Final Code

Team ID	PNT2022TM30650
Project Name	Smart Solutions for Railways

Source Code:

```
from flask import Flask, render_template, flash, request,session

from flask import Flask, render_template, request, jsonify
import datetime
import re

import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine('sqlite://',
                       echo = False)

dsn_hostname = "b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "bjb73167"
dsn_pwd = "uHziEzMelmTeO1Nn"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "32716"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
```

```
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd,dsn_security)
try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ", dsn_hostname)

except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )


app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'



@app.route("/")
def homepage():

    return render_template('index.html')


@app.route("/AdminLogin")
def AdminLogin():

    return render_template('AdminLogin.html')

@app.route("/NewUser")
```

```
def NewUser():


    return render_template('NewUser.html')
@app.route("/UserLogin")
def UserLogin():

    return render_template('UserLogin.html')

@app.route("/viewproduct", methods=['GET', 'POST'])
def viewproduct():

    t1 = request.form['t1']
    t2 = request.form['t2']



    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery =  "SELECT * from protb where Source like '%" + t1 + "%' and Destination like'%"+ t2 +"%' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()

    return render_template('ViewProduct.html', data=data)

@app.route("/AdminHome")
def AdminHome():



    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()

    return render_template('AdminHome.html', data=data)


@app.route("/NewProduct")
def NewProduct():
    return render_template('NewProduct.html')

@app.route("/Search")
def Search():

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM traintb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()


    return render_template('ViewProduct.html',data=data)



@app.route("/ProductInfo")
def ProductInfo():

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM traintb "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()

    return render_template('ProductInfo.html',data=data)

@app.route("/SalesInfo")
def SalesInfo():

    return render_template('SalesInfo.html')



@app.route("/FeedBackInfo")
def FeedBackInfo():

    return render_template('FeedBackInfo.html')

@app.route("/RNewUser", methods=['GET', 'POST'])
def RNewUser():
    if request.method == 'POST':

        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        address = request.form['address']
        pnumber = request.form['phone']
        uname = request.form['uname']
        password = request.form['psw']

        conn = ibm_db.connect(dsn, "", "")
```

```
        insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        # return 'file register successfully'

    return render_template('userlogin.html')

@app.route("/RNewProduct", methods=['GET', 'POST'])
def RNewProduct():
    if request.method == 'POST':



        file = request.files['file']
        file.save("static/upload/" + file.filename)



        t1 =request.form['t1']
        t2 = request.form['t2']
        t3 =request.form['t3']

        s1 =request.form['s1']
        s2 = request.form['s2']

        t4 = request.form['t4']
        t5 = request.form['t5']
        t6 = request.form['t6']



        conn = ibm_db.connect(dsn, "", "")

        insertQuery = "INSERT INTO traintb VALUES ('"+ t1 +"','" + t2 + "','" + t3 + "','" + s1
+ "','" + s2 + "','"+t4 +"','" + t5 +"','"+ t6+ "','"+ file.filename +"')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        # return 'file register successfully'
```

```
    return render_template('NewProduct.html')

@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():
    error = None
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']

        conn = ibm_db.connect(dsn, "", "")

        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * from regtb where UserName='" + username + "' and password='" + password + "'"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        if dataframe.empty:
            data1 = 'Username or Password is wrong'
            return render_template('goback.html', data=data1)
        else:
            print("Login")
            selectQuery = "SELECT * from regtb where UserName='" + username + "' and password='" + password + "'"
