DB_Project_Assignment

Name: Ronit Jain (202001081)

Nipun Shah (202001096)

Group No: 2

Section No: 6

Team Id: 6.1

Faculty Name: Minal Bhise and Rachit Chaya

TA Mentor: Pinak Gajera

Case Study

Beach Activity Management System:

All beach activity-related records should be created in this database. The beach-related activities like scuba diving, snorkeling, paragliding, flyboarding, windsurfing, and so on. Precautionary things like helmets, swimsuits, quality shoes, gloves, etc., for particular activities, must be appropriately managed for all customers. The other things related to taking pictures and videos are also included in this database.

Front-End Development:

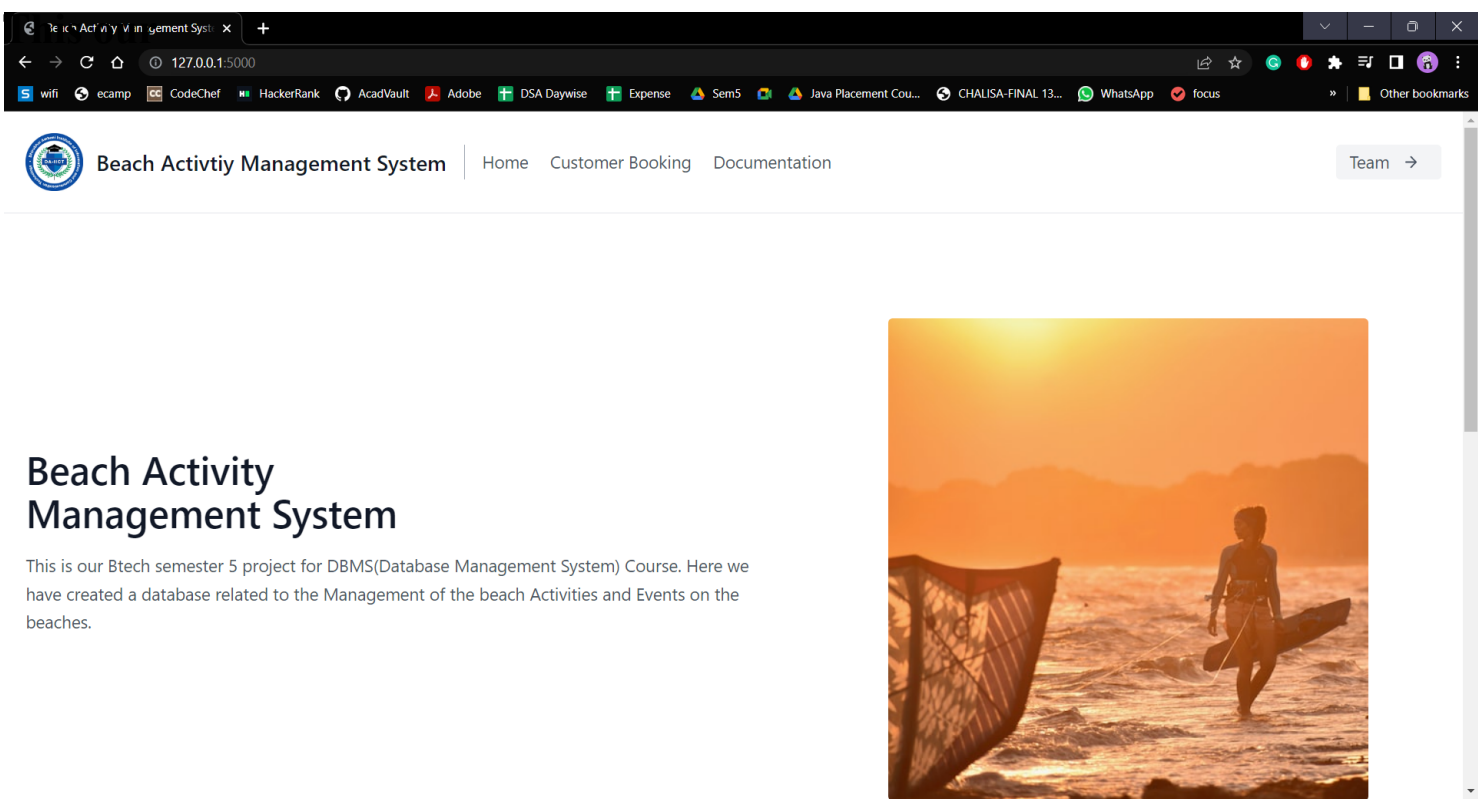
We are using python to connect our database. We have used Html code to create a webpage and CSS from a tailwind.

Connection Code:

```
from flask import Flask, render_template, request,session,redirect  ,jsonify
import psycopg2
from flask_session import Session
from json import dumps
from datetime import date
app = Flask(__name__)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
app.static_folder='static'
Session(app)
conn = psycopg2.connect(host='localhost', database='bams',
                        user='postgres', password='ronit123')
cursor = conn.cursor()
user_id = 0
```

