



Sri Lanka Institute of Information Technology

Distributed Systems (SE3020)

Assignment 02

Online Train Ticket Reservation System

Assignment Report

Submitted By: IT17134668(Gamage V.S)

Table of Contents

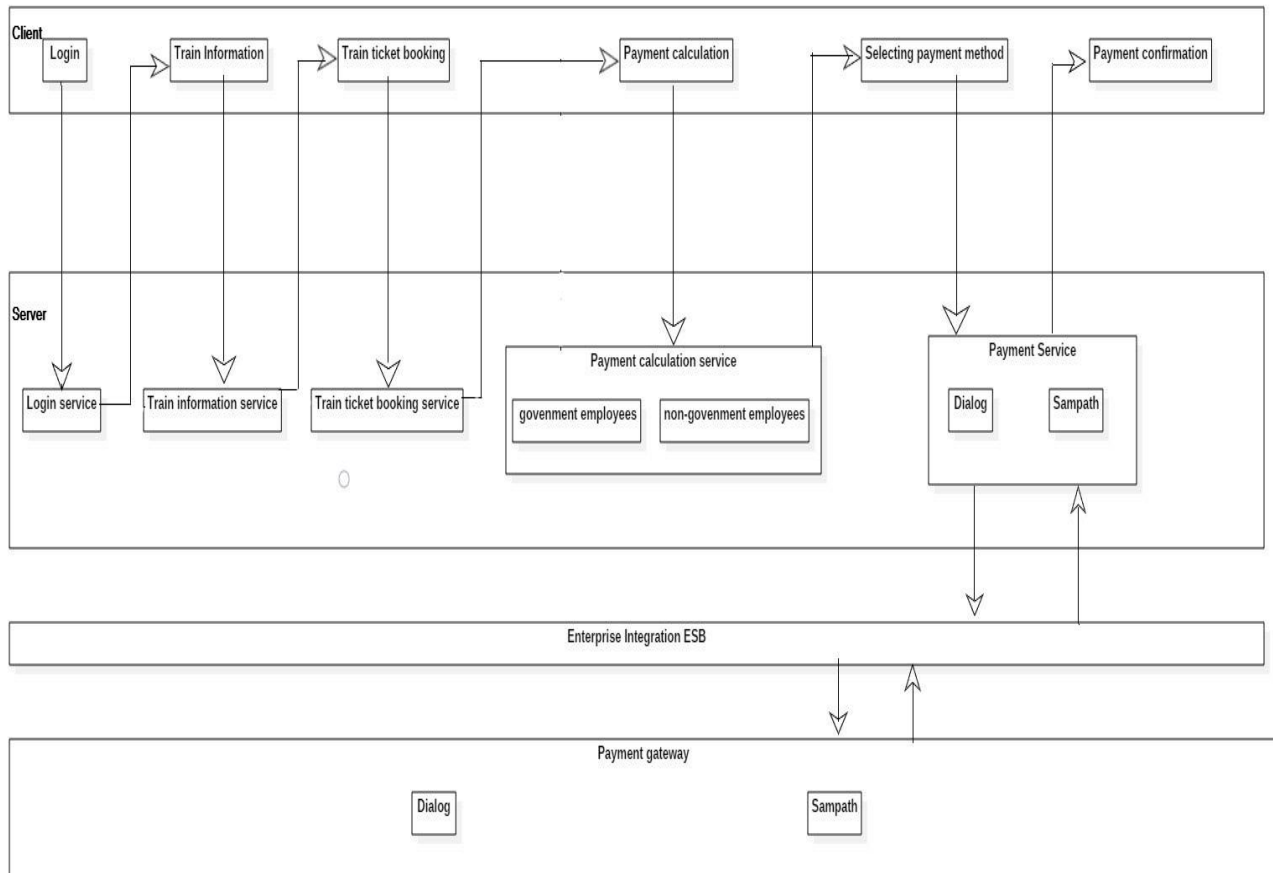
Introduction	3
High Level Diagram	4
Workflow Diagram	5
System Workflow Scenario Execution	6
Scenario	6
Authentication and Security	12
File Hierarchy	18
Appendix	21
Front End	21
Back End	39

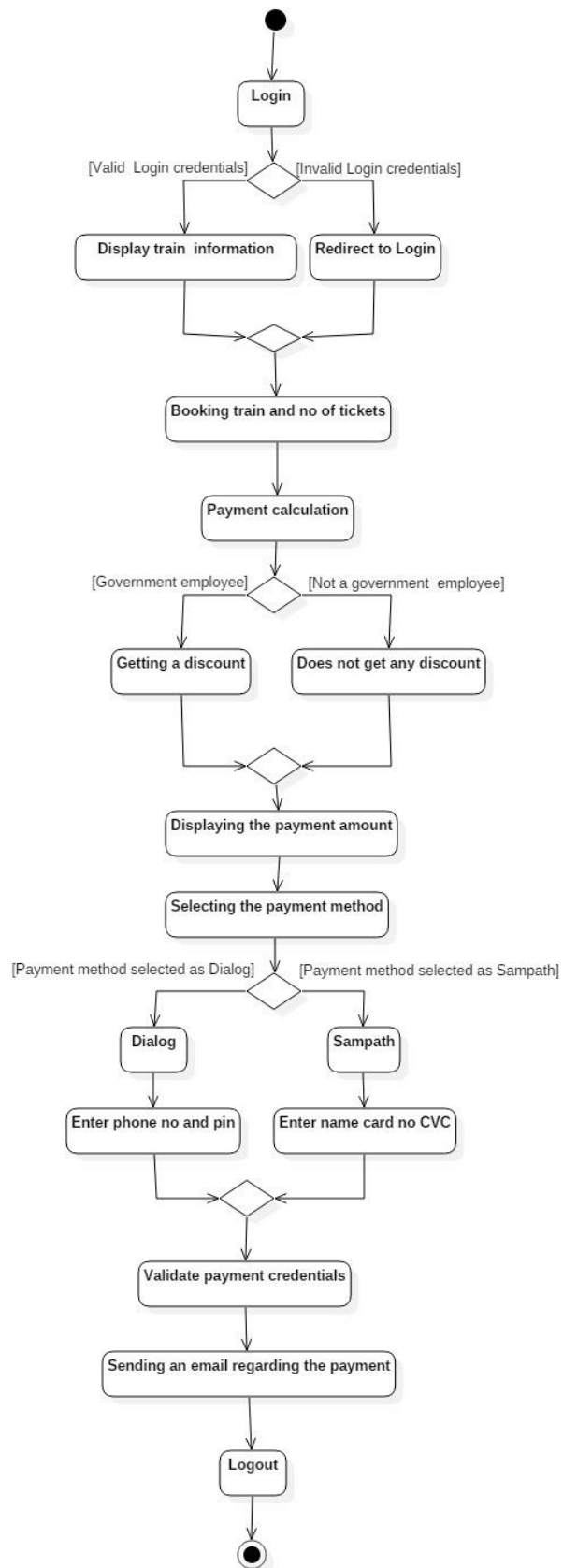
Introduction

The “Online Train Ticket Booking” web application is implemented using ReactJS for the front-end development, NodeJS, ExpressJS, Mongoose for the back-end development and MongoDB as the database. ‘Nodemailer’ module is used to send E-mail to the user once a successful train ticket booking is done.

The user must login to use the service of the application. After successful login the user can book a train ticket by looking at the train information delivered by the application. The user must provide the NIC number to identify the employee status (government or non-government), therefore if the employee status is ‘government’ the application will provide a 10% discount to the user. Once the payment is calculated the ticket/s can be purchased using a credit card or a dialog number and when the purchase is complete the user will get an E-mail regarding the ticket booking.

High Level Diagram






System Workflow Scenario Execution

Scenario

Customer who needs to reserve a train ticket, visits the website and creates an account by giving Full name, E-mail (Username), National Identity Card Number (NIC), Employee Status (Private/Government), Password accordingly as given bellow.



Reserve Your Train Ticket Today

Vinu Gamage
sahanigamage@gmail.com
985011214V
Private

Sign Up

Fig. 1.0

Once the user has successfully created the user account, the system redirects the user to the login page.

User Login

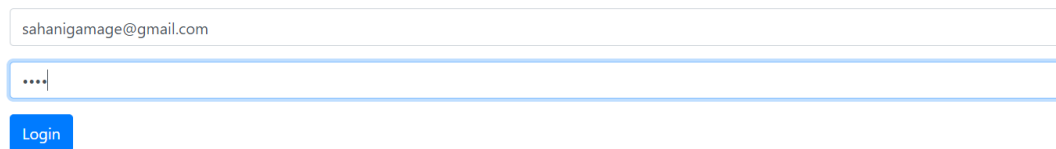
A user login form with two input fields and a button. The first input field contains the email address 'sahanigamage@gmail.com'. The second input field contains four dots, indicating a password. Below the second input field is a blue button labeled 'Login'.

Fig. 2.0

Once a successfully login is done the users account will be displayed with his Full name, E-mail, National Identity Card Number (NIC), Employee Status (Private, Government).

Welcome To Your Profile Vinu Gamage!!

