Sri Lanka Institute of Information Technology

Distributed Systems

Assignment 2

Online Train Reservation System

Student ID	Name with initials
IT17137492	Atheeq Mahroof

Table of Contents

Table of Figures

1 Introduction to the system

The following report reflects the "Online Train Reservation System" application which was developed for the academic purpose of the module Distributed Systems. This application has both a front end and back end implementation with the help of a database.

ReactJs is used to develop the front end of the application while NodeJs/Express is used to develop the backend of the application. MongoDB (cloud account is used to store the data collection) is the database used for this application and for the function of sending an email, we have used nodemailer.

Initially the user should have a valid account in our portal. If they don't have one then the user is asked to create an account. After the creation of a successful account, user should login to the system. Then the list of train details will be listed and the user should enter the required trainId, ticketsand nic (if they wish for a discount entertained by the government employee). Depending on these details, the user will be shown the total amount and the user should select the payment method either MobilePay or Credit Card payment. After the payment is validated the user will be sent an email and the user will be redirected to the users's transaction history which shows the entire transaction which user made with portal. Following are the workflow diagram and high-level diagram of the system.

2 Workflow Diagram

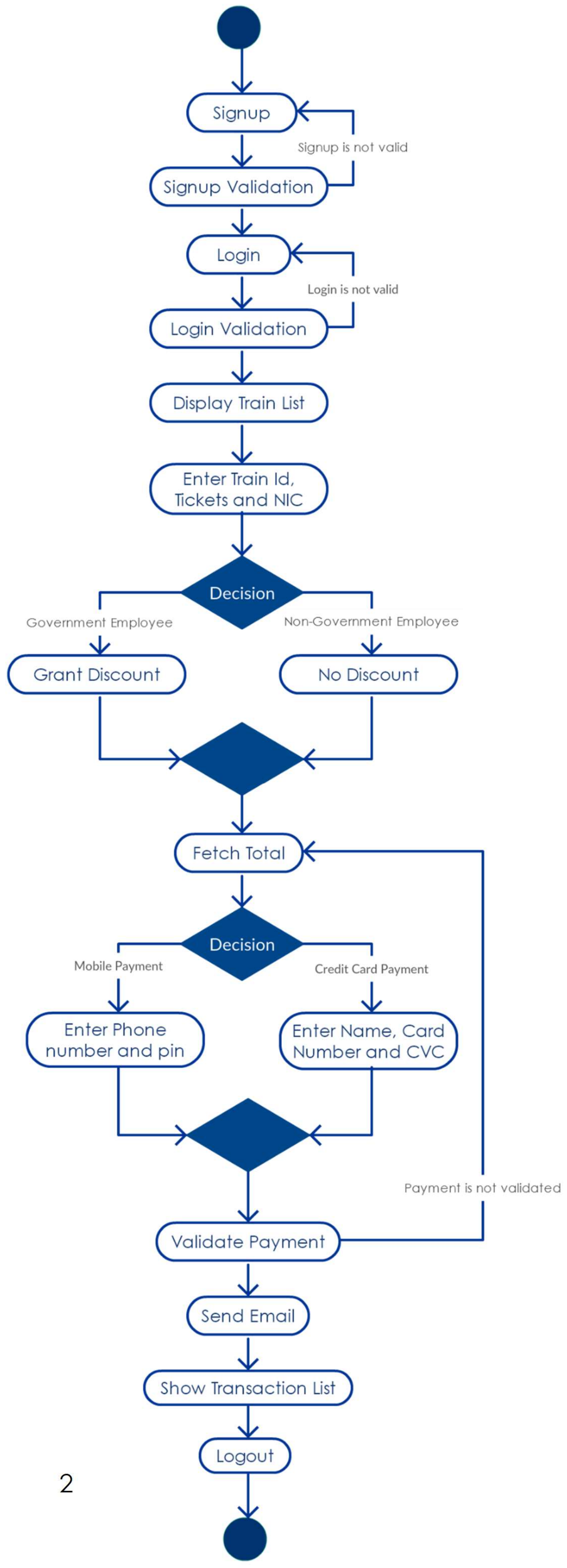


Figure 2-1 Workflow Diagram

3 High - level Diagram

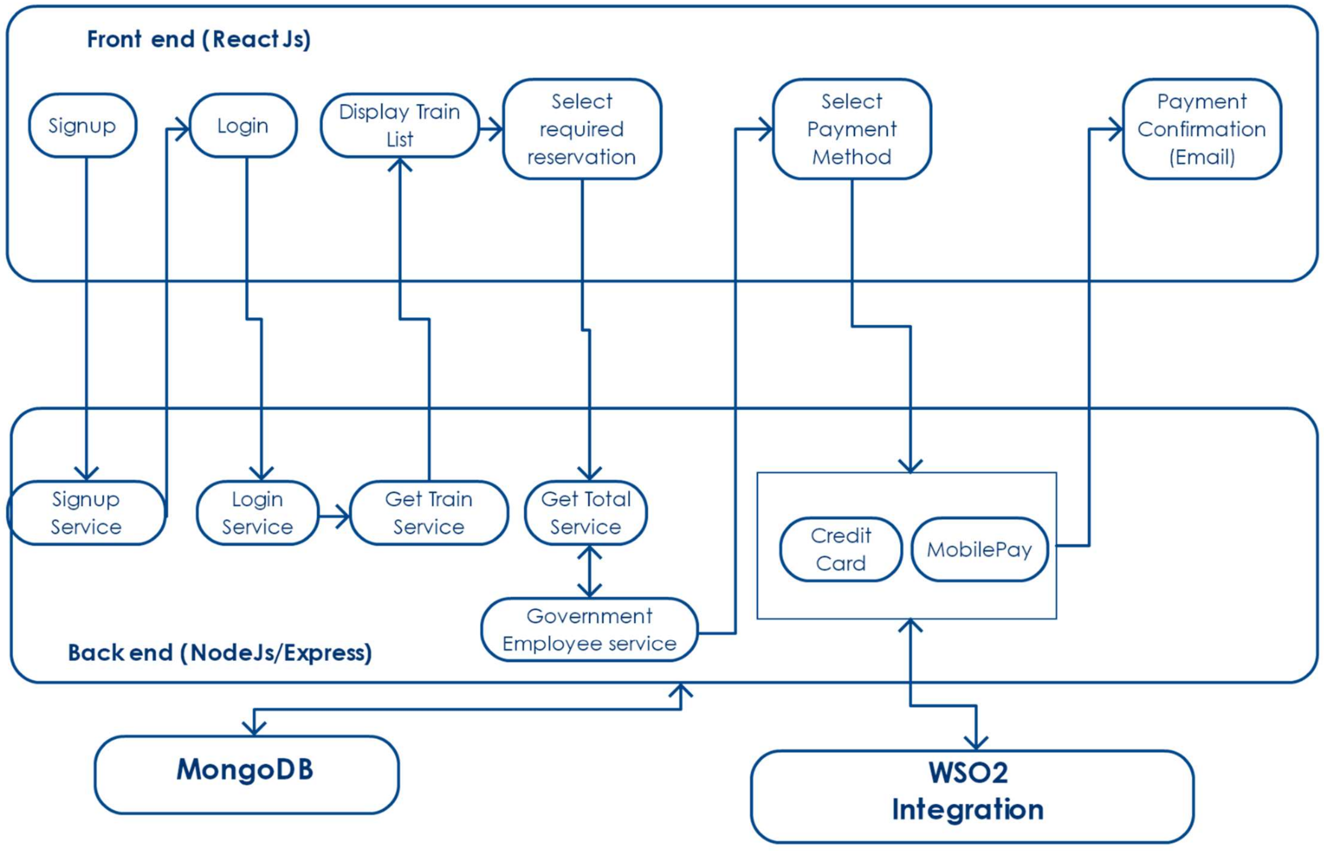
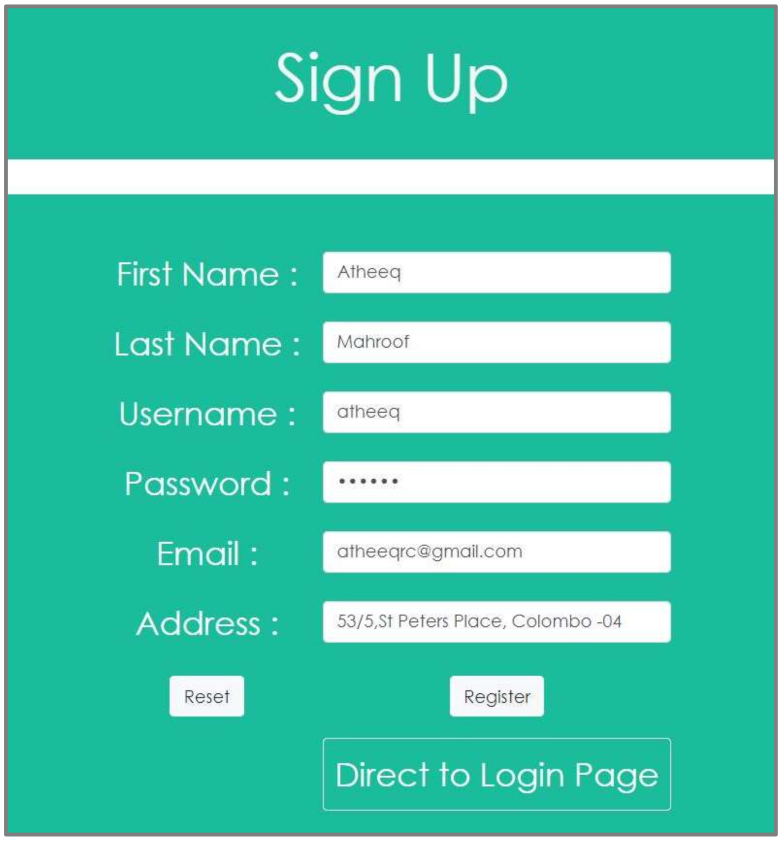


Figure 3-1 High Level Diagram

4 System workflow execution

Initially the customer should Signup for the system by entering first name, last name, a unique username, password, email and the address.

