**L7 Informatics Internship Program**

**Assignment**

* app(Backend directory)

1. main.py(FastAPI backend logic)
2. models.py(SQLAlchemy database models)
3. routes.py(API route definitions)

* frontend

1. app.py(Streamlit app file)

* Dockerfile
* requirements.txt(dependencies for the project)

**app(Backend directory)**

**main.py**  
  
from fastapi import FastAPI, HTTPException, Query, Body

from pydantic import BaseModel

from typing import List

# Initialize the FastAPI app

app = FastAPI()

# In-memory data

flavors = [

    {"id": 1, "name": "Vanilla", "ingredients": ["Milk", "Sugar"], "availability": True},

    {"id": 2, "name": "Chocolate", "ingredients": ["Cocoa", "Milk"], "availability": True},

]

ingredients = [

    {"id": 1, "name": "Milk", "stock": 100},

    {"id": 2, "name": "Sugar", "stock": 50},

    {"id": 3, "name": "Cocoa", "stock": 30},

]

allergens = ["Peanuts", "Gluten"]

cart = []

# Pydantic models

class FlavorRequest(BaseModel):

    name: str

    ingredients: List[str]

class AddToCartRequest(BaseModel):

    flavor\_id: int

class AllergenRequest(BaseModel):

    allergen: str

# Routes

@app.get("/")

def root():

    return {

        "welcome\_message": "Welcome to the Ice Cream Parlor API!",

        "instructions": "Use /docs to explore the API documentation.",

        "available\_endpoints": {

            "View Flavors": "/flavors/",

            "Add Flavor": "/add\_flavor/",

            "Add Allergen": "/add\_allergen/",

            "View Cart": "/cart/",

            "Add to Cart": "/cart/add/",

            "Search Flavors": "/search\_flavors/",

        },

    }

@app.get("/flavors/")

def get\_flavors():

    return {"flavors": flavors}

@app.post("/add\_flavor/")

def add\_flavor(flavor: FlavorRequest):

    # Check for duplicate flavors

    if any(f["name"].lower() == flavor.name.lower() for f in flavors):

        raise HTTPException(status\_code=400, detail="Flavor already exists")

    new\_flavor = {

        "id": len(flavors) + 1,

        "name": flavor.name,

        "ingredients": flavor.ingredients,

        "availability": True,

    }

    flavors.append(new\_flavor)

    return {"message": "Flavor added successfully!", "flavor": new\_flavor}

@app.get("/cart/")

def view\_cart():

    return {"cart": cart}

@app.post("/cart/add/")

def add\_to\_cart(request: AddToCartRequest):

    flavor = next((f for f in flavors if f["id"] == request.flavor\_id), None)

    if not flavor:

        raise HTTPException(status\_code=404, detail="Flavor not found")

    cart.append(flavor)

    return {"message": "Flavor added to cart", "cart": cart}

@app.post("/add\_allergen/")

def add\_allergen(request: AllergenRequest):

    if request.allergen.lower() in (a.lower() for a in allergens):

        raise HTTPException(status\_code=400, detail="Allergen already exists")

    allergens.append(request.allergen)

    return {"message": "Allergen added successfully", "allergens": allergens}

@app.get("/search\_flavors/")

def search\_flavors(keyword: str):

    # Convert the keyword to lowercase and search for matching flavor names

    results = [f for f in flavors if keyword.lower() in f["name"].lower()]

    # If no results found, return a message

    if not results:

        return {"message": "No flavors found", "results": []}

    return {"results": results}  
  
  
**models.py**

from sqlalchemy import Column, Integer, String, Boolean, ForeignKey, create\_engine

from sqlalchemy.orm import declarative\_base, relationship, sessionmaker

Base = declarative\_base()

class Flavor(Base):

    \_\_tablename\_\_ = "flavors"

    id = Column(Integer, primary\_key=True, index=True)

    name = Column(String, unique=True, nullable=False)

    ingredients = Column(String, nullable=False)

    availability = Column(Boolean, default=True)

class Ingredient(Base):

    \_\_tablename\_\_ = "ingredients"

    id = Column(Integer, primary\_key=True, index=True)

    name = Column(String, unique=True, nullable=False)

    stock = Column(Integer, nullable=False)

class Allergen(Base):

    \_\_tablename\_\_ = "allergens"

    id = Column(Integer, primary\_key=True, index=True)

    name = Column(String, unique=True, nullable=False)

class Suggestion(Base):

    \_\_tablename\_\_ = "suggestions"

    id = Column(Integer, primary\_key=True, index=True)

    suggestion = Column(String, nullable=False)

# Database setup

DATABASE\_URL = "sqlite:///./db.sqlite"

engine = create\_engine(DATABASE\_URL)

SessionLocal = sessionmaker(bind=engine)

Base.metadata.create\_all(bind=engine)

**routes.py**

from fastapi import APIRouter, Depends, HTTPException

from sqlalchemy.orm import Session

from app.models import Flavor, Ingredient, Allergen, Suggestion, SessionLocal

router = APIRouter()

def get\_db():

    db = SessionLocal()

    try:

        yield db

    finally:

        db.close()

@router.post("/add\_flavor/")

def add\_flavor(name: str, ingredients: str, db: Session = Depends(get\_db)):

    flavor = Flavor(name=name, ingredients=ingredients)

    db.add(flavor)

    db.commit()

    return {"message": "Flavor added successfully!"}

@router.get("/view\_flavors/")

def view\_flavors(db: Session = Depends(get\_db)):

    return db.query(Flavor).all()