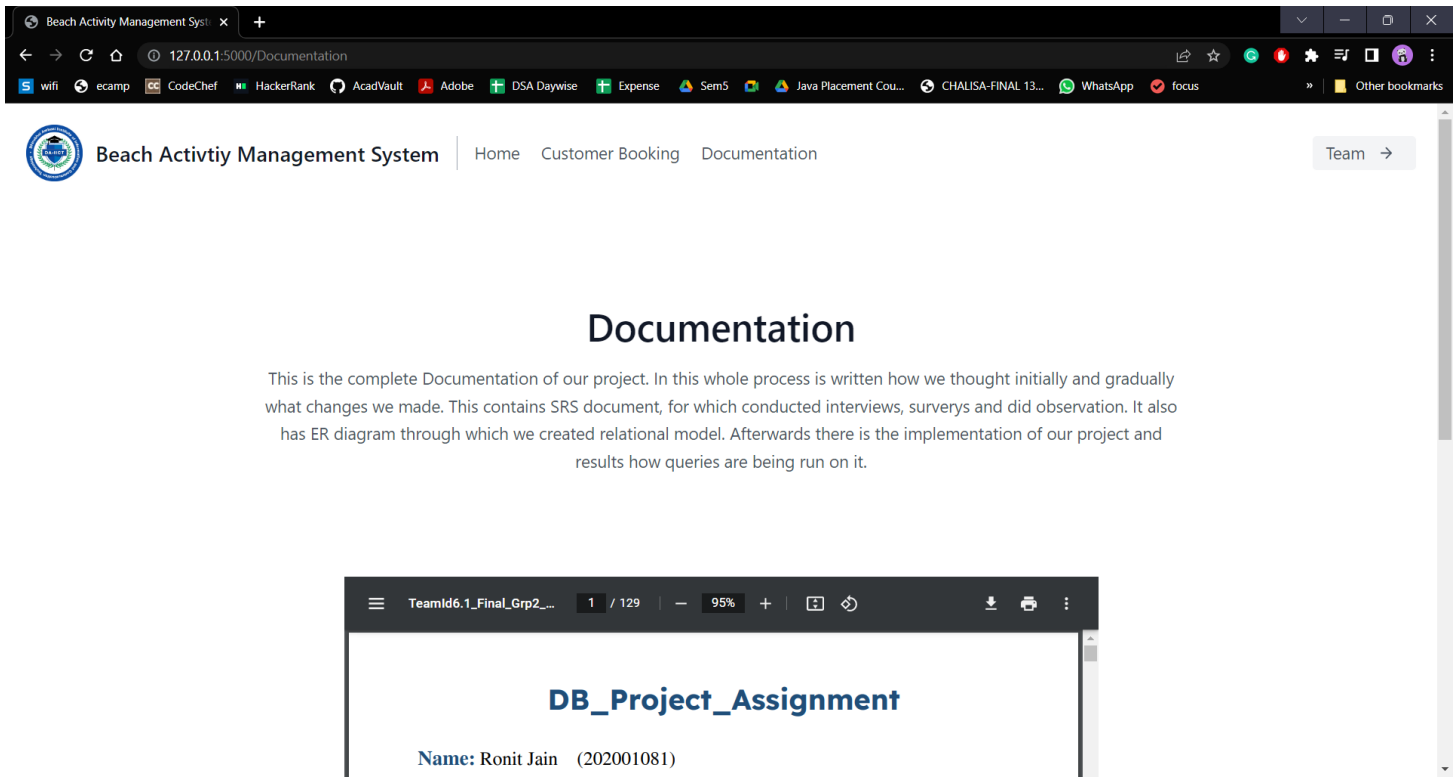
Functioning of Queries and Website:

Home Page:



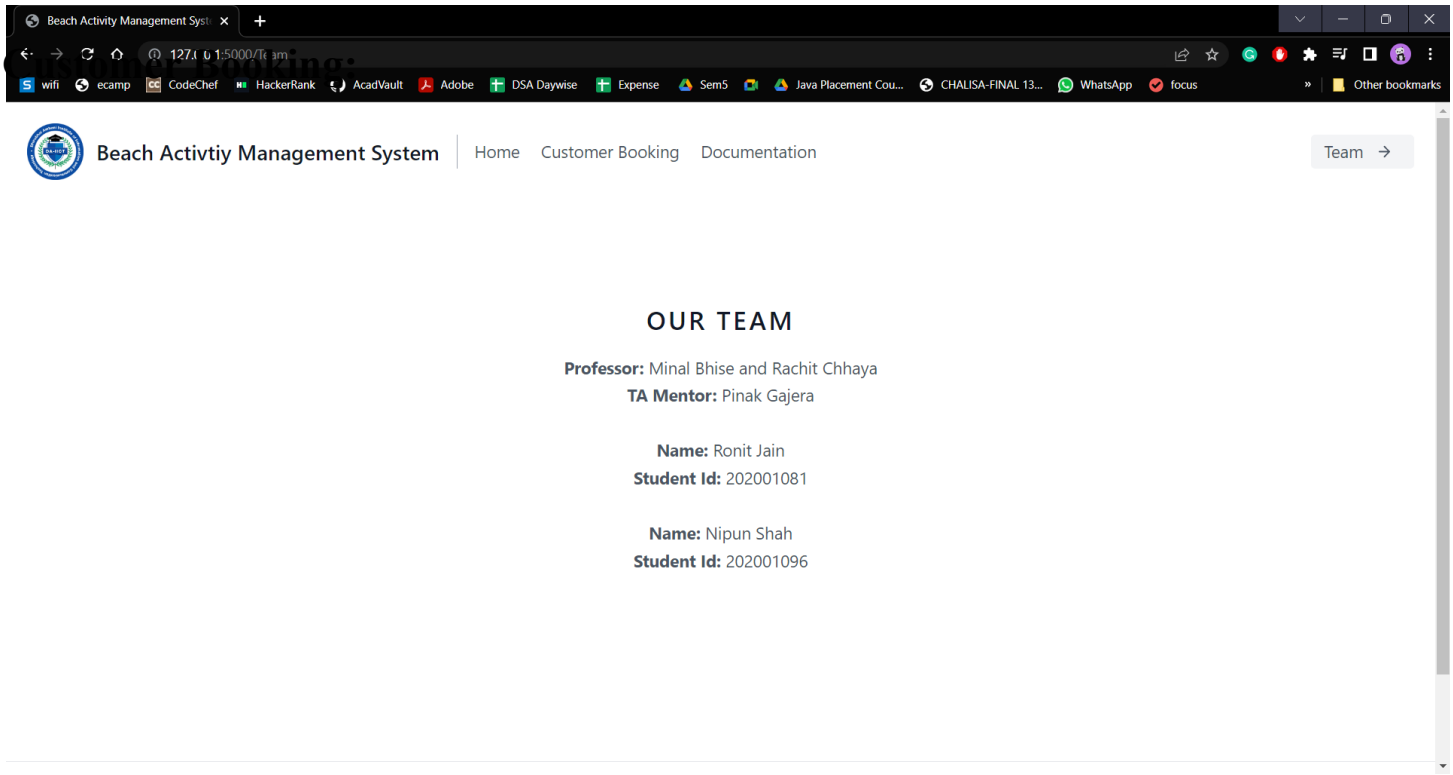
Documentation:

This page contains the pdf document of our project which shows all the details of the work done by us for creating database and beach management system.