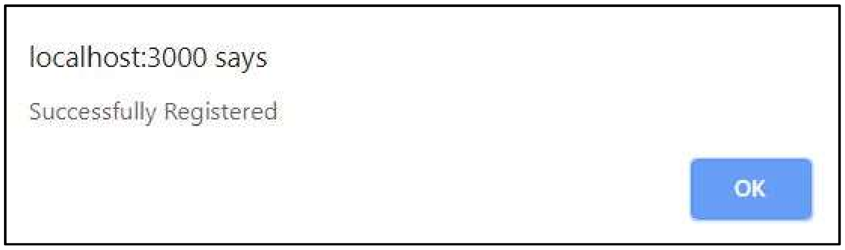


Sign Up

First Name :	Atheeq
Last Name :	Mahroof
Username :	atheeq
Password :	••••••
Email :	atheeqrc@gmail.com
Address :	53/5,St Peters Place, Colombo -04

Figure 4-1 Signup

After the successful registration of user, the system will display a success message.

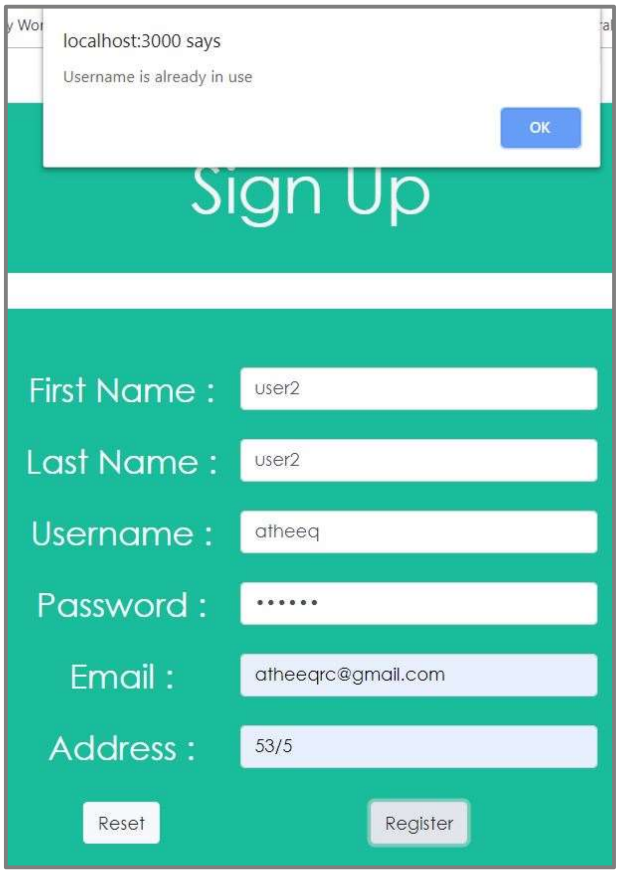


localhost:3000 says

Successfully Registered

Figure 4-2 SIgnup Successful

If the user creates an account with existing username, then the system will display an error. (Already an user with username – atheeq has been created)


localhost:3000 says

Username is already in use

Figure 4-3 Signup not unique username

After the registration process, the user is redirected to login page

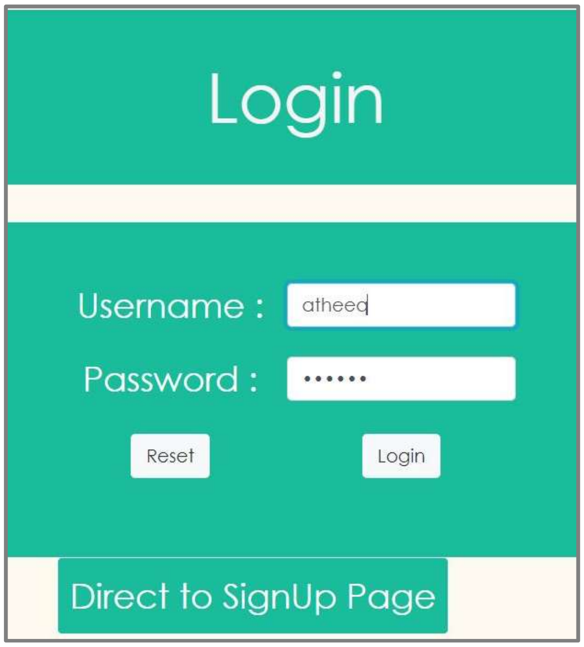
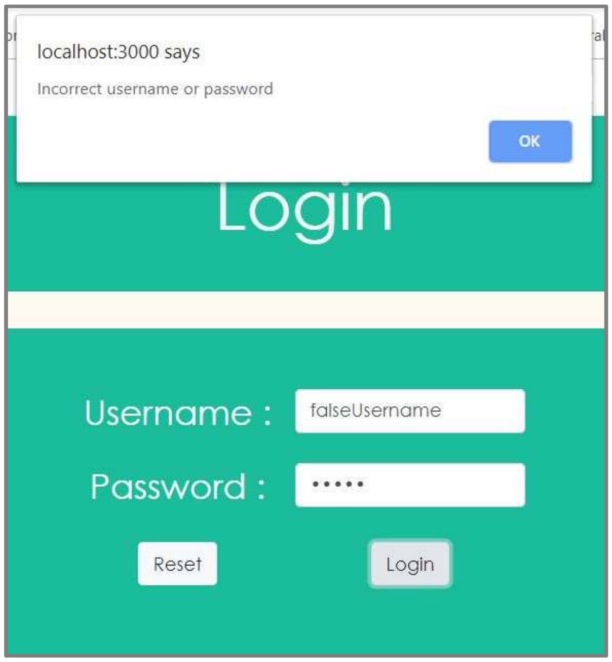


Figure 4-4 Login

If the username or password entered is incorrect or left blank, the system will show an error message. This process validates the authentication of the users.

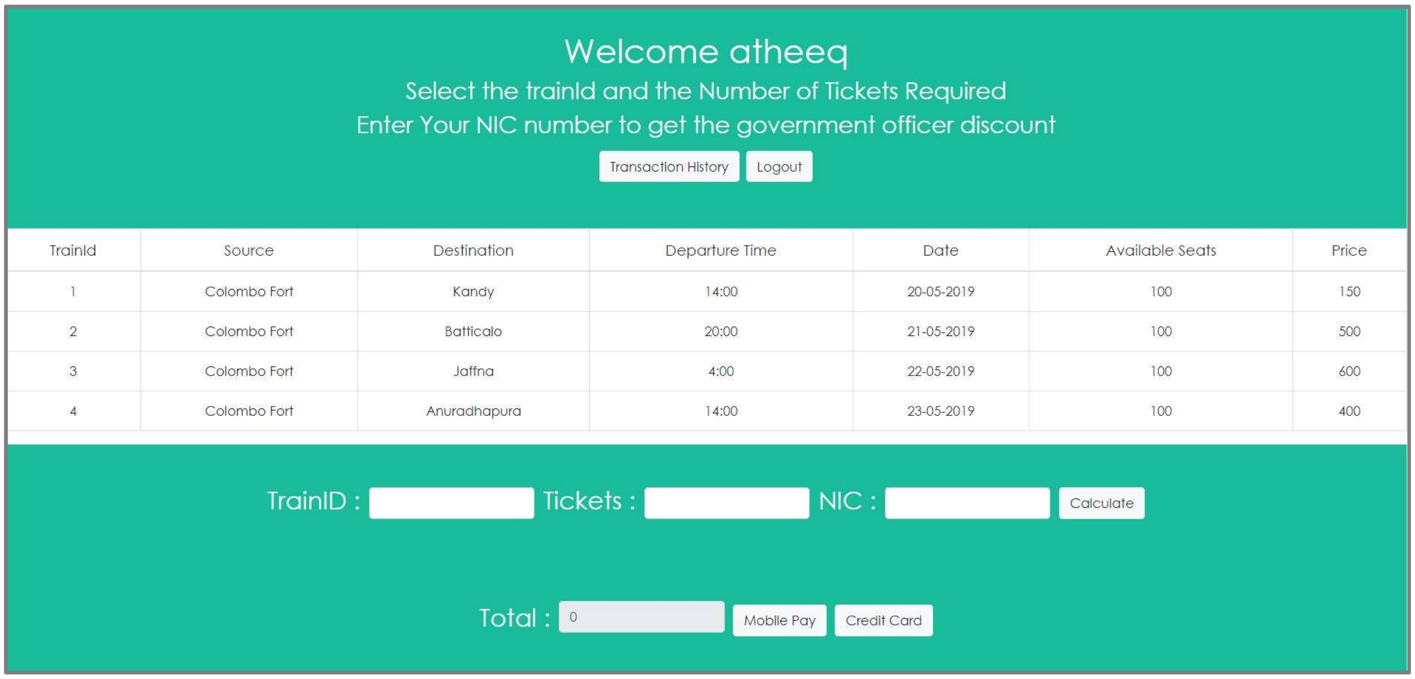

localhost:3000 says

Incorrect username or password

Figure 4-5 Invalid login

After the successful login, the user is redirected to view all the train details. In the header the username of the user is fetched using the state object.

```
<h1>Welcome  {this.state.username} </h1>
```



