Report

TELECOMMUNICATION DATABASE MANAGEMENT SYSTEM

PREPARED BY :

Devam Patel
22BCE066
B-Tech (CSE)

Introduction

The Telecommunication Management System is designed to facilitate the management of customer accounts, plans, recharges, and various other functionalities related to telecommunication services. This report provides an overview of the system's functionalities and the structure of the codebase.

Functionality Overview

The system provides the following functionalities:

1. User Login: Allows users to log in to their accounts.

2. Admin Login: Allows administrators to log in to perform administrative tasks.

3. User Functionality:

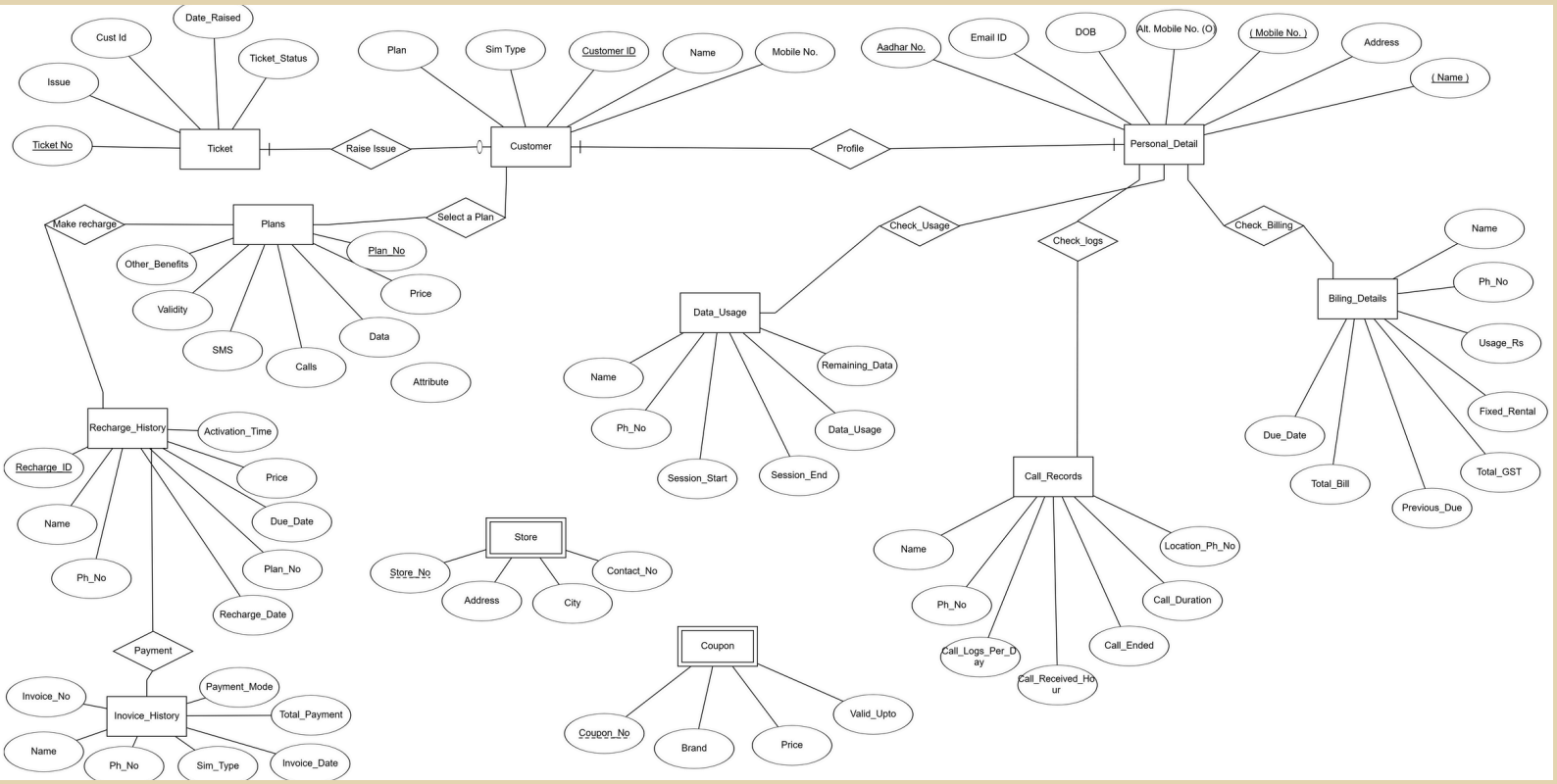
- View Plan Details
- Recharge Mobile Number
- Get Details of All Prepaid Plans Available
- View Invoice History
- View Recharge History
- Check Data Usage

- Check Call Logs
- View Coupons
- Update Profile
- View Profile
- Get Store Information Near Address
- View Raised Tickets

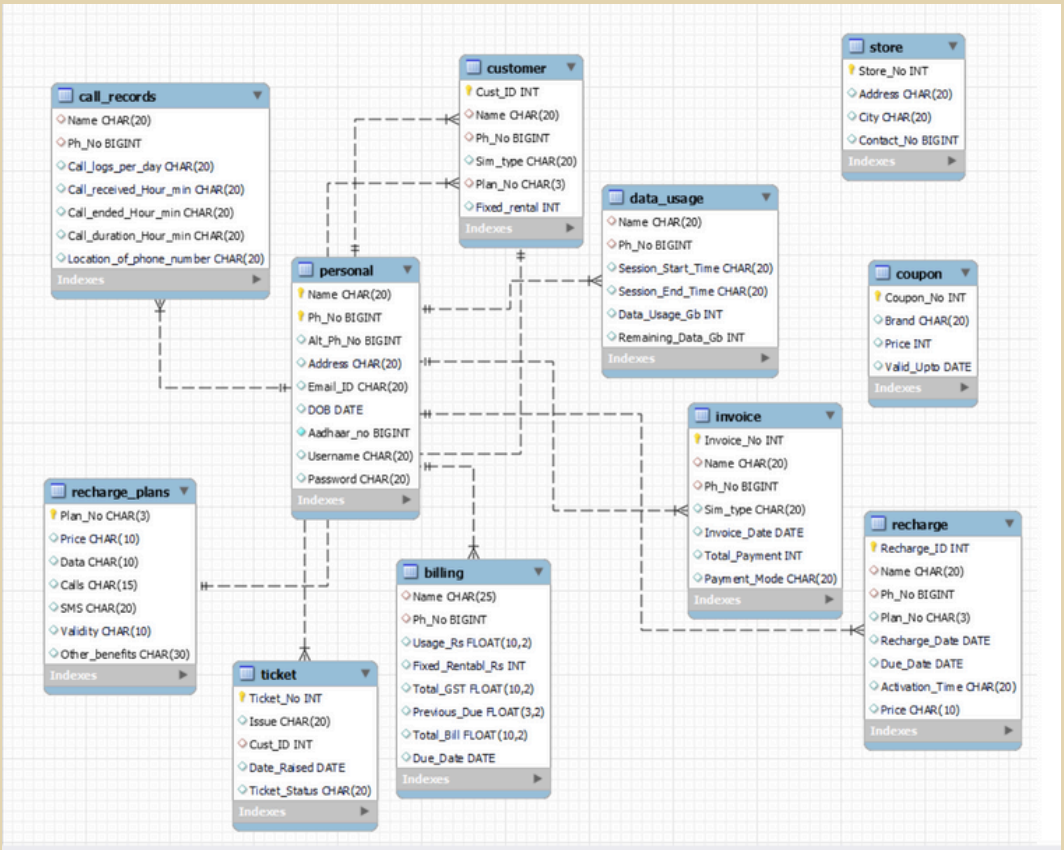
4. Admin Functionality:

