Redesigning Train Ticket Reservation System
A project Report

<u>Submitted by</u>

Kriti Chapagain -18BCE2483
Samar Pratap Singh-18BCE2169

<u>Submitted to</u>

Prof. Thenmozhi T
School of Computer Science and Engineering

Table of Contents

1. Introduction..3
2. Motivation...3
3. High Level Architecture...5
   3.1. Component diagram..5
   3.2. Overall system architecture...6
4. Rest APIs...7
   4.1. Railway..7
   4.2. Users..9
   4.3. Payment...9
   4.4. Register..9
   4.5. Login..9
5. State Transition Network..11
   5.1. System Workflow...11
   5.2. Hierarchical Task Analysis...13
   5.2. System Workflow Scenario...13
6. Heuristic Evaluation..20
7. Appendix..22
   7.1. Front-end (Client side) 22
   7.2. Back-end (Server side) 53
   7.3. WSO2 EI (Enterprise Integration – ESB)..69
8. Testing...74
   8.1. Manual Testing...74
9. Conclusion..78
10. Future Works..78

1. Introduction

This is the report of the “Train Ticket Reservation System” web application, in which front end (client side) is developed using React JS and back end (server side) is developed using Node JS and Express JS. This web application use MongoDB as the database, which is a cross-platform document-oriented database.

In this web application users can provide the start and the destination locations, train, class, time, ticket quantity and the date of booking. Users can pay using credit/debit cards or by mobile phone, in which the amount is added to the mobile bill.

In the web application booking page, when user select a start location (which contains all the stations of all routes), the destination locations are filtered according to the route of the selected station.

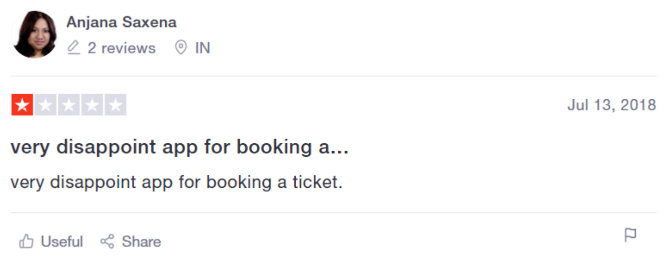
If the user is a government employee, they can have special discounts in this web application. Once user gave their NIC when registering, that NIC is validated using government web service to ensure that user is eligible to have discounts. NIC is optional and if user enter a NIC, then only it will be validated using government web service.

2. Motivation

A website should basically contain user friendly navigation with good usability . Our main focus is to build a website which a naive user can easily interact with, which is not taken care of in the existing system.

The drawbacks and the reviews found on the internet were the motivations for this project.i.e.

- Non user friendly
- Over crowded user interface
- Too many advertisements
- Confusing Booking system



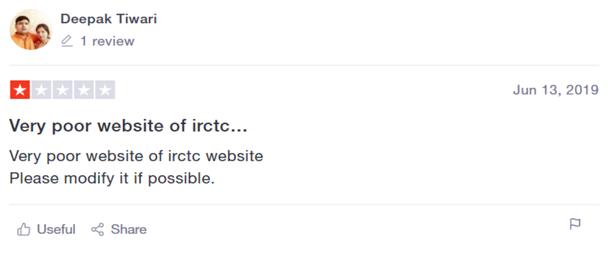
Anjana Saxena
2 reviews IN

★☆☆☆☆ Jul 13, 2018

very disappoint app for booking a...

very disappoint app for booking a ticket.

Useful Share



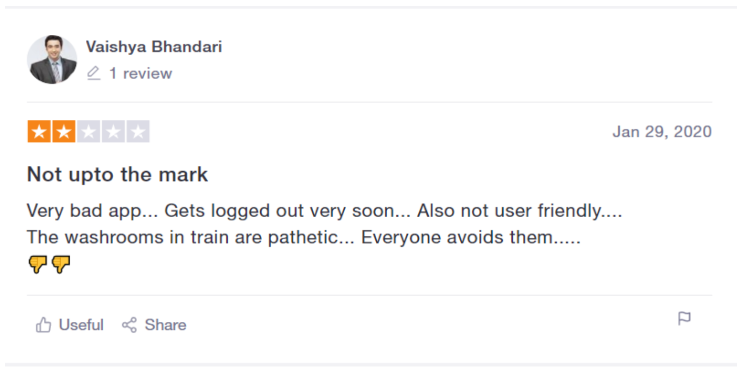
Deepak Tiwari
1 review

★☆☆☆☆ Jun 13, 2019

Very poor website of irctc...

Very poor website of irctc website
Please modify it if possible.

Useful Share



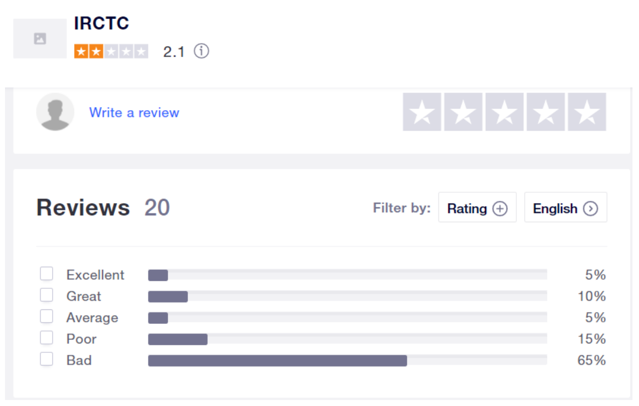
Vaishya Bhandari
1 review

★★☆☆☆ Jan 29, 2020

Not upto the mark

Very bad app... Gets logged out very soon... Also not user friendly....
The washrooms in train are pathetic... Everyone avoids them.....
👎👎

Useful Share



IRCTC
★★☆☆☆ 2.1

Write a review

Reviews 20

Filter by: Rating English

Excellent	5%
Great	10%
Average	5%
Poor	15%
Bad	65%

3. High Level Architectural Diagrams

Front end of the web application is developed using React.js, backend is developed using Node.js and Express.js, MongoDB database is connected to the back end and the front end and the back end communicates with WSO2 EI, which is a comprehensive integration solution that enables communication among various, disparate applications.

3.1. Component Diagram

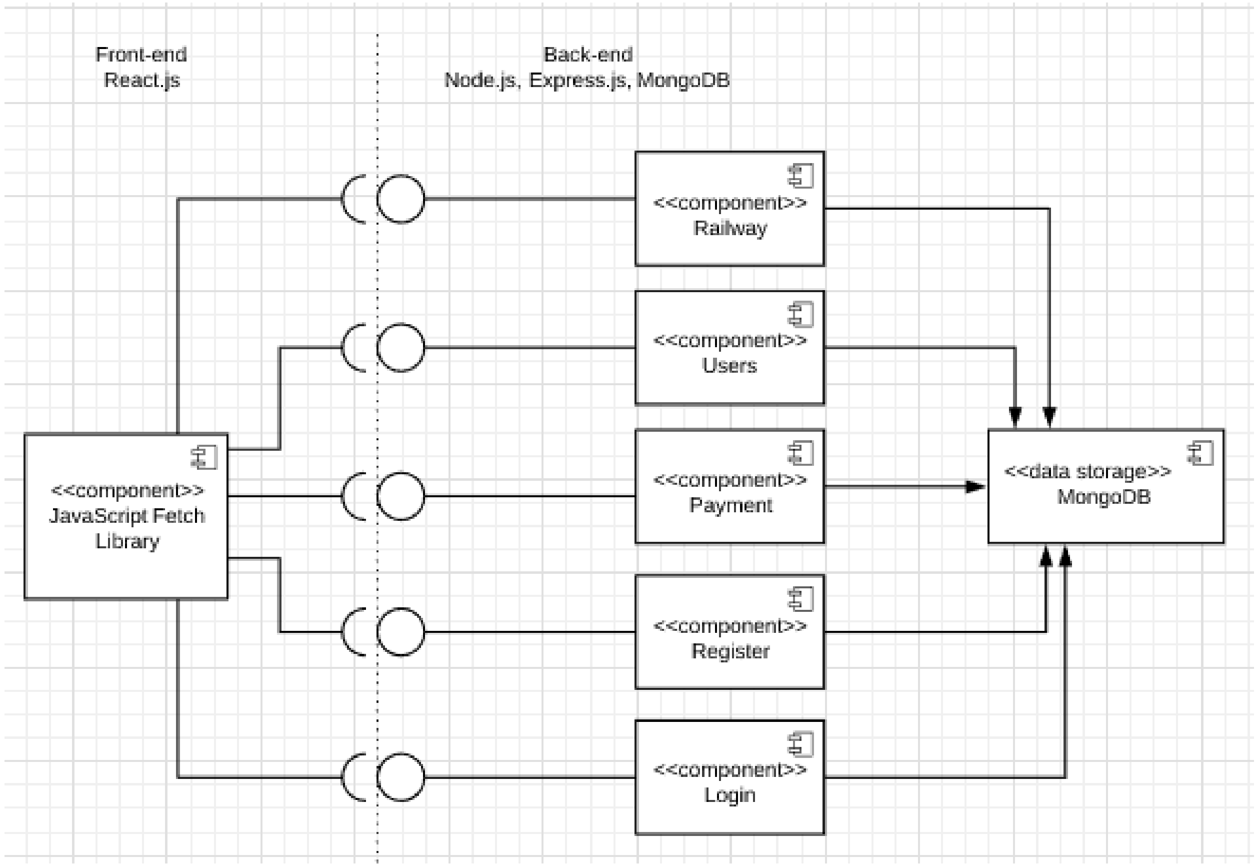


Fig 1: component diagram

3.2. Overall System Architecture

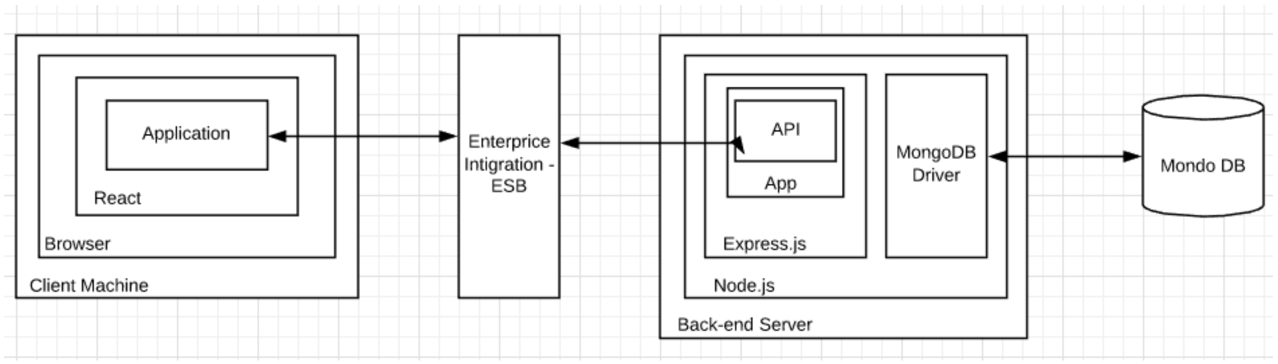


Fig 2: overall system architecture

4. Rest APIs

Following are the REST endpoints deployed in the back-end.

4.1. Railway

A. /railway/routes

This is a GET endpoint which returns an array of routes which includes route name and the array of stations in that route.

B. /railway/route/{id}

This is a GET endpoint which has a path parameter of route id. It returns all the stations for a given route id.

C. /railway/trains

This is a GET endpoint which returns array of all the trains in the database.

D. /railway/trains/{route}

This is a GET endpoint which has a path parameter of route id. It returns array of all the trains which are running on the specified route.

E. /railway/classes

This is a GET endpoint which returns array of all the train classes available in the database.

F. /railway/schedules

This is a GET endpoint which returns array of all the train schedules available in the database.

G. /railway/reservations

This endpoint support both GET and POST requests. If it is a GET request it returns all the reservations in the database. If it is a POST request it creates a new reservation according to the data in request body and save it in the database. After the new reservation saving it send email or a text message (according to the payment method, card payment - email, mobile payment - text message). Sample email and text messages are shown in Fig 1 and Fig 2 in introduction section.

H. /railway/reservations/{user}

This is a GET endpoint which contains path parameter of user id. It returns an array of all the reservations of the specified user.

I. /railway/reservations/{id}

This is a DELETE endpoint, which delete the specified reservation (reservation id) in the request path parameter.

J. /railway/contact

This is a POST endpoint, which used to process customer support requests. In this service it saves the support information given by the customer (email, phone, message, etc.) and send the confirmation details in email to both railway customer support team and to the customer.

4.2. Users

A. ***/users/{id}***

This endpoint supports PUT requests. User id should be specified in path parameter and the new user data should be send along with request body will be updated in the database.

4.3. Payment

Payment services are dummy web services which are used to represent the payment gateway.

A. ***/payment/card***

This is a POST endpoint, which validate the credit/debit card details and the payment amount send inside request body and send the validation status in response.

B. ***/payment/phone***

This is a POST endpoint, which validate the mobile phone details and the payment amount send inside request body and send the validation status in response.

4.4. Register

A. ***/Register***

This is a POST endpoint. It reads the new user details send inside the request body and save them in the database.

4.5. Login

A. ***/login***

This is a POST endpoint. It reads the username and password sent inside request body and validate them with values in the database and send the validation status in response.

4.6. Gov

This is a dummy government service which used to validate whether the user is a government employee or not.

A. */gov/employee/{nic}*

This is a GET endpoint, which accept the NIC number as a path parameter. It checks whether there is any employee with that NIC and validate. The validation status will be sent back in the response.

5. State Transition Networks

Following system workflow diagrams and system workflow scenario will help to get a better understanding on how the system works.