Welcome atheeq

Select the trainId and the Number of Tickets Required

Enter Your NIC number to get the government officer discount

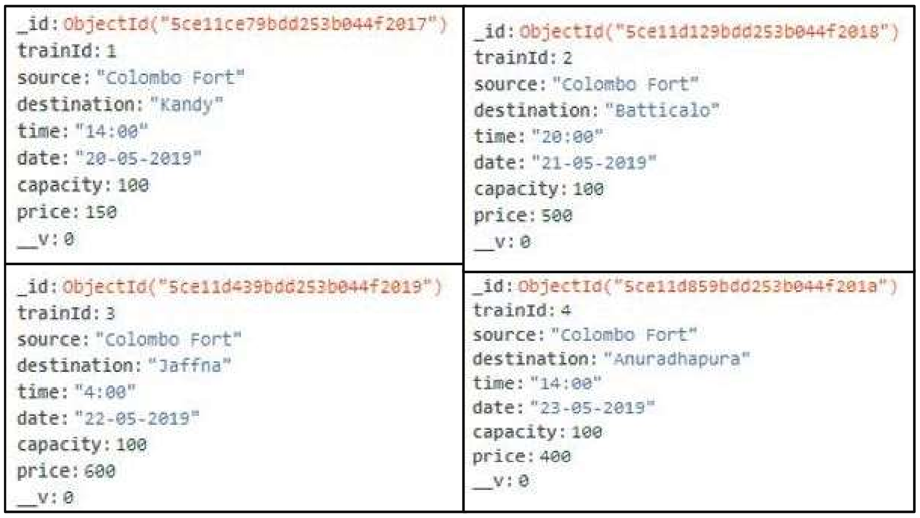
TrainId	Source	Destination	Departure Time	Date	Available Seats	Price
1	Colombo Fort	Kandy	14:00	20-05-2019	100	150
2	Colombo Fort	Batticalo	20:00	21-05-2019	100	500
3	Colombo Fort	Jaffna	4:00	22-05-2019	100	600
4	Colombo Fort	Anuradhapura	14:00	23-05-2019	100	400

TrainID : Tickets : NIC :

Total : 0

Figure 4-6 Train Details

The collection related to the above train details is shown in the below figure



```
_id: ObjectId("5ce11ce79bdd253b044f2017")
trainId: 1
source: "Colombo Fort"
destination: "Kandy"
time: "14:00"
date: "20-05-2019"
capacity: 100
price: 150
__v: 0
```

```
_id: ObjectId("5ce11d129bdd253b044f2018")
trainId: 2
source: "Colombo Fort"
destination: "Batticalo"
time: "20:00"
date: "21-05-2019"
capacity: 100
price: 500
__v: 0
```

```
_id: ObjectId("5ce11d439bdd253b044f2019")
trainId: 3
source: "Colombo Fort"
destination: "Jaffna"
time: "4:00"
date: "22-05-2019"
capacity: 100
price: 600
__v: 0
```

```
_id: ObjectId("5ce11d859bdd253b044f201a")
trainId: 4
source: "Colombo Fort"
destination: "Anuradhapura"
time: "14:00"
date: "23-05-2019"
capacity: 100
price: 400
__v: 0
```

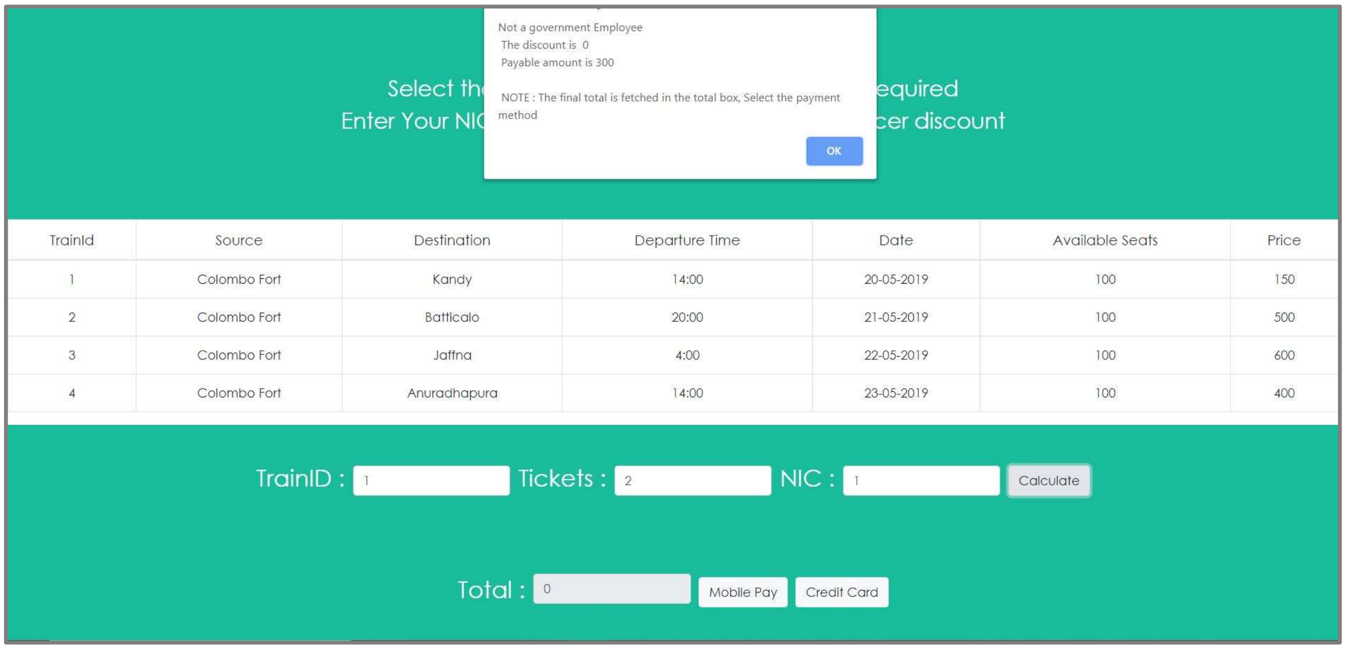
Figure 4-7 Train DB

Then the user should select the train details and depending whether user is a government employee or not the discount of 10 percent is given. Therefore, we have 2 scenarios. Dummy service is used to identify whether an user is a government employee or not (If NIC is even then government employee and if its odd the user is non – government employee)

Scenario 1 : User is a non – government employee and uses MobilePay option for payment.

Scenario 2 : User is a government employee and he chooses Credit Card option for payment.

4.1 Scenario 1


Not a government Employee
The discount is 0
Payable amount is 300

NOTE : The final total is fetched in the total box, Select the payment method

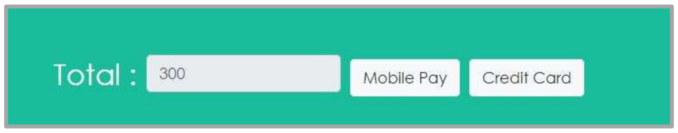
TrainId	Source	Destination	Departure Time	Date	Available Seats	Price
1	Colombo Fort	Kandy	14:00	20-05-2019	100	150
2	Colombo Fort	Batticalo	20:00	21-05-2019	100	500
3	Colombo Fort	Jaffna	4:00	22-05-2019	100	600
4	Colombo Fort	Anuradhapura	14:00	23-05-2019	100	400

TrainID : 1 Tickets : 2 NIC : 1

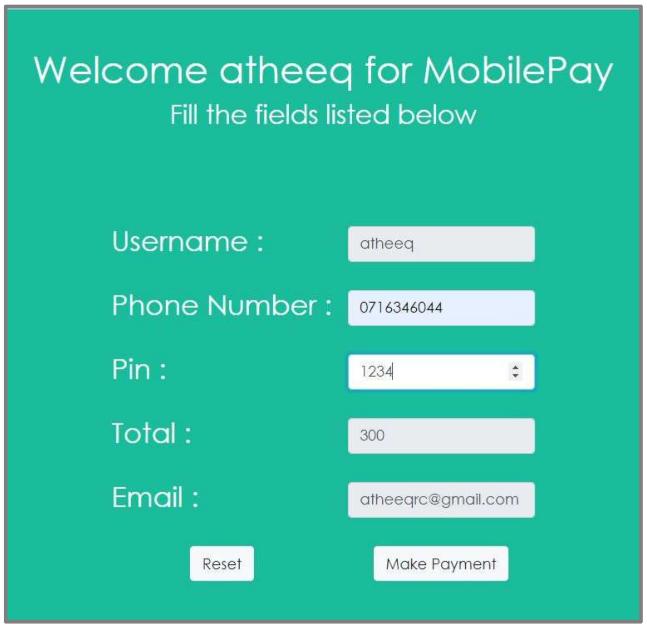
Total : 0

Figure 4-8 Government Employee Reservation

Initially the total is set to 0 and its disabled for editing. Train ID : 1, Tickets : 2 and NIC is 1 which is an odd number. Therefore, the discount isn't given. After pressing OK the total is fetched to the Total input box.


