

Indian Institute of Technology Gandhinagar



Report: Assignment 3

Authors

22110050	Birudugadda Srivibhav	20110106	Shriyash Mandavekar
22110260	Vamsi Chaturvedula	20110071	Haikoo Khandor
22110124	Sriman reddy	20110104	Madhav Kanda
22110162	Nikhilesh Myanapuri		

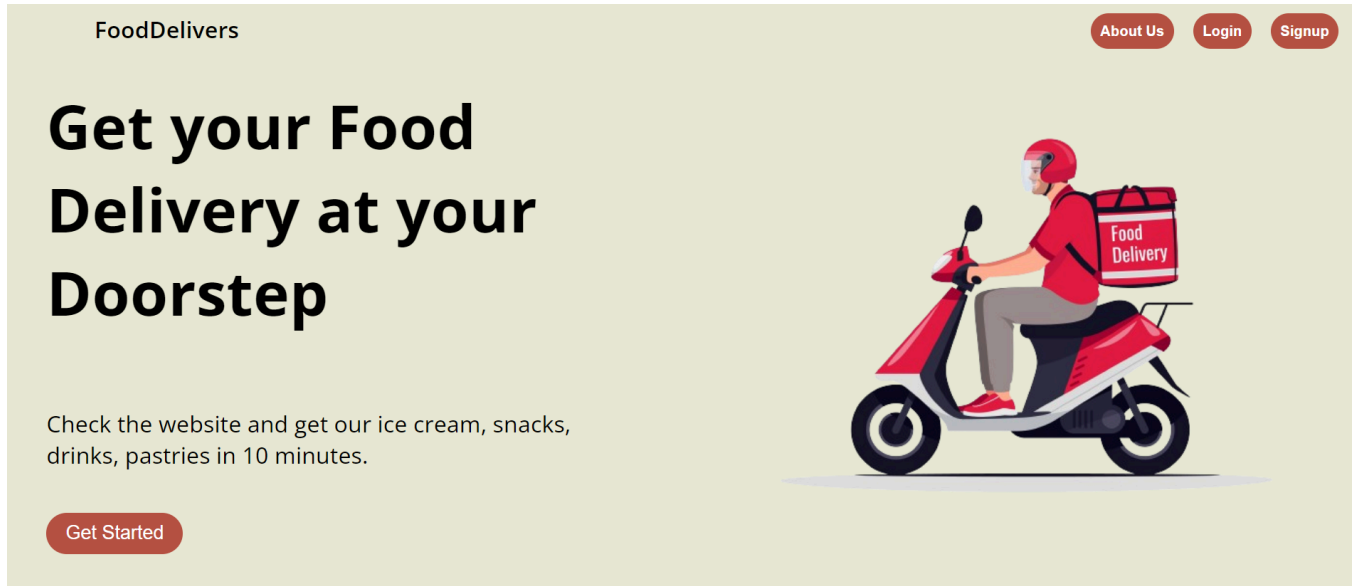
Under the Supervision of
Prof. Mayank Singh

April 4, 2024

Github repo link: https://github.com/Shriyash1234/Food_Delivery_System

Video Link: [Website Video](#)

1. Main Page:



The first page of our website includes login and signup for the people involved, such as customers, restaurants, and delivery executives. We have also added an “About Us” page to deliver specific statistics and introduce the team.

The home page for our web application looks like the one above.


```
@app.route('/')
def home():
    return render_template('home.html')
```

Coming to Login and Signup, we first talk about the Sign-up page:


The sign-up page for our system looks as shown below. The page consists of three options to either sign up as a user, restaurant or delivery agent. We could choose any of these options depending on our use case. Each button takes the user to their respective sign-up page.