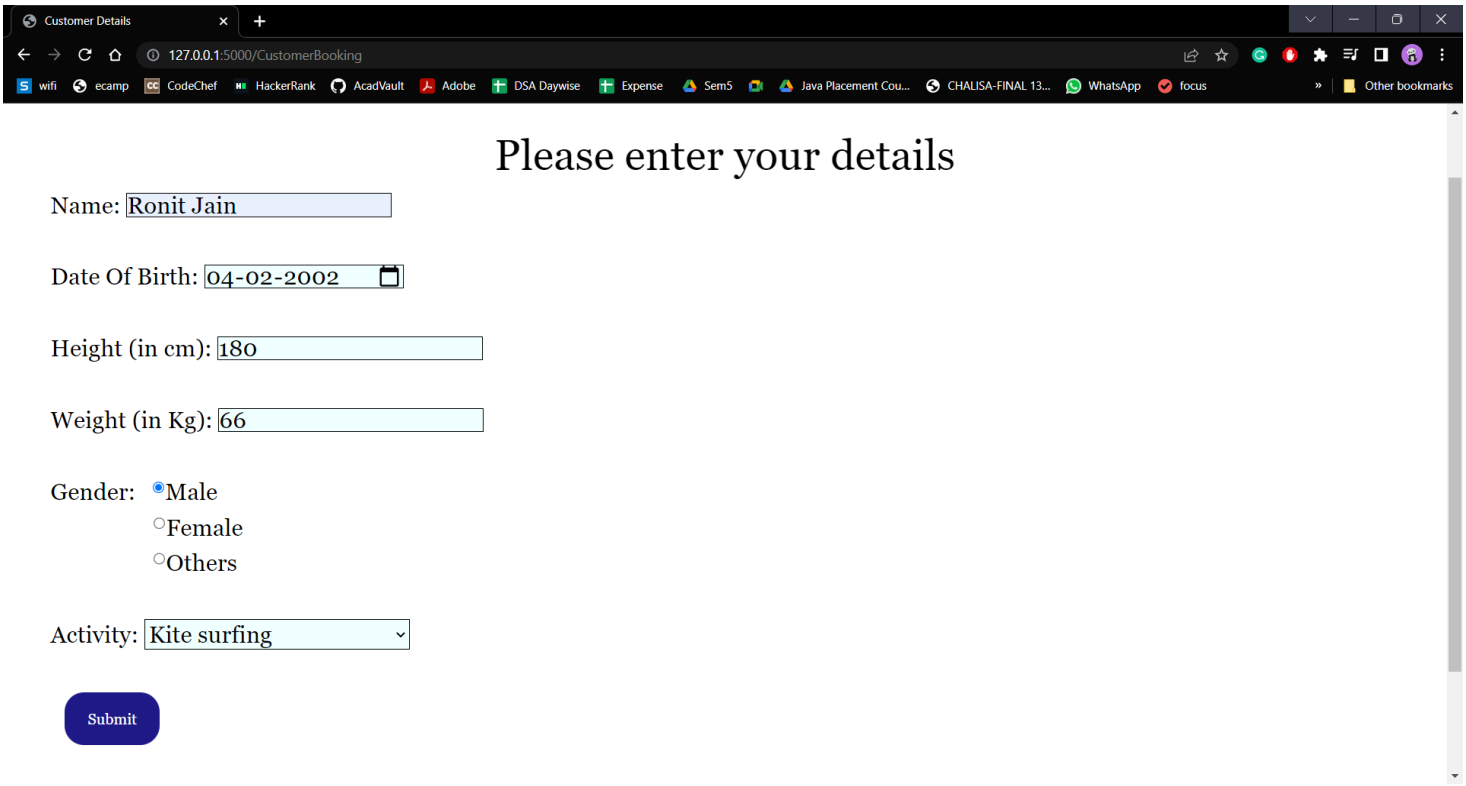
Team:

This page show details about our team, TA mentor and Professors.



Customer Booking:

This is our first page where customer enters his details about himself which when he submits are inserted into Customer table in our database and Customer_id is automatically generated.



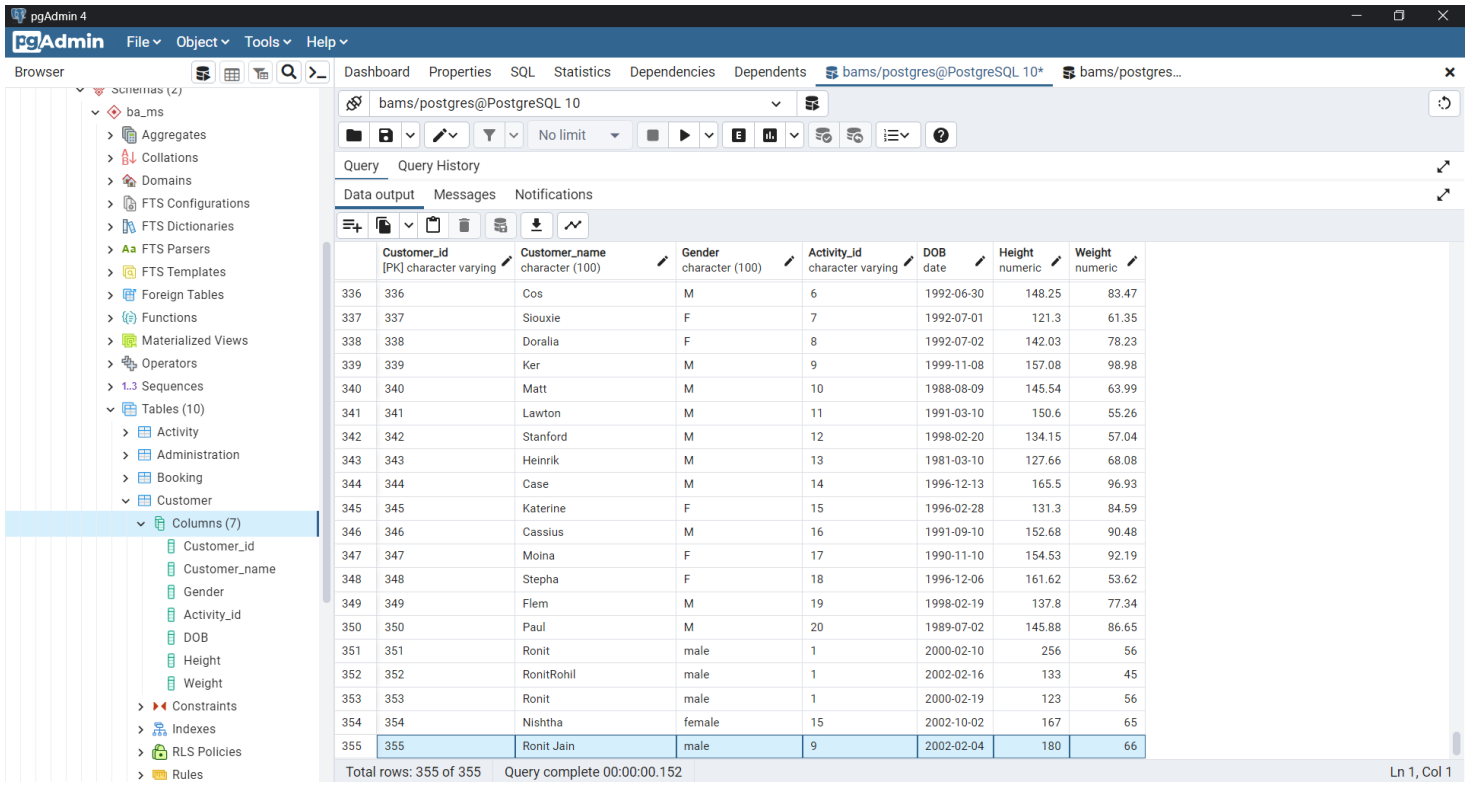
Here in this form we enter all the details and they are inserted in the customer table and Customer_id is automatically generated.

```
@app.route('/CustomerBooking', methods=['post','get'])
def CustomerBooking():
    print("Hello")
    if request.method == 'POST':
        Customer_name = request.form.get('name')
        DOB = request.form.get('dob')
        Height = request.form.get('height')
        Weight = request.form.get('weight')
        Gender = request.form.get('Gender')
        Activity_id = request.form.get('activity_id')
        cursor.execute('select count(*) from ba_ms.\"Customer\"')
        result = cursor.fetchone()
        count = result[0]+1
```

```
        print(result)
        insert_query = """ INSERT INTO ba_ms.\"Customer\" VALUES
(%s,%s,%s,%s,%s,%s,%s)"""
        try:
            record = (int(count), Customer_name, Gender, Activity_id, DOB,
int(Height), int(Weight))
            print(insert_query,record)
            cursor.execute(insert_query, record)
            conn.commit()
            count = cursor.rowcount
            print(count, "Record inserted successfully into Customer table")

        except:
            return "Invalid details"
    return render_template(Booking.html')
```