Total : 300

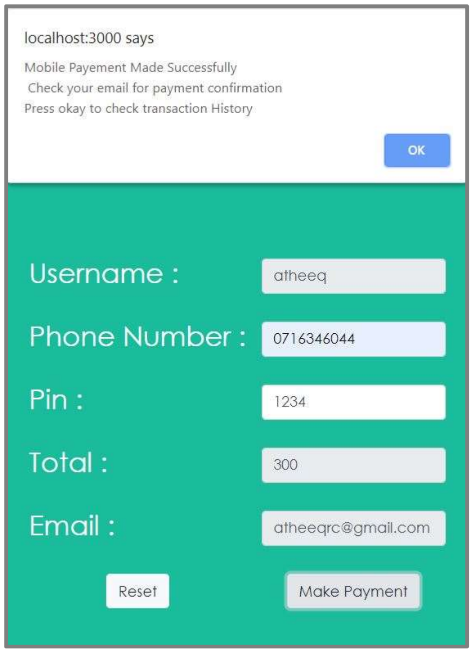
As per scenario the user has clicked the MobilePay option. The user should enter only the Phone Number and pin while others are automatically fetched from state object.



Welcome atheeq for MobilePay

Fill the fields listed below

Username :	atheeq
Phone Number :	0716346044
Pin :	1234
Total :	300
Email :	atheeqrc@gmail.com



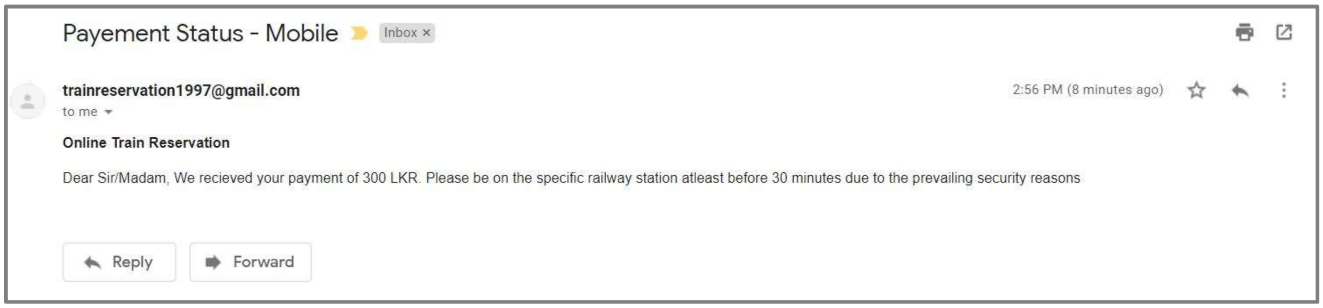
localhost:3000 says

Mobile Payement Made Successfully
Check your email for payment confirmation
Press okay to check transaction History

Username :	atheeq
Phone Number :	0716346044
Pin :	1234
Total :	300
Email :	atheeqrc@gmail.com

Figure 4-9 MobilePay Option

Then the user will receive an email, confirming their payment.



Payement Status - Mobile

trainreservation1997@gmail.com
to me

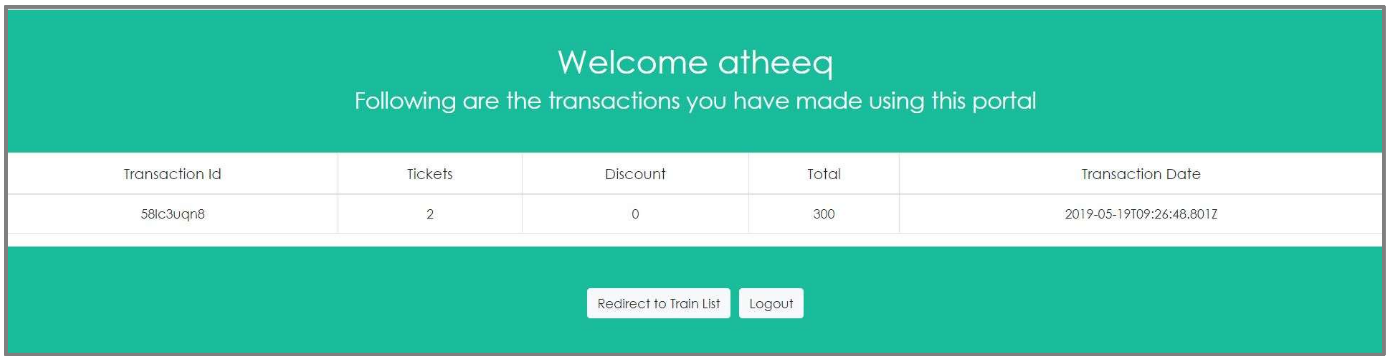
2:56 PM (8 minutes ago)

Online Train Reservation

Dear Sir/Madam, We recieved your payment of 300 LKR. Please be on the specific railway station atleast before 30 minutes due to the prevailing security reasons

Figure 4-10 MobilePay Confirmation Email

Then the user will be directed to transaction history page which displays all the transaction they made. (by using unique username). For the transaction id, ShortId module is used which generates random unique ids and this interface displays the Transaction Date and time as well.



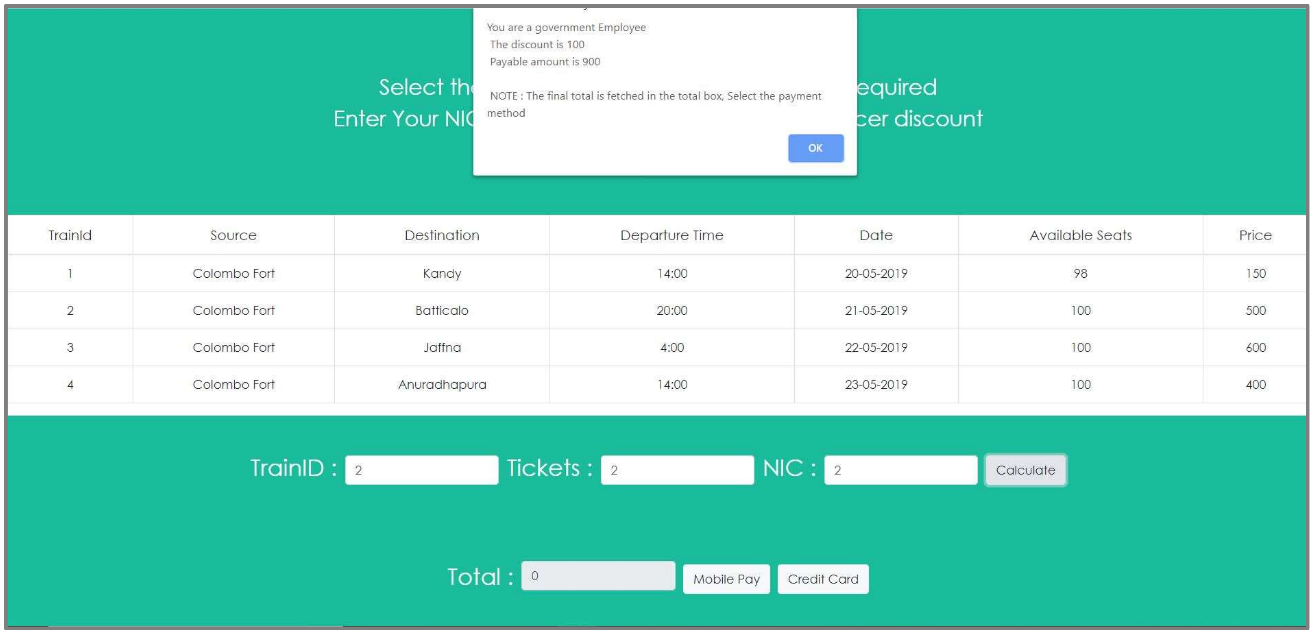
Welcome atheeq

Following are the transactions you have made using this portal

Transaction Id	Tickets	Discount	Total	Transaction Date
58lc3uqn8	2	0	300	2019-05-19T09:26:48.801Z

Figure 4-11 Government - Transaction History

4.2 Scenario 2



You are a government Employee
The discount is 100
Payable amount is 900

NOTE : The final total is fetched in the total box, Select the payment method

Select the … equired
Enter Your NIC … cer discount

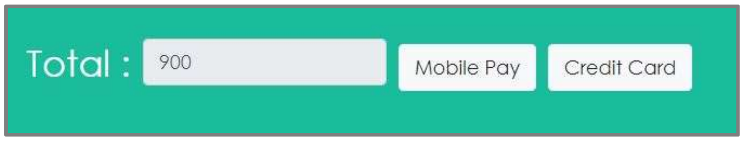
TrainId	Source	Destination	Departure Time	Date	Available Seats	Price
1	Colombo Fort	Kandy	14:00	20-05-2019	98	150
2	Colombo Fort	Batticalo	20:00	21-05-2019	100	500
3	Colombo Fort	Jaffna	4:00	22-05-2019	100	600
4	Colombo Fort	Anuradhapura	14:00	23-05-2019	100	400