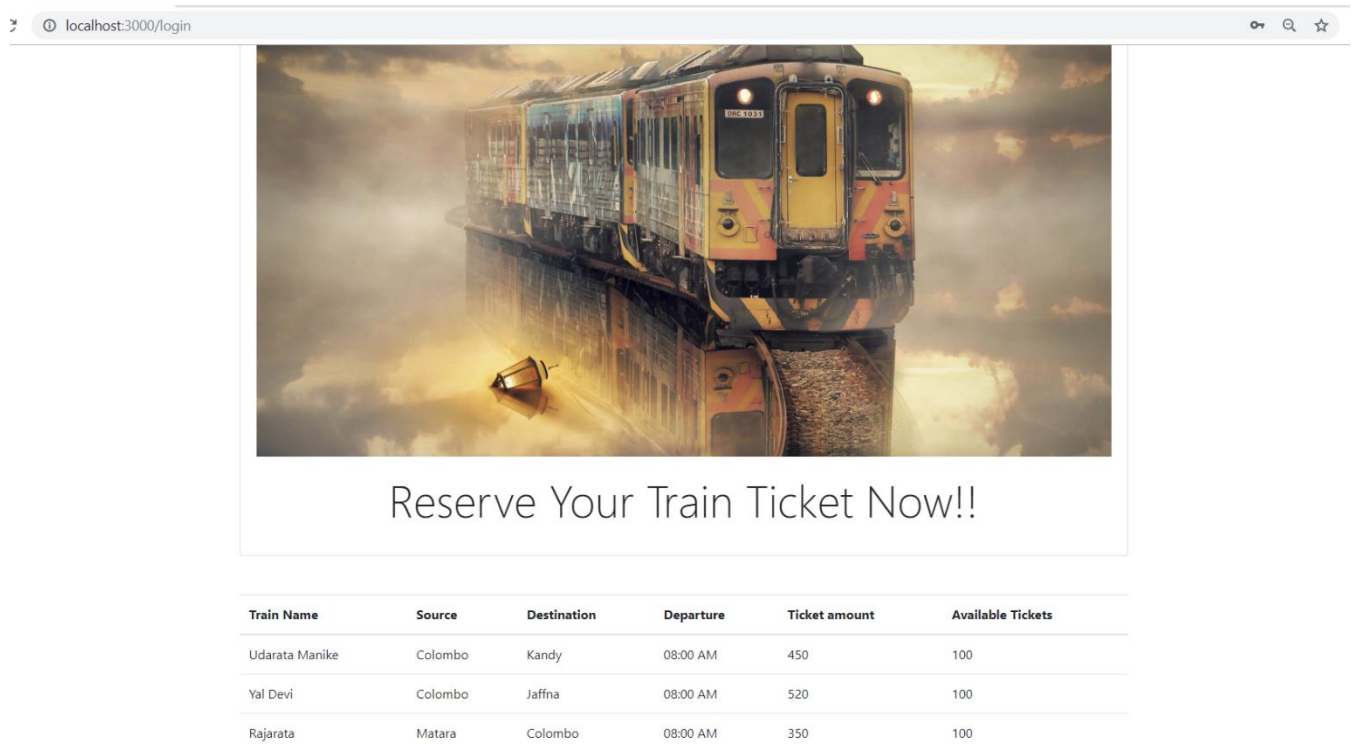
Full Name	Vinu Gamage
Email	sahanigamage@gmail.com
NIC	985011214V
Employee Status	private

Reserve tickets

Fig. 3.0

Once the user clicks the “**Reserve tickets**” button the user will be redirect to the Train Ticket Reservation page.

Train Reservation Page will be displaying the available trains with the source, destination, departure, ticket amount and available number of tickets.

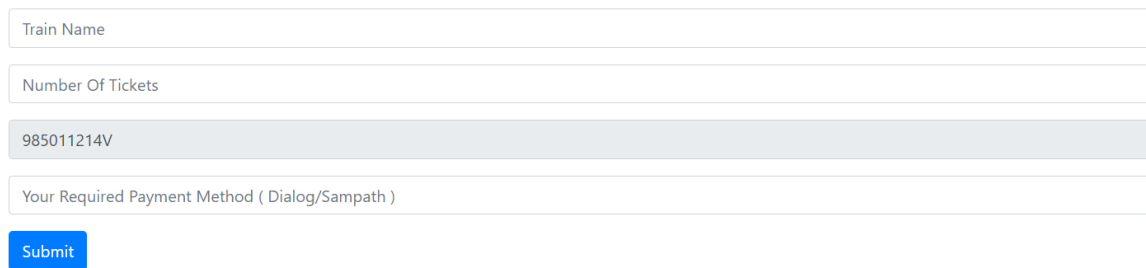


Train Name	Source	Destination	Departure	Ticket amount	Available Tickets
Udarata Manike	Colombo	Kandy	08:00 AM	450	100
Yal Devi	Colombo	Jaffna	08:00 AM	520	100
Rajarata	Matara	Colombo	08:00 AM	350	100

fig. 4.0

The user can reserve the train ticket by looking at the train departure timetable. Therefore, the user must enter the train name, no of tickets and payment method whether it is dialog or Sampath payment. The system will get the NIC number from the database according to the user and will save the employee status for train ticket amount calculation. (If the user is a **government** employee user will be getting a 10% discount from the total payment else if the employee status is **Private** will not be given any discount.)

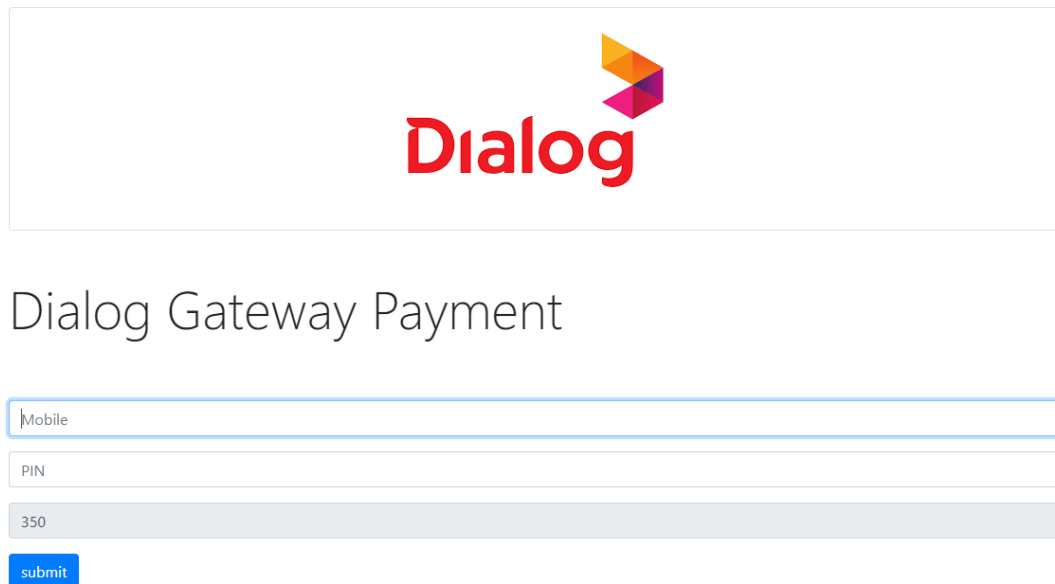
Train Ticket Reservation



The form consists of four input fields stacked vertically, followed by a submit button. The first field is labeled 'Train Name'. The second field is labeled 'Number Of Tickets'. The third field is a text box containing the value '985011214V'. The fourth field is labeled 'Your Required Payment Method (Dialog/Sampath)'. Below these fields is a blue button labeled 'Submit'.

fig. 5.0

When the user decides to make the payment using a dialog gateway, as the payment method **(fig. 5.0)** should enter dialog and submit the form so that it will redirect to the dialog payment gateway. The amount to be payed will be calculated and will be displayed as a read-only element. (In this scenario user have reserved a ticket from the train named “Rajarata”)



The image shows a web form for Dialog Gateway Payment. At the top, there is a large white box containing the Dialog logo, which consists of the word "Dialog" in red and a colorful geometric icon to its right. Below this box, the text "Dialog Gateway Payment" is displayed. The form itself consists of three input fields: the first is labeled "Mobile" and has a blue border; the second is labeled "PIN" and has a light gray border; the third is a read-only field displaying the number "350" in a light gray box. At the bottom left of the form is a blue "submit" button.

Fig. 6.0

If the customer is a “government” employee, they will be given a 10% discount as given bellow. (In this scenario a “government” user has reserved a ticket from the train named “Rajarata”. Indicates by **fig. 7.0 and fig. 8.0**)

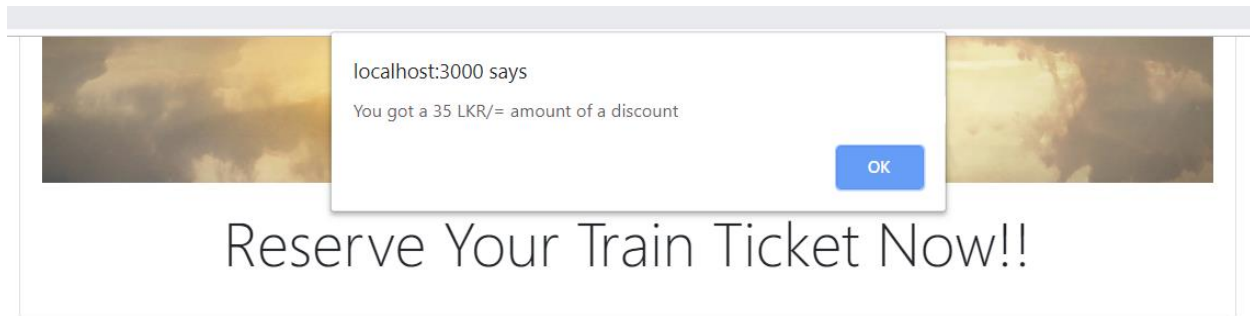


Fig. 7.0

A screenshot of a web application interface for "Dialog Gateway Payment". At the top, there is a large white box containing the "Dialog" logo, which consists of a stylized "D" in red and orange, followed by the word "ialog" in red. Below the logo, the text "Dialog Gateway Payment" is displayed in a large, black, sans-serif font. Underneath the text, there are three input fields: "Mobile", "PIN", and "315". Below the "315" field is a blue button with the text "submit".

fig. 8.0

Once the payment is successfully done the user will be getting an email confirming about the payment. And the user is directed to the login page.

