****

**SE3040**

**Application Frameworks (AF)**

**Final Exam**

**Student ID : IT18149654**

**Student Name : Rajapaksha T.N.**

**Category : Tourism**

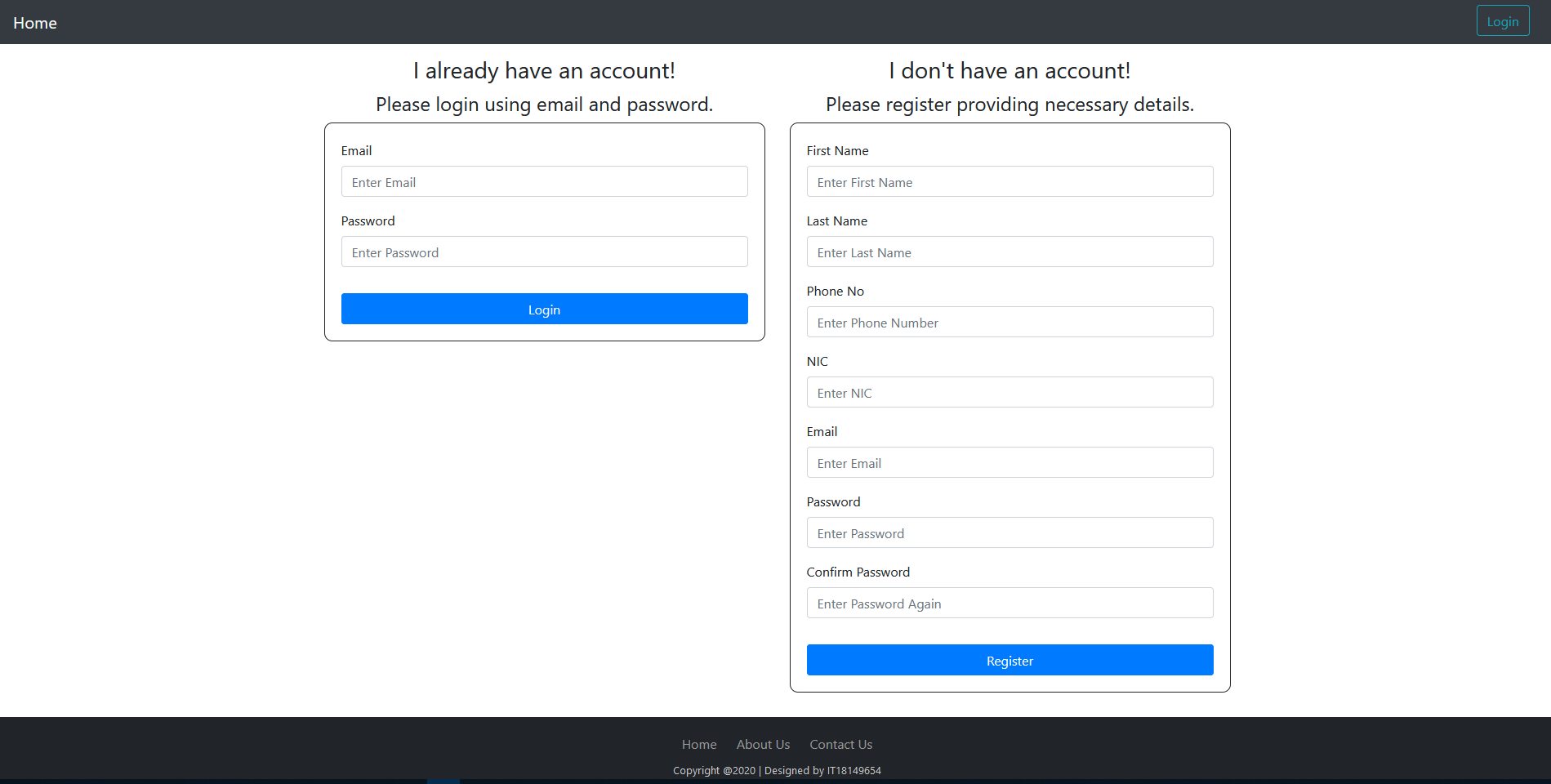
**Scenario**

This is an Online Tourism Web Application Management System.

There are three main actors in this scenario.

1. Administrator
2. Tour Manager
3. Customer

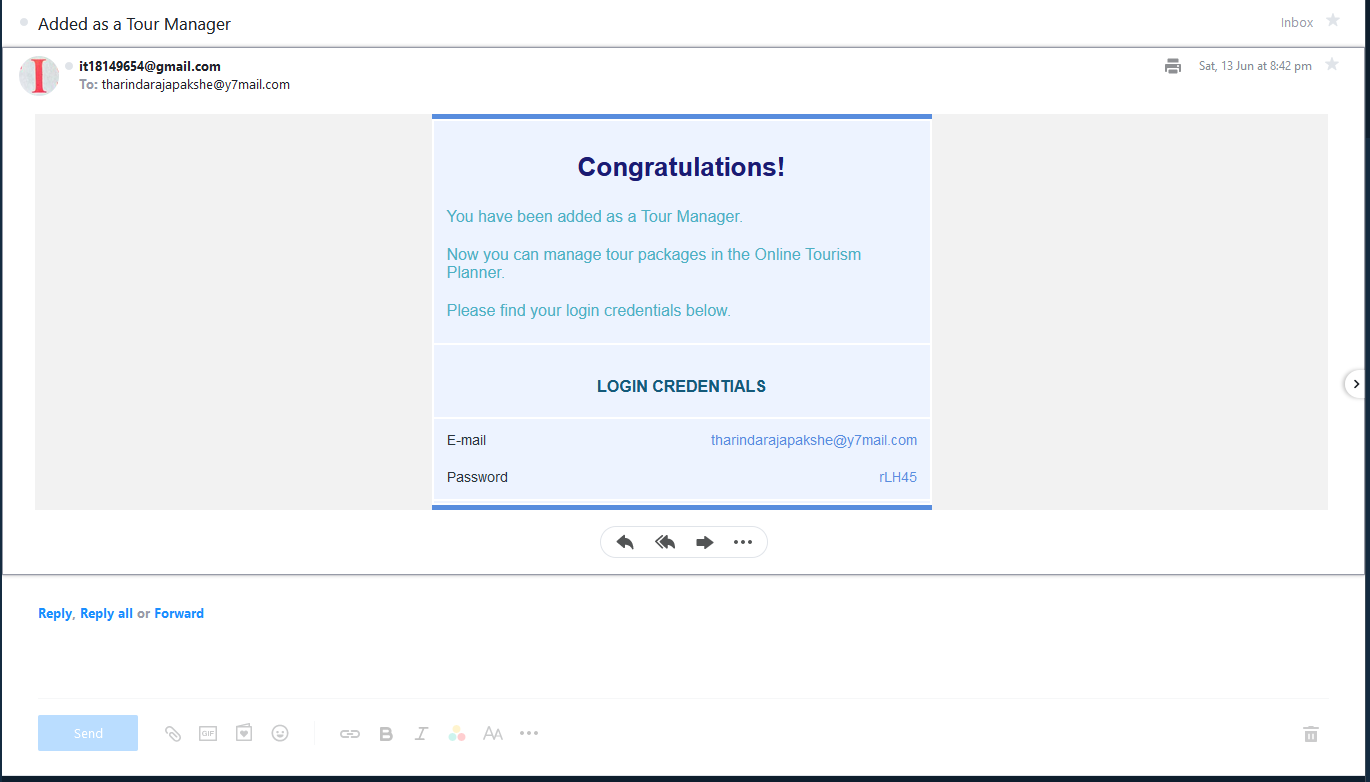
All the users use one login page to login to the system. System can identify the logged in user’s type and provide only the permitted functionality for that user type. The navigation bar will be updated accordingly.



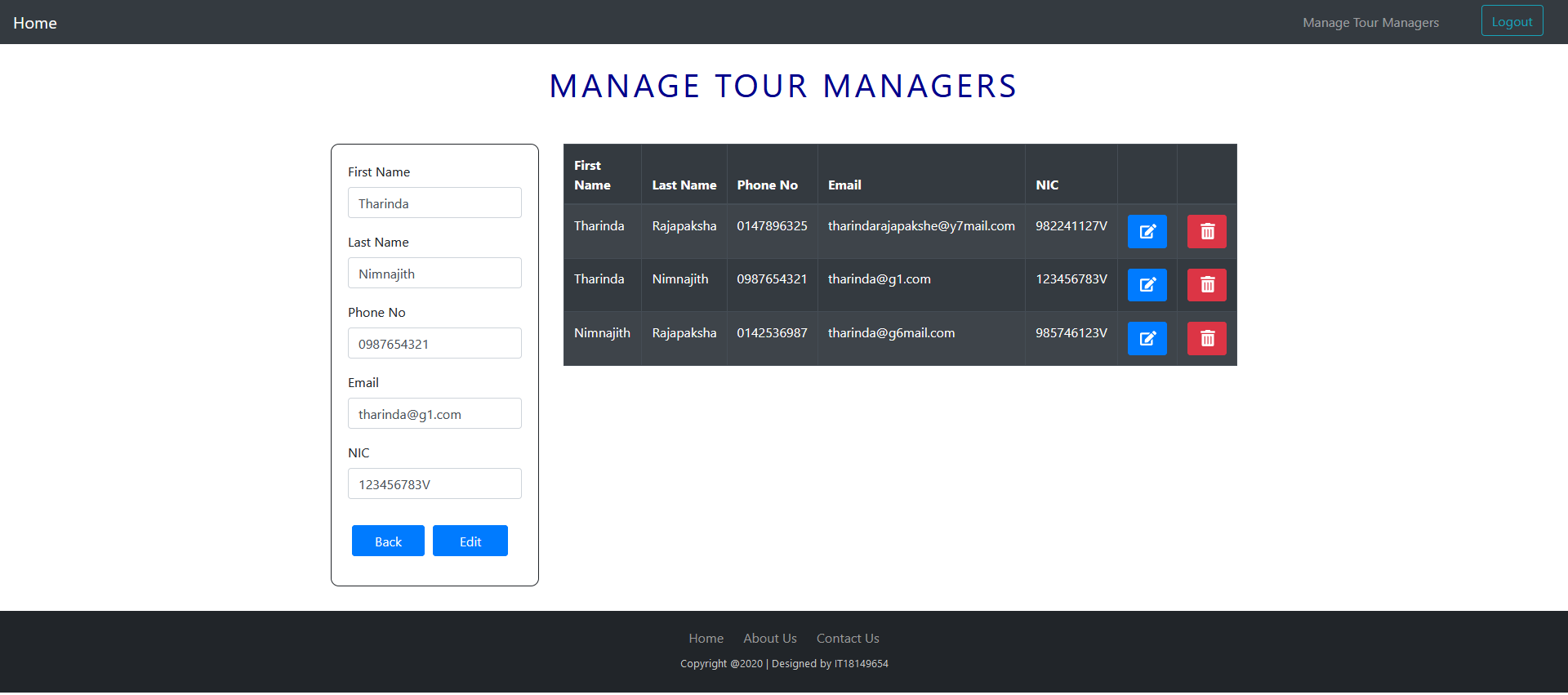
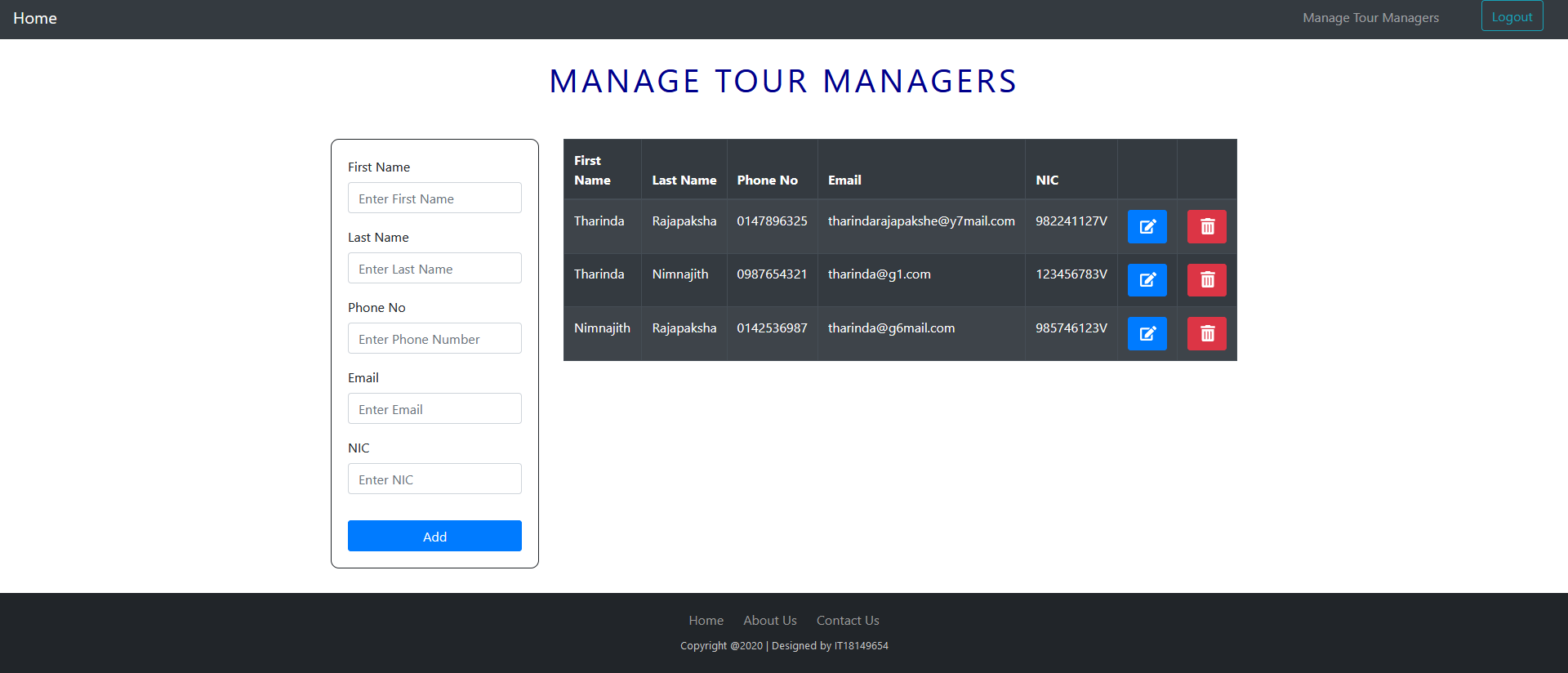
In order **to login to the system as the Administrator**, use the following logging credentials:

**Email = tharindarajapakshe@gmail.com**

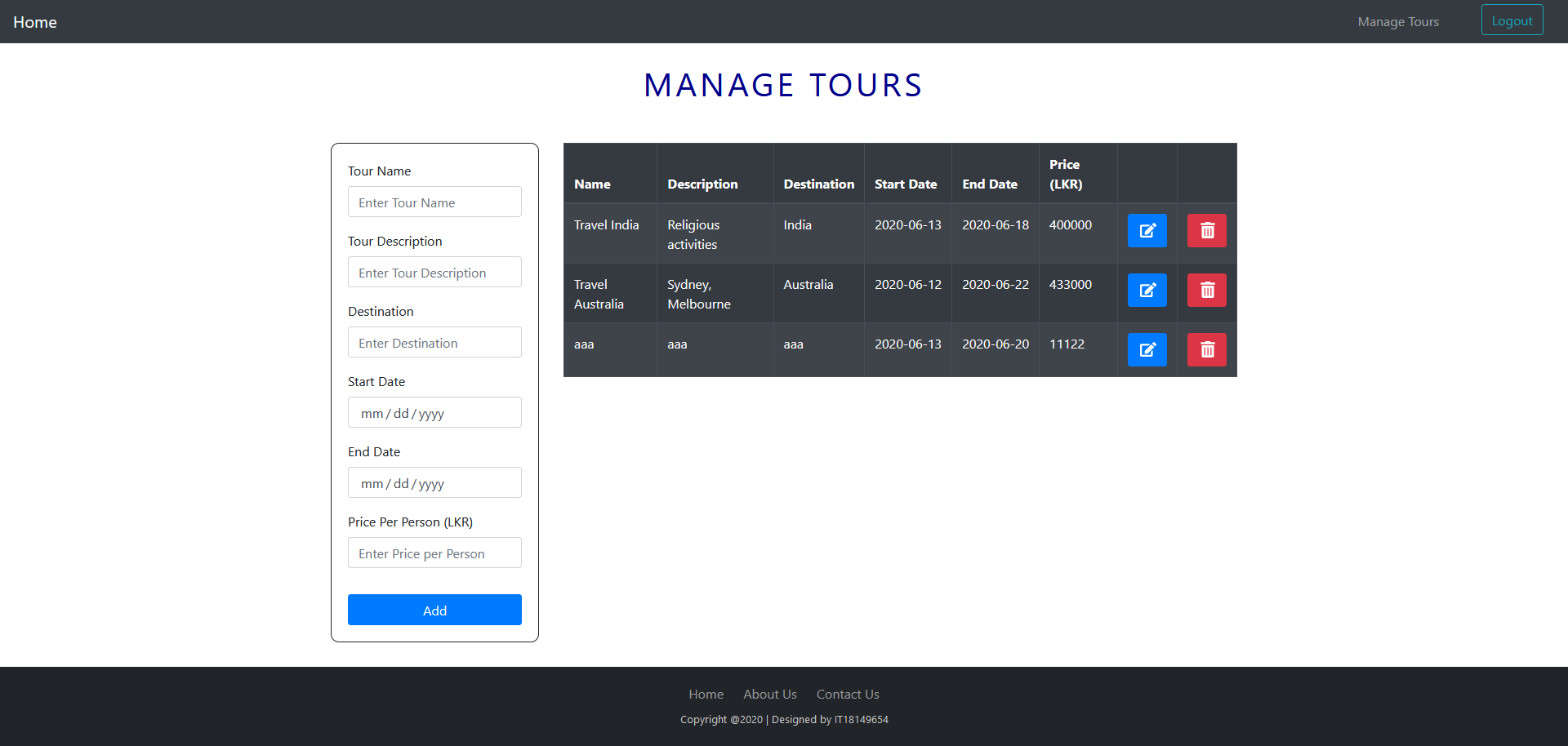
**Password = #It18149654**

The Administrator can login to the system and manage Tour Managers. Administrator can add Tour Managers. When the Tour Manager is added, an email is sent to the newly added Tour Manager with the login credentials (Auto generated password) for the Tour Manager to login.

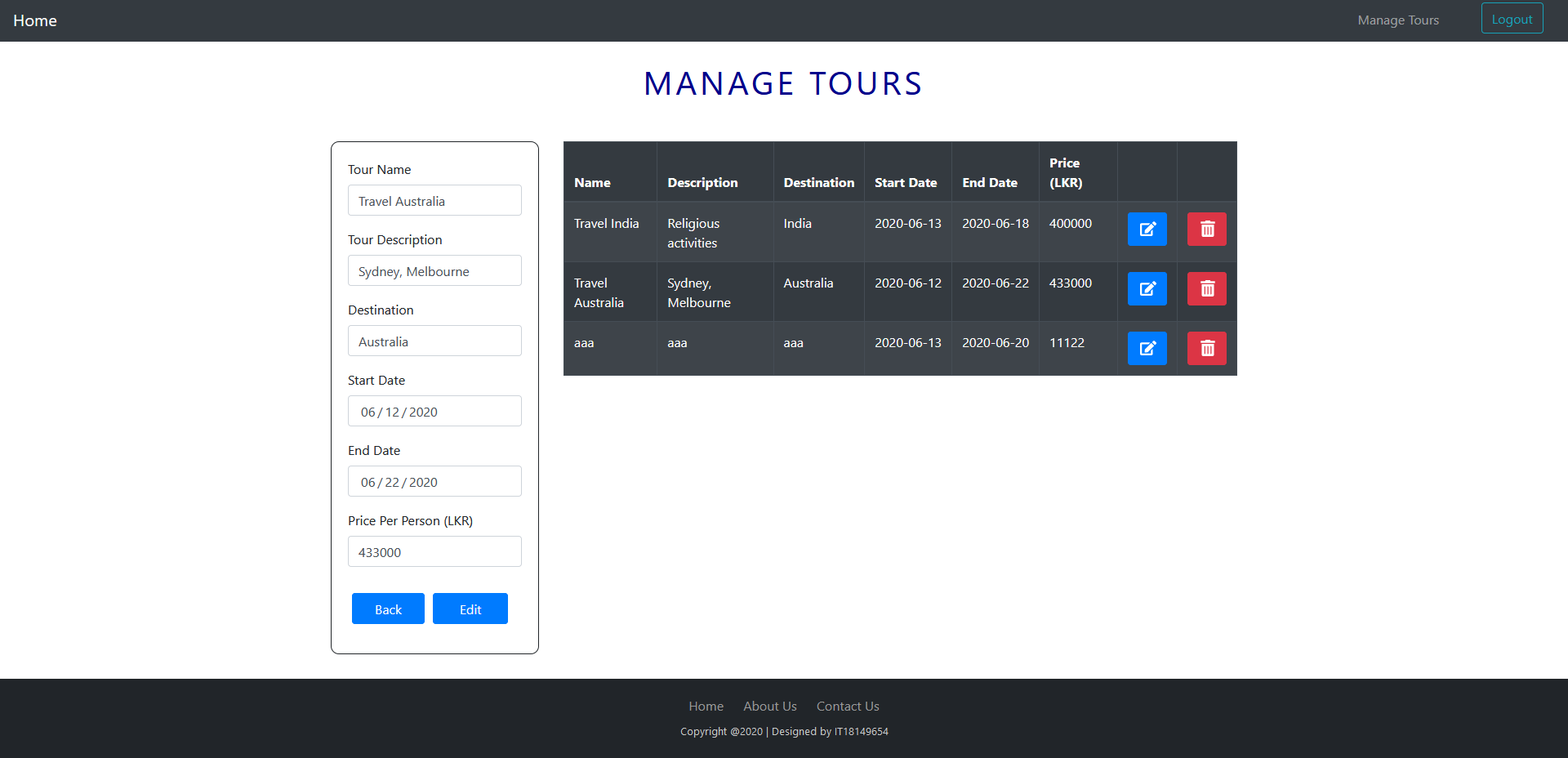
If Administrator details needs to be changed or to add new Administrators to be added, a separate API is provided in the adminRooutes.js in the express backend project. A tool like Postman can be used for this purpose.

****Further the Administrator can perform other CRUD operations on Tour Managers such as editing the details of the existing Tour Managers and removing the Tour Managers.

The Tour Manager can login to the system using the email and the automatically generated random password sent for his / her email once the Administrator adds him / her for the system.