TrainID : 2 Tickets : 2 NIC : 2

Total : 0

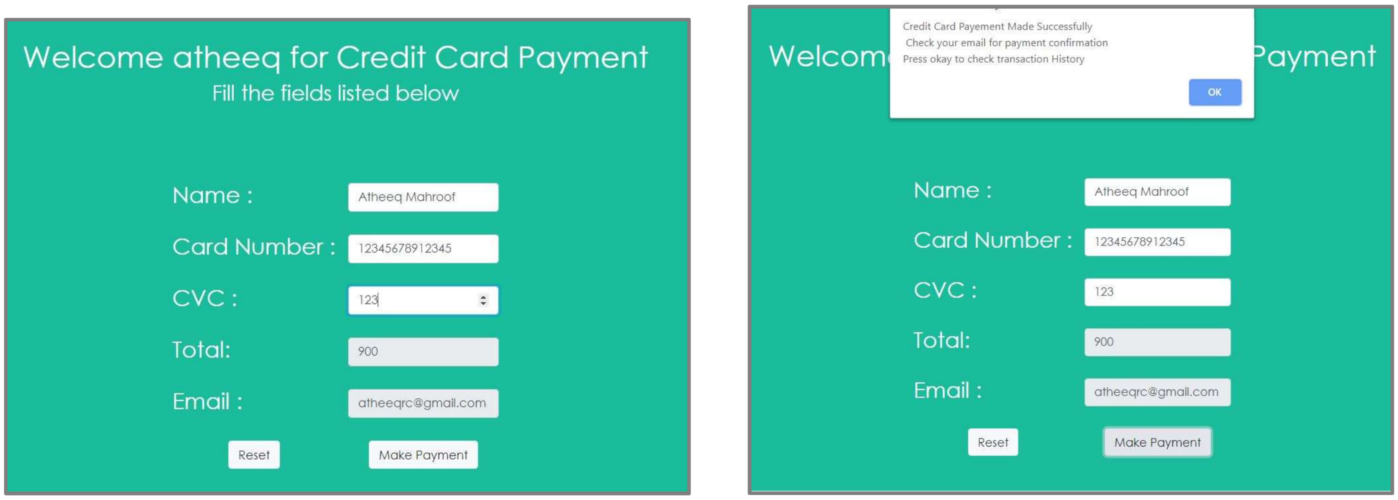
Figure 4-12 Non - Government Reservation

Now the NIC is even, therefore a 10% discount is given. And the total will be displayed in total input box.



Total : 900

As per the scenario, user will be selecting the credit card option.



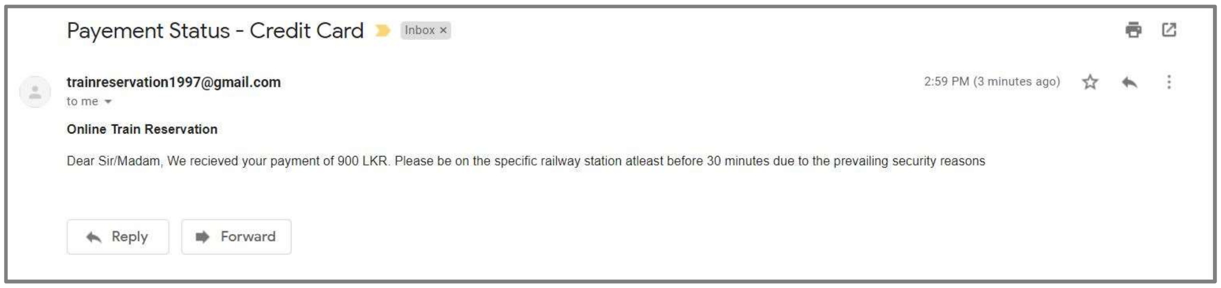
Welcome atheeq for Credit Card Payment
Fill the fields listed below

Name :	Atheeq Mahroof
Card Number :	12345678912345
CVC :	123
Total:	900
Email :	atheeqrc@gmail.com

Credit Card Payement Made Successfully
Check your email for payment confirmation
Press okay to check transaction History

Figure 4-13 Credit Card Payment

Then the user will receive an email, confirming their payment.



Payement Status - Credit Card

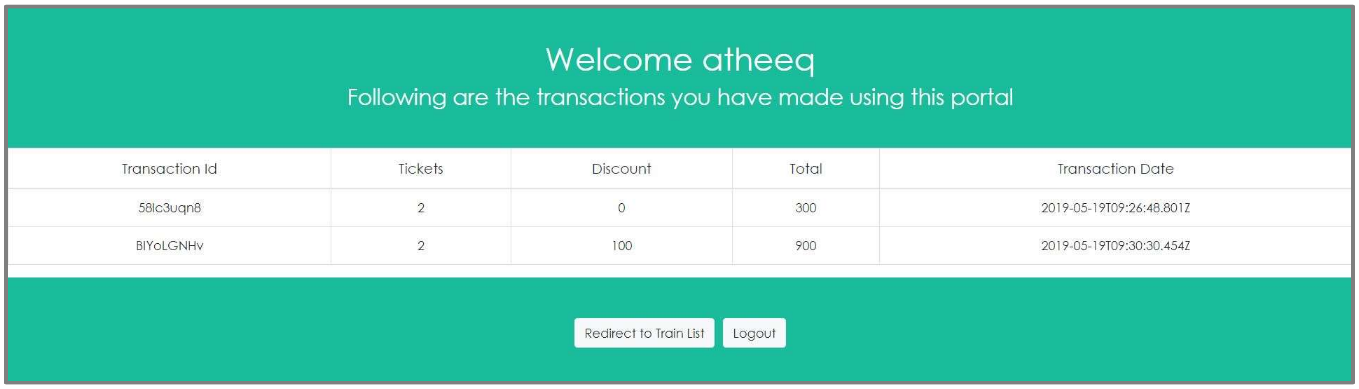
trainreservation1997@gmail.com
to me
2:59 PM (3 minutes ago)

Online Train Reservation

Dear Sir/Madam, We recieved your payment of 900 LKR. Please be on the specific railway station atleast before 30 minutes due to the prevailing security reasons

Figure 4-14 Credit Card Confirmation Email

Finally, the user is redirected to transactions history page which will show the transactions the user has made.



Welcome atheeq
Following are the transactions you have made using this portal

Transaction Id	Tickets	Discount	Total	Transaction Date
58lc3uqn8	2	0	300	2019-05-19T09:26:48.801Z
BIYoLGNHv	2	100	900	2019-05-19T09:30:30.454Z

Figure 4-15 Non-Government Transaction History

5 Appendix

5.1 Front End (Client)

5.1.1 Signup Component (SignUp.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import  Login from './Login';

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css'

class SignUp extends Component {

    constructor(props) {
        super(props);
    }

    signup = (event)=> {

        event.preventDefault();
        const firstname = this.refs.firstName.value;
        const lastname = this.refs.lastName.value;
        const username = this.refs.username.value;
        const password = this.refs.password.value;
        const email = this.refs.email.value;
        const address = this.refs.address.value;

        if(firstname ==='' || lastname ==='' ||username ===''
||password ==='' ||email ==='' ||address ==='' ){
            alert("One or more fields arent filled")
        }else {
            console.log("1")
           fetch ( 'http://localhost:5000/user/' + username, {
               method : 'GET',
               headers: {'Content-Type': 'application/json'}
           } ).then( res => {
               return res.json();
           }).then(data => {
               const user = JSON.stringify(data);
               console.log(user);
               console.log("2")
               if (user != '[]'){
                   console.log("3")
                   alert( "Username is already in use")
               } else {
                   const data = {"firstName" : firstname,
                                 "lastName" : lastname,
                                 "username" : username,
                                 "password" : password,
                                 "email" : email,
                                 "address": address}
                    console.log(data);
                   fetch("http://localhost:5000/user",{
```

```
                            method: 'POST',
                            body:JSON.stringify(data),
                            headers: {'Content-Type': 'application/json'}
                        } ).then(res => {
                            return res.json()
                        }).then(data => {
                            alert("Successfully Registered");
                            ReactDOM.render(<Login/>,
document.getElementById("root") )
                        }).catch(err => console.log(err))


                    }
                } ).catch(err => console.log(err))
            }

            console.log("4")

        }

        login = (event) => {

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

        render() {
            return (
                <div className="mt-5 " >
                    <b><h2 className="topDiv">Sign Up</h2></b>   <br/>
                     <form className="center"   >
                     <table cellPadding="10px">


                     <tbody>
                         <tr>
                             <td> First Name :  </td>
                             <td> <input  className="form-control"
placeholder= "First Name" type="text" ref="firstName"/></td>

                         </tr>
                         <tr>
                             <td> Last Name :  </td>
                             <td> <input className="form-control"
placeholder= "Last Name" type="text" ref="lastName"/></td>

                         </tr>
                         <tr>
                             <td> Username :  </td>
                             <td> <input className="form-control"
placeholder= "Username" type="text" ref="username"/></td>

                         </tr>
                         <tr>
                             <td> Password :  </td>
                             <td> <input className="form-control"
placeholder= "Password" type="password" ref="password"/></td>
```

```
                        </tr>
                        <tr>
                            <td> Email :  </td>
                            <td> <input className="form-control"
placeholder= "Email" type="email" ref="email"/></td>

                        </tr>
                        <tr>
                            <td> Address :  </td>
                            <td> <input className="form-control"
placeholder= "Address" type="text" ref="address"/></td>

                        </tr>

                        <tr>
                            <td> <button type="reset" className="btn
btn-light" > Reset</button>  </td>
                            <td> <button onClick={this.signup}
type="submit" className="btn btn-light" > Register</button> </td>
                        </tr>
                        <tr>
                            <td></td>

                                <td><button onClick={this.login}
className="btn btn-light centerPad" > Direct to Login
Page</button></td>