5.1. System Workflow

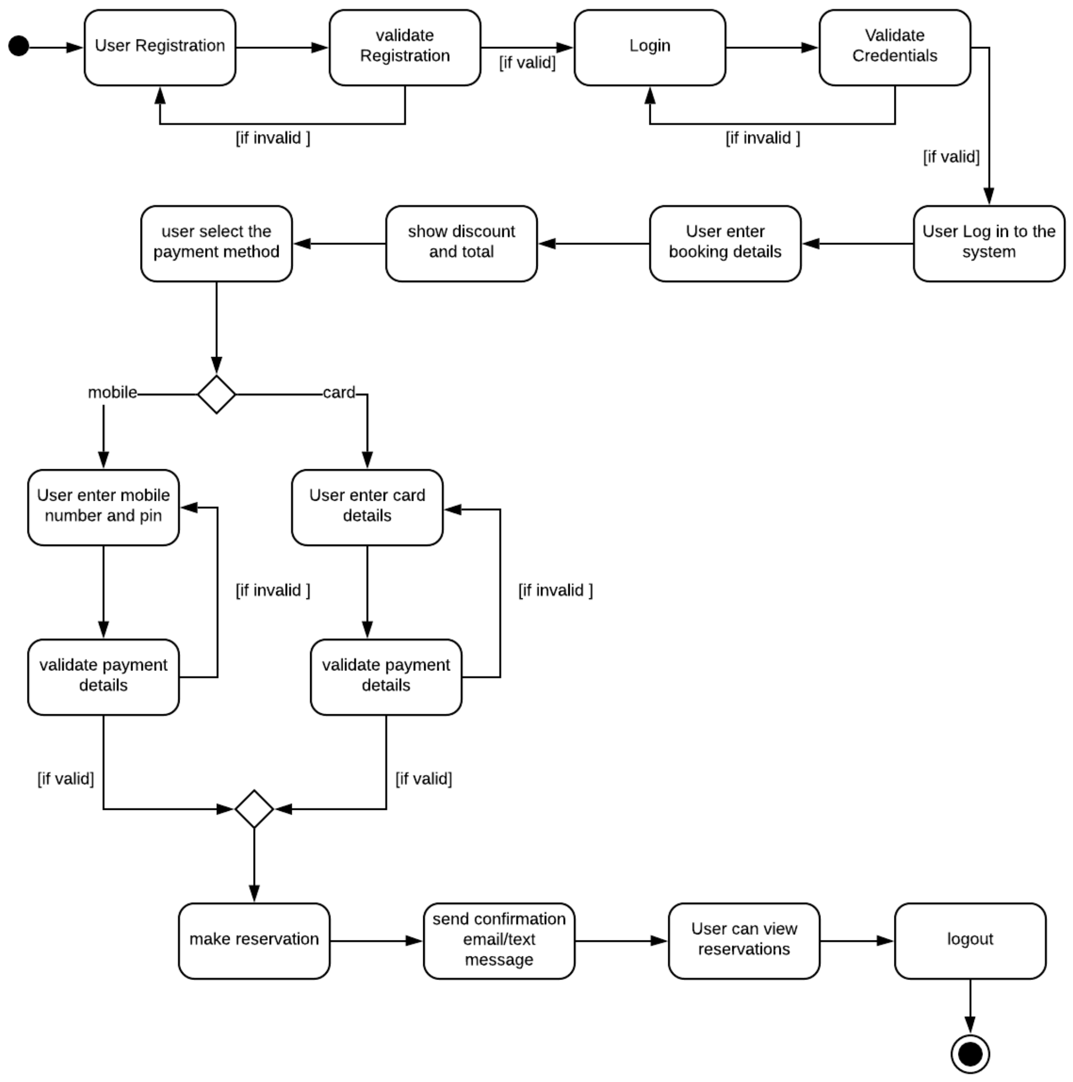


Fig 3: user make reservation

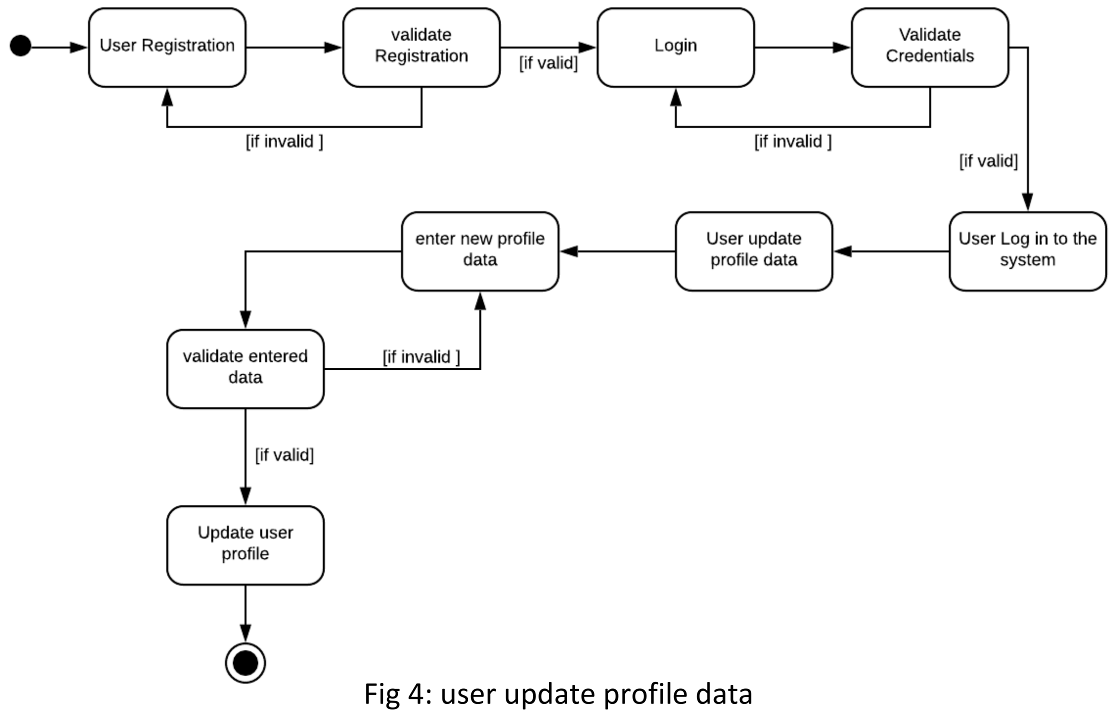


Fig 4: user update profile data

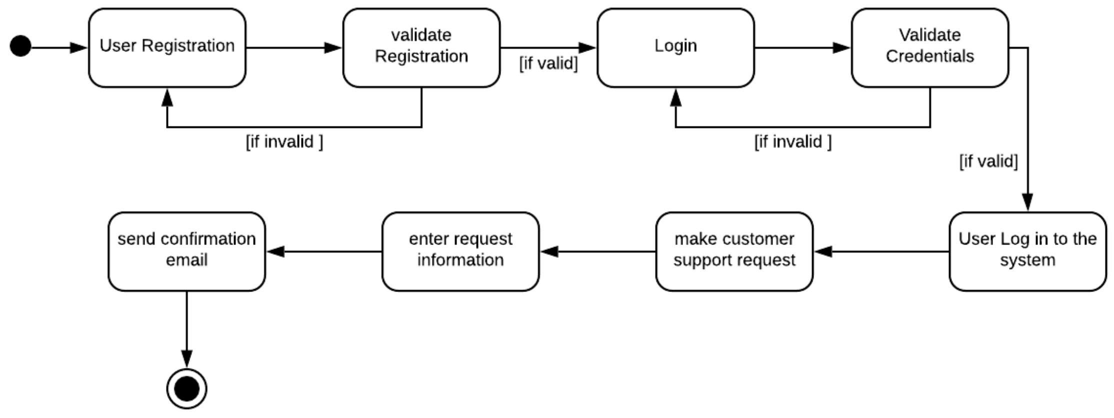


Fig 5: user make customer support request

5.2. Hierarchical Task Analysis

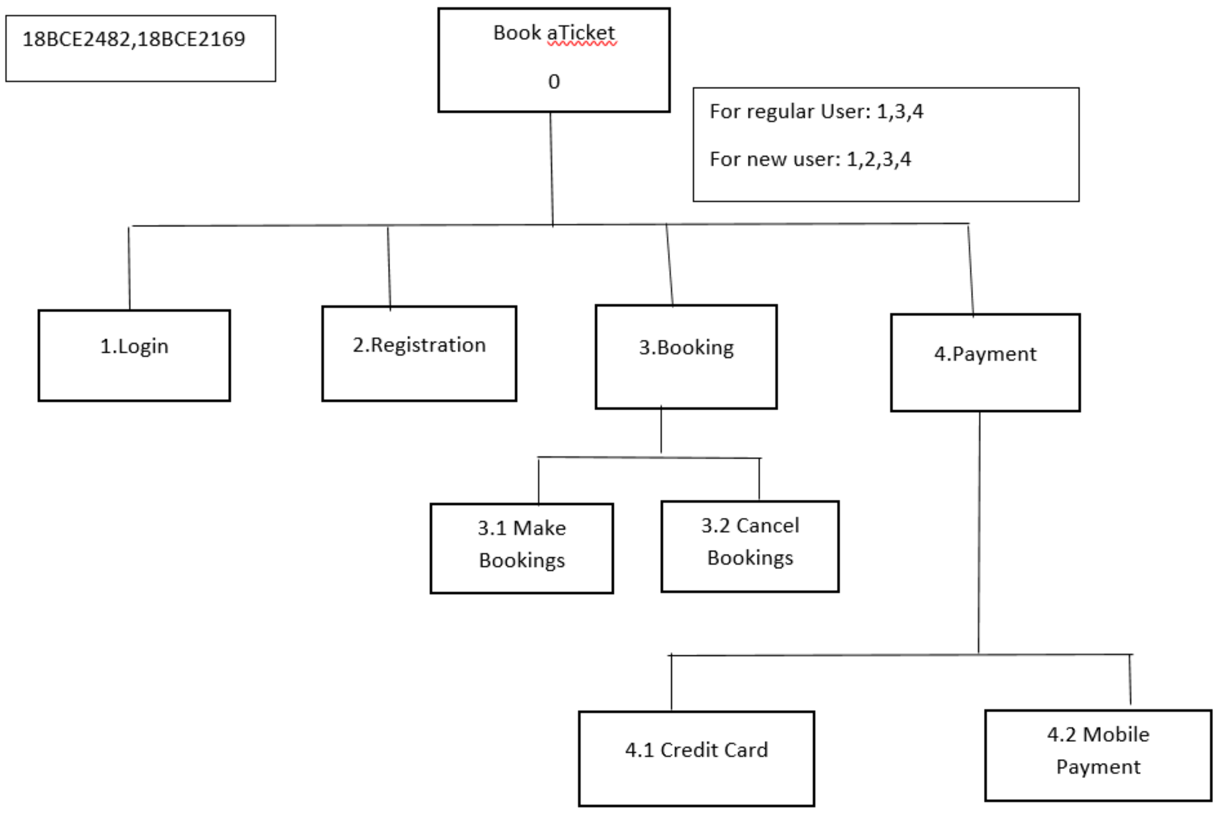


Fig 6: Hierarchical Task Analysis

5.3. System Workflow Scenario

The figure below shows the landing page of the web application. Any user can fill the reservation details in the form and view the cost of tickets, but if user want to make a reservation user should login first.

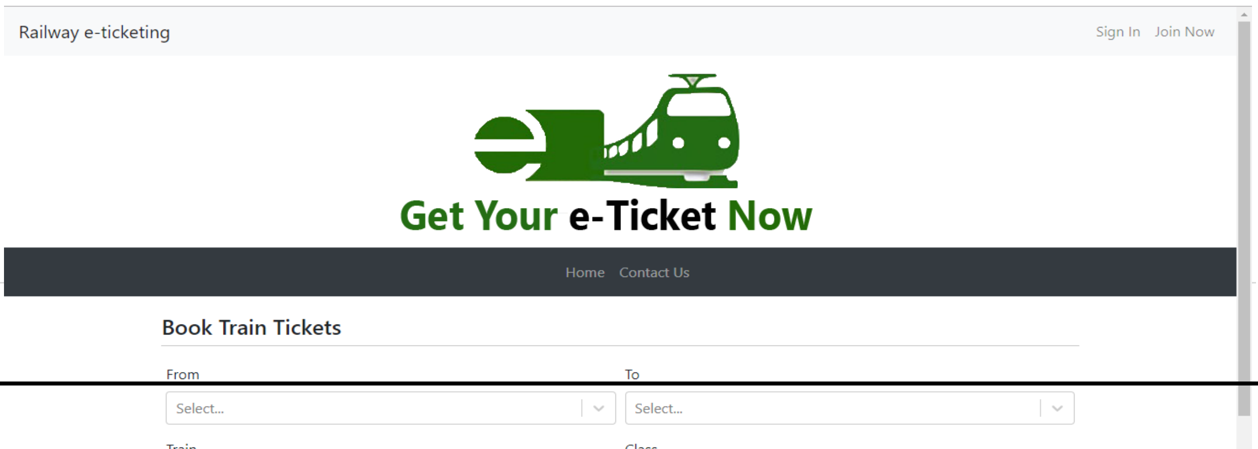
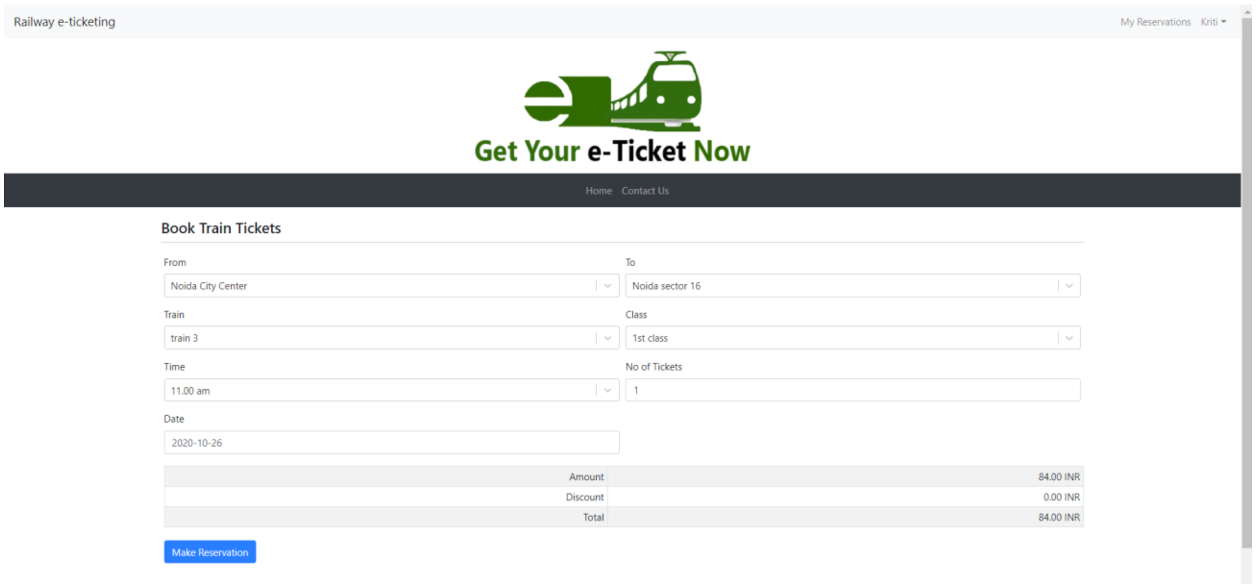


Fig 7: landing page

Firstly, user have to fill the "From" station, then the "To" stations are filtered according to the route of the selected "From" station. Once user filled the form, they can view the cost of tickets.



Book Train Tickets

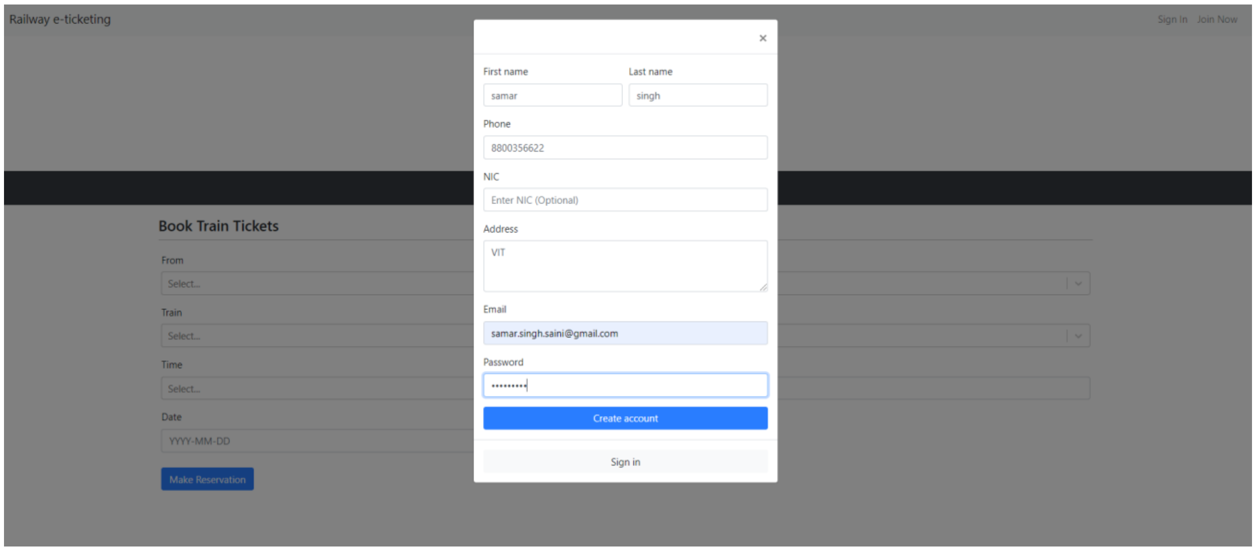
From: Noida City Center
To: Noida sector 16
Train: train 3
Class: 1st class
Time: 11.00 am
No of Tickets: 1
Date: 2020-10-26

Amount	84.00 INR
Discount	0.00 INR
Total	84.00 INR

Fig 8: user enter booking details

If user need to make a reservation, user need to login first. If user is not registered before, they can register their account first.