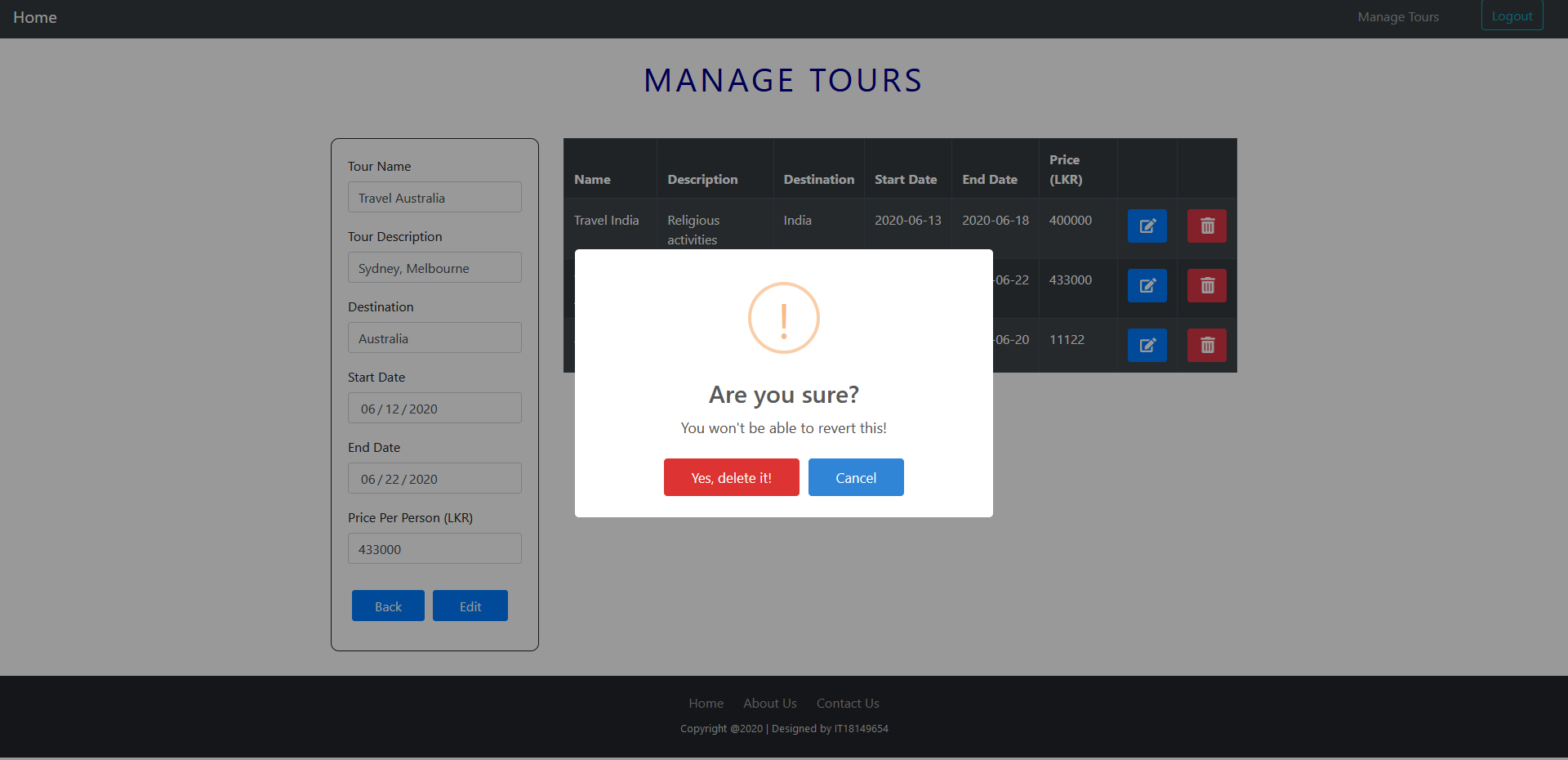


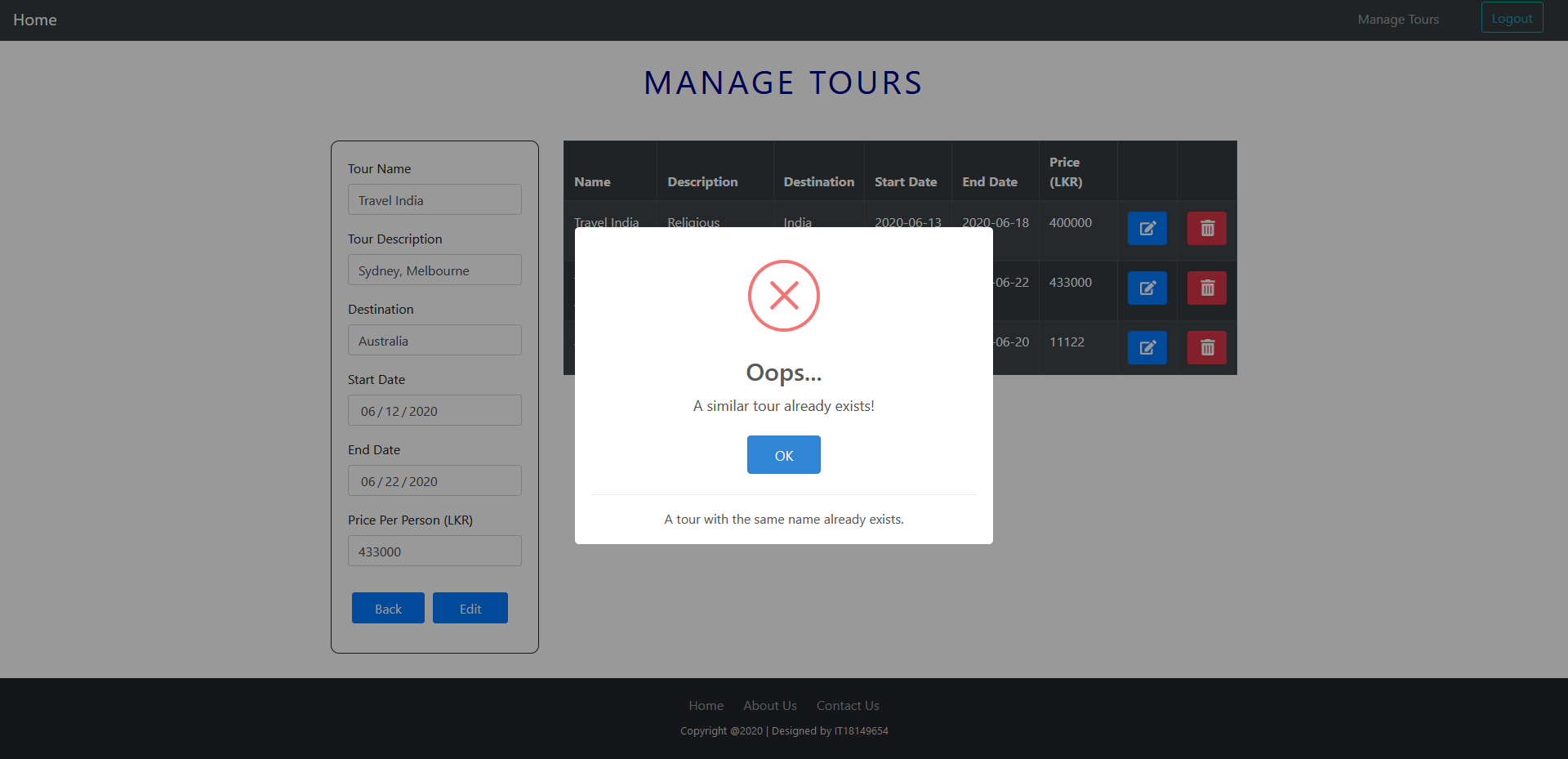
The Tour Manager can manage tour packages once logged in. Tour Manager can add new Tour Packages by providing details such as Tour Name, Tour Description, Destination, Start Date, End Date and Price per Person. Tour Manager can update the existing tour details as well as he / she can delete them if necessary.



A customer can view and book the tour packages. First the customer must be registered to the system and create user account by providing his / personal details. When the customer has an account, he / she can log in to the system.

All the forms are properly validated so that no invalid data will be sent to the backend and stored in the database. For example, data types, input format patterns as well as things such as not being able to enter two users with the same NIC or email are handled. Alerts and specially confirmation messages are provided at necessary situations such as deletion of a record.





**Implementation main functionalities (both frontend and backend)**

1. Login, Logout and Register Functionality

Backend Implementation

login-controller.js

*const* User = require('../models/user-model')  
  
*const* register = *async* (req, res, next) => {  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 } = req.body  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 *const* newUser = *new* User({  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 })  
  
 *try* {  
 *await* newUser.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(201).send({  
 message: 'Customer registered added successfully!'  
 })  
}  
  
*const* login = *async* (req, res, next) => {  
 *const* {  
 email,  
 password  
 } = req.body  
  
 *let* existingUser  
  
 *try* {  
 existingUser = *await* User.findOne({  
 email: email  
 })  
 } *catch* (error) {  
 res.send({  
 message: 'Login failed, please try again later.',  
 login: -1  
 })  
  
 *return* next(error)  
 }  
  
 *if* (!existingUser || existingUser.password !== password) {  
 res.send({  
 message: 'Invalid username or password.',  
 login: 0  
 })  
  
 *return* next(error)  
 }  
  
 res.send({  
 message: 'Logged in!',  
 login: 1,  
 type: existingUser.type,  
 userDetails: existingUser  
 })  
}  
  
*const* updateUser = *async* (req, res, next) => {  
 *let* user  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 } = req.body  
  
 *try* {  
 user = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail && email !== user.email) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC && nic !== user.nic) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 user.firstName = firstName  
 user.lastName = lastName  
 user.teleNo = phoneNo  
 user.email = email  
 user.nic = nic  
 user.password = password  
  
 *try* {  
 *await* user.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'User updated successfully!'  
 })  
}  
  
*const* deleteUser = *async* (req, res, next) => {  
 *let* user  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 user = *await* User.findById(id)  
 *await* user.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'User deleted successfully!'  
 })  
}  
  
*const* getUser = *async* (req, res, next) => {  
 *let* user  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 user = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(user)  
}  
  
*const* getCustomerList = *async* (req, res, next) => {  
 *let* customerList  
  
 *try* {  
 customerList = *await* User.find({  
 type: 'Customer'  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(customerList)  
}  
  
exports.register = register  
exports.login = login  
exports.updateUser = updateUser  
exports.deleteUser = deleteUser  
exports.getUser = getUser  
exports.getCustomerList = getCustomerList

user-model.js

*const* mongoose = require('mongoose')  
*const uniqueValidator* = require('mongoose-unique-validator')  
  
*const* Schema = mongoose.Schema  
  
*const* userSchema = *new* Schema({  
 firstName: {  
 type: String,  
 trim: *true* },  
 lastName: {  
 type: String,  
 trim: *true* },  
 phoneNo: {  
 type: String,  
 trim: *true* },  
 email: {  
 type: String,  
 required: *true*,  
 unique: *true*,  
 trim: *true* },  
 nic: {  
 type: String,  
 unique: *true*,  
 trim: *true* },  
 password: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 type: {  
 type: String,  
 *default*: 'Customer'  
 }  
}, {  
 timestamps: *true*,  
 collection: 'Users'  
})  
  
userSchema.plugin(*uniqueValidator*)  
  
module.exports = mongoose.model('Users', userSchema)

login-routes.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* LoginController = require('../controllers/login-controller')  
  
router.post('/register', LoginController.register);  
router.post('/login', LoginController.login);  
router.put('/user/:id', LoginController.updateUser)  
router.delete('/user/:id', LoginController.deleteUser)  
router.get('/user/:id', LoginController.getUser)  
router.get('/user', LoginController.getCustomerList)  
  
module.exports = router

Frontend Implementation

login-register-component.jsx

*import* React, {Component} *from* 'react'  
*import* 'bootstrap/dist/css/bootstrap.min.css'  
*import* Swal *from* 'sweetalert2'  
*import* axios *from* 'axios'  
*import* {proxy} *from* '../../conf'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Row *from* 'react-bootstrap/Row'  
*import* LoginComponent *from* '../../components/login-component/login-component'  
*import* RegisterComponent *from* '../../components/register-component/register-component'  
*import* HomeComponent *from* '../home-component/home-component'  
*import* NavigationBarComponent *from* '../../components/navigation-bar-component/navigation-bar-component'  
*import* './login-register-component-styles.scss'  
  
*class* LoginRegisterComponent *extends* Component {  
 constructor(props) {  
 *super*(props)  
 *this*.state = {  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: '',  
 password: '',  
 confirmPassword: '',  
 loginEmail: '',  
 loginPassword: '',  
 loggedIn: *false*,  
 userType: ''  
 }  
 }  
  
 *onChangeFirstName* = event => {  
 *this*.setState({  
 firstName: event.target.value  
 })  
 }  
  
 *onChangeLastName* = event => {  
 *this*.setState({  
 lastName: event.target.value  
 })  
 }  
  
 *onChangePhoneNo* = event => {  
 *this*.setState({  
 phoneNo: event.target.value  
 })  
 }  
  
 *onChangeEmail* = event => {  
 *this*.setState({  
 email: event.target.value  
 })  
 }  
  
 *onChangeNIC* = event => {  
 *this*.setState({  
 nic: event.target.value  
 })  
 }  
  
 *onChangePassword* = event => {  
 *this*.setState({  
 password: event.target.value  
 })  
 }  
  
 *onChangeConfirmPassword* = event => {  
 *this*.setState({  
 confirmPassword: event.target.value  
 })  
 }  
  
 *onChangeLoginEmail* = event => {  
 *this*.setState({  
 loginEmail: event.target.value  
 })  
 }  
  
 *onChangeLoginPassword* = event => {  
 *this*.setState({  
 loginPassword: event.target.value  
 })  
 }  
  
 *onSubmitLogin* = event => {  
 event.preventDefault()  
 *const* user = {  
 email: *this*.state.loginEmail,  
 password: *this*.state.loginPassword  
 }  
 axios.post(`${proxy}login/login`, user)  
 .then(res => {  
 *this*.setState({  
 loginEmail: '',  
 loginPassword: ''  
 })  
 *if* (res.data.login === 1) {  
 *this*.setState({  
 loggedIn: *true*,  
 userType: res.data.type  
 })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'Logged in Successfully!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 } *else if* (res.data.login === 0) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *onSubmitRegister* = event => {  
 event.preventDefault()  
 *if* (*this*.state.password !== *this*.state.confirmPassword) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'Passwords must match!',  
 footer: 'Please enter the same password for both Password & Confirm Password fields.'  
 }).then(() => {  
 })  
 } *else* {  
 *const* user = {  
 firstName: *this*.state.firstName,  
 lastName: *this*.state.lastName,  
 phoneNo: *this*.state.phoneNo,  
 email: *this*.state.email,  
 nic: *this*.state.nic,  
 password: *this*.state.password  
 }  
 axios.post(`${proxy}login/register`, user)  
 .then(res => {  
 *if* (res.data.exists === *true*) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'A similar user already exists!',  
 footer: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 *this*.setState({  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: '',  
 password: '',  
 confirmPassword: ''  
 })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'Registration Successful!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
 }  
  
 render() {  
 *if* (*this*.state.loggedIn) {  
 *return* (  
 <HomeComponent *loggedIn*={*this*.state.loggedIn}  
 *userType*={*this*.state.userType}/>  
 )  
 }  
  
 *return* (  
 <div *className*='container'>  
 <NavigationBarComponent *loggedIn*={*this*.state.loggedIn}  
 *userType*={*this*.state.userType}/>  
 <div *style*={{marginTop: '70px'}}>  
 <Row>  
 <Col *sm*='6'>  
 <h3 *align*={'center'}  
 *style*={{  
 marginBottom: '10px'  
 }}  
 >  
 I already have an account!  
 </h3>  
 <h4 *align*={'center'}  
 *style*={{  
 marginBottom: '10px'  
 }}  
 >  
 Please login using email and password.  
 </h4>  
 <LoginComponent *onChangeLoginEmail*={*this*.*onChangeLoginEmail*}  
 *onChangeLoginPassword*={*this*.*onChangeLoginPassword*}  
 *onSubmitLogin*={*this*.*onSubmitLogin*}  
 *loginEmail*={*this*.state.loginEmail}  
 *loginPassword*={*this*.state.loginPassword}/>  
 </Col>  
 <Col *sm*='6'>  
 <h3 *align*={'center'}  
 *style*={{  
 marginBottom: '10px'  
 }}  
 >  
 I don't have an account!  
 </h3>  
 <h4 *align*={'center'}  
 *style*={{  
 marginBottom: '10px'  
 }}  
 >  
 Please register providing necessary details.  
 </h4>  
 <RegisterComponent *onChangeFirstName*={*this*.*onChangeFirstName*}  
 *onChangeLastName*={*this*.*onChangeLastName*}  
 *onChangePhoneNo*={*this*.*onChangePhoneNo*}  
 *onChangeEmail*={*this*.*onChangeEmail*}  
 *onChangePassword*={*this*.*onChangePassword*}  
 *onChangeConfirmPassword*={*this*.*onChangeConfirmPassword*}  
 *onChangeNIC*={*this*.*onChangeNIC*}  
 *onSubmitRegister*={*this*.*onSubmitRegister*}  
 *firstName*={*this*.state.firstName}  
 *lastName*={*this*.state.lastName}  
 *phoneNo*={*this*.state.phoneNo}  
 *email*={*this*.state.email}  
 *nic*={*this*.state.nic}  
 *password*={*this*.state.password}  
 *confirmPassword*={*this*.state.confirmPassword}/>  
 </Col>  
 </Row>  
 </div>  
 </div>  
 )  
 }  
}  
  
