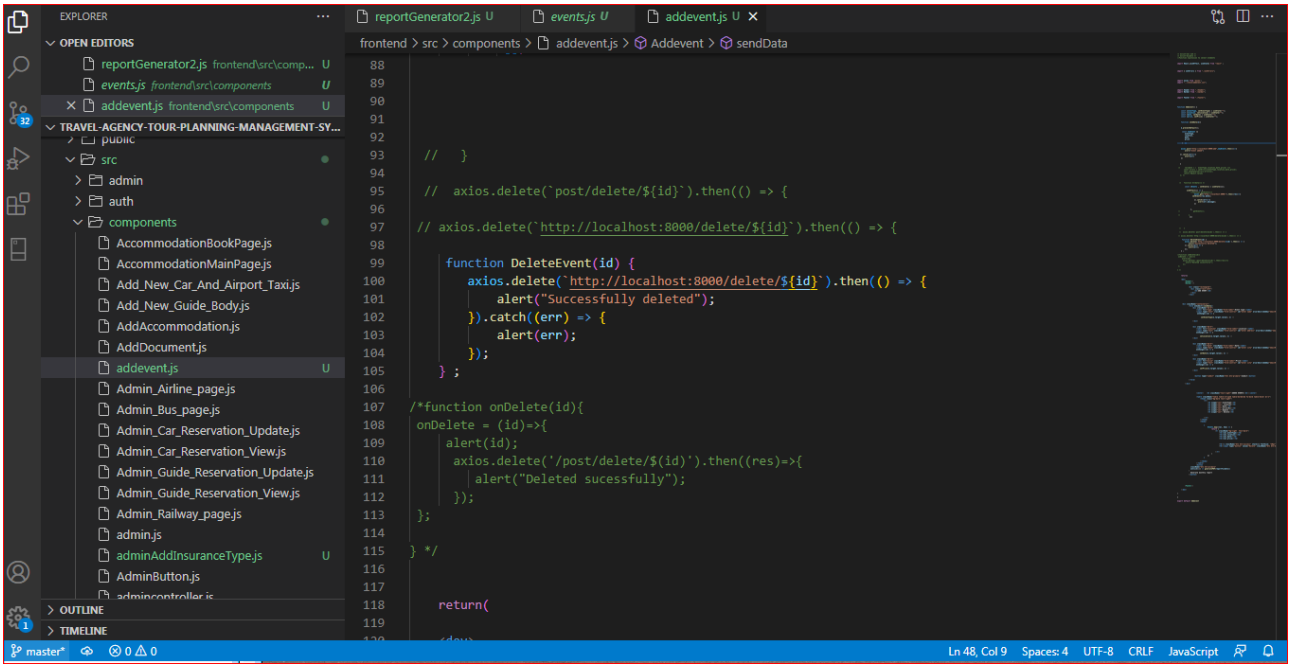

EXPLORER
OPEN EDITORS
reportGenerator2.js frontend\src\comp... U
events.js frontend\src\components U
addevent.js frontend\src\components U
TRAVEL-AGENCY-TOUR-PLANNING-MANAGEMENT-SY...
public
src
admin
auth
components
AccommodationBookPage.js
AccommodationMainPage.js
Add_New_Car_And_Airport_Taxi.js
Add_New_Guide_Body.js
AddAccommodation.js
AddDocument.js
addevent.js U
Admin_Airline_page.js
Admin_Bus_page.js
Admin_Car_Reservation_Update.js
Admin_Car_Reservation_View.js
Admin_Guide_Reservation_Update.js
Admin_Guide_Reservation_View.js
Admin_Railway_page.js
admin.js
adminAddInsuranceType.js U
AdminButton.js
OUTLINE
TIMELINE
reportGenerator2.js U
events.js U
addevent.js U
frontend > src > components > addevent.js > Addevent > sendData
// }
// axios.delete(`post/delete/${id}`).then(() => {
// axios.delete(`http://localhost:8000/delete/${id}`).then(() => {
function DeleteEvent(id) {
axios.delete(`http://localhost:8000/delete/${id}`).then(() => {
alert("Successfully deleted");
}).catch((err) => {
alert(err);
});
} ;
/*function onDelete(id){
onDelete = (id)=>{
alert(id);
axios.delete('/post/delete/${id}').then((res)=>{
alert("Deleted sucessfully");
});
};
} */
return(
master*
Ln 48, Col 9
Spaces: 4
UTF-8
CRLF
JavaScript