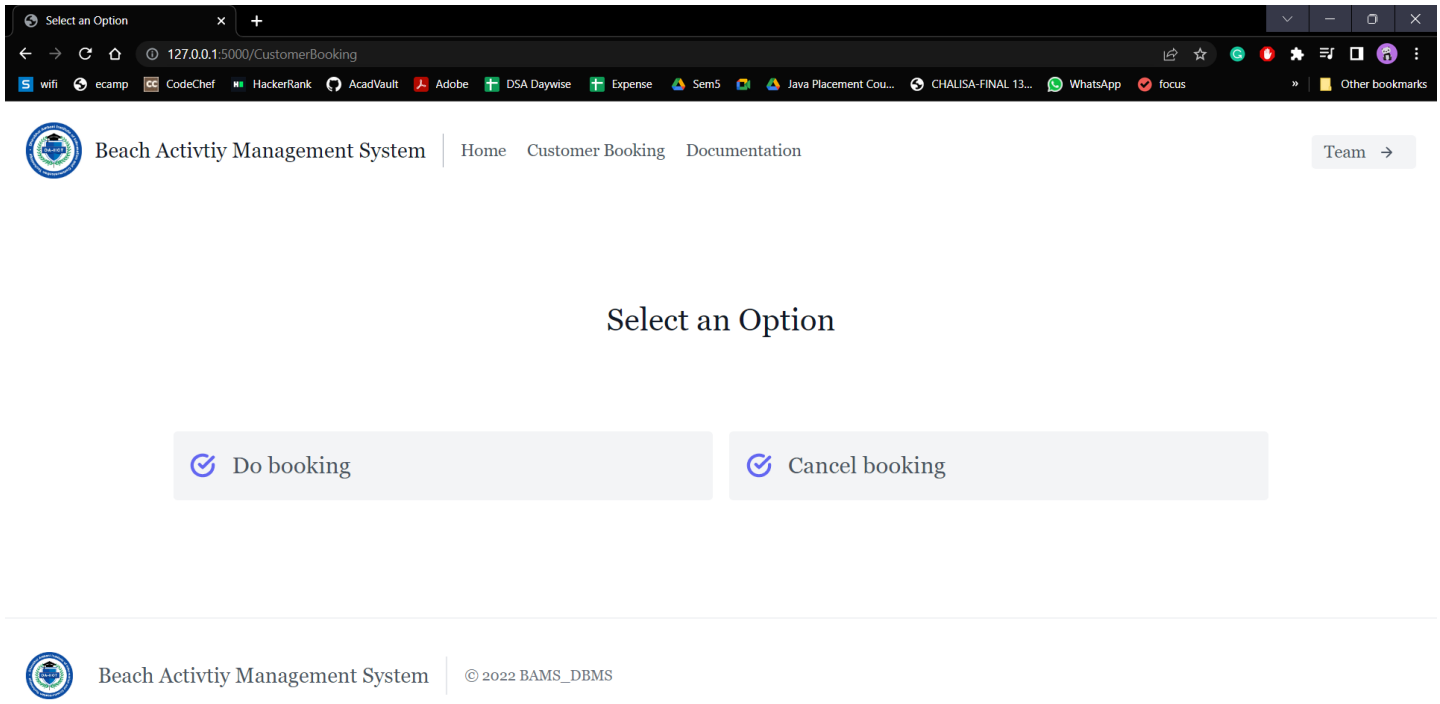
This code inserts data into Customer table entered on the website and generates automatically customer_id.