@router.post("/add\_allergen/")

def add\_allergen(name: str, db: Session = Depends(get\_db)):

    allergen = Allergen(name=name)

    db.add(allergen)

    db.commit()

    return {"message": "Allergen added successfully!"}

**Frontend**

**App.py**

import streamlit as st

import requests

# Backend base URL

BASE\_URL = "http://127.0.0.1:8000"

# App Title

st.title("🍦 Ice Cream Parlor App")

# Sidebar for navigation

st.sidebar.title("Navigation")

options = st.sidebar.radio(

    "Go to:",

    ["View Flavors", "Add Flavor", "Search Flavors", "Manage Cart", "Add Allergen"]

)

# View Flavors

if options == "View Flavors":

    st.header("Available Flavors")

    if st.button("Fetch Flavors"):

        response = requests.get(f"{BASE\_URL}/flavors/")

        if response.status\_code == 200:

            flavors = response.json()["flavors"]

            if flavors:

                for flavor in flavors:

                    st.subheader(flavor["name"])

                    st.write(f"Ingredients: {', '.join(flavor['ingredients'])}")

                    st.write(f"Availability: {'Yes' if flavor['availability'] else 'No'}")

                    st.write("---")

            else:

                st.info("No flavors available.")

        else:

            st.error("Failed to fetch flavors!")

# Add Flavor

elif options == "Add Flavor":

    st.header("Add a New Flavor")

    with st.form("Add Flavor Form"):

        name = st.text\_input("Flavor Name")

        ingredients = st.text\_area("Ingredients (comma-separated)")

        submitted = st.form\_submit\_button("Add Flavor")

        if submitted:

            if name and ingredients:

                ingredients\_list = [i.strip() for i in ingredients.split(",")]

                response = requests.post(

                    f"{BASE\_URL}/add\_flavor/",

                    json={"name": name, "ingredients": ingredients\_list}

                )

                if response.status\_code == 200:

                    st.success("Flavor added successfully!")

                elif response.status\_code == 400:

                    st.error(response.json()["detail"])

                else:

                    st.error("Failed to add flavor!")

            else:

                st.error("Please fill in all fields!")

# Search Flavors

elif options == "Search Flavors":

    st.header("Search for a Flavor")

    keyword = st.text\_input("Enter keyword to search for flavors")

    if st.button("Search"):

        response = requests.get(f"{BASE\_URL}/search\_flavors/?keyword={keyword}")

        if response.status\_code == 200:

            results = response.json()["results"]

            if results:

                for flavor in results:

                    st.subheader(flavor["name"])

                    st.write(f"Ingredients: {', '.join(flavor['ingredients'])}")

                    st.write(f"Availability: {'Yes' if flavor['availability'] else 'No'}")

                    st.write("---")

            else:

                st.info("No flavors found.")

        else:

            st.error("Failed to search flavors!")

# Manage Cart

elif options == "Manage Cart":

    st.header("Manage Your Cart")

    with st.form("Add to Cart Form"):

        flavor\_id = st.number\_input("Flavor ID to Add to Cart", min\_value=1, step=1)

        submitted = st.form\_submit\_button("Add to Cart")

        if submitted:

            response = requests.post(f"{BASE\_URL}/cart/add/", json={"flavor\_id": flavor\_id})

            if response.status\_code == 200:

                st.success("Flavor added to cart successfully!")

            elif response.status\_code == 404:

                st.error(response.json()["detail"])

            else:

                st.error("Failed to add flavor to cart!")

    if st.button("View Cart"):

        response = requests.get(f"{BASE\_URL}/cart/")

        if response.status\_code == 200:

            cart = response.json()["cart"]

            if cart:

                for item in cart:

                    st.subheader(item["name"])

                    st.write(f"Ingredients: {', '.join(item['ingredients'])}")

                    st.write("---")

            else:

                st.info("Your cart is empty.")

        else:

            st.error("Failed to fetch cart details!")

# Add Allergen

elif options == "Add Allergen":

    st.header("Add an Allergen")

    with st.form("Add Allergen Form"):

        allergen = st.text\_input("Allergen Name")

        submitted = st.form\_submit\_button("Add Allergen")

        if submitted:

            if allergen:

                response = requests.post(f"{BASE\_URL}/add\_allergen/", json={"allergen": allergen})

                if response.status\_code == 200:

                    st.success("Allergen added successfully!")

                elif response.status\_code == 400:

                    st.error(response.json()["detail"])

                else:

                    st.error("Failed to add allergen!")

            else:

                st.error("Please provide an allergen name!")  
  
  
**Dockerfile**

# Use a lightweight Python image

FROM python:3.10-slim

# Set the working directory in the container

WORKDIR /app

# Copy the project files into the container

COPY . /app

# Install dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Expose the ports for FastAPI (8000) and Streamlit (8502)

EXPOSE 8501

# Command to run both backend and frontend

CMD ["sh", "-c", "uvicorn main:app --host 0.0.0.0 --port 8000 & streamlit run frontend/app.py --server.port=8501 --server.enableCORS=false"]

**requirements.txt**

fastapi

uvicorn

sqlalchemy

streamlit

pytest

**Test\_main.py**

import sys

import os

import pytest

from sqlalchemy.orm import Session

from app.models import SessionLocal, Flavor

# Test function to add a flavor to the database

def test\_add\_flavor():

    # Create a new session

    db: Session = SessionLocal()

    # Add a new flavor

    new\_flavor = Flavor(name="Vanilla", ingredients="Milk, Sugar")

    db.add(new\_flavor)

    db.commit()

    # Query the database to check if the flavor exists

    flavor = db.query(Flavor).filter\_by(name="Vanilla").first()

    assert flavor is not None

    assert flavor.name == "Vanilla"

    assert flavor.ingredients == "Milk, Sugar"