- Add New Customer
- Delete Customer
- Check Call Records
- View Raised Tickets

Entity Relationship Diagram



Relational Model



Normalization

Ticket

Attributes: Ticket_Status,Date_Raised,Cust_Id,Issue,Ticket_No
Functional Dependency : Ticket_No --> Ticket_Status,Date_Raised,Cust_Id,Issue

2NF :

The candiates keys are { Ticket_No}, The set of key attributes are: { Ticket_No }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

The candiates keys are { Ticket_No}, The set of key attributes are: { Ticket_No }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Call Records

Attributes: Ph_No, Call_type_per_day,Name, Call_received_Hour_min, Call_ended_Hour_min, Call_duration_Hour_min, Location_of_phone_number

Functional Dependency : Ph_No, Call_type_per_day -> Name, Call_received_Hour_min, Call_ended_Hour_min, Call_duration_Hour_min, Location_of_phone_number

2NF :

The candiates keys are { Ph_No,Call_type_per_day}, The set of key attributes are: { Ph_No,Call_type_per_day }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF :

The candiates keys are { Ph_No,Call_type_per_day}, The set of key attributes are: { Ph_No,Call_type_per_day }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Personal

Attributes: Name,Address,Ph_No.,Alt. Mobile No.,DOB,Email_ID,Aadhar No.
Functional Dependency : Ph_No, Alt_Ph_No -> Name, Address, Email_ID, DOB, Aadhar_no, Username, Password

2NF :

The candiates keys are { Ph_No.,Alt.MobileNo.}, The set of key attributes are: { Ph_No.,Alt.MobileNo. }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

The candiates keys are { Ph_No.,Alt.MobileNo.}, The set of key attributes are: { Ph_No.,Alt.MobileNo. }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Customer

Attributes: Cust_ID,Name, Ph_No, Sim_type, Plan_No, Fixed_rental
Functional Dependency : Cust_ID -> Name, Ph_No, Sim_type, Plan_No, Fixed_rental

2NF :

The candiates keys are { Cust_ID}, The set of key attributes are: { Cust_ID }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

The candiates keys are { Cust_ID}, The set of key attributes are: { Cust_ID }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Store

Attributes: Store_No,Address, City, Contact_No
Functional Dependency : Store_No -> Address, City, Contact_No

2NF :

The candiates keys are { Store_No}, The set of key attributes are: { Store_No }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.
3NF:

The candiates keys are { Store_No}, The set of key attributes are: { Store_No }for each FD, check whether the LHS is superkey or the RHS are all key attributes.
BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Data Usage

Attributes: Ph_No, Session_Start_Time, Session_End_Time,Name, Data_Usage_Gb, Remaining_Data_Gb
Functional Dependency : Ph_No, Session_Start_Time, Session_End_Time -> Name, Data_Usage_Gb, Remaining_Data_Gb

2NF :
The candiates keys are { Ph_No,Session_Start_Time,Session_End_Time}, The set of key attributes are: { Ph_No,Session_Start_Time,Session_End_Time }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.
3NF:
The candiates keys are { Ph_No,Session_Start_Time,Session_End_Time}, The set of key attributes are: { Ph_No,Session_Start_Time,Session_End_Time }for each FD, check whether the LHS is superkey or the RHS are all key attributes.
BCNF:
A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Coupon

Attributes: Coupon_No,Brand, Price, Valid_Upto
Functional Dependency : Coupon_No -> Brand, Price, Valid_Upto

2NF :
The candiates keys are { Coupon_No}, The set of key attributes are: { Coupon_No }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:
The candiates keys are { Coupon_No}, The set of key attributes are: { Coupon_No }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:
A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Invoice

Attributes: Invoice_No,Name, Ph_No, Sim_type, Invoice_Date, Total_Payment, Payment_Mode

Functional Dependency : Invoice_No -> Name, Ph_No, Sim_type, Invoice_Date, Total_Payment, Payment_Mode

2NF :

The candiates keys are { Invoice_No}, The set of key attributes are: { Invoice_No }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

The candiates keys are { Invoice_No}, The set of key attributes are: { Invoice_No }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Recharge

Attributes: Recharge_ID,Name, Ph_No, Plan_No, Recharge_Date, Due_Date, Activation_Time, Price

Functional Dependency : Recharge_ID -> Name, Ph_No, Plan_No, Recharge_Date, Due_Date, Activation_Time, Price

2NF :

The candiates keys are { Recharge_ID}, The set of key attributes are: { Recharge_ID }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

The candiates keys are { Recharge_ID}, The set of key attributes are: { Recharge_ID }for each FD, check whether the LHS is superkey or the RHS are all key attributes.

BCNF:

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Recharge Plans

Attributes: Plan_No,Price, Data, Calls, SMS, Validity, Other_benefits
Functional Dependency : Plan_No -> Price, Data, Calls, SMS, Validity, Other_benefits

2NF :
The candiates keys are { Plan_No}, The set of key attributes are: { Plan_No }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:
The candiates keys are { Plan_No}, The set of key attributes are: { Plan_No }for each FD, check whether the LHS is superkey or the RHS are all key attributes.
BCNF:
A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Billing

Attributes: Ph_No, Due_Date,Name, Usage_Rs, Fixed_Rentalb_Rs, Total_GST, Previous_Due, Total_Bill
Functional Dependency : Ph_No, Due_Date -> Name, Usage_Rs, Fixed_Rentalb_Rs, Total_GST, Previous_Due, Total_Bill

2NF :
The candiates keys are { Ph_No,Due_Date}, The set of key attributes are: { Ph_No,Due_Date }for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

3NF:
The candiates keys are { Ph_No,Due_Date}, The set of key attributes are: { Ph_No,Due_Date }for each FD, check whether the LHS is superkey or the RHS are all key attributes

BCNF:
A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

The table is **normalized upto BCNF.**

Project Code

Main Program.py > ...

