Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – IV

Subject: Software Engineering

Project title: Railway Reservation System

By:

1) Ishita Chauhan, Roll no: CE015 , Id: 19CEUOS140

2) Princy Gajera , Roll no: CE033 , Id: 19CEUOS157

Guided by: Prof. Jigar M. Pandya

Prof. Pinkal C. Chauhan

Contents

Abstract

Railway Reservation system is software for the purpose of reserving train seats at any time, anywhere. Railway Reservation system facilitates passenger about train available on basics of source and destination. In this system train records are maintained and retrieved. Administrator has complete access to database and can add train or cancel train. There are multiple options for payment and user can cancel the ticket as well.

Technologies/tools used

Technologies:

Django , Python , MySQL , Bootstrap , HTML , CSS

Tools:

Git , Visual Studio Code

Platform:

Local development server

Software Requirements Specifications

1. Admin module:

1.1 Verify admin:

- Description:

  Admins can select their rolls and needs to enter their details and verify themselves.

- Input:

  Information of admin.

2. User module

- **Add user:**
  - Description:

    Users will have to enter their details like full name, age, email-id, contact-no, date of birth, city, state.

  - Input:

    Information of users.

3. Reservation module:

3.1 Route:

- Description:

  It will show details of available route and timing to User's desired destination.

- Input:

  Users have to add their source and destination.

- Output:

  Show details of available route and timings.

3.2 Ticket booking:

- Description:

  User can book ticket.

- Input:

  Source and destination.

  User need to select route, time, train, seats from available option.

3.3 Payment:

- Description:

  Users need to pay given amount to book their ticket.

- Input:

  Users have to select payment method and fill details according to selected method.

- Output:

  Display Ticket.

3.4 Ticket cancellation and refund:

- Description:

  User can cancel their ticket and get refund.

- Input:

  Click request cancellation of order.

4. Train-Details module:

- Description:

  User can see arrival and departure timing of trains.

5. Feedback:

- Description:

  User can give feedback.

- Input:

  User's suggestions.

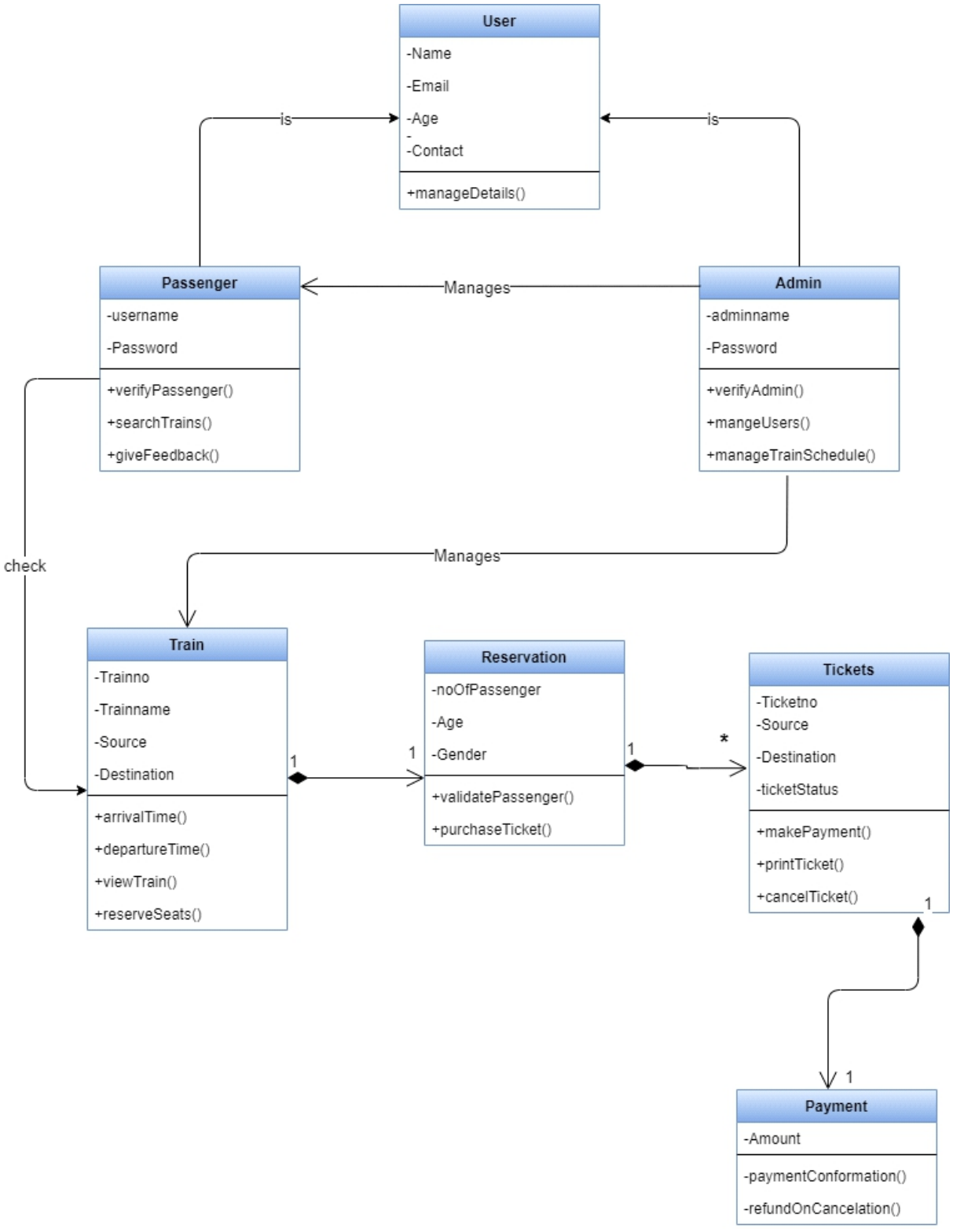
- Output:

  Message "Thanks for your feedback".

Design

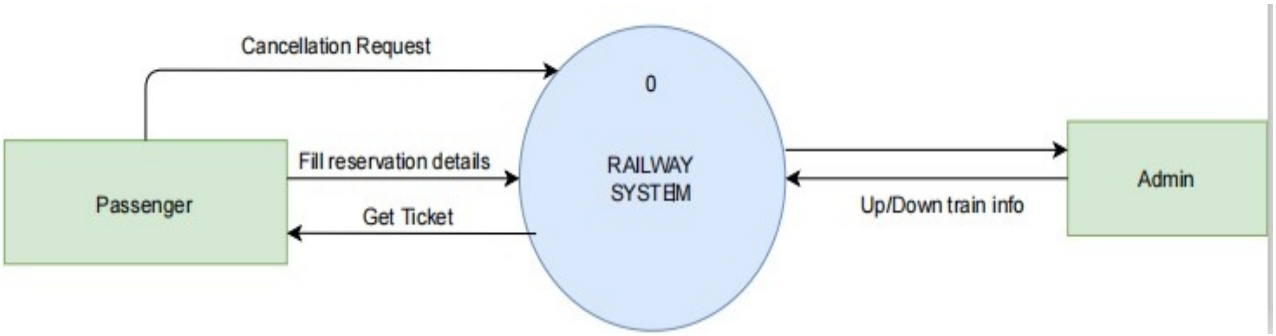
1. Use case diagram:

2. Class Diagram:

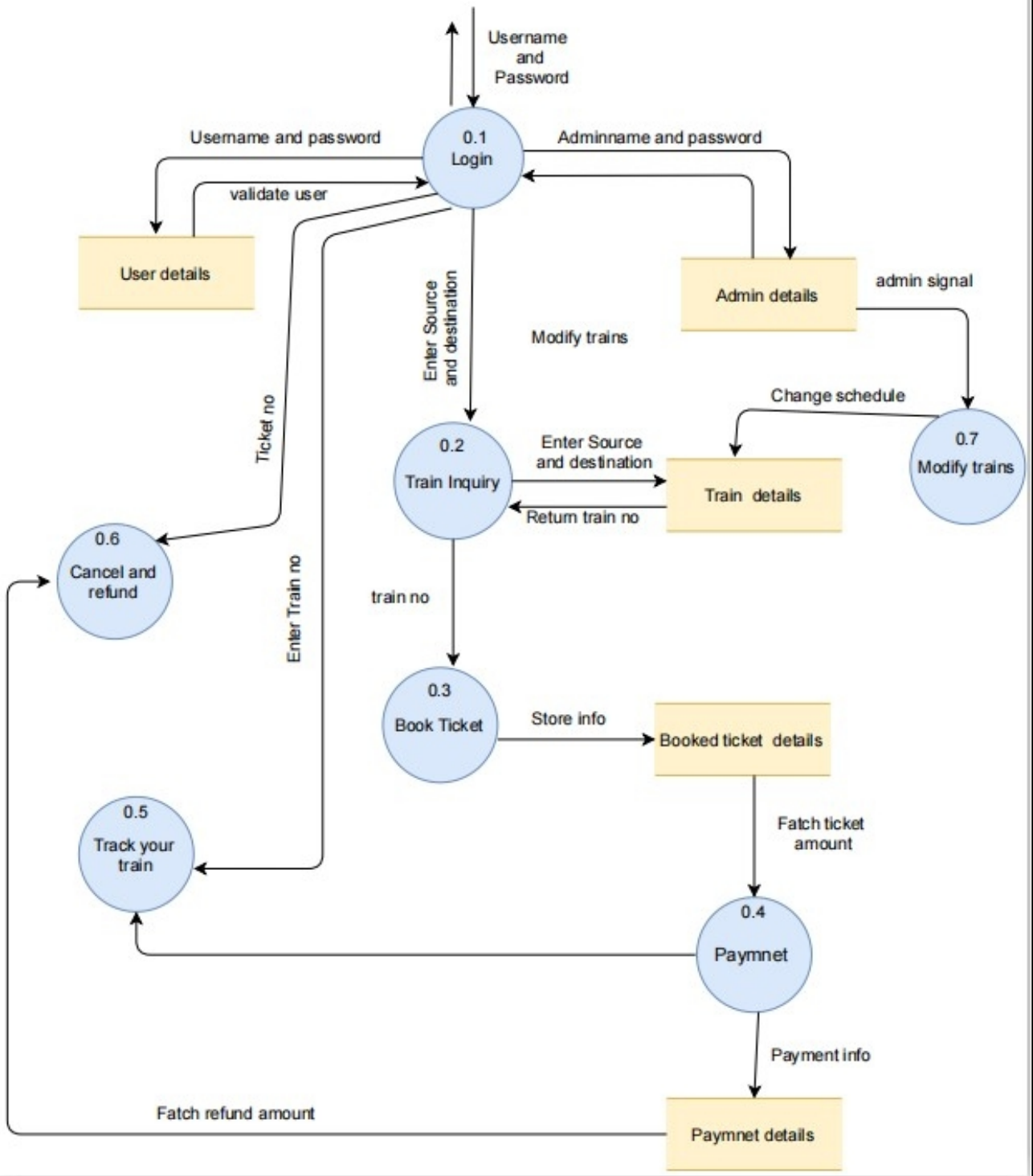


3. DFD Model

I. Level 0 : Context Diagram

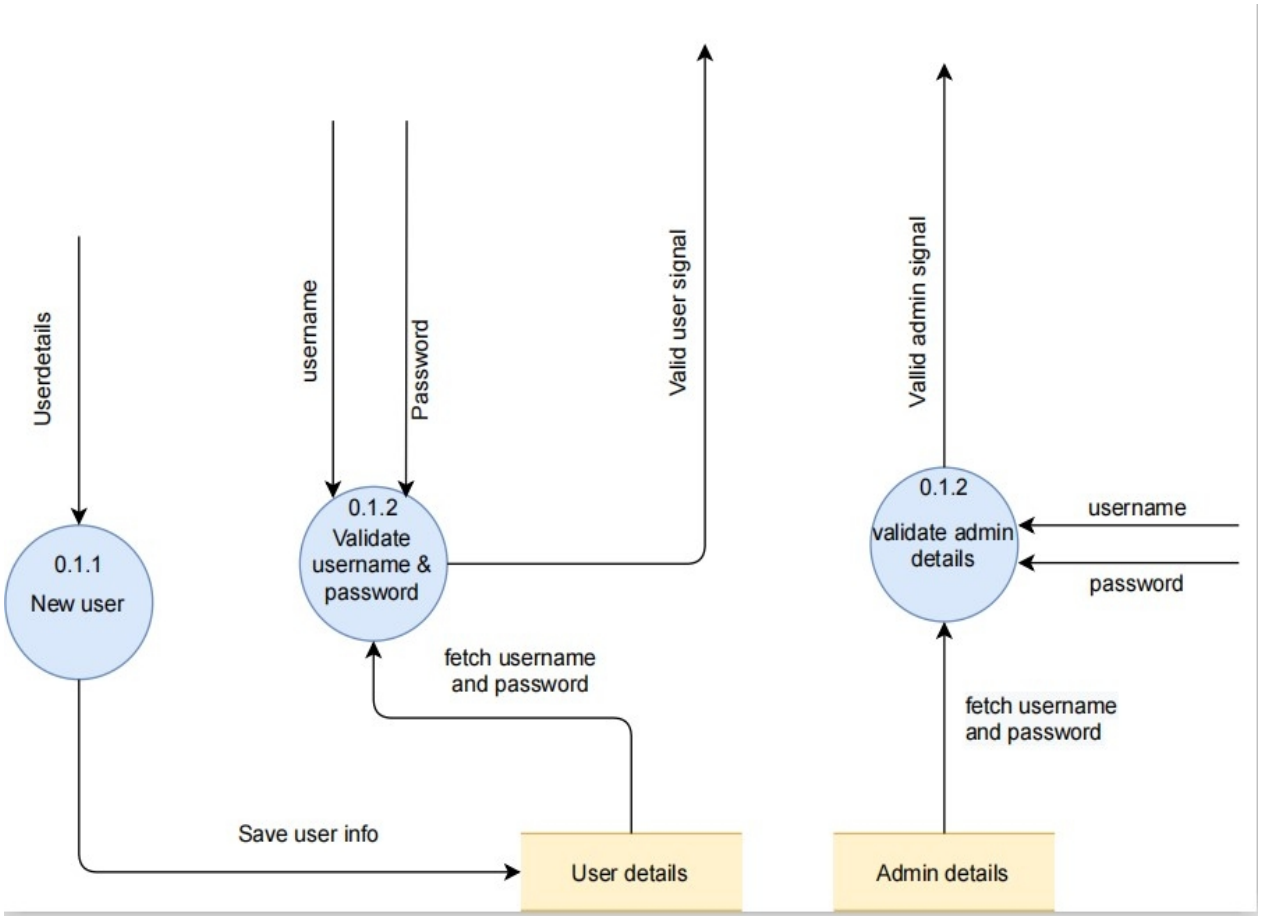


II. DFD Level 1 :

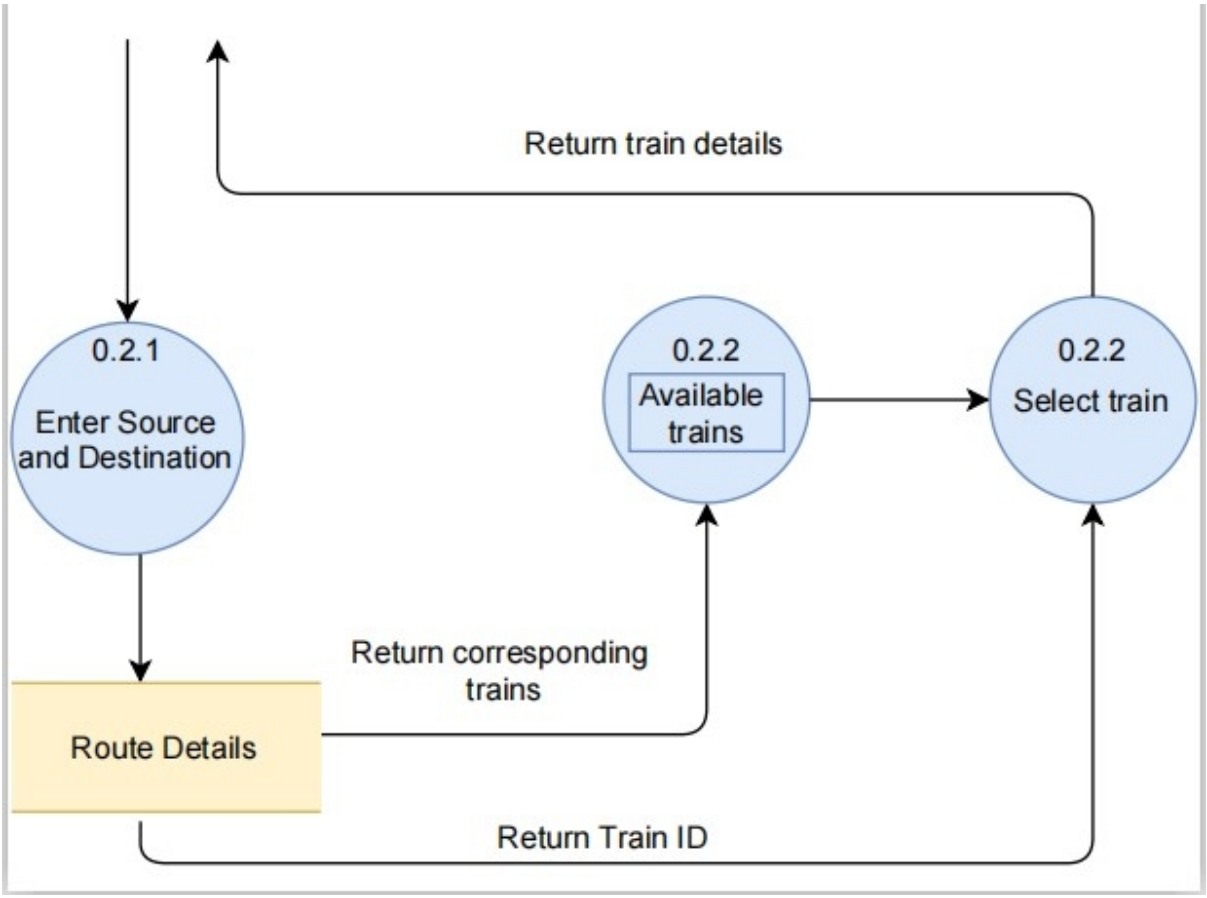


III. DFD Level 2