    # Clean up the test data

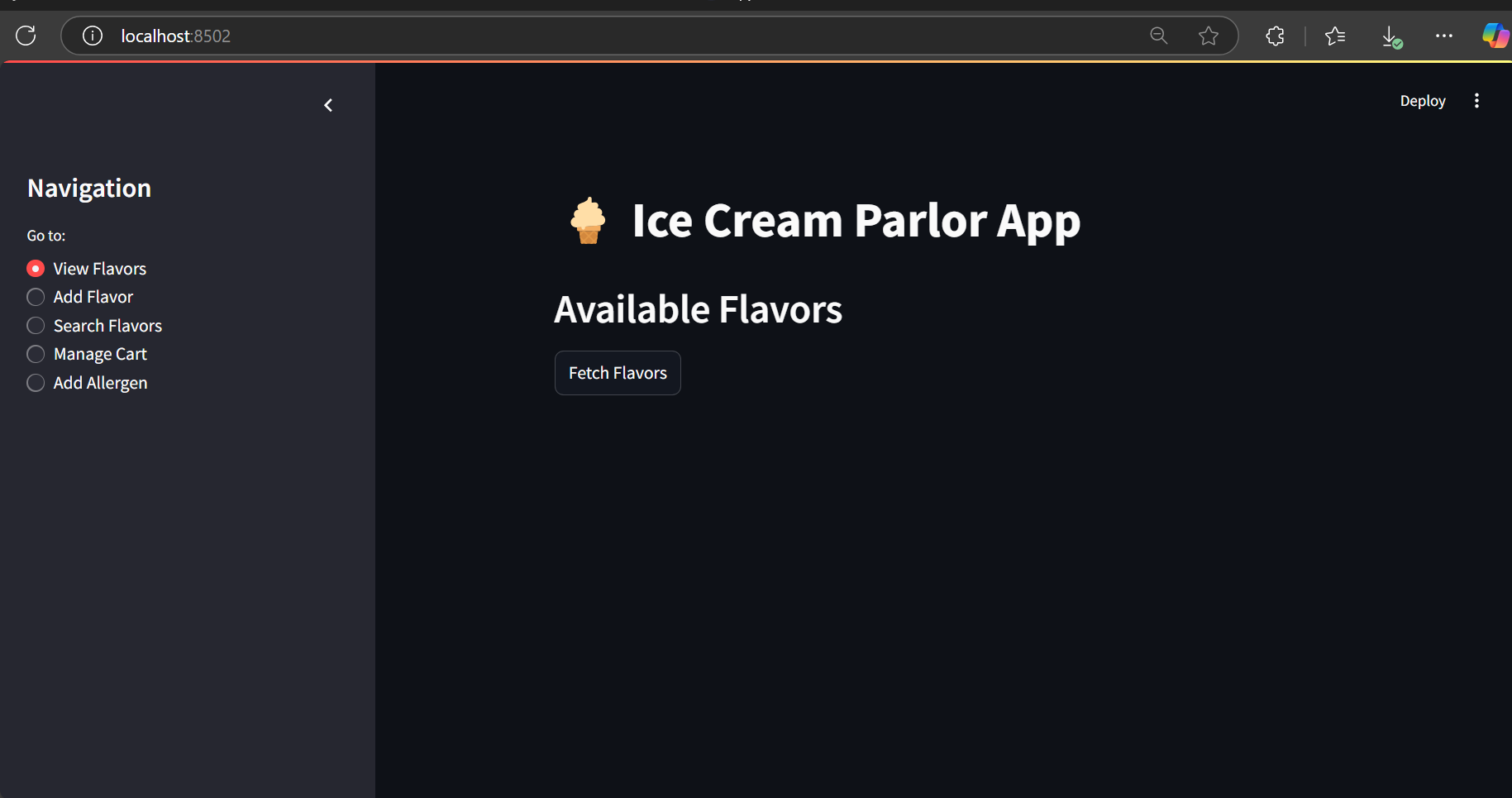
    db.delete(flavor)

