**PSG College of Technology**

**Akshayakumar N.G**

**21z206 ( BE CSE)**

**In this documentation, we have explained each part of the code, detailing how the function works, from the database connection to the execution of the main function.**

**Database Setup and Connection:**

**1.db\_connection()**

**Code:**

def db\_connection():

conn = sqlite3.connect(DB\_NAME)

conn.row\_factory = sqlite3.Row

return conn

**One-line description :** Establishes a connection to the SQLite database and returns the connection object.

**2. initialize\_db()**

**Code:**

def initialize\_db():

conn = db\_connection()

cursor = conn.cursor()

cursor.executescript("""

CREATE TABLE IF NOT EXISTS seasonal\_flavors (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

price REAL NOT NULL,

available INTEGER DEFAULT 1,

season TEXT

);

...

""")

conn.commit()

conn.close()

**One-line description :** Initializes the database by creating necessary tables if they do not exist.

**Flavor Management**

**3. add\_flavor()**

**Code:**

def add\_flavor(name, description, price, season):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("""

INSERT INTO seasonal\_flavors (name, description, price, season)

VALUES (?, ?, ?, ?)

""", (name, description, price, season))

conn.commit()

conn.close()

print(f"Flavor '{name}' added successfully.")

**One-line description :** Adds a new flavor with details to the seasonal\_flavors table.

**4. search\_flavors():**

**Code:**

def search\_flavors(search\_term=None, filter\_season=None):

conn = db\_connection()

cursor = conn.cursor()

query = "SELECT \* FROM seasonal\_flavors WHERE 1"

params = []

if search\_term:

query += " AND name LIKE ?"

params.append(f"%{search\_term}%")

if filter\_season:

query += " AND season = ?"

params.append(filter\_season)

cursor.execute(query, params)

rows = cursor.fetchall()

**One-line description :** Searches for flavors by name or filters them by season.

**5.Ingredient Management**

**add\_ingredient()**

**Code:**

def add\_ingredient(name, quantity, unit):

conn = db\_connection()

cursor = conn.cursor()

try:

cursor.execute("""

INSERT INTO ingredients (name, quantity, unit)

VALUES (?, ?, ?)

""", (name, quantity, unit))

conn.commit()

print(f"Ingredient '{name}' added successfully.")

except sqlite3.IntegrityError:

print(f"Ingredient '{name}' already exists.")

conn.close()

**One-line description :** Adds a new ingredient and ensures uniqueness.

**6.update\_ingredient\_quantity()**

**Code:**

def update\_ingredient\_quantity(name, quantity, operation):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT quantity FROM ingredients WHERE name = ?", (name,))

row = cursor.fetchone()

if row:

current\_quantity = row['quantity']

if operation == 'add':

new\_quantity = current\_quantity + quantity

elif operation == 'subtract':

if current\_quantity >= quantity:

new\_quantity = current\_quantity - quantity

**One-line description :** Updates the quantity of an ingredient based on the operation (add or subtract).

**7. Customer Suggestions**

**add\_customer\_suggestion()**

**Code:**

def add\_customer\_suggestion(customer\_name, flavor\_name, allergy\_concerns=None):

suggestion\_date = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("""

INSERT INTO customer\_suggestions (customer\_name, flavor\_name, suggestion\_date, allergy\_concerns)

VALUES (?, ?, ?, ?)

""", (customer\_name, flavor\_name, suggestion\_date, allergy\_concerns))

conn.commit()

conn.close()

print(f"Suggestion from {customer\_name} for flavor '{flavor\_name}' added successfully.")

**One-line description :** Records a customer's flavor suggestion with optional allergy concerns.

**8.view\_customer\_suggestions()**

**Code:**

def view\_customer\_suggestions():

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM customer\_suggestions")

rows = cursor.fetchall()

...

**One-line description :** Displays all recorded customer suggestions.

**9.Allergen Management**

**add\_allergen()**

**Code:**

def add\_allergen(name):

conn = db\_connection()

cursor = conn.cursor()

try:

cursor.execute("""

INSERT INTO allergens (name)

VALUES (?)

""", (name,))

conn.commit()

print(f"Allergen '{name}' added successfully.")

except sqlite3.IntegrityError:

print(f"Allergen '{name}' already exists.")

conn.close()

**One-line description :** Adds a new allergen and ensures uniqueness.

**10.view\_allergens()**

**Code:**

def view\_allergens():

conn = db\_connection()

cursor = conn.cursor()

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

rows = cursor.fetchall()

...