Authentication and Security

When the user is registering an account user cannot use an email that has been already used.

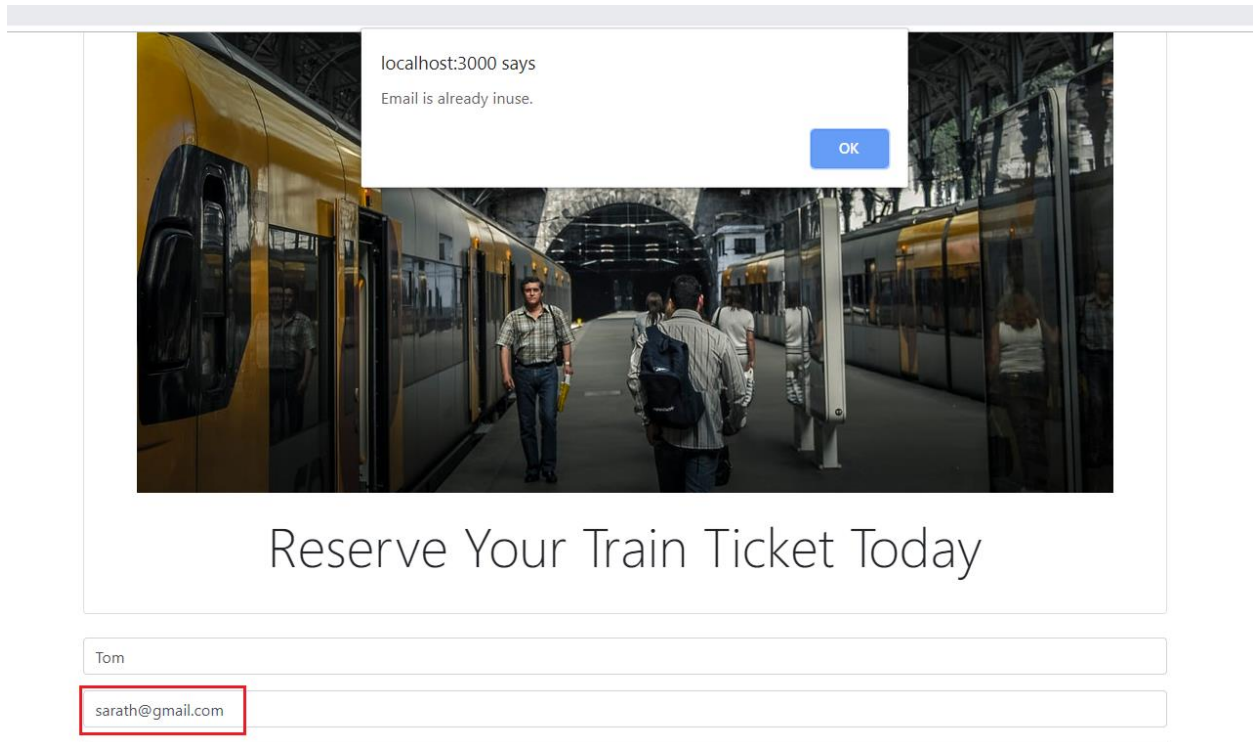


Fig. 9.0

Back End

(UserController.js)

```
//Retrieving data of a specific user according to email and password
this.findOne = (email,password) => {
  return new Promise((resolve,reject)=>{
    schema.find({email: email,password: password})
      .exec()
      .then(data => {
        resolve({status : 200,data:data});
      }).catch(err =>{
        reject({status:500,message:'Error:' + err});
      })
  })
}
```

(userRouter.js)

```
//verification of email
router.get('/:email', (req,res)=>{
  userController.checkEmail(req.params.email)
    .then(data=>{
      res.status(data.status).send(data.data);
    })
    .catch(err=>{
      res.status(err.status).send({message:err.message});
    })
})
```

Front End (signup.component.js)

```
//If any field is empty
if (fullname === '' || email === '' || nic === '' || empStatus === '' ||
password === ''){
  alert('One or more fields are empty');
} else{
  //verification of the email existence

  fetch('http://localhost:4000/user/' + email,{
    method : 'GET',
    headers:{'Content-Type' : 'application/json'}
  }).then(res=>{
    return res.json();
  }).then(data =>{
    let user = JSON.stringify(data);
    console.log(user);
  })
```

```
    if (user !== '[]') {
      alert('Email is already in use.');
```

```
    } else {
      let data = {"fullname": fullname, "email" :
email, "nic": nic, "empStatus" : empStatus, "password" : password};
      console.log(data);
      fetch('http://localhost:4000/user/', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: { 'Content-Type': 'application/json' }
      }).then(res => {
        return res.json();
      }).then(data => {
        alert('You are successfully signed up');

ReactDOM.render(<Login />, document.getElementById('root'));
      }).catch(err => {
        alert('Error occurred when signing up : ' + err);
      })
    }
  }).catch(err => {
    alert('Error occurred when signing up : ' + err);
  })
}
```

User that have valid credentials can only reserve train tickets.

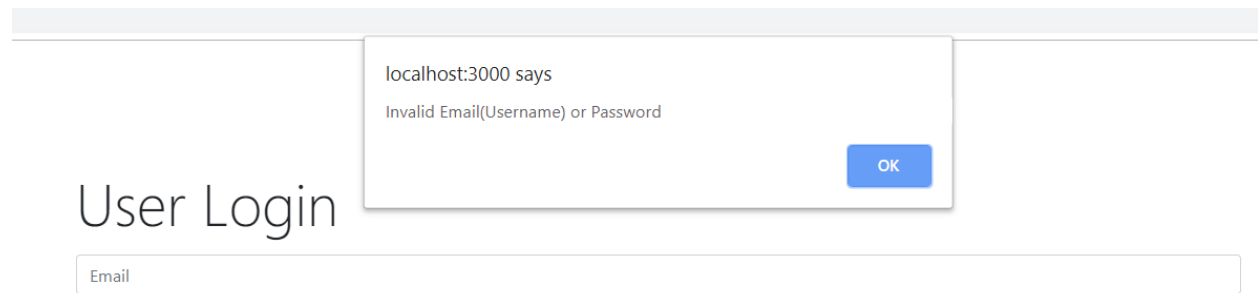


Fig. 10.0

Front End (login.component.js)

```

if (email === '' || password === '') {
  alert('Email or Password is empty');
} else {
  let credentials = {"email" : email, "password":password};
  let count = false;

  fetch('http://localhost:4000/user/' + credentials.email + '/' +
credentials.password, {
    method: 'GET',
    headers: {'Content-Type' : 'application/json'}
  }).then(res => {
    return res.json();
  }).then(data => {
    let user = JSON.stringify(data);
    if (user !== '[]') {
      console.log(user);
      count=true;
      console.log(data);
      for (let user of data) {
        let fullname = user.fullname;
        let nic = user.nic;
        let empStatus = user.empStatus;
        ReactDOM.render(<Profile fullname={fullname} nic={nic}
empStatus={empStatus}
email={email}/>, document.getElementById('root'));
      }
    } else {
      alert('Invalid Email(Username) or Password ');
    }
  }).catch(err=>{
    alert('Error :' + err);
  })
}

```

```

    if (count === true) {
      ReactDOM.render(<TrainReservation
/>, document.getElementById('root'));
    } else {
      ReactDOM.render(<Login/>, document.getElementById('root'));
    }
  }
}

```

Once the payment is successfully completed the user will be notified and will receive an email about the ticket purchasing, he/she has done. (In the above given scenario email is received to the email account "sahanigamage@gmail.com")