*export default* LoginRegisterComponent

login-component.jsx

*import* React, {Component} *from* 'react'  
*import* Form *from* 'react-bootstrap/Form'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Button *from* 'react-bootstrap/Button'  
*import* './login-component-styles.scss'  
  
*class* LoginComponent *extends* Component {  
 render() {  
 *const* {  
 onChangeLoginEmail,  
 onChangeLoginPassword,  
 onSubmitLogin,  
 loginEmail,  
 loginPassword  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Form  
 *onSubmit*={onSubmitLogin}  
 *style*={{  
 border: 'solid 1px',  
 padding: '20px',  
 borderRadius: '10px'  
 }}  
 >  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridLoginEmail'>  
 <Form.Label>Email</Form.Label>  
 <Form.Control *placeholder*='Enter Email'  
 *type*='text'  
 *onChange*={onChangeLoginEmail}  
 *value*={loginEmail}  
 *pattern*='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'  
 *title*='Please enter a valid email.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridLoginPassword'>  
 <Form.Label>Password</Form.Label>  
 <Form.Control *placeholder*='Enter Password'  
 *type*='password'  
 *onChange*={onChangeLoginPassword}  
 *value*={loginPassword}  
 *minlength*={5}  
 *title*='Please enter a valid password.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*={'btn btn-block btn-primary mt-3'}>  
 Login  
 </Button>  
 </Form>  
 </div>  
 )  
 }  
}  
  
*export default* LoginComponent

register-component.jsx

*import* React, {Component} from 'react'  
*import* Form *from* 'react-bootstrap/Form'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Button *from* 'react-bootstrap/Button'  
*import* './register-component-styles.scss'  
  
*class* RegisterComponent *extends* Component {  
 render() {  
 *const* {  
 onChangeFirstName,  
 onChangeLastName,  
 onChangePhoneNo,  
 onChangeEmail,  
 onChangePassword,  
 onChangeConfirmPassword,  
 onChangeNIC,  
 onSubmitRegister,  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password,  
 confirmPassword  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Form  
 *name*={'form'}  
 *onSubmit*={onSubmitRegister}  
 *style*={{  
 border: 'solid 1px',  
 padding: '20px',  
 borderRadius: '10px'  
 }}  
 >  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridFirstName'>  
 <Form.Label>First Name</Form.Label>  
 <Form.Control *placeholder*='Enter First Name'  
 *type*='text'  
 *onChange*={onChangeFirstName}  
 *value*={firstName}  
 *pattern*='[A-Za-z]{2,32}'  
 *title*='Please enter a valid first name.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridLastName'>  
 <Form.Label>Last Name</Form.Label>  
 <Form.Control *placeholder*='Enter Last Name'  
 *type*='text'  
 *onChange*={onChangeLastName}  
 *value*={lastName}  
 *pattern*='[A-Za-z]{2,32}'  
 *title*='Please enter a valid last name.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridPhoneNo'>  
 <Form.Label>Phone No</Form.Label>  
 <Form.Control *placeholder*='Enter Phone Number'  
 *type*='text'  
 *onChange*={onChangePhoneNo}  
 *value*={phoneNo}  
 *pattern*='0[0-9]{9}'  
 *title*='Please enter a valid phone number.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridNIC'>  
 <Form.Label>NIC</Form.Label>  
 <Form.Control *placeholder*='Enter NIC'  
 *type*='text'  
 *onChange*={onChangeNIC}  
 *value*={nic}  
 *pattern*='[0-9]{9}V'  
 *title*='Please enter a valid NIC.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridEmail'>  
 <Form.Label>Email</Form.Label>  
 <Form.Control *placeholder*='Enter Email'  
 *type*='email'  
 *onChange*={onChangeEmail}  
 *value*={email}  
 *pattern*='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'  
 *title*='Please enter a valid email.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridPassword'>  
 <Form.Label>Password</Form.Label>  
 <Form.Control *placeholder*='Enter Password'  
 *type*='password'  
 *onChange*={onChangePassword}  
 *value*={password}  
 *pattern*='(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{5,}'  
 *title*='Password must contain at least one number, one uppercase,  
 lowercase letter and at least 5 or more characters.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridConfirmPassword'>  
 <Form.Label>Confirm Password</Form.Label>  
 <Form.Control *placeholder*='Enter Password Again'  
 *type*='password'  
 *onChange*={onChangeConfirmPassword}  
 *value*={confirmPassword}  
 *pattern*='(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{5,}'  
 *title*='Password must contain at least one number, one uppercase,  
 lowercase letter and at least 5 or more characters.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*='btn btn-block btn-primary mt-3'>  
 Register  
 </Button>  
 </Form>  
 </div>  
 )  
 }  
}  
  
*export default* RegisterComponent

navigation-bar-component.jsx

*import* React, {Component} *from* 'react'  
*import* {*Link*} *from* 'react-router-dom'  
*import* Button *from* 'react-bootstrap/Button'  
*import* Navbar *from* 'react-bootstrap/Navbar'  
*import* Nav *from* 'react-bootstrap/Nav'  
*import* './navigation-bar-component-styles.scss'  
  
*class* NavigationBarComponent *extends* Component {  
 constructor(props) {  
 *super*(props)  
 *this*.state = {  
 loggedIn: *false*,  
 userType: ''  
 }  
 }  
  
 render() {  
 *const* {  
 loggedIn,  
 userType  
 } = *this*.props  
  
 *this*.state.loggedIn = loggedIn  
 *this*.state.userType = userType  
  
 *return* (  
 <Navbar *bg*='dark'  
 *variant*='dark'  
 *fixed*='top'  
 *expand*='md'  
 *style*={{  
 width: '100%'  
 }}  
 *collapseOnSelect* >  
 <Nav>  
 <div>  
 <Link *to*='/'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Navbar.Brand *href*='#home'>Home</Navbar.Brand>  
 </Link>  
 </div>  
 <div  
 *style*={{  
 position: 'fixed',  
 right: '150px'  
 }}  
 >  
 <Nav *className*='mr-auto'>  
 {  
 loggedIn && userType === 'Administrator' ? (  
 <Link *to*='/managers'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Nav.Link *href*='#managers'>Manage Tour Managers</Nav.Link>  
 </Link>  
 ) :  
 *null* }  
 {  
 loggedIn && userType === 'Tour Manager' ? (  
 <Link *to*='/tours'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Nav.Link *href*='#tours'>Manage Tours</Nav.Link>  
 </Link>  
 ) :  
 *null* }  
 {  
 loggedIn && userType === 'Customer' ? (  
 <Link *to*='/packages'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Nav.Link *href*='#book\_now'>  
 <Button *variant*={'danger'}  
 *style*={{  
 marginTop: '-8px'  
 }}  
 >  
 Book Now!  
 </Button>  
 </Nav.Link>  
 </Link>  
 ) :  
 *null* }  
 </Nav>  
 </div>  
 <div  
 *style*={{  
 position: 'fixed',  
 right: '30px'  
 }}  
 >  
 {  
 loggedIn ? (  
 <Link *to*='/'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Button *variant*='outline-info'>Logout</Button>  
 </Link>  
 ) :  
 <Link *to*='/login'  
 *style*={{  
 textDecoration: 'none'  
 }}  
 >  
 <Button *variant*='outline-info'>Login</Button>  
 </Link>  
 }  
 </div>  
 </Nav>  
 </Navbar>  
 )  
 }  
}  
  
*export default* NavigationBarComponent

1. Manage Tour Managers Functionality

Backend Implementation

tour-manager-controller.js

*const* nodemailer = require('nodemailer')  
  
*const* User = require('../models/user-model')  
  
require('dotenv').*config*()  
  
*const* addTourManager = *async* (req, res, next) => {  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic  
 } = req.body  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 *let* generatedPassword = generatePassword()  
  
 *const* newTourManager = *new* User({  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password: generatedPassword,  
 type: 'Tour Manager'  
 })  
  
 *try* {  
 *await* newTourManager.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *await* sendEmail(email, generatedPassword)  
  
 res.status(201).send({  
 message: 'New tour manager added successfully!'  
 })  
}  
  
*const* updateTourManager = *async* (req, res, next) => {  
 *let* tourManager  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic  
 } = req.body  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail && email !== tourManager.email) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC && nic !== tourManager.nic) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 tourManager.firstName = firstName  
 tourManager.lastName = lastName  
 tourManager.teleNo = phoneNo  
 tourManager.email = email  
 tourManager.nic = nic  
  
 *try* {  
 *await* tourManager.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour manager updated successfully!'  
 })  
}  
  
*const* deleteTourManager = *async* (req, res, next) => {  
 *let* tourManager  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 *await* tourManager.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour manager deleted successfully!'  
 })  
}  
  
*const* getTourManager = *async* (req, res, next) => {  
 *let* tourManager  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourManager)  
}  
  
*const* getTourManagerList = *async* (req, res, next) => {  
 *let* tourManagerList  
  
 *try* {  
 tourManagerList = *await* User.find({  
 type: 'Tour Manager'  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourManagerList)  
}  
  
*const* getAdminEmail = *async* () => {  
 *let* admin  
  
 *try* {  
 admin = *await* User.findOne({  
 type: 'Administrator'  
 })  
 } *catch* (error) {  
 *return* error  
 }  
  
 *return* admin.email;  
}  
  
*let* transporter = nodemailer.*createTransport*({  
 service: 'gmail',  
 auth: {  
 user: 'it18149654@gmail.com',  
 pass: process.env.PASSWORD  
 }  
})  
  
*const* sendEmail = *async* (email, password) => {  
 *let* adminEmail = getAdminEmail()  
  
 *let* info = {  
 from: adminEmail,  
 to: email,  
 subject: 'Added as a Tour Manager',  
 text:  
 `Congratulations!  
 You have been assigned as a Tour Manager.  
 Now you can manage tours related operations as a tour manager in the Online Tourism Planner.  
 Please find your login credentials below.  
 LOGIN CREDENTIALS  
 Email: ${email}  
 Password: ${password}  
 Thank you!  
 This is an auto-generated email.  
 If this has been sent by mistake, please delete this without sharing this.  
 All rights reserved.`,  
 html:  
 `<!--suppress HtmlDeprecatedAttribute -->  
 <div style="margin: 0; padding: 0; background-color: #f2f2f2; font-family: arial, serif;">  
 <table style="margin: 0 auto; background: white; max-width: 500px; padding-bottom: 0; border-top: 5px solid #588dde; border-bottom: 5px solid #588dde; width: 100%;">  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 100%;">  
 <tr>  
 <td style="padding: 10px;">  
 <h1 style="text-align: center; color: #1a1a72;">Congratulations!</h1>  
 <h2 style="margin-top:25px; margin-bottom: 0; color: #4db0c4; font-weight: 400; font-size: medium;">You have been added as a Tour Manager.</h2>  
 <h2 style="margin-top:20px; margin-bottom: 0; color: #4db0c4; font-weight: 400; font-size: medium;">Now you can manage tour packages in the Online Tourism Planner.</h2>  
 <h2 style="margin-top:20px; margin-bottom: 10px; color: #4db0c4; font-weight: 400; font-size: medium;">Please find your login credentials below.</h2>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 100%;">  
 <tr>  
 <td style="padding: 10px;">  
 <h4 style="margin-top:20px; margin-bottom: 8px; color: #145a7a; font-weight: 400; text-align: center; font-size: 16px;"><b>LOGIN CREDENTIALS</b></h4>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 50%;">  
 <tr>  
 <td align="left" valign="top" style="padding: 10px;">  
 <h6 style="font-size: 14px; margin-top: 0; margin-bottom: 0; color: #29353c; font-weight: 400;">E-mail</h6>  
 <h6 style="font-size: 14px; margin-top: 20px; margin-bottom: 0; color: #29353c; font-weight: 400;">Password</h6>  
 </td>  
 </tr>  
 </table>  
 <table align="left" style="width: 50%;">  
 <tr>  
 <td align="right" valign="top" style="padding: 10px;">  
 <h6 style="font-size: 14px; margin-top: 0; margin-bottom: 0; color: #588dde; font-weight: 400;">${email}</h6>  
 <h6 style="font-size: 14px; margin-top: 20px; margin-bottom: 0; color: #588dde; font-weight: 400;">${password}</h6>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>`  
 }  
  
 *// noinspection JSCheckFunctionSignatures* transporter.sendMail(info, (error, data) => {  
 *if* (error) {  
 *console*.log(error)  
 *console*.log('Email sending failed! Please try again.')  
 } *else* {  
 *console*.log(data)  
 *console*.log('An email is sent successfully to ' + email + '.')  
 }  
 })  
}  
  
*function* generatePassword() {  
 *let* length = 5  
 *let* randomPassword = ''  
 *let* characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'  
 *let* charactersLength = characters.length  
  
 *for* (*let* i = 0; i < length; i++)  
 randomPassword += characters.charAt(Math.floor(Math.random() \* charactersLength))  
  