Figure 112

2.4 Testing

2.4.1 User Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: **User Registration and Sign-in into the system.**					
Test case ID: TC001			Test designed by: Reg. No- IT20142964 Name- Dulakshi Hansini R.M		
Test priority (High/Medium/Low):			High		
Test Description: User must create a user account before sign-in to the system. For that user must fill an application in registration page. Then user can log into the system with given email and password for the user registration.					
Preconditions (if there are any): User must create a valid user account before sign-in.					
Test steps:					
User must visit the web site. Click the button Sign-up in the header. Fill User registration form. Click REGISTER button after filling the form. Click Sign-in button. Enter Email and password given in the registration. Click Sign-out button to sign-out from account.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC001	Check user enter an invalid password or email for the valid user account login.	System sends an error message as "Email and password don't match"	Error message was showed as Email and password don't match	Pass	Error message was shown and the test case worked as described.

Testing Function: Admin can delete and view user profiles					
Test case ID: TC002			Test designed by: Reg. No- IT20142964 Name- Dulakshi Hansini R.M		
Test priority (High/Medium/Low):			Medium		
Test Description: Admin can log in to the system by giving correct email address and password. After login to the system admin can view all user profile details in manage user page by clicking user management button which is in the side bar.					
Preconditions (if there are any): Admin must sign into the system by giving correct email and password and move to admin panel					
Test steps:					
Step1: Admin must login to the website. Step2: Admin should Click user management button which is in the admin panel sidebar. Step3: It navigate to Admin dashboard and click view user profiles button Step4: Admin can see all the users that registered into the system. Step5: Admin can click delete button, if admin wants to delete any user profile in the system Step6: It will display the confirmation message that confirm the deletion Step7: Admin wants to confirm it Step8: Therefore, the record will be deleted and updated the user profiles details					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC002	Click on delete button and check whether the user profile deletion is successful	Admin can see “Successfully deleted” message	Admin can see “Successfully deleted” message	Pass	Admin can delete user profile is successfully completed

2.4.2 Accommodation reservation Management

Project ID:					
Project Name: Travel Agency Tour Planning Management System					
Testing Function: Book an accommodation					
Test case ID:TC003			Test case Designed By: ID: IT20204648 Name: Nethu Nipuna M.		
Test Priority (High/Medium/Low)			High		
Test Description: registered customer who visit the website can reservation accommodations on their own. Then total cost can be calculated. You can easily book the accommodation through the website.					
Test steps: Step1: Register to the web site Step2: log in to the web site Step3: go to accommodation page Step4: customer can search accommodations what he or she want Step5: click book now Step6: fill the given form and click book now button					
Test ID	Test Inputs	Expected output	Actual output	Result (pass/fail)	Comments
TC003	Name, email, booking date, due date, No of accommodations, No of adult, No of child	Show total price and when press book now button system shows success message.	Success message displayed	pass	Booking function Successfully completed

Testing Function: insert an accommodation					
Test case ID:TC004			Test case Designed By: ID: IT20204648 Name: Nethu Nipuna M.		
Test Priority (High/Medium/Low)			High		
Test Description: registered admin can log in to admin panel. Admin can view all accommodations and add new one, edit or delete accommodations from the database					
Test steps: Step1: log in to the admin side Step2: go to accommodation admin page Step3: go to add new accommodation page Step4: fill the form Step5: click submit button Step6: newly added accommodation must display in all accommodation page					
Test ID	Test Inputs	Expected output	Actual output	Result (pass/fail)	Comments
TC004	Accommodation name, location, phone number, owner, price, type, image link	Display success message and show in all accommodation page	Success message displayed. Newly added item show in the all accommodation page	pass	Add new accommodation function successfully completed