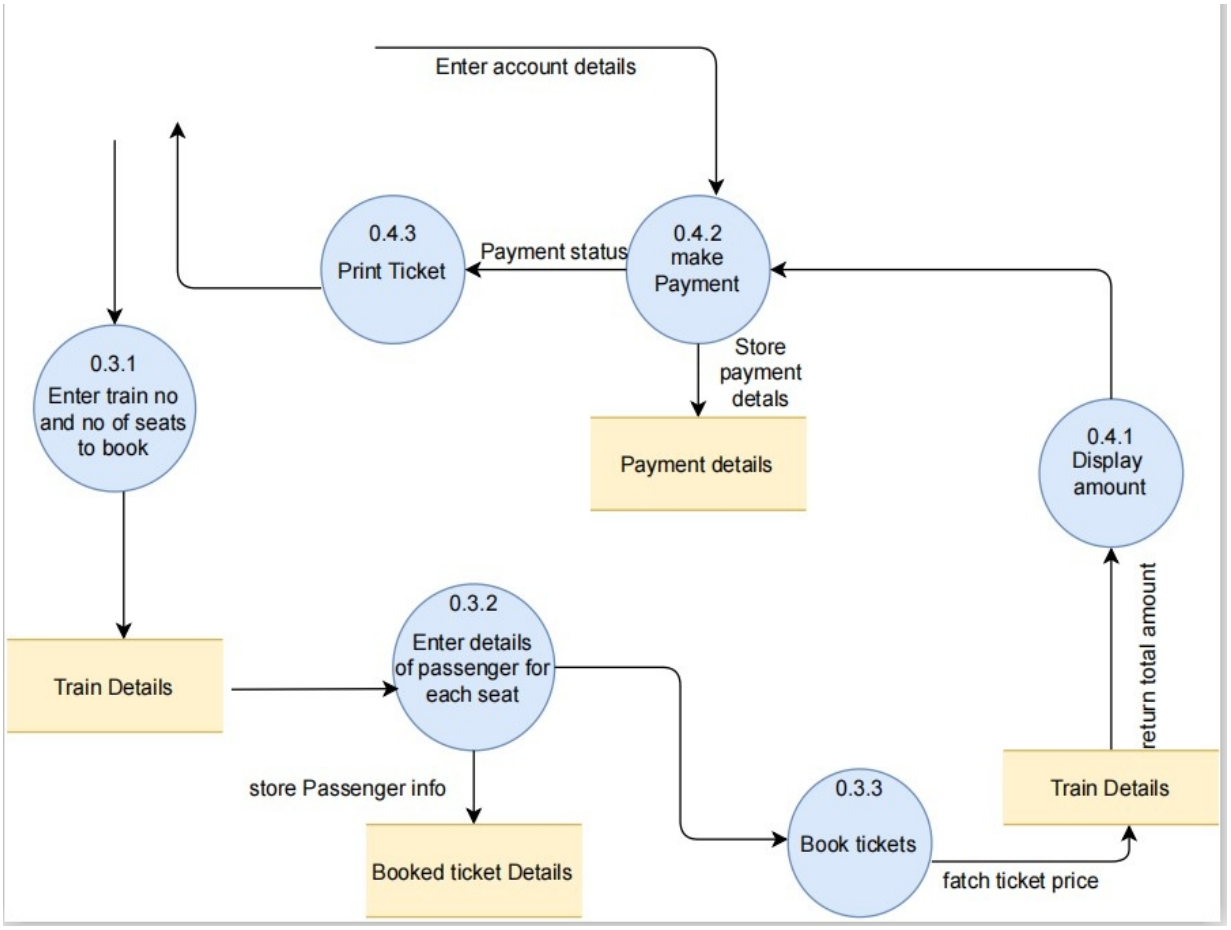
Login (Process 0.1)



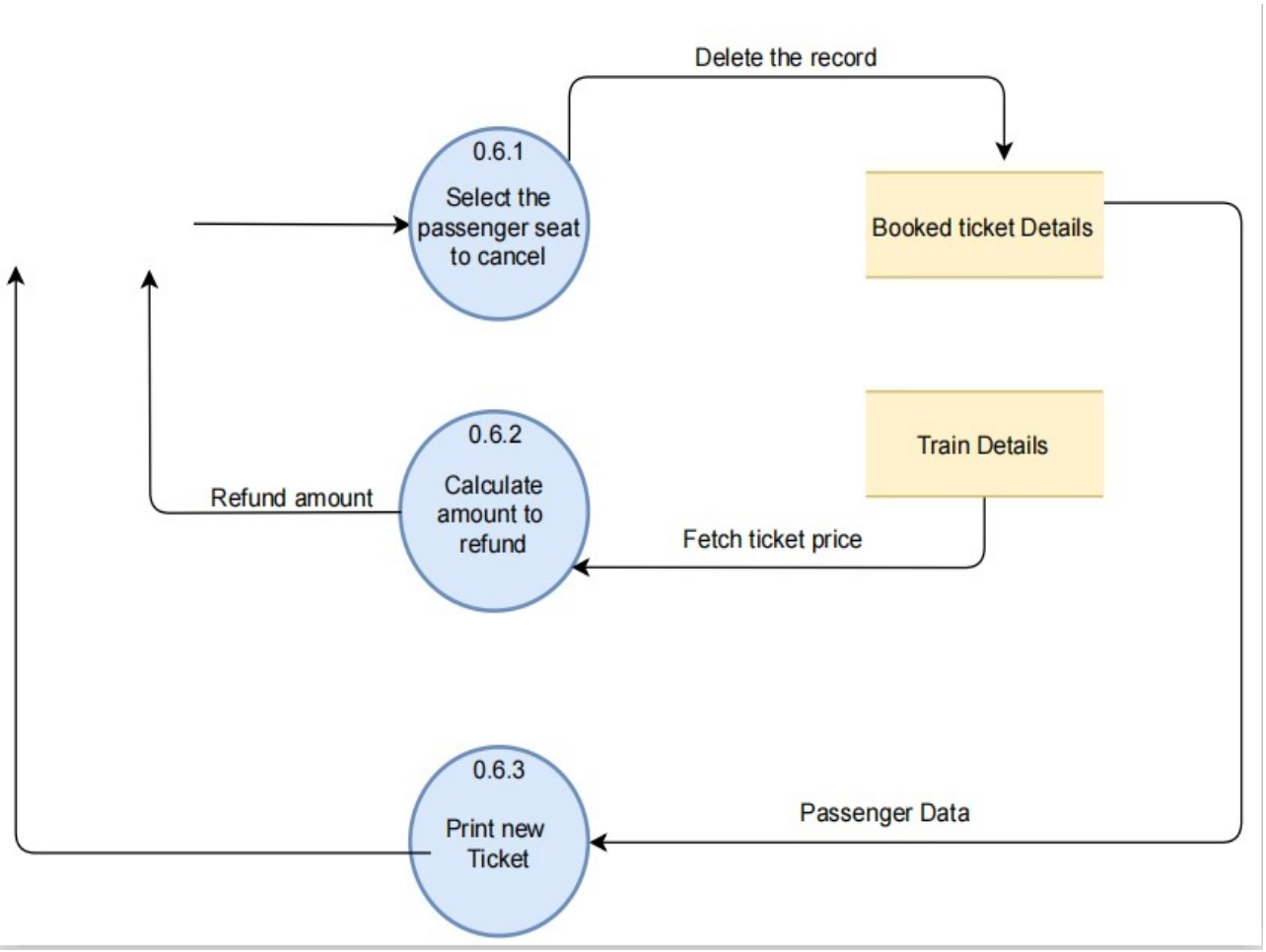
Train Inquiry(Process 0.2)



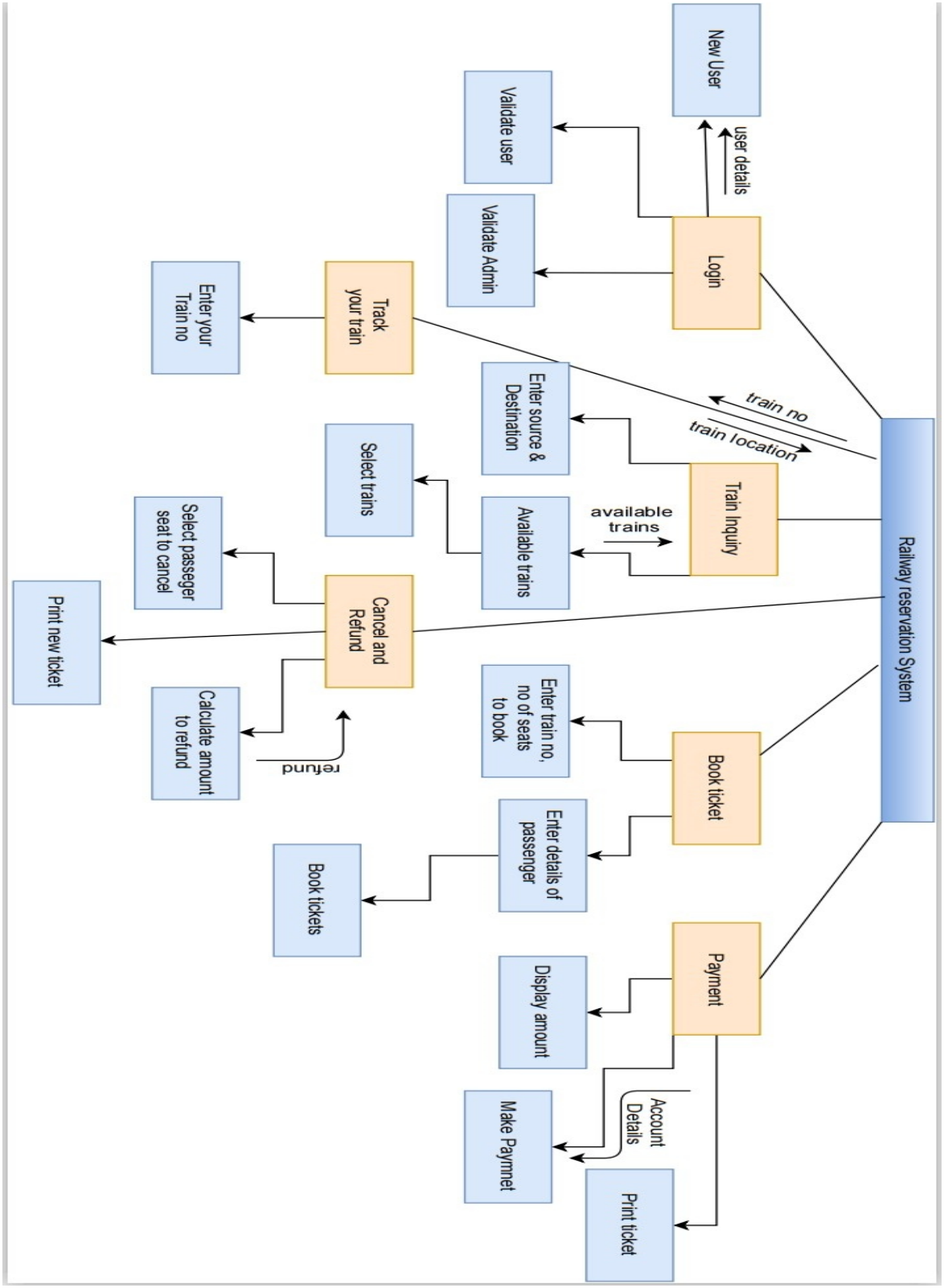
Reservation and Payment (Process 0.3 & 0.4)



Cancel and refund(Process 0.6)