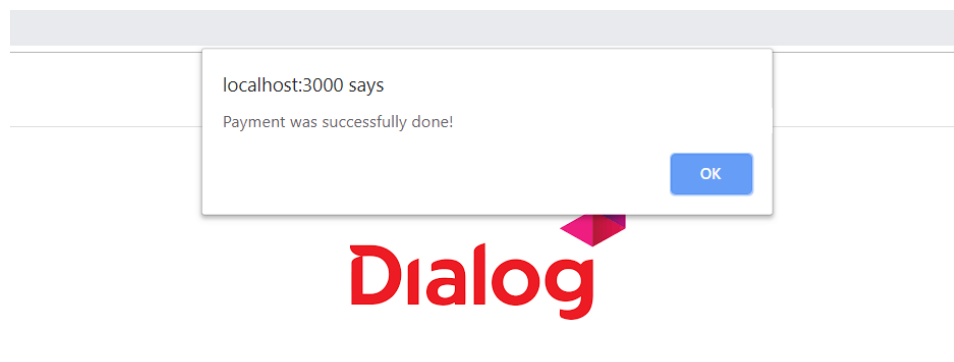


Fig. 11.0

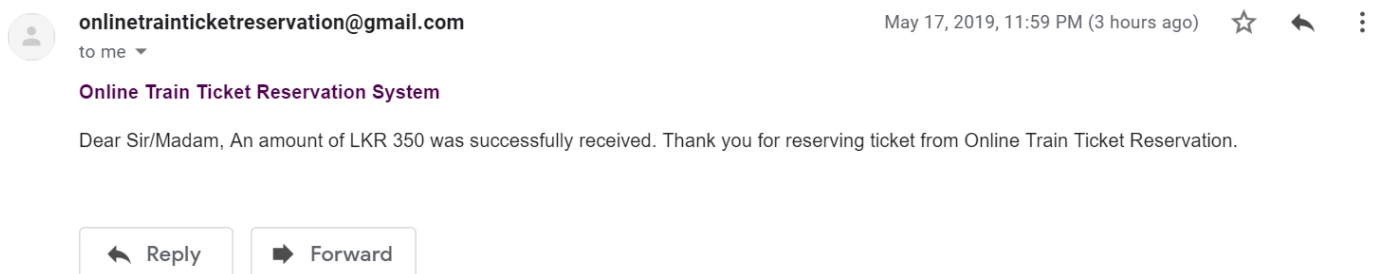


Fig. 12.0

Back End (sampathPaymentController.js)

```
sampathPay.save().then(() => {
    // //Sending the success payment mail
    let response = `Online Train Ticket Reservation System
    <p>Dear Sir/Madam,
        An amount of LKR ${data.paymentAmount} was
    successfully received. Thank you for reserving ticket from Online
    Train Ticket Reservation.
    </p>
    `;

    let sender = nodemailer.createTransport({
        service: 'gmail',
        secure: false,
        port: 25,
        auth: {
            user: 'onlinetrainticketreservation@gmail.com',
            pass: 'trainticket1234'
        },
        tls: {
            rejectUnauthorized: false
        }
    });

    let mailOptions = {
        from: "Online Train Ticket Reservation ",
        to: data.email,
        subject: "Payment Confirmation",
        text: "Thank you!!",
        html: response
    };

    console.log(response);

    return sender.sendMail(mailOptions);
}).then(() => {
    console.log("sent");
    // console.log('Message sent : %s', info, messageId);
    //console.log('Preview URL : %s',
    nodemailer.getTestMessageUrl(info));
    resolve({status: 200, message: 'Added new sampath payment
    details'});
}).catch(err => {
    reject({status: 500, message: 'Error :' + err});
})
})
```

Users are uniquely identified by the user email which is used as the username.

File Hierarchy

Project contains two main components such as backend and trainticket_reservation. Backend is a deployed using NodeJs and frontend (trainticket_reservation) is deployed using ReactJs.

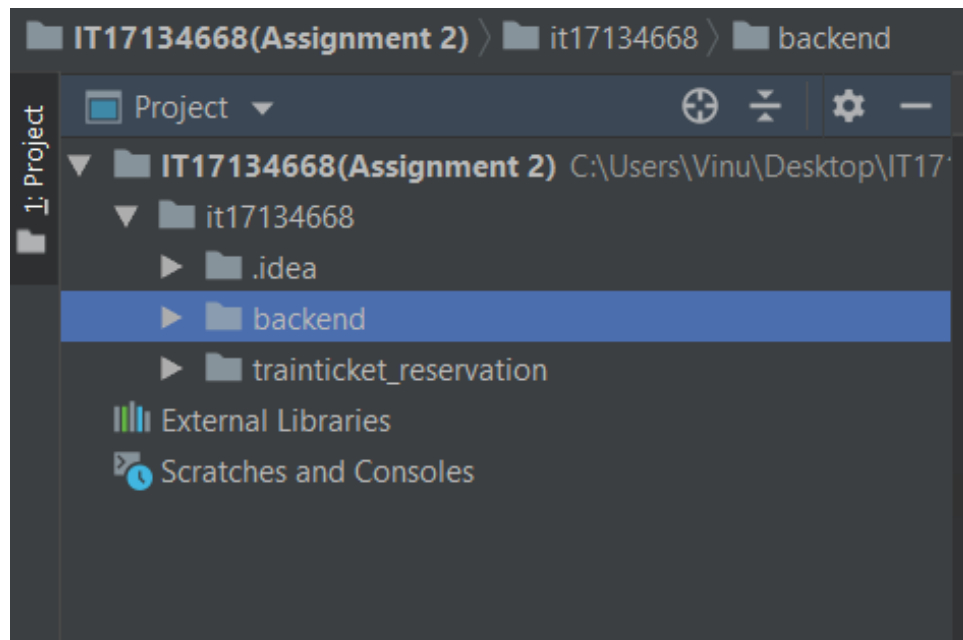
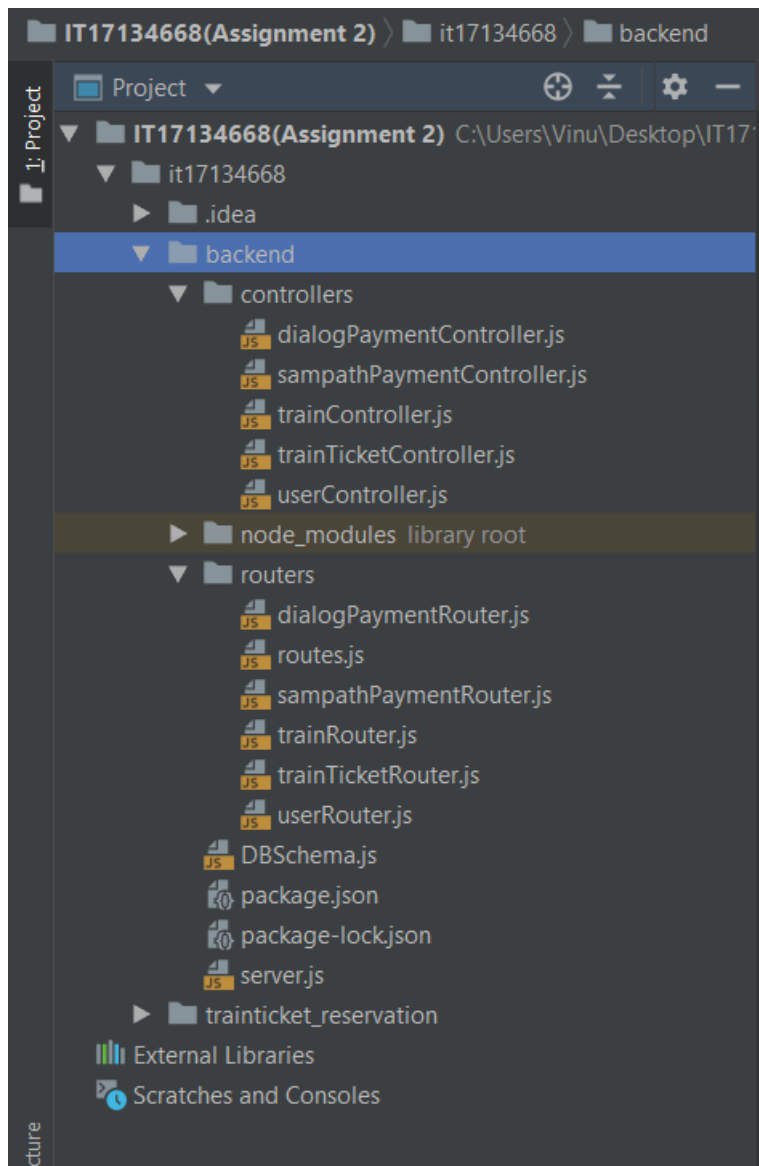
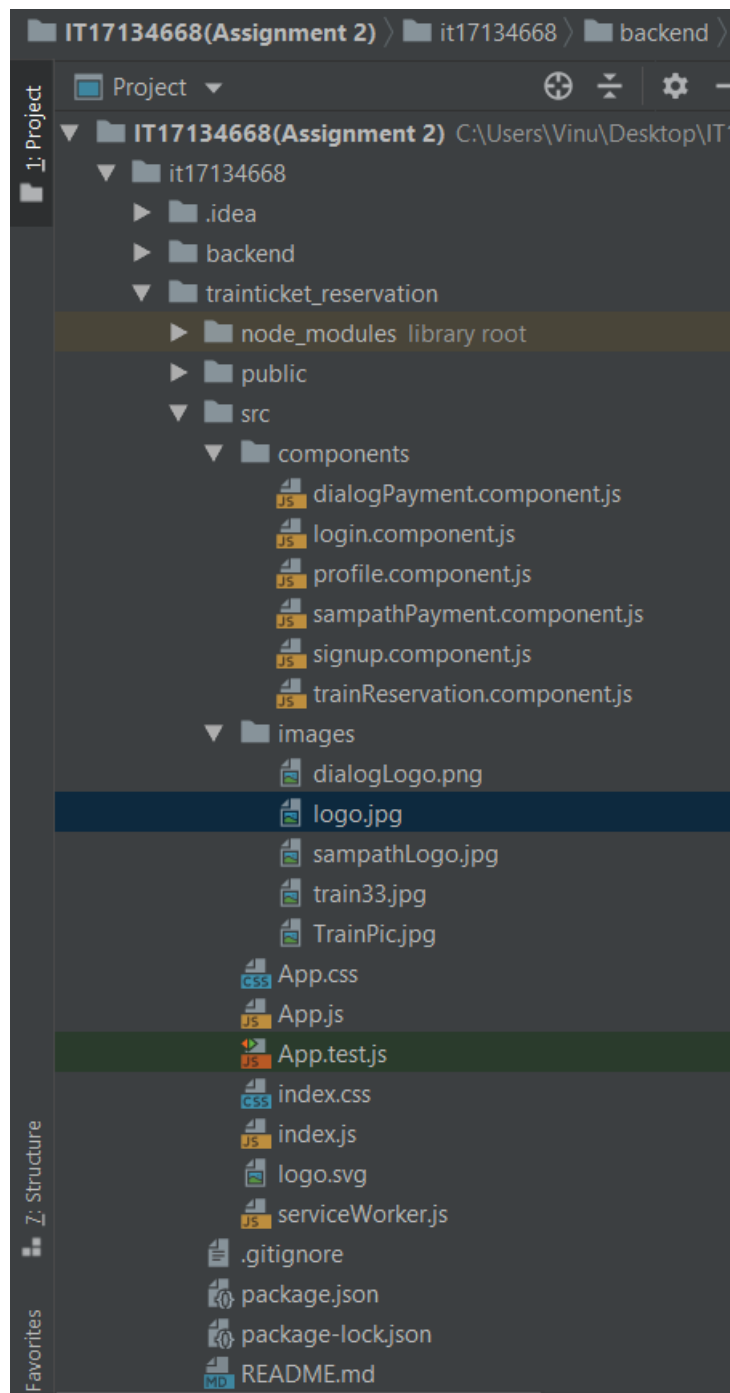