 *return* randomPassword  
}  
  
exports.addTourManager = addTourManager  
exports.updateTourManager = updateTourManager  
exports.deleteTourManager = deleteTourManager  
exports.getTourManager = getTourManager  
exports.getTourManagerList = getTourManagerList

tour-manager-controller.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* TourManagerController = require('../controllers/tour-manager-controller')  
  
router.post('/manager', TourManagerController.addTourManager)  
router.put('/manager/:id', TourManagerController.updateTourManager)  
router.delete('/manager/:id', TourManagerController.deleteTourManager)  
router.get('/manager/:id', TourManagerController.getTourManager)  
router.get('/manager', TourManagerController.getTourManagerList)  
  
module.exports = router

Frontend Implementation

manage-tour-manager-component.jsx

*import* React, {Component} *from* 'react'  
*import* 'bootstrap/dist/css/bootstrap.min.css'  
*import* Swal *from* 'sweetalert2'  
*import* axios *from* 'axios'  
*import* {proxy} *from* '../../conf'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Row *from* 'react-bootstrap/Row'  
*import* AddTourManagerComponent *from* '../../components/tour-manager-add-component/tour-manager-add-component'  
*import* ListTourManagerComponent *from* '../../components/tour-manager-list-component/tour-manager-list-component'  
*import* NavigationBarComponent *from* '../../components/navigation-bar-component/navigation-bar-component'  
*import* './manage-tour-manager-component-styles.scss'  
  
*class* ManageTourManagerComponent *extends* Component {  
 constructor(props) {  
 *super*(props)  
 *this*.state = {  
 users: [],  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: '',  
 editingUserId: '',  
 editingUser: *null*,  
 editUser: *false*,  
 loggedIn: *true*,  
 userType: 'Administrator'  
 }  
 }  
  
 componentDidMount() {  
 *this*.*getUsers*()  
 }  
  
 *getUsers* = () => {  
 axios.get(`${proxy}manager/manager`)  
 .then(res => {  
 *this*.setState({  
 users: res.data  
 })  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *onChangeFirstName* = event => {  
 *this*.setState({  
 firstName: event.target.value  
 })  
 }  
  
 *onChangeLastName* = event => {  
 *this*.setState({  
 lastName: event.target.value  
 })  
 }  
  
 *onChangePhoneNo* = event => {  
 *this*.setState({  
 phoneNo: event.target.value  
 })  
 }  
  
 *onChangeEmail* = event => {  
 *this*.setState({  
 email: event.target.value  
 })  
 }  
  
 *onChangeNIC* = event => {  
 *this*.setState({  
 nic: event.target.value  
 })  
 }  
  
 *onSubmitAdd* = event => {  
 event.preventDefault()  
 *const* user = {  
 firstName: *this*.state.firstName,  
 lastName: *this*.state.lastName,  
 phoneNo: *this*.state.phoneNo,  
 email: *this*.state.email,  
 nic: *this*.state.nic  
 }  
 axios.post(`${proxy}manager/manager`, user)  
 .then(res => {  
 *if* (res.data.exists === *true*) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'A similar user already exists!',  
 footer: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 *this*.*getUsers*()  
 *this*.setState({  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: ''  
 })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'A new tour manager added successfully!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *deleteUser* = userId => {  
 Swal.*fire*({  
 title: 'Are you sure?',  
 text: "You won't be able to revert this!",  
 icon: 'warning',  
 showCancelButton: *true*,  
 cancelButtonColor: '#3085d6',  
 confirmButtonColor: '#d33',  
 confirmButtonText: 'Yes, delete it!'  
 }).then((result) => {  
 *if* (result.value) {  
 axios.delete(`${proxy}manager/manager/${userId}`)  
 .then(() => {  
 *this*.*getUsers*()  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 Swal.*fire*(  
 'Deleted!',  
 'Tour manager has been deleted.',  
 'success'  
 )  
 }  
 })  
 }  
  
 *onBack* = () => {  
 *this*.setState({  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: '',  
 editingUserId: '',  
 editingUser: *null*,  
 editUser: *false* })  
 }  
  
 *onSubmitEdit* = userId => {  
 axios.get(`${proxy}manager/manager/${userId}`)  
 .then(res => {  
 *this*.setState({  
 editingUserId: userId,  
 editingUser: res.data  
 })  
 *this*.setState({  
 editUser: *true*,  
 firstName: *this*.state.editingUser.firstName,  
 lastName: *this*.state.editingUser.lastName,  
 phoneNo: *this*.state.editingUser.phoneNo,  
 email: *this*.state.editingUser.email,  
 nic: *this*.state.editingUser.nic  
 })  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *onSubmitUpdate* = event => {  
 event.preventDefault()  
 *const* user = {  
 firstName: *this*.state.firstName,  
 lastName: *this*.state.lastName,  
 phoneNo: *this*.state.phoneNo,  
 email: *this*.state.email,  
 nic: *this*.state.nic  
 }  
 axios.put(`${proxy}manager/manager/${*this*.state.editingUserId}`, user)  
 .then(res => {  
 *if* (res.data.exists === *true*) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'A similar user already exists!',  
 footer: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 *this*.*getUsers*()  
 *this*.setState({  
 firstName: '',  
 lastName: '',  
 phoneNo: '',  
 email: '',  
 nic: '',  
 editingUserId: '',  
 editingUser: *null*,  
 editUser: *false* })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'Tour manager updated successfully!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 render() {  
 *return* (  
 <div *className*='container'>  
 <NavigationBarComponent *loggedIn*={*this*.state.loggedIn}  
 *userType*={*this*.state.userType}/>  
 <h1 *style*={{  
 textAlign: 'center',  
 marginTop: '80px',  
 marginBottom: '50px',  
 textTransform: 'uppercase',  
 letterSpacing: '4px',  
 color: 'darkblue'  
 }}>  
 Manage Tour Managers  
 </h1>  
 <Row>  
 <Col *sm*='3'>  
 <AddTourManagerComponent *onBack*={*this*.*onBack*}  
 *onChangeFirstName*={*this*.*onChangeFirstName*}  
 *onChangeLastName*={*this*.*onChangeLastName*}  
 *onChangePhoneNo*={*this*.*onChangePhoneNo*}  
 *onChangeEmail*={*this*.*onChangeEmail*}  
 *onChangeNIC*={*this*.*onChangeNIC*}  
 *onSubmitAdd*={*this*.*onSubmitAdd*}  
 *onSubmitUpdate*={*this*.*onSubmitUpdate*}  
 *editUser*={*this*.state.editUser}  
 *firstName*={*this*.state.firstName}  
 *lastName*={*this*.state.lastName}  
 *phoneNo*={*this*.state.phoneNo}  
 *email*={*this*.state.email}  
 *nic*={*this*.state.nic}/>  
 </Col>  
 <Col *sm*='9'>  
 <ListTourManagerComponent *users*={*this*.state.users}  
 *onSubmitEdit*={*this*.*onSubmitEdit*}  
 *deleteUser*={*this*.*deleteUser*}/>  
 </Col>  
 </Row>  
 </div>  
 )  
 }  
}  
  
*export default* ManageTourManagerComponent

tour-manager-add-component.jsx

*import* React, {Component} *from* 'react'  
*import* Form *from* 'react-bootstrap/Form'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Button *from* 'react-bootstrap/Button'  
*import* './tour-manager-add-component-styles.scss'  
  
*class* AddTourManagerComponent *extends* Component {  
 render() {  
 *const* {  
 onBack,  
 onChangeFirstName,  
 onChangeLastName,  
 onChangePhoneNo,  
 onChangeEmail,  
 onChangeNIC,  
 onSubmitAdd,  
 onSubmitUpdate,  
 editUser,  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Form  
 *onSubmit*={editUser ? onSubmitUpdate : onSubmitAdd}  
 *style*={{  
 border: 'solid 1px',  
 padding: '20px',  
 borderRadius: '10px'  
 }}  
 >  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridFirstName'>  
 <Form.Label>First Name</Form.Label>  
 <Form.Control *placeholder*='Enter First Name'  
 *type*='text'  
 *onChange*={onChangeFirstName}  
 *value*={firstName}  
 *pattern*='[A-Za-z]{2,32}'  
 *title*='Please enter a valid first name.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridLastName'>  
 <Form.Label>Last Name</Form.Label>  
 <Form.Control *placeholder*='Enter Last Name'  
 *type*='text'  
 *onChange*={onChangeLastName}  
 *value*={lastName}  
 *pattern*='[A-Za-z]{2,32}'  
 *title*='Please enter a valid last name.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridPhoneNo'>  
 <Form.Label>Phone No</Form.Label>  
 <Form.Control *placeholder*='Enter Phone Number'  
 *type*='text'  
 *onChange*={onChangePhoneNo}  
 *value*={phoneNo}  
 *pattern*='0[0-9]{9}'  
 *title*='Please enter a valid phone number.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridEmail'>  
 <Form.Label>Email</Form.Label>  
 <Form.Control *placeholder*='Enter Email'  
 *type*='email'  
 *onChange*={onChangeEmail}  
 *value*={email}  
 *pattern*='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'  
 *title*='Please enter a valid email.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridNIC'>  
 <Form.Label>NIC</Form.Label>  
 <Form.Control *placeholder*='Enter NIC'  
 *type*='text'  
 *onChange*={onChangeNIC}  
 *value*={nic}  
 *pattern*='[0-9]{9}V'  
 *title*='Please enter a valid NIC.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 {  
 editUser ? (  
 <Form.Row>  
 <Form.Group>  
 <Button *variant*='primary'  
 *type*='button'  
 *className*={'btn btn-block btn-primary mt-3'}  
 *style*={{  
 marginLeft: '10px',  
 marginRight: '20px'  
 }}  
 *onClick*={onBack}  
 >  
 Back  
 </Button>  
 </Form.Group>  
 <Form.Group>  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*={'btn btn-block btn-primary mt-3'}  
 *style*={{  
 marginLeft: '20px',  
 marginRight: '20px'  
 }}  
 >  
 Edit  
 </Button>  
 </Form.Group>  
 </Form.Row>  
 ) : (  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*={'btn btn-block btn-primary mt-3'}  
 >  
 Add  
 </Button>  
 )  
 }  
 </Form>  
 </div>  
 )  
 }  
}  
  
*export default* AddTourManagerComponent

tour-manager-list-component.jsx

*import* React, {Component} *from* 'react'  
*import* Table *from* 'react-bootstrap/Table'  
*import* SingleUserComponent *from* '../tour-manager-single-component/tour-manager-single-component'  
*import* './tour-manager-list-component-styles.scss'  
  
*class* ListTourManagerComponent *extends* Component {  
 render() {  
 *const* {  
 users,  
 onSubmitEdit,  
 deleteUser  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Table *responsive striped bordered hover variant*='dark'>  
 <thead>  
 <tr>  
 <th>First Name</th>  
 <th>Last Name</th>  
 <th>Phone No</th>  
 <th>Email</th>  
 <th>NIC</th>  
 <th/>  
 <th/>  
 </tr>  
 </thead>  
 <tbody>  
 {  
 users.map(user => {  
 *return* <SingleUserComponent *user*={user}  
 *onSubmitEdit*={() => onSubmitEdit(user.\_id)}  
 *deleteUser*={() => deleteUser(user.\_id)}/>  
 })  
 }  
 </tbody>  
 </Table>  
 </div>  
 )  
 }  
}  
  
*export default* ListTourManagerComponent

tour-manager-single-component.jsx

*import* React, {Component} from 'react'  
*import* Button from 'react-bootstrap/Button'  
*import* {FaEdit, FaTrashAlt} *from* 'react-icons/fa'  
*import* './tour-manager-single-component-styles.scss'  
  
*class* SingleTourManagerComponent *extends* Component {  
 render() {  
 *const* {  
 user,  
 onSubmitEdit,  
 deleteUser  
 } = *this*.props  
  
 *return* (  
 <tr *key*={user.\_id}>  
 <td>{user.firstName}</td>  
 <td>{user.lastName}</td>  
 <td>{user.phoneNo}</td>  
 <td>{user.email}</td>  
 <td>{user.nic}</td>  
 <td>  
 <Button  
 *variant*={'primary'}  
 *onClick*={onSubmitEdit}  
 >  
 <FaEdit  
 *size*={20}  
 *style*={{  
 marginBottom: '4px',  
 marginLeft: '2px'  
 }}  
 />  
 </Button>  
 </td>  
 <td>  
 <Button  
 *variant*={'danger'}  
 *onClick*={deleteUser}  
 >  
 <FaTrashAlt  
 *size*={20}  
 *style*={{  
 marginBottom: '4px',  
 marginLeft: '2px'  
 }}  
 />  
 </Button>  
 </td>  
 </tr>  
 )  
 }  
}  
  
*export default* SingleTourManagerComponent

1. Manage Tour Packages Functionality

Backend Implementation

tour-routes.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* TourController = require('../controllers/tour-controller')  
  
router.post('/tour', TourController.addTour)  
router.put('/tour/:id', TourController.updateTour)  
router.delete('/tour/:id', TourController.deleteTour)  
router.get('/tour/:id', TourController.getTour)  
router.get('/tour', TourController.getTourList)  
  
module.exports = router

tour-model.js

*const* mongoose *=* require('mongoose')  
*const* uniqueValidator *=* require('mongoose-unique-validator')  
  
*const* Schema *=* mongoose.Schema  
  
*const* tourSchema *= new* Schema({  
 tourName: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 tourDescription: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 destination: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 startDate: {  
 type: Date,  
 required: *true*,  
 trim: *true* },  
 endDate: {  
 type: Date,  
 required: *true*,  
 unique: *true*,  
 trim: *true* },  
 pricePerPerson: {  
 type: Number,  
 required: *true*,  
 unique: *true*,  
 trim: *true* }  
}, {  
 timestamps: *true*,  
 collection: 'Tours'  
})  
  
tourSchema.plugin(uniqueValidator)  
  
*module*.*exports =* mongoose.model('Tours', tourSchema)

tour-controller.js

*const* Tour = require('../models/tour-model')  
  
require('dotenv').*config*()  
  