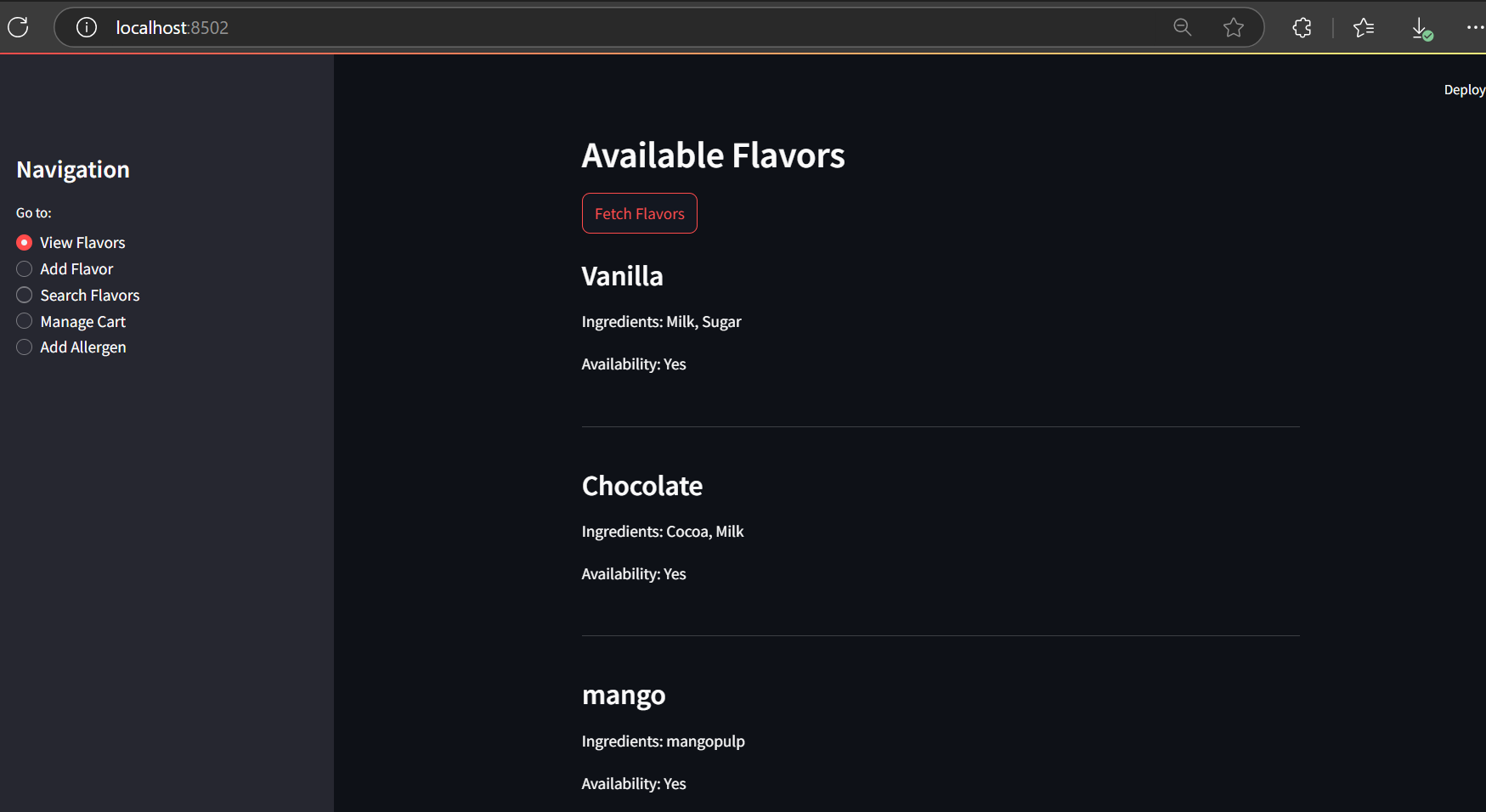
    db.commit()