Fig. 13.0

**Fig. 14.0**

Backend

**Fig. 15.0**

Front end (trainticket_reservation)

Appendix

Front End

● App.js

```
import React, {Component} from 'react';
import "bootstrap/dist/css/bootstrap.css";
import {BrowserRouter as Router,Route,Link} from "react-router-dom";
import logo from "./images/logo.jpg";
import train from "./images/TrainPic.jpg";
import SignUp from "./components/signup.component";
import Login from "./components/login.component";
import Profile from "./components/profile.component";
import TrainReservation from
'./components/trainReservation.component';
import DialogPayment from './components/dialogPayment.component';
import SampathPayment from
'./components/sampathPayment.component';

class App extends Component {

  render() {

    return (

      <Router>
        <div className="container">

          <nav className="navbar navbar-expand-lg
navbar-light bg-light">
            <a className="navbar-brand"
href="http://google.com" target="_blank">
              <img src={logo} width="30"
height="30"alt="CodingTheSmartWay.com"/>
            </a>
            <Link to="/" className="navbar-
brand">Train Ticket Reservation</Link>

            <div className="collapse navbar-
collapse">
              <ul className="navbar-nav mr-auto">
                <li className="navbar-item">
                  <Link to="/" className="nav-
link">Sign Up</Link>
                </li>
```

```

        <li className="navbar-item">
          <Link to="/login"
className="nav-link">Login</Link>
        </li>
        <li className="navbar-item">
          <Link to="/profile"
className="nav-link">Profile</Link>
        </li>
      </ul>
    </div>

    </nav>

    <br/>

    <div className="card text-center">
      <div className="card-body">
        <img src={train} className="img-
fluid" alt="Responsive image" style={{paddingBottom : 20}}/>
        <h2 className="display-4 mb-
3">Reserve Your Train Ticket Today</h2>
      </div>
    </div>

    <br/>
    <Route path="/" exact component={SignUp}/>
    <Route path="/login" component={Login}/>
    <Route path="/profile" component={Profile}/>
    <Route path="/trainTicketReservation"
component={TrainReservation}/>
    <Route path="/dialogPayment"
component={DialogPayment}/>
    <Route path="/sampathPayment"
component={SampathPayment}/>
  </div>
</Router>

  );
}

}

export default App;

```

- **Signup Component(signup.component.js)**

```
import React, {Component} from 'react';
import axios from 'axios';
import Login from './login.component';
import ReactDOM from 'react-dom';

class Signup extends Component {

    // emailRegex = RegExp(/^[a-zA-Z0-9.!#$%&'*/+=?^_`{|}~-]+@[a-
zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/);
    // nicRegex = RegExp(/^[0-9]+V/);

    constructor(props) {
        super(props);
    }

    onChangeFullname=(e)=>{
        this.setState({
            user_fullname : e.target.value
        });
    }

    onChangeEmail=(e)=>{
        this.setState({
            user_email : e.target.value
        });
    }

    onChangeEmpStatus=(e)=>{
        this.setState({
            user_empStatus : e.target.value
        });
    }

    onChangeNic=(e)=>{
        this.setState({
            user_nic : e.target.value
        });
    }

    onChangePassword=(e)=>{
        this.setState({
            user_password : e.target.value
        });
    }
}
```

```

    }

    onSubmit=(e)=>{
        e.preventDefault();

        const fullname = this.refs.fullname.value;
        const email = this.refs.email.value;
        const nic = this.refs.nic.value;
        const empStatus = this.refs.empStatusOptions.value;
        const password= this.refs.password.value;

        //If any field is empty
        if (fullname === '' || email ==='' || nic ==='' ||
empStatus ==='' || password === ''){
            alert('One or more fields are empty');
        } else{
            //verification of the email existence

            fetch('http://localhost:4000/user/' + email,{
                method : 'GET',
                headers:{'Content-Type' : 'application/json'}})
            .then(res=>{
                return res.json();
            }).then(data =>{
                let user = JSON.stringify(data);
                console.log(user);
                if (user !== '[]'){
                    alert('Email is already inuse.');
                }else{
                    let data = {"fullname":fullname,"email" :
email,"nic":nic,"empStatus" : empStatus,"password" : password};
                    console.log(data);
                    fetch('http://localhost:4000/user/',{
                        method:'POST',
                        body:JSON.stringify(data),
                        headers:{'Content-
Type':'application/json'}})
                    .then(res=>{
                        return res.json();
                    }).then(data=>{
                        alert('You are successfully signed
up');

ReactDOM.render(<Login/>, document.getElementById('root'));
                    }).catch(err=>{
                        alert('Error occurred when sighing up
:' + err);

                    })
                }
            }
        }
    }
}

```



```

        }).catch(err=>{
            alert('Error occurred when signing up : ' +
err);
        })
    }

    render() {
        return (
            <div>
                <form onSubmit={this.onSubmit}>
                    <div className="form-group">
                        <input className="form-control"
placeholder="Full Name" ref="fullname" type="text"
onChange={this.onChangeFullname} required/>
                    </div>
                    <div className="form-group">
                        <input className="form-control"
placeholder="E-mail" ref="email" type="email"
onChange={this.onChangeEmail} required/>
                    </div>
                    <div className="form-group">
                        <input className="form-control"
placeholder="NIC" ref="nic" type="text"
onChange={this.onChangeNic} required/>
                    </div>
                    <div className="form-group">
                        <input className="form-control"
placeholder="Your Employee Status ( Private/Government )"
ref="empStatusOptions" type="text" onChange={this.onChangeNic}
required/>
                    </div>
                    <div className="form-group">
                        <input className="form-control"
placeholder="Password" ref="password" type="password"
onChange={this.onChangePassword} required/>
                    </div>
                    <div className="form-group">
                        <input type="submit" className="btn btn-
primary" value="Sign Up"/>
                    </div>
                </form>
            </div>
        )
    }
}

export default Signup;

```

- Login Component(login.component.js)

```
import React, {Component} from 'react';
import TrainReservation from './trainReservation.component';
import Profile from './profile.component';
import ReactDOM from 'react-dom';

class Login extends Component {

  constructor(props) {
    super(props);
  }

  onChangeEmail=(e)=>{
    this.setState({
      user_email : e.target.value
    });
  }

  onChangePassword=(e)=>{
    this.setState({
      user_password : e.target.value
    });
  }

  onSubmit=(e)=>{
    e.preventDefault();

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

    if (email === '' || password === ''){
      alert('Email or Password is empty');
    } else {
      let credentials = {"email" : email,"password":password};
      let count = false;

      fetch('http://localhost:4000/user/' + credentials.email +
        '/' + credentials.password,{
          method:'GET',
          headers: {'Content-Type' : 'application/json'}
        }).then(res =>{
          return res.json();
        }).then(data => {
          let user = JSON.stringify(data);
          if (user !== '[]'){
            console.log(user);
            count=true;
          }
        })
    }
  }
}
```

```

        console.log(data);
        for (let user of data) {
            let fullname = user.fullname;
            let nic = user.nic;
            let empStatus = user.empStatus;
            ReactDOM.render(<Profile fullname={fullname}
nic={nic} empStatus={empStatus}
email={email}/>, document.getElementById('root'));
        }
    } else {
        alert('Invalid Email (Username) or Password ');
    }
}

}).catch(err=>{
    alert('Error : ' + err);
})
if (count === true) {
    ReactDOM.render(<TrainReservation
/>, document.getElementById('root'));
} else {

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

render() {
    return (
        <div className='container'>
            <br/><br/><br/><br/><br/>
            <h2 className="display-4 mb-3">User Login </h2>
            <form noValidate onSubmit={this.onSubmit}>
                <div className="form-group">
                    <input className="form-control" ref="email"
placeholder="Email" type="text" onChange={this.onChangeEmail}/>
                </div>
                <div className="form-group">
                    <input className="form-control"
ref="password" placeholder="Password" type="password"
onChange={this.onChangePassword} />
                </div>
                <div className="form-group">
                    <input type="submit" className="btn btn-
primary" value="Login"/>
                </div>
            </form>
        </div>

    );
}
}

