DEVELOPING AI ASSISTANT FOR GROCERY SHOPPING

A MINI PROJECT REPORT

Submitted by

ABDUL HASEEF S	**110120205004**
MOHAMED AJUMAL M	**110120205012**
MOHAMED ATHIF HUSSAIN S	**110120205015**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE :: 2023

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“DEVELOPING AI ASSISTANT FOR GROCERY SHOPPING”** is the Bonafide work of **“ABDUL HASEEF S (110120205004) , MOHAMED AJUMAL M (110120205012), MOHAMED ATHIF HUSSAIN S (110120205015)”** who carried

out project work under my supervision

SIGNATURE	**SIGNATURE**
Dr. S.ARIF ABDUL RAHUMAN **HEAD OF THE DEPARTMENT**	**Ms. C.S.NANDHINI** **SUPERVISOR**
Department of Information Technology Aalim Muhammed Salegh College of Engineering Muthapudupet,Avadi IAF, Chennai 600 055.	Department of Information Technology Aalim Muhammed Salegh College of Engineering Muthapudupet,Avadi IAF, Chennai 600 055.

CERTIFICATE OF EVALUATION

COLLEGE NAME	:	AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING
BRANCH	**:**	INFORMATION TECHNOLOGY
PROJECT TITLE	:	DEVELOPING AI ASSISTANT FOR GROCERY SHOPPING

NAME OF THE STUDENT	**REGISTRATION NUMBER**	**NAME OF THE SUPERVISOR**
ABDUL HASEEF S	110120205004	Ms.C.S.NANDHINI
MOHAMED AJUMAL M	110120205012	
MOHAMED ATHIF HUSSAIN S	110120205015	

The report of this project is submitted by the above students in partial fulfillment For the award of Bachelor of Technology in **INFORMATION TECHNOLOGY** of Anna University are evaluated and confirmed to report of the work done by the above students during the academic year of 2022-2023

This report work is submitted for the Anna University project viva voiceworkHeld on …………….

……………………………… ……………………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATION

XAMPP	X - Cross-platform A - Apache HTTP Server M - MySQL database P – PHP P - Perl
SQL	Structured Query Language
QB	Qubole

ABSTRACT

Artificial Intelligence (AI) has revolutionized various industries, and grocery shopping is no exception. This abstract presents an overview of AI-enabled grocery shopping systems that have transformed the traditional consumer experience. By leveraging cutting-edge technologies, such as machine learning, natural language processing, and computer vision, AI-powered grocery shopping platforms offer a range of benefits to consumers, retailers. The key features and functionalities of AI grocery shopping systems explores personalized shopping experiences facilitated by AI algorithms that analyze consumer preferences, purchase history, and contextual factors to suggest relevant products and offers. Through intelligent recommendations and tailored promotions, AI systems enhance customer satisfaction, increase sales, and foster brand loyalty. Additionally, AI-driven inventory management systems optimize stock levels, reduce waste, and streamline supply chains. Real-time data analysis and predictive analytics enable retailers to anticipate demand, manage perishable items efficiently, and maintain optimal inventory levels, ensuring product availability and reducing costs.

CHAPTER 1
1. INTRODUCTION

1.1 **DEVELOPING AI ASSISTANT FOR GROCERY SHOPPING**

The advent of Artificial Intelligence (AI) has brought about significant advancements and transformations across various industries, and the realm of grocery shopping is no exception. AI has revolutionized the way consumers browse, purchase, and experience groceries, creating a new era of intelligent and personalized shopping. By harnessing the power of machine learning, natural language processing, computer vision, and data analytics, AI grocery shopping systems have redefined convenience, efficiency, and customer satisfaction.

Traditional grocery shopping often involves navigating crowded aisles, searching for desired products, and waiting in long checkout lines. However, AI has emerged as a game-changer, empowering consumers with innovative solutions and enhancing the overall shopping experience. AI-powered grocery shopping platforms leverage sophisticated algorithms and technologies to analyze vast amounts of data, enabling intelligent recommendations, personalized offers, optimized inventory management, and streamlined transactions.

