Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

PROJECT REPORT
ON

ONLINE BUS-BOOKING SYSTEM

BY

INAMDAR AAKARSH (CE048)
JARIWALA SAHIL Y. (CE049)

B. Tech CE Semester- IV

Subject:
SOFTWARE ENGINEERING PRINCIPAL & PRACTICES

GUIDED BY
Prof. PINKAL CHAUHAN
Prof. JIGAR PANDYA
Dr. BRIJESH BHATT

INDEX:

1. ABSTRACT

Now-a-days, everyone is busy with their hectic schedules. Thus, online softwares are made to make their tasks easier.

Considering the case of ticket booking for bus, it is very time consuming for the travellers to stand in queue wasting their valuable time and energy for booking tickets and getting bus schedules. Also, people are bound in constraint like paying money in cash. Also, seats are not confirmed sometimes. There are even more constraints in offline booking of bus tickets. To overcome such constraints, online bus ticket booking software is used.

Online Bus-Booking Software is a software which make lives of people much easier by booking tickets efficiently in time, providing them cashless online payment modes (DIGITAL PAYMENT), providing easy ticket cancellation process, and also it takes feedback from users to improve services.

2. INTRODUCTION

This software is for bus ticket booking between the stations Surat to Ahmedabad and from Ahmedabad back to Surat with some more stations along the route. This system provides functionality such as ticket booking using online mode of payment (DIGITAL PAYMENT), providing discounts on bookings with some terms and conditions, cancellation of tickets in an easier way with some negligible penalties, showing status of available buses and seats available in these buses from different stations according the passenger's wish. Also, it provides a feedback form to share the experience of the passengers.

To use this software first time, the user must register himself/herself with providing some necessary details. Once registered, he/she can directly login from next time. After register/login, the user can select the bus type, route, seat according to time table of bus and availability of seats for the bus he/she wishes to book. Also, user is provided a facility to cancel the tickets but he/she will not receive 100% refund as some amount is deducted as penalties. Passengers can get discounts while booking tickets by some ways like applying promo code etc. He/she can get E-ticket of the booking he/she has done. At last, he/she can give feedback via feedback form.

2.1 TECHNOLOGY/TOOL USED

TECHNOLOGIES:

1. Django
2. Python
3. HTML
4. CSS
5. Bootstrap
6. SQLite3
7. Selenium Python

TOOLS:

- VS Code
- GITHUB
- PyCharm
- DBSQLite3

3. SOFTWARE REQUIREMENT SPECIFICATION:

1. MANAGE USER DETAILS

R.1.1: SIGN IN/UP

Description: For using system user must have an account. If user is new then he/she has to register himself/herself using this functionality by providing necessary information. If user is already registered, he/she can directly login into the system.

R.1.1.1 REGISTER NEW USERS

Input: Name, address, email (if user has), mobile no, new username and password.

Output: Confirmation of registration status and information stored in system and user directed to login page.

R.1.1.2 LOGIN

Input: Username and password

Output: If user is verified successfully, he/she will be prompted to Home page.

BELOW FUNCTIONALITIES CAN BE USED ONLY IF USER STATUS IS LOGGED IN.

R.1.2: UPDATE USER’S DETAILS

Description: Using this functionality a user can make change in his/her personal information as well as his password which he/she provided earlier.

R.1.2.1: DISPLAY CURRENT DETAILS.

Input: Selection
Output: Currently stored details.

R1.2.2: CHANGE OLD DETAILS.

Input: New details.
Output: Updated details with confirmation message of changes.

R.1.3: LOGOUT

Description: Using this functionality a user can come out of the system after the work is done.
Input: User Selection.
Output: Successfully logged out.

R.1.4: DELETE A USER ACCOUNT

Description: If user want to delete his/her account he/she can delete using this functionality or if user is inactive since long time then admin can delete his account.

R.1.4.1: USER WANT TO DELETE ACCOUNT

Input: Username and password.
Output: Confirmation message.

R.1.4.2: ADMIN WANTS TO DELETE ACCOUNT

Input: Username.
Output: Update the system record.

2. MANAGE BUS DETAILS

R.2.1: ADD NEW BUS

Description: Admin can add new bus using this functionality in system if organization have organized a new bus.
Input: Provide bus information.
Output: Update system records and generate bus-id.

R.2.2: REMOVE OLD BUS.

Description: Admin can remove bus from system if bus is no longer in working.
Input: Bus ID, bus number
Output: Update system records.

R.2.3: UPDATE BUS INFORMATION

Description: Admin can update bus timing, bus route, seat availability in bus by changing seat's color using this functionality.

R.2.3.1: CHANGE BUS TIME AND ROUTE.

Input: Bus ID, bus number
Output: change bus time and route.

R.2.3.2: CHANGE SEAT AVAILABILITY

Input: Bus ID, bus number
Output: change color of unavailable seat from green to red and for available seat change color from red to green.

R.2.4: SEARCH BUSES

Description: User can search for available buses of type he has selected. User have to provide source and destination and journey date.

Input: source name, destination, Date of journey.

Output: bus list

R.2.5: VIEW BUSES

Description: Any type of user can view all available buses with source, destination, time etc.

Input: select option

Output: bus list

3. MANAGE TICKET BOOKINGS

R.3.1: BOOK TICKETS

R.3.1.1: SELECT BUS TYPE

Description: This functionality helps the user to select the bus types from the available menu of buses like express, local, volvo, A/C bus.

Input: User’s selection

Output: Bus type is selected.

R.3.1.2: SELECT BUS

Description: After searching bus, this functionality helps the user to select the bus in which he/she wants to book tickets.

Input: User’s selection

Output: User prompted to select seats from available seats.

R.3.1.3: SELECT AVAILABLE SEATS.

Description: This functionality allows user to select the seats and number of seats from the available seats of the bus.
Input: seat number.
Output: Selects the seats with an appropriate confirmation message.

R.3.2: PAYMENT

Description: This functionality enables user to select the payment option and make payment for his booking. available Payment option are net-banking, credit card, debit card, wallet, UPI. If user have promo-code then he can apply that promo code and he can get discount.
Input: User's selection (promo code if promo code is available)
Output: Confirmation message with the transaction id and bill amount of the bookings are displayed if bookings are done.

R.3.3: GENERATE TICKET

Description: This functionality allows user to select the ways, he/she wants to get e-ticket. Available modes are download and print ticket, get an email ticket, get a SMS ticket.
Input: User Selection
Output: The ticket is generated accordingly and thank you page with "feedback" option is displayed.

R.3.4: CANCEL BOOKINGS

Description: This functionality enables user to cancel the tickets he/she has booked before or admin can cancel the whole bus booking. If user cancel the ticket then he will get refund with some negligible charges. If admin cancel the bus then all passenger will get full refund.

R.3.4.1: USER WANTS TO CANCEL TICKET

Input: transaction id
Output: Tickets are cancelled and refund is generated with some penalty.

R.3.4.2: ADMIN WANTS TO CANCEL BUS

Input: bus-id
Output: Remove bus from time table, message to all the bus passengers and give refund to all passengers.

R.3.5: DISPLAY BOOKINGS

Description: This functionality helps user to display the booking which user has done before.
Input: Transaction ID of booking.
Output: User’s previous bookings are displayed.

4. MANAGE STATISTIC:

R.4.1: GENERATE REPORT

Description: Manager can generate report for specific period of time. Report gives full information about users, booking, profit or loss etc.
Input: starting date and ending date
Output: generated report

R.4.2: FETCH HISTORY OF USER OR BUS

Description: Admin/User can see the history of given period of time.

R.4.2.1: FETCH HISTORY FOR USERS

Input: time period.
Output: User's bookings history for entered time period is displayed.

R.4.2.2: FETCH HISTORY FOR ADMIN

Input: Bus ID/User ID with time period
Output: Bus history or user history is displayed accordingly.

5. MANAGE FEEDBACK:

R.5.1: GIVE FEEDBACK

Description: Customer can give suggestion and rating or the system.
Input: feedback details
Output: greeting message

R.5.2: VIEW FEEDBACK

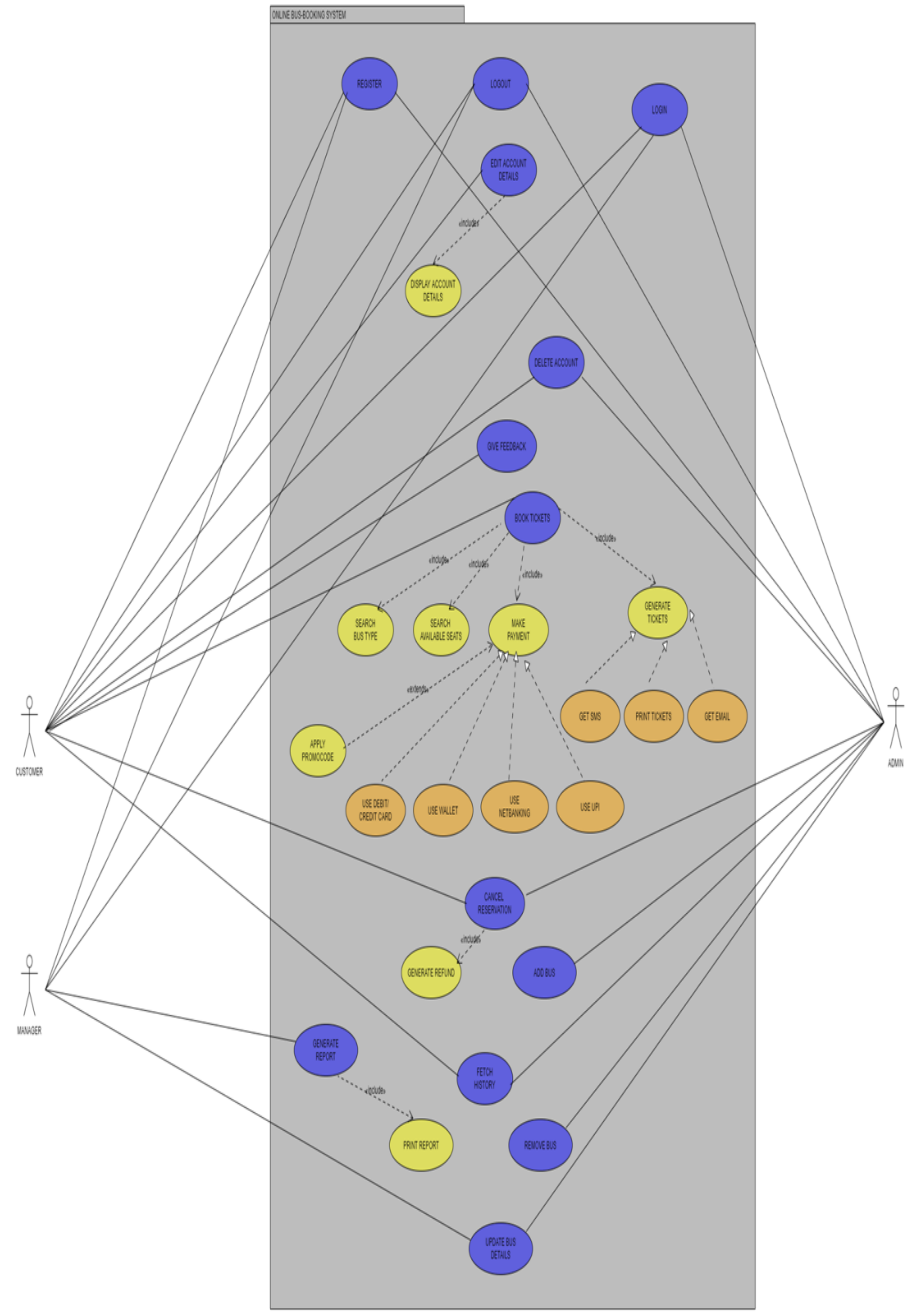
Description: Admin can see the feedback which is given by users
Input: Select view feedback
Output: Display feedback

R.5.3: RESPONSE TO FEEDBACK

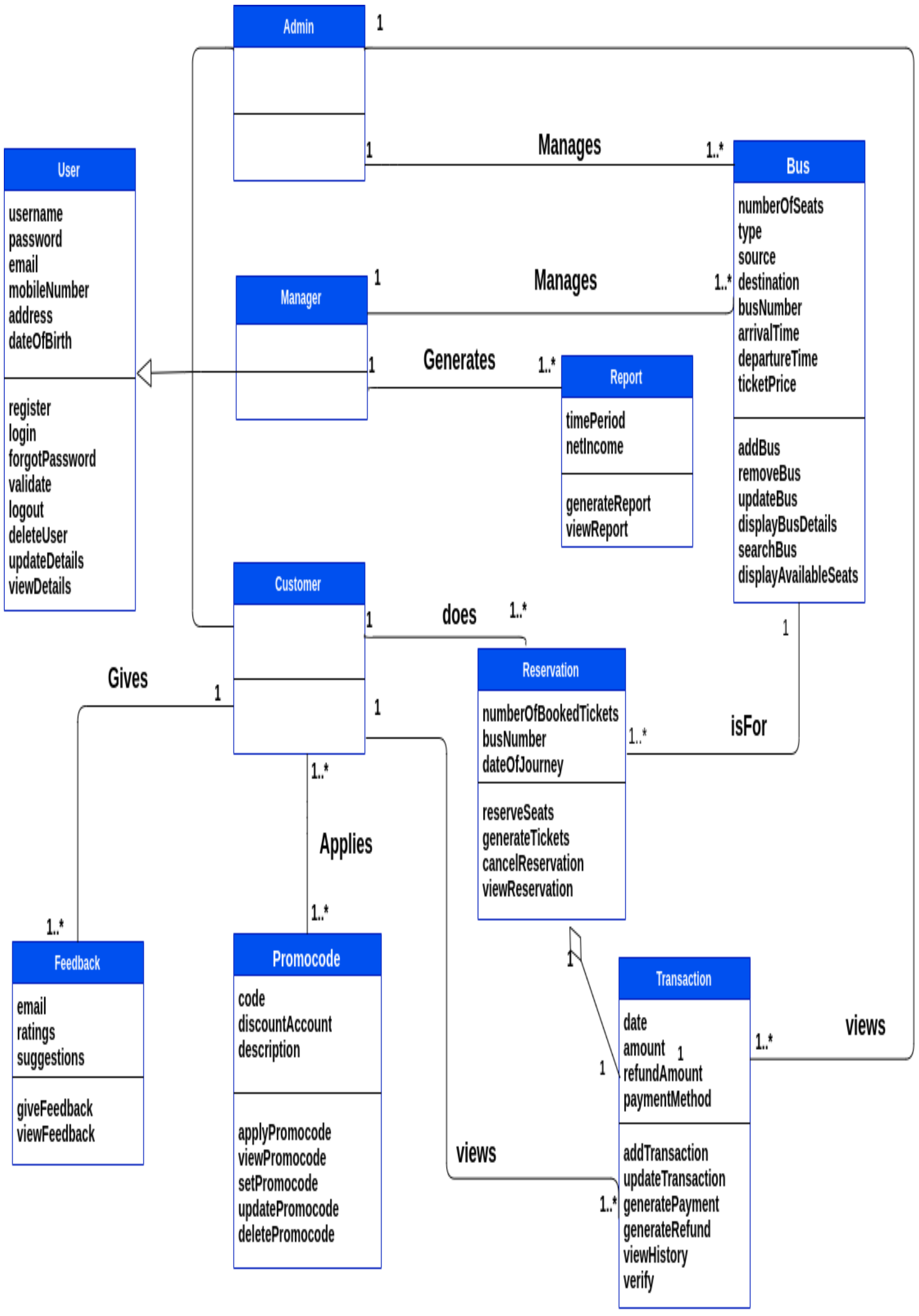
Description: Admin can give response as comment to given feedback
Input: select option
Output: response posted.