```

```
export default Login;
```

- Profile Component(profile.component.js)

```
import React, {Component} from 'react';
import ReactDOM from 'react-dom';
import TrainReservation from './trainReservation.component';

class Profile extends Component {

  constructor(props) {
    super(props);

    this.state = {
      fullname : this.props.fullname,
      email : this.props.email,
      nic : this.props.nic,
      empStatus : this.props.empStatus
    }
  }

  onSubmit=(e)=>{
    e.preventDefault();
    ReactDOM.render(<TrainReservation nic={this.state.nic}
empStatus={this.state.empStatus}
email={this.state.email}/>, document.getElementById('root'));
  }

  render() {
    return (
      <div className="container">
        <br/><br/><br/><br/>
        <h2 className="display-4 mb-3"> Welcome To Your
Profile {this.state.fullname}!! </h2>
        <br/><br/>
        <table className="table">
          <tbody>
            <tr>
              <td>Full Name</td>
              <td>{this.state.fullname}</td>
            </tr>
            <tr>
              <td>Email</td>
              <td>{this.state.email}</td>
            </tr>
            <tr>
              <td>NIC</td>
              <td>{this.state.nic}</td>
            </tr>
          </tbody>
        </table>
      </div>
    );
  }
}
```

```
        </tr>
        <tr>
            <td>Employee Status</td>
            <td>{this.state.empStatus}</td>
        </tr>
    </tbody>
</table>
<form onSubmit={this.onSubmit}>
    <div className="form-group">
        <input type="submit" className="btn btn-
primary" value="Reserve tickets"/>
    </div>
</form>
</div>
    );
}
}

export default Profile;
```

- Train Ticket Reservation Component(trainReservation.component.js)

```
import React, {Component} from 'react';
import SampathPayment from './sampathPayment.component';
import DialogPayment from './dialogPayment.component';
import trainPic from '../images/train33.jpg';
import ReactDOM from 'react-dom';

const Train = (props) =>{
  return (
    <tr>
      <td>{props.trainName}</td>
      <td>{props.source}</td>
      <td>{props.destination}</td>
      <td>{props.departure}</td>
      <td>{props.ticketAmount}</td>
      <td>{props.availableTickets}</td>
    </tr>
  );
};

class TrainReservation extends Component {
  constructor(props) {
    super(props);

    this.state = {
      trains:[],
      nic:this.props.nic,
      empStatus:this.props.empStatus,
      email:this.props.email,
      availableTickets:''
    }
  }

  displayTrains(trains){

    return trains.map((currentTrain,index)=>{
      return <Train trainName={currentTrain.trainName}
        source={currentTrain.source}
        destination={currentTrain.destination}
        departure={currentTrain.departure}
        ticketAmount={currentTrain.ticketAmount}
        availableTickets={currentTrain.availableTickets}
        key={index}/>
    })
  }
}
```

```

        });
    }

    componentDidMount() {
        fetch('http://localhost:4000/train/', {
            method: 'GET',
            headers: {'Content-Type': 'application/json'}
        }).then(res => {
            return res.json();
        }).then(trains => {
            this.setState({trains: trains.data});
        });
    }

    onChangeTrainName = (e)=>{
        this.setState({
            train_name : e.target.value
        });
    }

    onChangeTrainNoOfTickets = (e)=>{
        this.setState({
            train_noOfTickets : e.target.value
        });
    }

    onChangeTrainPayment = (e)=>{
        this.setState({
            train_payment : e.target.value
        });
    }

    onSubmit=(e)=> {
        e.preventDefault();

        const trainName = this.refs.trainName.value;
        const tickets = this.refs.tickets.value;
        const paymentMethod = this.refs.paymentOptions.value;
        let paymentFee = 0;

        console.log("Employee status : " + this.state.empStatus);
        switch (trainName.toLowerCase()) {
            case "rajarata":
                if (this.state.empStatus === 'private' ||
this.state.empStatus === 'Private') {
                    paymentFee = (tickets * 350) ;
                } else if (this.state.empStatus === 'Government'
|| this.state.empStatus === 'government') {
                    paymentFee = (tickets * 350) * 90/100;
                    let discount = (tickets * 350) * 10/100;
                    alert(`You got a ${discount} LKR/= amount of

```

```

    a discount`);
    }
    break;

    case "udarata manike":
        if (this.state.empStatus === 'private' ||
this.state.empStatus === 'Private'){
            paymentFee = tickets * 450;
        } else if (this.state.empStatus === 'Government'
|| this.state.empStatus === 'government') {
            paymentFee = (tickets * 450) * 90/100;
            let discount = (tickets * 450) * 10/100;
            alert(`You got a ${discount} LKR/= amount of
a discount`);
        }
        break;

    case "yal devi":
        if (this.state.empStatus === 'private' ||
this.state.empStatus === 'Private'){
            paymentFee = tickets * 520;
        } else if (this.state.empStatus === 'Government'
|| this.state.empStatus === 'government') {
            paymentFee = tickets * 520;
            let discount = (tickets * 520) * 10/100;
            alert(`You got a ${discount} LKR/= amount of
a discount`);
        }
        break;

    default:
        alert("Requested train is not available.");
        break;
    }

    if (trainName === '' || tickets === '' || paymentMethod
=== '') {
        alert('One or more fields are empty. ');
    } else {
        let
data={"trainName":trainName,"tickets":tickets,"nic":this.state.ni
c,"empStatus":this.state.empStatus,"paymentAmount":paymentFee,"pa
ymentMethod":paymentMethod}
        console.log(data);
        fetch('http://localhost:4000/trainTicket/', {
            method:"POST",
            body:JSON.stringify(data),
            headers:{"Content-Type":"application/json"}
        }).then(res=>{
            return res.json();
        }).then(data=>{
            fetch('http://localhost:4000/train/' + trainName

```



```

, {
    method: "PUT",
    body: JSON.stringify({noOfTickets: tickets}),
    headers: {"Content-Type": "application/json"}
  }).then(() => {
    alert("Train ticket/s was successfully
reserved.")
    if (paymentMethod.toLowerCase() === "dialog") {
      ReactDOM.render(<DialogPayment
paymentAmount={paymentFee} email={this.state.email}
nic={this.state.nic}/>, document.getElementById('root'));
    } else if (paymentMethod.toLowerCase() ===
"sampath") {
      ReactDOM.render(<SampathPayment
paymentAmount={paymentFee} email={this.state.email}
nic={this.state.nic}/>, document.getElementById('root'));
    }
  }).catch(err=> {
    alert("Error occurred in
trainReservation.component" + err);
  })
}

render() {
  return (
    <div className="container">
      <br/><br/>
      <div className="card text-center">
        <div className="card-body">
          <img src={trainPic} className="img-fluid"
alt="Responsive image" style={{paddingBottom : 20}}/>
          <h2 className="display-4 mb-3">Reserve
Your Train Ticket Now!!</h2>
        </div>
      </div>
      <br/><br/>
      <table className="table">
        <thead>
          <tr>
            <th scope="col1">Train Name</th>
            <th scope="col1">Source</th>
            <th scope="col1">Destination</th>
            <th scope="col1">Departure</th>
            <th scope="col1">Ticket amount</th>
            <th scope="col1">Available Tickets</th>
          </tr>
        </thead>
        <tbody>
          {
            this.displayTrains(this.state.trains)
          }
        </tbody>
      </table>
    </div>
  );
}

```

```

        </tbody>

    </table>
    <br/><br/> <br/>

    <div className="display-4 mb-3">Train Ticket
Reservation</div>
    <form onSubmit={this.onSubmit}>
        <div className="form-group">
            <input className="form-control"
ref="trainName" placeholder="Train Name" type="text"
onChange={this.onChangeTrainName} />
        </div>
        <div className="form-group">
            <input className="form-control"
ref="tickets" placeholder="Number Of Tickets" type="text"
onChange={this.onChangeTrainNoOfTickets} />
        </div>
        <div className="form-group">
            <input className="form-control"
placeholder="NIC" type="text" value={this.state.nic} readOnly/>
        </div>
        <div className="form-group">
            <input className="form-control"
placeholder="Your Required Payment Method ( Dialog/Sampath )"
ref="paymentOptions" type="text" onChange={this.onChangeNic}
required/>
        </div>
        <div className="form-group">
            <input type="submit" className="btn btn-
primary" value="Submit"/>
        </div>
    </form>
</div>

    );
}
}

export default TrainReservation;