                        </tr>
                    </tbody>
                    </table>

                    </form>

                    <br/><br/><br/>
                </div>
            );
    }
}

export default SignUp;
```

5.1.2 Login Component (Login.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import SignUp from "./SignUp";
import TrainList from "./trainList";

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css'
```

```
class Login extends Component {
    constructor(props) {
        super(props);
    }

    login = (event)=> {
        event.preventDefault(event)

        const username = this.refs.username.value;
        const password = this.refs.password.value;

        if (username ==='' ||password ===''){
            alert("one or more fields are empty")
        }else {
            fetch("http://localhost:5000/user/" + username +
"/"+password, {
                method : "GET",
                headers: {'Content-Type': 'application/json'}
            }).then(req => {
                return  req.json();
            }).then(data => {
                let user = JSON.stringify(data);
                if(user =='[]'){
                    alert( "Incorrect username or password")
                } else {
                    let username_session
                    let email_session
                    for (let user of data ){
                      username_session = user.username.toString()
                        email_session = user.email.toString();
                    }
                    console.log(username_session)
                    ReactDOM.render(<TrainList username =
{username_session} email = {email_session}
/>,document.getElementById("root"));
                }
            }).catch(err => console.log(err))
        }

    }

    signup = ()=> {
        ReactDOM.render(<SignUp/>, document.getElementById('root'));
    }

    render() {
        return (
            <div className="mt-5 backdiv" >
                <b><h2  className="topDiv" >Login</h2></b>   <br/>
                <form className="center"   >
                    <table cellPadding="10px">


                        <tbody>
                        <tr>
                            <td> Username :  </td>
                            <td> <input className="form-control"
placeholder= "Username" type="text" ref="username"/></td>

                        </tr>
                        <tr>
                            <td> Password :  </td>
                            <td> <input className="form-control"
placeholder= "Password" type="password" ref="password"/></td>

                        </tr>

                        <tr>
                            <td> <button type="reset" className="btn
btn-light" > Reset</button>  </td>
                            <td> <button onClick={this.login}
type="Login" className="btn btn-light" > Login</button> </td>
                        </tr>
                        <tr>
                            <td></td>



                        </tr>
                        </tbody>
                    </table>
                </form>
                <button onClick={this.signup} style={{marginLeft :
560, }} className="btn btn-light centerPad"> Direct to SignUp
Page</button>
                <br/><br/><br/>
            </div>
        );
    }
}

export default Login;
```

5.1.3 TrainList Component (trainList.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import MobilePay from "./MobilePay";
import CreditCard from "./CreditCard"
import Login from "./Login";
import Transaction from "./Transaction"

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css'


class TrainList extends Component {
    constructor(props){
        super(props);
        this.state= {
            username:this.props.username,
            email:this.props.email,
            train :[],
            total : "",
            tickets: '',
            discount: ''
            }
    }

    CalculateTotal = (event) =>{
        event.preventDefault(event)


        const trainId = this.refs.trainId.value;
        const tickets = this.refs.tickets.value;
        const nic =this.refs.nic.value;

        if( trainId ==='' ||tickets==='' || nic ===''){
            alert("One or more feilds are not filled")
        }else {
            fetch('http://localhost:5000/train/' + trainId + "/" +
tickets +"/" + nic , {
                method : "GET",
                headers: {'Content-Type': 'application/json'}
            }).then( req => {
                return req.json()
            }).then(data => {
                console.log(data.total)
                this.setState({total : data.total, tickets :
data.tickets, discount : data.discount})
                alert(`${data.status} \n The discount is
${data.discount} \n Payable amount is ${data.total}
                \n NOTE : The final total is fetched in the total
box, Select the payment method`)
                console.log(this.state.total)
            })

        this.updateTable();




        }
    }

    updateTable = ()=>{
        fetch('http://localhost:5000/train').then(req => {
            return req.json()
        }).then( data =>{
            this.setState({train: data})
        } )
    }
```

```
   async componentDidMount() {
        const url = "http://localhost:5000/train"
        const response = await fetch(url)
       const data = await response.json();
            this.setState({train: data, total : 0.00})
       console.log(this.state.train)

    }

    MobilePay = (event) =>{
        event.preventDefault()
        ReactDOM.render(<MobilePay  total={this.state.total} email={
this.state.email} username = {this.state.username} tickets =
{this.state.tickets} discount=  {this.state.discount}/>,
document.getElementById("root"))
    }
    CreditCard = (event) => {
        event.preventDefault()
        ReactDOM.render(<CreditCard  total={this.state.total}
email={ this.state.email} username ={this.state.username} tickets =
{this.state.tickets} discount=  {this.state.discount} />,
document.getElementById("root"))

    }
    logout = (e) =>{
        e.preventDefault()
        ReactDOM.render(<Login/>, document.getElementById('root'));

    }

    toHistory = (e) =>{
        e.preventDefault()
        ReactDOM.render(<Transaction username ={this.state.username}
email = {this.state.email}  />, document.getElementById('root'));

    }


    render() {
        return (
            <div>
                <div className="header">
                    <h1>Welcome  {this.state.username} </h1>
                    <h3>Select the trainId and the Number of Tickets
Required</h3>
                    <h3>Enter Your NIC number to get the government
officer discount</h3>
                    <p> <button onClick={this.toHistory}
className="btn btn-light">Transaction History</button> {"     "}
                        <button onClick={this.logout} className="btn
btn-light"> Logout </button>
                    </p>
                </div>
```

```
                    <table className="table table-bordered tableAlign">
                        <thead>
                        <tr className="boldHead">
                            <td>TrainId</td>
                            <td>Source</td>
                            <td>Destination</td>
                            <td>Departure Time</td>
                            <td>Date</td>
                            <td>Available Seats</td>
                            <td>Price</td>

                        </tr>
                        </thead>

                        <tbody>
                        {
                            this.state.train.map( (train) =>{
                                return [
                                    <tr>
                                        <td
className="">{train.trainId}</td>
                                        <td>{train.source}</td>
                                        <td>{train.destination}</td>
                                        <td>{train.time}</td>
                                        <td>{train.date}</td>
                                        <td>{train.capacity}</td>
                                        <td>{train.price}</td>

                                    </tr>
                                ]
                            })
                        }

                        </tbody>

                    </table>
                    <form>
                        <table className="center" cellPadding="5px">
                            <tr>
                                <td><label>TrainID :</label> </td>
                                <td><input className="form-control"
ref="trainId"/></td>

                                <td><label>Tickets : </label> </td>
                                <td><input className="form-control"
ref="tickets"/></td>

                                <td><label>NIC :</label>  </td>
                                <td><input className="form-control" ref
="nic"/></td>

                            </tr>

                            <tr>
                                <td><button
onClick={this.CalculateTotal} className="btn btn-light"> Calculate
</button></td>
                            </tr>
                    </table>
                   <table className="center" cellPadding="5px"
cellSpacing="10px">
                       <tr>
                           <td colSpan="2" >Total :</td>
                           <td colSpan="3" ><input className="form-
control" disabled={true} value={this.state.total} /></td>
                           <td colSpan="4" ><button
onClick={this.MobilePay} className="btn btn-light"> Mobile Pay
</button> </td>
                           <td colSpan="5" ><button
onClick={this.CreditCard} className="btn btn-light"> Credit Card
</button></td>
                       </tr>
                   </table>

                 </form>
             </div>
        );
    }
}

export default TrainList;
```

5.1.4 MobilePay Component (MobilePay.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import Transaction from "./Transaction"

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css';

class MobilePay extends Component {

    constructor(props){
        super(props);
        this.state = {
            total : this.props.total,
            email: this.props.email,
            username :this.props.username,
            discount :this.props.discount,
            tickets:this.props.tickets
        }
    }

    payment= (e)=> {
        e.preventDefault()

        const username  = this.state.username;
        const phoneNumber = this.refs.phone.value;
        const pin = this.refs.pin.value;
        const total = this.state.total;
        const email = this.state.email;


        if (phoneNumber === "" || pin ===""){
            alert ("One or more fields are empty")
        } else {
            const data = {
                "username" : username,
                "phoneNumber": phoneNumber,
                "pin": pin,
                "total": total,
                "email": email
            }
            console.log(data)

            fetch("http://localhost:5000/mobilePay", {
                method : "POST",
                body:JSON.stringify(data),
                headers: {'Content-Type': 'application/json'}
            }).then(res =>{
                return res.json()
            }). then( data  => {
                alert ("Mobile Payement Made Successfully \n Check
your email for payment confirmation \nPress okay to check
transaction History ")

                const transData = {
                    "username": this.state.username,
                    "tickets" : this.state.tickets,
                    "discount": this.state.discount,
                    "total" : this.state.total
                }

                console.log(transData)

                fetch('http://localhost:5000/user/transaction' , {
                    method : "POST",
                    body:JSON.stringify(transData),
                    headers: {'Content-Type': 'application/json'}
                }).then( res => {
                    return res.json()
                }).then( data => {
                    console.log( data + " Transaction added")
                }).catch (err => {
                        console.log( err)
                } )


                ReactDOM.render(<Transaction username =
{this.state.username} email
={this.state.email}/>,document.getElementById("root") );
            }).catch(err => console.log(err))
        }

    }
    render() {
        return (
            <div>
                <div className="header">
                    <h1>Welcome  {this.state.username} for MobilePay
</h1>
                    <h3>Fill the fields listed below</h3>

                </div>
            <form className="center">
                <table cellPadding="8px">
                    <tbody>
                    <tr>
                        <td style={{textAlign : "left"}}> <label >
Username : </label></td>
                        <td><input className="form-control"
disabled={true} value={this.state.username} /></td>
                    </tr>
                    <tr>
                        <td style={{textAlign :
"left"}}><label>Phone Number :</label></td>
                        <td><input  className="form-control"
placeholder= "Phone Number" type="number" ref="phone"/></td>
                    </tr>
                    <tr>
                        <td style={{textAlign : "left"}}><label>Pin
:</label></td>
                        <td><input  className="form-control"
placeholder= "pin" type="number" ref="pin"/></td>
                    </tr>
                    <tr>
                        <td style={{textAlign :
"left"}}><label>Total :</label></td>
                        <td><input className="form-control"
disabled={true} value={this.state.total} /></td>
                    </tr>
                    <tr>
                        <td style={{textAlign :
"left"}}><label>Email :</label></td>
                        <td ><input className="form-control"
disabled={true} value={this.state.email} /></td>
                    </tr>
                    <tr>
                        <td> <button type="reset" className="btn
btn-light" > Reset</button>  </td>
                        <td> <button onClick={this.payment}
type="submit" className="btn btn-light" > Make Payment</button>
</td>
                    </tr>
                    </tbody>
                </table>
            </form>

            </div>
        );
    }
}

export default MobilePay;
```