*const* addTour = *async* (req, res, next) => {  
 *let* existingTourName  
  
 *const* {  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 } = req.body  
  
 *try* {  
 existingTourName = *await* Tour.findOne({  
 tourName: tourName  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingTourName) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A tour with the same name already exists.'  
 })  
 *return* next('A tour with the same name already exists.')  
 }  
  
 *const* newTour = *new* Tour({  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 })  
  
 *try* {  
 *await* newTour.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(201).send({  
 message: 'New tour added successfully!'  
 })  
}  
  
*const* updateTour = *async* (req, res, next) => {  
 *let* tour  
 *let* existingTourName  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 } = req.body  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingTourName = *await* Tour.findOne({  
 tourName: tourName  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingTourName && tourName !== tour.tourName) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A tour with the same name already exists.'  
 })  
 *return* next('A tour with the same name already exists.')  
 }  
  
 tour.tourName = tourName  
 tour.tourDescription = tourDescription  
 tour.destination = destination  
 tour.startDate = startDate  
 tour.endDate = endDate  
 tour.pricePerPerson = pricePerPerson  
  
 *try* {  
 *await* tour.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour updated successfully!'  
 })  
}  
  
*const* deleteTour = *async* (req, res, next) => {  
 *let* tour  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 *await* tour.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour deleted successfully!'  
 })  
}  
  
*const* getTour = *async* (req, res, next) => {  
 *let* tour  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tour)  
}  
  
*const* getTourList = *async* (req, res, next) => {  
 *let* tourList  
  
 *try* {  
 tourList = *await* Tour.find({})  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourList)  
}  
  
exports.addTour = addTour  
exports.updateTour = updateTour  
exports.deleteTour = deleteTour  
exports.getTour = getTour  
exports.getTourList = getTourList

Frontend Implementation

manage-tour-component.jsx

*import* React, {Component} *from* 'react'  
*import* 'bootstrap/dist/css/bootstrap.min.css'  
*import* Swal *from* 'sweetalert2'  
*import* axios *from* 'axios'  
*import* {proxy} *from* '../../conf'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Row *from* 'react-bootstrap/Row'  
*import* AddTourComponent *from* '../../components/tour-add-component/tour-add-component'  
*import* ListTourComponent *from* '../../components/tour-list-component/tour-list-component'  
*import* NavigationBarComponent *from* '../../components/navigation-bar-component/navigation-bar-component'  
*import* './manage-tour-component-styles.scss'  
  
*class* ManageTourComponent *extends* Component {  
 constructor(props) {  
 *super*(props)  
 *this*.state = {  
 tours: [],  
 tourName: '',  
 tourDescription: '',  
 destination: '',  
 startDate: '',  
 endDate: '',  
 pricePerPerson: '',  
 editingTourId: '',  
 editingTour: *null*,  
 editTour: *false*,  
 loggedIn: *true*,  
 userType: 'Tour Manager'  
 }  
 }  
  
 componentDidMount() {  
 *this*.*getTours*()  
 }  
  
 *getTours* = () => {  
 axios.get(`${proxy}tour/tour`)  
 .then(res => {  
 *this*.setState({  
 tours: res.data  
 })  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *onChangeTourName* = event => {  
 *this*.setState({  
 tourName: event.target.value  
 })  
 }  
  
 *onChangeTourDescription* = event => {  
 *this*.setState({  
 tourDescription: event.target.value  
 })  
 }  
  
 *onChangeDestination* = event => {  
 *this*.setState({  
 destination: event.target.value  
 })  
 }  
  
 *onChangeStartDate* = event => {  
 *this*.setState({  
 startDate: event.target.value  
 })  
 }  
  
 *onChangeEndDate* = event => {  
 *this*.setState({  
 endDate: event.target.value  
 })  
 }  
  
 *onChangePricePerPerson* = event => {  
 *this*.setState({  
 pricePerPerson: event.target.value  
 })  
 }  
  
 *onBack* = () => {  
 *this*.setState({  
 tourName: '',  
 tourDescription: '',  
 destination: '',  
 startDate: '',  
 endDate: '',  
 pricePerPerson: '',  
 editingTourId: '',  
 editingTour: *null*,  
 editTour: *false* })  
 }  
  
 *onSubmitAdd* = event => {  
 event.preventDefault()  
 *const* tour = {  
 tourName: *this*.state.tourName,  
 tourDescription: *this*.state.tourDescription,  
 destination: *this*.state.destination,  
 startDate: *this*.state.startDate,  
 endDate: *this*.state.endDate,  
 pricePerPerson: *this*.state.pricePerPerson  
 }  
 axios.post(`${proxy}tour/tour`, tour)  
 .then(res => {  
 *if* (res.data.exists === *true*) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'A similar tour already exists!',  
 footer: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 *this*.*getTours*()  
 *this*.setState({  
 tourName: '',  
 tourDescription: '',  
 destination: '',  
 startDate: '',  
 endDate: '',  
 pricePerPerson: ''  
 })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'A new tour added successfully!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *deleteTour* = tourId => {  
 Swal.*fire*({  
 title: 'Are you sure?',  
 text: "You won't be able to revert this!",  
 icon: 'warning',  
 showCancelButton: *true*,  
 cancelButtonColor: '#3085d6',  
 confirmButtonColor: '#d33',  
 confirmButtonText: 'Yes, delete it!'  
 }).then((result) => {  
 *if* (result.value) {  
 axios.delete(`${proxy}tour/tour/${tourId}`)  
 .then(() => {  
 *this*.*getTours*()  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 Swal.*fire*(  
 'Deleted!',  
 'Tour has been deleted.',  
 'success'  
 )  
 }  
 })  
 }  
  
 *onSubmitEdit* = tourId => {  
 axios.get(`${proxy}tour/tour/${tourId}`)  
 .then(res => {  
 *this*.setState({  
 editingTourId: tourId,  
 editingTour: res.data  
 })  
 *this*.setState({  
 editTour: *true*,  
 tourName: *this*.state.editingTour.tourName,  
 tourDescription: *this*.state.editingTour.tourDescription,  
 destination: *this*.state.editingTour.destination,  
 startDate: *this*.state.editingTour.startDate,  
 endDate: *this*.state.editingTour.endDate,  
 pricePerPerson: *this*.state.editingTour.pricePerPerson  
 })  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 *onSubmitUpdate* = event => {  
 event.preventDefault()  
 *const* tour = {  
 tourName: *this*.state.tourName,  
 tourDescription: *this*.state.tourDescription,  
 destination: *this*.state.destination,  
 startDate: *this*.state.startDate,  
 endDate: *this*.state.endDate,  
 pricePerPerson: *this*.state.pricePerPerson  
 }  
 axios.put(`${proxy}tour/tour/${*this*.state.editingTourId}`, tour)  
 .then(res => {  
 *if* (res.data.exists === *true*) {  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'A similar tour already exists!',  
 footer: res.data.message  
 }).then(() => {  
 })  
 } *else* {  
 *this*.*getTours*()  
 *this*.setState({  
 tourName: '',  
 tourDescription: '',  
 destination: '',  
 startDate: '',  
 endDate: '',  
 pricePerPerson: '',  
 editingTourId: '',  
 editingTour: *null*,  
 editTour: *false* })  
 Swal.*fire*({  
 position: 'top-end',  
 icon: 'success',  
 title: 'Tour updated successfully!',  
 showConfirmButton: *false*,  
 timer: 1500  
 }).then(() => {  
 })  
 }  
 }).catch(error => {  
 *console*.log(error)  
 Swal.*fire*({  
 icon: 'error',  
 title: 'Oops...',  
 text: 'An unexpected error occurred. Please try again later.'  
 }).then(() => {  
 })  
 })  
 }  
  
 render() {  
 *return* (  
 <div *className*='container'>  
 <NavigationBarComponent *loggedIn*={*this*.state.loggedIn}  
 *userType*={*this*.state.userType}/>  
 <h1 *style*={{  
 textAlign: 'center',  
 marginTop: '80px',  
 marginBottom: '50px',  
 textTransform: 'uppercase',  
 letterSpacing: '4px',  
 color: 'darkblue'  
 }}>  
 Manage Tours  
 </h1>  
 <Row>  
 <Col *sm*='3'>  
 <AddTourComponent *onBack*={*this*.*onBack*}  
 *onChangeTourName*={*this*.*onChangeTourName*}  
 *onChangeTourDescription*={*this*.*onChangeTourDescription*}  
 *onChangeDestination*={*this*.*onChangeDestination*}  
 *onChangeStartDate*={*this*.*onChangeStartDate*}  
 *onChangeEndDate*={*this*.*onChangeEndDate*}  
 *onChangePricePerPerson*={*this*.*onChangePricePerPerson*}  
 *onSubmitAdd*={*this*.*onSubmitAdd*}  
 *onSubmitUpdate*={*this*.*onSubmitUpdate*}  
 *editTour*={*this*.state.editTour}  
 *tourName*={*this*.state.tourName}  
 *tourDescription*={*this*.state.tourDescription}  
 *destination*={*this*.state.destination}  
 *startDate*={*this*.state.startDate}  
 *endDate*={*this*.state.endDate}  
 *pricePerPerson*={*this*.state.pricePerPerson}/>  
 </Col>  
 <Col *sm*='9'>  
 <ListTourComponent *tours*={*this*.state.tours}  
 *onSubmitEdit*={*this*.*onSubmitEdit*}  
 *deleteTour*={*this*.*deleteTour*}/>  
 </Col>  
 </Row>  
 </div>  
 )  
 }  
}  
  
*export default* ManageTourComponent

tour-add-component.jsx

*import* React, {Component} from 'react'  
*import* Form from 'react-bootstrap/Form'  
*import* Col *from* 'react-bootstrap/Col'  
*import* Button *from* 'react-bootstrap/Button'  
*import* './tour-add-component-styles.scss'  
  
*class* AddTourComponent *extends* Component {  
 render() {  
 *const* {  
 onBack,  
 onChangeTourName,  
 onChangeTourDescription,  
 onChangeDestination,  
 onChangeStartDate,  
 onChangeEndDate,  
 onChangePricePerPerson,  
 onSubmitAdd,  
 onSubmitUpdate,  
 editTour,  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Form  
 *onSubmit*={editTour ? onSubmitUpdate : onSubmitAdd}  
 *style*={{  
 border: 'solid 1px',  
 padding: '20px',  
 borderRadius: '10px'  
 }}  
 >  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridTourName'>  
 <Form.Label>Tour Name</Form.Label>  
 <Form.Control *placeholder*='Enter Tour Name'  
 *type*='text'  
 *onChange*={onChangeTourName}  
 *value*={tourName}  
 *title*='Please enter a tour name.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridTourDescription'>  
 <Form.Label>Tour Description</Form.Label>  
 <Form.Control *placeholder*='Enter Tour Description'  
 *type*='text' *onChange*={onChangeTourDescription}  
 *value*={tourDescription}  
 *title*='Please enter a tour description.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridDestination'>  
 <Form.Label>Destination</Form.Label>  
 <Form.Control *placeholder*='Enter Destination'  
 *type*='text' *onChange*={onChangeDestination}  
 *value*={destination}  
 *title*='Please enter a destination.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridStartDate'>  
 <Form.Label>Start Date</Form.Label>  
 <Form.Control *placeholder*='Enter Start Date'  
 *type*='date'  
 *onChange*={onChangeStartDate}  
 *value*={startDate.substring(0, 10)}  
 *title*='Please enter a start date.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridEndDate'>  
 <Form.Label>End Date</Form.Label>  
 <Form.Control *placeholder*='Enter End Date'  
 *type*='date'  
 *onChange*={onChangeEndDate}  
 *value*={endDate.substring(0, 10)}  
 *title*='Please enter an end date.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 <Form.Row>  
 <Form.Group *as*={Col}  
 *controlId*='formGridPricePerPerson'>  
 <Form.Label>Price Per Person (LKR)</Form.Label>  
 <Form.Control *placeholder*='Enter Price per Person'  
 *type*='string'  
 *onChange*={onChangePricePerPerson}  
 *value*={pricePerPerson}  
 *pattern*='[0-9]{1,16}'  
 *title*='Please enter valid price per person.'  
 *required*/>  
 </Form.Group>  
 </Form.Row>  
 {  
 editTour ? (  
 <Form.Row>  
 <Form.Group>  
 <Button *variant*='primary'  
 *type*='button'  
 *className*={'btn btn-block btn-primary mt-3'}  
 *style*={{  
 marginLeft: '10px',  
 marginRight: '20px'  
 }}  
 *onClick*={onBack}  
 >  
 Back  
 </Button>  
 </Form.Group>  
 <Form.Group>  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*={'btn btn-block btn-primary mt-3'}  
 *style*={{  
 marginLeft: '20px',  
 marginRight: '20px'  
 }}  
 >  
 Edit  
 </Button>  
 </Form.Group>  
 </Form.Row>  
 ) : (  
 <Button *variant*='primary'  
 *type*='submit'  
 *className*={'btn btn-block btn-primary mt-3'}  
 >  
 Add  
 </Button>  
 )  
 }  
 </Form>  
 </div>  
 )  
 }  
}  
  
*export default* AddTourComponent

tour-list-component.jsx

*import* React, {Component} from 'react'  
*import* Table *from* 'react-bootstrap/Table'  
*import* SingleTourComponent *from* '../tour-single-component/tour-single-component'  
*import* './tour-list-component-styles.scss'  
  
*class* ListTourComponent *extends* Component {  
 render() {  
 *const* {  
 tours,  
 onSubmitEdit,  
 deleteTour  
 } = *this*.props  
  