```

- **Sampath Payment Component(sampathPayment.component.js)**

```
import React, {Component} from 'react';
import sampath from "../images/sampathLogo.jpg";
import Login from '../components/login.component';
import ReactDOM from "react-dom";

class SampathPayment extends Component {

  constructor(props) {
    super(props);

    this.state = {
      paymentAmount : this.props.paymentAmount,
      email: this.props.email,
      nic: this.props.nic
    }
  }

  onSubmit = (e) => {
    e.preventDefault();
    const name = this.refs.name.value;
    const cardNo = this.refs.cardNo.value;
    const cvc = this.refs.cvc.value;
    const paymentAmount = this.state.paymentAmount;
    const email = this.state.email;

    if (name === '' || cardNo === '' || cvc === '') {
      alert('One or more fields are empty. ');
    } else {
      let data = {"name" : name, "cardNo" : cardNo,
        "cvc":cvc, "paymentAmount":paymentAmount, "email":email};
      console.log(data);

      fetch('http://localhost:4000/creditCard/', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: { 'Content-Type': 'application/json' }
      }).then(res => {
        return res.json();
      }).then(data => {
        alert('Payment was successfully done!');
        console.log(data);
      });

      ReactDOM.render(<Login/>, document.getElementById('root'));
      // ReactDOM.render(<TrainReservation
      nic={this.state.nic}/>, document.getElementById('root'));
    }
  }).catch(err => {
    alert('Error occurred when inserting sampath
    payment data : ' + err);
  });
}
```

```

    })
  }
}

render() {
  return (
    <div className="container">
      <br/><br/><br/>
      <div className="card text-center">
        <div className="card-body">
          <img src={sampath} className="img-fluid"
alt="Responsive image" style={{paddingBottom : 20}}/>
        </div>
      </div>
      <br/><br/>
      <h2 className="display-4 mb-3">Sampath Gateway
Payment</h2>
      <br/><br/>
      <form onSubmit={this.onSubmit}>
        <div className="form-group">
          <input className="form-control"
ref="name" placeholder="Card Holders Name" type="text" />
        </div>
        <div className="form-group">
          <input className="form-control"
ref="cardNo" placeholder="Card No" type="text" />
        </div>
        <div className="form-group">
          <input className="form-control" ref="cvc"
placeholder="CVC" type="text" />
        </div>
        <div className="form-group">
          <input className="form-control"
ref="amount" placeholder="amount"
value={this.state.paymentAmount} type="text" readOnly/>
        </div>
        <div className="form-group">
          <input className="btn btn-primary"
type="submit" value="submit" />
        </div>
      </form>
    </div>
  );
}

export default SampathPayment;

```

- Dialog Payment Component(dialogPayment.component.js)

```
import React, {Component} from 'react';
import dialog from "../images/dialogLogo.png";
import ReactDOM from "react-dom";
import Login from '../components/login.component';

class DialogPayment extends Component {

  constructor(props) {
    super(props);

    this.state = {
      paymentAmount: this.props.paymentAmount,
      email: this.props.email,
      nic: this.props.nic
    }
  }

  onSubmit = (e) => {
    e.preventDefault();

    const mobile = this.refs.mobile.value;
    const pin = this.refs.pin.value;
    const paymentAmount = this.state.paymentAmount;
    const email = this.state.email;

    if (mobile === '' || pin === '' ) {
      alert('One or More fields are empty');
    } else {
      let data = {"mobile" :
mobile, "pin": pin, "paymentAmount": paymentAmount, "email": email};
      console.log(data);

      fetch('http://localhost:4000/dialogPay/', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: {'Content-Type': 'application/json'}
      }).then(res => {
        return res.json();
      }).then((data) => {
        console.log(data);
        alert('Payment was successfully done!');

ReactDOM.render(<Login/>, document.getElementById('root'));
//ReactDOM.render(<TrainReservation
nic={this.state.nic}/>)
      }).catch(err => {
        alert('Error occurred when inserting dialog
payment data :' + err)
```

```

    })
  }
}

render() {
  return (
    <div className="container">
      <br/><br/><br/>
      <div className="card text-center">
        <div className="card-body">
          <img src={dialog} className="img-fluid"
alt="Responsive image" style={{paddingBottom : 20}}/>
        </div>
      </div>

      <br/><br/>

      <h2 className="display-4 mb-3 ">Dialog Gateway
Payment</h2>

      <br/><br/>
      <form onSubmit={this.onSubmit}>
        <div className="form-group">
          <input className="form-control"
ref="mobile" placeholder="Mobile" type="text" />
        </div>
        <div className="form-group">
          <input className="form-control" ref="pin"
placeholder="PIN" type="text" />
        </div>
        <div className="form-group">
          <input className="form-control"
placeholder="amount" value={this.state.paymentAmount}
type="text" readOnly/>
        </div>
        <div className="form-group">
          <input className="btn btn-primary"
type="submit" value="submit" />
        </div>
      </form>

    </div>
  );
}

export default DialogPayment;

```

Back End

- **Server.js**

```
const express = require('express');
const body_parser = require('body-parser');
const cors = require('cors');

//importing the routes.js
const routes = require('./routers/routes');

const app = express();
const PORT = 4000;

app.use(body_parser.json());
app.use(cors());
app.use(body_parser.urlencoded({
  extended : false
}));
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-Credentials', true);
  next();
});

//Routes the requests to the routes.js
app.use('/', routes);

//server connection
app.listen(PORT, () => console.log('Server is running on port ' +
PORT));
```

- Database Schema (DBSchema.js)

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

//User Schema to store registered user details
const UserSchema = new Schema({
  fullname : {
    type :String
  },
  email : {
    type :String,
    required:true
  },
  nic : {
    type :String,
    required:true
  },
  password : {
    type :String,
    required: true
  },
  empStatus : {
    type :String
  }
});

//TrainSchema to store the available train details the
application provides
const TrainSchema = new Schema({
  trainName:{
    type:String
  },
  source:{
    type:String
  },
  destination:{
    type:String
  },
  departure:{
    type:String
  },
  ticketAmount:{
    type:String
  },
  availableTickets:{
    type:String
  }
});
```



```
//TrainSchema store the train ticket reservation details
const TrainTicketSchema = new Schema({
  trainName:{
    type:String,
    required:true
  },
  tickets:{
    type:String,
    required:true
  },
  nic:{
    type:String,
    required:true
  },
  empStatus:{
    type:String,
    required:true
  },
  paymentAmount:{
    type:String,
    required:true
  },
  paymentMethod:{
    type:String,
    required:true
  },
});

//Store payment details that have been taken placed using the
sampath credit card
const SampathPaymentSchema = new Schema({
  name:{
    type:String,
    required:true
  },
  cardNo:{
    type:String
  },
  cvc:{
    type:String,
    required:true
  },
  paymentAmount:{
    type:String,
  },
  email:{
    type:String,
  }
});
```

```
//Store payment details that have been taken placed using dialog
bill payments
const DialogPaymentSchema = new Schema({
  mobile:{
    type:String,
    required:true
  },
  pin:{
    type:String,
    required:true
  },
  paymentAmount:{
    type:String,
    required:true
  },
  email:{
    type:String,
    required:true
  }
})

//Renaming the schemas
mongoose.model('users', UserSchema);
mongoose.model('trains', TrainSchema);
mongoose.model('trainTickets', TrainTicketSchema);
mongoose.model('payment', SampathPaymentSchema);
mongoose.model('dialogPayment', DialogPaymentSchema);

//Connecting to the database
mongoose.connect('mongodb://localhost:27017/testDB', (err) =>{
  if(err){
    console.log(err);
    process.exit(-1);
  }else{
    console.log('MongoDB is connected successfully');
  }
});

module.exports = mongoose;
```

Routers

- Routes (Routes.js)

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

//importing the routes
const userRouter = require('./userRouter');
const trainRouter = require('./trainRouter');
const trainTicketRouter = require('./trainTicketRouter');
const sampathPaymentRouter = require('./sampathPaymentRouter');
const dialogPaymentRouter = require('./dialogPaymentRouter');

//using the routes
routes.use('/user', userRouter);
routes.use('/train', trainRouter);
routes.use('/trainTicket', trainTicketRouter);
routes.use('/creditCard', sampathPaymentRouter);
routes.use('/dialogPay', dialogPaymentRouter);

module.exports = routes;
```

- userRouter.js

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

//inserting the user
router.post('/', (req, res) => {
  userController.insert(req.body).then(data => {
    res.status(data.status).send({message: data.message});
  }).catch(err => {
    res.status(err.status).send({message: err.message});
  })
});

//Getting the user according to the email and password
router.get('/:email/:password', (req, res) => {
  userController.getOne(req.params.email, req.params.password)
    .then(data => {
      res.status(data.status).send(data.data);
    })
    .catch(err => {
```

```

        res.status(err.status).send({message:err.message});
    })
});

//verification of email
router.get('/:email', (req, res)=>{
    userController.checkEmail(req.params.email)
        .then(data=>{
            res.status(data.status).send(data.data);
        })
        .catch(err=>{
            res.status(err.status).send({message:err.message});
        })
});

module.exports = router;

```

● trainRouter.js

```

const trainController =
require('../controllers/trainController');
const express = require('express');
const router = express.Router();

//invoking getAll() function in the trainController to get all
details about trains
router.get('/', (req, res)=>{
    trainController.getAll()
        .then(data=>{
            res.status(data.status).send({data:data.data});
        }).catch(err=>{
            res.status(data.status).send({message: err.message});
        })
});

//invoking getOne() function in the trainController to get train
details according to the given train
router.get('/:trainName', (req, res)=>{
    trainController.getOne(req.params.trainName)
        .then(data=>{
            res.status(data.status).send({data:data.data});
        }).catch(err=>{
            res.status(err.status).send({message: err.message});
        })
});