One of the key aspects of AI grocery shopping is personalized experiences. AI algorithms analyze consumer preferences, purchase history, and contextual factors to understand individual tastes and preferences. By doing so, these systems can offer tailored recommendations, suggest relevant products, and present personalized promotions, ultimately enhancing customer satisfaction and loyalty. Through advanced data analytics and pattern recognition, AI systems can uncover hidden correlations and offer products that align with consumers' needs and preferences.

CHAPTER 2

2. SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS:

a) Servers or Cloud Infrastructure

b) Storage Systems

2.2 SOFTWARE REQUIREMENTS:

1. Data Management and Analytics
2. Recommendation Engines
3. Computer Vision Libraries
4. Cloud Services

2.3 SYSTEM ARCHITECTURE:

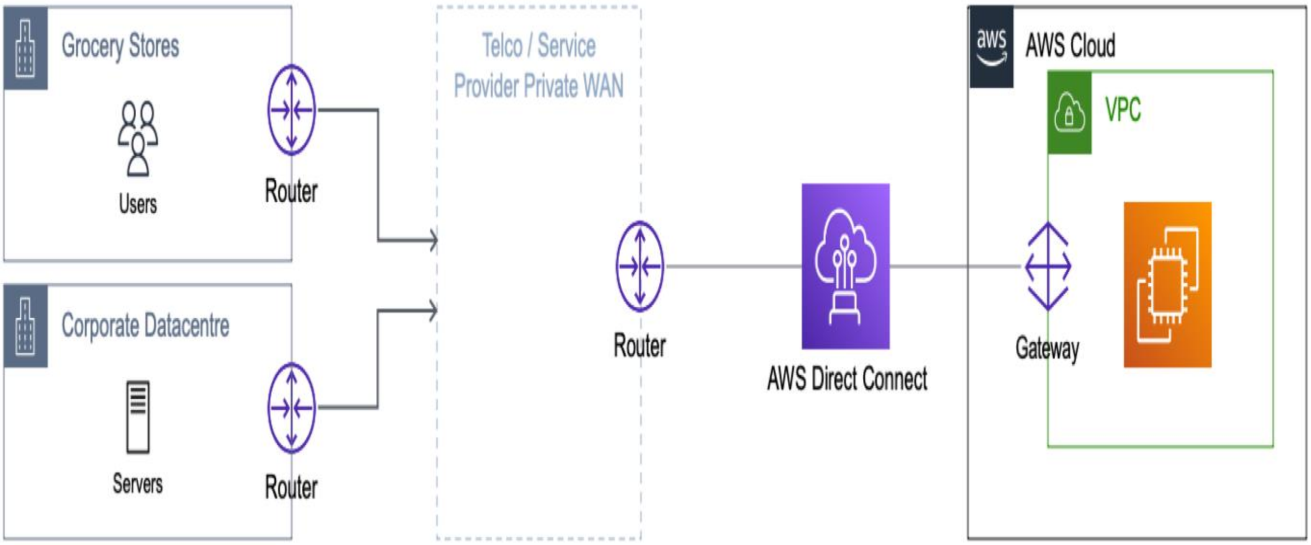


Fig 2.3.1 This figure shows the Architecture for developing AI assistant for grocery shopping