You, 2 seconds ago | 1 author (You)

```python
import mysql.connector      You, 7 days ago • user admin functionalitites ...
import random
from datetime import datetime, timedelta

Pieces: Comment | Pieces: Explain
def connect_to_database():
    try:
        conn = mysql.connector.connect(
            host="127.0.0.1",
            user="root",
            password="mysql",
            database="tele_dbms"
        )
        return conn
    except mysql.connector.Error as e:
        print("Error connecting to MySQL database:", e)
        return None

# Function to get user's plan details
Pieces: Comment | Pieces: Explain
def get_plan_details(customer,conn):

    cursor = conn.cursor()
    if customer['Sim_type'] == 'Prepaid':
        query = "SELECT * FROM plans WHERE Plan_No = %s"
        cursor.execute(query, (customer['Plan_no']))
        plan = cursor.fetchone()
        print("           Your prepaid plan details:")
        print("Plan No:", plan[0])
        print("Price:", plan[1])
        print("Data:", plan[2])
        print("Calls:", plan[3])
        print("SMS:", plan[4])
        print("Validity:", plan[5])
        print("Other Benefits:", plan[6])
        print("\n")
    elif customer['Sim_type'] == 'Postpaid':
        query = "SELECT * FROM billing WHERE Name = %s AND Ph_No = %s"
        cursor.execute(query, (customer['Name'], customer['Ph_No']))
        billing_details = cursor.fetchone()
        print("            Your billing details:")
        print("Usage Rs:", billing_details[2])
        print("Fixed Rental Rs:", billing_details[3])
        print("Total GST:", billing_details[4])
```

Main Program.py > ...

```python
def get_plan_details(customer,conn):
        print("Total GST:", billing_details[4])
        print("Previous Due:", billing_details[5])
        print("Total Bill:", billing_details[6])
        print("\n")
    cursor.close()

# Function to recharge user's mobile number
Pieces: Comment | Pieces: Explain
def recharge_mobile(customer,conn):
    cursor = conn.cursor()
    mobile_number = customer['Ph_No']

    cursor.execute("SELECT MAX(Recharge_ID) FROM recharge")
    max_recharge_id = cursor.fetchone()[0]
    new_recharge_id = max_recharge_id + 1 if max_recharge_id is not None else 1

    recharge_date = datetime.now().date()

    due_date = recharge_date + timedelta(days=random.randint(1, 30))

    activation_time = f"{random.randint(0, 23)}:{random.randint(0, 59)}"

    if customer['Sim_type'] == 'Prepaid':
        new_plan_no = str(input("Enter new plan number: "))
    elif customer['Sim_type'] == 'Postpaid':
        new_plan_no=None
        recharge_amount = float(input("Enter recharge amount: "))

    if recharge_amount <= 0:
        print("Invalid recharge amount. Please enter a valid amount.")
        return


    query = "UPDATE customer SET Plan_No = %s WHERE Ph_No = %s"
    cursor.execute(query, (new_plan_no, mobile_number))
    query = "INSERT INTO recharge (Recharge_ID, Name, Ph_No, Plan_No, Recharge_Date, Due_Date, Activation_Time, Price) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(query, (new_recharge_id, customer['Name'], mobile_number, new_plan_no, recharge_date, due_date, activation_time, recharge_amount))


    query="SELECT * from invoice"
    cursor.execute(query)
    invoices=cursor.fetchall()
    invoice_no=invoices[-1][0]+1
```

Main Program.py > ...

```
49  def recharge_mobile(customer,conn):
83      invoice_no=invoices[-1][0]+1
84      query = "INSERT INTO invoice (Invoice_No,Name, Ph_No, Invoice_Date, Total_Payment, Payment_Mode) VALUES (%s,%s, %s, CURDATE(), %s, %s)"
85      cursor.execute(query, (invoice_no,customer['Name'], mobile_number, recharge_amount, 'Online Payment'))
86      conn.commit()
87      print("Recharge successful!")
88      print("\n")
89      cursor.close()
90
91  # Function to get details of all prepaid plans available
92  def get_prepaid_plans(customer,conn):
93      cursor = conn.cursor()
94      query = "SELECT * FROM plans"
95      cursor.execute(query)
96      prepaid_plans = cursor.fetchall()
97      print("          Prepaid Plans Available:")
98      for plan in prepaid_plans:
99          print("Plan No:", plan[0])
100         print("Price:", plan[1])
101         print("Data:", plan[2])
102         print("Calls:", plan[3])
103         print("SMS:", plan[4])
104         print("Validity:", plan[5])
105         print("Other Benefits:", plan[6])
106         print("\n")
107     cursor.close()
108
109 # Function to get invoice history
    Pieces: Comment | Pieces: Explain
110 def get_invoice_history(user,conn):
111     cursor = conn.cursor()
112     query = "SELECT * FROM invoice WHERE Name = %s AND Ph_No = %s"
113     cursor.execute(query, (user['Name'], user['Ph_No']))
114     invoice_history = cursor.fetchall()
115     print("          Your invoice history:")
116     for invoice in invoice_history:
117         print("Invoice No:", invoice[0])
118         print("Invoice Date:", invoice[4])
119         print("Total Payment:", invoice[5])
120         print("Payment Mode:", invoice[6])
121         print("\n")
122     cursor.close()
123
124 # Function to get recharge history
```