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    return render_template('signup.html')
```


Sign Up



Sign Up



Sign Up



Sign Up

Already have an account [login](#)

After choosing any one of the options, We need to fill in the corresponding details. All the mentioned fields are mandatory. Each signup page is customised as per the user's authorisation. For example, if the user wants to sign up as a Delivery Agent, the page will be shown accordingly.

Delivery Agent Signup

First Name

Middle Name

Last Name

Email

dd-mm-yyyy

Vehicle Number

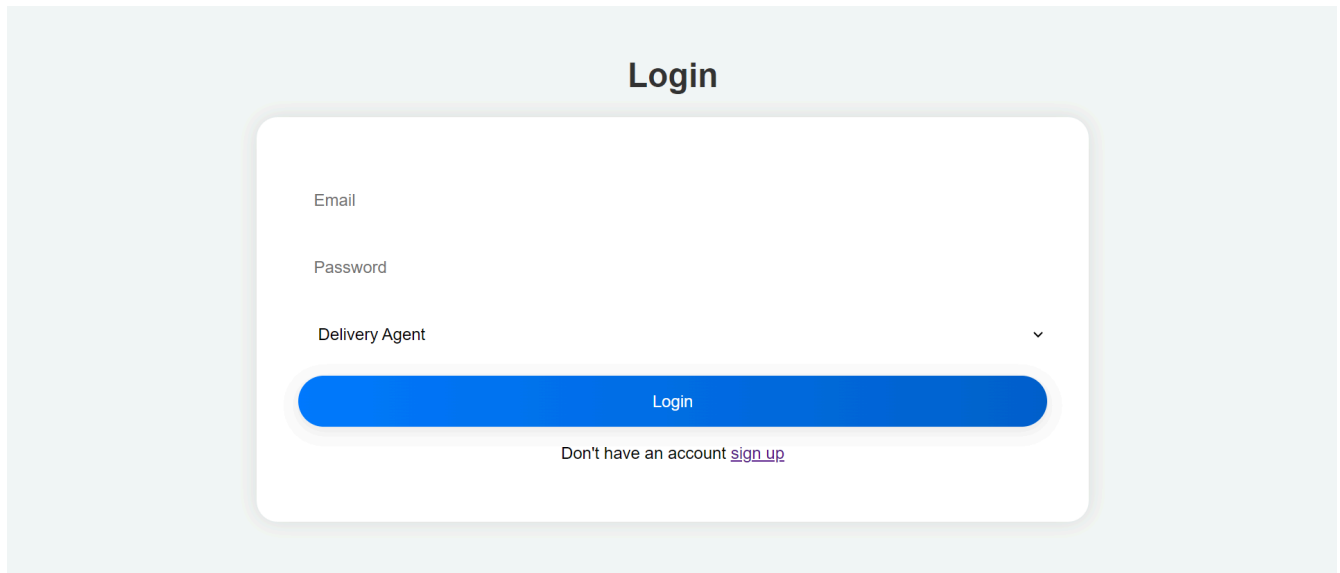
Phone Number

Password

Location

Sign Up

Now, let us talk about the Login page:



The image shows a login form titled "Login" centered on a light blue background. The form is a white rounded rectangle containing three input fields: "Email", "Password", and "Delivery Agent". The "Delivery Agent" field has a small downward arrow on its right side, indicating a pull-down menu. Below these fields is a prominent blue button with the text "Login". At the bottom of the form, there is a link that says "Don't have an account [sign up](#)".

On a single page, we keep the email, password and a pull-down menu. The concerned user will select their role, and based on that, an appropriate check is made to check if any such account with the provided password, user email and role exists or not.

If the user is a restaurant or delivery executive, they get redirected to their user details page on the condition of a successful login. If it is a customer, they get redirected to the dashboard page. In the case of a failed login, the details must be re-entered.

There are two login methods: GET and POST. Once someone logs in, details of “username”, “password”, and “authority” (customer/restaurant/delivery executive) are checked. For authority division in code, we use three boolean variables, namely customerbool, restbool and agentbool. For ex, for customers, customerbool = True, restbool == agentbool == False.

Apart from this, we also check whether the presence of *’* is for potential SQL injection attacks. The delivery executives see their user details and deliveries delivered before or in transit. The restaurants would see their basic details, the history of delivered orders and the current menu items they serve.

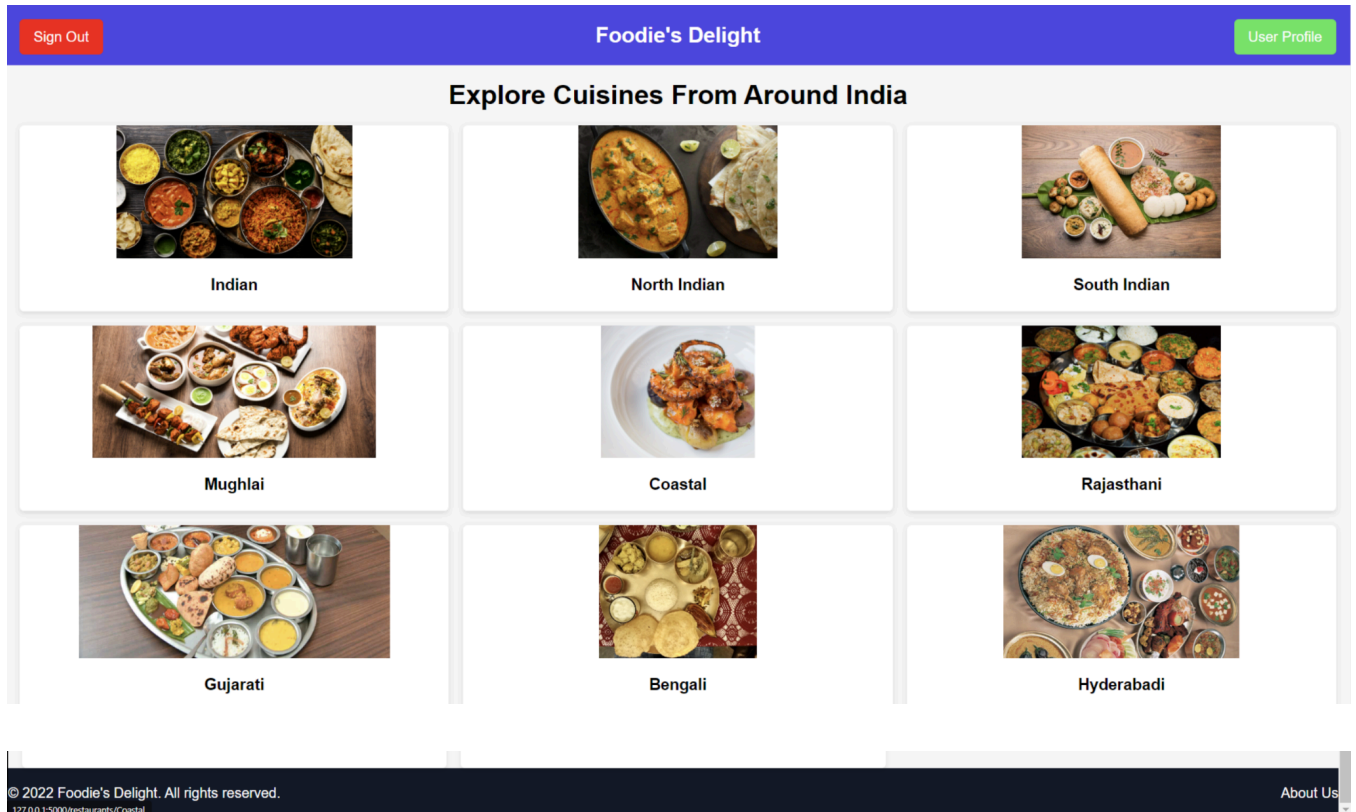
```

