

DS Assignment 2 – Online Train Reservation System

Module: Distributed Systems

Assignment: 2 - Rest API, WSO2 EI

Software Engineering Weekend Batch

Name

P.P.G.S.H.A.Guruge

Registration No

IT17042352

Introduction

This is a Web Application where the user can reserve train tickets based on the trains which are available. The Online Train Reservation is made using HTML, JavaScript, Java, jQuery and AJAX technologies. The services are integrated using WSO2 EI.

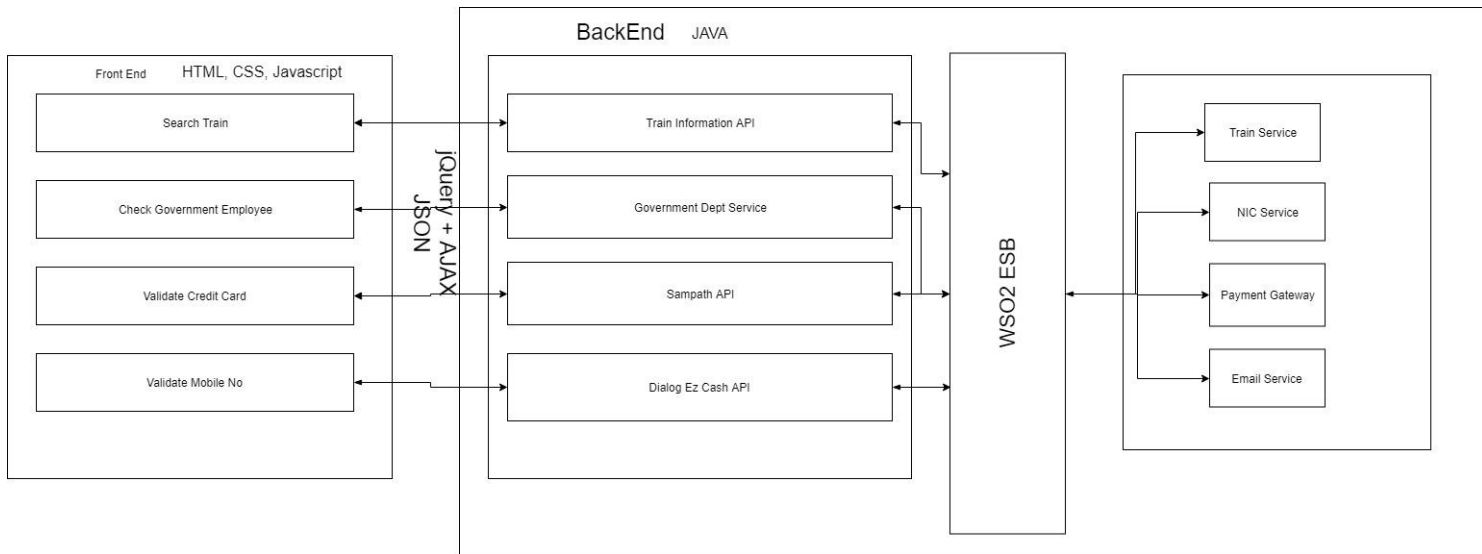
First the user should select the required train, and then the system shows the details of the selected train, then the user can enter the number of tickets that the user wishes to purchase, then after calculating the Total Bill, the user can select the Payment method.

If the user wishes to pay using Credit Card the user should enter Credit Card Details or if the user wishes to pay using Mobile Credit the should enter his mobile details.

