AI Recipe Finder Using Cloud Technologies

Project Report

Title and Participants

Title: AI Recipe Finder Using Cloud Technologies
Participants: Srimedha Bhavani Chandoo

Project Goals

The goal of this project is to design and implement an intelligent recipe recommendation system using advanced AI and cloud technologies. The system aims to simplify the process of finding recipes by allowing users to input a list of ingredients they have on hand. Based on this input, the system dynamically generates or retrieves appropriate recipes.

To achieve this, I leveraged several modern datacenter components, including Google Cloud Functions, Google Cloud Storage, Google Pub/Sub, and GPT-Neo for AI-based recipe generation. The project focuses on creating a scalable, serverless architecture that ensures fast and reliable responses to user queries. Additionally, I wanted to integrate multiple cloud services in a meaningful way to showcase a real-world application of distributed computing technologies.

Software and Hardware Components

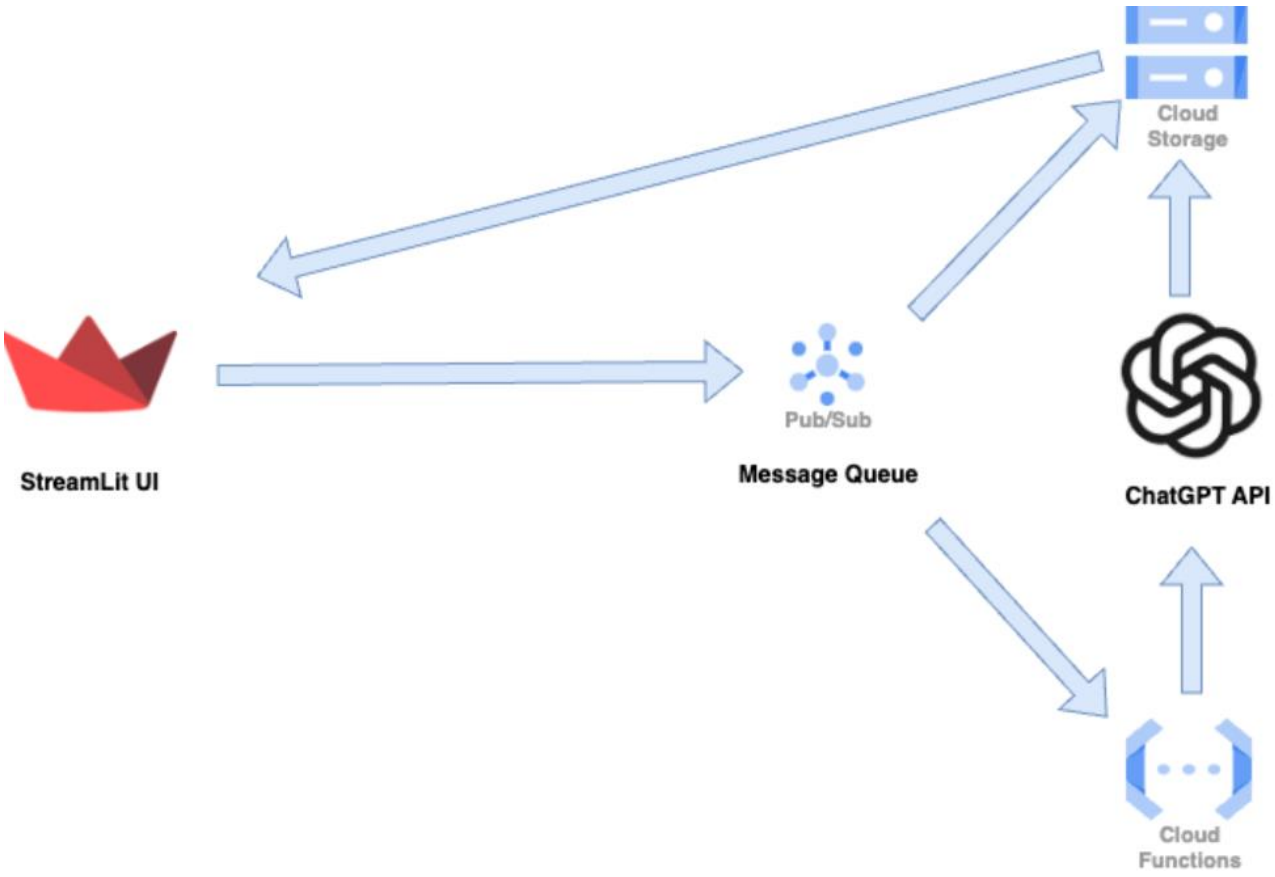
Software Components

1. **Streamlit**: A Python-based web framework used to create the user interface. It provides an interactive front-end where users input their ingredients and view recipes.
2. **GPT-Neo**: An open-source language model used to dynamically generate creative and human-like recipes.
3. **Google Cloud Functions**: A serverless compute platform used to process HTTP requests and retrieve recipes based on user input.
4. **Google Cloud Storage**: A persistent storage service used to save generated recipes in .txt format.
5. **Google Pub/Sub**: A messaging service used for asynchronous communication between different components.
6. **Python Libraries**: Includes torch, transformers, requests, and other libraries essential for integrating AI and cloud services.