# login for all
@app.route('/login',methods=['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'useremail' in request.form and 'password' in request.form and 'authority' in request.form:
        useremail = request.form['useremail']
        password = request.form['password']
        authority = request.form['authority']
        session['addr_ID'] = None
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        if (authority == "Customer"):
            if("'" in useremail):
                msg = "Single Quote (') is not allowed in username field."
                flask.flash(msg)
                return redirect(url_for('login'))
            cursor.execute("SELECT * FROM Customers WHERE contact_details->>'$.email' = %s AND password = %s", (useremail, password,))
            account = cursor.fetchone()
            if account:
                session['customerbool'] = True
                session['restbool'], session['agentbool'] = False, False
                session['customer_id'] = str(account['customer_id'])
                cursor.execute("select address_ID from customer_address where customer_id=%s", (session['customer_id'],))
                addr_ID = cursor.fetchall()
                session['addr_ID'] = addr_ID[0]
                msg = 'Logged in successfully !'
                flask.flash(msg)
                return redirect(url_for('index'))
            else:
                time.sleep(2)
                msg = 'Incorrect username / password !'
        elif (authority == "Delivery Agent"):
            cursor.execute("SELECT * FROM delivery_agent WHERE email = %s AND password = %s", (useremail, password, ))
            account = cursor.fetchone()
            if account:
                session['agentbool'] = True
                session['customerbool'], session['restbool'] = False, False
                session['agent_ID'] = account['agent_id']
                msg = 'Logged in successfully !'
                flask.flash(msg)
                return redirect(url_for('index_deliveryagent'))
            else:
                time.sleep(2)
                msg = 'Incorrect username / password !'
        elif (authority == "Restaurant"):
            cursor.execute("SELECT * FROM Restaurant WHERE contact_details->>'$.email' = %s AND password = %s", (useremail, password, ))
            # cursor.execute(f"SELECT * FROM restaurant WHERE email='{useremail}' AND password='{password}'")
            account = cursor.fetchone()
            if account:
                session['restbool'] = True
                session['agentbool'], session['customerbool'] = False, False
                session['restaurant_ID'] = account['restaurant_id']
                msg = 'Logged in successfully !'
                flask.flash(msg)
                return redirect(url_for('restaurant_details'))
            else:
                time.sleep(2)
                msg = 'Incorrect username / password !'
        else:
            time.sleep(2)
            msg = 'Incorrect username / password !'
            flask.flash(msg)
    return render_template('login.html', msg = msg)

```

2. Customer Pages:

Dashboard for customers:



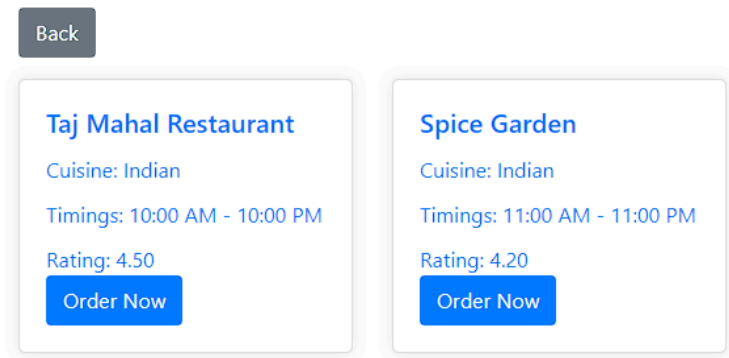
```
<h2 class="text-3xl font-bold text-center my-4">
  Explore Cuisines From Around India
</h2>
<ul class="grid grid-cols-1 md:grid-cols-3 gap-4 px-4">
  {% for cuisine in cuisines %}
  <li class="cuisine-card hover:bg-gray-100">
    <a
      href="{{ url_for('restaurants_by_cuisine', cuisine_type=cuisine.cuisine_type) }}"
      class="flex flex-col items-center justify-center"
    >
      