            dataframe = pandas.read_sql(selectQuery, pd_conn)

            dataframe.to_sql('Employee_Data',
                             con=engine,
                             if_exists='append')

            # run a sql query
            print(engine.execute("SELECT * FROM Employee_Data").fetchall())
```

```
        return render_template('UserHome.html', data=engine.execute("SELECT * FROM Employe_Data").fetchall())


@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']

        if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

            conn = ibm_db.connect(dsn, "", "")
            pd_conn = ibm_db_dbi.Connection(conn)
            selectQuery = "SELECT * FROM regtb  "
            dataframe = pandas.read_sql(selectQuery, pd_conn)
            dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
            data = engine.execute("SELECT * FROM Employee_Data").fetchall()

            return render_template('AdminHome.html', data=data)

        else:
            return render_template('index.html', error=error)

@app.route("/Remove", methods=['GET'])
def Remove():

    pid = request.args.get('id')
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    insertQuery = "Delete from traintb  where Source='" + pid + "'"
```

```
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

    selectQuery ="SELECT * FROM traintb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                     con=engine,
                     if_exists='append')

    # run a sql query
    print(engine.execute("SELECT * FROM Employee_Data").fetchall())

    return render_template('ProductInfo.html', data=engine.execute("SELECT * FROM
Employee_Data").fetchall())
    # return 'file register successfully'



@app.route("/Remove1", methods=['GET'])
def Remove1():



    pid = request.args.get('id')

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    insertQuery = "Delete from booktb  where TicketId='" + pid + "'"
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

    selectQuery = "SELECT * FROM booktb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
```

```
                con=engine,
                if_exists='append')

    # run a sql query
    print(engine.execute("SELECT * FROM Employee_Data").fetchall())
    return render_template('ProductInfo.html', data=engine.execute("SELECT * FROM
Employee_Data").fetchall())

    return render_template('UOrderInfo.html',data=data)

@app.route("/fullInfo")
def fullInfo():
    pid = request.args.get('pid')
    session['pid'] = pid

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM traintb where TrainNo='" + pid + "'  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data1 = engine.execute("SELECT * FROM Employee_Data").fetchall()


    return render_template('ProductFullInfo.html',data=data1 )

@app.route("/Book", methods=['GET', 'POST'])
def Book():
    if request.method == 'POST':


        uname = session['uname']
        pid = session['pid']

        qty = request.form['qty']
```

```
Bookingid = ''
ProductName =''
UserName= uname

Mobile=''
Email=''
Qty = qty
Amount=''

CardType =''
CardNo =' '
CvNo = ''
date = datetime.datetime.now().strftime('%d-%b-%Y')

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM traintb where TrainNo='" + pid + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
for item in data:
    source = item[1]
    des = item[2]

    price = item[6]
    print(price)
    Amount = float(price) * float(Qty)

    print(Amount)

selectQuery1 = "SELECT  *  FROM  regtb where  UserName='" + uname + "'"
dataframe = pandas.read_sql(selectQuery1, pd_conn)
```

```
dataframe.to_sql('regtb', con=engine, if_exists='append')
data1 = engine.execute("SELECT * FROM regtb").fetchall()

for item1 in data1:
    Mobile = item1[5]
    Email = item1[4]



selectQuery = "SELECT  *  FROM  booktb"
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('booktb', con=engine, if_exists='append')
data2 = engine.execute("SELECT * FROM booktb").fetchall()
count = 1

for item in data2:
    count += 1

Bookingid="BOOKID00" + str(count)
session['bid']=Bookingid

import qrcode

img = qrcode.make(Bookingid)

import random

pn = random.randint(1111, 9999)

img.save("static/Qrcode/" + str(pn) + ".png")

Qrcode = str(pn) + ".png"
```

```
        session["qr"]=Qrcode

        insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "','"+ source +"','" + des + "','" + uname + "','" + Mobile + "','" + Email + "','" + str(Qty) + "','" + str(Amount) + "','"+ str(CardType) +"','"+ str(CardNo) +"','"+ str(CvNo) +"','"+ str(date) +"','1','"+ Qrcode +"')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)