	Customer_id [PK] character varying	Customer_name character (100)	Gender character (100)	Activity_id character varying	DOB date	Height numeric	Weight numeric
336	336	Cos	M	6	1992-06-30	148.25	83.47
337	337	Siouxie	F	7	1992-07-01	121.3	61.35
338	338	Doralia	F	8	1992-07-02	142.03	78.23
339	339	Ker	M	9	1999-11-08	157.08	98.98
340	340	Matt	M	10	1988-08-09	145.54	63.99
341	341	Lawton	M	11	1991-03-10	150.6	55.26
342	342	Stanford	M	12	1998-02-20	134.15	57.04
343	343	Heinrik	M	13	1981-03-10	127.66	68.08
344	344	Case	M	14	1996-12-13	165.5	96.93
345	345	Katerine	F	15	1996-02-28	131.3	84.59
346	346	Cassius	M	16	1991-09-10	152.68	90.48
347	347	Moina	F	17	1990-11-10	154.53	92.19
348	348	Stepha	F	18	1996-12-06	161.62	53.62
349	349	Flem	M	19	1998-02-19	137.8	77.34
350	350	Paul	M	20	1989-07-02	145.88	86.65
351	351	Ronit	male	1	2000-02-10	256	56
352	352	RonitRohil	male	1	2002-02-16	133	45
353	353	Ronit	male	1	2000-02-19	123	56
354	354	Nishtha	female	15	2002-10-02	167	65
355	355	Ronit Jain	male	9	2002-02-04	180	66

Booking:

After submitting the details customer has two options from where he can either cancel the booking or make a new booking.



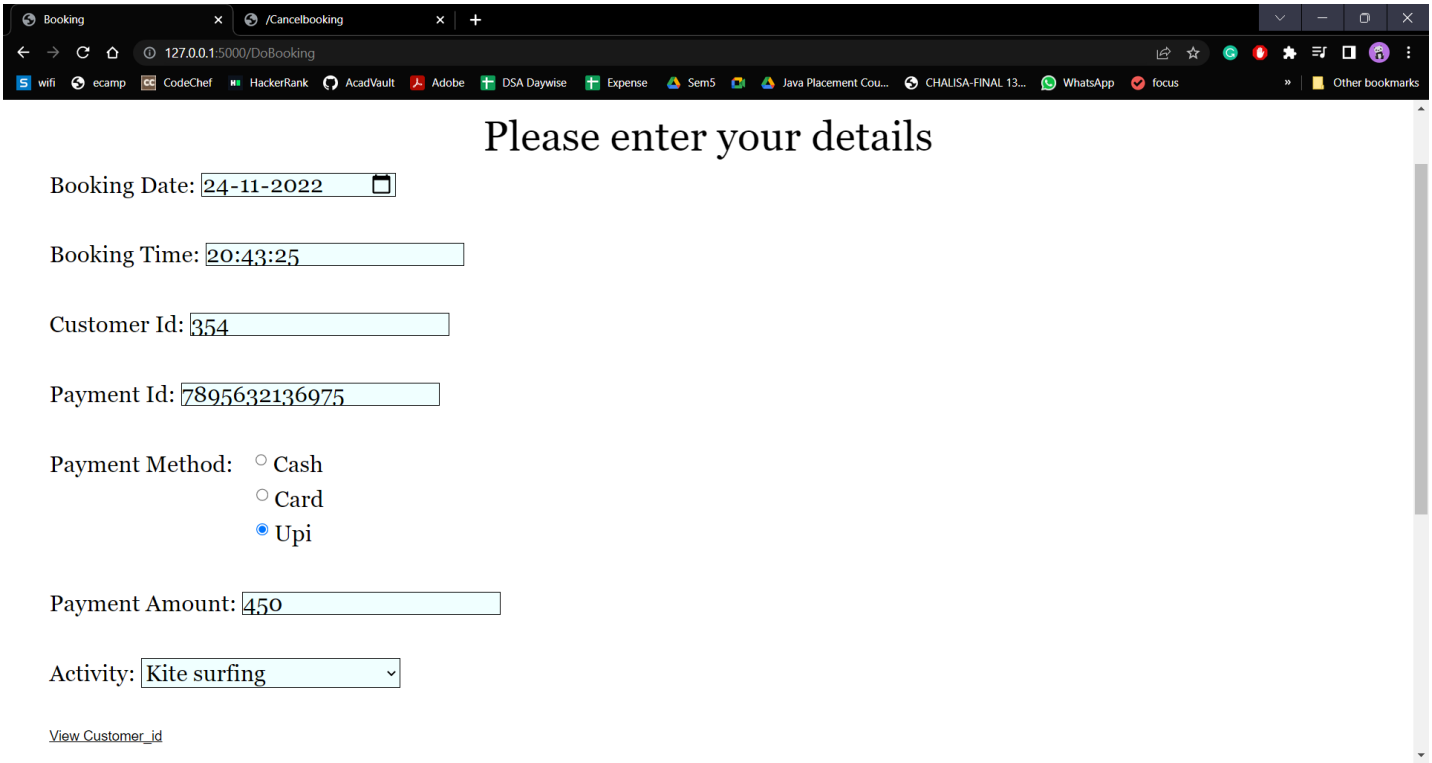
Do Booking:

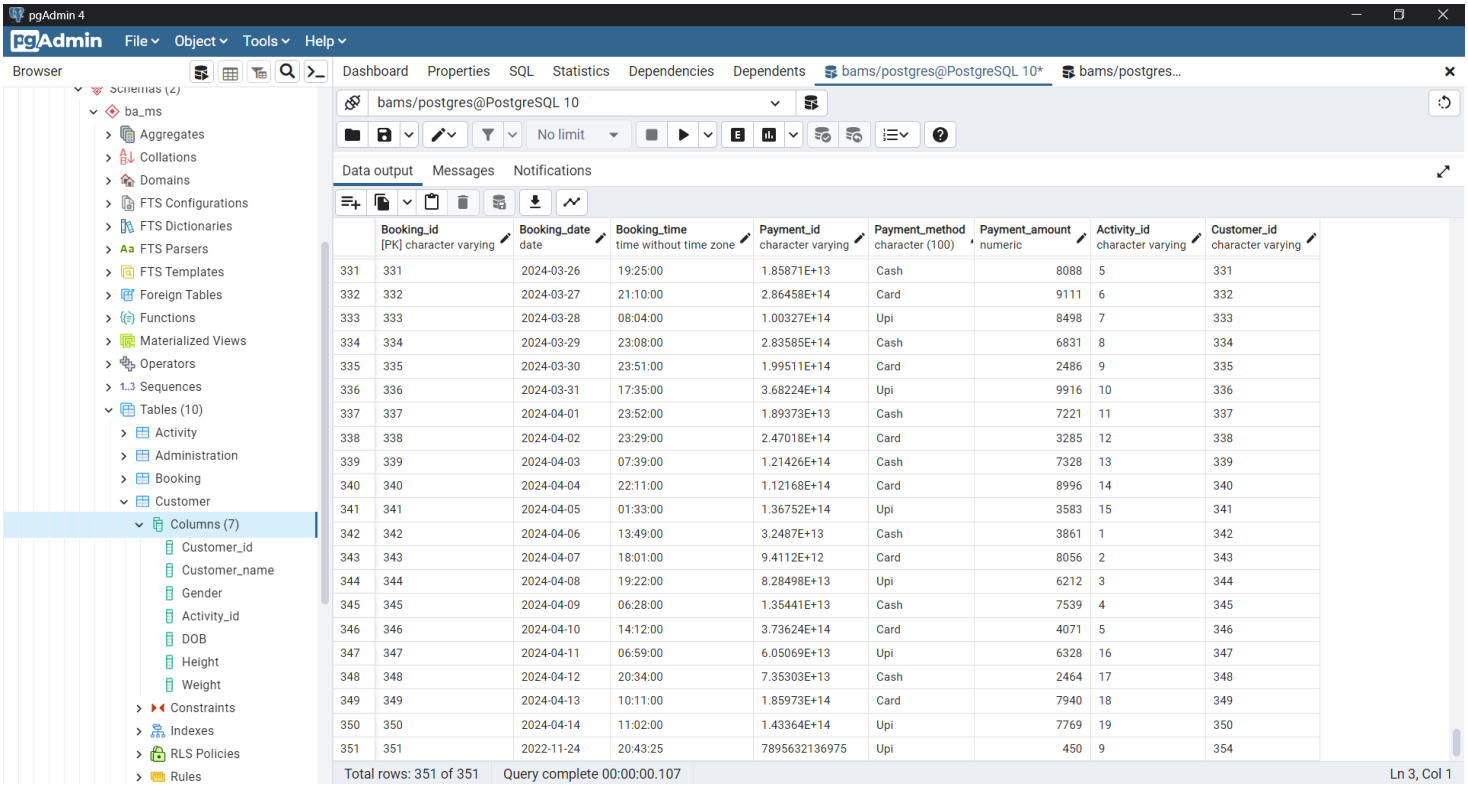
This page occurs when we select Do booking. This page helps customers to do booking. In this customer enter all the details required for booking. After submitting this data gets inserted into Booking table able booking_id is auto generated as in customer table.

```
@app.route('/DoBooking', methods=['post','get'])
def doBooking():
    print("Hello")
    if request.method == 'POST':
        Booking_date = request.form.get('Booking_date')
        Booking_time = request.form.get('Booking_time')
        Customer_id = request.form.get('Customer_id')
        Payment_id = request.form.get('Payment_id')
        Payment_method = request.form.get('Payment Method')
        Payment_amount = request.form.get('Payment_amount')
        Activity_id = request.form.get('activity_id')
        cursor.execute('select count(*) from ba_ms.\"Booking\"')
        result = cursor.fetchone()
```

```
        count = result[0]+1
        print(result)
        insert_query = """ INSERT INTO ba_ms.\"Booking\" VALUES
(%s,%s,%s,%s,%s,%s,%s,%s)"""
        try:
            record = (int(count), Booking_date, Booking_time, int(Payment_id),
Payment_method, int(Payment_amount), Activity_id, int(Customer_id))
            cursor.execute(insert_query, record)
            conn.commit()
            count = cursor.rowcount
            print(count, "Record inserted successfully into Booking table")
        except:
            return "Invalid details"
    return render_template('Booking_done.html')
```

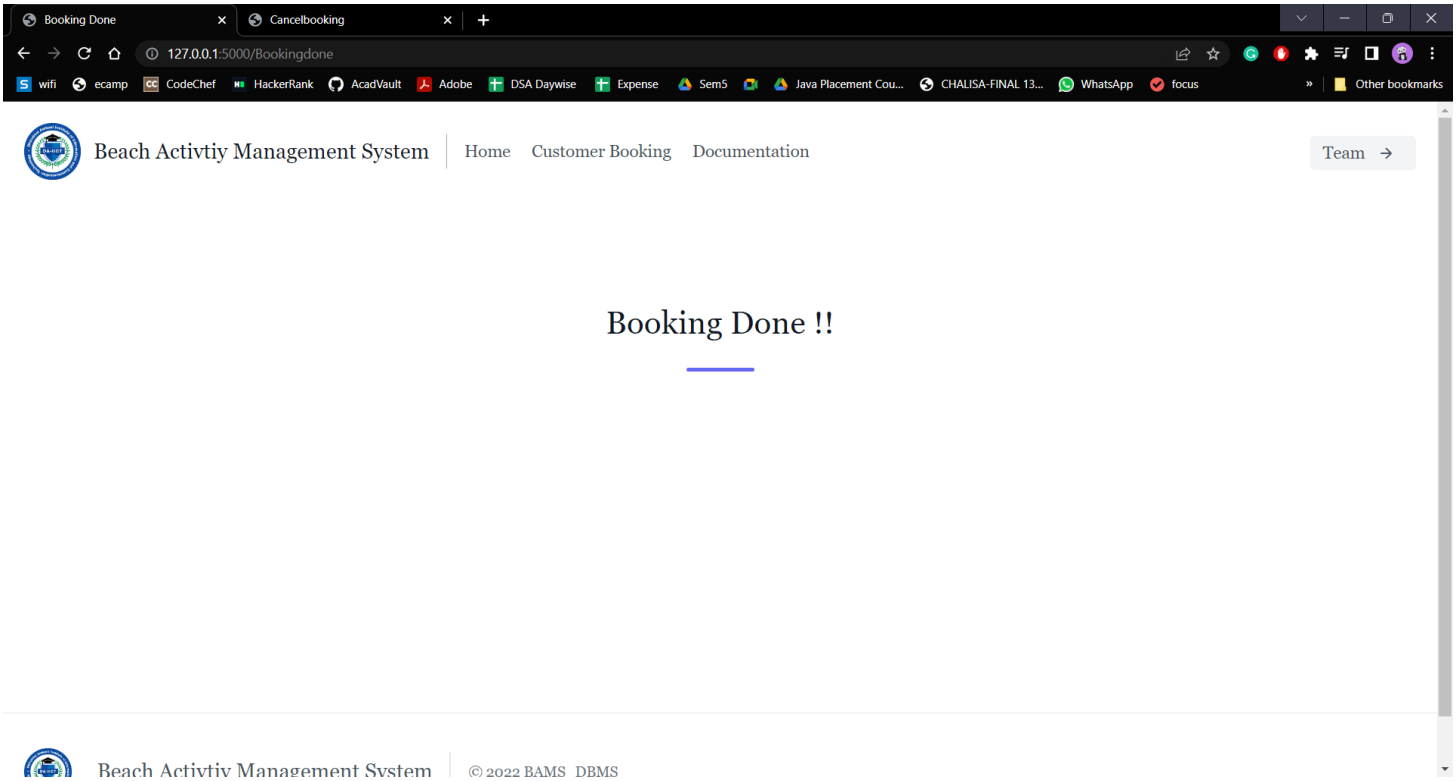
This code inserts data into Booking table entered on the website and generates automatically Booking_id.