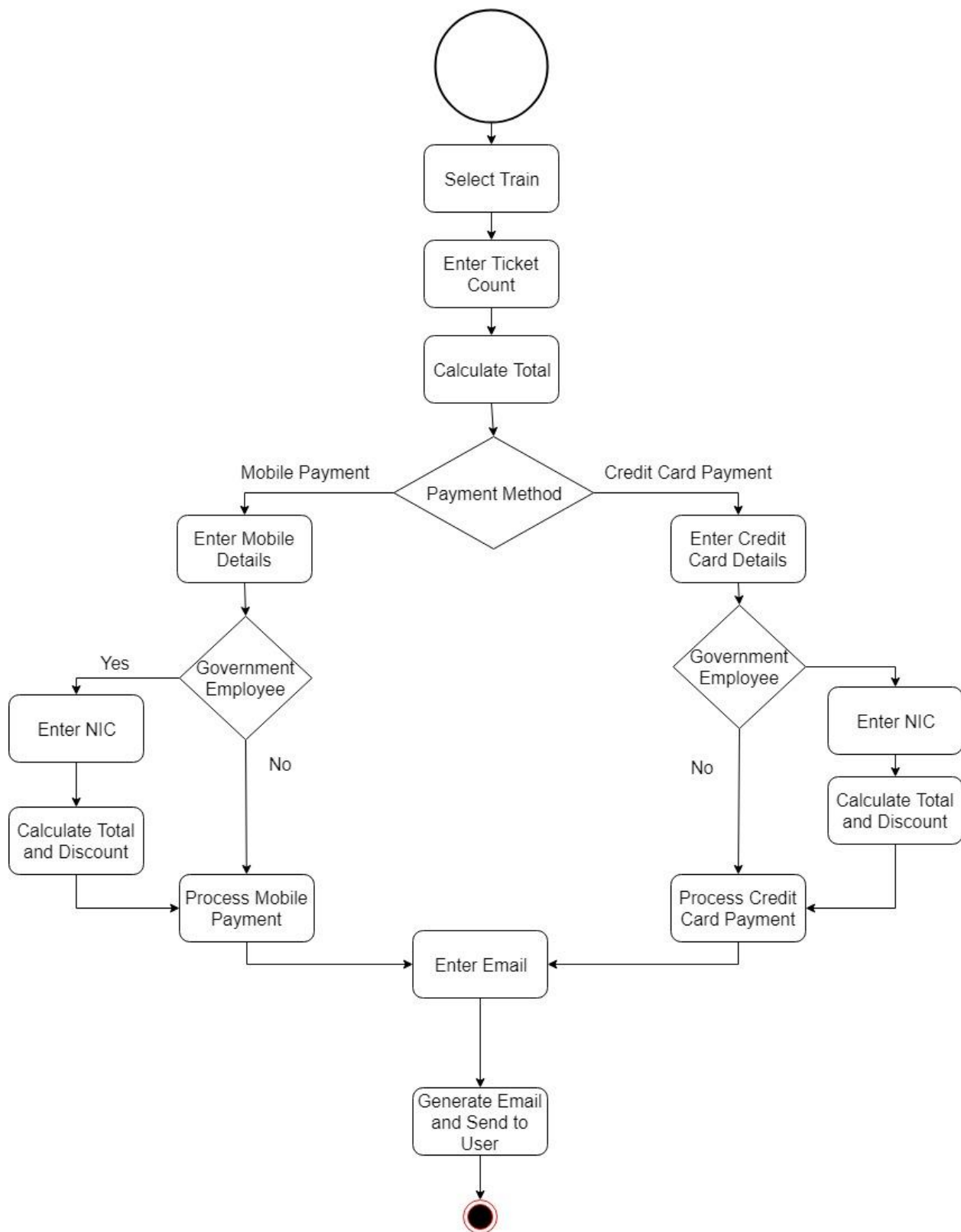
After successfully enter the payment details, if the user is a Government employee the user can enter his NIC number and get a discount.

If the payment is successful, the user is prompted to enter his email and an email is sent to the user confirming the booking is confirmed.