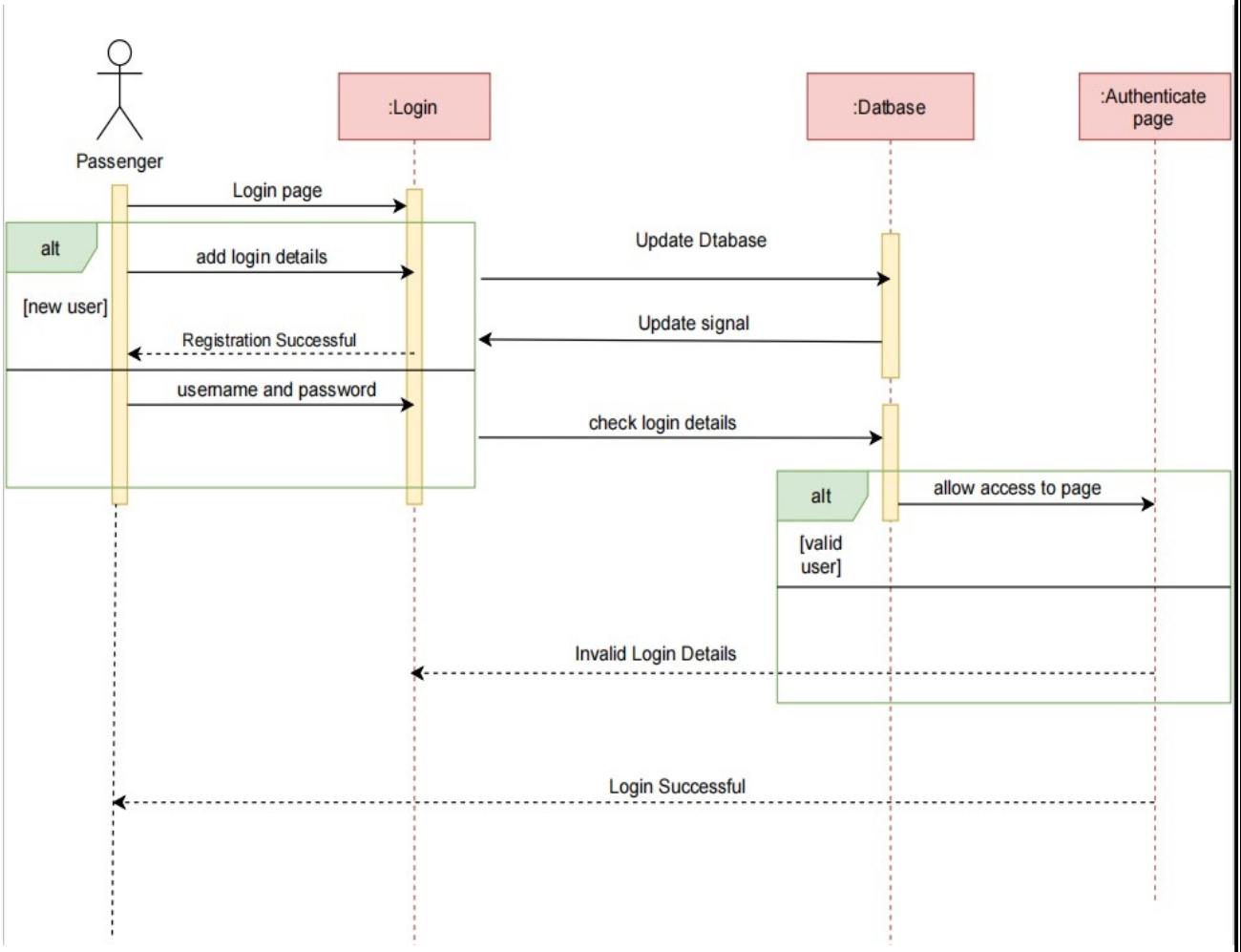


4. Structure chart

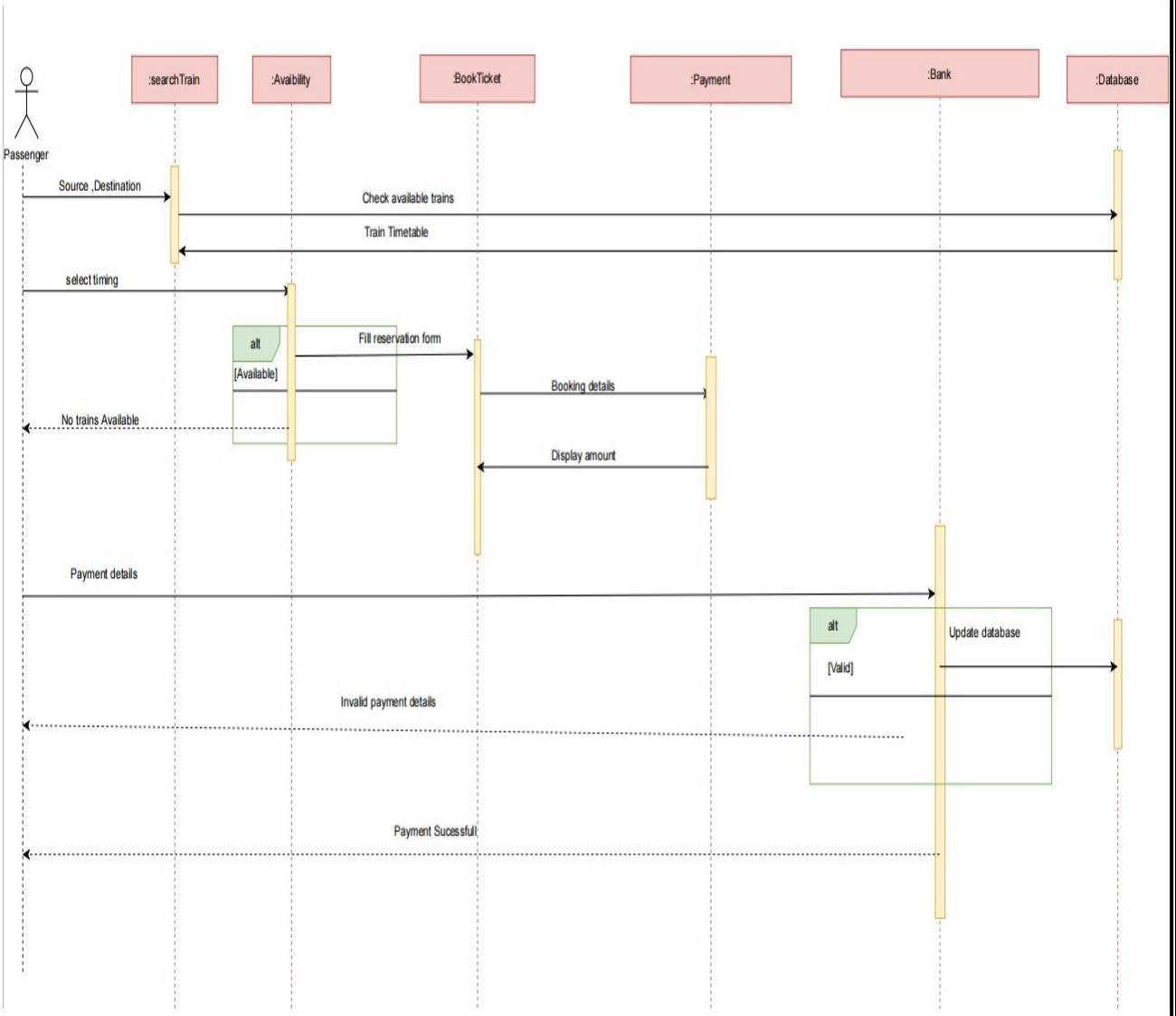


5. Sequence Diagram:

I. Login System

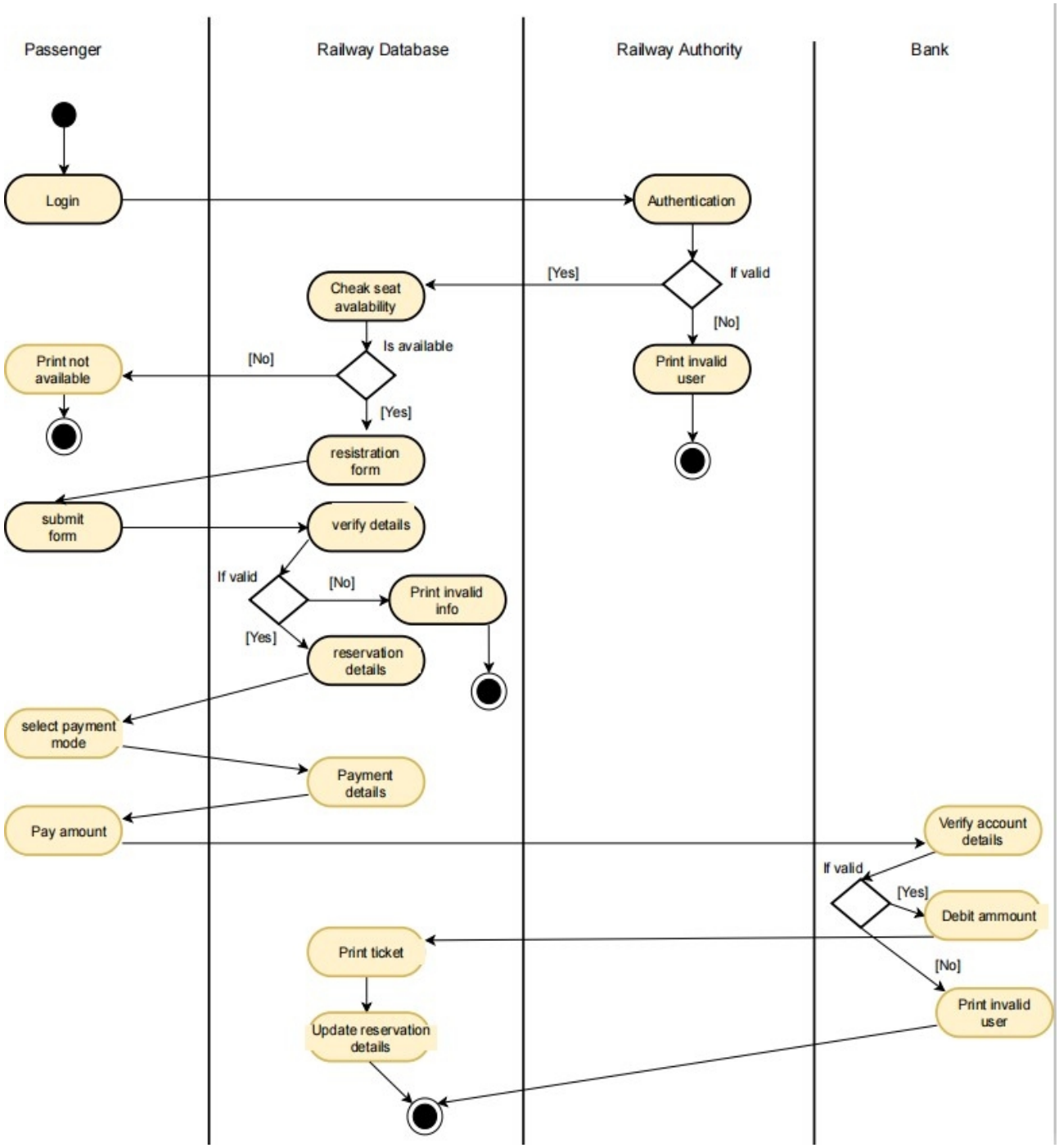


II. Reservation

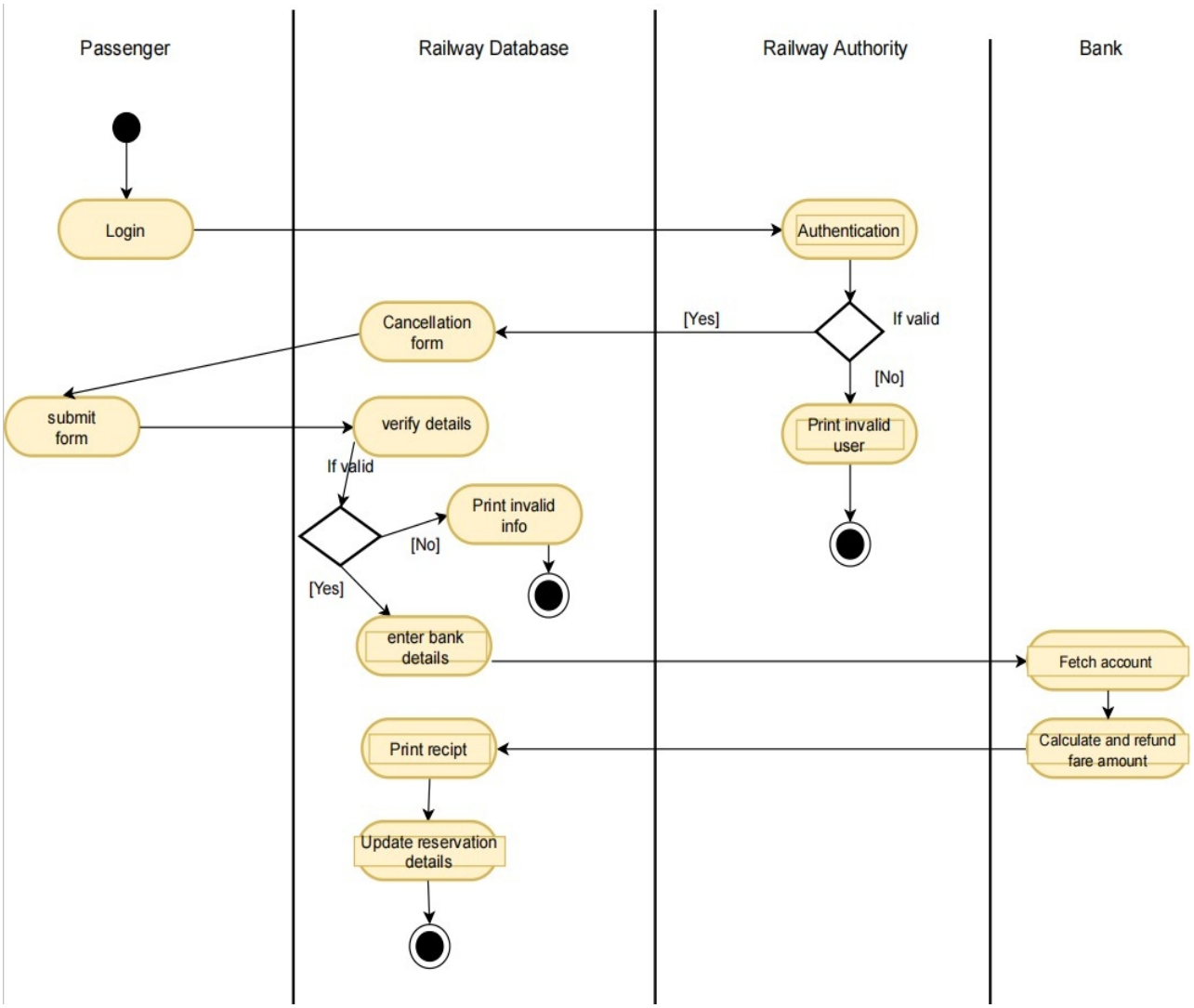


6. Activity Diagram:

I. Login & Reservation



II. Cancellation



Implementation Detail

1. Modules:

In the following section a brief description of each module is given. Related screenshots are attached in separate sections.

The system consists of 4 basic modules namely

I. Admin Module

II. User Module

III. Reservation Module

IV. Train details Module

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

I. Admin Module

Admin can modify time table of trains,add trains ,remove trains and can change timings of trains. Admin can view feedback from users. Admin can manage Users.

II. User Module

User can sign up to the system and then login.Users select source and and destination and book ticket. After users make payment they can view their ticket and they can also cancel the ticket. Users can give feedback based on their experience.

III. Reservation Module

In Reservation Module , according to the source and destination selected by users tickets will be generated, and then user needs to fill the form and make payment to book their ticket. After the payment is done users will able to see their ticket and will able to cancel the ticket.

IV. Train details Module

In this module information related to all the trains is visible to user and user can select train.

2. Major Functions prototypes

Login:

User needs to provide their username and password .If it matches to database then Users can login to the system.

```
def login(request):
      if request.method == 'POST':
       username = request.POST.get('username')
       password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('loginmodule:home')
    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)


    if user is not None:
        auth.login(request, user)
        return HttpResponseRedirect('/loginmodule/home/')
    else:
        return render(request,'invalidlogin.html')
```

Signup:

If user is new to the system he needs to fill sign up form.And details of form will be stored to database.

```
def signup(request):
    if request.method=="POST":
        fname=request.POST.get('fname')
        lname=request.POST.get('lname')
        email=request.POST.get('email')
        password=request.POST.get('pass1')
        conf_pass=request.POST.get('pass2')
        age=request.POST.get('bdate')
        contact=request.POST.get('contact')
        gender=request.POST.get('gender')
     try:
            user = User.objects.get(email=request.POST.get('username'))
            return render(request, 'signup.html', {'error': "Email Already Exists "}
)
        except User.DoesNotExist:
            user = User.objects.create_user(username=request.POST.get('username'),
password=request.POST.get('pass1'),first_name=request.POST.get('fname'),last_name=r
equest.POST.get('lname'), email=request.POST.get('email'))
            user.save()
            x=UserDetails(fname=fname,lname=lname,email=email,password=password,con