Users can click join now link in navigation bar and enter their details in the modal shown after clicking the link. NIC field is optional and if you enter a NIC it will be validated using government service to ensure that user is eligible to discounts.



First name: samar
Last name: singh
Phone: 8800356622
NIC: Enter NIC (Optional)
Address: VIT
Email: samar.singh.saini@gmail.com
Password: •••••••••

Fig 9: register modal

Once user has successfully register user can login to the system and make reservations.

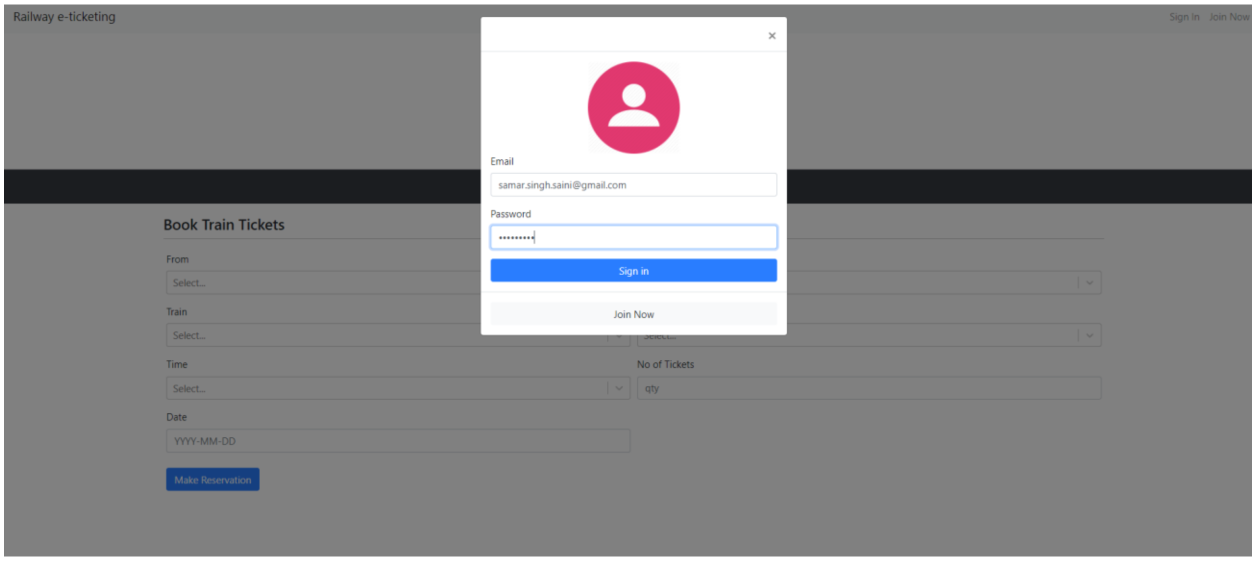


Fig 10: login modal

Once user login successfully the navigation bar will change and show user's first name in dropdown and the "My Reservations" link. The user data will be saved in local storage until they sign out.

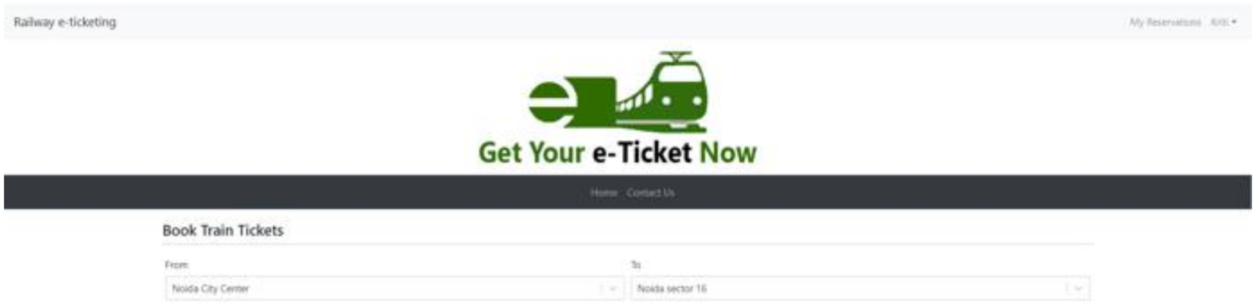
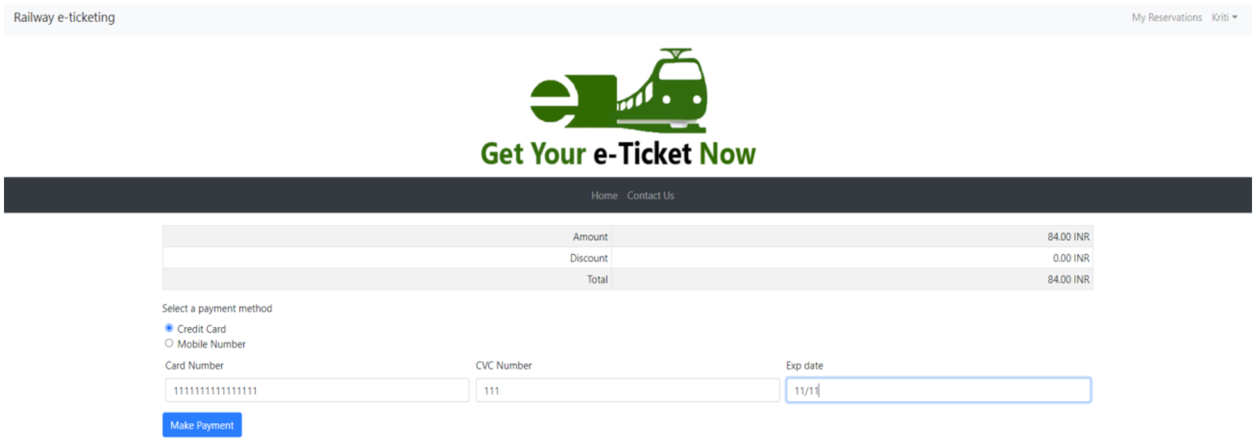


Fig 11: logged user

Then user can click "Make Reservation" button in the bottom of the form and the user will directed to "Payment" page. Only logged in users are allowed in payment page, otherwise they are asked to login.

In the payment page it shows the amount and user can select a payment method (card or phone).



Amount	84.00 INR
Discount	0.00 INR
Total	84.00 INR

Select a payment method

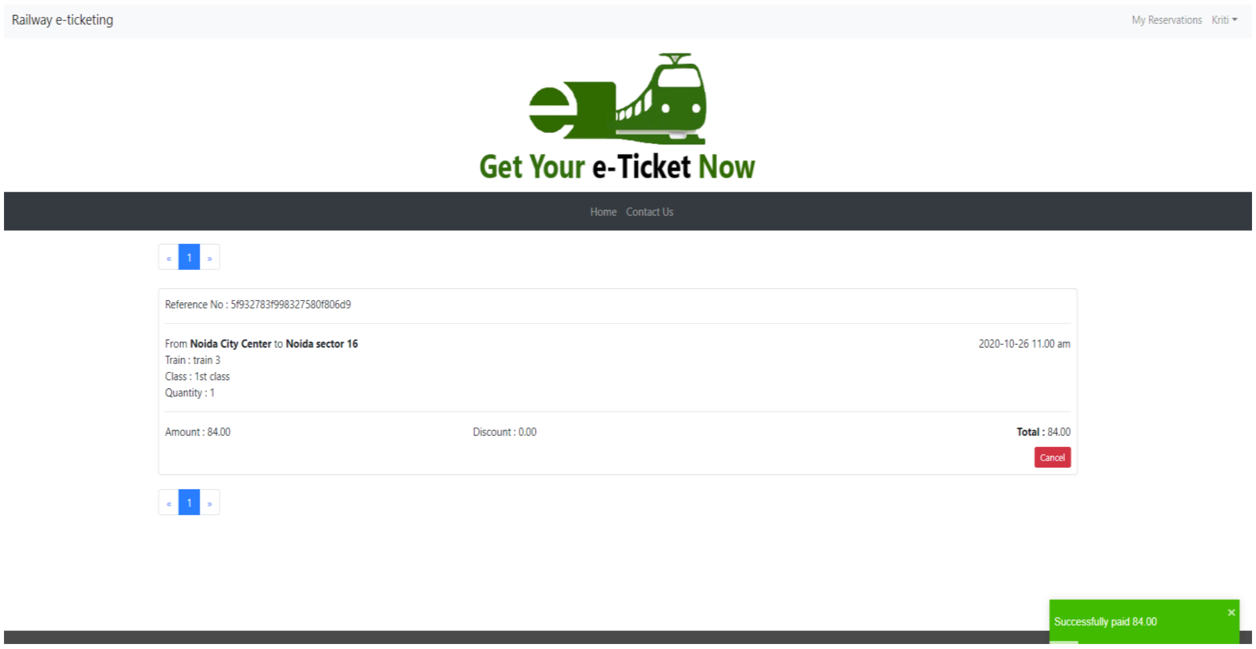
[✓ Credit Card]
Mobile Number

Card Number: 1111111111111111
CVC Number: 111
Exp date: 11/11

Fig 12: user select card payment

When user select mobile payment, the user's mobile number (entered when registering) will be auto filled, but user can change it if they want to make the payment with different mobile number.

Once user enter valid payment details, the reservation will be made and user will be directed to "My reservation" page, in where user can view all their previous reservations and new reservations.



Reference No : 5f932783f998327580f806d9

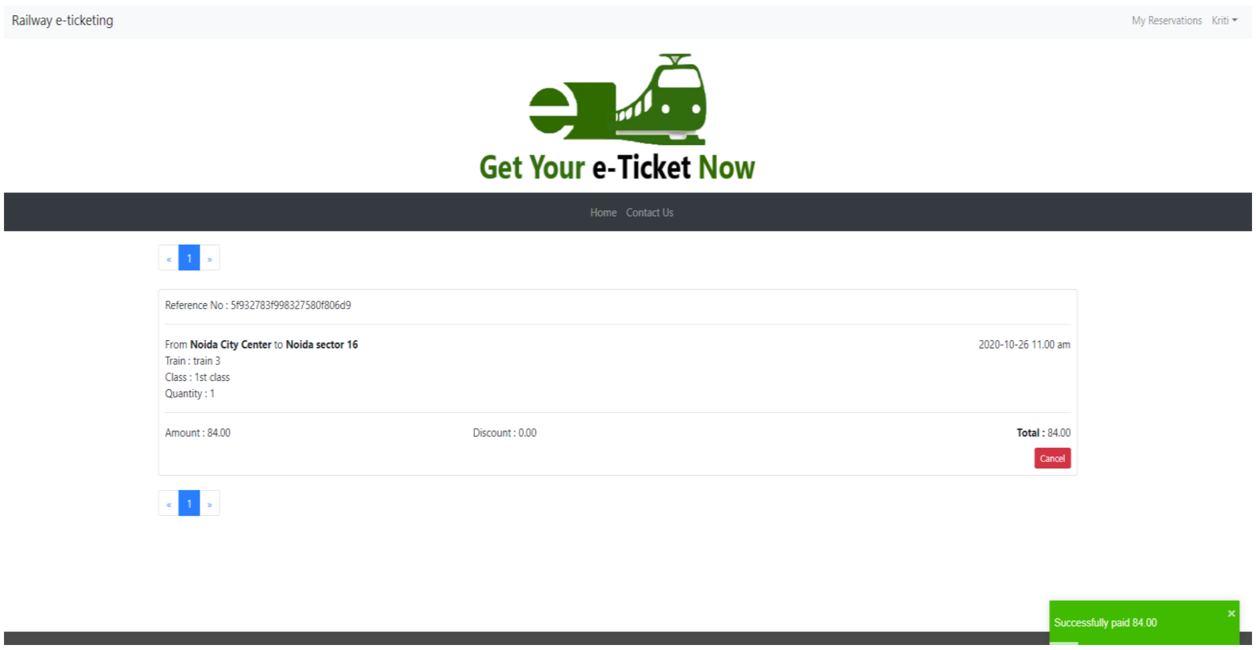
From **Noida City Center** to **Noida sector 16** 2020-10-26 11.00 am
Train : train 3
Class : 1st class
Quantity : 1

Amount : 84.00 Discount : 0.00 **Total :** 84.00

Successfully paid 84.00

Fig 13: my reservations page

Users can cancel the reservation by clicking the cancel button in the reservations shown in the "My Reservations" page. User will be asked confirmation after clicking the cancel button.



Reference No : 5f932783f998327580f806d9

From **Noida City Center** to **Noida sector 16** 2020-10-26 11.00 am

Train : train 3

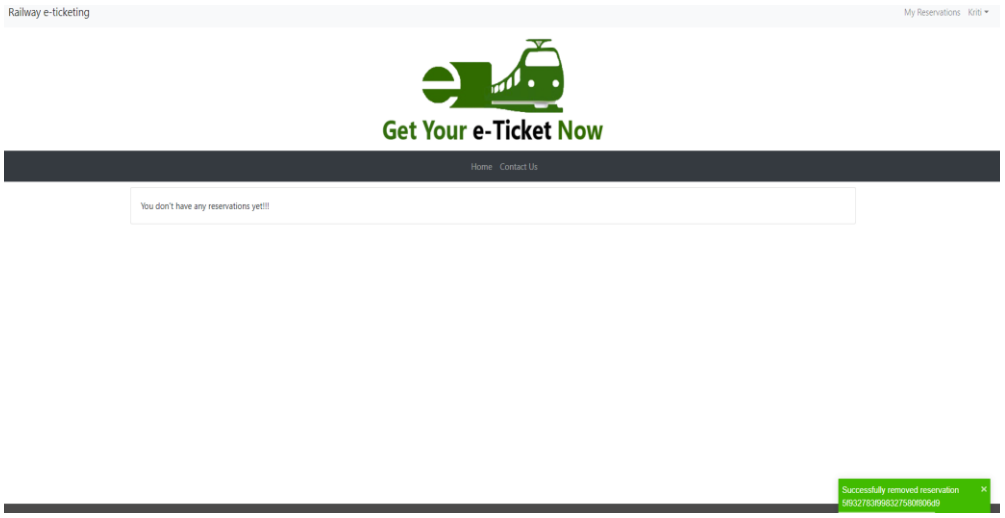
Class : 1st class

Quantity : 1

Amount : 84.00 Discount : 0.00 **Total :** 84.00

Successfully paid 84.00

Fig 14: cancel reservation



You don't have any reservations yet!!!

Successfully removed reservation 5f932783f998327580f806d9

Fig 15: cancel reservation confirmation

Users can edit their profile through the link in the navigation bar.