2.4.3 Reservation of Ticket Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: Airline Tickets Booking					
Test case ID: TC005			Test designed by: Reg. No- IT20046552 Name- Abewardhanage S. R.D		
Test priority (High/Medium/Low):			High		
Test Description: Registered Customer who visit the website they can easily Find Airline Reservation page and book the Airline Ticket Successfully. Customer Should pay their payment using correct payment details through using website.					
Preconditions (if there are any): User must sign up or sign in and move to Ticket Booking home page.					
Test steps:					
Step1: User must visit the web site. Step2: User must Register to the website Step3: User must login to the website Step4: User should Click Tickets button in navigation bar. Step5: User can see Four types of ticket in Ticket Home page and can Click Airline Ticket Book Button. Step6: User can see Timetable details and can search using starting places, destination places and flight numbers. Step7: After getting Idea, User should Fill airline booking form correctly and click book now button. Step8: User must fill the payment form. Step9: User must click the pay now button to confirm the Ticket Booking.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC006	Check User Fill the Airline booking Form Details and payment details with Valid Data.	User can See “Thank you for your Booking. Welcome” message.	User can See “Thank you for your Booking. Welcome” message	Pass	Booking Function Successfully Completed.

Testing Function: Insert Bus Details					
Test case ID: TC007			Test designed by: Reg. No- IT20046552 Name- Abewardhanage S. R.D		
Test priority (High/Medium/Low):			High		
Test Description: Registered admin can log in to admin panel using correct email address and password. Admin can add new bus details to a Bus Timetable and admin can view, update, and delete bus details from database.					
Preconditions (if there are any): Admin must sign up or sign in and move to admin panel.					
Test steps:					
Step1: Admin must Register to the website. Step2: Admin must login to the website. Step3: Admin should Click Tickets button in Admin Panel. Step4: Admin can see Insert Bus Page Button. Click the button and can go to an insert Bus page. Step5: Admin should Fill Insert Bus Form correctly and click Add now button. Step6: Admin can see new added Bus Details in the Table and can search using starting places, destination places. As well as those details updated in Bus Timetable.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC006	Check Admin Fill the Insert bus Form Details with Valid Data.	Admin can See “Bus added Successfully” message.	Admin can See “Bus added Successfully” message	Pass	Add new Bus Function Successfully Completed.

2.4.4 Reservation of Ground Resource Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: Add New Travel Guide					
Test case ID: TC007			Test designed by: Reg. No- IT20186906 Name- Tharuka Gayashan F.		
Test priority (High/Medium/Low):			Medium		
Test Description: Registered admin can log in to admin panel using correct email address and password. The administrator can add a new travel guide to the system. Go to the guide management page and the guide can be added to the system by entering valid information.					
Preconditions (if there are any): Must be an admin and must be Login to the System.					
Test steps:					
Step1: Admin must visit to the web application. Step2: Admin must Register to the website Step3: Admin must login to the website Step4: Admin should click Guide Management button in admin panel. Step5: Admin must enter the correct information in the form. Step6: Admin should click the insert button. Step7: Admin can see the "Guide added successfully" message.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC007	Inputs that belong to validations must be provided the admin	Admin can see "Guide added successfully" message.	Admin can see "Guide added successfully" message.	Pass	Ground **Resource** function Successfully Completed.

Testing Function: Add New Travel Guide					
Test case ID: TC008			Test designed by: Reg. No- IT20186906 Name- Tharuka Gayashan F.		
Test priority (High/Medium/Low):			High		
Test Description: Registered customer can log in to system using correct email address and password. The customer can make a new car reservation. For that customer has to go to the car profile and give the correct details. Then click the book now button.					
Preconditions (if there are any): User must sign up or sign in to the system.					
Test steps:					
Step1: Customer must Register to the website. Step2: Customer must login to the website. Step3: Customer should Click Ground reservation button in navigation bar. Step4: The customer is given 3 options. And the customer must click the car button. Step5: Customer must enter the correct information in the form. Step6: User should click the Book Now button. Step7: Customer can see the "Car reservation successfully" message.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC008	Inputs that belong to validations must be provided the customer	Customer can see "Car reservation successfully" message.	Customer can see "Car reservation successfully" message.	Pass	Car Reservation Function Successfully Completed.

2.4.5 Travel Document Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: Add Travel Document					
Test case ID: TC009			Test designed by: Reg. No- IT20172350 Name- M.P.O.M Gomes.		
Test priority (High/Medium/Low):			Medium		
Test Description: Registered customer can log in to system using correct email address and password. The Customer can add travel Document to the system from Travel document function. Go to the Travel Document Add page and the Travel Document can be added to the system by entering valid information.					
Preconditions (if there are any): Must be a Registered user and must be Login to the System.					
Test steps:					
Step1: Customer must visit to the web application. Step2: Customer must Register to the website Step3: Customer must login to the website Step4: Customer should click Document button in home navigation bar. Step5: Customer must enter the correct information in the form in Travel Document Add page. Step6: Customer should click the Submit button.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC009	Inputs valid details into Travel document Add form.	Customer can see "Travel Document added successfully" message.	Customer can see "Travel Document added successfully" message.	Pass	Travel Document Add function Successfully Completed.