**One-line description :** Displays a list of allergens.

**Shopping Cart Management**

**11. add\_to\_cart()**

**Code:**

def add\_to\_cart():

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM seasonal\_flavors WHERE available = 1")

rows = cursor.fetchall()

if rows:

flavor\_id = int(input("Enter the Flavor ID you want to add to the cart: "))

quantity = int(input("Enter the quantity: "))

...

**One-line description :** Prompts the user to add a flavor to the cart.

**12. view\_cart()**

**Code:**

def view\_cart():

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("""

SELECT c.id, s.name, c.quantity, s.price, (c.quantity \* s.price) AS total\_price

FROM cart c

JOIN seasonal\_flavors s ON c.flavor\_id = s.id

""")

rows = cursor.fetchall()

...

**One-line description :** Displays cart details with a grand total.

**13.remove\_from\_cart()**

**Code:**

def remove\_from\_cart(cart\_id):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("DELETE FROM cart WHERE id = ?", (cart\_id,))

conn.commit()

...

**One-line description :** Removes an item from the cart using its ID.

**Main Menu**

**14.main\_menu()**

def main\_menu():

print("\nWelcome to the Ice Cream Parlor Management System!")

user\_role = input("Are you an 'Owner' or a 'Customer'? ").strip().lower()

if user\_role not in ['owner', 'customer']:

print("Invalid role. Please restart and choose either 'Owner' or 'Customer'.")

return

...

**One-line description :**The main entry point, displaying role-specific options based on whether the user is an owner or a customer.

**Full Code:**

import sqlite3

import datetime

DB\_NAME = "ice\_cream\_parlor.db"

# Function to establish a connection to the database

def db\_connection():

conn = sqlite3.connect(DB\_NAME)

conn.row\_factory = sqlite3.Row # Allows access by column name instead of index

return conn

# Function to initialize the database with tables

def initialize\_db():

conn = db\_connection()

cursor = conn.cursor()

# Creating tables for seasonal\_flavors, ingredients, customer\_suggestions, allergens, and cart

cursor.executescript("""

CREATE TABLE IF NOT EXISTS seasonal\_flavors (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

price REAL NOT NULL,

available INTEGER DEFAULT 1,

season TEXT

);

CREATE TABLE IF NOT EXISTS ingredients (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

quantity REAL NOT NULL,

unit TEXT NOT NULL

);

CREATE TABLE IF NOT EXISTS customer\_suggestions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

customer\_name TEXT NOT NULL,

flavor\_name TEXT NOT NULL,

suggestion\_date TEXT NOT NULL,

allergy\_concerns TEXT

);

CREATE TABLE IF NOT EXISTS allergens (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE

);

CREATE TABLE IF NOT EXISTS cart (

id INTEGER PRIMARY KEY AUTOINCREMENT,

flavor\_id INTEGER NOT NULL,

quantity INTEGER NOT NULL,

FOREIGN KEY (flavor\_id) REFERENCES seasonal\_flavors (id)

);

""")

conn.commit()

conn.close()

initialize\_db()

# Function to add a new flavor to the seasonal\_flavors table

def add\_flavor(name, description, price, season):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("""

INSERT INTO seasonal\_flavors (name, description, price, season)

VALUES (?, ?, ?, ?)

""", (name, description, price, season))

conn.commit()

conn.close()

print(f"Flavor '{name}' added successfully.")

# Function to search and filter flavors based on search term and season

def search\_flavors(search\_term=None, filter\_season=None):

conn = db\_connection()

cursor = conn.cursor()

query = "SELECT \* FROM seasonal\_flavors WHERE 1"

params = []

if search\_term:

query += " AND name LIKE ?"

params.append(f"%{search\_term}%")

if filter\_season:

query += " AND season = ?"

params.append(filter\_season)

cursor.execute(query, params)

rows = cursor.fetchall()

if rows:

for row in rows:

print(f"Name: {row['name']}, Description: {row['description']}, Price: {row['price']}, Season: {row['season']}")

else:

print("No flavors found.")

conn.close()

# Function to add a new ingredient to the ingredients table

def add\_ingredient(name, quantity, unit):

conn = db\_connection()

cursor = conn.cursor()

try:

cursor.execute("""

INSERT INTO ingredients (name, quantity, unit)

VALUES (?, ?, ?)