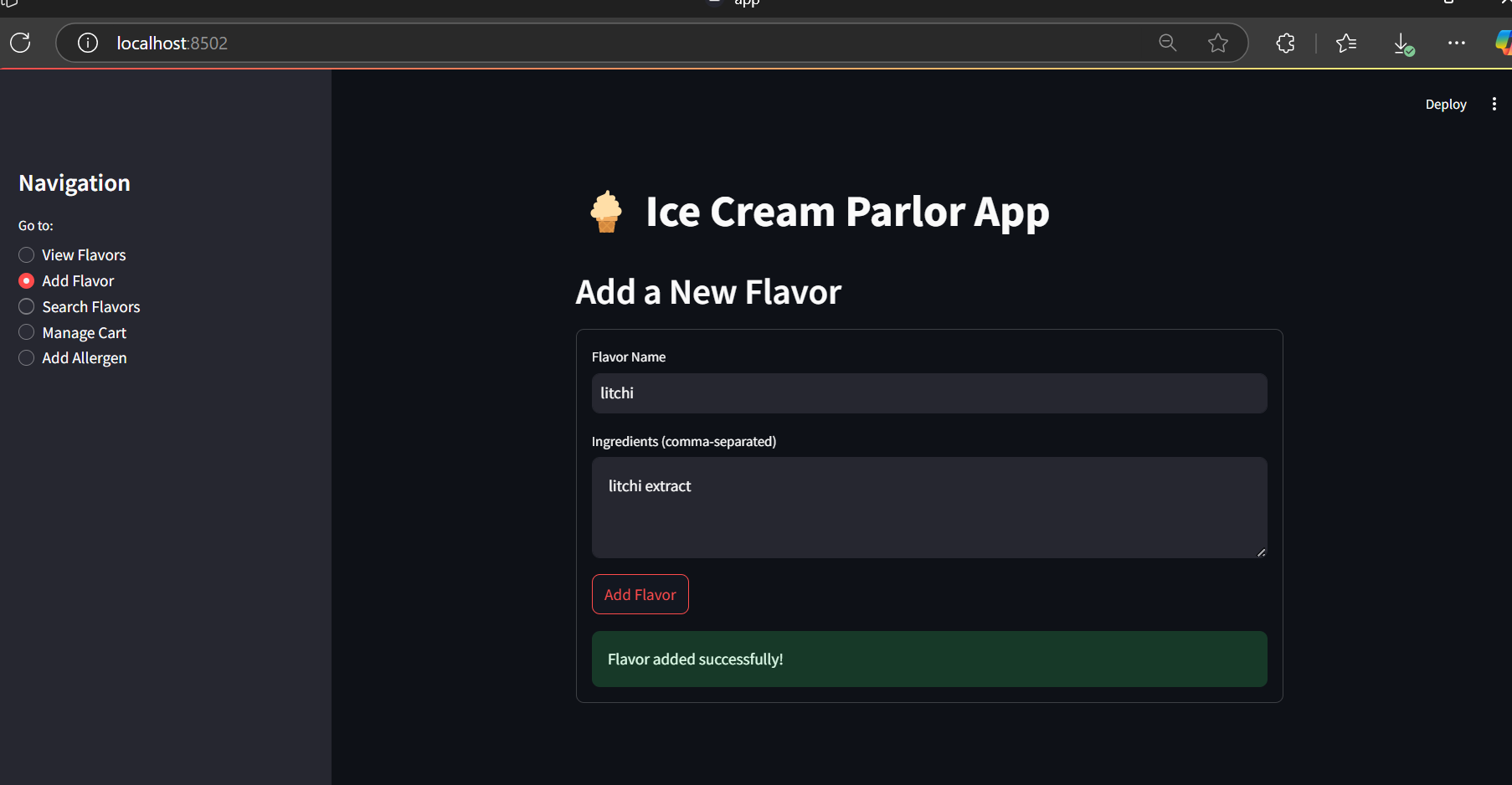
    # Close the session

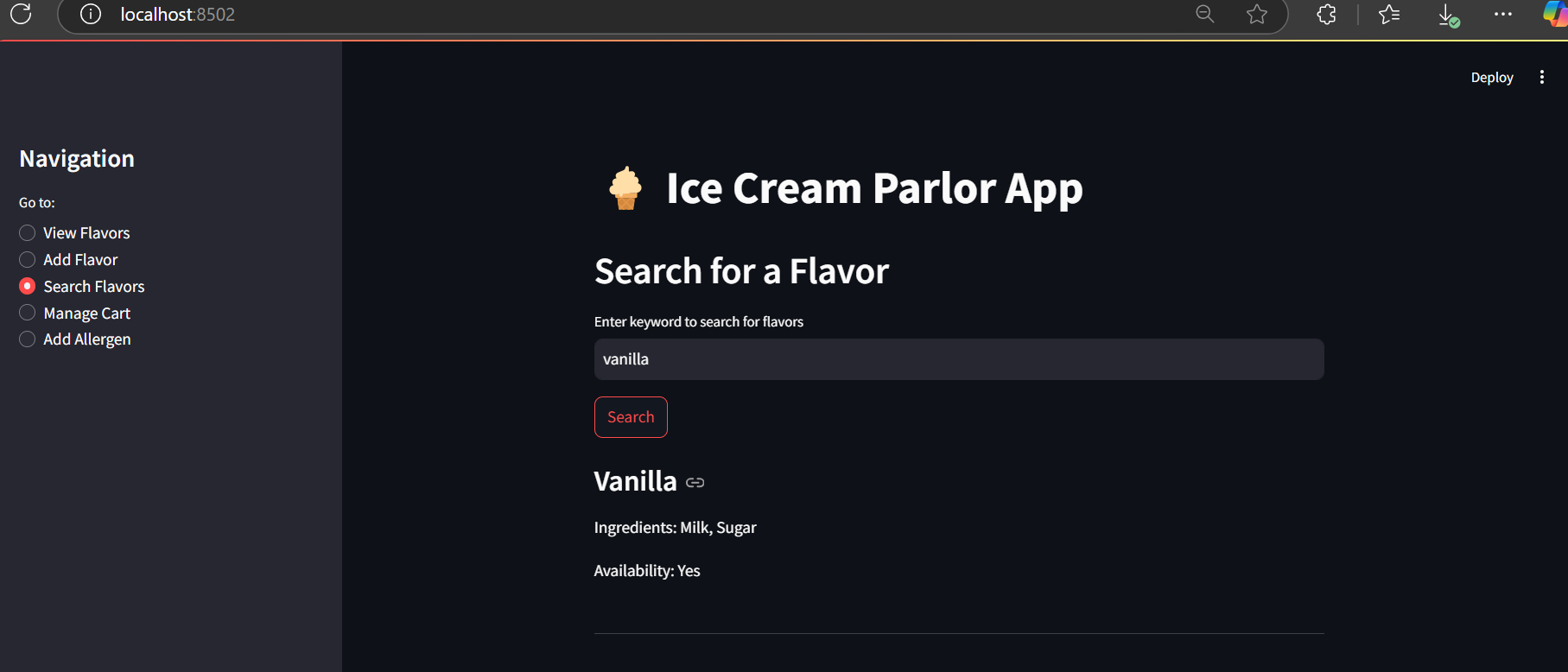
    db.close()