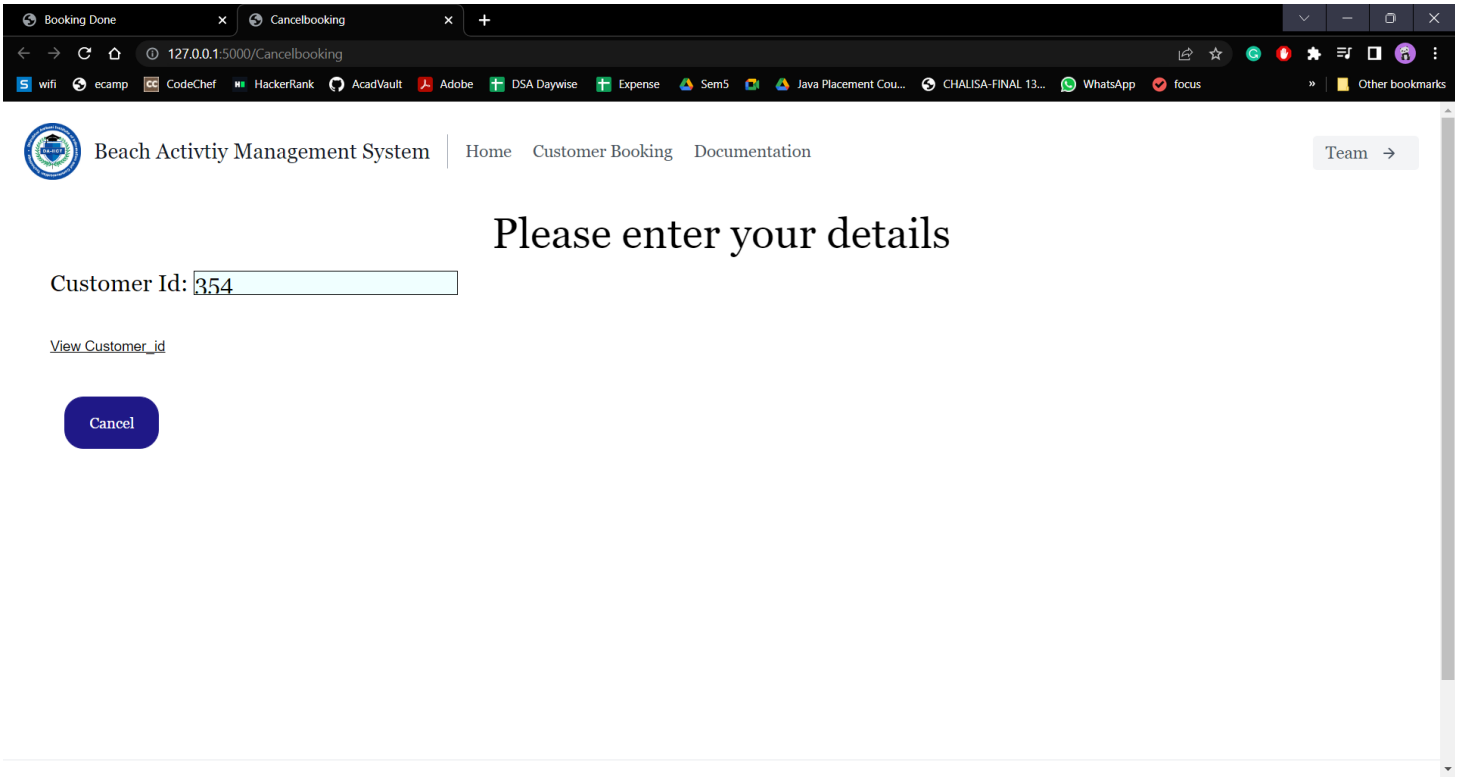
Booking_id [PK] character varying	Booking_date date	Booking_time time without time zone	Payment_id character varying	Payment_method character (100)	Payment_amount numeric	Activity_id character varying	Customer_id character varying
331	2024-03-26	19:25:00	1.85871E+13	Cash	8088	5	331
332	2024-03-27	21:10:00	2.86458E+14	Card	9111	6	332
333	2024-03-28	08:04:00	1.00327E+14	Upi	8498	7	333
334	2024-03-29	23:08:00	2.83585E+14	Cash	6831	8	334
335	2024-03-30	23:51:00	1.99511E+14	Card	2486	9	335
336	2024-03-31	17:35:00	3.68224E+14	Upi	9916	10	336
337	2024-04-01	23:52:00	1.89373E+13	Cash	7221	11	337
338	2024-04-02	23:29:00	2.47018E+14	Card	3285	12	338
339	2024-04-03	07:39:00	1.21426E+14	Cash	7328	13	339
340	2024-04-04	22:11:00	1.12168E+14	Card	8996	14	340
341	2024-04-05	01:33:00	1.36752E+14	Upi	3583	15	341
342	2024-04-06	13:49:00	3.2487E+13	Cash	3861	1	342
343	2024-04-07	18:01:00	9.4112E+12	Card	8056	2	343
344	2024-04-08	19:22:00	8.28498E+13	Upi	6212	3	344
345	2024-04-09	06:28:00	1.35441E+13	Cash	7539	4	345
346	2024-04-10	14:12:00	3.73624E+14	Card	4071	5	346
347	2024-04-11	06:59:00	6.05069E+13	Upi	6328	16	347
348	2024-04-12	20:34:00	7.35303E+13	Cash	2464	17	348
349	2024-04-13	10:11:00	1.85973E+14	Card	7940	18	349
350	2024-04-14	11:02:00	1.43364E+14	Upi	7769	19	350
351	2022-11-24	20:43:25	7895632136975	Upi	450	9	354

After submission this page comes which shows that booking is done.