5.1.5 Credit Card Component (CreditCard.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import Transaction from "./Transaction";

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css'



class CreditCard extends Component {

    constructor(props){
        super(props);
        this.state = {
            total : this.props.total,
            email: this.props.email,
            username :this.props.username,
            discount :this.props.discount,
            tickets:this.props.tickets
        }
    }

    payment= (e)=> {
        e.preventDefault()
        const name  = this.refs.name.value;
        const cardNumber = this.refs.cardNumber.value;
        const cvc = this.refs.cvc.value;
        const total = this.state.total;
        const email = this.state.email;

        if (name === "" || cardNumber ==="" || cvc === ""){
            alert ("One or more fields are empty")
        } else {
            const data = {
                "name" : name,
                "cardNumber" : cardNumber,
                "cvc" : cvc,
                "total" : total,
                "email" : email
            }
            console.log(data)

            fetch("http://localhost:5000/creditCard", {
                method : "POST",
                body:JSON.stringify(data),
                headers: {'Content-Type': 'application/json'}
            }).then(res =>{
                return res.json()
            }). then( data  => {
                alert ("Credit Card Payement Made Successfully \n
Check your email for payment confirmation \nPress okay to check
transaction History ")

                const transData = {
                    username: this.state.username,
                    tickets : this.state.tickets,
                    discount: this.state.discount,
                    total : this.state.total
                }

                fetch('http://localhost:5000/user/transaction' , {
                    method : "POST",
                    body:JSON.stringify(transData),
                    headers: {'Content-Type': 'application/json'}
                }).then( res => {
                    return res.json()
                }).then( data => {
                    console.log( data + " Transaction added")
                }).catch (err => {

                } )



                ReactDOM.render(<Transaction username =
{this.state.username} email
={this.state.email}/>,document.getElementById("root") );
            }).catch(err => console.log(err))
        }

    }
    render() {
        return (

            <div>
                <div className="header">
                    <h1>Welcome  {this.state.username} for Credit
Card Payment </h1>
                    <h3>Fill the fields listed below</h3>

                </div>
            <form className="center">
                <table cellPadding="8px">
                    <tbody>
                    <tr>
                        <td style={{textAlign : "left"}}><label>Name
: </label></td>
                        <td><input  className="form-control"
placeholder= "Card name" type="text" ref="name"/></td>
                    </tr>
```

```
                        <tr>
                            <td style={{textAlign : "left"}}><label>Card
Number : </label></td>
                            <td><input  className="form-control"
placeholder= "Card Number" type="number" ref="cardNumber"/></td>
                        </tr>
                        <tr>
                            <td style={{textAlign : "left"}}><label>CVC
: </label></td>
                            <td><input  className="form-control"
placeholder= "CVC" type="number" ref="cvc"/></td>
                        </tr>
                        <tr>
                            <td style={{textAlign :
"left"}}><label>Total:</label></td>
                            <td><input className="form-control"
disabled={true} value={this.state.total} /></td>
                        </tr>
                        <tr>
                            <td style={{textAlign :
"left"}}><label>Email : </label></td>
                            <td><input className="form-control"
disabled={true} value={this.state.email} /></td>
                        </tr>
                        <tr>
                            <td> <button type="reset" className="btn
btn-light" > Reset</button>  </td>
                            <td> <button onClick={this.payment}
type="submit" className="btn btn-light" > Make Payment</button>
</td>
                        </tr>
                        </tbody>
                    </table>
                </form>



                </div>
        );
    }
}

export default CreditCard;
```

5.1.6 Transaction History Component (Transaction.js)

```
import React, {Component} from 'react';
import ReactDOM from "react-dom";

import 'bootstrap/dist/css/bootstrap.min.css';
import './common.css'

import Login from "./Login";
import TrainList from './trainList';
```

```
class Transaction extends Component {

    constructor (props) {
        super(props);
        this.state = {
            username : this.props.username,
            email : this.props.email,
            transaction : []
        }
    }

    logout = (e) =>{
        e.preventDefault()
        ReactDOM.render(<Login/>, document.getElementById('root'));

    }

    train = (event) => {
        event.preventDefault()
        ReactDOM.render(<TrainList  username={ this.state.username}
email = {this.state.email}  />, document.getElementById('root'));
    }

    async componentDidMount() {
        const url = "http://localhost:5000/mobilePay/" +
this.state.username
        const response = await fetch(url)
        const data = await response.json();
        this.setState({transaction: data})
        console.log(this.state.transaction)

    }


    render() {
        return (
            <div>


                <div className="header">
                    <h1>Welcome  {this.state.username} </h1>
                    <h3>Following are the transactions you have made
using this portal </h3>

                </div>


                <table className="table table-bordered tableAlign">
                    <thead>
                    <tr className="boldHead">
                        <td>Transaction Id</td>
                        <td>Tickets</td>
                        <td>Discount</td>
                        <td>Total</td>
                        <td>Transaction Date</td>

                    </tr>
                    </thead>

                    <tbody>
                    {
                        this.state.transaction.map( (transaction)
=>{

                            return [
                                <tr>
                                    <td >{transaction.transId}</td>
                                    <td>{transaction.tickets}</td>
                                    <td>{transaction.discount}</td>
                                    <td>{transaction.total}</td>
                                    <td>{transaction.Date}</td>


                                </tr>
                            ]
                        })
                    }

                    </tbody>

                </table>

                <table className="center" cellPadding="5px"
cellSpacing="10px">
                    <tr>
                       <td colSpan="4" ><button onClick={this.train}
className="btn btn-light"> Redirect to Train List </button> </td>
                        <td colSpan="5" ><button
onClick={this.logout} className="btn btn-light"> Logout
</button></td>
                    </tr>
                </table>


            </div>
        );
    }
}

export default Transaction;
```

5.1.7 Package.json

```json
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "bootstrap": "^4.3.1",
    "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "react-scripts": "3.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

5.2 Backend

5.2.1 Index.js

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const cors = require('cors');
const routes = require('./routes/mainRoute')
const  PORT = 5000;


app.use(bodyParser.urlencoded({extended: false}));
app.use(bodyParser.json());
app.use(cors());
app.use((req,res,next)=>{
    res.setHeader('Access-Control-Allow-Origin','*');
    res.setHeader('Access-Control-Allow-Methods','GET, POST,
OPTIONS, PUT, PATCH, DELETE');
    // res.setHeader('Access-Control-Allow-Headers','X-Requested-
With,content-type');
    res.setHeader('Access-Control-Allow-Credentials',true);
    next();
})

app.use('/', routes);

app.listen(PORT, ()=> {
    console.log(`Listening to  port ${PORT} `)})
```

5.2.2 DBSchema.js

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

const schema = mongoose.Schema;

        const user = new schema({
            username : {
                type: String,
                required: true

            },
            password : {
                type: String,
                required: true

            },
            firstName : {
                type : String,
                required: true
            },
            lastName : {
                type : String,
                required: true
            },
            email : {
```

```
            type : String,
            required: true
        },
        address : {
            type : String,
            required: true
        }
    })

    const train = new schema({
        trainId : {
            type : Number,
            required: true
        },
        source : {
            type : String,
            required: true
        },
        destination : {
            type : String,
            required: true
        },
        time : {
            type : String,
            required: true
        },
        date : {
            type : String,
            required: true
        },
        capacity : {
            type : Number,
            required : true
        },
        price : {
            type : Number,
            required : true
        }

    })

    const creditCard = new schema ({
        name : {
            type : String,
            required: true
        },
        cardNumber : {
            type : Number,
            required: true
        },
        cvc : {
            type : Number,
            required: true
        },
        total: {
            type : Number,
            required: true
        },
        email : {
            type : String,
            required: true
        }
})

const mobilePay = new schema ({
        username : {
            type : String,
            required: true
        },
        phoneNumber : {
            type : Number,
            required: true
        },
        pin : {
            type : Number,
            required: true
        },
        total: {
            type : Number,
            required: true
        },
        email : {
            type : String,
            required: true
        }
})

const transaction = new schema ({
        transId : {
            type : String,
            required : true

        },
        username : {
            type : String,
            required : true
        },
        tickets : {
            type : Number,
            required : true
        },
        discount : {
            type : Number,
            required : true
        },
        total : {
            type : Number,
            required : true
        },
        Date : {
            type : Date,
            required : true
            }

        })

    mongoose.model('user',user);
    mongoose.model('train',train);
    mongoose.model('creditCard',creditCard);
    mongoose.model('mobilePay',mobilePay);
    mongoose.model('transaction', transaction);

    mongoose.connect('mongodb://atheeq:atheeq@mernshopping-shard-00-
00-fbkn6.mongodb.net:27017,mernshopping-shard-00-01-
fbkn6.mongodb.net:27017,mernshopping-shard-00-02-
fbkn6.mongodb.net:27017/test?ssl=true&replicaSet=MernShopping-shard-
0&authSource=admin&retryWrites=true',{ useNewUrlParser: true })
        .then(
    ()=> console.log("DB Connected"),
    error => console.log(error)
    )


module.exports = mongoose;
```