High Level Architectural Diagram



Workflow



1. User Selects Train from the drop-down List
2. User Enters the Ticket Count and clicks Calculate Total
3. User Select Mobile as Payment Method in the dropdown
4. User clicks Book
5. Tickets

Mount Lavinia/Kankesanthurai - AC Intercity ▼

Show Train Details

	Train	Train Name	Train Number	Arrival Time	Departure Time	Ticket Price per Head
1	Mount Lavinia/Kankesanthurai AC Intercity	4021	4:11	24:12	1000	

Ticket Count :

1

Calculate Total

Total Price :

1000

Select Payment Method :

Mobile ▼

Book Tickets

6. User Enters Mobile No and 4-digit pin no

7. User is a government Employee so, he selects Yes from the dropdown

8. User enters his NIC and clicks Validate NIC button and system displays the status of the NIC (Verified or Not)

Mobile No :

Pin No :

Are you a Government Employee :

NIC :

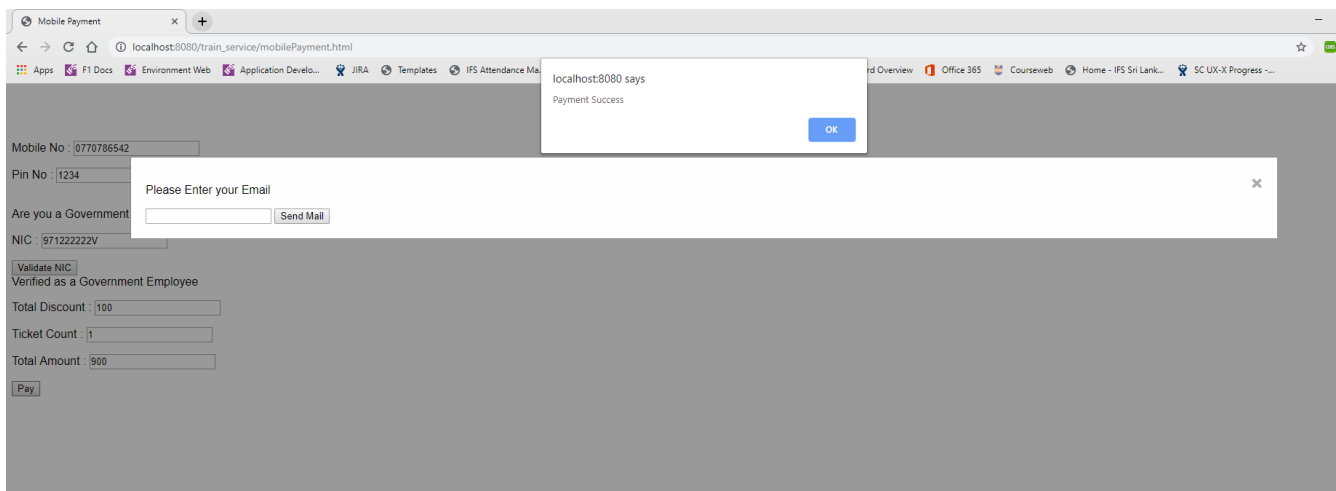
Verified as a Government Employee


Total Discount :

Ticket Count :

Total Amount :

9. System displays payment status in alert box.
10. System prompts the user to enter his email address
11. System display email status in alert box.
12. System sends the email to the user's email address



 **reservationtrain880@gmail.com**
to me ▾

You have sucessfully Booked 1 tickets in

Train : Mount Lavinia/Kankesanthurai
Train Name : AC Intercity
Purchase made on : 2019/05/04 19:33:55
Total : 900
Booking Reference : dee62120-150f-4c7f-9a4c-436fef3b1f72


Please provide your booking reference at the Railway Station

[Booking confirmed.](#)

[Thanks a lot.](#)

[I haven't received it yet.](#)

 Reply

 Forward

Credit Card Validation and Payment Gateway

1. The Credit Card Number, CVC no and holder's name is matched against the provided Credit Card Details by the User.
2. After the validation, before processing the payment, credit card balance is checked ensuring that the credit card limit has not been exceeded.
3. Then the credit card balance is compared with ticket price ensuring that credit card has enough credit to complete the purchase.
4. Then the ticket amount is deducted from the credit card