Main Program.py > ...

```
123
124 # Function to get recharge history
    Pieces: Comment | Pieces: Explain
125 def get_recharge_history(user,conn):
126     cursor = conn.cursor()
127     query = "SELECT * FROM recharge WHERE Name = %s AND Ph_No = %s"
128     cursor.execute(query, (user['Name'], user['Ph_No']))
129     recharge_history = cursor.fetchall()
130     print("            Your recharge history:")
131     for recharge in recharge_history:
132         print("Recharge ID:", recharge[0])
133         print("Recharge Plan:", recharge[1])
134         print("Recharge Date:", recharge[4])
135         print("Valid Upto :", recharge[5])
136         print("Activation Time:", recharge[6])
137         print("Recharge Amount:", recharge[7])
138         print("\n")
139     cursor.close()
140
141 # Function to check data usage
    Pieces: Comment | Pieces: Explain
142 def check_data_usage(user,conn):
143     cursor = conn.cursor()
144     query = "SELECT * FROM data_usage WHERE Name = %s AND Ph_No = %s"
145     cursor.execute(query, (user['Name'], user['Ph_No']))
146     data_usage = cursor.fetchone()
147     print("             Your data usage:")
148     print("Session Start Time:", data_usage[2])
149     print("Session End Time:", data_usage[3])
150     print("Data Usage (GB):", data_usage[4])
151     print("Remaining Data (GB):", data_usage[5])
152     print("\n")
153     cursor.close()
154
155 # Function to check call logs
    Pieces: Comment | Pieces: Explain
156 def check_call_logs(user,conn):
157     cursor = conn.cursor()
158     query = "SELECT * FROM call_records WHERE Name = %s AND Ph_No = %s"
159     cursor.execute(query, (user['Name'], user['Ph_No']))
160     call_logs = cursor.fetchall()
161     print("              Your call logs:")
162     for log in call_logs:
163         print("Reciever's Mobile Number:", log[2])
164         print("Call Received Hour Min:", log[3])
```

```
def check_call_logs(user,conn):
            print("Call Received Hour Min:", log[3])
            print("Call Ended Hour Min:", log[4])
            print("Call Duration Hour Min:", log[5])
            print("Location of Phone Number:", log[6])
            print("\n")
        cursor.close()

    # Function to get user's coupons
    Pieces: Comment | Pieces: Explain
    def get_coupons(user,conn):
        cursor = conn.cursor()
        query = "SELECT * FROM coupon "
        cursor.execute(query)
        coupons = cursor.fetchall()
        print("              Your coupons:")
        for coupon in coupons:
            print("Coupon No:", coupon[0])
            print("Brand:", coupon[1])
            print("Price:", coupon[2])
            print("Valid Upto:", coupon[3])
            print("\n")
        cursor.close()

    # Function to update user's profile
    Pieces: Comment | Pieces: Explain
    def update_profile(user,conn):
        cursor = conn.cursor()

        # Fetch user's current profile details

        query = "SELECT * FROM personal WHERE Name = %s AND Ph_No = %s"
        cursor.execute(query, (user['Name'], user['Ph_No']))
        current_profile = cursor.fetchone()
        print("                Current Profile Details:")
        print("Name:", current_profile[0])
        print("Phone Number:", current_profile[1])
        print("Alternate Phone Number:", current_profile[2])
        print("Address:", current_profile[3])
        print("Email ID:", current_profile[4])
        print("Date of Birth:", current_profile[5])
        print("Aadhaar Number:", current_profile[6])
        print("\n")
```

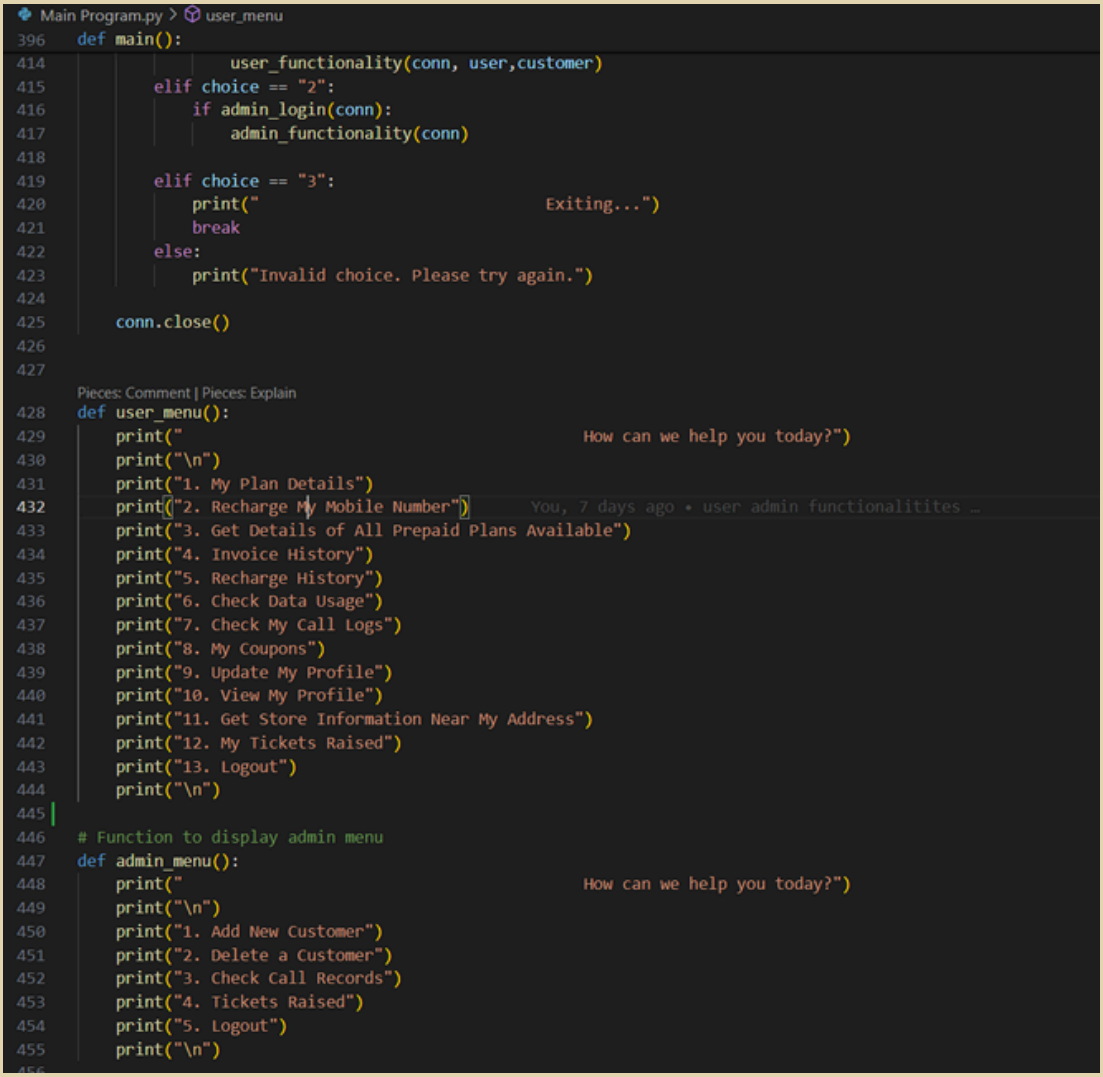
```
def update_profile(user,conn):

    # Ask user for updated details

    new_name = input("Enter new name (leave empty to keep current): ").strip() or current_profile[0]
    new_alt_ph_no = input("Enter new alternate phone number (leave empty to keep current): ").strip() or current_profile[2]
    new_address = input("Enter new address (leave empty to keep current): ").strip() or current_profile[3]
    new_email = input("Enter new email ID (leave empty to keep current): ").strip() or current_profile[4]
    new_dob = input("Enter new date of birth (YYYY-MM-DD) (leave empty to keep current): ").strip() or current_profile[5]

    # Update user's profile in the database

    query = "UPDATE personal SET Name = %s, Alt_Ph_No = %s, Address = %s, Email_ID = %s, DOB = %s WHERE Name = %s AND Ph_No = %s"
    cursor.execute(query, (new_name, new_alt_ph_no, new_address, new_email, new_dob, user['Name'], user['Ph_No']))
    conn.commit()
    print("\n")
    print("Profile updated successfully!")
    print("\n")
    cursor.close()

# Function to view user's profile
def view_profile(user,conn):
    cursor = conn.cursor()
    query = "SELECT * FROM personal WHERE Name = %s AND Ph_No = %s"
    cursor.execute(query, (user['Name'], user['Ph_No']))
    profile_details = cursor.fetchone()
    print("              Your Profile:")
    print("Name:", profile_details[0])
    print("Phone Number:", profile_details[1])
    print("Alternate Phone Number:", profile_details[2])
    print("Address:", profile_details[3])
    print("Email ID:", profile_details[4])
    print("Date of Birth:", profile_details[5])
    print("Aadhaar Number:", profile_details[6])
    print("\n")
    cursor.close()

# Function to get store information near user's address
Pieces: Comment | Pieces: Explain
def get_store_information(user,conn):
    cursor = conn.cursor()
    query = "SELECT * FROM store WHERE Address = %s"
    cursor.execute(query, (user['Address'],))
    store_info = cursor.fetchall()
    print("                Store Information Near Your Address:")
```

```
241    def get_store_information(user,conn):
248            print("Store No:", store[0])
249            print("Address:", store[1])
250            print("City:", store[2])
251            print("Contact No:", store[3])
252            print("\n")
253
254        cursor.close()
255
256    # Function to get user's raised tickets
       Pieces: Comment | Pieces: Explain
257    def get_tickets(customer,conn):
258        cursor = conn.cursor()
259        query = "SELECT * FROM ticket WHERE Cust_ID = %s"
260        cursor.execute(query, (customer['Cust_ID'],))
261        tickets = cursor.fetchall()
262        print("                  Your raised tickets:")
263        for ticket in tickets:
264            print("Ticket No:", ticket[0])
265            print("Issue:", ticket[1])
266            print("Date Raised:", ticket[3])
267            print("Ticket Status:", ticket[4])
268            print("\n")
269
270        cursor.close()
271
272    # Function to add new customer (admin functionality)
       Pieces: Comment | Pieces: Explain
273    def add_new_customer(conn):
274        cursor = conn.cursor()
275        print("                Add New Customer:")
276        query="SELECT * FROM customer "
277        cursor.execute(query)
278        customers=cursor.fetchall()
279        cust_id=customers[-1][0]+1
280        name = str(input("Enter customer name: "))
281        ph_no = int(input("Enter customer phone number: "))
282        alt_no = int(input("Enter customer's alternate phone number: "))
283        address = str(input("Enter customer address: "))
284        email = str(input("Enter customer email ID: "))
285        dob = str(input("Enter customer date of birth (YYYY-MM-DD): "))
286        aadhaar = int(input("Enter customer Aadhaar number: "))
287        user_name = str(input("Enter username: "))
288        password = str(input("Enter password: "))
289        sim_type = str(input("Enter SIM type (Prepaid/Postpaid): "))
```