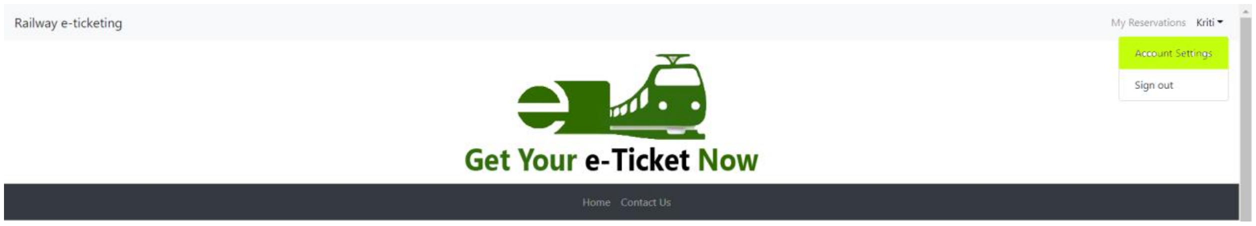
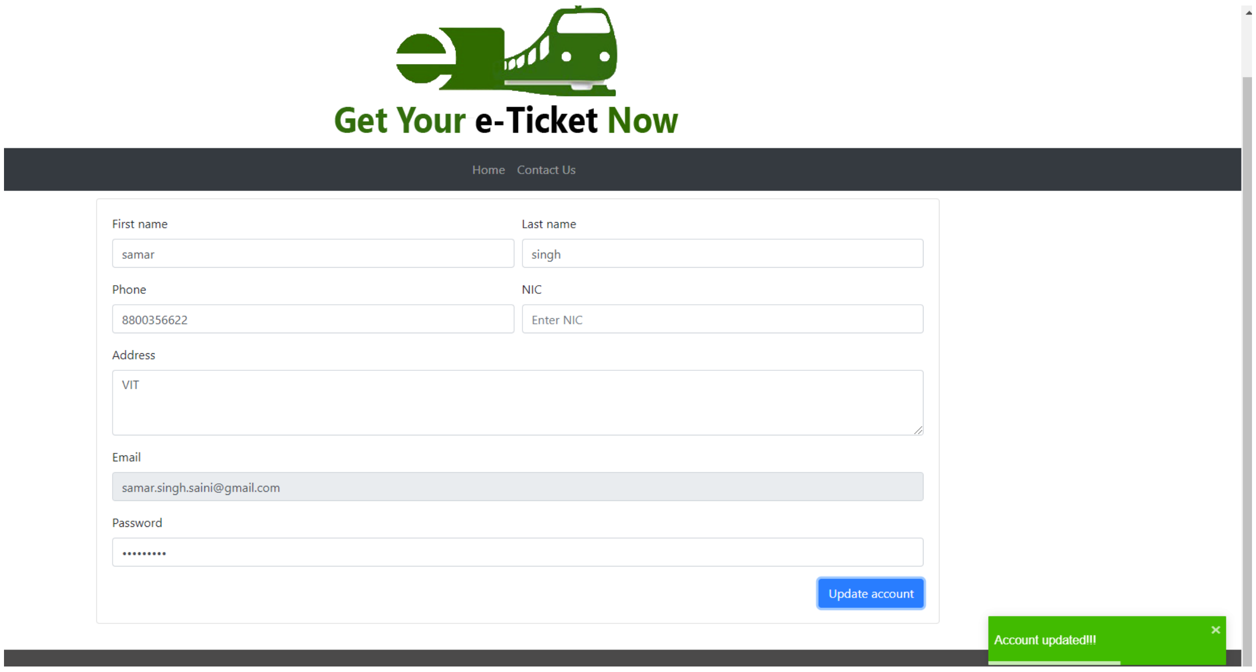


Fig 16: change profile data

Users cannot change the username (email).


Get Your e-Ticket Now

Home Contact Us

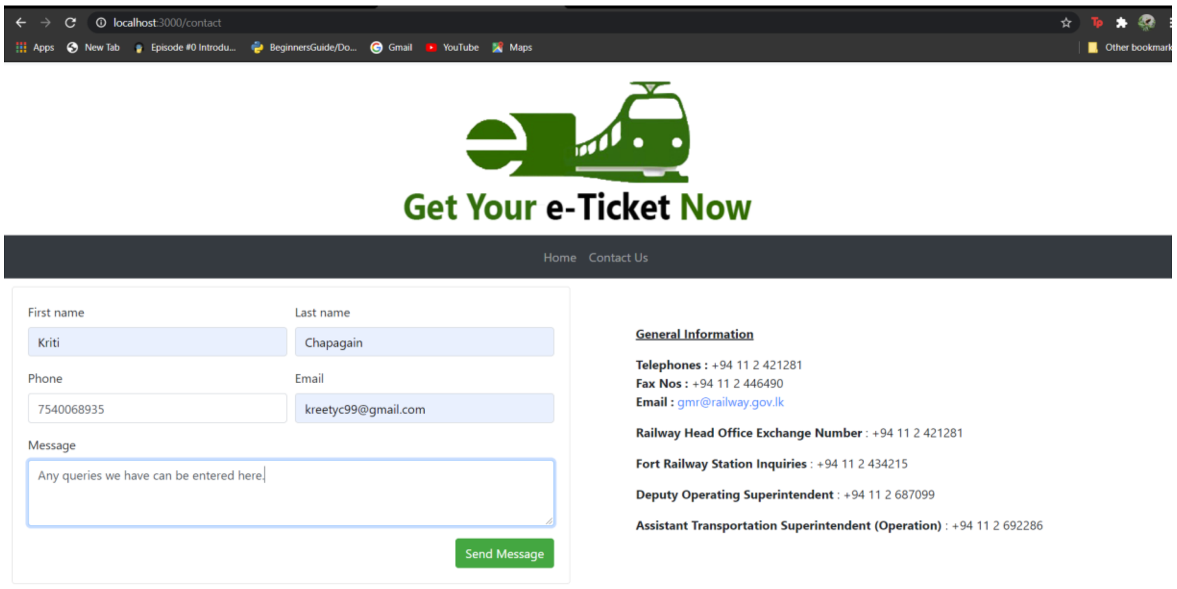
First name: samar
Last name: singh
Phone: 8800356622
NIC: Enter NIC
Address: VIT
Email: samar.singh.saini@gmail.com
Password: •••••••••

Update account

Account updated!!!

Fig 17: successfully change profile data

Users can view contact details from "Contact Us" page or they can send a message to support team regarding any of their issues. Once user send support request an email will send to both the user and the support team.


localhost:3000/contact

Get Your e-Ticket Now

Home Contact Us

First name: Kriti
Last name: Chapagain
Phone: 7540068935
Email: kreetyc99@gmail.com
Message: Any queries we have can be entered here

Send Message

General Information

Telephones : +94 11 2 421281
Fax Nos : +94 11 2 446490
Email : gmr@railway.gov.lk

Railway Head Office Exchange Number : +94 11 2 421281

Fort Railway Station Inquiries : +94 11 2 434215

Deputy Operating Superintendent : +94 11 2 687099

Assistant Transportation Superintendent (Operation) : +94 11 2 692286

Fig 18: contact us page

6. HEURISTIC EVALUATION

HOME PAGE:

The home page provides all the necessary information required for the user to book a ticket, login or signup, and offers a minimalistic design.

REGISTRATION PAGE:

The registration page is clean, and shows only important information

SIGN IN PAGE:

The credentials to sign up are easy.

PAYMENT PAGE:

It provides a simple design with all important information, such as type of card, and mode of payment

BOOKING PAGE:

System design consistency is maintained, and the user is also notified about the recent payment, in the bottom right corner

CANCELLATION PAGE:

Similarly, the user is informed about his/her recent cancellation of the booking

6.1 PARAMETERS CONSIDERED IN OUR HEURISTIC EVALUATION

1.Visibility of system status:

The necessary information is all available on the home page of the website, such as login, sign up, and booking the train tickets straight away, with no unnecessary distractions. Furthermore, after performing certain functions in the website, a notification pops up that informs the user of the action that had been performed, and whether it was successful, such as payment successful and login successful .

2.Match between system and the real world:

The user can easily get to know what is going on in the website, and can navigate easily through the website, with clear icons and language being used.

3.User control and freedom:

The user can navigate through various options such as search for trains, register or login, PNR status, and other such essential options required in a website homepage

4.Consistency and standards:

The website remains consistent in the design implementation across all pages, which makes it easier for the user, as well as makes it look more aesthetic to the user

5.Error prevention:

Wherever possible, the necessary steps were taken to reduce errors such as in the payment gateway, as well as the booking window, which is more susceptible to errors

6.Recognition rather than recall:

Due to the system wide consistency of the website, the user finds it easier to navigate. Also, the icons used are easily interpreted by the user, and the minimalist design makes the best possible recognition technique for the user

7.Flexibility and efficiency of use:

Although there are no specified shortcuts or extra implementation for making hidden commands, the website is overall easy to navigate through, and only requires a single path to be taken by the user in most cases. Due to this linearity of the process of booking a train ticket, requiring shortcuts and hidden functions is not necessary

8.Aesthetic and minimalist design:

The website shows the highest level of minimalism, with only necessary options available, and clearly navigate-able buttons. The icons are also well refined and are pleasing to view for the user

9. Help users recognize, diagnose and recover from errors:

The website informs the user whether functions such as payment, login, signup, booking confirmation has been successful or not. If the function has not been successful, then the user can navigate to the previous page to re perform the function.

10. Help and documentation:

The website provides an email id to which the user can email, regarding his/her concern

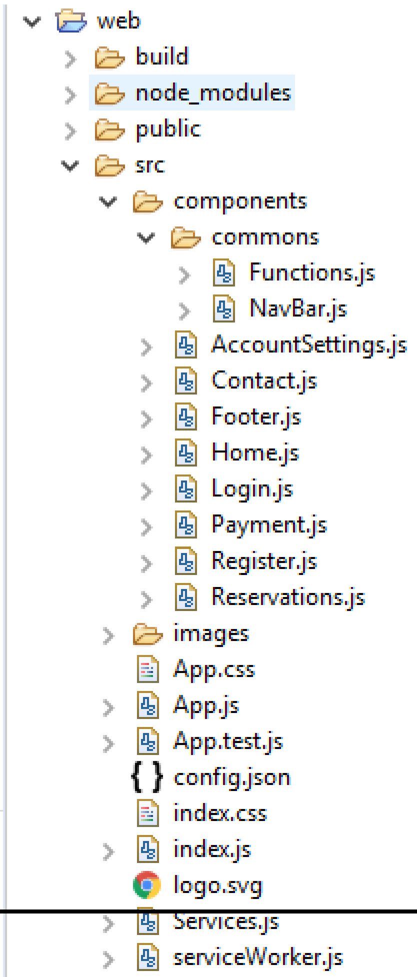
CONCLUSION:

Hence, after taking into all these considerations, it can be noted that the website ‘Railway Reservation System’ is meeting majority of the parameters, and offers a system wide, easy and user friendly approach to booking a train, and is far more superior than the existing IRCTC booking website, and excels in areas such as design implementation, in which the IRCTC website is fairly lacking.

6. Appendix

6.1. Front-End (ReactJS)

Following figure shows the folder structure of the web (front-end) component.



```
web
  build
  node_modules
  public
  src
    components
      commons
        Functions.js
        NavBar.js
      AccountSettings.js
      Contact.js
      Footer.js
      Home.js
      Login.js
      Payment.js
      Register.js
      Reservations.js
    images
    App.css
    App.js
    App.test.js
    config.json
    index.css
    index.js
    logo.svg
    Services.js
    serviceWorker.js
```

Fig 19: front-end folder structure

- **Config file (/web/src/config.json)**

This is where the configs of web application are stored. The `"baseUrl"` is where the back-end services deployed. In this case I have given the URL of WSO2 EI API base URL. If you have the WSO2 server in a separate location, you have to simply change the URL in the config file and deploy the application.

```
{
    "baseUrl": "http://localhost:8280"
}
```

- **Index component (/web/src/index.js)**

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

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

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

- **App component (/web/src/App.js)**

```
import React, { Component, Suspense } from 'react'
import 'bootstrap/dist/css/bootstrap.min.css'
import 'react-toastify/dist/ReactToastify.css'
import 'react-datepicker/dist/react-datepicker.css'
import './App.css'
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom'
import { ToastContainer } from 'react-toastify'

import NavBar from './components/commons/NavBar'
import Login from './components/Login'
import Register from './components/Register'
import Home from './components/Home'
import Contact from './components/Contact'
import Reservations from './components/Reservations'
import Payment from './components/Payment'
import AccountSettings from './components/AccountSettings'
import Footer from './components/Footer'


class App extends Component {
  constructor(props, context) {
    super(props, context)

 this.state = {
      showLogin: false,
      showRegister: false
    }

    this.config = {
      selected: 'home'
    }

    this.baseState = this.state
  }

  handleChange = obj => {
    if (obj instanceof Object) {
      this.setState({ ...obj })
    }
  }

  handleLogout = () => {
    this.setState(this.baseState)
    localStorage.clear()
  }

  handleLoginShow = () => {
    this.setState({ showLogin: true })
  }

  handleLoginClose = () => {
    this.setState({ showLogin: false })
  }

  handleRegisterShow = () => {
    this.setState({ showRegister: true })
  }

  handleRegisterClose = () => {
    this.setState({ showRegister: false })
  }

  render() {

    return (
      <>
        <div className="main-container">
          <NavBar
            handleLoginShow={this.handleLoginShow}
            handleRegisterShow={this.handleRegisterShow}
            logout={this.handleLogout}
            {...this.state}
          />

          <Login
            showLogin={this.state.showLogin}
            handleShow={this.handleLoginShow}
            handleClose={this.handleLoginClose}
            handleRegisterShow={this.handleRegisterShow}
          />

          <Register
            showRegister={this.state.showRegister}
            handleShow={this.handleRegisterShow}
            handleClose={this.handleRegisterClose}
            handleLoginShow={this.handleLoginShow}
          />

          <Router>
            <Suspense fallback={<div>Loading...</div>}>
              <Switch>
                <Route exact path="/" component={Home} />
                <Route path="/contact" component={Contact} />
                <Route path="/reservations" component={Reservations} />
                <Route path="/payment" component={Payment} />
                <Route path="/account" component={AccountSettings} />
              </Switch>
            </Suspense>
          </Router>
        </div>

        <Footer />

        <ToastContainer
          autoClose={3000}
          position="bottom-right"
        />
      </>
    );
  }
}

export default App;
```

- **Styles (/web/src/App.css)**

```
.App {
  text-align: center;
}

.App-logo {
  animation: App-logo-spin infinite 20s linear;
  height: 40vmin;
  pointer-events: none;
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

.react-datepicker-wrapper, .react-datepicker__input-container {
  display: block;
}

body {
  height: 100%;
  position: relative;
 }

 * {
  box-sizing: border-box;
 }

 *:before,
 *:after {
  box-sizing: border-box;
 }

 .main-container {
  min-height: 100vh; /* will cover the 100% of viewport */
  overflow: hidden;
  display: block;
  position: relative;
  padding-bottom: 80px; /* height of your footer */
 }

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

- **Login component (/web/src/components/Login.js)**