4. DESIGN

4.1 USE CASE

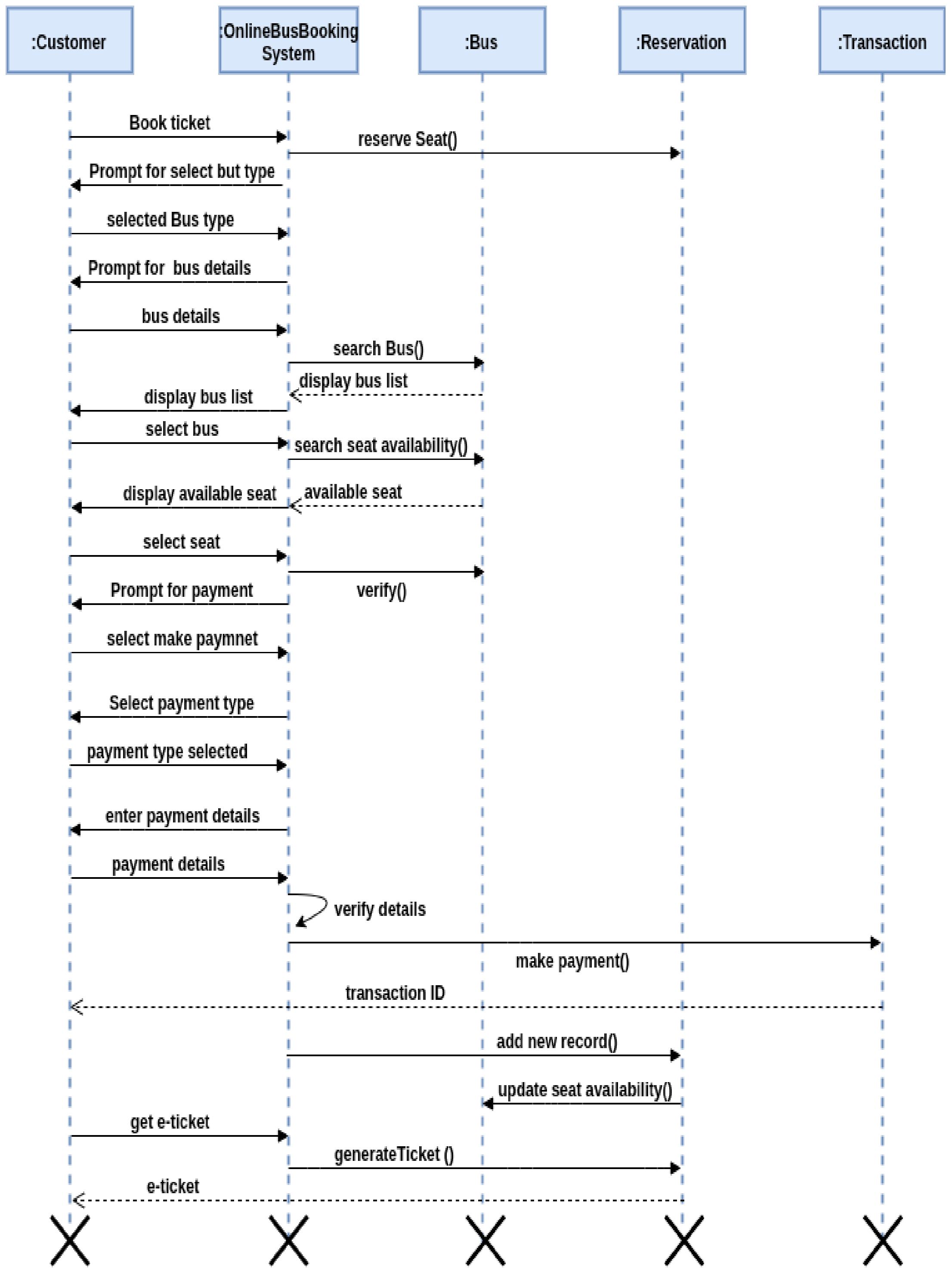


4.2 CLASS DIAGRAM

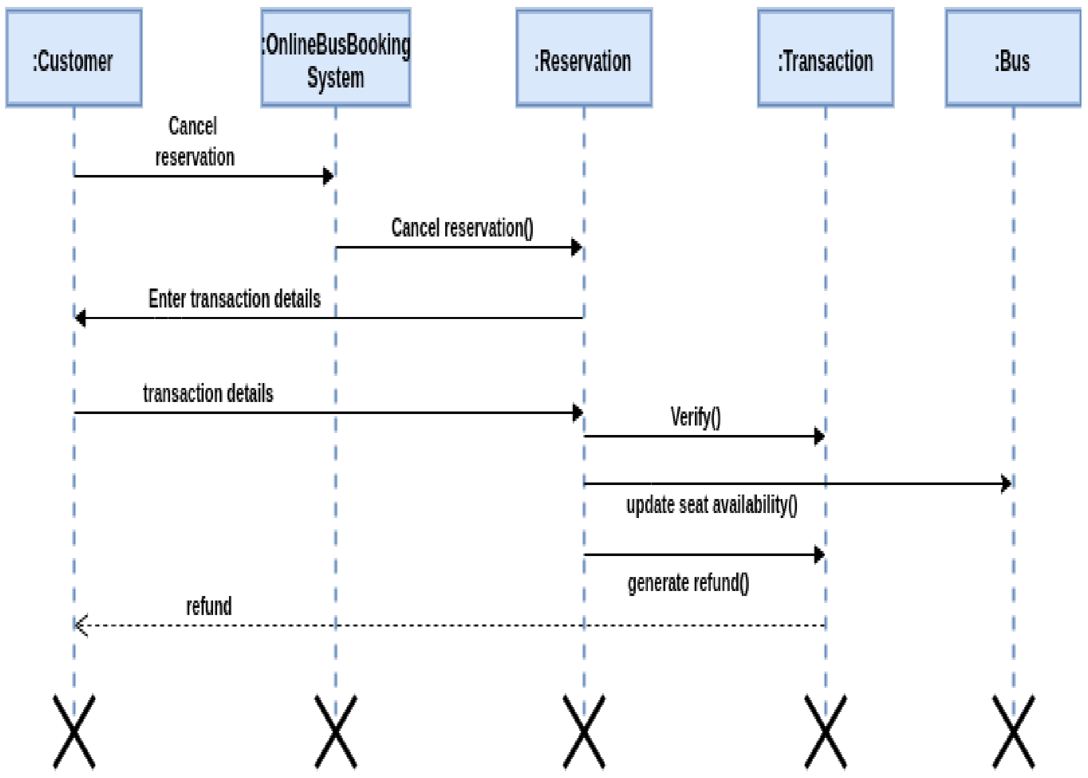


4.3 SEQUENCE DIAGRAM

4.3.1 BUS BOOKING

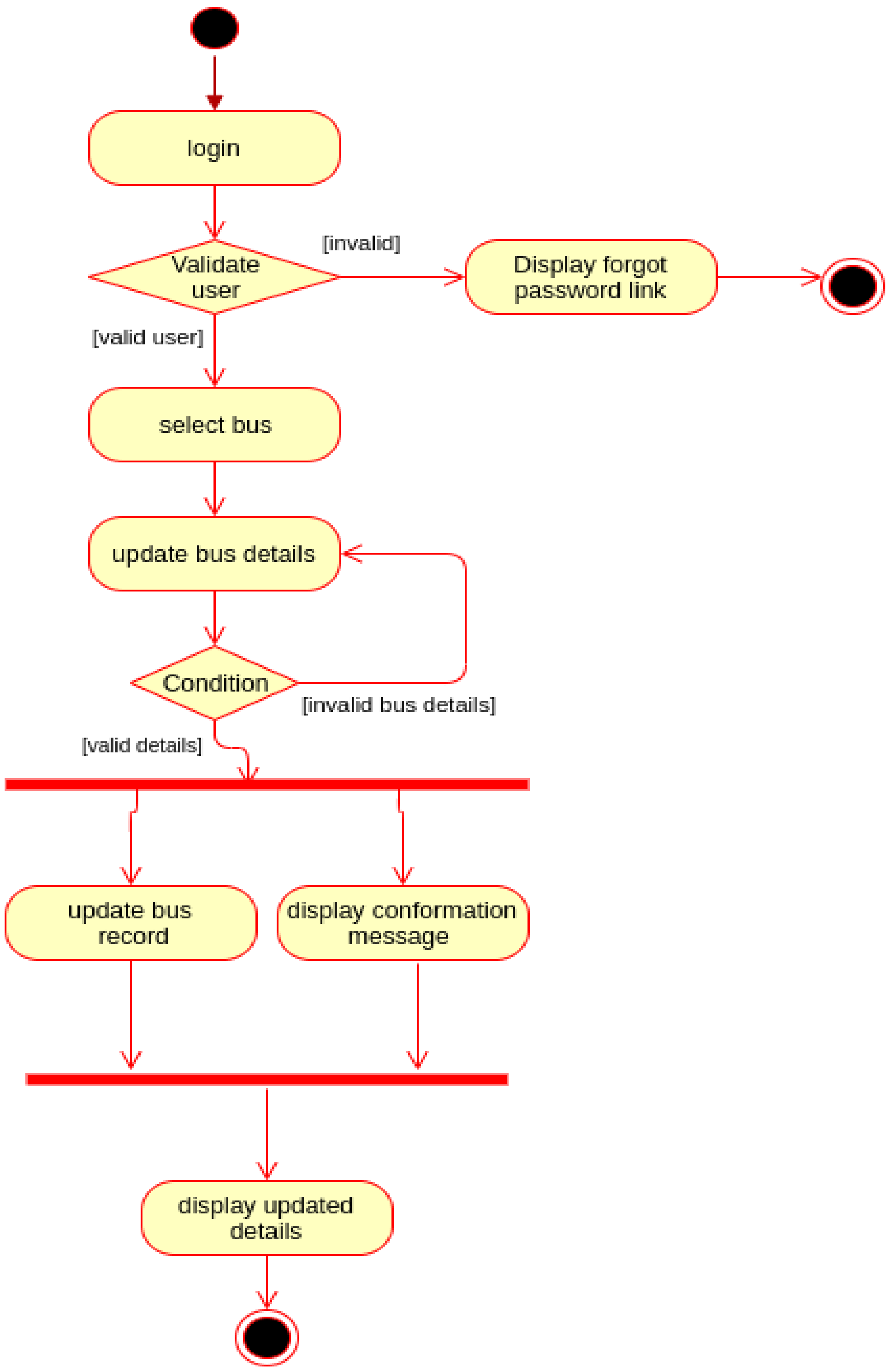


4.3.2 CANCEL BOOKINGS

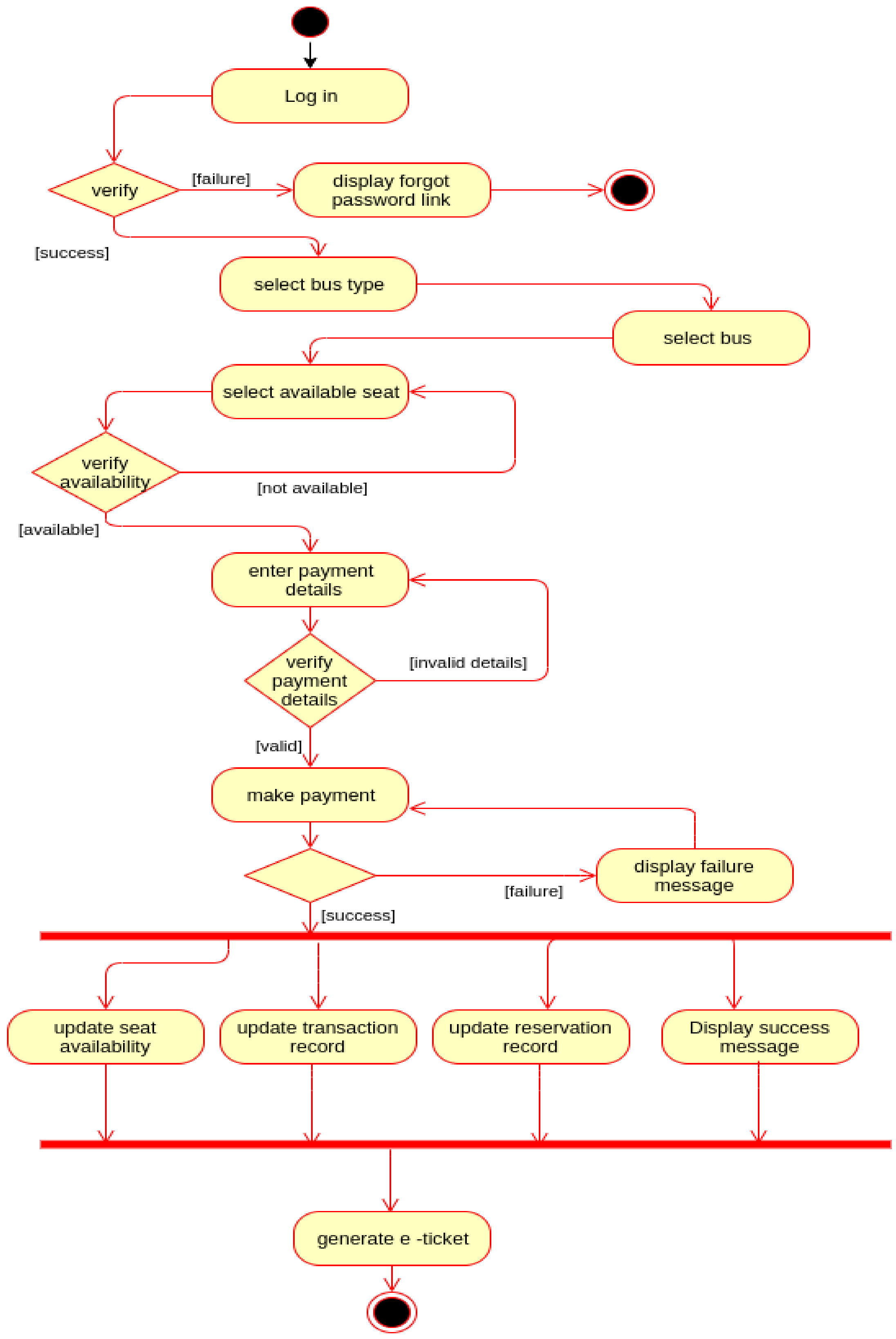


4.4 ACTIVITY DIAGRAM

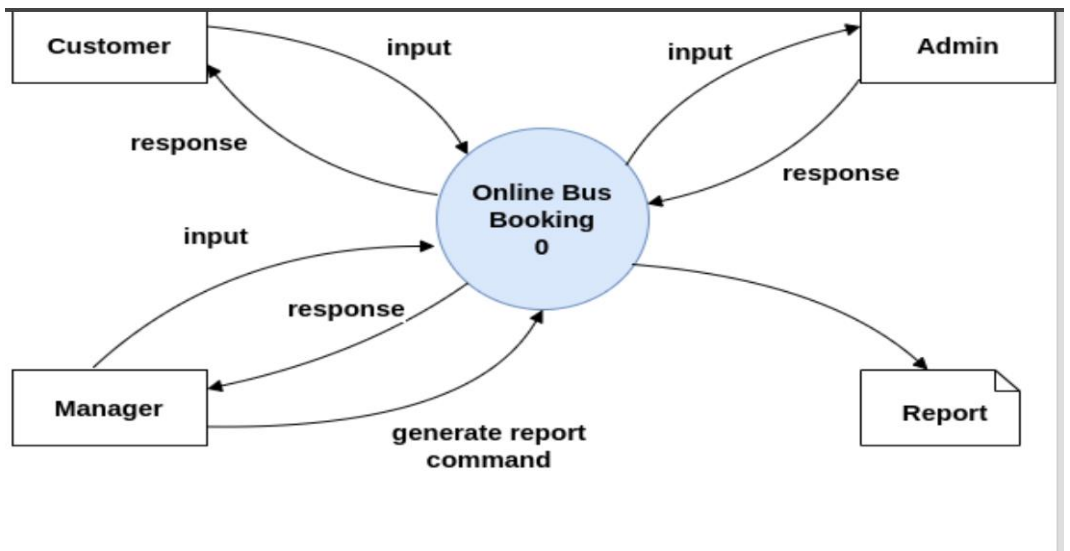
4.4.1 UPDATE BUS DETAILS



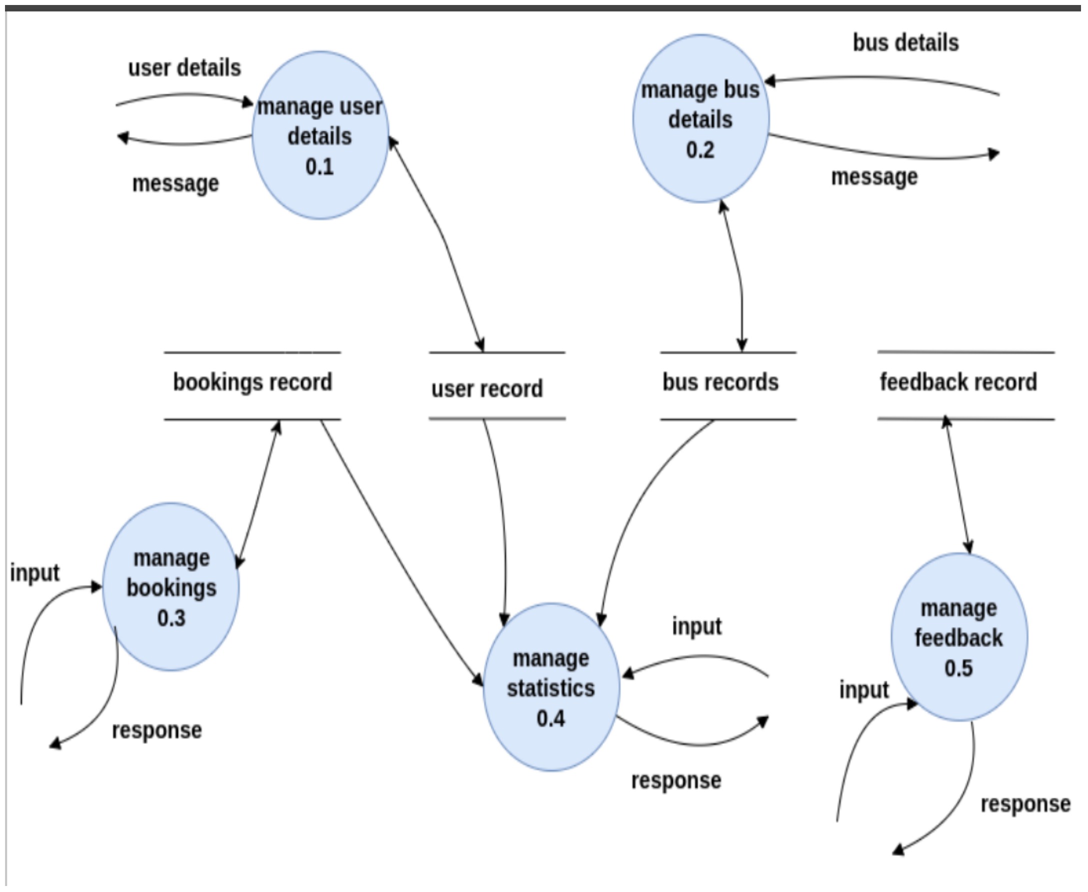
4.4.2 BUS BOOKINGS



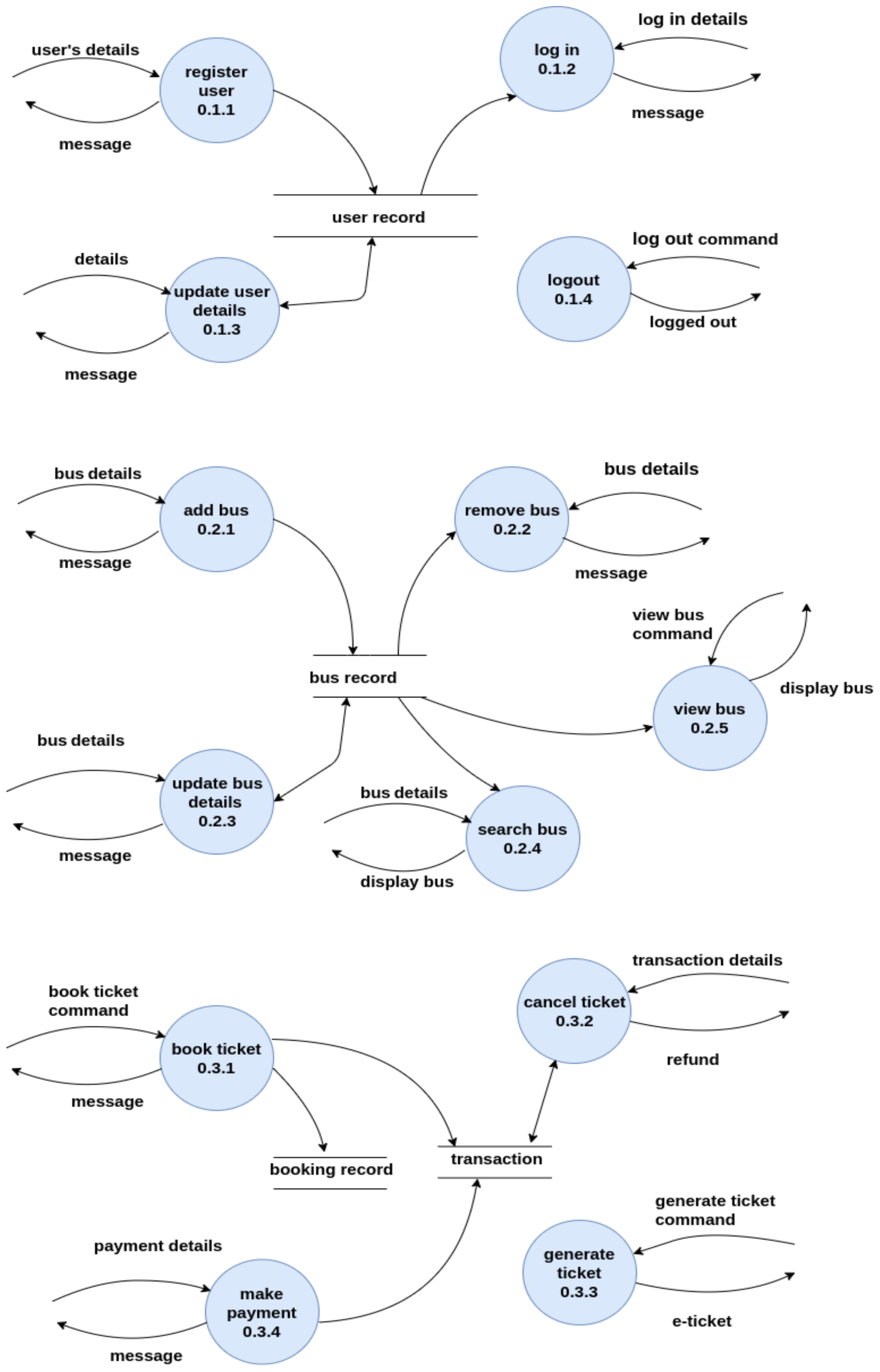
4.5 DATA FLOW DIAGRAM



LEVEL 0 (CONTEXT DIAGRAM)

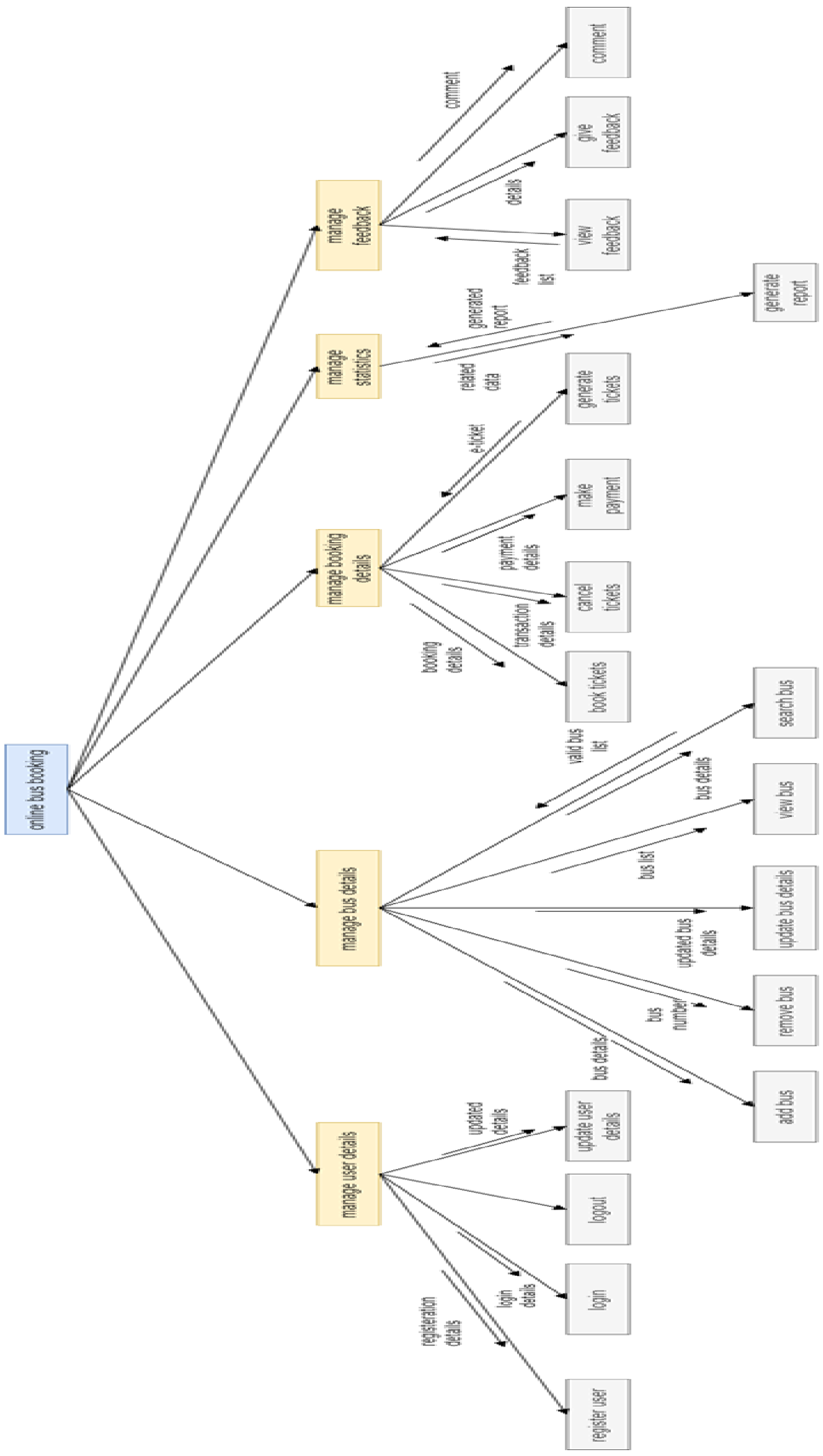


LEVEL 1


user's details
register user 0.1.1
message
log in details
log in 0.1.2
message
user record
details
update user details 0.1.3
message
log out command
logout 0.1.4
logged out
bus details
add bus 0.2.1
message
bus details
remove bus 0.2.2
message
view bus command
display bus
view bus 0.2.5
bus record
bus details
update bus details 0.2.3
message
bus details
search bus 0.2.4
display bus
transaction details
cancel ticket 0.3.2
refund
book ticket command
book ticket 0.3.1
message
booking record
transaction
generate ticket command
generate ticket 0.3.3
e-ticket
payment details
make payment 0.3.4
message

LEVEL 2

4.6 STRUCTURE CHART



5. IMPLEMENTATION DETAILS

The system consists of 4 basic modules namely
1. Login Module
2. Bus Module
3. Booking Module
4. Feedback Module

Each module consists of several methods to implement
the required functionality. Implementation is done
using Django. Database used in these modules is SQLite3.