 *return* (  
 <div>  
 <Table *responsive striped bordered hover variant*='dark'>  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Description</th>  
 <th>Destination</th>  
 <th>Start Date</th>  
 <th>End Date</th>  
 <th>Price (LKR)</th>  
 <th/>  
 <th/>  
 </tr>  
 </thead>  
 <tbody>  
 {  
 tours.map(tour => {  
 *return* <SingleTourComponent *tour*={tour}  
 *onSubmitEdit*={() => onSubmitEdit(tour.\_id)}  
 *deleteTour*={() => deleteTour(tour.\_id)}/>  
 })  
 }  
 </tbody>  
 </Table>  
 </div>  
 )  
 }  
}  
  
*export default* ListTourComponent

tour-single-component.jsx

*import* React, {Component} *from* 'react'  
*import* Button *from* 'react-bootstrap/Button'  
*import* {FaEdit, FaTrashAlt} *from* 'react-icons/fa'  
*import* './tour-single-component-styles.scss'  
  
*class* SingleTourComponent *extends* Component {  
 render() {  
 *const* {  
 tour,  
 onSubmitEdit,  
 deleteTour  
 } = *this*.props  
  
 *return* (  
 <tr *key*={tour.\_id}>  
 <td>{tour.tourName}</td>  
 <td>{tour.tourDescription}</td>  
 <td>{tour.destination}</td>  
 <td>{tour.startDate.substring(0, 10)}</td>  
 <td>{tour.endDate.substring(0, 10)}</td>  
 <td>{tour.pricePerPerson}</td>  
 <td>  
 <Button  
 *variant*={'primary'}  
 *onClick*={onSubmitEdit}  
 >  
 <FaEdit  
 *size*={20}  
 *style*={{  
 marginBottom: '4px',  
 marginLeft: '2px'  
 }}  
 />  
 </Button>  
 </td>  
 <td>  
 <Button  
 *variant*={'danger'}  
 *onClick*={deleteTour}  
 >  
 <FaTrashAlt  
 *size*={20}  
 *style*={{  
 marginBottom: '4px',  
 marginLeft: '2px'  
 }}  
 />  
 </Button>  
 </td>  
 </tr>  
 )  
 }  
}  
  
*export default* SingleTourComponent

**RESTful web service (implemented code)**

admin-controller.js

*const* User = require('../models/user-model')  
  
*const* addAdmin = *async* (req, res, next) => {  
 *let* existingUser  
  
 *const* {  
 email,  
 password  
 } = req.body  
  
 *try* {  
 existingUser = *await* User.findOne({  
 email: email  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUser) {  
 res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *const* newAdmin = *new* User({  
 email,  
 password,  
 type: 'Administrator'  
 })  
  
 *try* {  
 *await* newAdmin.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(201).send({  
 message: 'New administrator added successfully!'  
 })  
}  
  
*const* updateAdmin = *async* (req, res, next) => {  
 *let* admin  
 *let* existingUser  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 email,  
 password  
 } = req.body  
  
 *try* {  
 admin = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingUser = *await* User.findOne({  
 email: email  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUser && email !== admin.email) {  
 res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 admin.email = email  
 admin.password = password  
  
 *try* {  
 *await* admin.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Administrator updated successfully!',  
 })  
}  
  
*const* deleteAdmin = *async* (req, res, next) => {  
 *let* admin  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 admin = *await* User.findById(id)  
 *await* admin.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Administrator deleted successfully!',  
 })  
}  
  
*const* getAdmin = *async* (req, res, next) => {  
 *let* admin  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 admin = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(admin)  
}  
  
*const* getAdminList = *async* (req, res, next) => {  
 *let* adminList  
  
 *try* {  
 adminList = *await* User.find({  
 type: 'Administrator'  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(adminList)  
}  
  
exports.addAdmin = addAdmin  
exports.updateAdmin = updateAdmin  
exports.deleteAdmin = deleteAdmin  
exports.getAdmin = getAdmin  
exports.getAdminList = getAdminList

login-controller.js

*const* User = require('../models/user-model')  
  
*const* register = *async* (req, res, next) => {  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 } = req.body  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 *const* newUser = *new* User({  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 })  
  
 *try* {  
 *await* newUser.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(201).send({  
 message: 'Customer registered added successfully!'  
 })  
}  
  
*const* login = *async* (req, res, next) => {  
 *const* {  
 email,  
 password  
 } = req.body  
  
 *let* existingUser  
  
 *try* {  
 existingUser = *await* User.findOne({  
 email: email  
 })  
 } *catch* (error) {  
 res.send({  
 message: 'Login failed, please try again later.',  
 login: -1  
 })  
  
 *return* next(error)  
 }  
  
 *if* (!existingUser || existingUser.password !== password) {  
 res.send({  
 message: 'Invalid username or password.',  
 login: 0  
 })  
  
 *return* next(error)  
 }  
  
 res.send({  
 message: 'Logged in!',  
 login: 1,  
 type: existingUser.type,  
 userDetails: existingUser  
 })  
}  
  
*const* updateUser = *async* (req, res, next) => {  
 *let* user  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password  
 } = req.body  
  
 *try* {  
 user = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail && email !== user.email) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC && nic !== user.nic) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 user.firstName = firstName  
 user.lastName = lastName  
 user.teleNo = phoneNo  
 user.email = email  
 user.nic = nic  
 user.password = password  
  
 *try* {  
 *await* user.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'User updated successfully!'  
 })  
}  
  
*const* deleteUser = *async* (req, res, next) => {  
 *let* user  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 user = *await* User.findById(id)  
 *await* user.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'User deleted successfully!'  
 })  
}  
  
*const* getUser = *async* (req, res, next) => {  
 *let* user  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 user = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(user)  
}  
  
*const* getCustomerList = *async* (req, res, next) => {  
 *let* customerList  
  
 *try* {  
 customerList = *await* User.find({  
 type: 'Customer'  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(customerList)  
}  
  
exports.register = register  
exports.login = login  
exports.updateUser = updateUser  
exports.deleteUser = deleteUser  
exports.getUser = getUser  
exports.getCustomerList = getCustomerList

tour-controller.js

*const* Tour = require('../models/tour-model')  
  
require('dotenv').*config*()  
  
*const* addTour = *async* (req, res, next) => {  
 *let* existingTourName  
  
 *const* {  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 } = req.body  
  
 *try* {  
 existingTourName = *await* Tour.findOne({  
 tourName: tourName  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingTourName) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A tour with the same name already exists.'  
 })  
 *return* next('A tour with the same name already exists.')  
 }  
  
 *const* newTour = *new* Tour({  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 })  
  
 *try* {  
 *await* newTour.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(201).send({  
 message: 'New tour added successfully!'  
 })  
}  
  
*const* updateTour = *async* (req, res, next) => {  
 *let* tour  
 *let* existingTourName  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
 } = req.body  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingTourName = *await* Tour.findOne({  
 tourName: tourName  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingTourName && tourName !== tour.tourName) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A tour with the same name already exists.'  
 })  
 *return* next('A tour with the same name already exists.')  
 }  
  
 tour.tourName = tourName  
 tour.tourDescription = tourDescription  
 tour.destination = destination  
 tour.startDate = startDate  
 tour.endDate = endDate  
 tour.pricePerPerson = pricePerPerson  
  
 *try* {  
 *await* tour.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour updated successfully!'  
 })  
}  
  
*const* deleteTour = *async* (req, res, next) => {  
 *let* tour  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 *await* tour.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour deleted successfully!'  
 })  
}  
  
*const* getTour = *async* (req, res, next) => {  
 *let* tour  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tour = *await* Tour.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tour)  
}  
  
*const* getTourList = *async* (req, res, next) => {  
 *let* tourList  
  
 *try* {  
 tourList = *await* Tour.find({})  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourList)  
}  
  
exports.addTour = addTour  
exports.updateTour = updateTour  
exports.deleteTour = deleteTour  
exports.getTour = getTour  
exports.getTourList = getTourList

tour-manager-controller.js

*const* nodemailer = require('nodemailer')  
  
*const* User = require('../models/user-model')  
  
require('dotenv').*config*()  
  
*const* addTourManager = *async* (req, res, next) => {  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic  
 } = req.body  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 *let* generatedPassword = generatePassword()  
  
 *const* newTourManager = *new* User({  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic,  
 password: generatedPassword,  
 type: 'Tour Manager'  
 })  
  
 *try* {  
 *await* newTourManager.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *await* sendEmail(email, generatedPassword)  
  
 res.status(201).send({  
 message: 'New tour manager added successfully!'  
 })  
}  
  
*const* updateTourManager = *async* (req, res, next) => {  
 *let* tourManager  
 *let* existingUserEmail  
 *let* existingUserNIC  
  
 *const* {  
 id  
 } = req.params  
  
 *const* {  
 firstName,  
 lastName,  
 phoneNo,  
 email,  
 nic  
 } = req.body  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *try* {  
 existingUserEmail = *await* User.findOne({  
 email: email  
 })  
 existingUserNIC = *await* User.findOne({  
 nic: nic  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 *if* (existingUserEmail && email !== tourManager.email) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
 }  
  
 *if* (existingUserNIC && nic !== tourManager.nic) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A user with the same NIC already exists.'  
 })  
 *return* next('A user with the same NIC already exists.')  
 }  
  
 tourManager.firstName = firstName  
 tourManager.lastName = lastName  
 tourManager.teleNo = phoneNo  
 tourManager.email = email  
 tourManager.nic = nic  
  
 *try* {  
 *await* tourManager.save()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour manager updated successfully!'  
 })  
}  
  
*const* deleteTourManager = *async* (req, res, next) => {  
 *let* tourManager  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 *await* tourManager.remove()  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send({  
 message: 'Tour manager deleted successfully!'  
 })  
}  
  
*const* getTourManager = *async* (req, res, next) => {  
 *let* tourManager  
  
 *const* {  
 id  
 } = req.params  
  
 *try* {  
 tourManager = *await* User.findById(id)  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourManager)  
}  
  
*const* getTourManagerList = *async* (req, res, next) => {  
 *let* tourManagerList  
  
 *try* {  
 tourManagerList = *await* User.find({  
 type: 'Tour Manager'  
 })  
 } *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
 }  
  
 res.status(200).send(tourManagerList)  
}  
  
*const* getAdminEmail = *async* () => {  
 *let* admin  
  
 *try* {  
 admin = *await* User.findOne({  
 type: 'Administrator'  
 })  
 } *catch* (error) {  
 *return* error  
 }  
  
 *return* admin.email;  
}  
  
*let* transporter = nodemailer.*createTransport*({  
 service: 'gmail',  
 auth: {  
 user: 'it18149654@gmail.com',  
 pass: process.env.PASSWORD  
 }  
})  
  
*const* sendEmail = *async* (email, password) => {  
 *let* adminEmail = getAdminEmail()  
  
 *let* info = {  
 from: adminEmail,  
 to: email,  
 subject: 'Added as a Tour Manager',  
 text:  
 `Congratulations!  
 You have been assigned as a Tour Manager.  
 Now you can manage tours related operations as a tour manager in the Online Tourism Planner.  
 Please find your login credentials below.  
 LOGIN CREDENTIALS  
 Email: ${email}  
 Password: ${password}  
 Thank you!  
 This is an auto-generated email.  
 If this has been sent by mistake, please delete this without sharing this.  
 All rights reserved.`,  
 html:  
 `<!--suppress HtmlDeprecatedAttribute -->  
 <div style="margin: 0; padding: 0; background-color: #f2f2f2; font-family: arial, serif;">  
 <table style="margin: 0 auto; background: white; max-width: 500px; padding-bottom: 0; border-top: 5px solid #588dde; border-bottom: 5px solid #588dde; width: 100%;">  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 100%;">  
 <tr>  
 <td style="padding: 10px;">  
 <h1 style="text-align: center; color: #1a1a72;">Congratulations!</h1>  
 <h2 style="margin-top:25px; margin-bottom: 0; color: #4db0c4; font-weight: 400; font-size: medium;">You have been added as a Tour Manager.</h2>  
 <h2 style="margin-top:20px; margin-bottom: 0; color: #4db0c4; font-weight: 400; font-size: medium;">Now you can manage tour packages in the Online Tourism Planner.</h2>  
 <h2 style="margin-top:20px; margin-bottom: 10px; color: #4db0c4; font-weight: 400; font-size: medium;">Please find your login credentials below.</h2>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 100%;">  
 <tr>  
 <td style="padding: 10px;">  
 <h4 style="margin-top:20px; margin-bottom: 8px; color: #145a7a; font-weight: 400; text-align: center; font-size: 16px;"><b>LOGIN CREDENTIALS</b></h4>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>  
 <table align="left" style="width: 50%;">  
 <tr>  
 <td align="left" valign="top" style="padding: 10px;">  
 <h6 style="font-size: 14px; margin-top: 0; margin-bottom: 0; color: #29353c; font-weight: 400;">E-mail</h6>  
 <h6 style="font-size: 14px; margin-top: 20px; margin-bottom: 0; color: #29353c; font-weight: 400;">Password</h6>  
 </td>  
 </tr>  
 </table>  
 <table align="left" style="width: 50%;">  
 <tr>  
 <td align="right" valign="top" style="padding: 10px;">  
 <h6 style="font-size: 14px; margin-top: 0; margin-bottom: 0; color: #588dde; font-weight: 400;">${email}</h6>  
 <h6 style="font-size: 14px; margin-top: 20px; margin-bottom: 0; color: #588dde; font-weight: 400;">${password}</h6>  
 </td>  
 </tr>  
 </table>  
 </td>  
 </tr>  
 <tr style="background: rgb(237, 243, 255); padding-left: 20px; padding-right: 20px;">  
 <td>`  
 }  
  