5.2.3 Main Routes (mainRoute.js)

```
const express=require('express');
const router=express.Router();

const userRouter = require("./userRoutes");
const trainRouter = require("./trainRoutes");
const creditCardRouter = require("./creditCardRoutes");
const mobilePayRouter = require("./mobilePayRoutes");


router.use ('/user',userRouter);
router.use ('/train',trainRouter);
router.use ('/creditCard',creditCardRouter);
router.use ('/mobilePay',mobilePayRouter);


module.exports = router;
```

5.2.4 User Routes (userRoutes.js)

```
const express=require('express');
const router=express.Router();
const shortid = require('shortid');

const mongoose = require('../DBSchema');
const userSchema = mongoose.model('user');
const transSchema = mongoose.model('transaction')
```

```
// localhost/user POST
    router.post('/', (req,res) => {
        const newUser =  userSchema ({
                username : req.body.username,
                password :  req.body.password,
                firstName : req.body.firstName,
                lastName : req.body.lastName,
                address: req.body.address,
                email : req.body.email
        })
        console.log(newUser);
        newUser.save().then( (user) => {
            res.status(200).send({Message : `Successfully Added the
user` })
            console.log (user)
        }).catch((err) => {
            res.status(400).send({Message : `Error occured :
${err}`})
        })
    })

// localhost/user/email GET

    router.get ('/:username', (req,res) => {
        userSchema.find ( {username :
req.params.username}).exec().then((user)=> {

            console.log(req.url + req.params.username)
            res.status(200).send(user)
        }).catch(err => {
            res.status(400).send({Message : `Error occured :
${err}`})
        })
    })

// localhost/username/password
    router.get ('/:username/:password', (req,res) => {
        userSchema.find ( {username : req.params.username, password
: req.params.password}).exec().then((user)=> {

            console.log(req.url)

            res.status(200).send(user)
        }).catch(err => {
            res.status(400).send({Message : `Error occured :
${err}`})
        })
    })


//localhost/transaction/

router.post('/transaction', (req,res) => {
        const  newTransaction = transSchema ({
            username : req.body.username,
            tickets : req.body.tickets,
            discount : req.body.discount,
            total : req.body.total,
            Date: Date.now(),
            transId:  shortid.generate()
        })

    newTransaction.save().then( (transaction) => {
        res.status(200).send({Message : `Successfully Transaction
added` })
        console.log (transaction)
    }).catch((err) => {
        res.status(400).send({Message : `Error occured : ${err}`})
    })
})



module.exports = router;
```

5.2.5 Train Routes (trainRoutes.js)

```
const express=require('express');
const router=express.Router();

const mongoose = require('../DBSchema');
const trainSchema = mongoose.model('train');


// localhost/train POST
    router.post ('/', (req,res) => {
        const newTrain = trainSchema ( {
                trainId : req.body.trainId,
                source : req.body.source,
                destination : req.body.destination,
                time :  req.body.time,
                date: req.body.date,
                capacity : req.body.capacity,
                price : req.body.price
        })
        newTrain.save().then( train => {
            res.status(200).send({Message : "Train record is
added"});
            console.log(train);
        }).catch( err => {
            res.status(400).send({Message : `Error occured :
${err}`})
        })
    } )

// localhost/train  GET

    router.get ('/', (req,res) => {
        trainSchema.find().then( trains => {
            res.status(200).send(trains);
            console.log (trains)
        }).catch( err => {
            res.status(400).send({Message : `Error occured :
${err}`})
        })
    })

    router.get ('/:trainId/:noTicket/:nic', (req,res) => {
        trainSchema.find( {trainId:
req.params.trainId}).exec().then( (train) => {

            console.log(train[0].price)
            let discount = ' 0';
            let status = 'Not a government Employee'

            let newTotal = train[0].price * req.params.noTicket;
            let newCapacity = train[0].capacity -
req.params.noTicket;
            trainSchema.update ({trainId : req.params.trainId},
{$set : {capacity : newCapacity }}).catch(err => console.log(`Error
is ${err}`) )


            if (req.params.nic % 2 == 0) {
                discount = newTotal * 0.1
                status = 'You are a government Employee'
                newTotal = newTotal * 0.9 // 10 percent discount
            }
            res.status(200).send({total : newTotal,discount:
discount,status: status, tickets: req.params.noTicket })
        }).catch( err => {
            console.log(err)
        })
    })


module.exports = router;
```

5.2.6 MobilePay Routes (mobilePay.js)

```
const express=require('express');
const router=express.Router();
const nodemailer = require('nodemailer');
//npm install nodemailer@4.7.0
const mongoose = require('../DBSchema');
const mobilePaySchema = mongoose.model('mobilePay');
const transSchema = mongoose.model('transaction')


//localhost/mobilePay  POST
    router.post("/", (req,res) => {
        const  newMobilePay = mobilePaySchema ({
            username : req.body.username,
            phoneNumber: req.body.phoneNumber,
            pin: req.body.pin,
            total: req.body.total,
            email: req.body.email
        })

        newMobilePay.save().then( mobilePay => {
            res.status(200).send({Message : "Successfully Payement 
Made"});
            console.log(mobilePay)
        }).catch(error => {
            res.status(400).send({Message : `Error occured : 
${error}`})
        })

        const output=` <b>Online Train Reservation</b> <p>Dear 
Sir/Madam, We recieved your payment of ${req.body.total}
                        LKR. Please be on the specific railway 
station atleast before 30 minutes due to the prevailing security 
reasons</p> `;

        let transporter = nodemailer.createTransport ({

            service: 'Gmail',

            auth : {
                user : 'trainreservation1997@gmail.com',
                pass : 'trainreservation'
            },
            tls:{ rejectUnauthorized:false }
        })



        let mailOptions = {
            from : 'trainreservation1997@gmail.com',
            to: req.body.email,
            subject : "Payement Status - Mobile",
            html : output
        }

        transporter.sendMail(mailOptions, (err, info) =>{
            if (err) {
                console.log("error")
                return console.log(err)

            }
            console.log("no error")
            console.log(`Message sent : ${info.messageId} `)
            console.log (`Preview URL ${nodemailer.getTestMessageUrl
(info)}`)
        })
    })

router.get('/:username', (req,res) => {
    transSchema.find ( {username :
```

```
req.params.username}).exec().then((transaction)=> {

        console.log(req.url + req.params.username)
        console.log(transaction)
        res.status(200).send(transaction)
    }).catch(err => {
        res.status(400).send({Message : `Error occured : ${err}`})
    })
})


module.exports = router;
```

5.2.7 Credit Card Routes (creditCard.js)

```
const express=require('express');
const router=express.Router();
const nodemailer = require('nodemailer');

const mongoose = require('../DBSchema');
const creditCardSchema = mongoose.model('creditCard');


// localhost/creditCard  POST
    router.post ('/', (req,res) => {
        const newCreditCard = creditCardSchema({
            name : req.body.name,
            cardNumber : req.body.cardNumber,
            cvc : req.body.cvc,
            total : req.body.total,
            email : req.body.email
        })

        newCreditCard.save().then( creditCard => {
            res.status(200).send({Message : "Successfully Payement
Made"});
            console.log(mobilePay)

        }).catch(error => {
            res.status(400).send({Message : `Error occured :
${error}`})
        } )

        const output=` <b>Online Train Reservation</b> <p>Dear
Sir/Madam, We recieved your payment of ${req.body.total}
                        LKR. Please be on the specific railway
station atleast before 30 minutes due to the prevailing security
reasons</p> `;

        let transporter = nodemailer.createTransport ({

            service: 'Gmail',

            auth : {
                user : 'trainreservation1997@gmail.com',
                pass : 'trainreservation'
            },
            tls:{ rejectUnauthorized:false }
        })



        let mailOptions = {
            from : 'trainreservation1997@gmail.com',
            to: req.body.email,
            subject : "Payement Status - Credit Card",
            html : output
        }

        transporter.sendMail(mailOptions, (err, info) =>{
            if (err) {
                console.log("error")
                return console.log(err)

            }
            console.log("no error")
            console.log(`Message sent : ${info.messageId} `)
            console.log (`Preview URL ${nodemailer.getTestMessageUrl
(info)}`)
        })

    })




module.exports = router;
```

5.2.8 Package.json

```json
{
  "name": "TrainReservationSystem",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "express": "^4.17.0",
    "mongoose": "^5.5.9",
    "node-mailer": "^0.1.1",
    "nodemailer": "^4.7.0",
    "shortid": "^2.2.14"
  }
}
```

NOTE: All the backend implementations are made inside the specific JavaScript files.