//invoking update() function in the trainController to update the

```

```

    available no of tickets
    router.put('/:trainName', (req, res)=>{
        trainController.update(req.params.trainName, req.body)
        .then(data=>{
            res.status(data.status).send({data:data.data});
        }).catch(err=>{
            res.status(err.status).send({message:err.message});
        })
    });

    module.exports = router;

```

● trainTicketRouter.js

```

const trainTicketController =
require('../controllers/trainTicketController');
const express = require('express');
const router = express.Router();

//Inserting train Ticket reservation details
router.post('/', (req, res)=>{
    trainTicketController.insert(req.body)
    .then(data=>{
        res.status(data.status).send({message:data.message});
    }).catch(err=>{
        res.status(err.status()).send({message:
err.message});
    })
});

//Retrieving ticket reservation details according to the user
router.get('/:nic', (req, res)=>{
    trainTicketController.getOne(req.params.nic)
    .then(data=>{
        res.status(data.status).send(data.data);
    }).catch(err=>{
        res.status(err.status).send({message:err.message});
    })
});

module.exports = router;

```

● sampathPaymentRouter.js

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

//inserting the user
router.post('/', (req, res)=>{
    sampathPaymentController.insert(req.body).then(data=>{
        res.status(data.status).send({message: data.message});
    }).catch(err=>{
        res.status(err.status).send({message: err.message});
    })
});

//invoking getAll() function in the trainController to get all
details about trains
router.get('/', (req, res)=>{
    sampathPaymentController.getAll()
        .then(data=>{
            res.status(data.status).send({data: data.data});
        }).catch(err=>{
            res.status(data.status).send({message: err.message});
        })
});

module.exports = router;
```

● dialogPaymentRouter.js

```
const express = require('express');
const router = express.Router();
const dialogPaymentControlller =
require('../controllers/dialogPaymentController');

//Inserting dialog payment details with the help of the
controller class
router.post('/', (req, res)=>{

    dialogPaymentControlller.insert(req.body).then(data=>{
        res.status(data.status).send({message: data.message});
    }).catch(err=>{
        res.status(err.status).send({message: err.message});
    })

});

module.exports = router
```

Controllers.

- **UserController.js**

```
const mongoose = require('../DBSchema');
const schema = mongoose.model('users');

const userController = function () {

  //Adding user information
  this.insert = function (data) {
    return new Promise((resolve, reject) => {
      let user = schema({
        fullname:data.fullname,
        email:data.email,
        nic:data.nic,
        empStatus:data.empStatus,
        password:data.password
      });
      user.save().then(()=>{
        resolve({status:200,message:'Added a new user'});
      }).catch(err=>{
        reject({status:500,message:'Error :' + err});
      })
    })
  }

  //Retrieving data of a specific user according to email and
  password
  this.getOne = (email,password) => {
    return new Promise((resolve,reject)=>{
      schema.find({email: email,password: password})
        .exec()
        .then(data => {
          resolve({status : 200,data:data});
        }).catch(err =>{
          reject({status:500,message:'Error:' + err});
        })
    })
  }

  //Checking whether the entered mail is existing or not
  this.checkEmail = (email)=>{
    return new Promise((resolve, reject) => {
      schema.find({email : email})
        .exec()
        .then(data =>{
          resolve({status:200,data:data});
        })
    })
  }
}
```

```

        .catch(err=>{
            reject({status:500,message:'Error : ' + err});
        })
    })

}

}

module.exports = new userController();

```

- **trainController.js**

```

const mongoose = require('../DBSchema');
const schema = mongoose.model('trains');

const trainController = function () {

    //retrieving all trains available
    this.getAll=()=>{
        return new Promise((resolve, reject) => {
            schema.find()
                .exec()
                .then(data=>{
                    resolve({status:200,data:data});
                })
                .catch(err=>{
                    reject({status:500,message:"Error : " +
err});
                })
        })
    }

    //retrieving train details according to the name
    this.getOne=(trainName)=>{
        return new Promise((resolve,reject)=>{
            schema.find({trainName : trainName})
                .then(data =>{
                    resolve({status:200,data:data});
                }).catch(err=>{
                    reject({status:500,message:"Error while
retrieving details on " +trainName + " train : " + err});
                })
        })
    }

    this.update=(train_name,data)=>{

```



```

    return schema.find({trainName : train_name}).then(train
=> {
        console.log(JSON.stringify(train));
        console.log(JSON.stringify(data));

        let availableTicketUpdated =
train[0].availableTickets - data.noOfTickets
        console.log(availableTicketUpdated);
        return new Promise((resolve, reject) => {
            schema.update({trainName:
train_name},{ $set:{"availableTickets" : availableTicketUpdated}})
                .then(data=>{
                    resolve({status : 200,message:"Available
Tickets are updated."});
                })
                .catch(err=>{
                    reject({status:500,message:"Error while
updating the available Tickets:" + err});
                })
            })
        });
    }

}

module.exports = new trainController();

```

- trainTicketController.js

```

const mongoose = require('../DBSchema');
const schema = mongoose.model('trainTickets');

const trainTicketController = function () {

    //Inserting train ticket data
    this.insert = function (data) {
        return new Promise((resolve, reject) => {

            let trainTicket = schema({
                trainName:data.trainName,
                tickets:data.tickets,
                nic:data.nic,
                empStatus:data.empStatus,
                paymentAmount:data.paymentAmount,
                paymentMethod:data.paymentMethod
            })
            trainTicket.save()
                .then(()=>{

```

```

    resolve({status:200,message:trainTicket.tickets + " were reserved
in " + trainTicket.trainName });
    }).catch(err=>{
        reject({status:500,message:"Error : " +
err});
    })
  })
}

//Retrieving data according to the user using the nic
this.getOne=(nic)=>{
  return new Promise((resolve, reject) => {
    schema.find({nic:nic})
      .exec
      .then(data=>{
        resolve({status:200,data:data});
      }).catch(err=>{
        reject({status:500,message:"Error while
retrieving data of ticket reserving : " + err})
      })
  })
}

}

module.exports = new trainTicketController();

```

- **sampathPaymentController.js**

```

const mongoose = require('../DBSchema');
const schema = mongoose.model('payment');
const nodemailer = require('nodemailer');

const sampathPayController = function () {

  //Adding user information
  this.insert = function (data) {
    return new Promise((resolve, reject) => {
      let sampathPay = schema({
        name: data.name,
        creditNo: data.creditNo,
        cvc: data.cvc,
        paymentAmount: data.paymentAmount,
        email: data.email,
      });
      sampathPay.save().then(() => {

```

```

        // //Sending the success payment mail
        let response = `Online Train Ticket
Reservation System
        <p>Dear Sir/Madam,
            An amount of LKR ${data.paymentAmount}
was successfully received. Thank you for reserving ticket from
Online Train Ticket Reservation.
        </p>
        `;

        let sender = nodemailer.createTransport({
            service: 'gmail',
            secure: false,
            port: 25,
            auth: {
                user:
'onlinetrainticketreservation@gmail.com',
                pass: 'trainticket1234'
            },
            tls: {
                rejectUnauthorized: false
            }
        });

        let mailOptions = {
            from: "Online Train Ticket Reservation ",
            to: data.email,
            subject: "Payment Confirmation",
            text: "Thank you!!",
            html: response
        };

        console.log(response);

        return sender.sendMail(mailOptions);
    }).then(() => {
        console.log("sent");
        // console.log('Message sent : %s', info,
messageId);
        //console.log('Preview URL : %s',
nodemailer.getTestMessageUrl(info));
        resolve({status: 200, message: 'Added new
sampath payment details'});
    }).catch(err => {
        reject({status: 500, message: 'Error :' + err});
    })
})

```

```

    }
  };

  module.exports = new sampathPayController();

```

- dialogPaymentController.js

```

const mongoose = require('../DBSchema');
const schema = mongoose.model('dialogPayment');
const nodemailer = require('nodemailer');

const dialogPayController = function () {

  //Inserting the dialog payment details
  this.insert = function (data) {
    return new Promise((resolve, reject) => {

      let dialogPay = schema({
        mobile:data.mobile,
        pin:data.pin,
        paymentAmount:data.paymentAmount,
        email:data.email
      })

      dialogPay.save().then(()=>{

        // //Sending the success payment mail
        let response = `Online Train Ticket
Reservation System </b>
        <p>Dear Sir/Madam,
          An amount of LKR ${data.paymentAmount} was
successfully received. Thank you for reserving ticket from Online
Train Ticket Reservation.
        </p>
        `;

        let sender = nodemailer.createTransport({
          service: 'gmail',
          secure: false,
          port: 25,
          auth: {
            user:
'onlinetrainticketreservation@gmail.com',
            pass: 'trainticket1234'
          },
          tls: {
            rejectUnauthorized: false
          }

```

```
});

let mailOptions = {
  from: "Online Train Ticket Reservation ",
  to: data.email,
  subject: "Payment Confirmation",
  text: "Thank you!!",
  html: response

};
console.log(response);

return sender.sendMail(mailOptions);
}).then(() => {
  console.log("sent");
  // console.log('Message sent : %s', info,
messageId);
  //console.log('Preview URL : %s',
nodemailer.getTestMessageUrl(info));
  resolve({status: 200, message: 'Added new sampath
payment details'});
}).catch(err => {
  reject({status: 500, message: 'Error :' + err});
})
})

}

};

module.exports = new dialogPayController();
```