f_pass=conf_pass,age=age,contact=contact,gender=gender)
            x.save()
            auth.login(request, user)
            return render(request,'home.html')
    else:
        return render(request,'signup.html')
```

Feedback:

Users can give feedback that will be stored in database and admin of the system will able to see it.

```
@login_required(login_url='loginmodule:login')
def feedback(request):
    if request.method=="POST":
       email=request.POST.get('email')
       feedback=request.POST.get('feedback')
       x=Feedback(email=email,Feedback=feedback)
       x.save()
       return render(request,'loggedin.html')
    else:
        return render(request,'feedback.html')
```

Registration:

User needs to fill the form to reserve the ticket and the data of the form will be stored to database.

```
@login_required(login_url='loginmodule:login')
def registration(request):
    if request.method=="POST":
          gender=request.POST.get('gender')
          seats=request.POST.get('seats')
          age=request.POST.get('age')
          print(seats)
          x=Reservation(Age=age,noOfPassenger=seats,Gender=gender)
          x.save()
          return HttpResponseRedirect("/bookticket/payment/")
    else:
        return render(request,'registration.html')
```

Search:

Users can select train based on their choice in order to book ticket.

```
@login_required(login_url='loginmodule:login')
def search(request):
    if request.method == "POST":
        if request.POST.get("book"):
            arrival = request.POST.get('arrival')
            departure = request.POST.get('departure')
            trainno = request.POST.get('trainno')
            print(arrival)
            request.session['trainno']=trainno
            request.session['arrival']=arrival
            request.session['departure']=departure
            return HttpResponseRedirect("/bookticket/registration/")
        else:
            source = request.POST.get("source")
            destination=request.POST.get("dest")
            print(source)
            print(destination)
            request.session['source'] = source
            request.session['destination']=destination
            trains=Train.objects.filter(Source__icontains=source,Destination__icont
ains=destination)
            print(trains)
            return render(request,'search.html',{'trains':trains})
```

Source Destination:

In this function based on source and destination users have given the data from the database will be fetched and different trains will be shown to user.

```
@login_required(login_url='loginmodule:login')
def source_dest(request):
    if request.method=="POST":
        source = request.POST.get("source")
        destination=request.POST.get("dest")
        print(source)
        print(destination)
        request.session['source'] = source
        request.session['destination']=destination
        trains=Train.objects.filter(Source__icontains=source,Destination__icontains
=destination)

        print(trains)
        return render(request,'search.html',{'trains':trains})
    else:
        return render(request,'source_dest.html')
```