      <p class="cuisine-name text-lg">{{ cuisine.cuisine_type }}</p>
    </a>
  {% endfor %}
```

Each cuisine is fetched and has an image for an appealing visual appearance.

The user can select the cuisine of their choice, and restaurants are displayed inside them. Upon selecting a particular cuisine, let us say “Indian”, the following page shows up:

For the restaurants, details like Name, Cuisine type, operating times and rating are displayed. The “Back” button helps us to navigate back to the cuisine dashboard. We get redirected to the `restaurants/{cuisine_type}` page:

Restaurants



```
@app.route('/restaurants/<cuisine_type>')
def restaurants_by_cuisine(cuisine_type):
    cur = mysql.connection.cursor()

    # Execute query to fetch restaurants by cuisine type
    cur.execute('SELECT * FROM restaurant WHERE cuisine_type = %s', (cuisine_type,))
    restaurant_data = cur.fetchall()
    restaurant_columns = [col[0] for col in cur.description]
    restaurants = [dict(zip(restaurant_columns, row)) for row in restaurant_data]

    return render_template("/customers/restaurants.html", cuisine_type=cuisine_type, restaurants=restaurants)
```

[WHERE clause]

After the available restaurants are shown for a cuisine, click on “Order Now”. After clicking on “Taj Mahal Restaurant”, we get the below page. Items served by the restaurant are visible with information like Price, Rating, Type of food, Orders (number of times the food item has been ordered before), Availability (0 - not available, 1 - available right now), Item quantity (0 initially as not selected for order, here you can increase the quantity) and Notes for any special instructions to be given to the restaurant for a particular food item.

Back

Taj Mahal Restaurant

Paneer Tikka

Price: 250.00

Rating: 4.50

Type: Starter

Orders: 50

Availability: 1

Vegetarian: 0

Item Quantity: 0

Notes:



Chicken Tikka

Price: 270.00

Rating: 4.40

Type: Starter

Orders: 40

Availability: 1

Vegetarian: 0

Item Quantity: 0

Notes:



Vegetable Biryani

Price: 220.00

Rating: 4.20

Type: Main Course

Orders: 60

Availability: 1

Vegetarian: 1

Item Quantity: 0

Notes:



Dal Makhani

Price: 200.00

Rating: 4.50

Type: Main Course

Orders: 55

Availability: 1

Vegetarian: 1

Item Quantity: 0

Notes:



Gulab Jamun

Price: 80.00

Rating: 4.60

Type: Dessert

Orders: 70

Availability: 1

Vegetarian: 1

Item Quantity: 0

Notes:



Aloo Paratha

Price: 120.00

Rating: 4.30

Type: Breakfast

Orders: 45

Availability: 1

Vegetarian: 1

Item Quantity: 0

Notes:

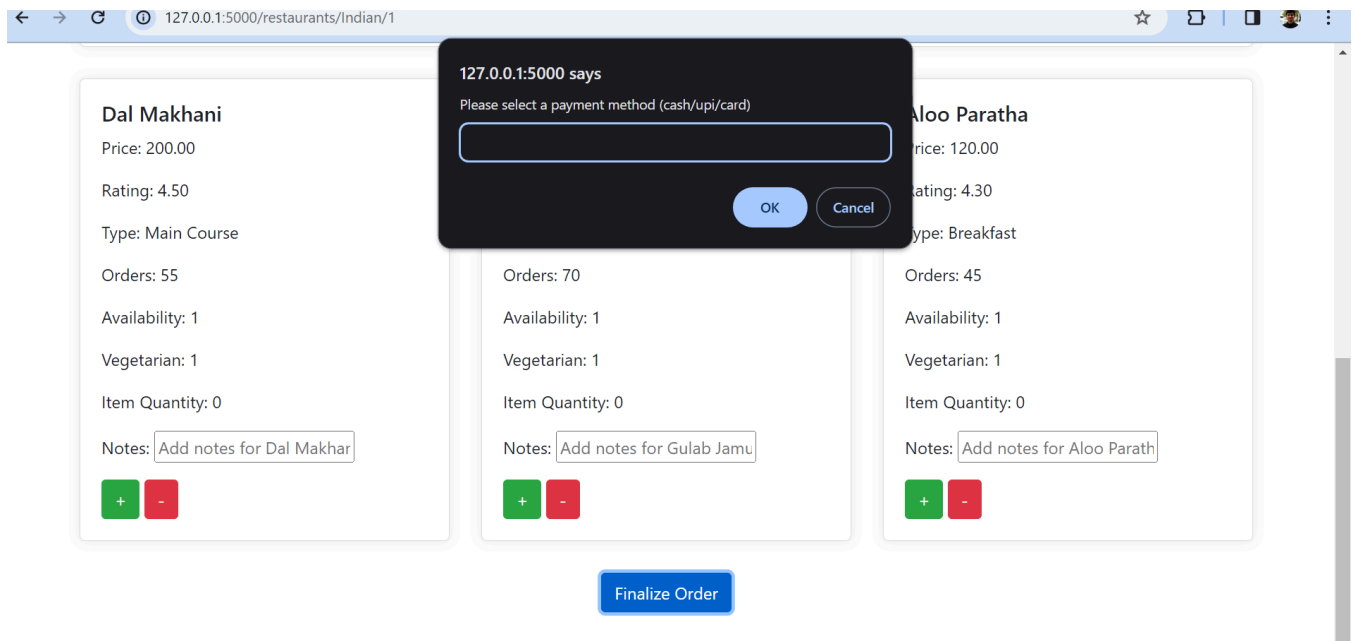


Finalize Order

For each restaurant, to display the items it serves, we use a for loop and fetch each food item and display its Price, Rating, Type, Orders, Availability and Vegetarian (1 if yes, 0 if not).