```
import React, { Component } from 'react'

import { Modal, Button, Form, Image, Row } from 'react-bootstrap'
import { login } from '../Services'
import { getHash } from './commons/Functions'

class Login extends Component {

    constructor(props, context) {
        super(props, context)
        this.state = {
            // validated: false,
            modalShowErr: false,
            modalErrMsg: "Incorrect username or password!!!",
            username: "",
            password: ""
        }
        this.baseState = this.state
    }

    componentWillUnmount() {
        this.setState(this.baseState)
    }

    handleChange = type => event => {
        let value = event;
        if (event.target) {
            value = event.target.value;
        }
        this.setState({ [type]: value })
    }

    handleSubmit = event => {
        this.setState({ modalShowErr: false })
        const form = event.currentTarget

        if (form.checkValidity() === true) {
            login({ username: this.state.username, password:
getHash(this.state.password) })
                .then(res => {
                    localStorage.setItem('user', JSON.stringify(res))
                    this.props.handleClose()
                })
                .catch(err => {
                    console.log(err)
                    this.setState({ modalShowErr: true })
                })
        }
        event.preventDefault()
        event.stopPropagation()
    }

    joinClick = () => {
        this.props.handleClose()
        this.props.handleRegisterShow()
    }

    render() {
        return (
            <Modal show={this.props.showLogin} onHide={this.props.handleClose}>
                <Form onSubmit={e => this.handleSubmit(e)}>
                    <Modal.Header closeButton>
                    </Modal.Header>
                    <Modal.Body>
                        <Row style={{ alignItems: 'center', justifyContent: 'center'
}}>
                            <Image src={require("../images/login.png")} width='30%'
/>
                        </Row>
                        <Form.Group controlId="formBasicEmail">
                            <Form.Label>Email</Form.Label>
                            <Form.Control required type="username" placeholder="Enter
email" onChange={this.handleChange('username')} />
                        </Form.Group>

                        <Form.Group controlId="formBasicPassword">
                            <Form.Label>Password</Form.Label>
                            <Form.Control required type="password" placeholder="Enter
Password" onChange={this.handleChange('password')} />
                        </Form.Group>
                        {this.state.modalShowErr && <p style={{ color: 'red'
}}>{this.state.modalErrMsg}</p>}
                        <Button variant="primary" type="submit" block>
                            Sign in
                        </Button>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button variant="light" block onClick={this.joinClick}>
```

```
                            Join Now
                        </Button>
                    </Modal.Footer>
                </Form>
            </Modal>
        );
    }
}

export default Login;
```

- **Register component (/web/src/components/Register.js)**

```
import React, { Component } from 'react'

import { Modal, Button, Form, Col } from 'react-bootstrap'
import { register } from '../Services'
import { toast } from 'react-toastify'
import { getHash } from './commons/Functions'

class Register extends Component {

    constructor(props, context) {
        super(props, context)
        this.state = {
            // validated: false,
            modalShowErr: false,
            modalErrMsg: "Entered email already exist!!!",
        }
        this.baseState = this.state
    }

    componentWillUnmount() {
        this.setState(this.baseState)
    }

    handleChange = type => event => {
        let value = event
        if (event.target) {
            value = event.target.value
        }
        this.setState({ [type]: value })
    }

    handleSubmit = event => {
        this.setState({ modalShowErr: false })
        const form = event.currentTarget

        if (form.checkValidity() === true) {
            var body = { ...this.state, password: getHash(this.state.password) }
            register(body)
                .then(res => {
                    toast.success("Account Created Please Sign In")
                    this.loginClick()
                })
                .catch(err => {
                    if (err.then && typeof err.then === 'function') {
                        err.then(e => {
                            toast.error("Unable to register the new user")
                            if (e.exist) {
                                this.setState({ modalShowErr: true })
                            }
                        })
                    } else {
                        console.log(err)
                    }
                })
        }
        // this.setState({ validated: true })
        event.preventDefault()
        event.stopPropagation()
    }

    loginClick = () => {
        this.props.handleClose()
        this.props.handleLoginShow()
    }

    render() {

        return (
            <Modal show={this.props.showRegister} onHide={this.props.handleClose}>
                <Form onSubmit={e => this.handleSubmit(e)}>
                    <Modal.Header closeButton>
                    </Modal.Header>
                    <Modal.Body>
                        <Form.Row>
                            <Form.Group as={Col} controlId="formGridFName">
                                <Form.Label>First name</Form.Label>
                                <Form.Control required type="username"
placeholder="Enter first name" onChange={this.handleChange('fname')} />
                            </Form.Group>

                            <Form.Group as={Col} controlId="formGridLName">
                                <Form.Label>Last name</Form.Label>
                                <Form.Control required type="username"
placeholder="Enter last name" onChange={this.handleChange('lname')} />
                            </Form.Group>
                        </Form.Row>
                        <Form.Group controlId="formGridPhone">
                            <Form.Label>Phone</Form.Label>
                            <Form.Control required type="username" placeholder="Enter
Phone Number" onChange={this.handleChange('phone')} />
                        </Form.Group>
                        <Form.Group controlId="formGridNIC">
                            <Form.Label>NIC</Form.Label>
                            <Form.Control type="username" placeholder="Enter NIC
(Optional)" onChange={this.handleChange('nic')} />
                        </Form.Group>
                        <Form.Group controlId="controlTextarea1">
                            <Form.Label>Address</Form.Label>
                            <Form.Control required as="textarea" rows="3"
onChange={this.handleChange('address')} />
                        </Form.Group>
                        <Form.Group controlId="formGridEmail">
                            <Form.Label>Email</Form.Label>
                            <Form.Control required type="email" placeholder="Enter
email" onChange={this.handleChange('email')} />
                        </Form.Group>
                        <Form.Group controlId="formBasicPassword">
                            <Form.Label>Password</Form.Label>
                            <Form.Control required type="password" placeholder="Enter
Password" onChange={this.handleChange('password')} />
                        </Form.Group>
                        {this.state.modalShowErr && <p style={{ color: 'red'
}}>{this.state.modalErrMsg}</p>}
                        <Button variant="primary" type="submit" block>
                            Create account
                        </Button>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button variant="light" block onClick={this.loginClick}>
                            Sign in
                        </Button>
                    </Modal.Footer>
                </Form>
            </Modal >
        )
    }
}

export default Register;
```

- **Navigation Bar (/web/src/components/commons/NavBar.js)**

```
import React, { Component } from 'react'

import { Navbar, Nav, NavDropdown, Image, Row } from 'react-bootstrap'

class NavBar extends Component {

    render() {
        var user = localStorage.getItem('user')
        if (user) {
            user = JSON.parse(user)
        }
        return (
            <>
                <Navbar bg="light" expand="sm">
                    <Navbar.Brand href="/">
                        Railway e-ticketing
                    </Navbar.Brand>
                    <Navbar.Toggle aria-controls="basic-navbar-nav" />
                    <Navbar.Collapse id="basic-navbar-nav">
                        <Nav className="ml-auto">
                            {user ?
                                <>
                                    <Nav.Link href="/reservations" >My
Reservations</Nav.Link>
                                    <NavDropdown title={user.fname} id="nav-dropdown"
alignRight>
                                        <NavDropdown.Item href="/account">Account
Settings</NavDropdown.Item>
                                        <NavDropdown.Divider />
                                        <NavDropdown.Item
onClick={this.props.logout}>Sign out</NavDropdown.Item>
                                    </NavDropdown>
                                </>
                                :
                                <>
                                    <Nav.Link href=""
onClick={this.props.handleLoginShow}>Sign In</Nav.Link>
                                    <Nav.Link href=""
onClick={this.props.handleRegisterShow}>Join Now</Nav.Link>
                                </>

                            }
                        </Nav>
                    </Navbar.Collapse>
                </Navbar>

                <Row style={{ alignItems: 'center', justifyContent: 'center', width:
'100%' }}>
                    <div >
                        <Image src={require("../../images/railway.png")} />
                    </div>
                </Row>

                <Navbar style={{ justifyContent: 'space-between' }} bg="dark"
variant="dark" expand="sm">
                    <Navbar.Brand href="/"></Navbar.Brand>
                    <Navbar.Toggle aria-controls="basic-navbar-nav" />
                    <Navbar.Collapse id="basic-navbar-nav" data-collapsed="false">
                        <Nav className="mx-auto">
                            <Nav.Link href="/" >{'Home'}</Nav.Link>
                            <Nav.Link href="/contact">{'Contact Us'}</Nav.Link>
                        </Nav>
                    </Navbar.Collapse>
                </Navbar>
            </>
        );
    }
}

export default NavBar;
```

- **Services component (/web/src /Services.js)**

All the functions of service calls to the back-end are stored in this component

```
import config from './config.json'

const baseUrl = config.baseUrl

export function login(body) {
    return callPost(baseUrl + '/login', body);
}

export function register(body) {
    return callPost(baseUrl + '/register', body);
}

export function routes() {
    return callGet(baseUrl + '/railway/routes');
}

export function route(station) {
    return callGet(baseUrl + '/railway/route/' + station);
}

export function trains() {
    return callGet(baseUrl + '/railway/trains/');
}

export function trainsByRoute(route) {
    return callGet(baseUrl + '/railway/trains/' + route);
}

export function classes() {
    return callGet(baseUrl + '/railway/classes/');
}
```

```
export function schedules() {
    return callGet(baseUrl + '/railway/schedules/');
}

export function validateCard(body) {
    return callPost(baseUrl + '/payment/card', body);
}

export function validatePhone(body) {
    return callPost(baseUrl + '/payment/phone', body);
}

export function makeReservation(body) {
    return callPost(baseUrl + '/railway/reservations', body);
}


export function getReservations(user) {
    return callGet(baseUrl + '/railway/reservations/' + user);
}

export function deleteReservation(id) {
    return callDelete(baseUrl + '/railway/reservations/' + id);
}

export function updateAccount(body, id) {
    return callPut(baseUrl + '/users/' + id, body)
}

export function contact(body) {
    return callPost(baseUrl + '/railway/contact', body);
}

const callGet = (url) => {
    return fetch(url).then(handleres);
}

const callPost = (url, body) => {
    return fetch(url, {
        method: 'POST',
        body: JSON.stringify(body),
        headers: { "Content-Type": "application/json" }
    }).then(handleres);
}

const callPut = (url, body) => {
    return fetch(url, {
        method: 'PUT',
        body: JSON.stringify(body),
        headers: { "Content-Type": "application/json" }
    }).then(handleres);
}

const callDelete = (url) => {
    return fetch(url, {
        method: 'DELETE'
    }).then(handleres);
}

const handleres = (res) => {
    if (res.ok) {
        return res.json();
    }
    else {
        if (res.status === 404) {
            return Promise.reject();
        } else {
            throw res.json();
        }
    }
}
```

- **Home component (/web/src/components/Register.js)**

```
import React, { Component } from 'react'
import { routes, route, trainsByRoute, classes, schedules } from '../Services'

import { Button, Form, Col, Row, Table } from 'react-bootstrap'
import Select from 'react-select'
import DatePicker from "react-datepicker"
import moment from 'moment'

class Home extends Component {

    constructor(props) {
        super(props);
        this.state = {
            fromOptions: [],
            toOptions: [],
            trains: [],
            errMsg: 'Please fill all the fields!!!',
            showErr: false,
        };
    }

    componentDidMount() {
        var options = []
        routes()
            .then(res => {
                res.map((item, i) => {
                    return item.route.map((station, i) => {
                        return options.push({ value: station.name, label:
station.name, route: item._id, id: i, fair: station.fair })
                    })
                })
                this.setState({ fromOptions: options })
            })
            .catch(err => {
                console.log(err)
            })
        classes()
            .then(res => {
                var classes = []
                res.map((trainClass, i) => {
                    return classes.push({ value: trainClass.name, label:
trainClass.name, id: trainClass._id, fairRatio: trainClass.fairRatio })
                })
                this.setState({ classes: classes })
            })
            .catch(err => {
                console.log(err)
            })
        schedules()
            .then(res => {
                var schedules = []
                res.map((schedule, i) => {
                    return schedules.push({ value: schedule.time, label:
schedule.time, id: schedule._id })
                })
                this.setState({ schedules: schedules })
            })
            .catch(err => {
                console.log(err)
            })

    }

    handleChange = type => selectedOption => {
        this.setState({ [type]: selectedOption }, () => {
            this.calculateFair()
        });
        if (type === 'from') {
            this.setState({ to: '', train: '' })
            route(selectedOption.route)
                .then(res => {
                    var options = [];
                    res.route.map((station, i) => {
                        return options.push({ value: station.name, label:
station.name, route: res._id, id: i, fair: station.fair })
                    })
                    this.setState({ toOptions: options })
                })
                .catch(err => {
                    console.log(err)
                })
            trainsByRoute(selectedOption.route)
                .then(res => {
                    var options = [];
                    res.map((train, i) => {
                        return options.push({ value: train.name, label: train.name,
id: train._id })
                    })
                    this.setState({ trains: options })
                })
                .catch(err => {
                    console.log(err)
                })
        }
    }

    handleQtyChange = () => event => {
        this.setState({ qty: event.target.value }, () => this.calculateFair())
    }

    calculateFair = () => {
        var user = localStorage.getItem('user')
        if (user) {
            user = JSON.parse(user)
        }
        if (this.state.to && this.state.from && this.state.trainClass &&
this.state.qty) {
            var amount = Math.abs(this.state.to.fair - this.state.from.fair) *
this.state.trainClass.fairRatio * this.state.qty
            amount = amount.toFixed(2)
            var discount = (user && user.discount ? 0.1 * amount : 0).toFixed(2)
            var total = (amount - discount).toFixed(2)
            this.setState({ amount: amount, discount: discount, total: total })
        }
    }

    handleSubmit = event => {
        this.setState({ showErr: false })
        const state = this.state
        var user = localStorage.getItem('user')
        if (!user) {
            alert("Please Sign In Before Make a Reservation!!!")
            this.props.history.push('/')
        } else if (state.from && state.to && state.train && state.trainClass &&
state.time && state.qty && state.date) {
            this.props.history.push("/payment", { ...this.state })
        } else {
            this.setState({ showErr: true })
        }
        event.preventDefault()
        event.stopPropagation()

    }

    handleDateChange = dt => {
        const date = moment(dt).format('YYYY-MM-DD')
        this.setState({ date: date })
    }

    render() {
        return (
            <Form style={{ padding: 20 }} onSubmit={(e) => this.handleSubmit(e)}>
                <Row style={{ alignItems: 'center', justifyContent: 'center' }}>
                    <Form.Row style={{ width: '75%', borderBottom: '1px solid
rgb(200,200,200)', marginBottom: 20 }}>
                        <h4>Book Train Tickets</h4>
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        <Form.Group as={Col} controlId="from">
                            <Form.Label>From</Form.Label>
                            <Select options={this.state.fromOptions}
onChange={this.handleChange("from")} />
                        </Form.Group>
                        <Form.Group as={Col} controlId="to">
                            <Form.Label>To</Form.Label>
                            <Select options={this.state.toOptions}
onChange={this.handleChange("to")} value={this.state.to} />
                        </Form.Group>
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        <Form.Group as={Col} controlId="from">
                            <Form.Label>Train</Form.Label>
                            <Select options={this.state.trains}
onChange={this.handleChange("train")} value={this.state.train} />
                        </Form.Group>
                        <Form.Group as={Col} controlId="to">
                            <Form.Label>Class</Form.Label>
                            <Select options={this.state.classes}
onChange={this.handleChange("trainClass")} value={this.state.trainClass} />
                        </Form.Group>
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        <Form.Group as={Col} controlId="from">
                            <Form.Label>Time</Form.Label>
                            <Select options={this.state.schedules}
onChange={this.handleChange("time")} value={this.state.time} />
                        </Form.Group>
                        <Form.Group as={Col} controlId="formGridEmail">
                            <Form.Label>No of Tickets</Form.Label>
                            <Form.Control placeholder="qty"
onChange={this.handleQtyChange()} />
                        </Form.Group>
                    </Form.Row>
                    <Form.Row style={{ width: '75%', paddingBottom: 20 }}>
                        <Col md={6} lg={6} xl={6}>
                            <Form.Label>Date</Form.Label>
                            <DatePicker
                                className="form-control"
                                onChange={this.handleDateChange}
                                minDate={new Date()}
                                value={this.state.date}
                                placeholderText="YYYY-MM-DD"
                            />
                        </Col>
                    </Form.Row>
                    <Form.Row style={{ width: '75%', paddingLeft: 5, align: 'right'
}}>
                        {this.state.amount &&
                            <Table striped bordered hover size="sm">
                                <tbody>
                                    <tr>
```