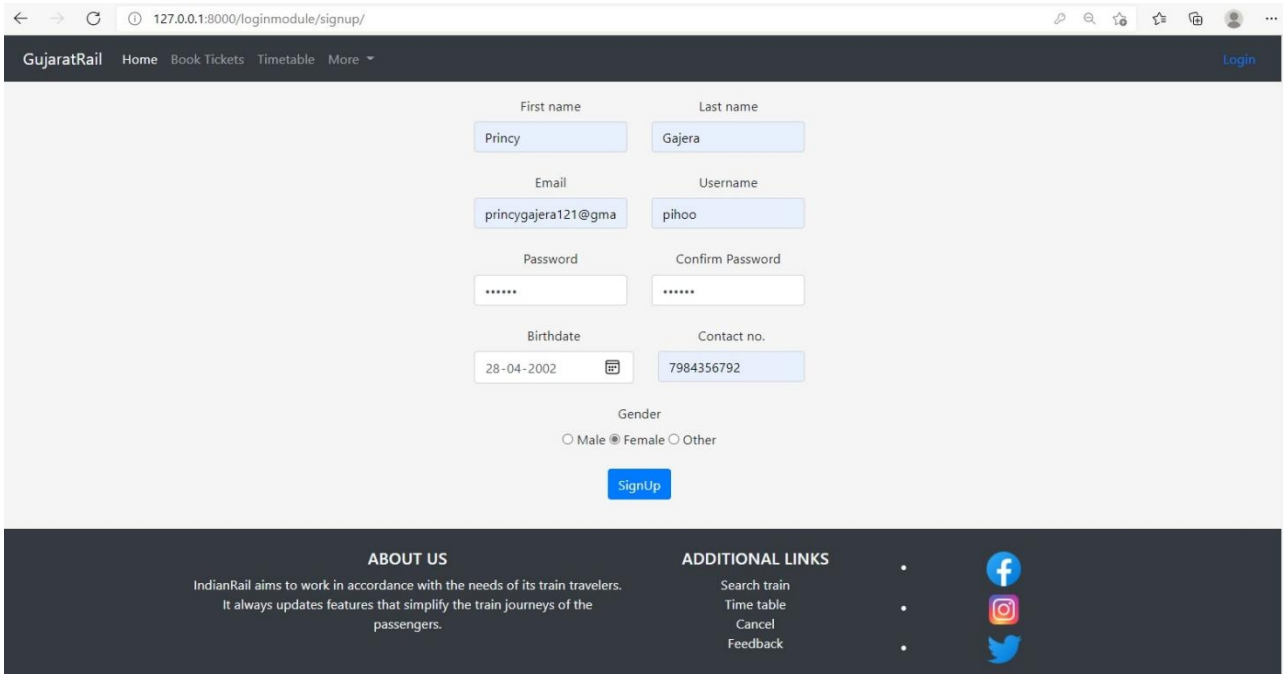
Ticket:

Tickets will automatically generated by the system based on which details Users have filled.

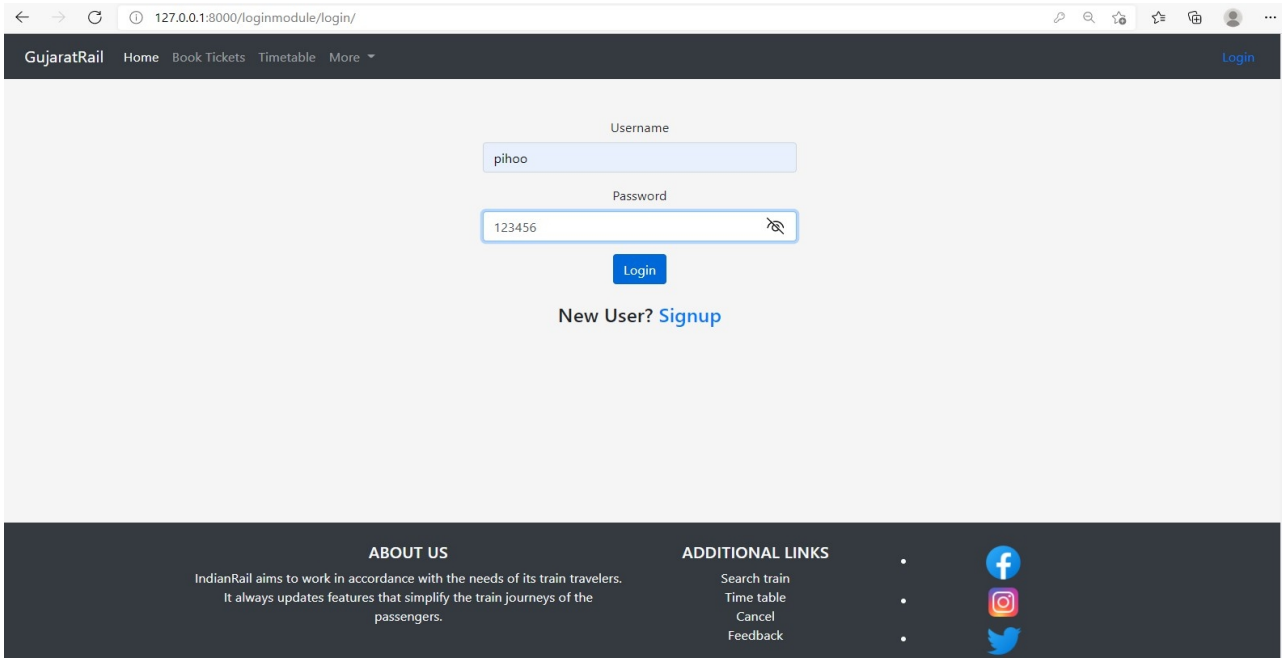
```
def ticket(request):
    #c=request.user
    #print(c)
    #print(c.first_name)
    #print(c.last_name)

    source=request.session['source']
    destination= request.session['destination']
    trainno=request.session['trainno']
    arrival=request.session['arrival']
    departure=request.session['departure']
    print(arrival)
    t=Tickets(Source=source,Destination=destination,Trainno=trainno,arrivaltime=arr
ival,departuretime=departure)

    t.save()
    return render(request,'ticket.html',{'t':t})
```