LOGIN MODULE

This module functions mainly for management of users. It is used for registration, authorization, login, view profile, update profile, delete account, forgot password and logout for users. It also includes the functionality for displaying home page (which contains all other functionalities) for specific user.

BUS MODULE

This module handles all the functionalities related to buses like add bus, see bus, search bus, update bus details etc specific to user.

BOOKING MODULE

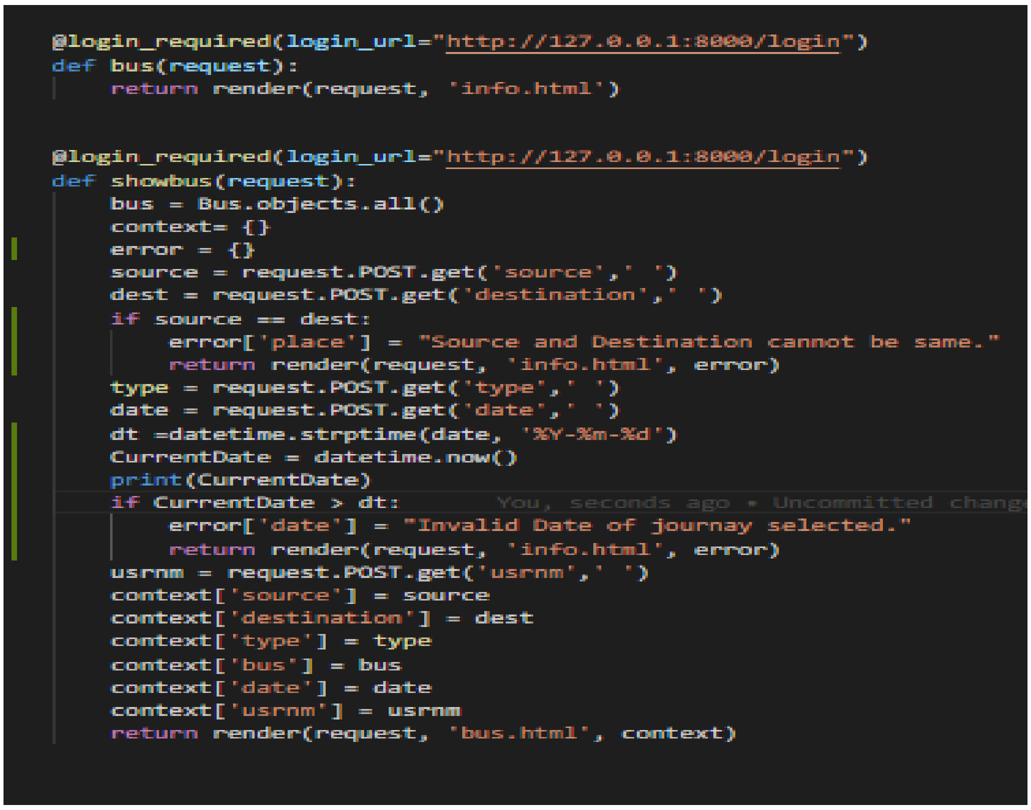
This module works for ticket reservation, ticket cancellation, fetch ticket, download ticket, see transactions etc specific to user.

FEEDBACK MODULE

This module functions for the management of feedbacks. The functionalities available here are give feedback, view feedback.

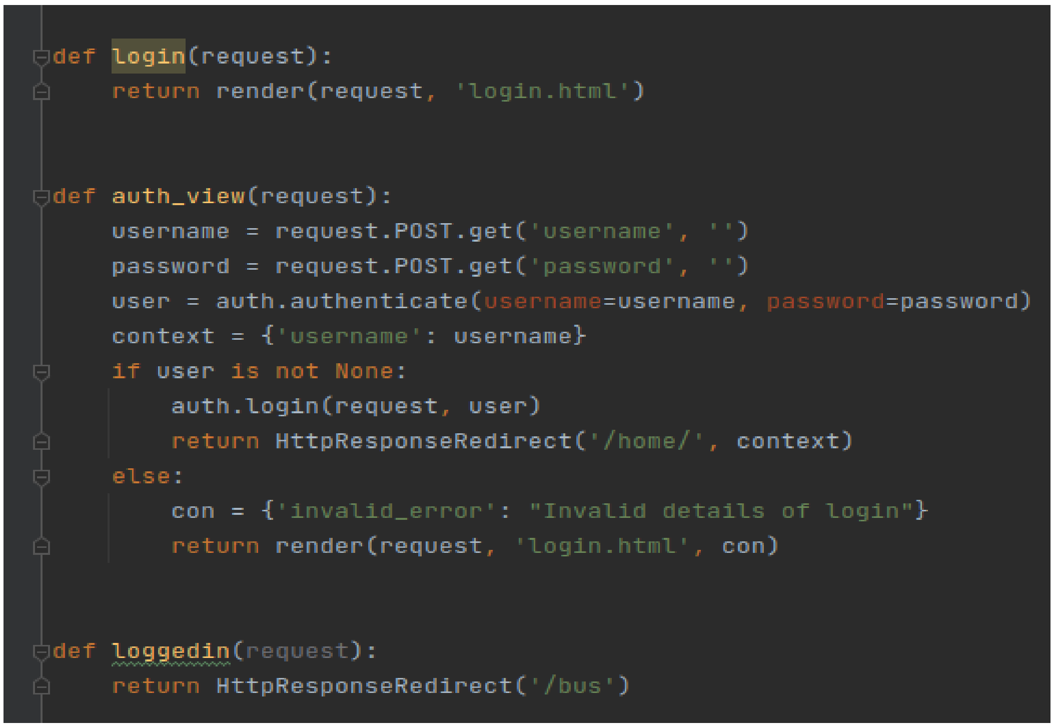
CODE:

BUS MODULE

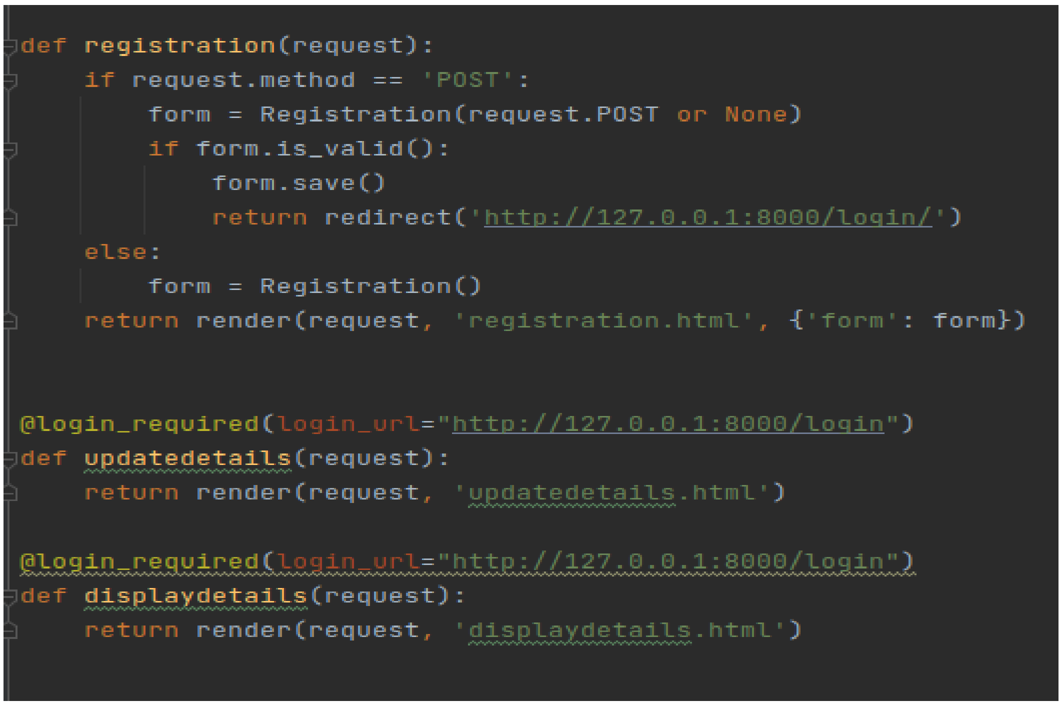


```
@login_required(login_url="http://127.0.0.1:8000/login")
def bus(request):
    return render(request, 'info.html')

@login_required(login_url="http://127.0.0.1:8000/login")
def showbus(request):
    bus = Bus.objects.all()
    context= {}
    error = {}
    source = request.POST.get('source',' ')
    dest = request.POST.get('destination',' ')
    if source == dest:
        error['place'] = "Source and Destination cannot be same."
        return render(request, 'info.html', error)
    type = request.POST.get('type',' ')
    date = request.POST.get('date',' ')
    dt =datetime.strptime(date, '%Y-%m-%d')
    CurrentDate = datetime.now()
    print(CurrentDate)
    if CurrentDate > dt:          You, seconds ago • Uncommitted change
        error['date'] = "Invalid Date of journay selected."
        return render(request, 'info.html', error)
    usrnm = request.POST.get('usrnm',' ')
    context['source'] = source
    context['destination'] = dest
    context['type'] = type
    context['bus'] = bus
    context['date'] = date
    context['usrnm'] = usrnm
    return render(request, 'bus.html', context)
```

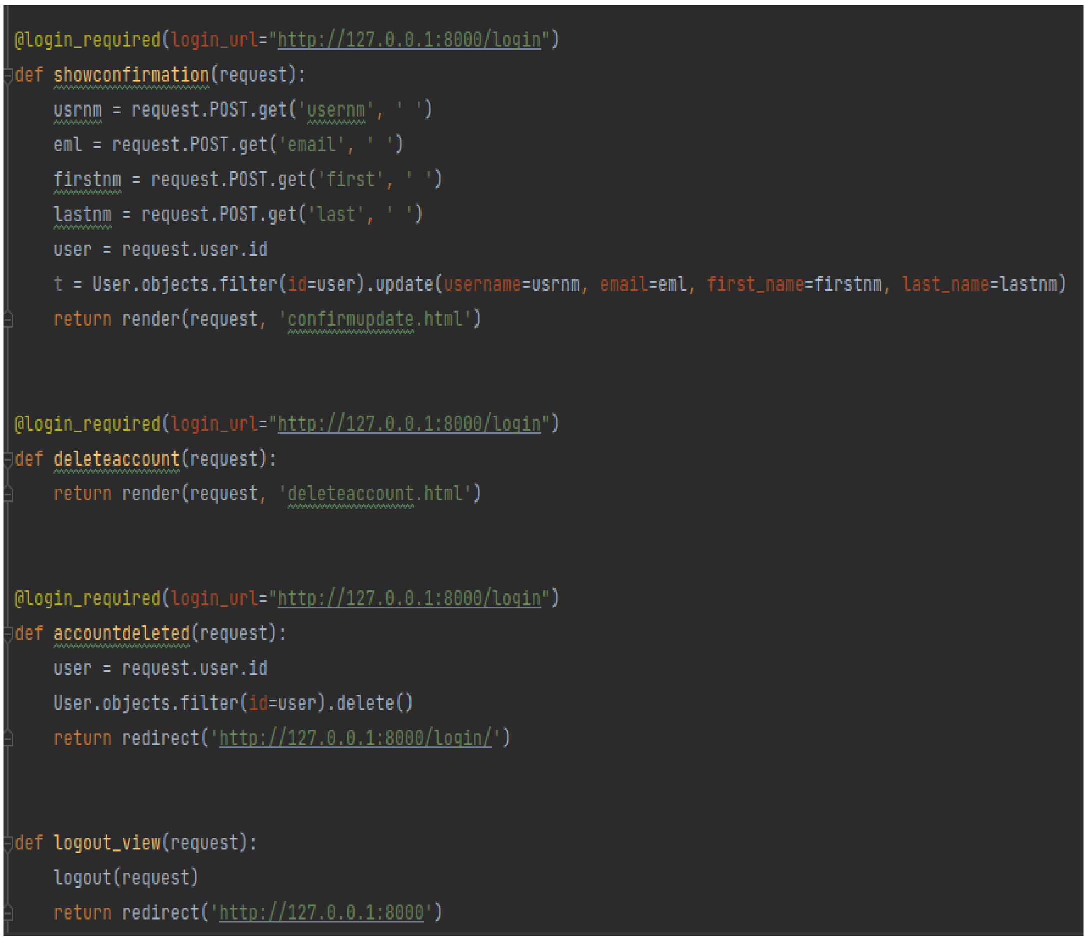
LOGIN MODULE



```
def login(request):
    return render(request, 'login.html')


def auth_view(request):
    username = request.POST.get('username', '')
    password = request.POST.get('password', '')
    user = auth.authenticate(username=username, password=password)
    context = {'username': username}
    if user is not None:
        auth.login(request, user)
        return HttpResponseRedirect('/home/', context)
    else:
        con = {'invalid_error': "Invalid details of login"}
        return render(request, 'login.html', con)


def loggedin(request):
    return HttpResponseRedirect('/bus')
```