```
                                        <td align='right'>Amount</td>
                                        <td align='right'>{this.state.amount}
INR</td>
                                    </tr>
                                    <tr>
                                        <td align='right'>Discount</td>
                                        <td align='right'>{this.state.discount}
INR</td>
                                    </tr>
                                    <tr>
                                        <td align='right'>Total</td>
                                        <td align='right'>{this.state.total} INR</td>
                                    </tr>
                                </tbody>
                            </Table>
                        }
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        {this.state.showErr && <p style={{ color: 'red'
}}>{this.state.errMsg}</p>}
                    </Form.Row>
                    <Form.Row style={{ width: '75%', padding: 5 }}>
                        <Button variant="primary" type="submit">
                            Make Reservation
                        </Button>
                    </Form.Row>
                </Row>
            </Form >
        );
    }
}

export default Home;
```

- **Payment component (/web/src/components/Payment.js)**

```
import React, { Component } from 'react'

import { Table, Row, Form, Col, Button } from 'react-bootstrap'
import { validateCard, validatePhone, makeReservation } from '../Services'
import { toast } from 'react-toastify'

class Payment extends Component {

    constructor(props) {
        super(props);
        this.state = {
            checked: 'card',
            errMsg: 'Please fill all the fields!!!',
            showPaymentErr: false,
            validateErrMsg: 'Entered data not valid!!!',
            showValidateErr: false,
            cardNo: '',
            cvc: '',
            exp: '',
            phoneNo: '',
            pin: ''
        };
    }

    componentDidMount() {
        if (this.props.location) {
            this.setState({ ...this.props.location.state })
        }
        var user = localStorage.getItem('user')
        if (user) {
            this.setState({ phoneNo: JSON.parse(user).phone })
        }
    }

    componentWillUpdate() {
        var user = localStorage.getItem('user')
        if (!user) {
            this.props.history.push('/')
        }
    }

    handleChange = type => event => {
        var value = event.target.value
        if (type === 'card' || type === 'phone') {
            this.setState({ checked: type })
        } else {
            this.setState({ [type]: value })
        }
    }

    handleSubmit = async  event => {
        event.preventDefault()
        event.stopPropagation()
        this.setState({ showPaymentErr: false, showValidateErr: false })
        const state = this.state;
        if (state.checked === 'card') {
            if (state.cardNo && state.cvc && state.exp) {
                validateCard({ card: state.cardNo, cvc: state.cvc, exp: state.exp,
total: state.total })
                    .then(res => {
                        if (res.validated) {
                            this.createReservation({ card: state.cardNo })
                        } else {
                            this.setState({ showValidateErr: true })
                        }
                    })
                    .catch(err => {
                        console.log(err)
                    })

            } else {
                this.setState({ showPaymentErr: true })
            }
        }
        if (state.checked === 'phone') {
            if (state.phoneNo && state.pin) {
                validatePhone({ phone: state.phoneNo, pin: state.pin, total:
state.total })
                    .then(res => {
                        if (res.validated) {
                            this.createReservation({ phone: state.phoneNo })
                        } else {
                            this.setState({ showValidateErr: true })
                        }
                    })
                    .catch(err => {
                        console.log(err)
                    })
            } else {
                this.setState({ showPaymentErr: true })
            }
        }
    }

    createReservation = (paymentMethod) => {
        const state = this.state
        var user = localStorage.getItem('user')
        if (user) {
            user = JSON.parse(user)
            const reservation = {
                ...paymentMethod,
                user: user._id,
                email: user.email,
                from: state.from.value,
                to: state.to.value,
                train: state.train.value,
                trainClass: state.trainClass.value,
                time: state.time.value,
                qty: state.qty,
                date: state.date,
                amount: state.amount,
                discount: state.discount,
                total: state.total
            }
            makeReservation(reservation)
                .then(res => {
                    toast.success("Successfully paid " + reservation.total)
                    this.props.history.push('/reservations')
                })
                .catch(err => {
                    console.log(err)
                })
        }

    }
    render() {
        return (
            <Form style={{ padding: 20 }} onSubmit={(e) => this.handleSubmit(e)}>
                <Row style={{ alignItems: 'center', justifyContent: 'center' }}>
                    <Form.Row style={{ width: '75%' }}>
                        <Table striped bordered hover size="sm">
                            <tbody>
                                <tr>
                                    <td align='right'>Amount</td>
                                    <td align='right'>{this.state.amount} INR</td>
                                </tr>
                                <tr>
                                    <td align='right'>Discount</td>
                                    <td align='right'>{this.state.discount} INR</td>
                                </tr>
                                <tr>
                                    <td align='right'>Total</td>
                                    <td align='right'>{this.state.total} INR</td>
                                </tr>
                            </tbody>
                        </Table>
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        <Form.Label as="legend">
                            Select a payment method
                        </Form.Label>
                    </Form.Row>
                    <Form.Row style={{ width: '75%', paddingBottom: 10 }}>
                        <Col>
                            <Form.Check
                                type="radio"
                                label="Credit Card"
                                name="formHorizontalRadios"
                                id="formHorizontalRadios1"
                                defaultChecked
                                onChange={this.handleChange('card')}
                            />
                            <Form.Check
                                type="radio"
                                label="Mobile Number"
                                name="formHorizontalRadios"
                                id="formHorizontalRadios2"
                                onChange={this.handleChange('phone')}
                            />
                        </Col>
                    </Form.Row>
                    {this.state.checked === 'card' &&
                        <Form.Row style={{ width: '75%' }}>
                            <Form.Group as={Col} controlId="cardNo">
                                <Form.Label>Card Number</Form.Label>
                                <Form.Control placeholder="card number"
onChange={this.handleChange('cardNo')} value={this.state.cardNo} />
                            </Form.Group>
                            <Form.Group as={Col} controlId="cvc">
                                <Form.Label>CVC Number</Form.Label>
                                <Form.Control placeholder="CVC"
onChange={this.handleChange('cvc')} value={this.state.cvc} />
                            </Form.Group>
                            <Form.Group as={Col} controlId="exp">
                                <Form.Label>Exp date</Form.Label>
                                <Form.Control placeholder="dd/mm"
onChange={this.handleChange('exp')} value={this.state.exp} />
                            </Form.Group>
                        </Form.Row>
                    }
                    {this.state.checked === 'phone' &&
                        <Form.Row style={{ width: '75%' }}>
                            <Form.Group as={Col} controlId="phoneNo">
                                <Form.Label>Phone Number</Form.Label>
                                <Form.Control placeholder="Phone number"
onChange={this.handleChange('phoneNo')} value={this.state.phoneNo} />
                            </Form.Group>
                            <Form.Group as={Col} controlId="pin">
                                <Form.Label>PIN</Form.Label>
                                <Form.Control placeholder="PIN"
onChange={this.handleChange('pin')} value={this.state.pin} />
                            </Form.Group>
                        </Form.Row>
                    }
                    <Form.Row style={{ width: '75%' }}>
                        {this.state.showPaymentErr && <p style={{ color: 'red'
}}>{this.state.errMsg}</p>}
                        {this.state.showValidateErr && <p style={{ color: 'red'
}}>{this.state.validateErrMsg}</p>}
                    </Form.Row>
                    <Form.Row style={{ width: '75%' }}>
                        <Button variant="primary" type="submit">
                            Make Payment
                        </Button>
                    </Form.Row>
                </Row>
            </Form>
        )
    }
}

export default Payment
```

- **Reservations Component (/web/src/components/Reservations.js)**

```
import React, { Component } from 'react';

import { Row, Col, Button, Card, Pagination } from 'react-bootstrap'
import { getReservations, deleteReservation } from '../Services'
import { toast } from 'react-toastify'

class Reservations extends Component {
    constructor(props) {
        super(props);
        this.state = {
            reservations: [],
            items: [],
            offset: 1,
            lastPage: 1,
            paginateItems: []
        };
    }

    componentDidMount() {
        this.updateReservations()
    }

    componentWillUpdate() {
        var user = localStorage.getItem('user')
        if (!user) {
            this.props.history.push('/')
        }
    }

    updateReservations = () => {
        var user = localStorage.getItem('user')
        if (!user) {
            this.props.history.push('/')
        } else {
            user = JSON.parse(user)
            getReservations(user._id)
                .then(res => {
                    this.setState({ reservations: res }, () =>
this.paginateReservations())
                })
                .catch(err => {
                    console.log(err)
                })
        }
    }

    cancelReservation = id => {
        var c = window.confirm("The reservation " + id + " will be deleted")
        if (c) {
            deleteReservation(id)
                .then(res => {
                    toast.success("Successfully removed reservation " + id)
                    this.updateReservations()
                })
                .catch(err => {
                    console.log(err)
                })
        }
    }

    paginateReservations = () => {
        let items = [];
        const offset = (this.state.offset - 1) * 5

        for (let number = offset; number < offset + 5; number++) {
            const reservation = this.state.reservations[number]
            if (reservation) {
                items.push(
                    <Row style={{ width: '75%' }} key={number}>
                        <Col>
                            <Card style={{ padding: 10, marginTop: 10 }}>
                                <Row>
                                    <Col>Reference No : {reservation._id}</Col>
                                </Row>
                                <hr />
                                <Row>
                                    <Col>From <b>{reservation.from}</b> to
<b>{reservation.to}</b></Col>
                                    <Col align='right'>{reservation.date}
{reservation.time}</Col>
                                </Row>
                                <Row>
                                    <Col>Train : {reservation.train}</Col>
                                </Row>
                                <Row>
                                    <Col>Class : {reservation.trainClass}</Col>
                                </Row>
                                <Row>
                                    <Col>Quantity : {reservation.qty}</Col>
                                </Row>
                                <hr />
                                <Row>
                                    <Col>Amount :
{reservation.amount.toFixed(2)}</Col>
                                    <Col>Discount :
{reservation.discount.toFixed(2)}</Col>
                                    <Col align='right'><b>Total :</b>
{reservation.total.toFixed(2)}</Col>
                                </Row>
                                <Row>
                                    <Col style={{ paddingTop: 10 }} align='right'>
                                        <Button variant="danger" size="sm"
onClick={() => this.cancelReservation(reservation._id)}>Cancel</Button>
                                    </Col>
                                </Row>
                            </Card>
                        </Col>
                    </Row>
                )
            }
        }
        let paginateItems = [];
        const lastPage = Math.ceil(this.state.reservations.length / 5)
        for (let number = 1; number <= lastPage; number++) {
            paginateItems.push(
                <Pagination.Item key={number} active={number === this.state.offset}
onClick={() => this.pageChange(number)}>
                    {number}
                </Pagination.Item>,
            );
        }
        this.setState({ paginateItems: paginateItems, items: items, lastPage:
lastPage })
    }

    pageChange = n => {
        console.log(n)
        this.setState({ offset: n }, () => this.paginateReservations())
    }

    render() {
        return (
            <Row style={{ alignItems: 'center', justifyContent: 'center', width:
'100%' }}>
                {this.state.reservations.length <= 0 &&
                    <Row style={{ width: '75%', padding: 10 }}>
                        <Col>
                            <Card>
                                <Card.Body>You don't have any reservations
yet!!!</Card.Body>
                            </Card>
                        </Col>
                    </Row>
                }
                {this.state.reservations.length > 0 &&
                    <>
                        <Row style={{ width: '75%', paddingTop: 20, paddingLeft: 15
}}>
                            <Pagination>
                                <Pagination.First onClick={() => this.pageChange(1)}
/>
                                {this.state.paginateItems}
                                <Pagination.Last onClick={() =>
this.pageChange(this.state.lastPage)} />
                            </Pagination>
                        </Row>
                        {this.state.items.map((reservation, i) => {
                            return (
                                reservation
                            )
                        })}
                        <Row style={{ width: '75%', paddingTop: 20, paddingLeft: 15
}}>
                            <Pagination>
                                <Pagination.First onClick={() => this.pageChange(1)}
/>
                                {this.state.paginateItems}
                                <Pagination.Last onClick={() =>
this.pageChange(this.state.lastPage)} />
                            </Pagination>
                        </Row>
                    </>
                }
            </Row>
        );
    }
}

export default Reservations;
```

- **Contact component (/web/src/components/Contact.js)**

```
import React, { Component } from 'react';

import { Col, Button, Form, Card, Row } from 'react-bootstrap'
import { contact } from '../Services'
import { toast } from 'react-toastify'

class Contact extends Component {

    constructor(props) {
        super(props);
        this.state = {
            fname: '',
            lname: '',
            phone: '',
            email: '',
            message: ''
        };
        this.baseState = this.state
    }

    handleChange = type => event => {
        let value = event
        if (event.target) {
            value = event.target.value
        }
        this.setState({ [type]: value })
    }

    handleSubmit = event => {
        event.preventDefault()
        event.stopPropagation()
        contact(this.state)
            .then(res => {
                toast.success("Your message has been sent..")
                this.setState({ ...this.baseState })
            })
            .catch(err => {
                console.log(err)
            })
    }

    render() {
        return (
            <Row style={{ alignItems: 'center', justifyContent: 'center' }}>
                <Col>
                    <Card style={{ padding: 20, margin: 10 }}>
                        <Form onSubmit={e => this.handleSubmit(e)}>
                            <Form.Row>
                                <Form.Group as={Col} controlId="formGridFName">
                                    <Form.Label>First name</Form.Label>
                                    <Form.Control required type="username"
placeholder="Enter first name" onChange={this.handleChange('fname')}
value={this.state.fname} />
                                </Form.Group>

                                <Form.Group as={Col} controlId="formGridLName">
                                    <Form.Label>Last name</Form.Label>
                                    <Form.Control required type="username"
placeholder="Enter last name" onChange={this.handleChange('lname')}
value={this.state.lname} />
                                </Form.Group>
                            </Form.Row>
                            <Form.Row>
                                <Form.Group as={Col} controlId="formGridPhone">
                                    <Form.Label>Phone</Form.Label>
                                    <Form.Control type="username" placeholder="Enter
Phone Number" onChange={this.handleChange('phone')} value={this.state.phone} />
                                </Form.Group>
                                <Form.Group as={Col} controlId="formGridEmail">
                                    <Form.Label>Email</Form.Label>
                                    <Form.Control required type="email"
placeholder="Enter email" onChange={this.handleChange('email')}
value={this.state.email} />
                                </Form.Group>
                            </Form.Row>
                            <Form.Group controlId="controlTextarea1">
                                <Form.Label>Message</Form.Label>
                                <Form.Control required as="textarea" rows="3"
onChange={this.handleChange('message')} value={this.state.message} />
                            </Form.Group>
                            <Col style={{ paddingRight: 0 }} align='right'>
                                <Button variant="success" type="submit">
                                    Send Message
                                </Button>
                            </Col>
                        </Form>
                    </Card>
                </Col>
                <Col>
                    <Row style={{ alignItems: 'center', justifyContent: 'center',
margin: 30 }}>
```

```
                        <Col>
                            <div id="page">
                                <p><strong><span style={{ textDecoration: 'underline'
}}>General Information</span></strong></p>
                                <p><strong>Telephones : </strong>+94 11 2 421281 <br
/><strong>Fax Nos : </strong>+94 11 2 446490<br /><strong>Email : </strong>
                                    <a
href="mailto:gmr@railway.gov.lk">gmr@railway.gov.lk</a>
                                    <span style={{ display: 'none' }}>This e-mail
address is being protected from spambots. You need JavaScript enabled to view it
                                        </span>
                                </p>
                                <p><strong>Railway Head Office Exchange
Number</strong> : +94 11 2 421281</p>
                                <p><strong>Fort Railway Station Inquiries</strong> :
+94 11 2 434215</p>
                                <p><strong>Deputy Operating Superintendent</strong> :
+94 11 2 687099</p>
                                <p className="MsoNormal"><strong>Assistant
Transportation Superintendent (Operation)</strong> : +94 11 2 692286</p>
                            </div>
                        </Col>
                    </Row>
                </Col>
            </Row>
        );
    }
}

export default Contact;
```

- **Account Settings component (/web/src/components/AccountSettings.js)**