The following code explains the above scenario

```
function payCreditCard() {
    jQuery
        .ajax({
            url : "http://localhost:8280/paymentGateway/processCard/"
                + $("#cardNo").val()
                + "/"
                + $("#cvcNo").val()
                + "/"
                + $("#holderName").val(),
            type : "GET",
            contentType : "application/json",
            dataType : 'json',
            success : function(data, textStatus, errorThrown) {
                var cardNo = data.creditCardNo;
                var limit = data.limit;
                var currentAmount = data.currentAmount;
                var availableBalance = limit - currentAmount;
                var ticketPrice = $("#total").val();
                if (availableBalance > 0) {
                    if (availableBalance >= ticketPrice) {
                        updateBalance(cardNo, ticketPrice);
                        var modal = document.getElementById('myModal');
                        modal.style.display = "block";
                    } else
                        alert("Credit Card Balance not Sufficient");
                    } else
                        alert("Card Limit has Exceeded, Contact your Bank");
            },
            error : function(jqXHR, textStatus, errorThrown) {
                alert("Payment Failed, Invalid Card Details");
            },
            timeout : 12000,
        });
};
```


Mobile Payment Validation and Mobile Payment Gateway

1. The Mobile No, and 4-digit PIN is matched against the provided Mobile Details by the User.
2. After the validation, before processing the payment, mobile balance is checked ensuring that the mobile balance is not zero.
3. Then the credit card balance is compared with ticket price ensuring that mobile has enough credit to complete the purchase.
4. Then the ticket amount is deducted from the mobile

The following code explains the above scenario

```
function payMobile() {
    jQuery
        .ajax({
            url : "http://localhost:8280/dialogPayment/processMobilePayment/"
                + $("#mobileNo").val() + "/" + $("#pin").val(),

            type : "GET",
            contentType : "application/json",
            dataType : 'json',
            success : function(data, textStatus, errorThrown) {
                var mobileNo = data.mobileNo;
                var mobileBalance = data.balance;
                var ticketPrice = $("#total").val();
                if (mobileBalance > 0) {
                    if (mobileBalance >= ticketPrice) {
                        updateMobileBalance(mobileNo, ticketPrice);
                        var modal = document.getElementById('myModal');
                        modal.style.display = "block";
                    } else
                        alert("Mobile Balance not Sufficient");
                    } else
                        alert("Insufficient Mobile Balance !");
                },
            error : function(jqXHR, textStatus, errorThrown) {
                alert("Payment Failed, Invalid Mobile Details");
            },
            timeout : 12000,
        });
};
```

Appendix

Front End

AJAX Request – Process Mobile Payments

```
function payMobile() {
    jQuery
        .ajax({
            url: "http://localhost:8280/dialogPayment/processMobilePayment/"
            + $("#mobileNo").val() + "/" + $("#pin").val(),
            type : "GET",
            contentType : "application/json",
            dataType : 'json',
            success : function(data, textStatus, errorThrown) {
                var mobileNo = data.mobileNo;
                var mobileBalance = data.balance;
                var ticketPrice = $("#total").val();
                if (mobileBalance > 0) {
                    if (mobileBalance >= ticketPrice) {
                        updateMobileBalance(mobileNo, ticketPrice);
                        var modal = document.getElementById('myModal');
                        modal.style.display = "block";
                    } else
                        alert("Mobile Balance not Sufficient");
                    } else
                        alert("Insufficient Mobile Balance !");
                },
                error : function(jqXHR, textStatus, errorThrown) {
                    alert("Payment Failed, Invalid Mobile Details");
                },
                timeout : 12000,
            });
};
```

Routing

Mobile Payments Routing

```
public class DialogService {
    List<Phone> dialogPhones;

    public DialogService() {
        dialogPhones = DialogPhones.getPhones();
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Phone> getPhones() {
        return dialogPhones;
    }

    @Path("/{mobileNo}/{pin}")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Phone getMobbile(@PathParam("mobileNo") String mobileNo, @PathParam("pin") int
pin) {
        for (Phone b : dialogPhones) {
            if (b.getMobileNo().equals(mobileNo) && b.getPin() == pin)
                return b;
        }
        throw new NotFoundException();
    }

    @Path("/updateMobileAmount/{mobileNo}/{amount}")
    @POST
    @Produces(MediaType.APPLICATION_JSON)
    public Phone updateAmount(@PathParam("mobileNo") String mobileNo, @PathParam("amount")
double amount) {
        for (Phone b : dialogPhones) {
            if (b.getMobileNo().equals(mobileNo)) {
                b.setBalance(b.getBalance() - amount);
                return b;
            }
        }
        throw new NotFoundException();
    }
}
```