2.4 WORKING DIAGRAM

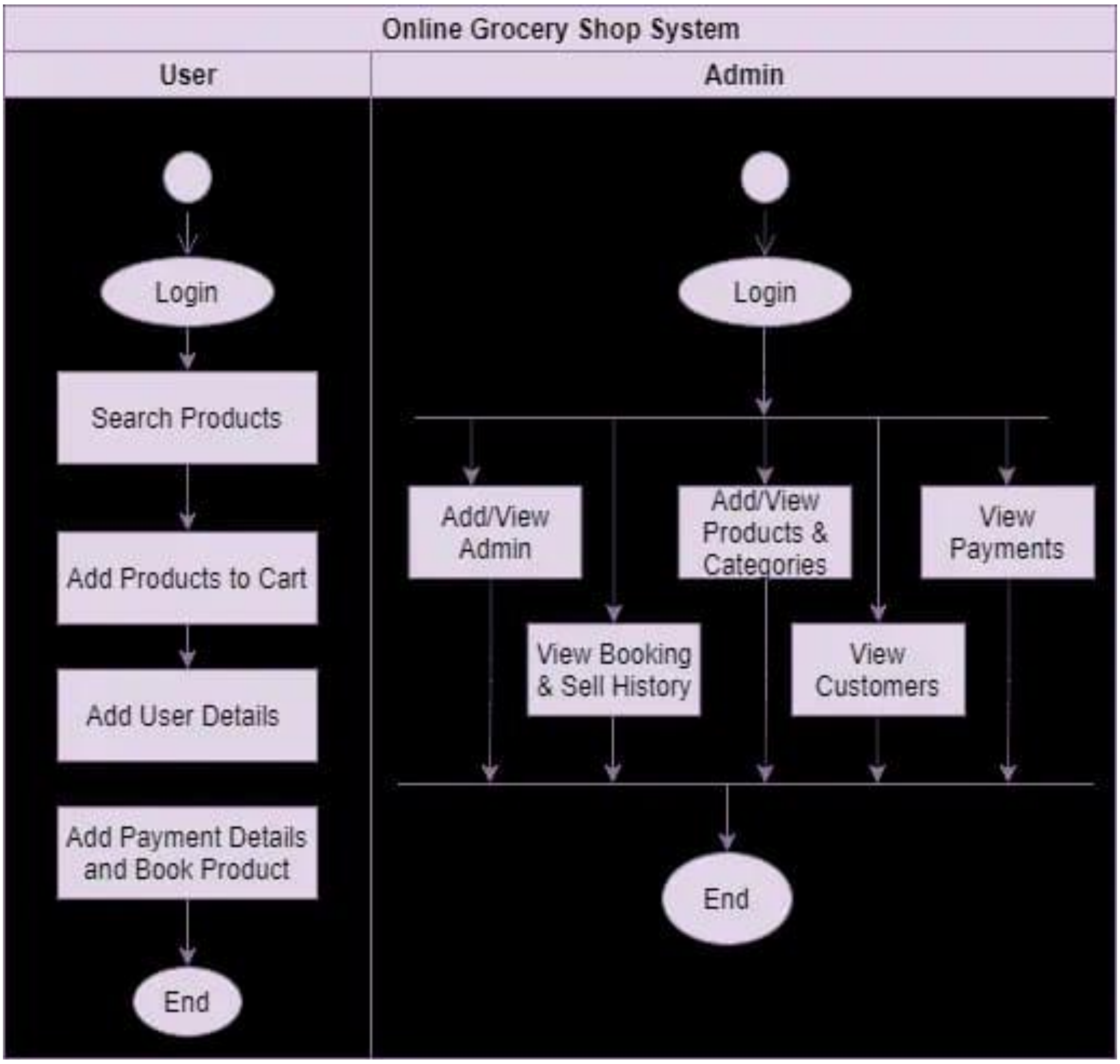

Online Grocery Shop System
User
Admin
Login
Search Products
Add Products to Cart
Add User Details
Add Payment Details and Book Product
End
Login
Add/View Admin
View Booking & Sell History
Add/View Products & Categories
View Customers
View Payments
End

Fig 2.4.1 1This figure shows the Working flow for developing ai assistant for grocery shopping