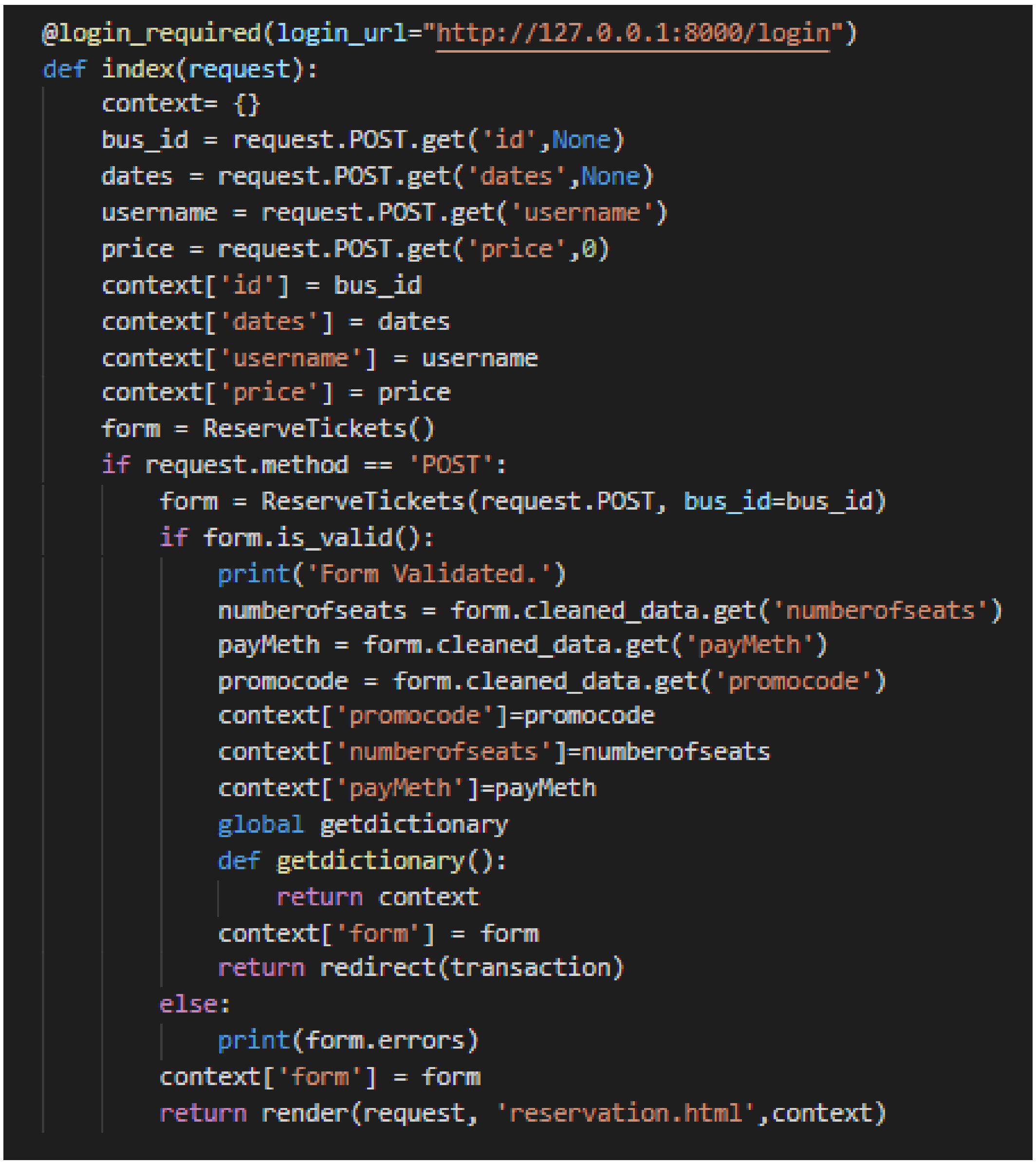


```
def registration(request):
    if request.method == 'POST':
        form = Registration(request.POST or None)
        if form.is_valid():
            form.save()
            return redirect('http://127.0.0.1:8000/login/')
    else:
        form = Registration()
    return render(request, 'registration.html', {'form': form})


@login_required(login_url="http://127.0.0.1:8000/login")
def updatedetails(request):
    return render(request, 'updatedetails.html')

@login_required(login_url="http://127.0.0.1:8000/login")
def displaydetails(request):
    return render(request, 'displaydetails.html')
```

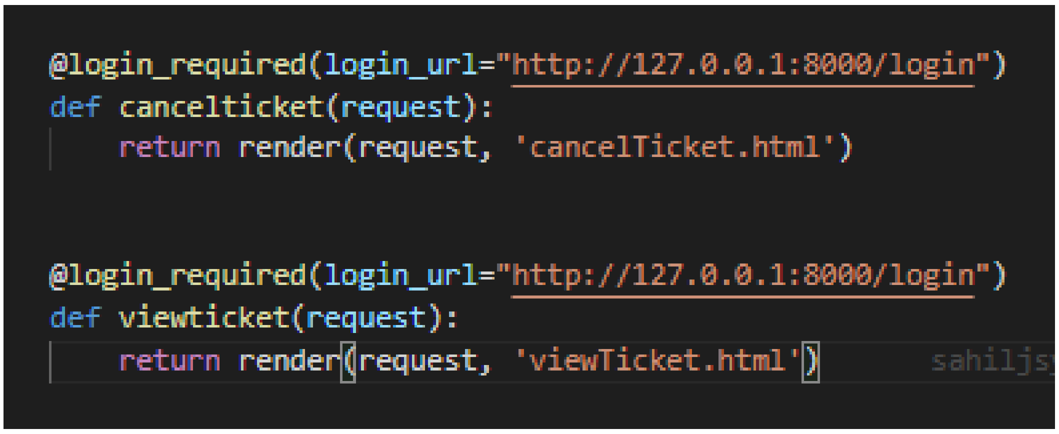


```
@login_required(login_url="http://127.0.0.1:8000/login")
def showconfirmation(request):
    usrnm = request.POST.get('usernm', ' ')
    eml = request.POST.get('email', ' ')
    firstnm = request.POST.get('first', ' ')
    lastnm = request.POST.get('last', ' ')
    user = request.user.id
    t = User.objects.filter(id=user).update(username=usrnm, email=eml, first_name=firstnm, last_name=lastnm)
    return render(request, 'confirmupdate.html')


@login_required(login_url="http://127.0.0.1:8000/login")
def deleteaccount(request):
    return render(request, 'deleteaccount.html')


@login_required(login_url="http://127.0.0.1:8000/login")
def accountdeleted(request):
    user = request.user.id
    User.objects.filter(id=user).delete()
    return redirect('http://127.0.0.1:8000/login/')


def logout_view(request):
    logout(request)
    return redirect('http://127.0.0.1:8000')
```

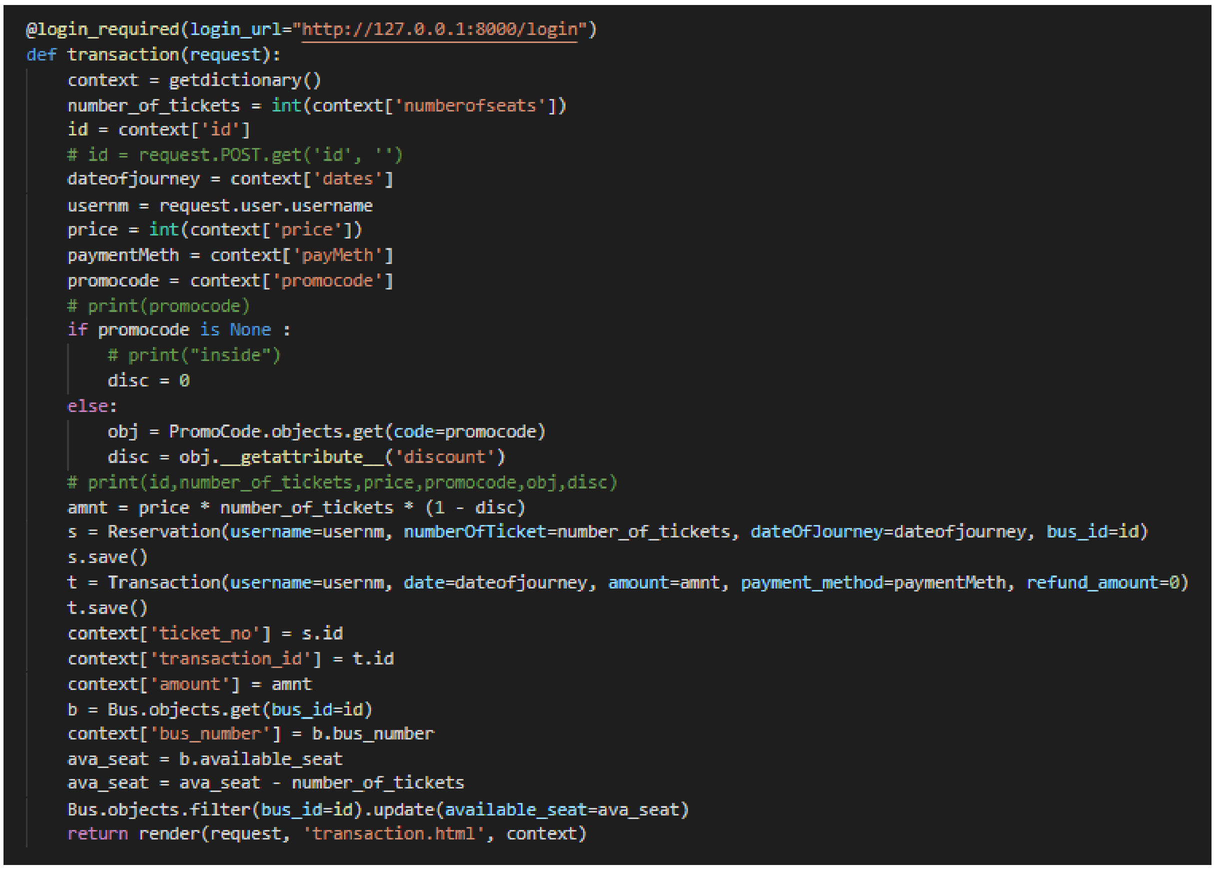
BOOKING MODULE



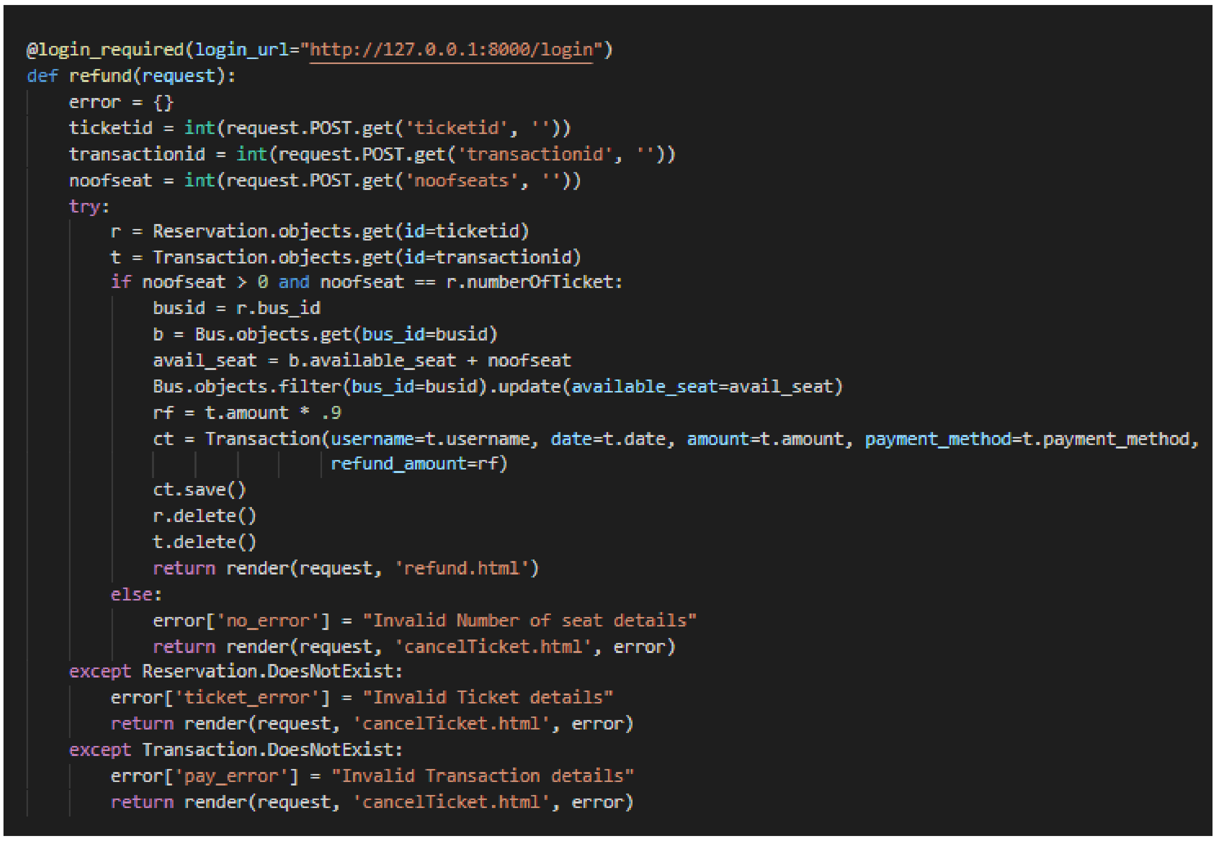
```python
@login_required(login_url="http://127.0.0.1:8000/login")
def index(request):
    context= {}
    bus_id = request.POST.get('id',None)
    dates = request.POST.get('dates',None)
    username = request.POST.get('username')
    price = request.POST.get('price',0)
    context['id'] = bus_id
    context['dates'] = dates
    context['username'] = username
    context['price'] = price
    form = ReserveTickets()
    if request.method == 'POST':
        form = ReserveTickets(request.POST, bus_id=bus_id)
        if form.is_valid():
            print('Form Validated.')
            numberofseats = form.cleaned_data.get('numberofseats')
            payMeth = form.cleaned_data.get('payMeth')
            promocode = form.cleaned_data.get('promocode')
            context['promocode']=promocode
            context['numberofseats']=numberofseats
            context['payMeth']=payMeth
            global getdictionary
            def getdictionary():
                return context
            context['form'] = form
            return redirect(transaction)
        else:
            print(form.errors)
        context['form'] = form
        return render(request, 'reservation.html',context)
```