Testing Function: Update User's Travel Document					
Test case ID: TC010			Test designed by: Reg. No- IT20172350 Name- M.P.O.M. Gomes		
Test priority (High/Medium/Low):			Medium		
Test Description: Registered admin can log in to admin panel using correct email address and password. The administrator can view customers added travel document details and update them in the system. Go to the User's Travel Document page and the selected travel Document can be updated in the system by entering valid information and click submit button at last.					
Preconditions (if there are any): Must be an admin and must be Login to the System.					
Test steps:					
Step1: Admin must visit to the web application. Step2: Admin must Register to the website Step3: Admin must login to the website Step4: Admin should click User's Travel Document button in admin panel. Step5: Admin must click updated button from selected travel document. Step6: Admin must enter the updated information in the form. Step7: Admin should click the submit button.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC010	Inputs that updated details into the form	Admin can see "Travel Document update successfully" message.	Admin can see "Travel Document update successfully" message.	Pass	Travel document update Function Successfully Completed.

2.4.6 Tour Package Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: User input No of adults and child calculate the price for the tour package.					
Test case ID: TC011			Test designed by: Reg. No- IT20255510 Name- Y. N Jayasekara		
Test priority (High/Medium/Low):			Medium		
Test Description: User first select Tour package what he/she wants. Then user must fill the reservation details with No of Adults and child. Then user can see the price for the No of adults and child separately and full amount for the tour package. Then user must confirm the tour reservation by fill payment details and proceed it.					
Preconditions (if there are any): User must select the travel package for what he/she wants.					
Test steps:					
User mut go to the web site. User must select the tour in navigation bar. User must select tour package among Tour packages. User must select a date for the tour package. User input No of adults and children User clicks Procced button User must fill the payment form User must click the submit button to confirm the reservation.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC011	User input No of child and adults as 1 and 2.	System will calculate price for the child and adults separately and display the total amount of the child and adults' price as a tour package amount (Total:250000/=)	Total:250000/=	Pass	Output showed as expected.

Testing Function: Admin can view, add, update, delete tour packages					
Test case ID: TC012			Test designed by: Reg. No- IT20255510 Name- Y.N Jayasekara		
Test priority (High/Medium/Low):			Medium		
Test Description: Tour Package Manager/Admin can log in to admin dashboard. Admin can view all tour packages and add new one, edit or delete tour packages from the database.					
Preconditions (if there are any): Must be an admin and must be Login to the System.					
Test steps:					
Admin must go to the admin panel. Admin must select the tour package section in dashboard.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC012	packageID: TP001 packageName: Universal Studios Singapore PackageType: Family & Holiday NoOfDays: 5 MealPlan: Breakfast and Dinner Transportation: Airport Transfers Activities: Ride the roller coasters Location List: 8 Sentosa Gateway, Sentosa Island, Singapore 098269, Singapore PriceForAdult: 100000 PriceForChild: 50000 (Admin must insert new package data to form)	Success message display redirect to current page.	Success message display and redirect to current page.	Pass	Output showed as expected.

2.4.7 Travel Insurance Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: Add new insurance type					
Test case ID: TC013			Test case Designed by: Reg. No- IT20163204 Name- Manimendra N. H.		
Test priority (High/Medium/Low):			Medium		
Test Description: Registered admin can log in to admin panel. The admin can add a new insurance types to the system.					
Preconditions (if there are any): Login account must be an admin account.					
Test steps:					
Step1: Admin login to the admin panel. Step2: Admin go to insurance page by clicking insurance button in admin panel. Step3: Admin go to add new insurance page by clicking add button. Step4: Admin must fill the form. Step5: Admin should click the add button. Step6: Admin can see the successful message. Step7: Admin can see newly added insurance type display in all insurance types page.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC013	Admin input insurance ID, insurance type, price, description and image link	Admin can see success message.	Success message displayed.	Pass	Add new insurance type function successfully completed.