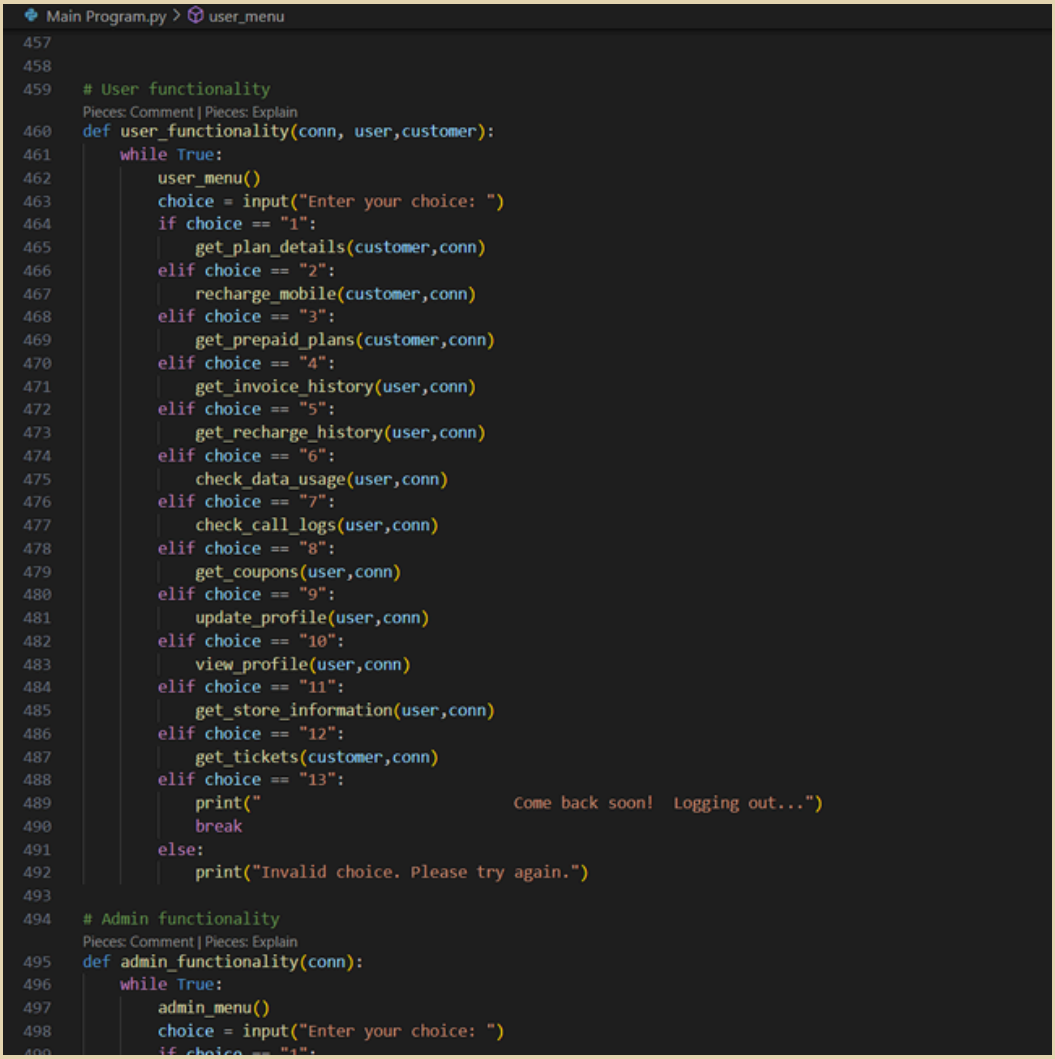
```
273  def add_new_customer(conn):
288      password = str(input("Enter password: "))
289      sim_type = str(input("Enter SIM type (Prepaid/Postpaid): "))
290      if (sim_type == 'Prepaid'):
291          plan_no = str(input("Enter plan number "))
292      else:
293          plan_no = None
294      fixed_rental = float(input("Enter fixed rental: "))
295
296      query = "INSERT INTO personal (Name, Ph_No, Alt_Ph_No, Address, Email_ID, DOB, Aadhaar_No, Username, Password) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
297      cursor.execute(query, (name, ph_no, alt_no, address, email, dob, aadhaar, user_name, password))
298      conn.commit()
299      query = "INSERT INTO customer (Cust_ID,Name, Ph_No, Sim_type, Plan_No,Fixed_rental) VALUES (%s,%s, %s, %s, %s,%s)"
300      cursor.execute(query, (cust_id,name, ph_no, sim_type, plan_no,fixed_rental))
301      conn.commit()
302      print("\n")
303      print("New customer added successfully!")
304      print("\n")
305      cursor.close()
306
307  # Function to delete a customer (admin functionality)
     Pieces: Comment | Pieces: Explain
308  def delete_customer(conn):
309      cursor = conn.cursor()
310      print("                Delete a Customer:")
311      ph_no = input("Enter customer phone number to delete: ")
312      query = "DELETE FROM customer WHERE Ph_No = %s"
313      cursor.execute(query, (ph_no))
314      conn.commit()
315      print("\n")
316      print("Customer deleted successfully!")
317      print("\n")
318      cursor.close()
319
320  # Function to check call records (admin functionality)
321  def check_call_records(conn):
322      cursor = conn.cursor()
323      print("              Check Call Records:")
324      query = "SELECT * FROM call_records"
325      cursor.execute(query)
326      call_records = cursor.fetchall()
327      for record in call_records:
328          print("Name:", record[0])
329          print("Phone Number:", record[1])
330          print("Call Logs Per Day:", record[2])
```

```
def check_call_records(conn):
        print("Call Logs Per Day:", record[2])
        print("Call Received Hour Min:", record[3])
        print("Call Ended Hour Min:", record[4])
        print("Call Duration Hour Min:", record[5])
        print("Location of Phone Number:", record[6])
        print("\n")
    cursor.close()


# Function to check tickets raised (admin functionality)
def check_tickets_raised(conn):
    cursor = conn.cursor()
    print("               Tickets Raised:")
    query = "SELECT * FROM ticket"
    cursor.execute(query)
    tickets = cursor.fetchall()
    for ticket in tickets:
        print("Ticket No:", ticket[0])
        print("Issue:", ticket[1])
        print("Customer ID:", ticket[2])
        print("Date Raised:", ticket[3])
        print("Ticket Status:", ticket[4])
        print("\n")

    cursor.close()

Pieces: Comment | Pieces: Explain
def user_login(conn):
    cursor = conn.cursor()
    while True:
        username = input("Enter username: ")
        password = input("Enter password: ")
        query = "SELECT * FROM personal WHERE Username = %s AND Password = %s"
        cursor.execute(query, (username, password))
        user_data = cursor.fetchone()
        if user_data:
            user = dict(zip([column[0] for column in cursor.description], user_data))
            query2 = "SELECT * FROM customer WHERE Name = %s"
            cursor.execute(query2, (user['Name'],))
            customer_data = cursor.fetchone()
            customer = dict(zip([column[0] for column in cursor.description], customer_data))
            print("\n")
            print("                    Login successful!")
```

```
def user_login(conn):
                print("                    Login successful!")
                print("\n")
                cursor.close()
                return user, customer
            else:
                print("Invalid username or password. Please try again.")


# Function to handle admin login
Pieces: Comment | Pieces: Explain
def admin_login(conn):
    cursor = conn.cursor()
    while True:
        username = input("Enter username: ")
        password = input("Enter password: ")
        if username == "admin" and password == "admin123":
            print("\n")
            print("                    Login successful!")
            print("\n")
            cursor.close()
            return True
        else:
            print("Invalid admin username or password. Please try again.")


# Main function
Pieces: Comment | Pieces: Explain
def main():
    conn = connect_to_database()
    if conn is None:
        return

    while True:
        print("                                        Glad to have you back!")
        print("\n")
        print("1. User Login")
        print("2. Admin Login")
        print("3. Exit")
        print("\n")
        choice = input("Enter your choice: ")
        print("\n")

        if choice == "1":
            user,customer = user_login(conn)
```



```
Main Program.py > user_menu
396  def main():
414                  user_functionality(conn, user,customer)
415          elif choice == "2":
416              if admin_login(conn):
417                  admin_functionality(conn)
418
419          elif choice == "3":
420              print("                              Exiting...")
421              break
422          else:
423              print("Invalid choice. Please try again.")
424
425      conn.close()
426
427
         Pieces: Comment | Pieces: Explain
428  def user_menu():
429      print("                                      How can we help you today?")
430      print("\n")
431      print("1. My Plan Details")
432      print("2. Recharge My Mobile Number")      You, 7 days ago • user admin functionalitites …
433      print("3. Get Details of All Prepaid Plans Available")
434      print("4. Invoice History")
435      print("5. Recharge History")
436      print("6. Check Data Usage")
437      print("7. Check My Call Logs")
438      print("8. My Coupons")
439      print("9. Update My Profile")
440      print("10. View My Profile")
441      print("11. Get Store Information Near My Address")
442      print("12. My Tickets Raised")
443      print("13. Logout")
444      print("\n")
445
446  # Function to display admin menu
447  def admin_menu():
448      print("                                      How can we help you today?")
449      print("\n")
450      print("1. Add New Customer")
451      print("2. Delete a Customer")
452      print("3. Check Call Records")
453      print("4. Tickets Raised")
454      print("5. Logout")
455      print("\n")
```



```
Main Program.py > user_menu
457
458
459  # User functionality
     Pieces: Comment | Pieces: Explain
460  def user_functionality(conn, user,customer):
461      while True:
462          user_menu()
463          choice = input("Enter your choice: ")
464          if choice == "1":
465              get_plan_details(customer,conn)
466          elif choice == "2":
467              recharge_mobile(customer,conn)
468          elif choice == "3":
469              get_prepaid_plans(customer,conn)
470          elif choice == "4":
471              get_invoice_history(user,conn)
472          elif choice == "5":
473              get_recharge_history(user,conn)
474          elif choice == "6":
475              check_data_usage(user,conn)
476          elif choice == "7":
477              check_call_logs(user,conn)
478          elif choice == "8":
479              get_coupons(user,conn)
480          elif choice == "9":
481              update_profile(user,conn)
482          elif choice == "10":
483              view_profile(user,conn)
484          elif choice == "11":
485              get_store_information(user,conn)
486          elif choice == "12":
487              get_tickets(customer,conn)
488          elif choice == "13":
489              print("                          Come back soon!  Logging out...")
490              break
491          else:
492              print("Invalid choice. Please try again.")
493
494  # Admin functionality
     Pieces: Comment | Pieces: Explain
495  def admin_functionality(conn):
496      while True:
497          admin_menu()
498          choice = input("Enter your choice: ")
```

```python

# Admin functionality
Pieces: Comment | Pieces: Explain
def admin_functionality(conn):
    while True:
        admin_menu()
        choice = input("Enter your choice: ")
        if choice == "1":
            add_new_customer(conn)
        elif choice == "2":
            delete_customer(conn)
        elif choice == "3":
            check_call_records(conn)
        elif choice == "4":
            check_tickets_raised(conn)
        elif choice == "5":
            print("                          Come back soon!  Logging out...")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()