 *// noinspection JSCheckFunctionSignatures* transporter.sendMail(info, (error, data) => {  
 *if* (error) {  
 *console*.log(error)  
 *console*.log('Email sending failed! Please try again.')  
 } *else* {  
 *console*.log(data)  
 *console*.log('An email is sent successfully to ' + email + '.')  
 }  
 })  
}  
  
*function* generatePassword() {  
 *let* length = 5  
 *let* randomPassword = ''  
 *let* characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'  
 *let* charactersLength = characters.length  
  
 *for* (*let* i = 0; i < length; i++)  
 randomPassword += characters.charAt(Math.floor(Math.random() \* charactersLength))  
  
 *return* randomPassword  
}  
  
exports.addTourManager = addTourManager  
exports.updateTourManager = updateTourManager  
exports.deleteTourManager = deleteTourManager  
exports.getTourManager = getTourManager  
exports.getTourManagerList = getTourManagerList

admin-routes.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* AdminController = require('../controllers/admin-controller')  
  
router.post('/admin', AdminController.addAdmin)  
router.put('/admin/:id', AdminController.updateAdmin)  
router.delete('/admin/:id', AdminController.deleteAdmin)  
router.get('/admin/:id', AdminController.getAdmin)  
router.get('/admin', AdminController.getAdminList)  
  
module.exports = router

tour-manager-routes.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* TourManagerController = require('../controllers/tour-manager-controller')  
  
router.post('/manager', TourManagerController.addTourManager)  
router.put('/manager/:id', TourManagerController.updateTourManager)  
router.delete('/manager/:id', TourManagerController.deleteTourManager)  
router.get('/manager/:id', TourManagerController.getTourManager)  
router.get('/manager', TourManagerController.getTourManagerList)  
  
module.exports = router

login-routes.js

*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* LoginController = require('../controllers/login-controller')  
  
router.post('/register', LoginController.register);  
router.post('/login', LoginController.login);  
router.put('/user/:id', LoginController.updateUser)  
router.delete('/user/:id', LoginController.deleteUser)  
router.get('/user/:id', LoginController.getUser)  
router.get('/user', LoginController.getCustomerList)  
  
module.exports = router

tour-routes.js

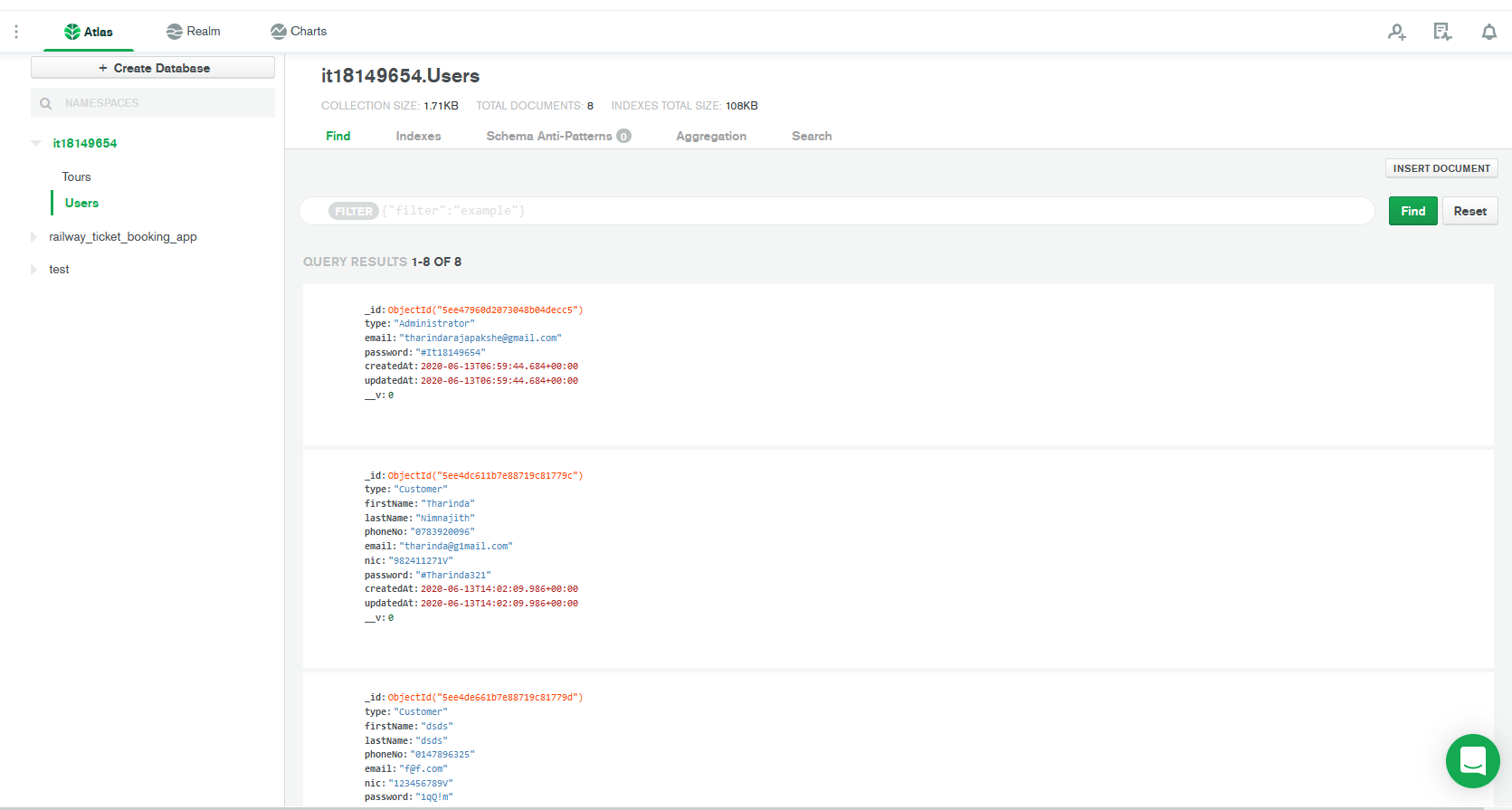
*const* express = require('express')  
*const* router = express.*Router*()  
  
*const* TourManagerController = require('../controllers/tour-manager-controller')  
  
router.post('/manager', TourManagerController.addTourManager)  
router.put('/manager/:id', TourManagerController.updateTourManager)  
router.delete('/manager/:id', TourManagerController.deleteTourManager)  
router.get('/manager/:id', TourManagerController.getTourManager)  
router.get('/manager', TourManagerController.getTourManagerList)  
  
module.exports = router

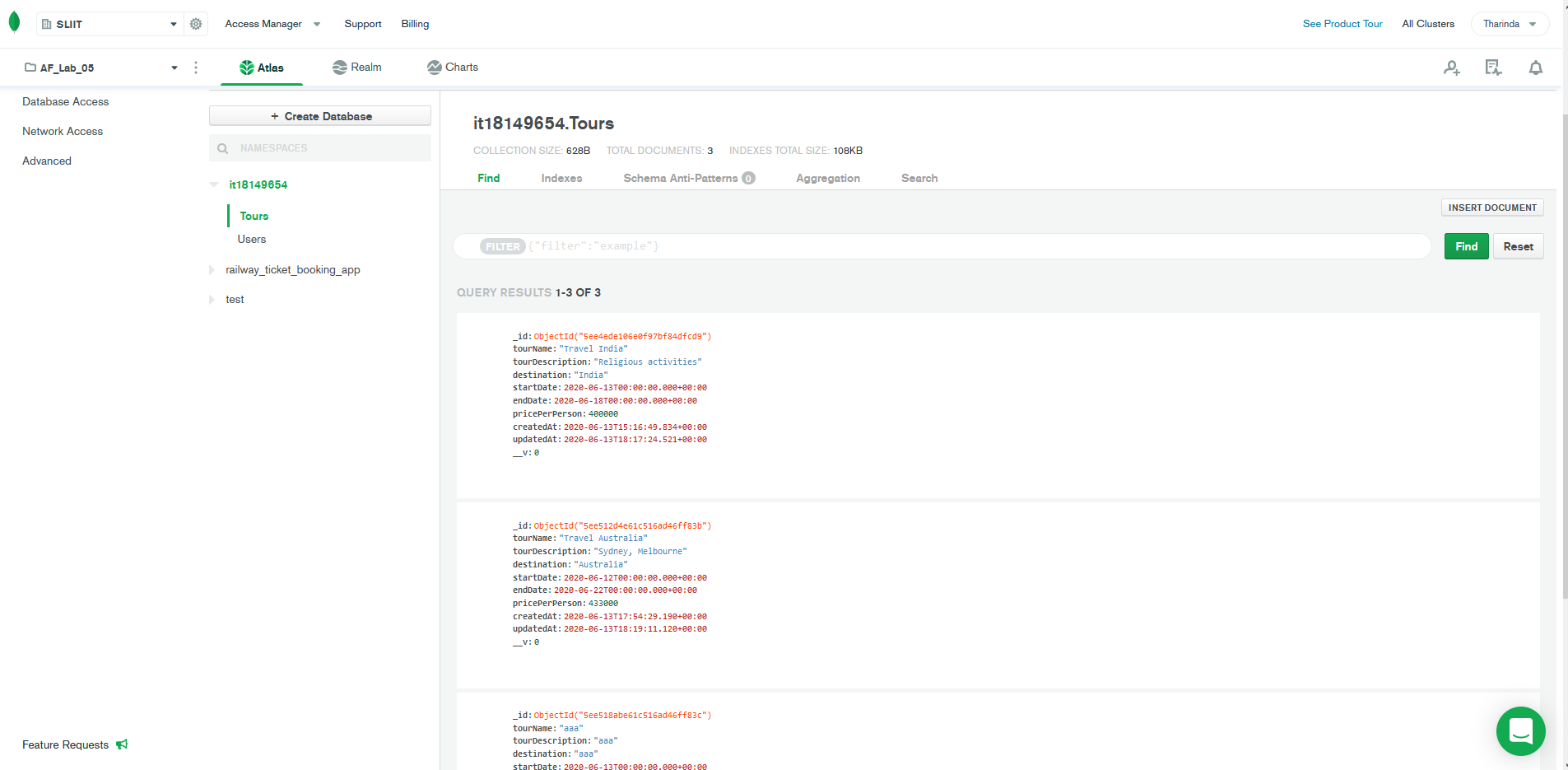
server.js

app.use('/manager', TourManagerRoutes)  
app.use('/admin', AdminRoutes)  
app.use('/tour', TourRoutes)  
app.use('/login', LoginRoutes)

**Mongo query for a specific Mongo collection**

*const* {  
 email,  
 password  
} = req.body  
  
*try* {  
 admin = *await* User.findById(id)  
} *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
}  
  
*try* {  
 existingUser = *await* User.findOne({  
 email: email  
 })  
} *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
}  
  
*if* (existingUser && email !== admin.email) {  
 res.json({  
 exists: *true*,  
 message: 'A user with the same email already exists.'  
 })  
 *return* next('A user with the same email already exists.')  
}





*const* {  
 tourName,  
 tourDescription,  
 destination,  
 startDate,  
 endDate,  
 pricePerPerson  
} = req.body  
  
*try* {  
 tour = *await* Tour.findById(id)  
} *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
}  
  
*try* {  
 existingTourName = *await* Tour.findOne({  
 tourName: tourName  
 })  
} *catch* (error) {  
 *return* next('Unexpected internal server error occurred, please try again later.')  
}  
  
*if* (existingTourName && tourName !== tour.tourName) {  
 *await* res.json({  
 exists: *true*,  
 message: 'A tour with the same name already exists.'  
 })  
 *return* next('A tour with the same name already exists.')  
}

*const* mongoose = require('mongoose')  
*const uniqueValidator* = require('mongoose-unique-validator')  
  
*const* Schema = mongoose.Schema  
  
*const* userSchema = *new* Schema({  
 firstName: {  
 type: String,  
 trim: *true* },  
 lastName: {  
 type: String,  
 trim: *true* },  
 phoneNo: {  
 type: String,  
 trim: *true* },  
 email: {  
 type: String,  
 required: *true*,  
 unique: *true*,  
 trim: *true* },  
 nic: {  
 type: String,  
 unique: *true*,  
 trim: *true* },  
 password: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 type: {  
 type: String,  
 *default*: 'Customer'  
 }  
}, {  
 timestamps: *true*,  
 collection: 'Users'  
})  
  
userSchema.plugin(*uniqueValidator*)  
  
module.exports = mongoose.model('Users', userSchema)

*const* mongoose *=* require('mongoose')  
*const* uniqueValidator *=* require('mongoose-unique-validator')  
  
*const* Schema *=* mongoose.Schema  
  
*const* tourSchema *= new* Schema({  
 tourName: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 tourDescription: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 destination: {  
 type: String,  
 required: *true*,  
 trim: *true* },  
 startDate: {  
 type: Date,  
 required: *true*,  
 trim: *true* },  
 endDate: {  
 type: Date,  
 required: *true*,  
 unique: *true*,  
 trim: *true* },  
 pricePerPerson: {  
 type: Number,  
 required: *true*,  
 unique: *true*,  
 trim: *true* }  
}, {  
 timestamps: *true*,  
 collection: 'Tours'  
})  
  
tourSchema.plugin(uniqueValidator)  
  
*module*.*exports =* mongoose.model('Tours', tourSchema)

**Unit test using JEST to evaluate a core logic or a service (implemented code)**

**Screenshot of the home page of the running application on localhost**

Note: I added screenshots of some of the other pages in the “Scenario” section.

**Thank You!**