Hardware Components

1. **Google Cloud Infrastructure**: Hosted services for Cloud Functions, Storage, and Pub/Sub.
2. **Local Machine**: Used for development, testing, and running the Streamlit application.

Architectural Diagram



Description of the Architecture

The architecture consists of the following:

- **User Input**: The user inputs a list of ingredients through the Streamlit app, which serves as the front-end.
- **Google Cloud Functions**: The app sends an HTTP POST request to the deployed Cloud Function. The function processes the input and either retrieves a predefined recipe or generates one dynamically.
- **Google Cloud Storage**: The generated recipes are saved here for persistence. This ensures recipes can be retrieved later if needed.
- **Google Pub/Sub**: Messages related to ingredient inputs are published to a topic for potential asynchronous processing.
- **GPT-Neo**: Acts as the AI engine, generating recipes when requested by the Cloud Function.

Component Interactions

Streamlit App

The Streamlit app is the primary interface for users. It accepts a list of ingredients and displays the retrieved or generated recipe. The app interacts with the Cloud Function using HTTP POST requests and handles the response to display the recipe.

Google Cloud Functions

This is the core processing unit of the system. It:

1. Receives ingredient input from the Streamlit app.
2. Matches the input with predefined recipes or generates a new recipe using GPT-Neo.
3. Returns the recipe as a JSON response.

Google Cloud Storage

Recipes generated by the Cloud Function are saved in .txt format. This ensures that recipes can be stored persistently and accessed later for debugging or reusability.

Google Pub/Sub

Ingredient inputs are published to a Pub/Sub topic to demonstrate asynchronous communication. Subscribers can retrieve these messages for further processing, such as logging or analytics.

GPT-Neo

GPT-Neo generates human-like recipes when a predefined recipe is not found. It takes user ingredients as input and outputs a recipe that includes a title, a list of ingredients, and step-by-step instructions.

Testing and Debugging

Testing

To ensure the system functions correctly, I performed the following tests:

1. **Unit Testing**:
    - Verified that the Cloud Function correctly processes ingredient input and returns recipes.
    - Ensured that recipes are properly saved to Cloud Storage.
2. **Integration Testing**:
    - Tested the interaction between the Streamlit app, Cloud Function, and Cloud Storage.
    - Simulated real-world scenarios to ensure seamless data flow.

3. **Load Testing**:
    - Sent multiple concurrent requests to evaluate the scalability of the Cloud Function.

Debugging

1. **Google Cloud Logs**:
    - Used to trace errors and monitor the execution of Cloud Functions.
2. **Postman**:
    - Verified the API responses from the Cloud Function to ensure data correctness.
3. **Local Testing**:
    - Ran the Streamlit app locally with mocked Cloud Function endpoints for faster debugging.

Capabilities and Bottlenecks

Capabilities

1. **Real-Time Recipe Recommendations**:
    - The system responds instantly to user input by retrieving or generating recipes.
2. **Scalable and Serverless**:
    - The use of Google Cloud Functions ensures scalability without manual server management.
3. **Persistent Storage**:
    - Recipes are stored in Google Cloud Storage for future retrieval.

Potential Bottlenecks

1. **Latency**:
    - Network delays can occur when communicating with cloud services.
2. **Cloud Function Limits**:
    - Execution time is capped at 540 seconds, which may limit complex operations.
3. **Model Performance**:
    - Generating recipes using GPT-Neo requires significant computational resources.

Conclusion

The **AI Recipe Finder** demonstrates how modern cloud technologies can be integrated with AI to solve real-world problems. This project successfully combines multiple datacenter components into a cohesive system, showcasing both technical and architectural design. While there are limitations, the project highlights the potential of serverless and distributed systems in creating scalable applications.

Appendices

- **Github Repository**: https://github.com/cu-csci-4253-datacenter-fall-2024/finalproject-final-project-team-31
- **Demo Video**: https://youtu.be/K7k5v-cphJ8