```

Output

```
PS C:\DEVAM\MY PROJECTS\Telecommunication DBMS> & C:/Users/devam/AppData/Local/Programs/Python/Python312/python.exe "c:/DEVAM/MY PROJECTS/Telecommunication DBMS/Main Program.py"
                                Glad to have you back!


1. User Login
2. Admin Login
3. Exit


Enter your choice: 1


Enter username: yash
Enter password: 100


                  Login successful!


                                How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 1
          Your billing details:
Usage Rs: 451.29
Fixed Rental Rs: 45
Total GST: 570.73
Previous Due: 4.11
Total Bill: 574.84


                                How can we help you today?
```

```
                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 2
Enter recharge amount: 2090
Recharge successful!

                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 3
        Prepaid Plans Available:
Plan No: P1
Price: Rs 409
Data: 4 GB/Day
Calls: Unlimited
```

```
Enter your choice: 3
        Prepaid Plans Available:
Plan No: P1
Price: Rs 409
Data: 4 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 56 Days
Other Benefits: Binge all night offer


Plan No: P10
Price: Rs 16
Data: 1 GB
Calls: Null
SMS: Null
Validity: 24 Hours
Other Benefits: Null


Plan No: P2
Price: Rs 559
Data: 1.5 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 84 Days
Other Benefits: Weekend data rollover


Plan No: P3
Price: Rs 149
Data: 2 GB
Calls: Unlimited
SMS: 100 SMS
Validity: 28 Days
Other Benefits: Null


Plan No: P4
Price: Rs 761
Data: 3 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 84 Days
Other Benefits: 1 Year hotstar VIP


Plan No: P5
Price: Rs 1157
```

```
Plan No: P5
Price: Rs 1157
Data: 1.5 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 180 Days
Other Benefits: Zee5 premium-1 year


Plan No: P6
Price: Rs 1499
Data: 24 GB
Calls: Unlimited
SMS: 100 SMS
Validity: 365 Days
Other Benefits: Null


Plan No: P7
Price: Rs 2359
Data: 1.5 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 365 Days
Other Benefits: Zee5 premium-1 year


Plan No: P8
Price: Rs 499
Data: 1.5 GB/Day
Calls: Unlimited
SMS: 100 SMS/Day
Validity: 70 Days
Other Benefits: Weekend data rollover


Plan No: P9
Price: Rs 447
Data: 50 GB
Calls: Unlimited
SMS: 100 SMS
Validity: 60 Days
Other Benefits: Null

                                    How can we help you today?


1. My Plan Details
```

```
                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 4
          Your invoice history:
Invoice No: 201
Invoice Date: 2023-11-02
Total Payment: 2999
Payment Mode: Credit Card


Invoice No: 209
Invoice Date: 2024-04-16
Total Payment: 139
Payment Mode: Online Payment


Invoice No: 210
Invoice Date: 2024-04-16
Total Payment: 234
Payment Mode: Online Payment


Invoice No: 211
Invoice Date: 2024-04-22
Total Payment: 5000
Payment Mode: Online Payment


Invoice No: 212
Invoice Date: 2024-04-23
Total Payment: 139
```

```
Invoice No: 212
Invoice Date: 2024-04-23
Total Payment: 139
Payment Mode: Online Payment


Invoice No: 213
Invoice Date: 2024-04-23
Total Payment: 353
Payment Mode: Online Payment


Invoice No: 214
Invoice Date: 2024-04-23
Total Payment: 873
Payment Mode: Online Payment


Invoice No: 215
Invoice Date: 2024-04-23
Total Payment: 342
Payment Mode: Online Payment


Invoice No: 216
Invoice Date: 2024-04-23
Total Payment: 2090
Payment Mode: Online Payment

                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout
```

```
Enter your choice: 5
            Your recharge history:
Recharge ID: 209
Recharge Plan: Yashraj Oberoi
Recharge Date: 2024-04-23
Valid Upto : 2024-05-04
Activation Time: 6:26
Recharge Amount: 873.0


Recharge ID: 210
Recharge Plan: Yashraj Oberoi
Recharge Date: 2024-04-23
Valid Upto : 2024-05-02
Activation Time: 11:1
Recharge Amount: 342.0


Recharge ID: 211
Recharge Plan: Yashraj Oberoi
Recharge Date: 2024-04-23
Valid Upto : 2024-05-21
Activation Time: 15:23
Recharge Amount: 2090.0

                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 6
            Your data usage:
Session Start Time: 12:00
Session End Time: 12:30
```

```
Enter your choice: 6
            Your data usage:
Session Start Time: 12:00
Session End Time: 12:30
Data Usage (GB): 1
Remaining Data (GB): 2


                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 7
            Your call logs:
Reciever's Mobile Number: 6543209122
Call Received Hour Min: 00:00
Call Ended Hour Min: 01:21
Call Duration Hour Min: 01:21
Location of Phone Number: Tamil Nadu


                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
```

```
Enter your choice: 8
                Your coupons:
Coupon No: 322
Brand: Ajio
Price: 1000
Valid Upto: 2025-04-03


Coupon No: 342
Brand: Puma
Price: 750
Valid Upto: 2024-12-01


Coupon No: 343
Brand: Mamaearth
Price: 350
Valid Upto: 2024-07-22


Coupon No: 345
Brand: Amazon Pay
Price: 150
Valid Upto: 2025-11-14


Coupon No: 356
Brand: Cred
Price: 450
Valid Upto: 2024-10-23


                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout
```

```
Enter your choice: 9
                Current Profile Details:
Name: Yashraj Oberoi
Phone Number: 8570099914
Alternate Phone Number: 8475647384
Address: Gujarat
Email ID: yashraj@gmail.com
Date of Birth: 2002-04-15
Aadhaar Number: 657465746574


Enter new name (leave empty to keep current):
Enter new alternate phone number (leave empty to keep current):
Enter new address (leave empty to keep current):
Enter new email ID (leave empty to keep current):
Enter new date of birth (YYYY-MM-DD) (leave empty to keep current):


Profile updated successfully!


                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 10
            Your Profile:
Name: Yashraj Oberoi
Phone Number: 8570099914
Alternate Phone Number: 8475647384
Address: Gujarat
Email ID: yashraj@gmail.com
Date of Birth: 2002-04-15
Aadhaar Number: 657465746574
```