```
import React, { Component } from 'react'

import { Col, Button, Form, Card, Row } from 'react-bootstrap'
import { updateAccount } from '../Services'
import { toast } from 'react-toastify'
import { getHash } from './commons/Functions'

class AccountSettings extends Component {

    constructor(props, context) {
        super(props, context)
        this.state = {
            fname: '',
            lname: '',
            phone: '',
            nic: '',
            email: '',
            address: ''
        }
        this.baseState = this.state
    }

    componentDidMount() {
        var user = localStorage.getItem('user')
        if (user) {
            user = JSON.parse(user)
            this.setState({
                fname: user.fname,
                lname: user.lname,
                phone: user.phone,
                nic: user.nic || '',
                email: user.email,
                address: user.address
            })
        }
    }

    componentWillUpdate() {
        var user = localStorage.getItem('user')
        if (!user) {
            this.props.history.push('/')
        }
    }

    handleChange = type => event => {
        let value = event
        if (event.target) {
            value = event.target.value
        }
        this.setState({ [type]: value })
    }

    handleSubmit = event => {
        const form = event.currentTarget
        const id = JSON.parse(localStorage.getItem('user'))._id
        if (form.checkValidity() === true) {
            var body = { ...this.state }
            if (this.state.password) {
                body = { ...body, password: getHash(this.state.password) }
            }
            updateAccount(body, id)
                .then(res => {
                    toast.success("Account updated!!!")
                    localStorage.setItem('user', JSON.stringify(res))
                })
                .catch(err => {
                    toast.error("Unable to update new data!!!")
                })
        }
        event.preventDefault()
        event.stopPropagation()
    }

    render() {
        return (
            <Row style={{ alignItems: 'center', justifyContent: 'center' }}>
                <Row style={{ width: '60%', padding: 10 }}>
                    <Col>
                        <Card style={{ padding: 20 }}>
                            <Form onSubmit={e => this.handleSubmit(e)}>
                                <Form.Row>
                                    <Form.Group as={Col} controlId="formGridFName">
                                        <Form.Label>First name</Form.Label>
                                        <Form.Control required type="username"
placeholder="Enter first name" onChange={this.handleChange('fname')}
value={this.state.fname} />
                                    </Form.Group>

                                    <Form.Group as={Col} controlId="formGridLName">
                                        <Form.Label>Last name</Form.Label>
                                        <Form.Control required type="username"
placeholder="Enter last name" onChange={this.handleChange('lname')}
value={this.state.lname} />
                                    </Form.Group>
                                </Form.Row>
                                <Form.Row>
                                    <Form.Group as={Col} controlId="formGridPhone">
                                        <Form.Label>Phone</Form.Label>
                                        <Form.Control required type="username"
placeholder="Enter Phone Number" onChange={this.handleChange('phone')}
value={this.state.phone} />
                                    </Form.Group>
                                    <Form.Group as={Col} controlId="formGridNIC">
                                        <Form.Label>NIC</Form.Label>
                                        <Form.Control type="username"
placeholder="Enter NIC" onChange={this.handleChange('nic')} value={this.state.nic} />
                                    </Form.Group>
                                </Form.Row>
                                <Form.Group controlId="controlTextarea1">
                                    <Form.Label>Address</Form.Label>
                                    <Form.Control required as="textarea" rows="3"
onChange={this.handleChange('address')} value={this.state.address} />
                                </Form.Group>
                                <Form.Group controlId="formGridEmail">
                                    <Form.Label>Email</Form.Label>
                                    <Form.Control required type="email"
placeholder="Enter email" onChange={this.handleChange('email')}
value={this.state.email} disabled />
                                </Form.Group>
                                <Form.Group controlId="formBasicPassword">
                                    <Form.Label>Password</Form.Label>
                                    <Form.Control type="password" placeholder="Enter
New Password" onChange={this.handleChange('password')} />
                                </Form.Group>
                                <Col style={{ paddingRight: 0 }} align='right'>
                                    <Button variant="primary" type="submit">
                                        Update account
                                    </Button>
                                </Col>
```

```
                              </Form>
                          </Card>
                      </Col>
                  </Row>
              </Row>
          )
      }
  }

  export default AccountSettings
```

- **Common functions (/web/src/components/commons/Functions.js)**

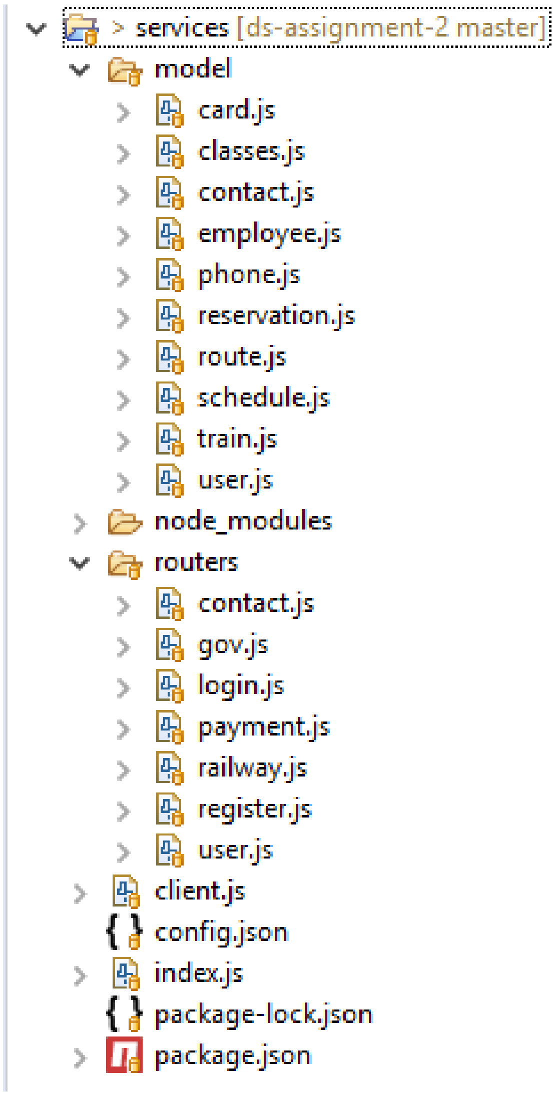
```
export function getHash(str) {
    let hash = 0
       for (let i = 0; i < str.length; i++) {
              hash += Math.pow(str.charCodeAt(i) * 31, str.length - i)
              hash = hash & hash // Convert to 32bit integer
       }
       return hash
};
```

- **Footer component (/web/src/components/Footer.js)**

```
import React, { Component } from 'react'

class Footer extends Component {

    render() {
        return (
            <footer className="page-footer font-small" style={{ backgroundColor:
'#4B4A4A', color: 'white', position: 'absolute', bottom: 0, width: '100%' }}>
                <div className="footer-copyright text-center py-3">© 2019 Copyright:
                    <a href="https://tenusha.wordpress.com">
tenusha.wordpress.com</a>
                </div>
            </footer>
        )
    }
}

export default Footer
```

6.2. Back-End (NodeJS, ExpressJS, MongoDB)

Following figure shows the folder structure of the services (back-end) component.



```
> services [ds-assignment-2 master]
    model
        card.js
        classes.js
        contact.js
        employee.js
        phone.js
        reservation.js
        route.js
        schedule.js
        train.js
        user.js
    node_modules
    routers
        contact.js
        gov.js
        login.js
        payment.js
        railway.js
        register.js
        user.js
    client.js
    config.json
    index.js
    package-lock.json
    package.json
```

Fig 20: back-end folder structure

- **Config file (/services/config.json)**

This is where all the configs of the back-end are stored.

```
{
    "mongoDB": "mongodb://localhost/railway",
    "govAPI": "http://localhost:3001/gov/employee/",
    "emailClient": {
        "host": "smtp.gmail.com",
        "email": "sl.railway.e.ticketing@gmail.com",
        "auth": {
            "user": "sl.railway.e.ticketing@gmail.com",
            "pass": "railway@123"
        }
    },
    "messageClient": {
        "accountSid": "AC86b7448c3ed5e78b18f44d5e84fbdcb1",
        "authToken": "fee993da9d4cafa918929607a8b37827",
        "phoneNo": "+18504040553"
    }
}
```

It contains the URL of government service to validate NIC of users, email client information and the configs of Twilio text message service. If you want to use a premium Twilio account, you only have to change the configs in this file.

- **Index component (/services/index.js)**

```
'use strict'
const express = require('express')
const app = express()
const config = require('./config.json')
const login = require('./routers/login')
const register = require('./routers/register')
const railway = require('./routers/railway')
const payment = require('./routers/payment')
const gov = require('./routers/gov')
const user = require('./routers/user')
const contact = require('./routers/contact')
const mongoose = require('mongoose')

mongoose.connect(config.mongoDB, { useNewUrlParser: true }, function (err) {
    if (err) throw err
    console.log('mongo db connected')
}).catch(err => console.log(err))

app.use(express.json());
app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
    res.header("Access-Control-Allow-Methods", "GET, POST, OPTIONS, PUT, DELETE");
    next();
});

app.use(login)
app.use(register)
app.use(railway)
app.use(payment)
app.use(gov)
app.use(user)
app.use(contact)

app.listen(3001, err => {
    if (err) {
        console.error(err)
        return
    }
    console.log('app listening on port 3001')
});
```

- **Client component (/services/Client.js)**

This is where all the connections to the outside network are stored. (government service, email service and Twilio text message service)

```
const fetch = require("node-fetch")
const nodemailer = require('nodemailer')
const config = require('./config.json')

const twilio = require('twilio')(config.messageClient.accountSid,
config.messageClient.authToken);

module.exports = {
    validateNIC: function (nic) {
        return fetch(config.govAPI + nic)
            .then(handleErrors)
            .then(res => res.json())
            .then(data => {
                return data.validated
            })
            .catch(err => {
                console.log(err)
            })
    },

    sendEmail: async function (body) {

        const emailConfig = config.emailClient

        const transporter = nodemailer.createTransport({
            host: emailConfig.host,
            port: 465,
            secure: true,
            auth: emailConfig.auth
        });

        var mailOptions = {
            from: '"Sri Lanka Railways"' + emailConfig.email,
            to: body.email,
            subject: body.subject,
            html: body.html
        };

        transporter.sendMail(mailOptions, function (error, info) {
            if (error) {
                console.log(error)
            } else {
                console.log('Email sent: ' + info.response);
            }
        });
    },

    sendTextMessage: async function (body) {
        var to = body.phone
        if (to.startsWith("0")) {
            to = to.replace("0", "+94")
        }
        twilio.messages
            .create({
                body: "Sri Lanka Railway - Reservation Slip \n\n Reference No : " +
body.reservationID + " \n\n From " + body.from + " to " + body.to + " \n Date : " +
body.date + " \n Time : " + body.time + " \n Train : " + body.train + " \n Class: " +
body.trainClass + " \n Quantity : " + body.qty + " \n Total : " + body.total + "
INR",
                from: config.messageClient.phoneNo,
                to: to
            })
            .then(message => console.log(message.sid))
            .catch(err => console.log(err))
    }
}
handleErrors = response => {
    if (!response.ok) {
        throw new Error("Request failed " + response.statusText)
    }
    return response
}
```

- **Railway services (/services/routers/railway.js)**

```
const express = require('express')
const router = express.Router()
const routeModel = require('../model/route')
const trainModel = require('../model/train')
const classModel = require('../model/classes')
const scheduleModel = require('../model/schedule')
const reservationModel = require('../model/reservation')
const client = require('../client')

router.get('/railway/routes', async (req, res) => {
    try {
        const result = await routeModel.find()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/route/:id', async (req, res) => {
    try {
        const result = await routeModel.findOne({ '_id': req.params.id })
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/trains', async (req, res) => {
    try {
        const result = await trainModel.find()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/trains/:route', async (req, res) => {
    try {
        const route = await routeModel.findOne({ '_id': req.params.route })
        const result = await trainModel.find({ route: route.name })
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/classes', async (req, res) => {
    try {
        const result = await classModel.find()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/schedules', async (req, res) => {
    try {
        const result = await scheduleModel.find()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.post('/railway/reservations', async (req, res) => {
    try {
        const body = req.body
        var reservation = new reservationModel(body)
        var result = await reservation.save()
        if (body.phone) {
            client.sendTextMessage({ ...body, reservationID: result._id })
        } else if (body.card) {
            const html = '<h2><u>Reservation Slip</u></h2><p>Reference No : <b> ' +
result._id + ' </b><br><br>From <b> ' + body.from + ' </b> to <b> ' + body.to + '
</b><br>' + 'Date :<b> ' + body.date + ' </b> Time :<b> ' + body.time + '
</b><br>Train : <b>' + body.train + ' </b> Class: <b> ' + body.trainClass + '
</b><br>Quantity : <b> ' + body.qty + ' </b></p><p>Total : <b> ' + body.total + '
INR</b></p> '
            client.sendEmail({ ...body, html: html, subject: 'Railway e-Ticket' })
        }
        res.status(200).json(result)
    }
    catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/reservations', async (req, res) => {
    try {
        const result = await reservationModel.find()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.get('/railway/reservations/:user', async (req, res) => {
    try {
        const result = await reservationModel.find({ user: req.params.user })
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

router.delete('/railway/reservations/:id', async (req, res) => {
    try {
        const result = await reservationModel.deleteOne({ _id: req.params.id
}).exec()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **Register service (/services/routers/register.js)**

```
const express = require('express')
const router = express.Router()
const UserModel = require('../model/user')
const client = require('../client')

router.post('/register', async (req, res) => {
    const body = req.body
    const email = body.email

    var exist = ""
    try {
        await UserModel.findOne({ email: email }, (err, val) => {
            if (err) {
                console.log(err);
            } else {
                exist = val
            }
        });

        if (exist) {
            res.status(409).json({ exist: true })
        } else {
            const discount = await client.validateNIC(body.nic)
            var user = new UserModel({ ...body, discount: discount })
            var result = await user.save()
            res.status(200).json(result)
        }
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **User service (/services/routers/user.js)**

```
const express = require('express')
const router = express.Router()
const UserModel = require('../model/user')
const client = require('../client')

router.put('/users/:id', async (req, res) => {
    const body = req.body
    try {
        var user = await UserModel.findById(req.params.id).exec()
        const discount = body.nic ? await client.validateNIC(body.nic) : false
        user.set({ ...body, discount: discount })
        var result = await user.save()
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **Login service (/services/routers/login.js)**

```
const express = require('express')
const router = express.Router()
const UserModel = require('../model/user')

router.post('/login', (req, res) => {
    const body = req.body
    const username = body.username
    const password = body.password

    try {
        UserModel.findOne({ email: username, password: password }, (err, val) => {
            if (err) {
                console.log(err);
            } else {
                if (val) {
                    res.status(200).json(val)
                } else {
                    res.status(401).json("unauthorized")
                }
            }
        });
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **Payment service (/services/routers/payment.js)**

