**Ice Cream Parlor Web Application**

**Done By**

**SANJAY KARTHICK V**

**23MX224-MCA-G1**

**1. Project Overview**

This project is a web-based ice cream parlor application built using Flask. It enables users to view available ice cream flavors, add them to their cart, search for specific flavors, and manage allergens.

**2.Files Description**

1. app.py

* **Purpose**: Main Flask application handling routing and backend logic.
* **Key Functions**:
  + home(): Fetches and displays all flavors from the database.
  + add\_to\_cart(flavor\_id): Adds a selected flavor to the cart.
  + cart(): Displays items in the cart.
  + add\_allergen(): Adds allergens to the database.
  + search(): Searches for flavors matching a query.

2. database.py

* **Purpose**: Sets up the SQLite database schema and provides helper functions.
* **Key Functions**:
  + create\_tables(): Creates necessary tables and populates default data.
  + get\_connection(): Establishes a connection to the database.

3. cart.html

* **Purpose**: Displays items added to the user's cart.
* **Key Features**:
  + Iterates over cart\_items to list each item.
  + Provides a link to return to the homepage.

4. flavors.html

* **Purpose**: Displays search results for flavors.
* **Key Features**:
  + Iterates over flavors to list matched results.
  + Provides a link to return to the homepage.

5. index.html

* **Purpose**: Serves as the homepage to display available flavors and perform actions.
* **Key Features**:
  + Lists all flavors with an option to add them to the cart.
  + Provides a search form for finding specific flavors.
  + Allows users to add allergens.

**3.Work Flow**

1. When the user visits the homepage (/), the home() function retrieves flavors from the database and renders index.html.
2. Users can add items to the cart by clicking the "Add to Cart" button, triggering the add\_to\_cart(flavor\_id) route.
3. The cart() function displays the contents of the cart via cart.html.
4. Users can search for flavors using the search bar, which calls the search() function and renders results in flavors.html.
5. Allergens are managed via the add\_allergen() function, which updates the database

**4.Tables in Database**

1. **flavors**:
   * Stores ice cream flavors, their descriptions.
2. **cart**:
   * Tracks items added to the cart.
3. **allergens**:
   * Maintains a list of allergens to account for user preferences.

**5.Future Improvements**

* Add user authentication.
* Allow dynamic cart quantities.
* Enhance the search feature with filtering.
* Include detailed allergen warnings.

1. **Steps in the ReadMe.md file to validate the application**

irm get.scoop.sh | iex

-It is used to install the Scoop package manager on Windows for easier software installation.

scoop install python

-It installs Python via Scoop.

python --version

-To check the version of python and also to verify whether python is installed correctly.

python database.py

-It executes the database.py script to initialize the database.

pip install flask

-It installs the Flask web framework using Python's package manager, pip.

pip show flask

-Displays detailed information about the Flask installation, including its version and location.

sqlite3 parlor.db

-Opens the SQLite database file parlor.db for direct interaction via the SQLite command-line interface.

python app.py

-To Run the Application.

**SQL Queries Implementation**

**a)in app.py**

@app.route('/')

def home():

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM flavors")

flavors = cursor.fetchall()

conn.close()

return render\_template('index.html', flavors=flavors)

@app.route('/add\_to\_cart/<int:flavor\_id>')

def add\_to\_cart(flavor\_id):

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("INSERT INTO cart (flavor\_id) VALUES (?)", (flavor\_id,))

conn.commit()

conn.close()

return redirect(url\_for('cart'))

@app.route('/cart')

def cart():

conn = get\_connection()

cursor = conn.cursor()

cursor.execute('''

SELECT flavors.name

FROM cart

JOIN flavors ON cart.flavor\_id = flavors.id

''')

**FETCH CART ITEMS WITH FLAVOR NAMES**

cart\_items = cursor.fetchall()

conn.close()

return render\_template('cart.html', cart\_items=cart\_items)

@app.route('/add\_allergen', methods=['POST'])

def add\_allergen():

allergen\_name = request.form.get('allergen')

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("INSERT OR IGNORE INTO allergens (name) VALUES (?)", (allergen\_name,))

conn.commit()

conn.close()

return redirect('/')

**TO Search Flavors by Name**

@app.route('/search')

def search():

query = request.args.get('query')

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM flavors WHERE name LIKE ?", (f'%{query}%',))

results = cursor.fetchall()

conn.close()

return render\_template('flavors.html', flavors=results)

b) **in database.py**

def create\_tables():

conn = sqlite3.connect('parlor.db')

cursor = conn.cursor()

**TO CREATE TABLES**

cursor.execute('''

CREATE TABLE IF NOT EXISTS flavors (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

description TEXT,

is\_seasonal INTEGER DEFAULT 0

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS ingredients (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

quantity INTEGER DEFAULT 0

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS allergens (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS cart (

id INTEGER PRIMARY KEY AUTOINCREMENT,

flavor\_id INTEGER,

FOREIGN KEY (flavor\_id) REFERENCES flavors (id)

)

''')

**To Insert Default Allergens into allergens Table.**

cursor.execute("INSERT OR IGNORE INTO allergens (name) VALUES ('Nuts'), ('Dairy'), ('Soy')")

**To Insert default Flavor into flavors table**

cursor.executemany(

"INSERT OR IGNORE INTO flavors (name, description, is\_seasonal) VALUES (?, ?, ?)",

[

('Vanilla', 'Classic Vanilla', 0),

('Pumpkin Spice', 'Seasonal Pumpkin flavor', 1)

]

)

conn.commit()

conn.close()