SNACK SQUAD : A CUSTOMIZABLE SNACK ORDERING AND DELIVERY APP

Snack Ordering :

a customizable snack ordering and delivery app like "Snack Squad" would involve several steps, including designing the user interface, backend database, and the ordering and delivery management system. Here's a simplified breakdown of the app and its components:

Features of Snack Squad:

1. **User Registration/Login**:
    a. Allows users to sign up and log in.
    b. Store user preferences and order history.
2. **Snack Catalog**:
    a. A list of snacks with images, prices, and customizable options.
3. **Customization Options**:
    a. Add toppings, choose portion sizes, or add sides.
4. **Order Placement**:
    a. Users can place orders, view their cart, and proceed to checkout.
5. **Delivery System**:
    a. Track delivery status in real time.
6. **Payment Gateway**:
    a. Secure payment options like credit cards, wallets, and UPI.
7. **Admin Panel**:
    a. Manage snacks, inventory, and track orders.

Steps to Code and Output:

1. **Frontend**:
    a. Use **React.js** or **Flutter** for the user interface.
    b. Design snack catalog, cart, and delivery tracker.
2. **Backend**:
    a. Use **Node.js** or **Django** for the backend logic.
    b. Manage user authentication, order processing, and snack database.
3. **Database**:
    a. Use **MongoDB** or **PostgreSQL** to store snack details, user data, and order history.
4. **APIs**:
    a. REST or GraphQL APIs to connect the frontend with the backend.
5. **Payment Gateway**:
    a. Integrate APIs like Stripe, PayPal, or Razorpay.

Simplified Example Code:

Backend (using Flask and SQLite)

```
python
Copy code
```

```
from flask import Flask, jsonify, request
import sqlite3

app = Flask(__name__)

# Initialize the database
def init_db():
    conn = sqlite3.connect("snack_squad.db")
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS snacks (id
INTEGER PRIMARY KEY, name TEXT, price REAL)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS orders (id
INTEGER PRIMARY KEY, snack_id INTEGER, quantity INTEGER)''')
    conn.commit()
    conn.close()

@app.route('/snacks', methods=['GET'])
def get_snacks():
    conn = sqlite3.connect("snack_squad.db")
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM snacks")
    snacks = cursor.fetchall()
    conn.close()
    return jsonify(snacks)

@app.route('/order', methods=['POST'])
def place_order():
    data = request.json
    snack_id = data['snack_id']
    quantity = data['quantity']
    conn = sqlite3.connect("snack_squad.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO orders (snack_id, quantity)
VALUES (?, ?)", (snack_id, quantity))
```

```
    conn.commit()
    conn.close()
    return jsonify({"message": "Order placed
successfully!"})

if __name__ == '__main__':
    init_db()
    app.run(debug=True)
```

Frontend (HTML + Fetch API)

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Snack Squad</title>
</head>
<body>
    <h1>Welcome to Snack Squad</h1>
    <div id="snack-list"></div>
    <button onclick="orderSnack(1, 2)">Order Snack
1</button>

    <script>
        // Fetch snacks from the backend
        fetch('/snacks')
            .then(response => response.json())
            .then(data => {
                let snackList =
document.getElementById('snack-list');
                data.forEach(snack => {
                    let div = document.createElement('div');
```

```
                    div.textContent = `${snack[1]} -
$${snack[2]}`;
                    snackList.appendChild(div);
                });
            });

        // Place an order
        function orderSnack(snack_id, quantity) {
            fetch('/order', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ snack_id, quantity })
            })
            .then(response => response.json())
            .then(data => alert(data.message));
        }
    </script>
</body>
</html>
```

Output

1. **Homepage**:
    a. Displays a list of available snacks (retrieved from the backend).
2. **Order Button**:
    a. Allows users to place orders for specific snacks.
3. **Backend Logs**:
    a. Displays messages when orders are placed.