```
const express = require('express')
const router = express.Router()
const CardModel = require('../model/card')
const PhoneModel = require('../model/phone')

router.post('/payment/card', (req, res) => {

    const body = req.body

    try {
        CardModel.findOne({ card: body.card, cvc: body.cvc, exp: body.exp }, (err,
val) => {
            if (err) {
                console.log(err);
                res.status(500).json(err)
            } else if (!val) {
                res.status(200).json({ validated: false })
            } else {
                console.log(req.body.total + " paid")
                res.status(200).json({ validated: true })
            }
        });
    } catch (err) {
        res.status(500).json(err)
    }
});

router.post('/payment/phone', (req, res) => {

    const body = req.body

    try {
        PhoneModel.findOne({ phone: body.phone, pin: body.pin }, (err, val) => {
            if (err) {
                console.log(err);
                res.status(500).json(err)
            } else if (!val) {
                res.status(200).json({ validated: false })
            } else {
                console.log(req.body.total + " paid")
                res.status(200).json({ validated: true })
            }
        });
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **Contact service (/services/routers/contact.js)**

```
const express = require('express')
const router = express.Router()
const contactModel = require('../model/contact')
const client = require('../client')

router.post('/railway/contact', async (req, res) => {
    try {
        const body = req.body
        var contact = new contactModel(body)
        var result = await contact.save()
        const phone = body.phone ? 'Phone :<b> ' + body.phone + ' </b><br> ' : ''
        const html = '<h2><u>User Contact</u></h2><p>Reference No : <b> ' +
result._id + ' </b><br><br>Name : <b> ' + body.fname + ' ' + body.lname + ' </b><br>
' + phone + '  Email :<b> ' + body.email + ' </b><br>Message : <b>' + body.message +
' </b></p> '
        client.sendEmail({ ...body, html: html, subject: 'User Contact', email:
body.email + ', sl.railway.e.ticketing@gmail.com' })
        res.status(200).json(result)
    } catch (err) {
        res.status(500).json(err)
    }
});

module.exports = router
```

- **Government service (/services/routers/gov.js)**

```
const express = require('express')
const router = express.Router()
const employeeModel = require('../model/employee')

router.get('/gov/employee/:nic', (req, res) => {
    try {
        employeeModel.findOne({ nic: req.params.nic }, (err, val) => {
            if (err) {
                console.log(err);
            } else {
                if (val) {
                    res.status(200).json({ validated: true })
                } else {
                    res.status(200).json({ validated: false })
                }
            }
        });
    } catch (err) {
        res.status(500).json(err)
    }
});
module.exports = router
```

- **User DB Schema (/services/model/user.js)**

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

const userSchema = mongoose.Schema({
    fname: {
        type: String,
        required: true,
    },
    lname: {
        type: String,
        required: true,
    },
    phone: {
        type: String,
        required: true,
    },
    nic: {
        type: String
    },
    address: {
        type: String,
        required: true,
    },
    email: {
        type: String,
        required: true,
    },
    password: {
        type: String,
        required: true,
    },
    discount: {
        type: Boolean,
        required: true
    }
})

const user = module.exports = mongoose.model('User', userSchema)
```

- **Train DB Schema (/services/model/train.js)**

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

const trainSchema = mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
    route: {
        type: String,
        required: true,
    }
})

const train = module.exports = mongoose.model('Train', trainSchema)
```

- **Schedule DB Schema (/services/model/schedule.js)**

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

const scheduleSchema = mongoose.Schema({
    time: {
        type: String,
        required: true,
    }
})

const schedule = module.exports = mongoose.model('Schedule', scheduleSchema)
```

- **Route DB Schema (/services/model/route.js)**

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

const routeSchema = mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
    route: [
        {
            name: {
                type: String,
                required: true,
            },
            fair: {
                type: Number,
                required: true,
                }
            }
        ]
})

const route = module.exports = mongoose.model('Route', routeSchema)
```

- **Reservation DB Schema (/services/model/reservation.js)**

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

const reservationSchema = mongoose.Schema({
    user: {
        type: String,
        required: true,
    },
    from: {
        type: String,
        required: true,
    },
    to: {
        type: String,
        required: true,
    },
    train: {
        type: String,
        required: true,
    },
    trainClass: {
        type: String,
        required: true,
    },
    time: {
        type: String,
        required: true,
    },
    qty: {
        type: Number,
        required: true,
    },
    date: {
        type: String,
        required: true,
    },
    amount: {
        type: Number,
        required: true,
    },
    discount: {
        type: Number,
        required: true,
    },
    total: {
        type: Number,
        required: true,
    },
    card:{
        type: String
    },
    phone:{
        type: String
    },
    email:{
        type: String
    }
})

const reservation = module.exports = mongoose.model('Reservation', reservationSchema)
```

- **Phone DB Schema (/services/model/phone.js)**

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

const phoneSchema = mongoose.Schema({
    phone: {
        type: String,
        required: true,
    },
    pin: {
        type: String,
        required: true,
    }
})

const phone = module.exports = mongoose.model('Phone', phoneSchema)
```

- **Employee DB Schema (/services/model/employee.js)**

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

const employeeSchema = mongoose.Schema({
    firstName: {
        type: 'String'
    },
    lastName: {
        type: 'String'
    },
    nic: {
        type: 'String'
    },
    address: {
        type: [
            'Mixed'
        ]
    }
})

const employee = module.exports = mongoose.model('Employee', employeeSchema)
```

- **Contact DB Schema (/services/model/contact.js)**

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

const contactSchema = mongoose.Schema({
    fname: {
        type: String,
        required: true,
    },
    lname: {
        type: String,
        required: true,
    },
    phone: {
        type: String,
    },
    email: {
        type: String,
        required: true,
    },
    message: {
        type: String,
        required: true,
    }
})

const contact = module.exports = mongoose.model('Contact', contactSchema)
```

- **Classes DB Schema (/services/model/classes.js)**

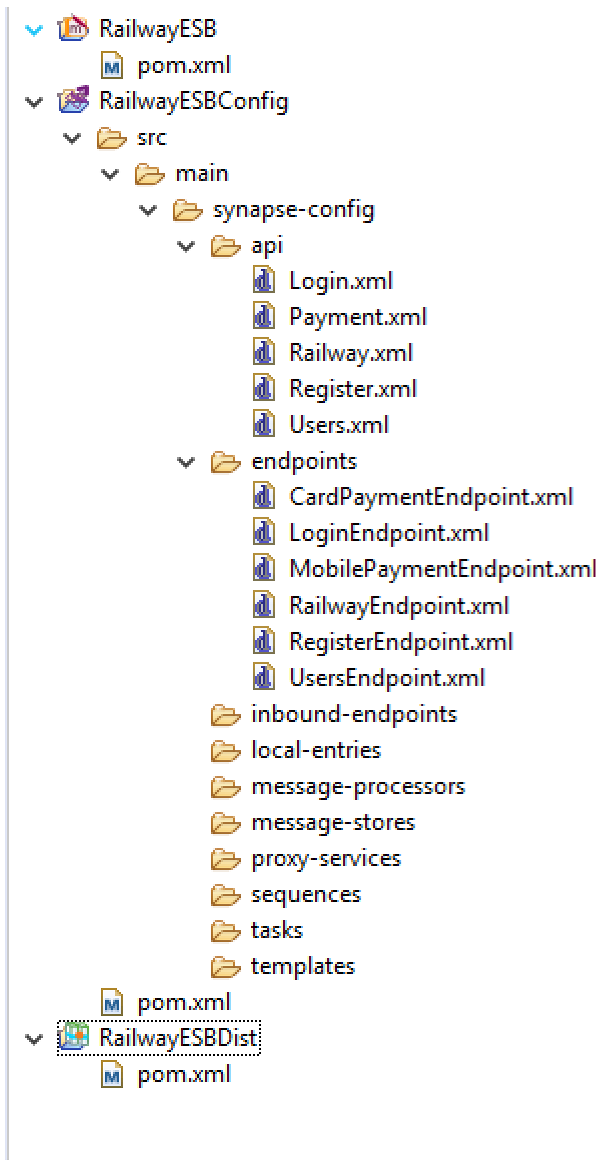
```
const mongoose = require('mongoose')

const classSchema = mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
    fairRatio: {
        type: Number,
        required: true,
    }
})

const trainClass = module.exports = mongoose.model('Class', classSchema)
```

- **Card DB Schema (/services/model/card.js)**

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

const cardSchema = mongoose.Schema({
    card: {
        type: String,
        required: true,
    },
    cvc: {
        type: String,
        required: true,
    },
    exp: {
        type: String,
        required: true,
    }
})

const card = module.exports = mongoose.model('Card', cardSchema)
```

6.3. WSO2 EI (Enterprise Integration – ESB)

Following figure shows the project structure of the Enterprise Integration.



```
RailwayESB
    pom.xml
RailwayESBConfig
    src
        main
            synapse-config
                api
                    Login.xml
                    Payment.xml
                    Railway.xml
                    Register.xml
                    Users.xml
                endpoints
                    CardPaymentEndpoint.xml
                    LoginEndpoint.xml
                    MobilePaymentEndpoint.xml
                    RailwayEndpoint.xml
                    RegisterEndpoint.xml
                    UsersEndpoint.xml
                inbound-endpoints
                local-entries
                message-processors
                message-stores
                proxy-services
                sequences
                tasks
                templates
    pom.xml
RailwayESBDist
    pom.xml
```

Fig 30: WSO2 -EI project structure

- **Payment API (/wso2-ei/RailwayESBConfig/src/main/synapse-config/api/Payment.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<api context="/payment" name="Payment" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="OPTIONS POST" protocol="http" uri-template="/{method}">
        <inSequence>
            <property action="remove" name="REST_URL_POSTFIX" scope="axis2"/>
            <switch source="get-property('uri.var.method')">
                <case regex="phone">
                    <send>
                        <endpoint key="MobilePaymentEndpoint"/>
                    </send>
                </case>
                <case regex="card">
                    <send>
                        <endpoint key="CardPaymentEndpoint"/>
                    </send>
                </case>
                <default/>
            </switch>
        </inSequence>
        <outSequence>
            <respond/>
        </outSequence>
        <faultSequence/>
    </resource>
</api>
```

- **CardPayment Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/CardPaymentEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="CardPaymentEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/payment/card"/>
</endpoint>
```

- **MobilePayment Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/MobilePaymentEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="MobilePaymentEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/payment/phone"/>
</endpoint>
```

- **Login API (/wso2-ei/RailwayESBConfig/src/main/synapse-config/api/Login.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<api context="/login" name="Login" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="OPTIONS POST" protocol="http">
        <inSequence>
            <send>
                <endpoint key="LoginEndpoint"/>
            </send>
        </inSequence>
        <outSequence>
            <respond/>
        </outSequence>
        <faultSequence/>
    </resource>
</api>
```

- **Login Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/LoginEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="LoginEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/login"/>
</endpoint>
```

- **Railway API (/wso2-ei/RailwayESBConfig/src/main/synapse-config/api/Railway.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<api context="/railway" name="Railway" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="DELETE OPTIONS POST PUT GET" protocol="http">
        <inSequence>
            <send>
                <endpoint key="RailwayEndpoint"/>
            </send>
        </inSequence>
        <outSequence>
            <respond/>
        </outSequence>
        <faultSequence/>
    </resource>
</api>
```

- **Railway Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/RailwayEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="RailwayEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/railway"/>
</endpoint>
```

- **Register API (/wso2-ei/RailwayESBConfig/src/main/synapse-config/api/Register.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<api context="/register" name="Register" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="OPTIONS POST" protocol="http">
        <inSequence>
            <send>
                <endpoint key="RegisterEndpoint"/>
            </send>
        </inSequence>
        <outSequence>
            <respond/>
        </outSequence>
        <faultSequence/>
    </resource>
</api>
```

- **Register Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/RegisterEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="RegisterEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/register"/>
</endpoint>
```

- **Users API (/wso2-ei/RailwayESBConfig/src/main/synapse-config/api/Users.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<api context="/users" name="Users" xmlns="http://ws.apache.org/ns/synapse">
    <resource methods="DELETE OPTIONS POST PUT GET" protocol="http" uri-
template="/{id}">
        <inSequence>
            <log>
                <property expression="fn:concat('User ID - ',get-
property('uri.var.id'))" name="text"/>
            </log>
            <send>
                <endpoint key="UsersEndpoint"/>
            </send>
        </inSequence>
        <outSequence>
            <respond/>
        </outSequence>
        <faultSequence/>
    </resource>
</api>
```

- **Users Endpoint (/wso2-ei/RailwayESBConfig/src/main/synapse-config/endpoints/UsersEndpoint.xml)**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoint name="UsersEndpoint" xmlns="http://ws.apache.org/ns/synapse">
    <http uri-template="http://localhost:3001/users"/>
</endpoint>
```

7. Testing

7.1 Manual Testing

Test performed by: Kriti Chapagain, Samar Pratap Singh
Test performed for: Train Ticket Reservation System

1. Purpose

This document explains the various activities performed as part of Testing of 'Train Ticket Reservation system' application.

2. Application Overview

'Train Ticket Reservation system' is a web based train ticket booking application. Tickets for various buses can be booked using the online facilities. Real time passenger information is received from a 'Central repository system', which will be referred before booking is confirmed. There are several modules like Registration, Booking, Payment which are integrated to fulfill the purpose.

3. Testing Scope

This section explains about the functions/modules in scope & out of scope for testing; Any items which are not tested due to any constraints/dependencies/restrictions.

In Scope

Functional Testing for the following modules are in Scope of Testing

- Registration
- Login
- Booking
- Payment

Items not tested

Verification of connectivity with the third party system ‘Central repository system’ was not tested, as the connectivity could not be established due to some technical limitations. This can be verified during UAT (User Acceptance Testing) where the connectivity is available or can be established.

INDIVIDUAL MODULE TESTING

Module Name	**Description**	**Valid Test cases & Invalid Test cases**	**Expected Result**	**Actual Result**	**Status Pass/fail (If ER matched with AR)**
Registration	In this module , user needs fill in his details in order to get registered to the portal.	**Invalid Test case:** If the user is already registered, they can't register again with same credentials**.**	The user is not able to register with same E-mail address.	The user is not able to register with same E-mail address.	**PASS**
		Valid Test case: If the new user registers for the portal, the will successfully create a new account.	The user gets logged in to the portal.	The user gets logged in to the portal.	**PASS**
Login	In this module, user gets logged in to the portal with the registered credentials.	**Invalid test case:** If any of the credentials is wrong, they won't be able to log in to the portal.	The user is not able to login to the portal.	The user is not able to login to the portal.	**PASS**
		Valid test case: If the credentials are correct, the user will be able to log in to the portal**.**	The user is able to login the portal.	The user is able to login to the portal.	**PASS**
		Test case 01:		User is able to select	**PASS**

			User is able to select the destination.	the destination.	
Booking	In this module, User gets to: 1. Select destination 2. Train 3. Date 4. Time 5. No. of tickets	**Test case 02:**	User can select train of their preference.	User can select train of their preference.	**PASS**
		Test case 03:	Date according to their preference.	Date according to their preference.	**PASS**
		Test case 04:	Flexible time	Flexible time	**PASS**
		Test case 05:	Can select required quantity of tickets.	Can select required quantity of tickets.	**PASS**
My Reservation	In this module, user can view their bookings and cancel at any time.	________________	user can view their bookings and cancel at any time.	user can view their bookings and cancel at any time.	**PASS**
Payment	Two options will be provided to the user for payment.	This module should allow user to make payment via credit card or Phone number.	Users are able to complete the payment successfully	Users are able to complete the payment successfully	**PASS**

			through any of the mode.	through any of the mode.	

9. Conclusion:

In sum up,The overall modules of our project work as per expected and the user Inteface is quite simple and easy to use which was our main motto.The goal which we have set for the project was sucessfully achieved.

10. Future Works:

1. Mail after the booking of the ticket.
2. Message in the phone number after paying via Mobile.
3. More databases (locations).