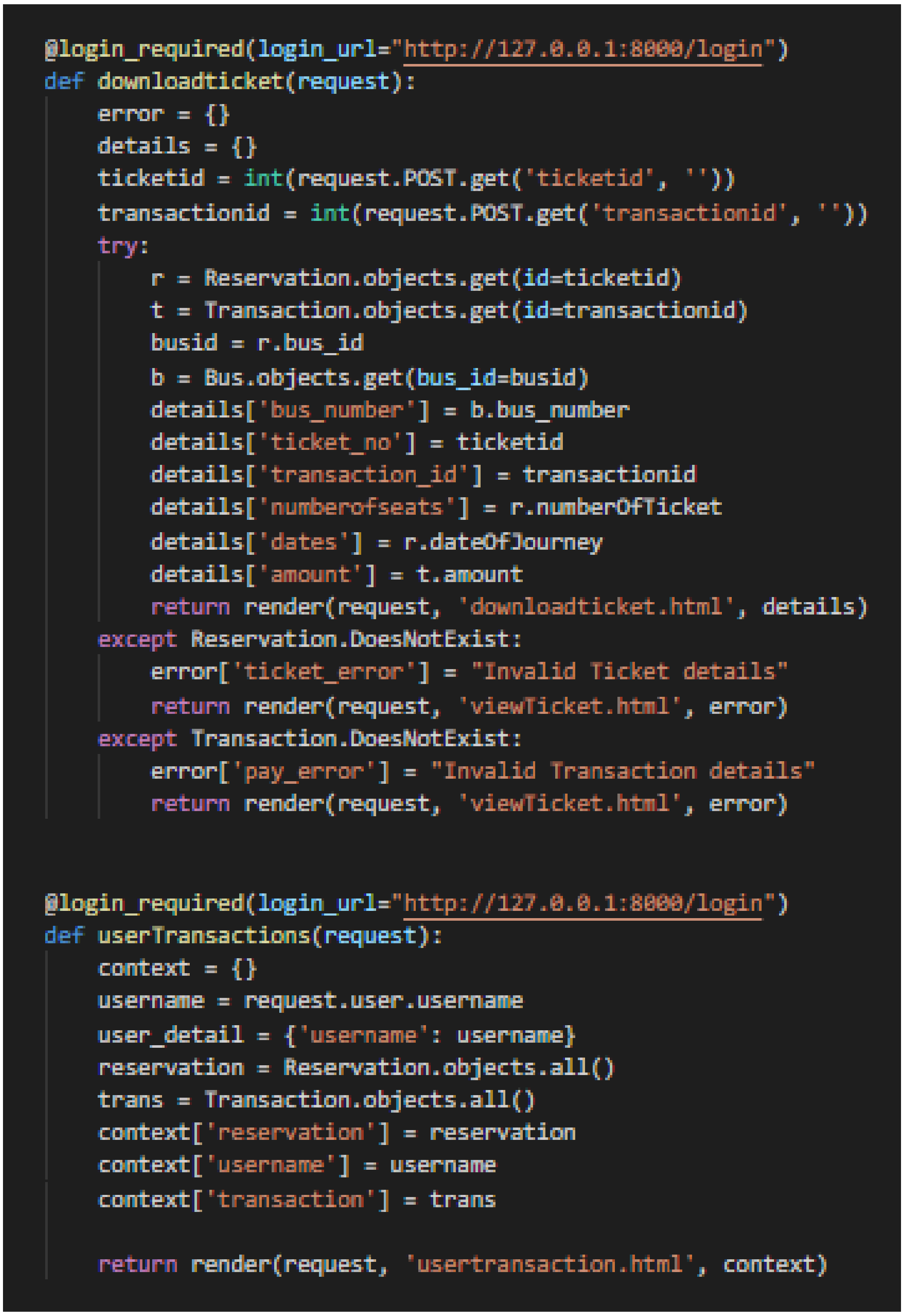
```python
@login_required(login_url="http://127.0.0.1:8000/login")
def cancelticket(request):
    return render(request, 'cancelTicket.html')


@login_required(login_url="http://127.0.0.1:8000/login")
def viewticket(request):
    return render(request, 'viewTicket.html')        sahiljs
```



```
@login_required(login_url="http://127.0.0.1:8000/login")
def transaction(request):
    context = getdictionary()
    number_of_tickets = int(context['numberofseats'])
    id = context['id']
    # id = request.POST.get('id', '')
    dateofjourney = context['dates']
    usernm = request.user.username
    price = int(context['price'])
    paymentMeth = context['payMeth']
    promocode = context['promocode']
    # print(promocode)
    if promocode is None :
        # print("inside")
        disc = 0
    else:
        obj = PromoCode.objects.get(code=promocode)
        disc = obj.__getattribute__('discount')
    # print(id,number_of_tickets,price,promocode,obj,disc)
    amnt = price * number_of_tickets * (1 - disc)
    s = Reservation(username=usernm, numberOfTicket=number_of_tickets, dateOfJourney=dateofjourney, bus_id=id)
    s.save()
    t = Transaction(username=usernm, date=dateofjourney, amount=amnt, payment_method=paymentMeth, refund_amount=0)
    t.save()
    context['ticket_no'] = s.id
    context['transaction_id'] = t.id
    context['amount'] = amnt
    b = Bus.objects.get(bus_id=id)
    context['bus_number'] = b.bus_number
    ava_seat = b.available_seat
    ava_seat = ava_seat - number_of_tickets
    Bus.objects.filter(bus_id=id).update(available_seat=ava_seat)
    return render(request, 'transaction.html', context)
```



```
@login_required(login_url="http://127.0.0.1:8000/login")
def refund(request):
    error = {}
    ticketid = int(request.POST.get('ticketid', ''))
    transactionid = int(request.POST.get('transactionid', ''))
    noofseat = int(request.POST.get('noofseats', ''))
    try:
        r = Reservation.objects.get(id=ticketid)
        t = Transaction.objects.get(id=transactionid)
        if noofseat > 0 and noofseat == r.numberOfTicket:
            busid = r.bus_id
            b = Bus.objects.get(bus_id=busid)
            avail_seat = b.available_seat + noofseat
            Bus.objects.filter(bus_id=busid).update(available_seat=avail_seat)
            rf = t.amount * .9
            ct = Transaction(username=t.username, date=t.date, amount=t.amount, payment_method=t.payment_method,
                             refund_amount=rf)
            ct.save()
            r.delete()
            t.delete()
            return render(request, 'refund.html')
        else:
            error['no_error'] = "Invalid Number of seat details"
            return render(request, 'cancelTicket.html', error)
    except Reservation.DoesNotExist:
        error['ticket_error'] = "Invalid Ticket details"
        return render(request, 'cancelTicket.html', error)
    except Transaction.DoesNotExist:
        error['pay_error'] = "Invalid Transaction details"
        return render(request, 'cancelTicket.html', error)
```



```python
@login_required(login_url="http://127.0.0.1:8000/login")
def downloadticket(request):
    error = {}
    details = {}
    ticketid = int(request.POST.get('ticketid', ''))
    transactionid = int(request.POST.get('transactionid', ''))
    try:
        r = Reservation.objects.get(id=ticketid)
        t = Transaction.objects.get(id=transactionid)
        busid = r.bus_id
        b = Bus.objects.get(bus_id=busid)
        details['bus_number'] = b.bus_number
        details['ticket_no'] = ticketid
        details['transaction_id'] = transactionid
        details['numberofseats'] = r.numberOfTicket
        details['dates'] = r.dateOfJourney
        details['amount'] = t.amount
        return render(request, 'downloadticket.html', details)
    except Reservation.DoesNotExist:
        error['ticket_error'] = "Invalid Ticket details"
        return render(request, 'viewTicket.html', error)
    except Transaction.DoesNotExist:
        error['pay_error'] = "Invalid Transaction details"
        return render(request, 'viewTicket.html', error)


@login_required(login_url="http://127.0.0.1:8000/login")
def userTransactions(request):
    context = {}
    username = request.user.username
    user_detail = {'username': username}
    reservation = Reservation.objects.all()
    trans = Transaction.objects.all()
    context['reservation'] = reservation
    context['username'] = username
    context['transaction'] = trans

    return render(request, 'usertransaction.html', context)
```

6. WORK FLOW/ LAYOUT WITH TESTING

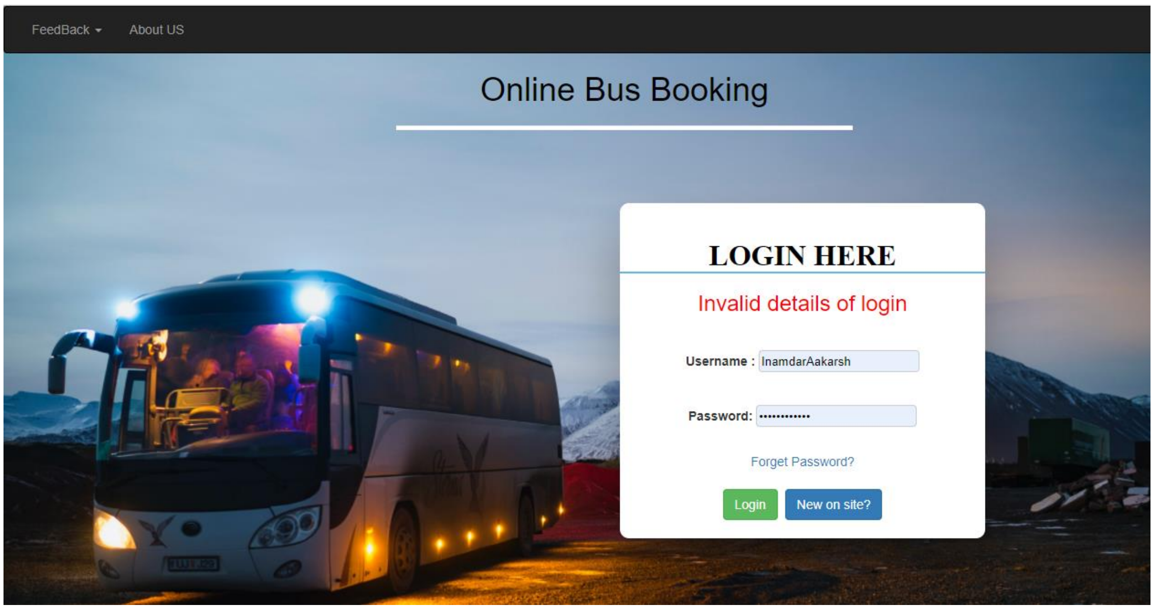
MANUAL TEST CASES:

SR no.	Test Scenario	Expected Result	Actual Result	Status
1.	Login with Invalid Details	Error Message	Error Message Display	PASS
2.	Login with correct Details	Redirect to Home Page	Redirected to Home Page	PASS
3.	Cancel Ticket with Wrong Ticket id	Error message	Error Message Displayed	PASS
4.	Cancel Ticket with correct details	Ticket Canceled and message	Ticket canceled and refund generated message displayed	PASS
5.	Give feedback	Redirect to greeting page	Redirected to greeting page	PASS
6.	Give feedback with invalid ratings	Error Message	Error Message	PASS
7.	View feedback	Display all feedback	All feedbacks are displayed	PASS
8.	View Profile	Display User's current details	User's current details displayed	PASS
9.	Update Profile	Display User's updated details	User's updated details displayed	PASS
10.	Search bus with invalid details	Error message	Error message displayed	PASS
11.	Search bus with Correct details	Display All valid buses	All valid buses displayed	PASS
12.	Book seats with wrong details	Error message	Error message displayed	PASS

13.	Book seats with correct details	Book seats and display ticket	Ticket displayed with ticket id	PASS
14.	Fetch Ticket with wrong details	Error message	Error message displayed	PASS
15.	Fetch ticket with correct details	Display ticket	Ticket displayed	PASS
16.	Delete account	Remove user’s account	User’s record deleted and redirected to login page	PASS
17.	Try to open any page of system without login which required login	Redirect to login page	Redirected to login page	PASS
18.	Forgot password	Send link via email after verification	After verification reset password link sent via email	PASS

Screenshots of Work flow:

invalid login details

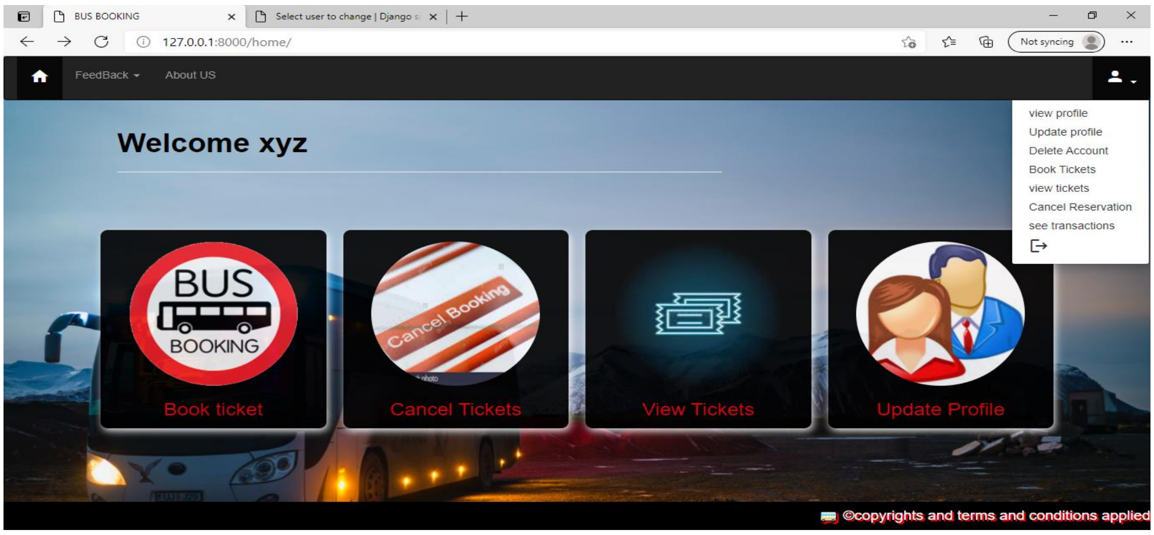


Invalid details of login

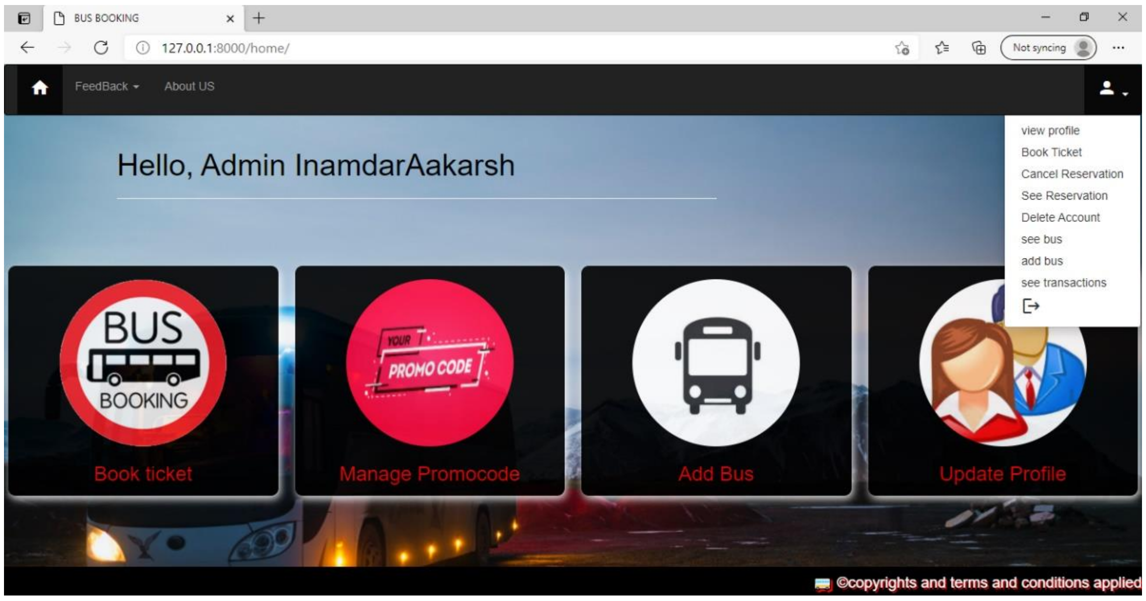
Username : InamdarAakarsh

Password: ••••••••••••

Valid login details

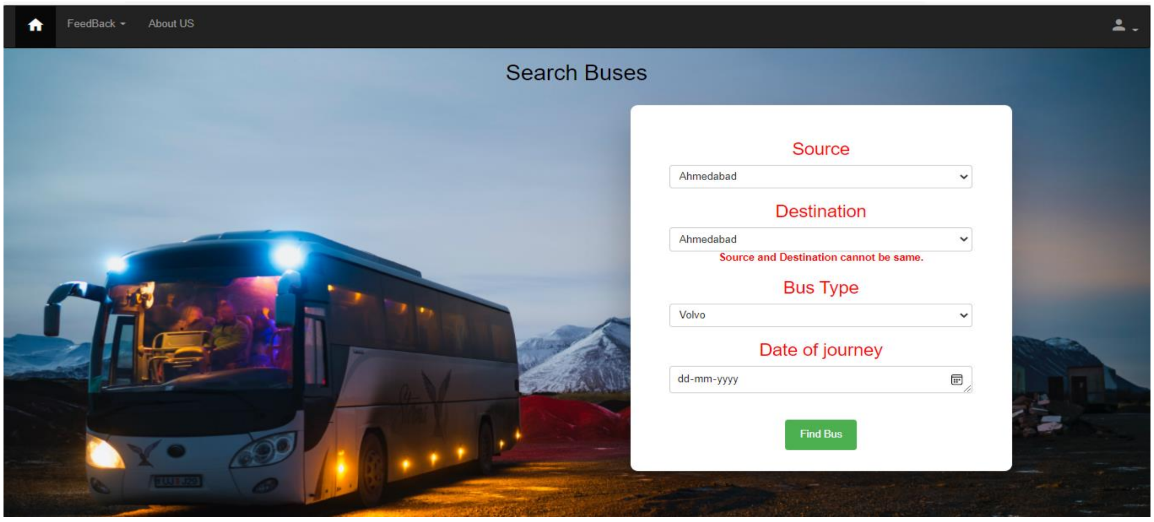


For user



For Admin

Booking Process:



Search Buses

Source

Ahmedabad

Destination

Ahmedabad

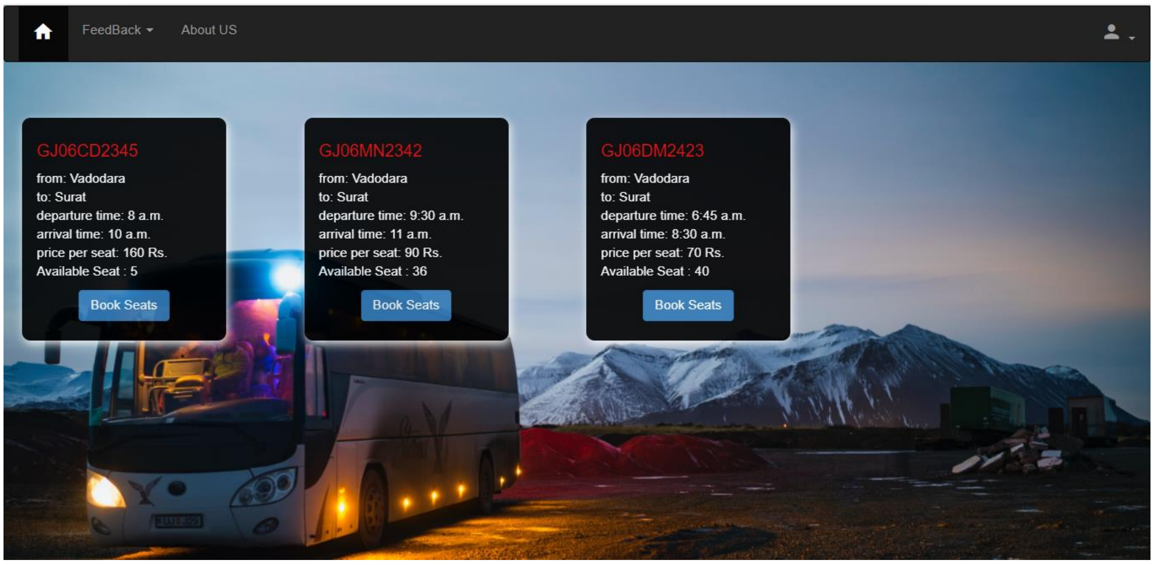
Source and Destination cannot be same.

Bus Type

Volvo

Date of journey

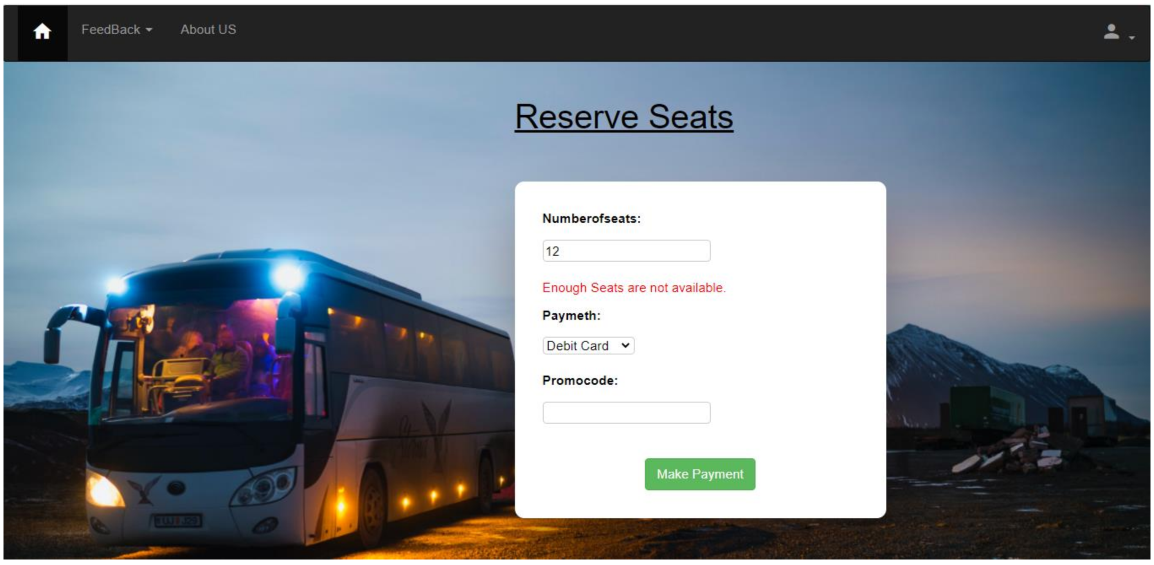
dd-mm-yyyy



GJ06CD2345
from: Vadodara
to: Surat
departure time: 8 a.m.
arrival time: 10 a.m.
price per seat: 160 Rs.
Available Seat : 5

GJ06MN2342
from: Vadodara
to: Surat
departure time: 9:30 a.m.
arrival time: 11 a.m.
price per seat: 90 Rs.
Available Seat : 36

GJ06DM2423
from: Vadodara
to: Surat
departure time: 6:45 a.m.
arrival time: 8:30 a.m.
price per seat: 70 Rs.
Available Seat : 40



Reserve Seats

Numberofseats:

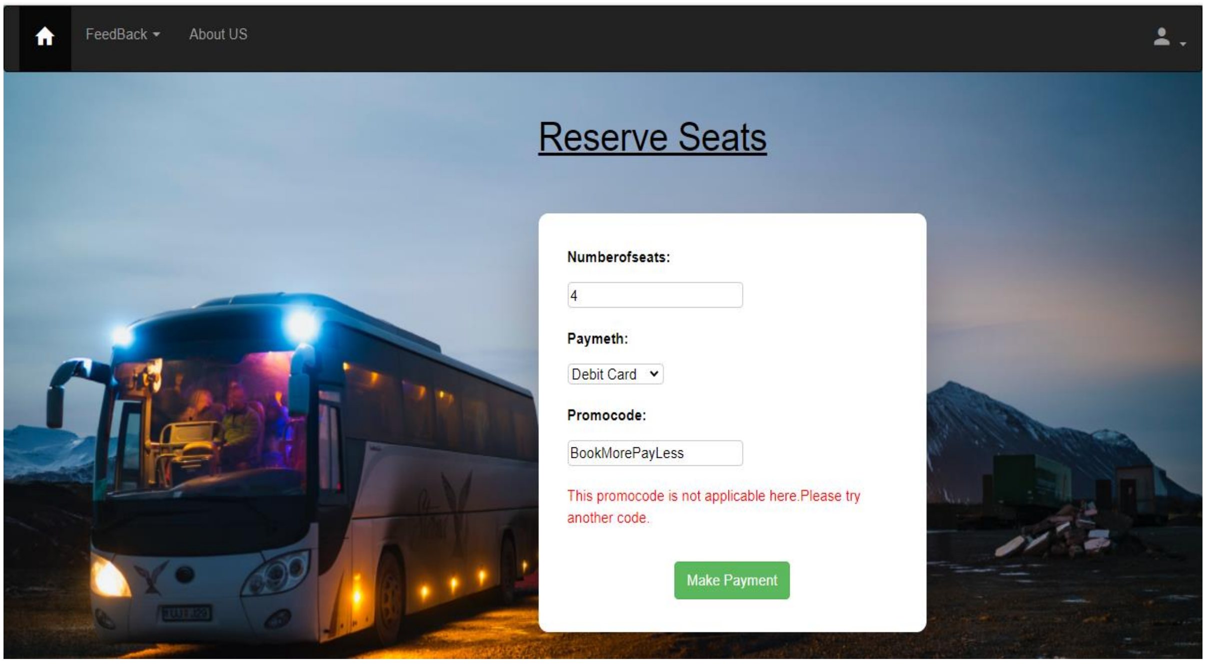
12

Enough Seats are not available.

Paymeth:

Debit Card

Promocode:

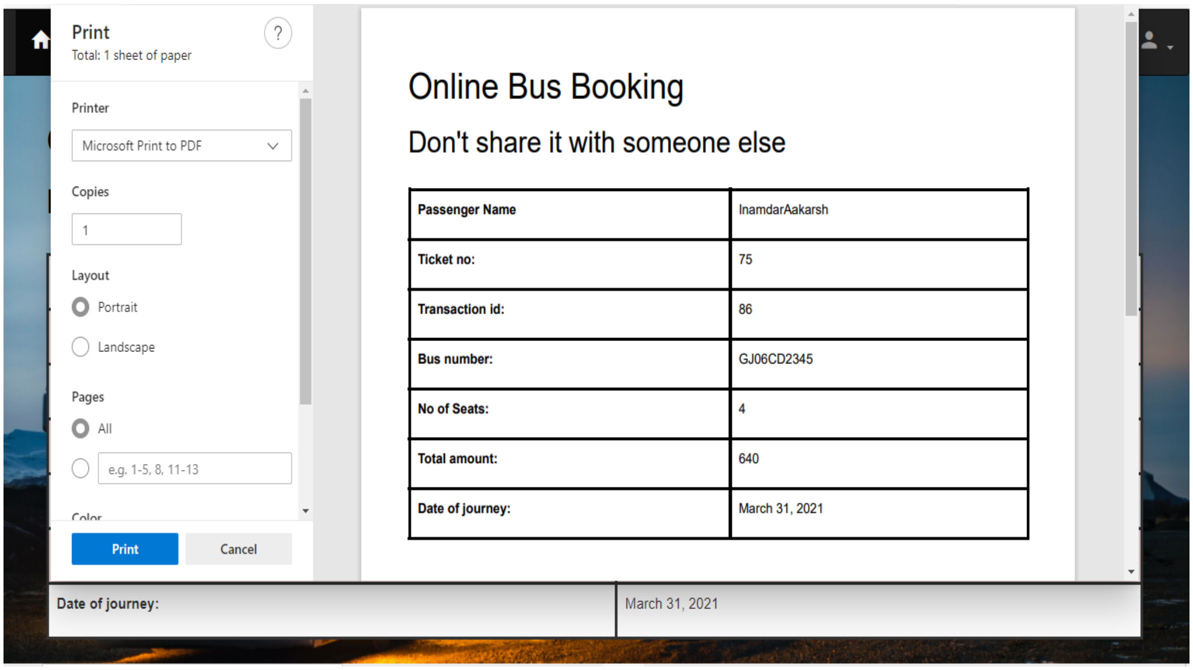


Reserve Seats

Numberofseats:	4
Paymeth:	Debit Card
Promocode:	BookMorePayLess

This promocode is not applicable here.Please try another code.