Web.xml – Configuration, mapping

```
<servlet>
  <servlet-name>dialog_service</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>dialogservice, com.jersey.jaxb, com.fasterxml.jackson.jaxrs.json
  </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>dialog_service</servlet-name>
  <url-pattern>/rest/mobilepaymentgateway/*</url-pattern>
</servlet-mapping>
```

Database

Government's Database of NICs

The third parameter in the constructor is the government Employee flag.

```
private static ArrayList<NIC> nicCards = new ArrayList<>();
static {
    nicCards.add(new NIC(1, "971222222V", true));
    nicCards.add(new NIC(2, "866652121V", true));
    nicCards.add(new NIC(3, "121212123V", false));
    nicCards.add(new NIC(4, "787232131V", true));
    nicCards.add(new NIC(5, "984324344V", false));
    nicCards.add(new NIC(6, "837233123V", true));
    nicCards.add(new NIC(7, "232323223V", true));
}
```

Dialog's Database of Dialog Mobile Numbers

```
private static ArrayList<Phone> phoneNumbers = new ArrayList<>();
static {
    phoneNumbers.add(new Phone("0770786542", "Tom", 5000, 1234));
    phoneNumbers.add(new Phone("0770721542", "Jerry", 51000, 1111));
    phoneNumbers.add(new Phone("0770123542", "Alan", 61000, 6543));
    phoneNumbers.add(new Phone("0770745542", "David", 9000, 6523));
    phoneNumbers.add(new Phone("0770786542", "Marie", 1000, 9876));
    phoneNumbers.add(new Phone("0770124542", "Daniel", 100, 1123));
    phoneNumbers.add(new Phone("0770098542", "Nicolson", 500, 8876));
}
```

Sampath's Database of Credit Cards

```
private static ArrayList<CreditCard> cards = new ArrayList<>();
static {
    cards.add(new CreditCard("4024007155211112", "123", "YaraLuna", 5000, 0,
"22/08"));
    cards.add(new CreditCard("4929570311913059", "423", "Ace Morin", 85000, 1000,
"23/09"));
    cards.add(new CreditCard("4024007195367999", "543", "Darlene Timms", 57000, 1000,
"23/09"));
    cards.add(new CreditCard("4916107452462243", "323", "Amiya Romero", 56000, 1000,
"23/09"));
    cards.add(new CreditCard("4916286362099228", "873", "Daryl Mccullough", 55000,
1000, "23/09"));
    cards.add(new CreditCard("5153705312725508", "433", "Lilia Mcfarland", 2000, 1000,
"23/09"));
    cards.add(new CreditCard("5578824892969357", "443", "Cruz Weaver", 23000, 1000,
"23/09"));
}
```

Train Service's Database of Trains

```
private static ArrayList<Train> trains = new ArrayList<>();
    static {
        trains.add(new Train(1, "Mount Lavinia/Kankesanthurai", "AC Intercity", "4021",
"4:11", "24:12", 1000));
        trains.add(new Train(2, "Colombo/Kandy", "Night Mail", "1009", "4:11", "24:12",
2000));
        trains.add(new Train(3, "Colombo/Batticaloa", "Udaya Devi", "6011", "4:11",
"24:12", 2500));
        trains.add(new Train(4, "Colombo/Batticaloa", "Night mail Express", "6079",
"4:11", "24:12", 1500));
        trains.add(new Train(5, "Colombo/Badulla", "Udarata Manike", "3011", "4:11",
"24:12", 1000));
        trains.add(new Train(6, "Colombo/Badulla", "Night mail Express", "5021", "4:11",
"24:12", 3500));
        trains.add(new Train(7, "Colombo/Kandy", "AC Intercity", "4009", "4:11", "24:12",
1500));
    }
```