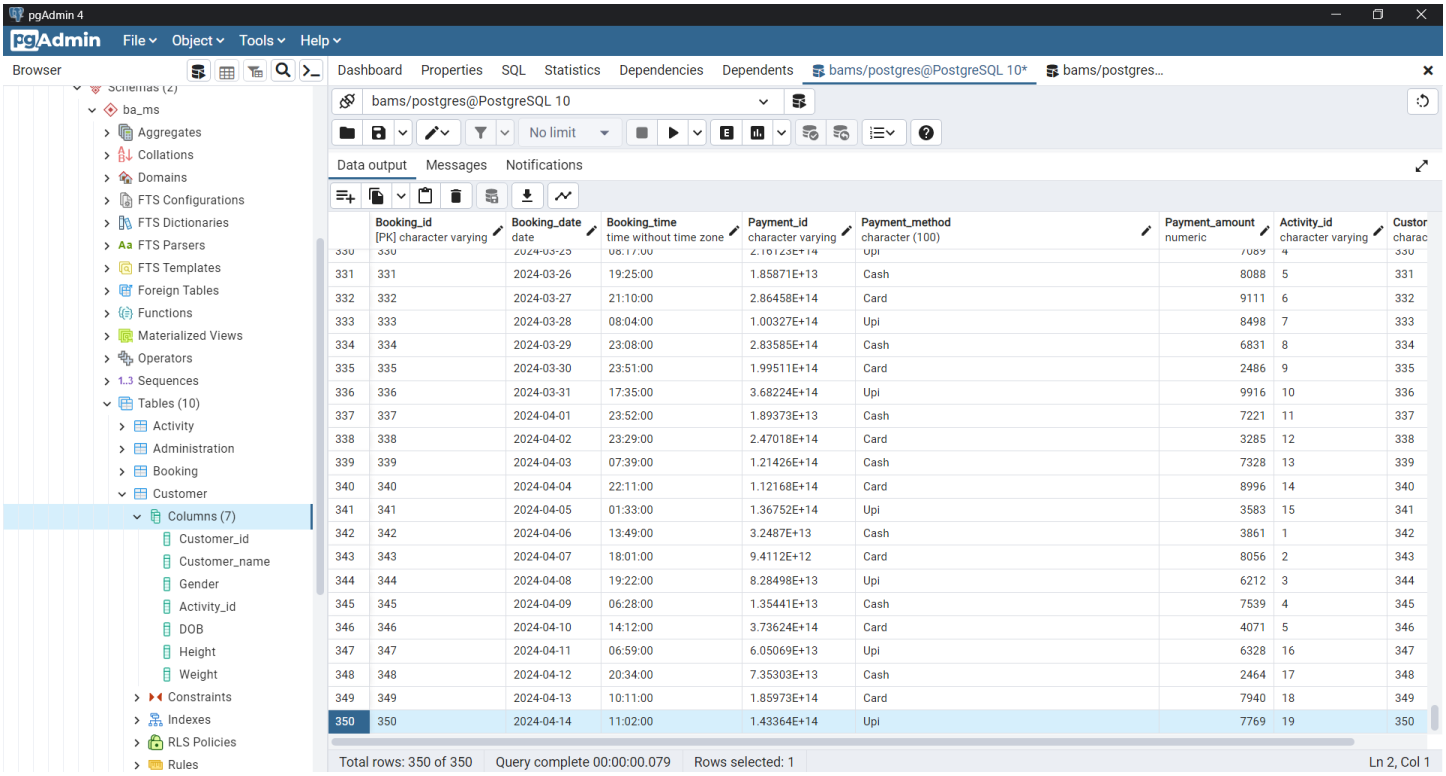


Cancel Booking:

This feature let us cancel our booking by entering customer_id in the form.




```
@app.route('/Cancel', methods=['post','get'])
def CancelBooking():
    print("Hello")
    if request.method == 'POST':
        Customer_id = request.form.get('Customer_id')
        delete_query = """ DELETE FROM ba_ms.\"Booking\" where
ba_ms.\"Booking\".
        "Customer_id\" = %s"""
        try:
            record = (Customer_id)
            cursor.execute(delete_query, record)
            conn.commit()
            count = cursor.rowcount
            print(count, "Record deleted successfully from Booking table")
        except:
            return "Invalid details"
    return redirect('/Canceldone')
```



Booking_id [PK] character varying	Booking_date date	Booking_time time without time zone	Payment_id character varying	Payment_method character (100)	Payment_amount numeric	Activity_id character varying	Custor charac
330	330	2024-03-25	08:17:00	2.16123E+14	Upi	7089	4	330
331	331	2024-03-26	19:25:00	1.85871E+13	Cash	8088	5	331
332	332	2024-03-27	21:10:00	2.86458E+14	Card	9111	6	332
333	333	2024-03-28	08:04:00	1.00327E+14	Upi	8498	7	333
334	334	2024-03-29	23:08:00	2.83585E+14	Cash	6831	8	334
335	335	2024-03-30	23:51:00	1.99511E+14	Card	2486	9	335
336	336	2024-03-31	17:35:00	3.68224E+14	Upi	9916	10	336
337	337	2024-04-01	23:52:00	1.89373E+13	Cash	7221	11	337
338	338	2024-04-02	23:29:00	2.47018E+14	Card	3285	12	338
339	339	2024-04-03	07:39:00	1.21426E+14	Cash	7328	13	339
340	340	2024-04-04	22:11:00	1.12168E+14	Card	8996	14	340
341	341	2024-04-05	01:33:00	1.36752E+14	Upi	3583	15	341
342	342	2024-04-06	13:49:00	3.2487E+13	Cash	3861	1	342
343	343	2024-04-07	18:01:00	9.4112E+12	Card	8056	2	343
344	344	2024-04-08	19:22:00	8.28498E+13	Upi	6212	3	344
345	345	2024-04-09	06:28:00	1.35441E+13	Cash	7539	4	345
346	346	2024-04-10	14:12:00	3.73624E+14	Card	4071	5	346
347	347	2024-04-11	06:59:00	6.05069E+13	Upi	6328	16	347
348	348	2024-04-12	20:34:00	7.35303E+13	Cash	2464	17	348
349	349	2024-04-13	10:11:00	1.85973E+14	Card	7940	18	349
350	350	2024-04-14	11:02:00	1.43364E+14	Upi	7769	19	350

This page shows that our booking is cancelled and deleted from the database and Booking Table.