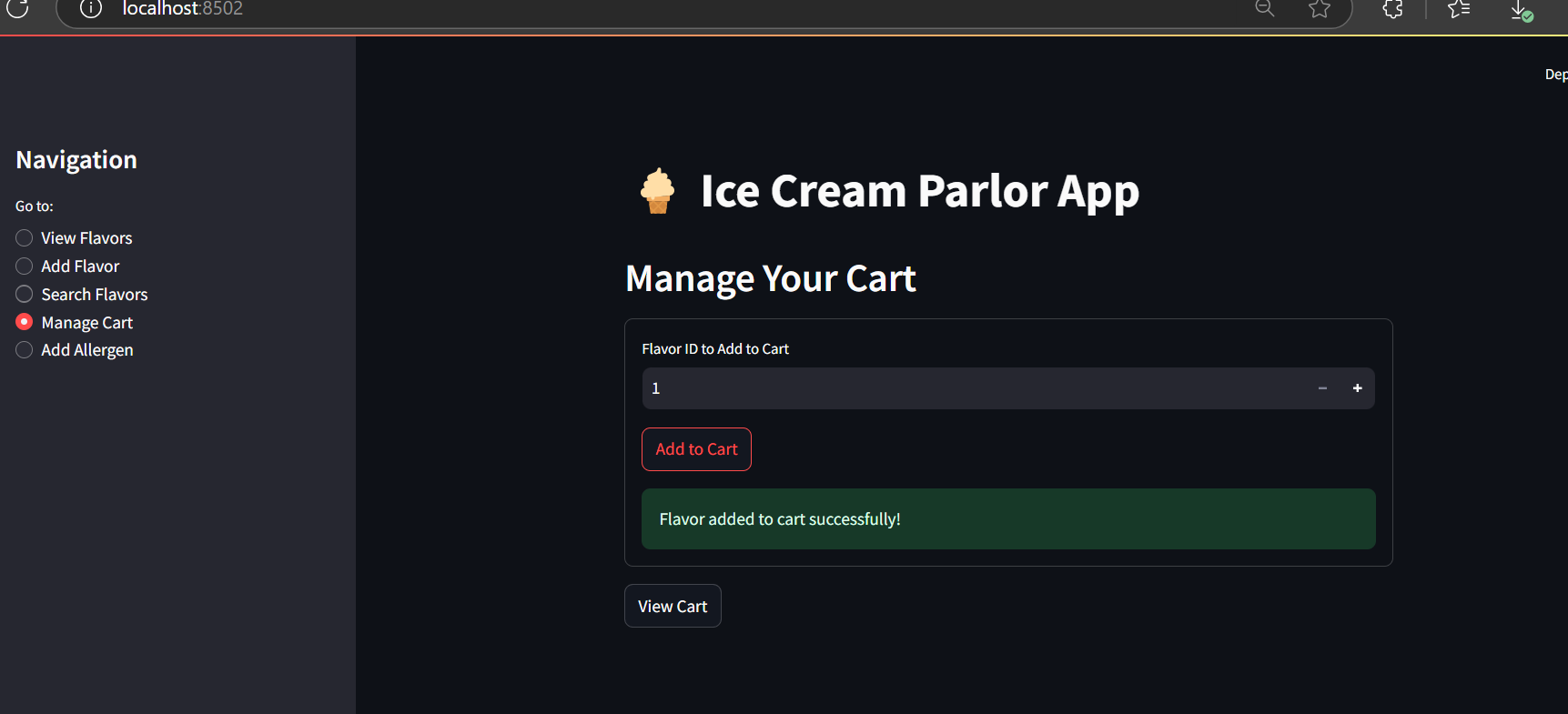
**OUTPUT SCREENSHOTS:**

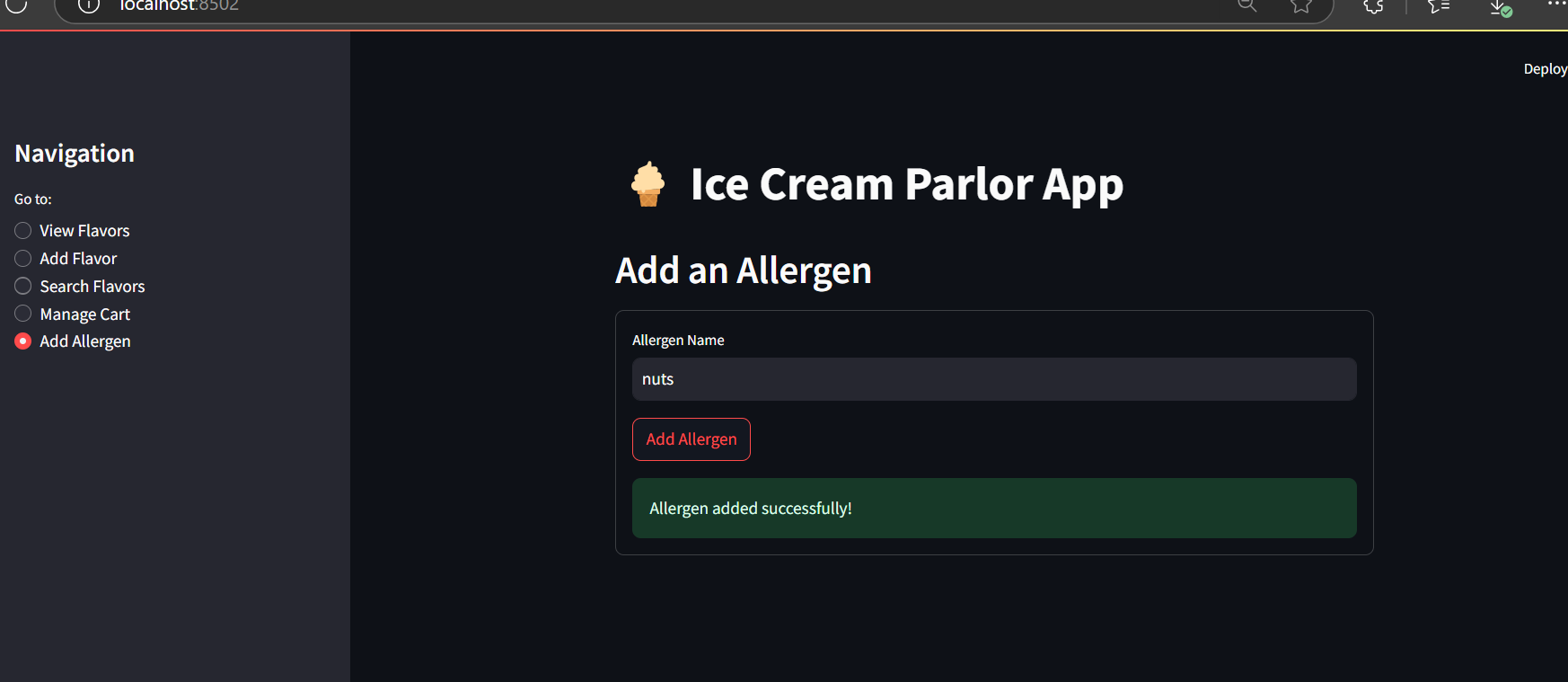