```
{% for item in restaurant_name %}
<h1 class="text-center mb-4">{{ item.restaurant_name }}</h1>
{% endfor %}
<div class="row">
  {% for food in food_items %}
    <div class="col-md-6 col-lg-4">
      <div class="menu-item">
        <h5>{{ food.item_name }}</h5>
        <p>Price: {{ food.item_price }}</p>
        <p>Rating: {{ food.item_rating }}</p>
        <p>Type: {{ food.item_type }}</p>
        <p>Orders: {{ food.order_count }}</p>
        <p>Availability: {{ food.availability }}</p>
        <p>Vegetarian: {{ food.vegetarian }}</p>
        <p>
          Item Quantity: <span id="order-count-{{ food.item_id }}">0</span>
        </p>
      </div>
    </div>
  {% endfor %}
</div>
```

There is a “Finalize Order” button, which takes you to payment after you have selected your order. On clicking the button, a dialogue box will appear stating which payment method you would like to use. We have three options, namely cash, UPI, and card.



After entering the preferred payment method, it will redirect you to the user details page, where you can check the order you just placed:

User Details

Account Details

Back

Customer ID	First Name	Middle Name	Last Name	Date of Birth	Phone Number	Email	Password
1	Rajesh	Kumar	Sharma	1980-05-10	9876543210	rajesh.sharma@example.com	password1

Address

Building Name	Street	Pin Code	City	State
Building 1	MG Road	560001	Bangalore	Karnataka

Orders

Order ID	Date	Time	Total Amount	Ordered items															
1	2024-04-04	15:42:56	850.00	<table><thead><tr><th>Food Item</th><th>Item Type</th><th>Item Price</th><th>Item Rating</th><th>Count</th></tr></thead><tbody><tr><td>Butter Chicken</td><td>Main Course</td><td>350.00</td><td>4.30</td><td>60</td></tr></tbody></table>	Food Item	Item Type	Item Price	Item Rating	Count	Butter Chicken	Main Course	350.00	4.30	60					
Food Item	Item Type	Item Price	Item Rating	Count															
Butter Chicken	Main Course	350.00	4.30	60															
16	2024-04-04	16:37:08	360.00	<table><thead><tr><th>Food Item</th><th>Item Type</th><th>Item Price</th><th>Item Rating</th><th>Count</th></tr></thead><tbody><tr><td>Dal Makhani</td><td>Main Course</td><td>200.00</td><td>4.50</td><td>56</td></tr><tr><td>Gulab Jamun</td><td>Dessert</td><td>80.00</td><td>4.60</td><td>71</td></tr></tbody></table>	Food Item	Item Type	Item Price	Item Rating	Count	Dal Makhani	Main Course	200.00	4.50	56	Gulab Jamun	Dessert	80.00	4.60	71
Food Item	Item Type	Item Price	Item Rating	Count															
Dal Makhani	Main Course	200.00	4.50	56															
Gulab Jamun	Dessert	80.00	4.60	71															

Notice how the orders (Count) update for Gulab Jamun and Dal Makhani (increased each by one). In the assignment, it was mentioned about the snapshots for each query through a dialogue box. Or directly show before and after changes to be shown. In the pic above, Dal Makhani is 55, and after ordering, Dal Makhani now has a count of 56 in the user profile. Here is the code snippet of how an order gets added to the database and assigned to a random delivery agent.

```

@app.route('/ordersummary', methods=['GET', 'POST'])
def ordersummary():
    rest_id = request.args.get('rest_id')
    payment_method = request.args.get('payment_method')
    payment_status = request.args.get('payment_status')
    ordered_items = json.loads(request.args.get('ordered_items'))
    customer_id = session["customer_id"]
    order_status = "Processing"
    placed_time = datetime.now()
    amount = 0
    for item in ordered_items:
        item_ID = item["item_id"]
        item_quantity = item["item_quantity"]
        notes = item["notes"]
        item_price = item['item_price']
        if (item_quantity != "0"):
            amount += item_price * int(item_quantity)
    cursor = mysql.connection.cursor()
    cursor.execute('select max(order_id) from Orders;')
    order_ID = cursor.fetchone()
    order_ID = str(int(order_ID[0]) + 1)
    cursor.execute('select max(payment_id) from Payment;')
    payment_ID = cursor.fetchone()
    payment_ID = str(int(payment_ID[0]) + 1)
    cursor.execute('select max(agent_id) from delivery_agent;')
    num_delivery_agents = cursor.fetchone()
    agent_id = random.randint(1, int(num_delivery_agents[0]))
    cursor.execute('insert into payment (payment_id, payment_method, payment_status, amount, time) values (%s, %s, %s, %s, %s);')
    cursor.execute('insert into orders (order_id, customer_id, restaurant_id, payment_id, order_status, placed_time, amount) values (%s, %s, %s, %s, %s, %s, %s);')
    cursor.execute('insert into delivery (order_id, agent_id, customer_id, restaurant_id, delivery_review, delivery_rating, delivery_time) values (%s, %s, %s, %s, %s, %s, %s);')
    # ordered_items is list of dictionaries where each dictionary contains item_id, item_quantity, notes, item_price
    for item in ordered_items:
        item_ID = item["item_id"]
        item_quantity = item["item_quantity"]
        notes = item["notes"]
        item_price = item['item_price']
        if (item_quantity != "0"):
            cursor.execute('update food_item set order_count = order_count + 1 where item_id = %s;', (item_ID,))
            cursor.execute('insert into ordered_items (order_id, item_id, item_quantity, item_rating, item_review, notes) values (%s, %s, %s, %s, %s, %s);')
    mysql.connection.commit()