Layout

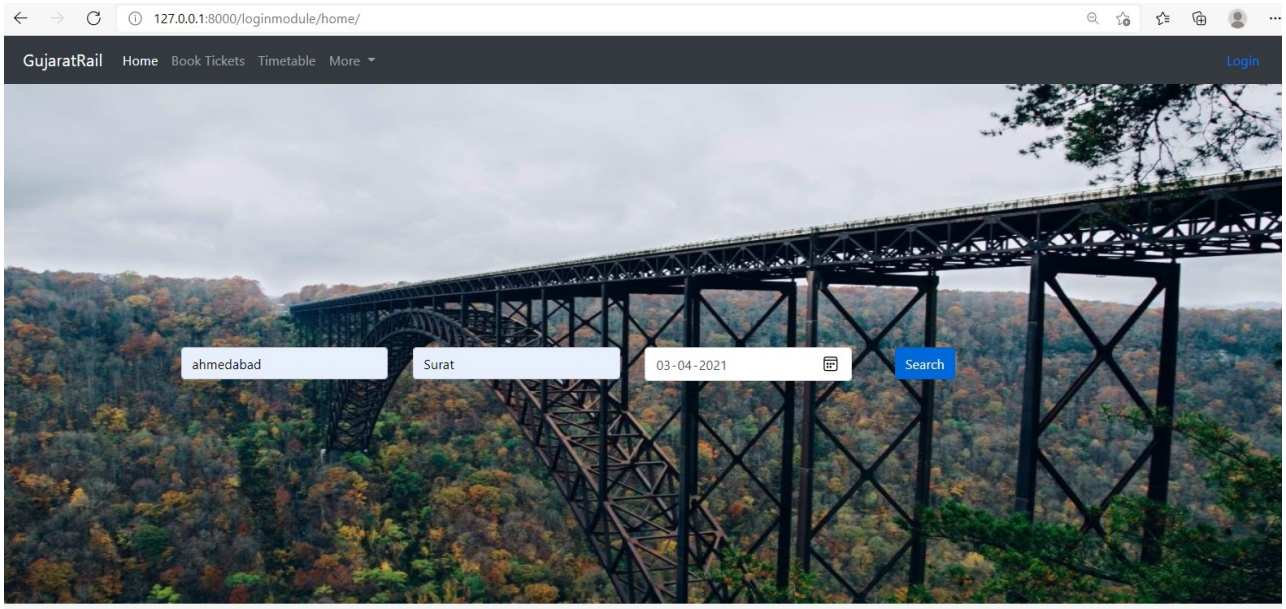
Signup Page



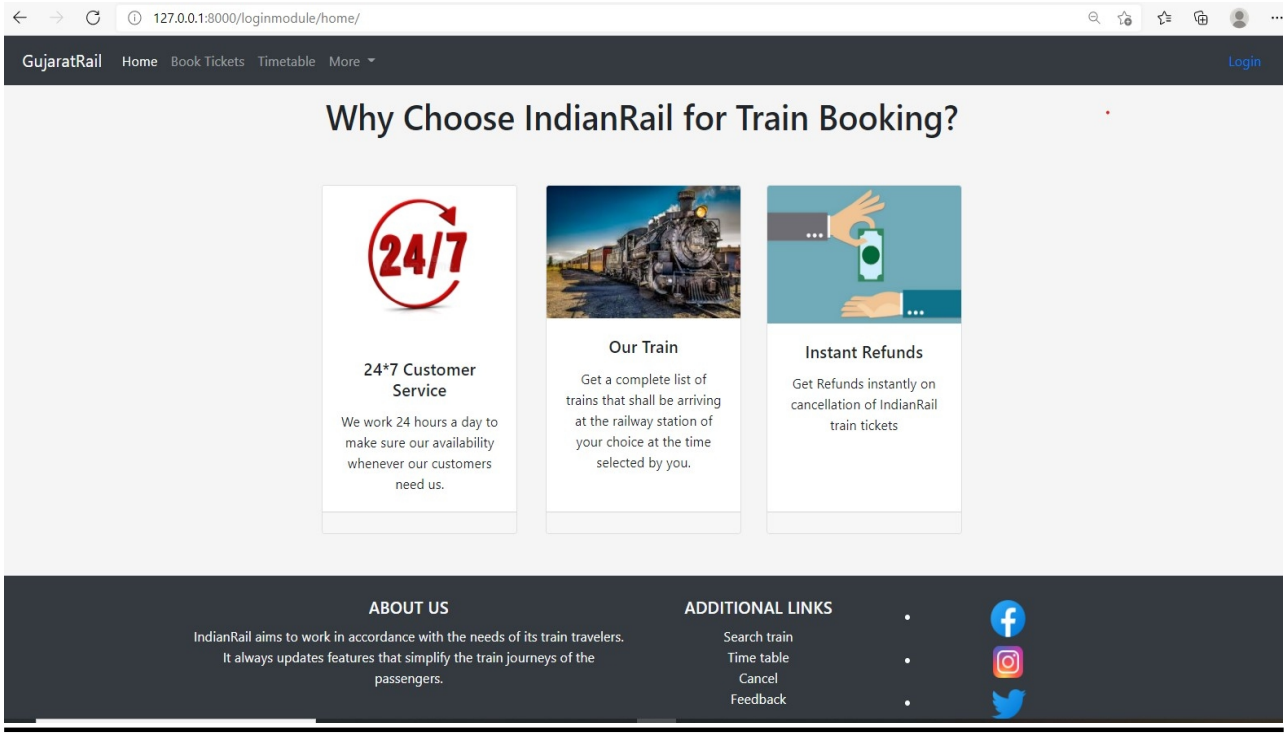
Login Page



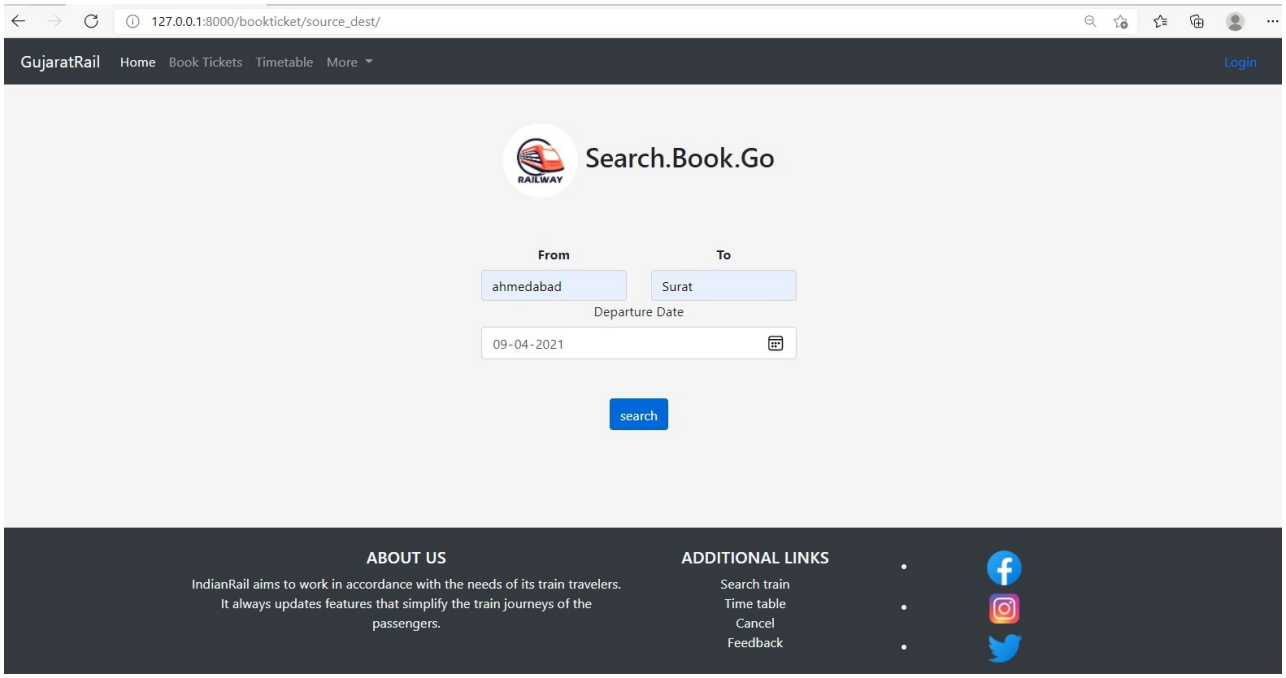
Home Page



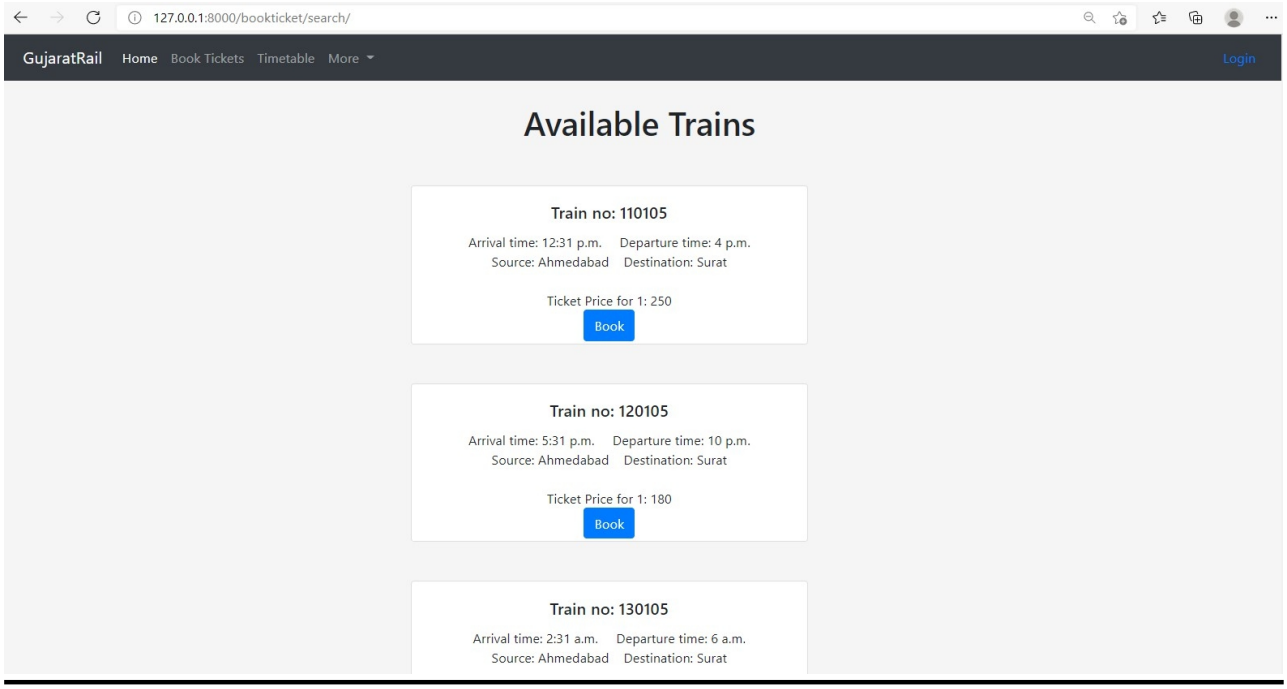
Services



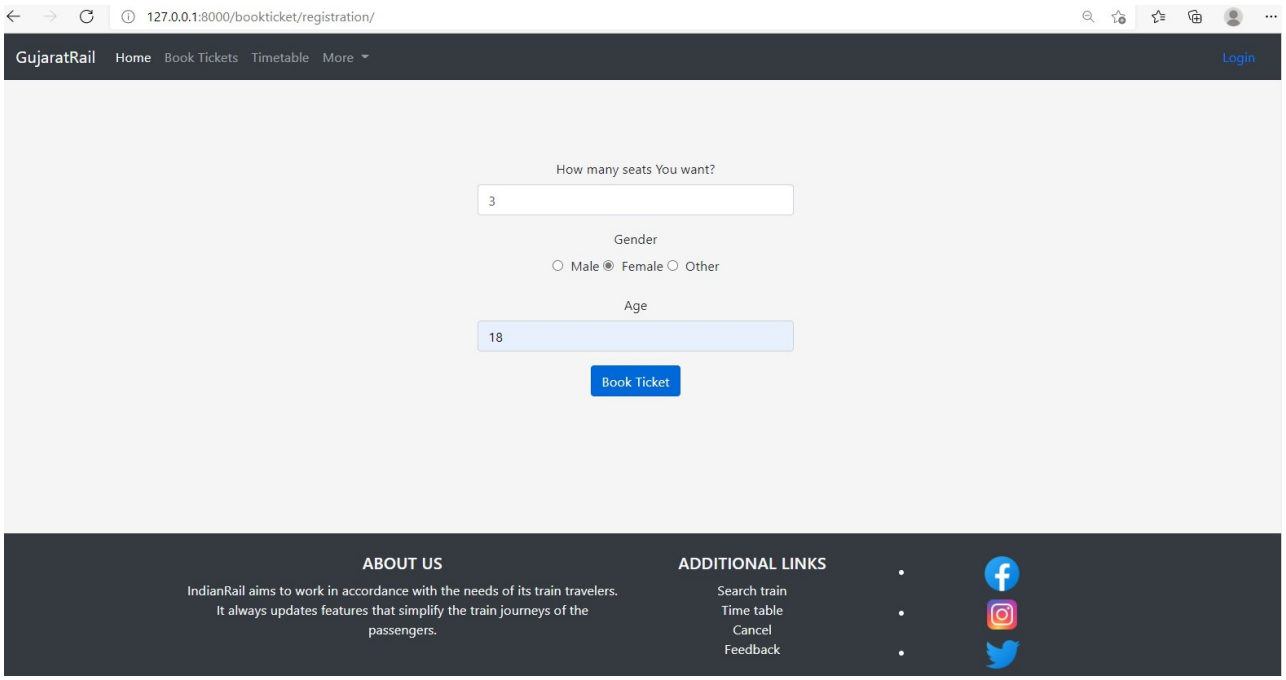
Search Train



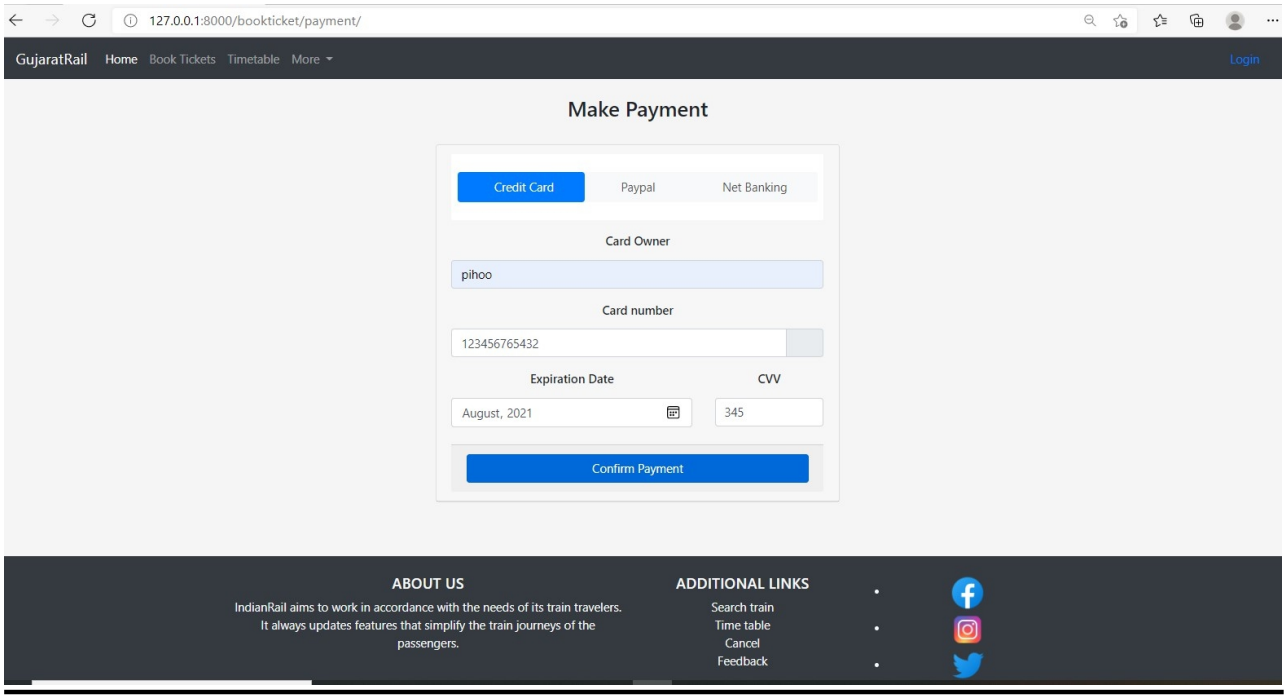
Available Trains



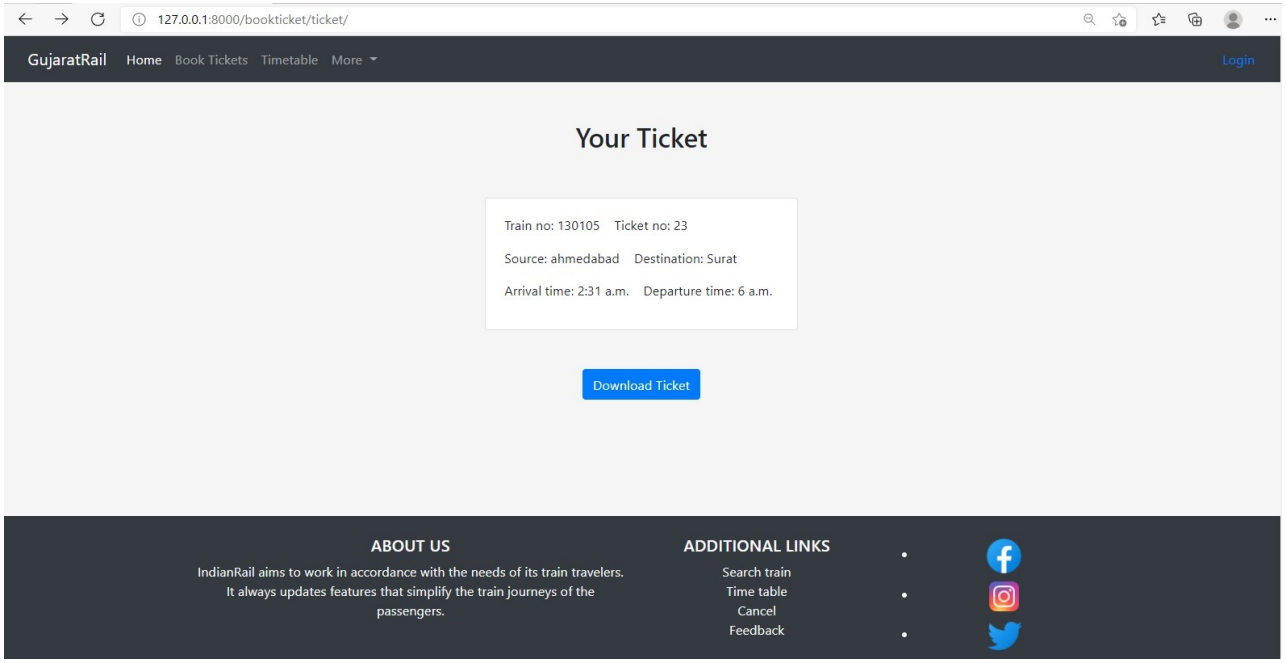
Book Ticket



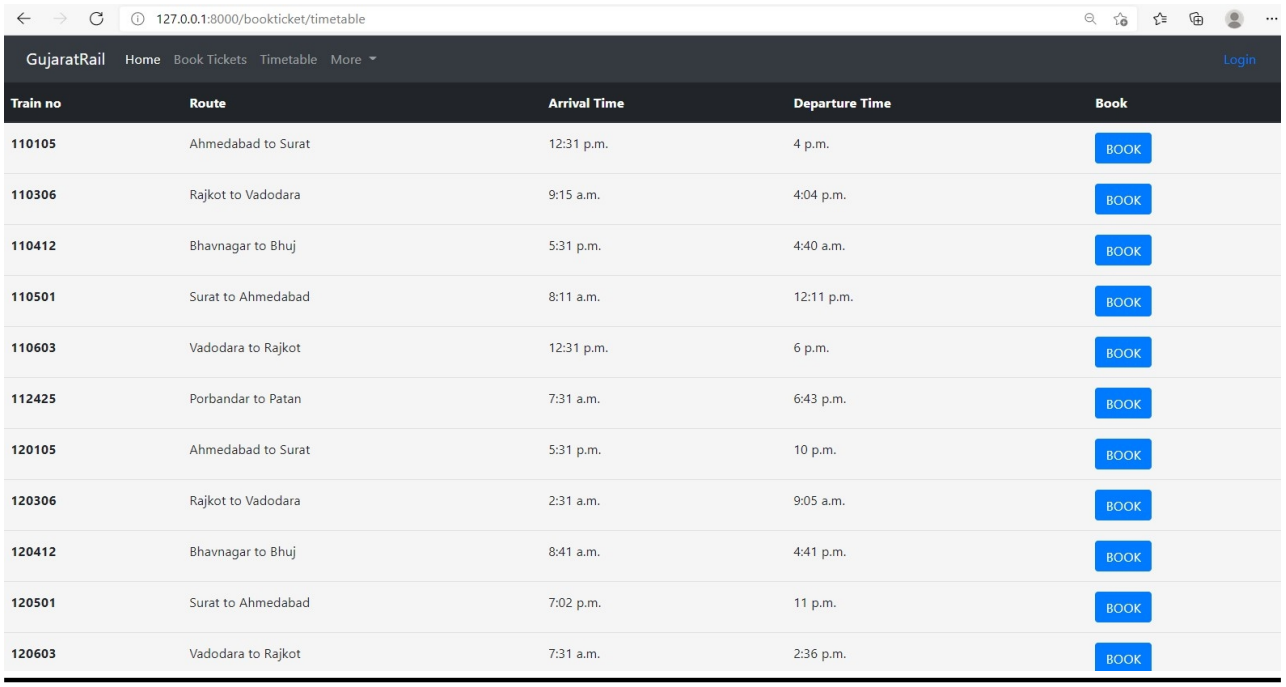
Payment



Display Ticket



Time Table



Train no	Route	Arrival Time	Departure Time	Book
110105	Ahmedabad to Surat	12:31 p.m.	4 p.m.	BOOK
110306	Rajkot to Vadodara	9:15 a.m.	4:04 p.m.	BOOK
110412	Bhavnagar to Bhuj	5:31 p.m.	4:40 a.m.	BOOK
110501	Surat to Ahmedabad	8:11 a.m.	12:11 p.m.	BOOK
110603	Vadodara to Rajkot	12:31 p.m.	6 p.m.	BOOK
112425	Porbandar to Patan	7:31 a.m.	6:43 p.m.	BOOK
120105	Ahmedabad to Surat	5:31 p.m.	10 p.m.	BOOK
120306	Rajkot to Vadodara	2:31 a.m.	9:05 a.m.	BOOK
120412	Bhavnagar to Bhuj	8:41 a.m.	4:41 p.m.	BOOK
120501	Surat to Ahmedabad	7:02 p.m.	11 p.m.	BOOK
120603	Vadodara to Rajkot	7:31 a.m.	2:36 p.m.	BOOK

Conclusion

Hence-forth in this project we have successfully implemented the Admin-side & Passenger-side functionality, Admin will add the trains & Passenger can login to the System and can select source and destination.According to that system will display trains to Passenger and passenger can book tickets. Passenger can give their ticket no to cancel the ticket.

Limitations

I .Users can not track at which location the train is at current time.

II.Users can’t modify their details once they signup.

III. If users forget password they can’t recover it.

Future Extension

To take over the limitations we are planning this future extension in our system.

I. let users to modify their details.

II. add forgot password functionality to the system.

III. add functionality to track the trains by their train no.

Bibliography

References/resources used for developing project:

I.docs.djangoproject.com

II.https://www.railyatri.in