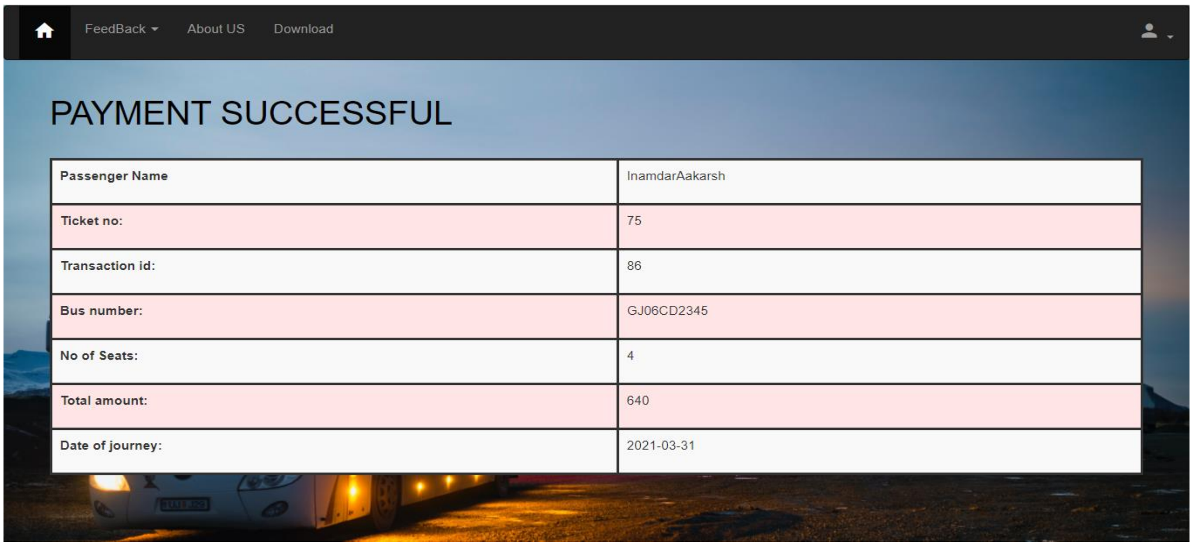
Download Ticket



Online Bus Booking

Don't share it with someone else

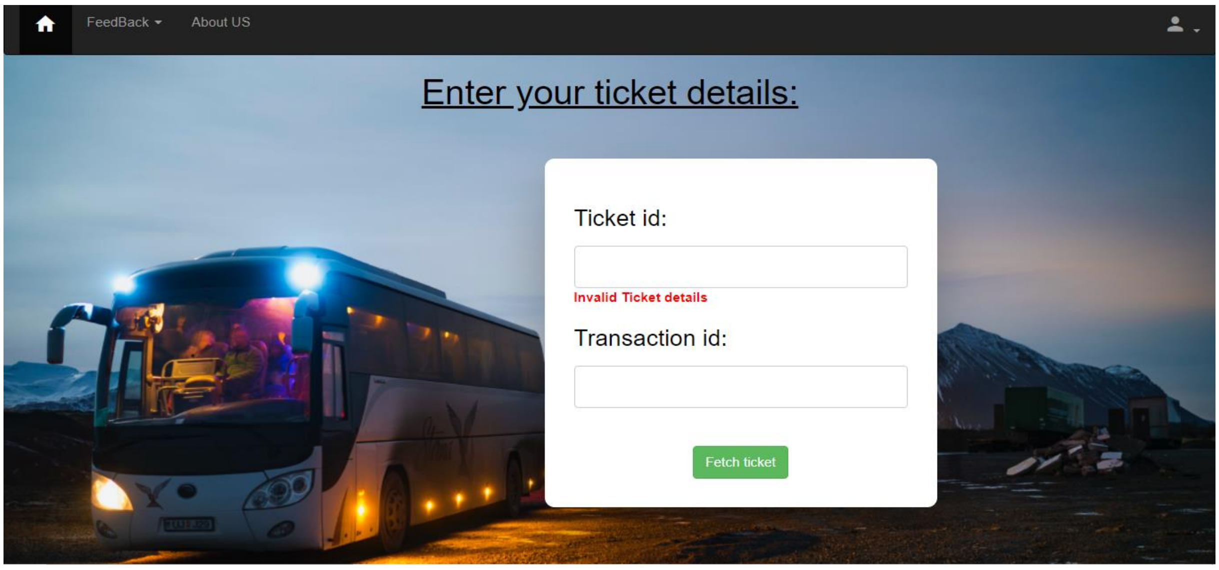
Passenger Name	InamdarAakarsh
Ticket no:	75
Transaction id:	86
Bus number:	GJ06CD2345
No of Seats:	4
Total amount:	640
Date of journey:	March 31, 2021



PAYMENT SUCCESSFUL

Passenger Name	InamdarAakarsh
Ticket no:	75
Transaction id:	86
Bus number:	GJ06CD2345
No of Seats:	4
Total amount:	640
Date of journey:	2021-03-31

Cancel Tickets

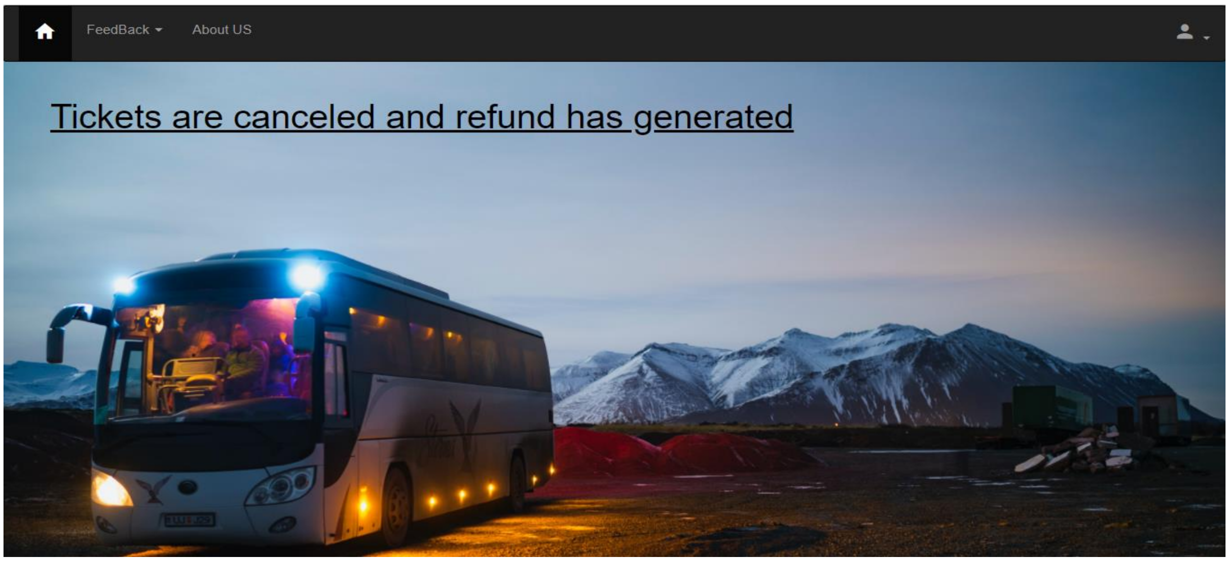


Enter your ticket details:

Ticket id:

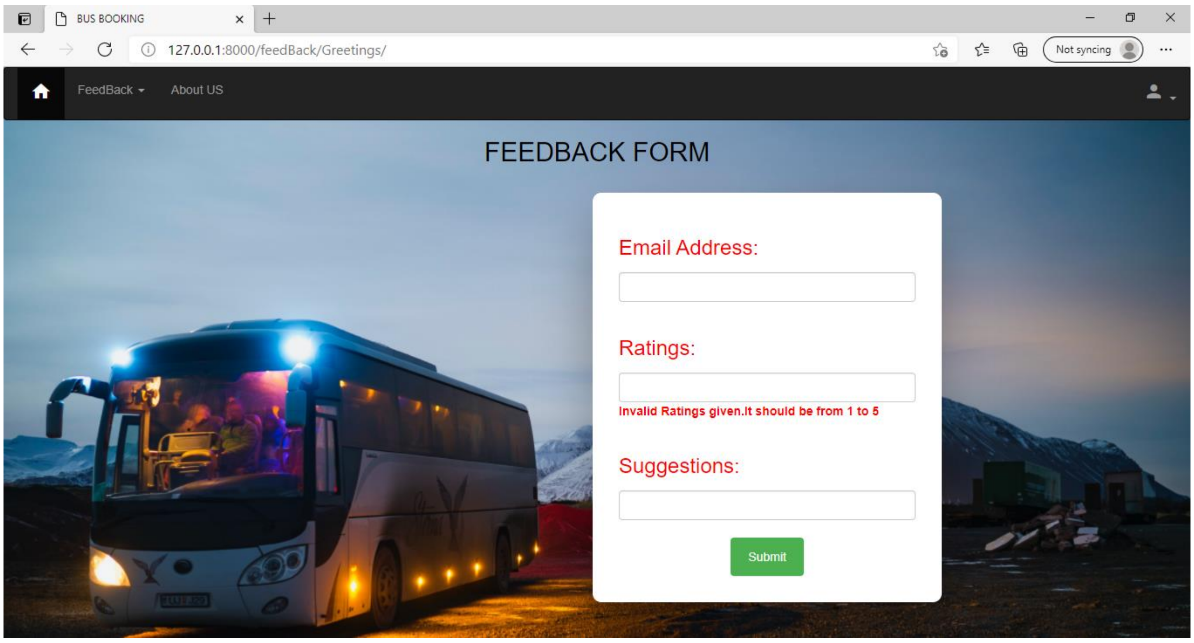
Invalid Ticket details

Transaction id:



Tickets are canceled and refund has generated

Feedback:



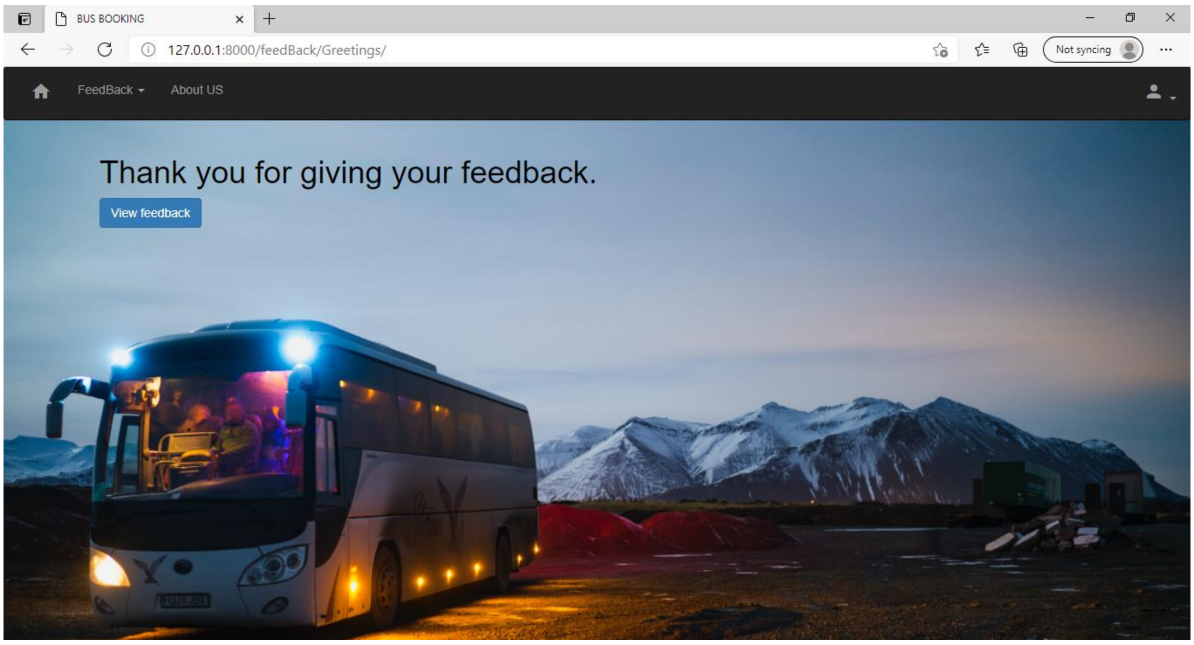
FEEDBACK FORM

Email Address:

Ratings:

Invalid Ratings given.It should be from 1 to 5

Suggestions:



Thank you for giving your feedback.

7. CONCLUSION

The functionalities are implemented in system after understanding all the system modules according to the requirements.

Functionalities that are successfully implemented in the system are:

- **User registration**
- **Login**
- **User authentication**
- **Logout**
- **Search Bus**
- **Display Bus**
- **Bus Booking without making payment (With/Without Promocode)**
- **Cancel Ticket**
- **Fetch Ticket**
- **Download Ticket**
- **See Previous Transactions**
- **Give Feedback**
- **View Feedback**
- **Display User's Details**
- **Update User's Details**
- **Delete Account**
- **Admin Panel using Django default with some modifications**

After the implementation and coding of system, comprehensive testing was performed on the system to determine the errors and possible flaws in the system.

BOOKING PROCESS, LOGIN PROCESS AND FEEDBACK MODULE HAS BEEN TESTED USING "SELENIUM PYTHON".

8. LIMITATIONS AND FUTURE EXTENSIONS

LIMITATION:

1. Actual payment gateway not created but database updation is done successfully.
2. Only Bus Booking for the cities Surat, Vadodara, Ahmedabad is available.

FUTURE EXTENSIONS:

1. Actual payment gateway system.
2. Report generation for manager.
3. For seat selection we can give bus top view so user can differentiate booked and available seats.
4. User profile view can be improved.

9. BIBLIOGRAPHY

* Stack Overflow
* YouTube
* GeeksForGeeks
* Selenium- https://selenium-python.readthedocs.io/
* Django - https://www.djangoproject.com