    # cursor.execute("select name from restaurant where restaurant_ID = %s;", (str(rest_id),))
    # rest_name = cursor.fetchone()[0]
    cursor.close()
    # flash("Order successfully submitted.")
    return render_template('customers/ordersummary.html', total_price=amount, items=ordered_items, rest_name="rest_name")

```

[INSERT AND UPDATE CLAUSE]

This code collects user input regarding payment method and desired food items for ordering and then proceeds to insert this information into the corresponding tables within the database. It also shows the total amount the customer is required to pay.

3. Restaurant Pages:

The restaurant pages include the details about the restaurant. Only people logged in as a restaurant can open the corresponding pages. After the user has logged in as a customer, the website will show a home page that includes the details about the restaurant, such as restaurant_name, rating, and contact details.

The orders from the given restaurant are shown below. The order details include order_id, the names of the items in the order, their quantities, any notes given, order_status, placed_time and the total_amount.

Taj Mahal Restaurant

Cuisine: Indian

Rating: 4.50

Contact details: {"email": "tajmahal@example.com", "phone": "+91 9876543210"}

Orders

Order ID	Item Names	Item Quantities	Notes	Order Status	Placed Time	Amount
1	Paneer Tikka	2	Extra spicy please	Delivered	2024-04-04 17:30:53	850.00

Menu

Add Item

Add Item

Item Name:

Item Price:

Item Type:

Vegetarian:

☐

Availability:

☐

Add Item

The user can add an item using the add item option. The order_count will be 0 as the item has just been added to the menu.

Menu

Add Item

Paneer Tikka
Price: 250.00
Rating: 4.50
Type: Starter
Orders: 51
Availability: 1
Vegetarian: 0
Edit Menu **Delete**

Chicken Tikka
Price: 270.00
Rating: 4.40
Type: Starter
Orders: 40
Availability: 1
Vegetarian: 0
Edit Menu **Delete**

Vegetable Biryani
Price: 220.00
Rating: 4.20
Type: Main Course
Orders: 61
Availability: 1
Vegetarian: 1
Edit Menu **Delete**

The user also has the option to check and edit the menu. We have listed the menu of the restaurant after the orders. The menu contains individual items and details such as price rating, type, orders, availability, veg/non-veg. For each item, there is an option to edit the menu. Upon clicking the respective, the user will be redirected to a different page where he can edit that specific item.

Edit Menu Item

Item Name:
Paneer Tikka

Item Price:
250.00

Item Type:
Starter

Vegetarian:
☐

Availability:
☒

Submit

The user can edit the item_name, item_price, and item_type, including whether it is vegetarian or non-vegetarian and its availability. The user can not edit the rating of the item or the amount of orders placed.

Users can delete the item by clicking on the delete option provided.

In the backend:

```
@app.route('/restaurant')
def restaurant_details():
    cur = mysql.connection.cursor()
    restaurant_id = session["restaurant_ID"]
    # Fetch restaurant details
    cur.execute('''
    SELECT *
    FROM restaurant
    WHERE restaurant_id = %s
    ''', (restaurant_id,))
    restaurant_details_data = cur.fetchone()
    restaurant_details_columns = [col[0] for col in cur.description]
    restaurant_details = dict(zip(restaurant_details_columns, restaurant_details_data))

    # Fetch order details
    cur.execute('''
    SELECT
        o.order_id,
        GROUP_CONCAT(oi.item_quantity) AS item_quantities,
        GROUP_CONCAT(oi.notes) AS notes,
        GROUP_CONCAT(fi.item_name) AS item_names,
        SUM(fi.item_price) AS total_price,
        AVG(fi.item_rating) AS avg_food_rating,
        o.order_status,
        o.placed_time,
        o.amount
    FROM orders o
    JOIN ordered_items oi ON o.order_id = oi.order_id
    JOIN food_item fi ON oi.item_id = fi.item_id
```

We are using one query to collect the restaurant data from the database. Where as the second query is used to get the orders ordered by joining the food_item, orders and ordered_item tables.