****

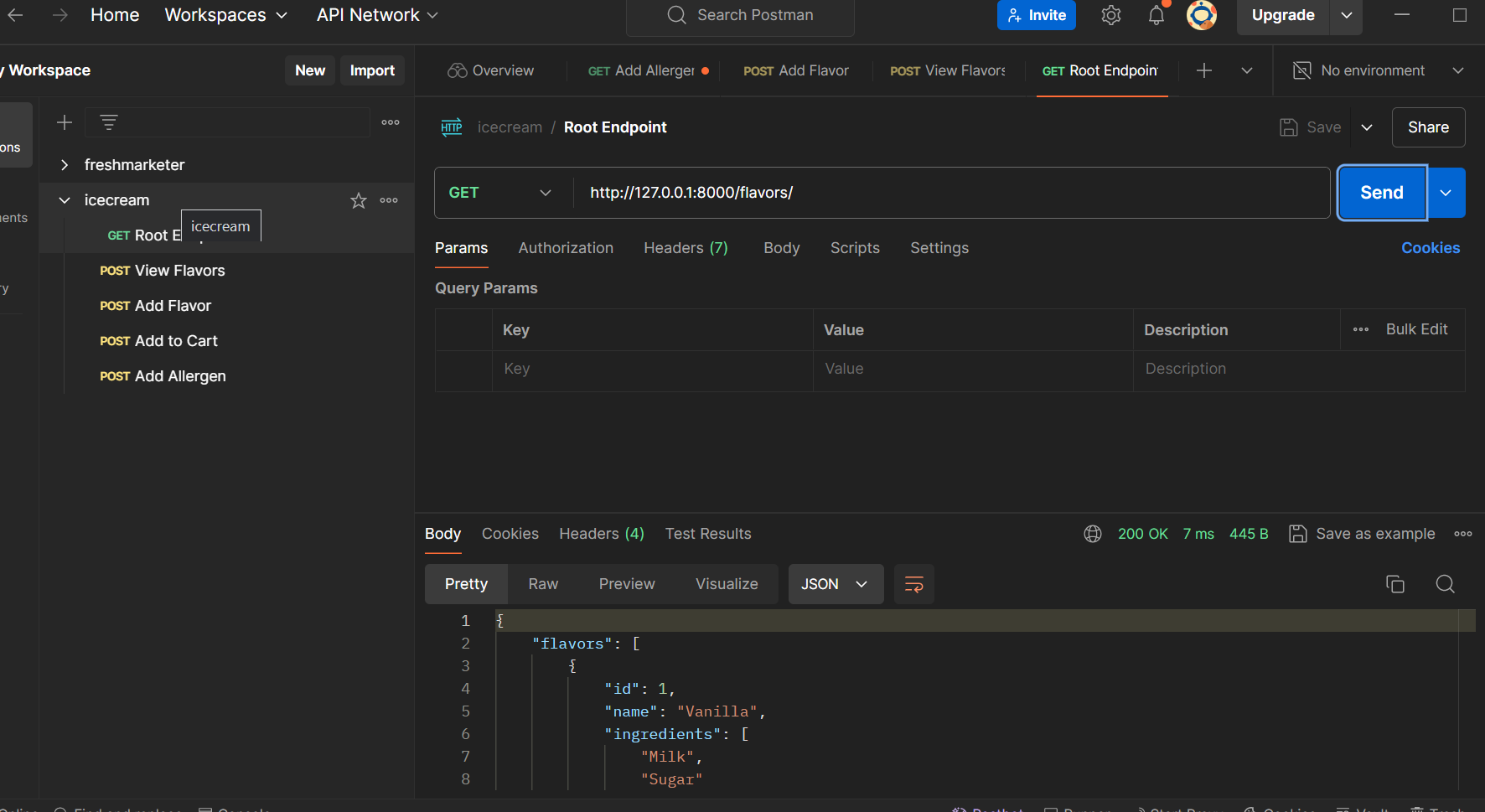




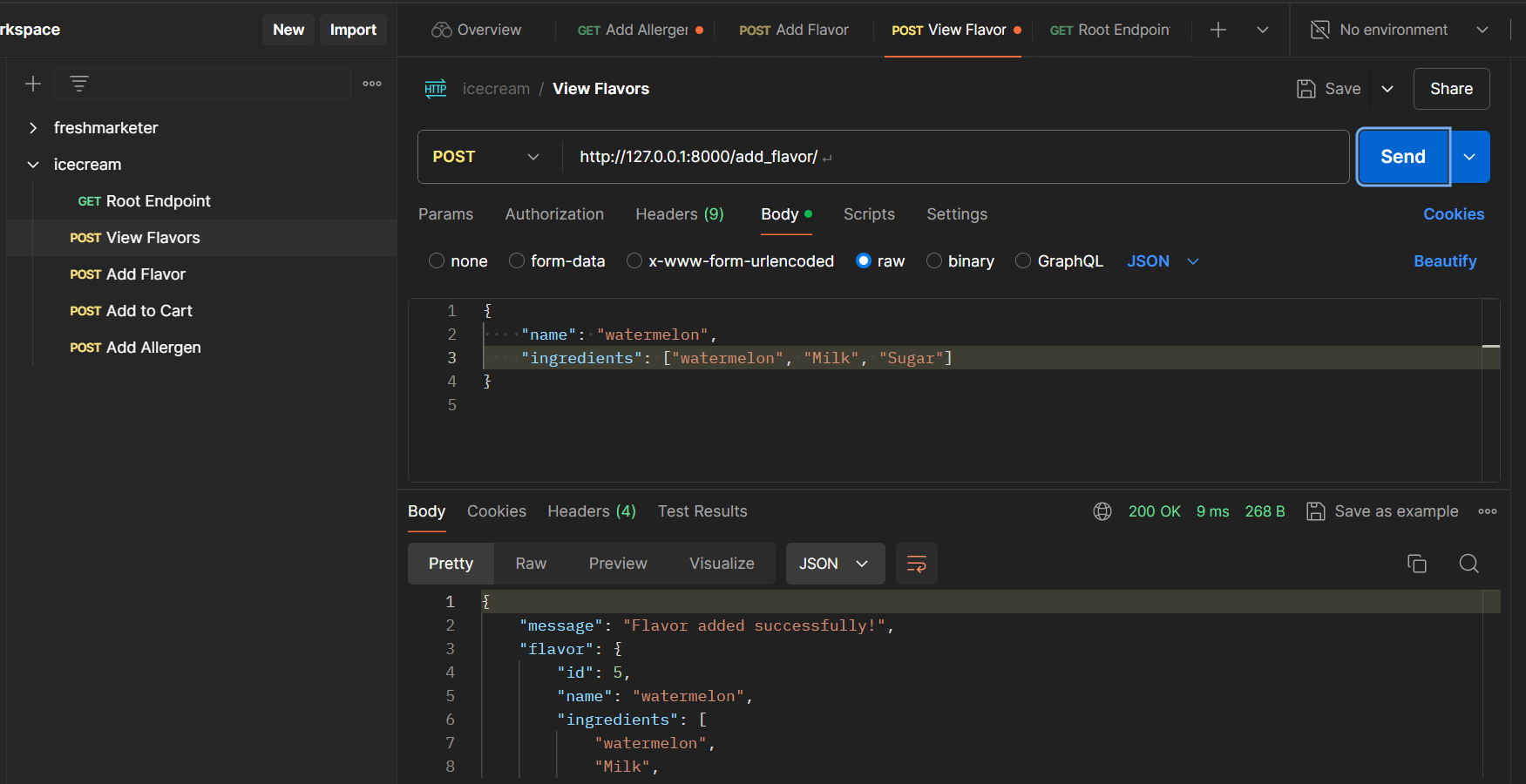




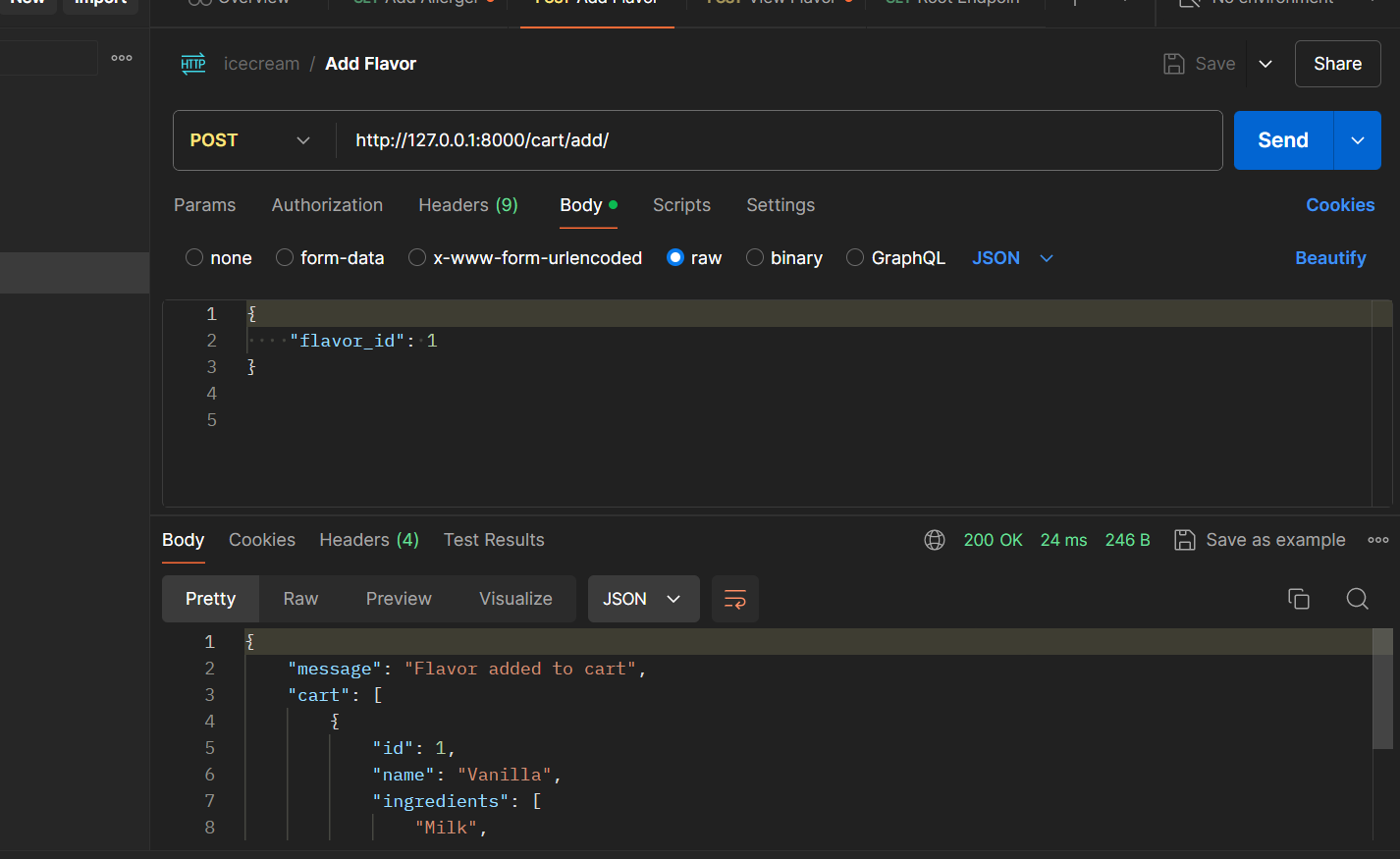


API-POSTMAN  
1) ROOT ENDPOINT  


2) VIEW FLAVOURS



3) ADD FLAVOURS



4) ADD TO CART