""", (name, quantity, unit))

conn.commit()

print(f"Ingredient '{name}' added successfully.")

except sqlite3.IntegrityError:

print(f"Ingredient '{name}' already exists.")

conn.close()

# Function to add a customer suggestion to the customer\_suggestions table

def add\_customer\_suggestion(customer\_name, flavor\_name, allergy\_concerns=None):

suggestion\_date = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("""

INSERT INTO customer\_suggestions (customer\_name, flavor\_name, suggestion\_date, allergy\_concerns)

VALUES (?, ?, ?, ?)

""", (customer\_name, flavor\_name, suggestion\_date, allergy\_concerns))

conn.commit()

conn.close()

print(f"Suggestion from {customer\_name} for flavor '{flavor\_name}' added successfully.")

# Function to view all customer suggestions

def view\_customer\_suggestions():

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM customer\_suggestions")

rows = cursor.fetchall()

if rows:

for row in rows:

print(f"Customer: {row['customer\_name']}, Flavor: {row['flavor\_name']}, Date: {row['suggestion\_date']}, Allergy Concerns: {row['allergy\_concerns']}")

else:

print("No customer suggestions found.")

conn.close()

# Function to add an allergen to the allergens table

def add\_allergen(name):

conn = db\_connection()

cursor = conn.cursor()

try:

cursor.execute("""

INSERT INTO allergens (name)

VALUES (?)

""", (name,))

conn.commit()

print(f"Allergen '{name}' added successfully.")

except sqlite3.IntegrityError:

print(f"Allergen '{name}' already exists.")

conn.close()

# Function to view all allergens in the allergens table

def view\_allergens():

conn = db\_connection()

cursor = conn.cursor()

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

rows = cursor.fetchall()

if rows:

for row in rows:

print(f"Allergen: {row['name']}")

else:

print("No allergens found.")

conn.close()

# Function to add a flavor to the cart

def add\_to\_cart():

conn = db\_connection()

cursor = conn.cursor()

# Display all available flavors with details

cursor.execute("SELECT \* FROM seasonal\_flavors WHERE available = 1")

rows = cursor.fetchall()

if rows:

print("\nAvailable Flavors:")

for row in rows:

print(f"ID: {row['id']}\nFlavor: {row['name']}\nDescription: {row['description']}\nPrice: ${row['price']:.2f}\n")

# Prompt the user for flavor ID and quantity

flavor\_id = int(input("Enter the Flavor ID you want to add to the cart: "))

quantity = int(input("Enter the quantity: "))

try:

# Verify the selected flavor exists and is available

cursor.execute("SELECT \* FROM seasonal\_flavors WHERE id = ? AND available = 1", (flavor\_id,))

flavor = cursor.fetchone()

if flavor:

# Add the flavor to the cart

cursor.execute("""

INSERT INTO cart (flavor\_id, quantity)

VALUES (?, ?)

""", (flavor\_id, quantity))

conn.commit()

print(f"Added {quantity} of '{flavor['name']}' to the cart.")

else:

print("Invalid Flavor ID or flavor is not available.")

except ValueError:

print("Invalid input. Please enter valid numbers.")

else:

print("No flavors are currently available.")

conn.close()

# Function to view the cart with total cost

def view\_cart():

conn = db\_connection()

cursor = conn.cursor()

# Query to get cart details including price and quantity

cursor.execute("""

SELECT c.id, s.name, c.quantity, s.price, (c.quantity \* s.price) AS total\_price

FROM cart c

JOIN seasonal\_flavors s ON c.flavor\_id = s.id

""")

rows = cursor.fetchall()

if rows:

grand\_total = 0 # To keep track of the total cost of the cart

print("\nCart Details:")

for row in rows:

print(f"Cart ID: {row['id']}, Flavor: {row['name']}, Quantity: {row['quantity']}, Price per unit: ${row['price']:.2f}, Total: ${row['total\_price']:.2f}")

grand\_total += row['total\_price'] # Accumulate the total price

print(f"\nGrand Total: ${grand\_total:.2f}")

else:

print("Your cart is empty.")

conn.close()

# Function to remove an item from the cart by Cart ID

def remove\_from\_cart(cart\_id):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("DELETE FROM cart WHERE id = ?", (cart\_id,))

conn.commit()

if cursor.rowcount > 0:

print(f"Item with Cart ID {cart\_id} removed from cart.")

else:

print(f"Item with Cart ID {cart\_id} not found.")

conn.close()

# Function to update the ingredient quantity (add or subtract)

def update\_ingredient\_quantity(name, quantity, operation):

conn = db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT quantity FROM ingredients WHERE name = ?", (name,))

row = cursor.fetchone()

if row:

current\_quantity = row['quantity']

if operation == 'add':

new\_quantity = current\_quantity + quantity

elif operation == 'subtract':

if current\_quantity >= quantity:

new\_quantity = current\_quantity - quantity

else:

print(f"Cannot subtract {quantity} from {name}, insufficient stock.")

conn.close()

return

else:

print("Invalid operation. Use 'add' or 'subtract'.")

conn.close()

return

cursor.execute("""

UPDATE ingredients

SET quantity = ?