```
def restaurant_details():
    order_details_columns = [col[0] for col in cur.description]
    order_details = [dict(zip(order_details_columns, row)) for row in order_details_data]

    cur.execute('''
    SELECT *
    FROM food_item
    JOIN restaurant ON food_item.restaurant_id = restaurant.restaurant_id
    WHERE restaurant.restaurant_id = %s
    ''', (restaurant_id,))
    food_item_data = cur.fetchall()
    food_item_columns = [col[0] for col in cur.description]
    food_items = [dict(zip(food_item_columns, row)) for row in food_item_data]

    return render_template("/restaurants/details.html", restaurant_details=restaurant_details, order_details=order_details)
```

[WHERE CLAUSE]

The restaurant's menu is fetched by joining the food_item and restaurant table.

For adding the item in the DB:

```

@app.route('/restaurant/add_item', methods=['GET', 'POST'])
def add_item():
    if request.method == 'POST':
        # Fetch form data
        item_name = request.form['item_name']
        item_price = float(request.form['item_price'])
        item_type = request.form['item_type']
        vegetarian = True if request.form.get('vegetarian') else False
        availability = True if request.form.get('availability') else False
        restaurant_id = session["restaurant_ID"]
        cursor = mysql.connection.cursor()
        cursor.execute('select max(item_id) from food_item;')
        item_ID = cursor.fetchone()
        item_ID = str(int(item_ID[0]) + 1)
        # order_count = 0
        # Insert food item details into the database
        cur = mysql.connection.cursor()
        cur.execute('''
            INSERT INTO food_item (item_id,item_name, item_price, item_type, vegetarian, availability, restaurant_id)
            VALUES (%s,%s, %s, %s, %s, %s, %s)
            ''', (item_ID, item_name, item_price, item_type, vegetarian, availability, restaurant_id))
        mysql.connection.commit()

        # Redirect the user back to the menu page after adding the new item
        return redirect(url_for('restaurant_details'))
    # return render_template('restaurants/details.html')
    else:
        return render_template('restaurants/add_item.html')

```

We are passing the details such as item_name, item_price, item_type, vegetarian and availability using the forms. Item_id is created by incrementing the highest item_id available. After the item is added, the user is redirected to the restaurant_details page.

For deleting the food_item from the DB:

```

@app.route('/delete_item/<item_id>')
def delete_item(item_id):
    cur = mysql.connection.cursor()
    cur.execute('''
        DELETE FROM food_item
        WHERE item_id = %s
        ''', (item_id,))
    mysql.connection.commit()
    return redirect(url_for('restaurant_details'))

```

[DELETE CLAUSE]

Food_item is deleted based on the item_id of the item.

Delivery Agent:

The third and final stakeholders are the delivery executives/agents. On logging into the application, the delivery executive would be able to see their own details.

Log Out

User Details							
ID	Name	Vehicle Number	License ID	Phone Number	Email	Location	Availability
1	Rahul Kumar	KA01AB1234	DL123456	+91 9876543210	rahul.kumar@example.com	Bangalore	1

Moreover, they can view all the deliveries made by them (Status: Delivered), ongoing (Status: On the way) and Placed (Status: Placed).

Log Out

User Details							
ID	Name	Vehicle Number	License ID	Phone Number	Email	Location	Availability
1	Rahul Kumar	KA01AB1234	DL123456	+91 9876543210	rahul.kumar@example.com	Bangalore	1

Orders										
Order ID	Restaurant Address		Customer Address		Delivery Charges	Pickup Time	Delivery/ETA Time	Status	Rating	Review
1	Building 16, Residency Road, Bangalore, Karnataka - 560025		Building 1, MG Road, Bangalore, Karnataka - 560001		5.00	2024-02-14 12:00:00	2024-02-14 12:30:00	Delivered	4	Good service
2	Building 17, Mount Road, Chennai, Tamil Nadu - 600002		Building 2, Park Street, Kolkata, West Bengal - 700001		4.50	2024-02-14 13:15:00	2024-02-14 13:45:00	Delivered	5	Prompt delivery
3	Building 18, Hill Road, Mumbai, Maharashtra - 400050		Building 3, Lalbagh Road, Bangalore, Karnataka - 560004		6.00	2024-02-14 11:45:00	2024-02-14 12:15:00	Delivered	3	Excellent service
4	Building 19, Gariahat Road, Kolkata, West Bengal - 700029		Building 4, Bandra Kurla Complex, Mumbai, Maharashtra - 400051		7.00	2024-02-14 14:30:00	2024-02-14 15:00:00	On the way	NA	N/A

```
@app.route('/delivery_dashboard', methods=['GET', 'POST'])
def index_deliveryagent():
    agent_id = session.get('agent_ID')
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM Delivery WHERE agent_id = %s", (agent_id,))
    delivery_data = cur.fetchall()
    delivery_data_columns = [col[0] for col in cur.description]
    delivery = [dict(zip(delivery_data_columns, row)) for row in delivery_data]
    for delivery_item in delivery:
        # Fetch customer address
        customer_id = delivery_item['customer_id']
        cur.execute("SELECT a.building_name, a.street, a.pin_code, a.city, a.state FROM Customer_Address ca JOIN Delivery d ON ca.customer_id = d.customer_id WHERE d.id = %s", (customer_id,))
        customer_address_data = cur.fetchone()
        if customer_address_data:
            customer_address = {
                'building_name': customer_address_data[0],
                'street': customer_address_data[1],
                'pin_code': customer_address_data[2],
                'city': customer_address_data[3],
                'state': customer_address_data[4]
            }
        else:
            customer_address = None
        delivery_item['customer_address'] = customer_address