        sendmsg(Email, "Ticket Booked ")
        # return 'file register successfully'

    return render_template('Payment.html', Amount=Amount)

@app.route("/UOrderInfo")
def UOrderInfo():

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb where  UserName= '" + uname + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()

    return render_template('UOrderInfo.html', data=data)

@app.route("/UserHome")
def UserHome():
    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where  UserName= '" + uname + "' "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()


    return render_template('UserHome.html', data=data)


@app.route("/pay", methods=['GET', 'POST'])
def pay():
    if request.method == 'POST':

        data = session['bid']
        qr = session["qr"]


        ctype = request.form['ctype']
        cardno = request.form['cardno']
        cvno = request.form['cvno']

        conn = ibm_db.connect(dsn, "", "")

        insertQuery = "update booktb set CardType='"+ ctype  +"', CardNo='"+ cardno +"', Cvno='"+ cvno +"' where  TicketId = '" + str(data) + "'  "
        insert_table = ibm_db.exec_immediate(conn, insertQuery)


        return render_template('Payment.html', data=qr)


@app.route("/check", methods=['GET', 'POST'])
def check():
    if request.method == 'POST':
```

```
file = request.files['file']
file.save("static/upload/" + file.filename)

import cv2

detector = cv2.QRCodeDetector()

img = cv2.imread("static/upload/" + file.filename)
qdata, bbox, _ = detector.detectAndDecode(img)

print(qdata)

conn = ibm_db.connect(dsn, "", "")

pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT  *  FROM  booktb where  TicketId = '" + str(qdata) + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)

if dataframe.empty:
    return render_template('ASalesInfo.html', res='No Record Found!')
else:
    print("Login")
    selectQuery = "SELECT * FROM booktb where TicketId = '" + str(qdata) + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                     con=engine,
                     if_exists='append')

    # run a sql query
    print(engine.execute("SELECT * FROM Employee_Data").fetchall())
```

```
        return render_template('ASalesInfo.html', data=engine.execute("SELECT * FROM Employee_Data").fetchall())

@app.route("/ASalesInfo")
def ASalesInfo():

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()



    return render_template('ASalesInfo.html', data=data)

def sendmsg(Mailid,message):
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders

    fromaddr = "sampletest685@gmail.com"
    toaddr = Mailid

    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr
```

```
    # storing the subject
    msg['Subject'] = "Alert"

    # string to store the body of the mail
    body = message

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))

    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)

    # start TLS for security
    s.starttls()

    # Authentication
    s.login(fromaddr, "hneucvnontsuwgpj")

    # Converts the Multipart msg into a string
    text = msg.as_string()

    # sending the mail
    s.sendmail(fromaddr, toaddr, text)

    # terminating the session
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug='TRUE')


CREATE TABLE booktb (

  TicketId varchar(250) NOT NULL,
  Source varchar(250) NOT NULL,
```

```
  Destination varchar(250) NOT NULL,
  UserName varchar(250) NOT NULL,
  Mobile varchar(250) NOT NULL,
  Email varchar(250) NOT NULL,
  Qty varchar(250) NOT NULL,
  Amount decimal(18,0) NOT NULL,
  CardType varchar(250) NOT NULL,
  CardNo varchar(250) NOT NULL,
  Cvno varchar(250) NOT NULL,
  Date varchar(250) NOT NULL,
  Status varchar(250) NOT NULL,
   Qrcode varchar(250) NOT NULL

)

CREATE TABLE regtb (
  Name varchar(250) NOT NULL,
  Gender varchar(250) NOT NULL,
  Age varchar(250) NOT NULL,
  Email varchar(250) NOT NULL,
  Mobile varchar(250) NOT NULL,
  Address varchar(250) NOT NULL,
  UserName varchar(250) NOT NULL,
  Password varchar(250) NOT NULL
)

CREATE TABLE traintb (

  Source varchar(250) NOT NULL,
  Destination varchar(250) NOT NULL,
  TrainNo varchar(250) NOT NULL,
  Category varchar(250) NOT NULL,
 Coach varchar(250) NOT NULL,
  Amount varchar(250) NOT NULL,
```

```
  Date varchar(250) NOT NULL,
  Info varchar(250) NOT NULL,
  Image varchar(250) NOT NULL

)
```