WHERE name = ?

""", (new\_quantity, name))

conn.commit()

print(f"Updated '{name}' quantity to {new\_quantity}.")

else:

print(f"Ingredient '{name}' does not exist in the database.")

conn.close()

# Function to display the main menu and allow user to select options based on their role

def main\_menu():

print("\nWelcome to the Ice Cream Parlor Management System!")

user\_role = input("Are you an 'Owner' or a 'Customer'? ").strip().lower()

if user\_role not in ['owner', 'customer']:

print("Invalid role. Please restart and choose either 'Owner' or 'Customer'.")

return

while True:

print("\nMain Menu:")

if user\_role == 'owner':

print("1. Add New Flavor")

print("2. Search & Filter Flavors")

print("3. Add Ingredient")

print("4. Update Ingredient Quantity")

print("5. View Customer Suggestions")

print("6. Add Allergen")

print("7. View Allergen List")

print("8. Exit")

choice = input("Enter your choice (1-8): ").strip()

if choice == '1':

name = input("Enter flavor name: ").strip()

description = input("Enter flavor description: ").strip()

price = float(input("Enter price: ").strip())

season = input("Enter season: ").strip()

add\_flavor(name, description, price, season)

elif choice == '2':

search\_term = input("Enter search term (or press Enter to skip): ").strip()

filter\_season = input("Enter season to filter by (or press Enter to skip): ").strip()

search\_flavors(search\_term, filter\_season)

elif choice == '3':

name = input("Enter ingredient name: ").strip()

quantity = float(input("Enter quantity: ").strip())

unit = input("Enter unit (e.g., kg, liters): ").strip()

add\_ingredient(name, quantity, unit)

elif choice == '4':

name = input("Enter ingredient name to update: ").strip()

quantity = float(input("Enter quantity to add/subtract: ").strip())

operation = input("Enter operation ('add' or 'subtract'): ").strip().lower()

update\_ingredient\_quantity(name, quantity, operation)

elif choice == '5':

view\_customer\_suggestions()

elif choice == '6':

allergen\_name = input("Enter allergen name to add: ").strip()

add\_allergen(allergen\_name)

elif choice == '7':

view\_allergens()

elif choice == '8':

print("Exiting system. Goodbye!")

break

else:

print("Invalid choice. Please try again.")

elif user\_role == 'customer':

print("1. Search & Filter Flavors")

print("2. Add Customer Suggestion")

print("3. Add to Cart")

print("4. View Cart")

print("5. Remove from Cart")

print("6. Exit")

choice = input("Enter your choice (1-6): ").strip()

if choice == '1':

search\_term = input("Enter search term (or press Enter to skip): ").strip()

filter\_season = input("Enter season to filter by (or press Enter to skip): ").strip()

search\_flavors(search\_term, filter\_season)

elif choice == '2':

customer\_name = input("Enter your name: ").strip()

flavor\_name = input("Enter flavor name suggestion: ").strip()

allergy\_concerns = input("Enter any allergy concerns (or press Enter to skip): ").strip()

add\_customer\_suggestion(customer\_name, flavor\_name, allergy\_concerns)

elif choice == '3':

add\_to\_cart()

elif choice == '4':

view\_cart()

elif choice == '5':

cart\_id = int(input("Enter Cart ID to remove: ").strip())

remove\_from\_cart(cart\_id)

elif choice == '6':

print("Exiting system. Goodbye!")

break

else:

print("Invalid choice. Please try again.")

main\_menu()

**SQL**

**SQL Query or ORM abstraction Implementation**

**1. Creating Tables**

These SQL queries create the tables needed for the application if they don’t already exist:

**CREATE TABLE for seasonal\_flavors:**

sql

CREATE TABLE IF NOT EXISTS seasonal\_flavors (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

price REAL NOT NULL,

available INTEGER DEFAULT 1,

season TEXT

);

**One-line description :**Creates a table for ice cream flavors with columns for the flavor's name, description, price, availability, and season.