Testing Function: Insurance Reservation					
Test case ID: TC014			Test designed by: Reg. No - IT20163204 Name - Manimendra N. H.		
Test priority (High/Medium/Low):			High		
Test Description: Registered customer can log in to system. The customer can reseve the insurance easily.					
Preconditions (if there are any): User must sign up or sign in to the system.					
Test steps:					
Step1: Customer login to the system. Step2: Customer go to insurance page by clicking insurance button in navigation bar. Step3: Customer can search insurance what he/she want Step4: Customer can select the insurance what he/she want Step5: Customer fill the form. Step6: Customer click the apply button for reserve the insurance. Step7: Customer can see the successfulmessage.					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC014	User input due date and reservation date	Admin can see success message.	Success message displayed.	Pass	Insurance reservation function successfully completed.

2.4.8 Event Management

Group ID: ITP2021_S2_MTR_G05					
Project title: Travel Agency and Tour Planning					
Testing Function: **Add a new event**					
Test case ID: TC015			**Test case Designed by:** **ID: IT20142728** **Name: Ranawaka. T.D**		
Test priority (High/Medium/Low):			Medium		
Test Description: The event manager of the company can login to the system with admin credentials and add new events to the database.					
Preconditions (if there are any): Must be a Registered user and must be Login to the System.					
Test steps:					
Step 1:login to the website as the event manager Step 2:go to add new events page Step 3: add a new event to database					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC015	Event type-entertainment Location - Matara Date-2/09/2021 Price-1200rs	Show newly added events to the event manager as a table	Error message show “cannot submit with empty fields”	Pass	Every field is required in this form

Testing Function: **Delete an added event**					
Test case ID: TC016			Test case Designed by: ID: IT20142728 Name: Ranawaka. T.D		
Test priority (High/Medium/Low):			Medium		
Test Description: The event manager of the company can login to the system with admin credentials and delete the events from database which are not available.					
Preconditions (if there are any): Must be an admin and must be Login to the System.					
Test steps:					
Step 1: login to the website as the event manager Step 2: go to add new events page Step 3: click delete button from each record Step 4: delete event from database					
Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
TC016	Event ID (auto generated by database)	Events which are select to delete will deleted from database using event ID	Show message "Event deleted"	Pass	Event manager can delete any event that he wants to delete

3. Conclusion

It's been almost close to 6 months since our project was started. Finally, we come up with the system that all client requirements are fulfilled. We are really delighted as a group to have finished this assignment successfully. As second year second semester undergraduate students, the best reward we can earn is being able to apply our whole knowledge and talents to impress our client. Having to work on such a high-level project in our second year of academic life is a key element in generating future great professionals. Let's hope we took huge advantage of this great chance. This project was not as simple as we had expected. It was somewhat hard because that the project will be held out during the Covid 19 pandemic situation. We aimed to achieve two main goals through this project. One is to get professional experience and develop our knowledge. The second goal is to provide excellent service to our clients and establish our reputation in IT field.

In this project, the project's main goal was to cover the Travel Agency System with all functions according to client expectation. In this system, we've gone through eight different functions for each eight members. The functions are namely, User management system, Tickets reservation management system, Tour package management system, Travel documentation management system, Ground reservation management system, Accommodation management system, Travel insurance management system, and an Event management system.

Basically, in the User management system contains the details of all customers of the company and also it handles the user admin authentication. Tickets reservation management system handles all the tickets reservations and its payments such as train, bus, air tickets. In Ground reservation management system contains all the ground reservations that customer expected. As same as Accommodation management system can reserve any accommodations such as hotel, motel, etc. In Tour package management system managing all the tour packages of the system with combination of purchasing system. Travel documentation management system managing all the client reports and documentation which are essential for the customer’s travel purpose. Travel insurance management system is responsible for the security and safe of all clients. Moreover, in the Event management system customized and organize all the events and functions according to the customer willing. Besides of this every single subsystem implemented with retrieval, insert, update, and delete operations along with a report generation function and search function.

Based on this experience, we also hope to launch more successful projects in the future. Not only that but also, we got the chance to recognize our capabilities and weaknesses throughout this period when working with as a group.

4. References

[1] Lecture 5 - Final Report.
[2] IEEE Citation Guidelines
[3] https://ng.ant.design/docs/introduce/en