        # Fetch restaurant details
        restaurant_id = delivery_item['restaurant_id']
```


In the backend, we have fetched the details of the delivery agent. We stored the customer_id and restaurant_id in the delivery schema. For address, we join the respective tables to address table to get the addresses of customers and restaurants.

4. About Us Page:

FoodDelivers

Back

About Us

Welcome aboard our food delivery app, where convenience meets culinary delight! Our team is driven by a passion for great food and customer satisfaction. We understand the hustle and bustle of modern life, leaving little time for cooking, which is why we're committed to bringing the best dining experiences right to your doorstep. With our user-friendly interface, ordering from your go-to restaurants has never been smoother. Whether it's a quick bite from your local diner or a special treat from a renowned chain, we've got you covered. Our diverse selection of eateries ensures there's something to please every palate. And with our efficient delivery service, you can enjoy restaurant-quality meals in the comfort of your own home. So sit back, relax, and let us take care of your cravings. With our food delivery app, delicious meals are just a few taps away.

10+

Restaurant Partners

10+

Happy Customers

10+

Delivery Executives

Team Details

email_id	first_name	last_name	roll_number
----------	------------	-----------	-------------

FoodDelivers

Back

Team Details

email_id	first_name	last_name	roll_number
haikoo.ashok@iitgn.ac.in	Haikoo	Khandor	20110071
madhav.kanda@iitgn.ac.in	Madhav	Kanda	20110104
mandavekar.shriyash@iitgn.ac.in	Shriyash	Mandavekar	20110106
birudugadda.srivibhav@iitgn.ac.in	Birudugadda	Srivibhav	22110050
kondam.reddy@iitgn.ac.in	Sriman	Reddy	22110124
rajesh.sharma@example.com	Nikhilesh	Myanapuri	22110162
vamsi.srivathsa@iitgn.ac.in	Srivathsa	Vamsi	22110260

Column to rename

email_id

Column Name

Rename Column Name

The Aboutus page gives a brief description of our food startup with some basic statistics. Finally, at the end, we mention the team members, roll numbers and email addresses.

```

@app.route('/aboutus', methods=["GET", "POST"])
def aboutus():
    if (request.method=="POST"):
        cur = mysql.connection.cursor()
        old_col_name =str( request.values.get("col_name"))
        new_name =str( request.values.get("new_name"))
        if (new_name != ""):
            sql_query = f"ALTER TABLE `team_details` RENAME COLUMN `{old_col_name}` TO `{new_name}`;"
            cur.execute(sql_query)
            mysql.connection.commit()
            cur.close()
            flask.flash("Successfully renamed the column")
        else:
            flask.flash("Please put up rename value of the column.")
    sql_query = "SELECT column_name FROM information_schema.columns WHERE table_name = %s"
    tablename = 'team_details'
    cur = mysql.connection.cursor()
    cur.execute(sql_query, ("team_details",))
    col_names = cur.fetchall()

    table = {'col1':col_names[0][0], 'col2':col_names[1][0], 'col3':col_names[2][0], 'col4':col_names[3][0],}
    sql_query = f"SELECT `{table['col1']}`, `{table['col2']}`, `{table['col3']}`, `{table['col4']}` FROM `team_details`;"
    cur.execute(sql_query)
    students = cur.fetchall()
    student_details=[]
    for student in students:
        temp = {
            'col1':student[0],
            'col2':student[1],
            'col3':student[2],
            'col4':student[3]
        }
        student_details.append(temp)

    return render_template('aboutus.html',tablename=tablename, table = table, student_details= student_details)

```

[RENAME CLAUSE]

Acknowledgements

We want to thank Prof Mayank Singh and TA Ritesh Patidar for their support during the assignment. A big thanks to all the stakeholders for their help during the interviews; their contributions were crucial to the assignment's success.

References

- Classroom slides
- [SQL Queries](#)
- [Flask](#)

Work Distribution

Team Member	Contribution
1. Birudugadda Srivibhav	Contributed to making the front end, making up the final report, pointing out bugs, and recording video.
2. Srivathsa Vamsi	Contributed in writing some of the backend functions of customer pages and fixed bugs also involved in making the final report.
3. Sriman Reddy	Making up the final report, pointing out bugs and recording video.
4. Nikhilesh Myanapuri	Contributed in writing some of the backend functions of customer pages and fixed bugs also involved in making the final report.
5. Haikoo Khandor	Designed the aboutus.html page, added basic functionalities for navigation and log out in all pages and involved in making the final report.
6. Madhav Kanda	Created delivery agent pages. Updated the database according to schema. Styled some pages in the frontend.
7. Shriyash Mandavekar	Created the landing page of the website. Built the frontend and backend part for the restaurants pages. Updated the database and fixing bugs. Final report