**CREATE TABLE for ingredients:**

**sql**

CREATE TABLE IF NOT EXISTS ingredients (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

quantity REAL NOT NULL,

unit TEXT NOT NULL

);

**One-line description :**Creates a table for ingredients with columns for the ingredient name, quantity, and unit of measurement.

**CREATE TABLE for customer\_suggestions:**

**sql**

CREATE TABLE IF NOT EXISTS customer\_suggestions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

customer\_name TEXT NOT NULL,

flavor\_name TEXT NOT NULL,

suggestion\_date TEXT NOT NULL,

allergy\_concerns TEXT

);

**One-line description :**Creates a table for customer suggestions, storing the customer’s name, flavor suggestion, date, and any allergy concerns.

**CREATE TABLE for allergens:**

**sql**

CREATE TABLE IF NOT EXISTS allergens (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE

);

**One-line description :**Creates a table to store allergens, with each allergen’s name.

**CREATE TABLE for cart:**

**sql**

CREATE TABLE IF NOT EXISTS cart (

id INTEGER PRIMARY KEY AUTOINCREMENT,

flavor\_id INTEGER NOT NULL,

quantity INTEGER NOT NULL,

FOREIGN KEY (flavor\_id) REFERENCES seasonal\_flavors (id)

);

**One-line description :**Creates a table to store items in a shopping cart, linking each item to a flavor in the seasonal\_flavors table.

**2. Inserting Data**

These SQL queries add new data to the tables.

**Add Flavor:**

**sql**

INSERT INTO seasonal\_flavors (name, description, price, season)

VALUES (?, ?, ?, ?)

**One-line description :** Adds a new flavor to the seasonal\_flavors table.

**Add Ingredient:**

**sql**

INSERT INTO ingredients (name, quantity, unit)

VALUES (?, ?, ?)

**One-line description :**Adds a new ingredient to the ingredients table.

**Add Customer Suggestion:**

**sql**

INSERT INTO customer\_suggestions (customer\_name, flavor\_name, suggestion\_date, allergy\_concerns)

VALUES (?, ?, ?, ?)

**One-line description :** Adds a customer’s suggestion to the customer\_suggestions table.

**Add Allergen:**

**sql**

INSERT INTO allergens (name)

VALUES (?)

**One-line description :**Adds a new allergen to the allergens table.

**Add to Cart:**

**sql**

INSERT INTO cart (flavor\_id, quantity)

VALUES (?, ?)

**One-line description :**Adds an item to the shopping cart.

**3. Selecting Data**

These SQL queries retrieve data from the tables.

**Get All Flavors:**

**Sql**

SELECT \* FROM seasonal\_flavors WHERE 1

**One-line description :**Retrieves all flavors from the seasonal\_flavors table.

**Search Flavors by Name:**

**sql**

SELECT \* FROM seasonal\_flavors WHERE name LIKE ?

**One-line description :** Retrieves flavors whose name contains a specific search term.

**Get Flavors by Season:**

**sql**

SELECT \* FROM seasonal\_flavors WHERE season = ?

**One-line description :** Retrieves all flavors for a specific season.

**Get All Customer Suggestions:**

**sql**

SELECT \* FROM customer\_suggestions

**One-line description :** Retrieves all customer suggestions.

**Get All Allergens:**

**sql**

SELECT \* FROM allergens

**One-line description :**Retrieves all allergens.

**Get Available Flavors:**

**sql**

SELECT \* FROM seasonal\_flavors WHERE available = 1

**One-line description :**Retrieves all available flavors (where the available column is 1).

**Get Ingredient Quantity:**

**sql**

SELECT quantity FROM ingredients WHERE name = ?

**One-line description :**Retrieves the quantity of a specific ingredient.

**View Cart with Flavor Details:**

**sql**

SELECT c.id, s.name, c.quantity, s.price, (c.quantity \* s.price) AS total\_price

FROM cart c

JOIN seasonal\_flavors s ON c.flavor\_id = s.id

**One-line description :**Retrieves details of all items in the cart, including the flavor name, price, quantity, and total price for each item.

**4. Updating Data**

These queries update existing data in the tables.

**Update Ingredient Quantity:**

**sql**

UPDATE ingredients

SET quantity = ?

WHERE name = ?

**One-line description :**Updates the quantity of a specific ingredient.

**5. Deleting Data**

This query removes data from the cart.

**Remove Item from Cart:**

sql

DELETE FROM cart WHERE id = ?

**One-line description :**Deletes an item from the cart by its id.