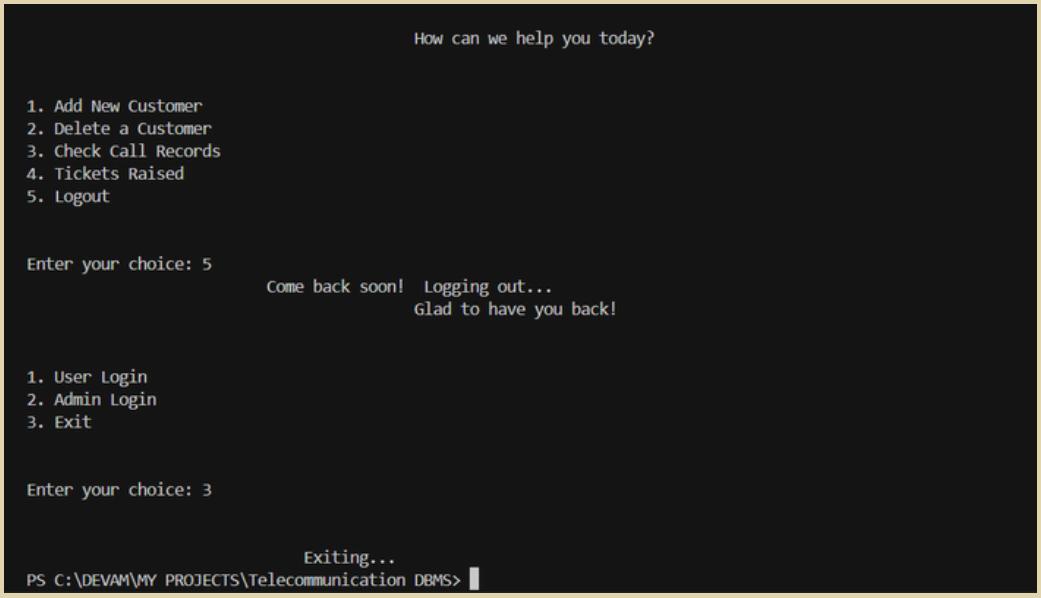
```
                                        How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 11
                    Store Information Near Your Address:
Store No: 1900
Address: Gujarat
City: Surat
Contact No: 7867564736


Store No: 1902
Address: Gujarat
City: Ahmedabad
Contact No: 9089898999


Store No: 1903
Address: Gujarat
City: Rajkot
Contact No: 7878788987


Store No: 1907
Address: Gujarat
City: Surat
Contact No: 3435343635


Store No: 1908
Address: Gujarat
City: Vadodara
Contact No: 6745563746
```

```
                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 12
                Your raised tickets:
Ticket No: 781
Issue: Payment error
Date Raised: 2024-04-15
Ticket Status: Pending


                                    How can we help you today?


1. My Plan Details
2. Recharge My Mobile Number
3. Get Details of All Prepaid Plans Available
4. Invoice History
5. Recharge History
6. Check Data Usage
7. Check My Call Logs
8. My Coupons
9. Update My Profile
10. View My Profile
11. Get Store Information Near My Address
12. My Tickets Raised
13. Logout


Enter your choice: 13
                    Come back soon!  Logging out...
                                    Glad to have you back!
```

```
1. User Login
2. Admin Login
3. Exit


Enter your choice: 2


Enter username: admin
Enter password: admin123


                    Login successful!


                                        How can we help you today?


1. Add New Customer
2. Delete a Customer
3. Check Call Records
4. Tickets Raised
5. Logout


Enter your choice: 1
                Add New Customer:
Enter customer name: Devam Patel
Enter customer phone number: 67457987645
Enter customer's alternate phone number: 34875483570
Enter customer address: Delhi
Enter customer email ID: dev@gmail.com
Enter customer date of birth (YYYY-MM-DD): 2002-12-2
Enter customer Aadhaar number: 543682374693
Enter username: deva
Enter password: 234
Enter SIM type (Prepaid/Postpaid): Prepaid
Enter plan number P4
Enter fixed rental: 37


New customer added successfully!


                                        How can we help you today?
```

```
                                    How can we help you today?


1. Add New Customer
2. Delete a Customer
3. Check Call Records
4. Tickets Raised
5. Logout


Enter your choice: 3
                Check Call Records:
Name: Yashraj Oberoi
Phone Number: 8570099914
Call Logs Per Day: 6543209122
Call Received Hour Min: 00:00
Call Ended Hour Min: 01:21
Call Duration Hour Min: 01:21
Location of Phone Number: Tamil Nadu


Name: Shraddha Juneja
Phone Number: 7400999822
Call Logs Per Day: +4498200091
Call Received Hour Min: 03:41
Call Ended Hour Min: 03:51
Call Duration Hour Min: 00:10
Location of Phone Number: Canada


Name: Ram Malhotra
Phone Number: 9925678011
Call Logs Per Day: +168709240
Call Received Hour Min: 06:03
Call Ended Hour Min: 08:09
Call Duration Hour Min: 02:06
Location of Phone Number: America


Name: Parth Pillai
Phone Number: 6942000971
Call Logs Per Day: 1800709821
Call Received Hour Min: 06:53
Call Ended Hour Min: 07:01
Call Duration Hour Min: 00:08
Location of Phone Number: Ranchi
```

```
                                    How can we help you today?


1. Add New Customer
2. Delete a Customer
3. Check Call Records
4. Tickets Raised
5. Logout


Enter your choice: 4
                Tickets Raised:
Ticket No: 781
Issue: Payment error
Customer ID: 1001
Date Raised: 2024-04-15
Ticket Status: Pending


Ticket No: 783
Issue: Network Issue
Customer ID: 1003
Date Raised: 2024-03-23
Ticket Status: Resolved


Ticket No: 785
Issue: Plan not activated
Customer ID: 1005
Date Raised: 2024-04-03
Ticket Status: Pending


Ticket No: 786
Issue: Refund pending
Customer ID: 1007
Date Raised: 2024-04-01
Ticket Status: Resolved


Ticket No: 790
Issue: Nework Issue
Customer ID: 1008
Date Raised: 2024-04-05
Ticket Status: Pending


                                    How can we help you today?
```



Conclusion

The Telecommunication Management System provides a comprehensive set of features for both users and administrators to manage telecommunication services efficiently. With functionalities ranging from plan management to customer support ticket handling, the system aims to streamline operations and enhance customer satisfaction.