2.5 PROCESS DIAGRAM

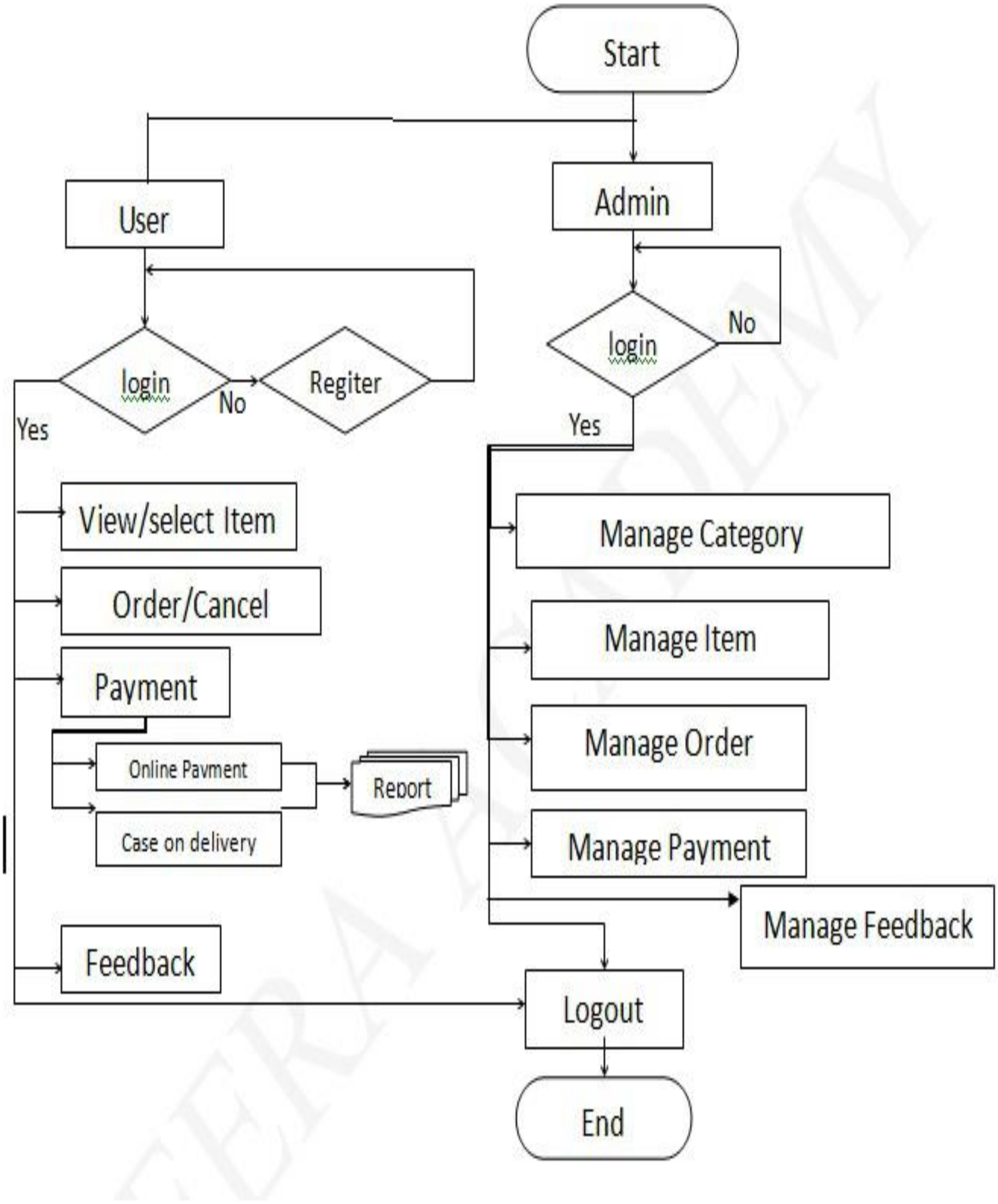


Fig 2.5.1This figure shows the Over all Process developing ai assistant for grocery shopping

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Amazon Fresh: Amazon Fresh is an online grocery shopping service powered by AI. Customers can browse through a wide selection of groceries, add items to their cart, and choose convenient delivery options. Amazon's AI algorithms analyze customer data to provide personalized recommendations and offers. The system also uses smart algorithms for inventory management, ensuring timely delivery and minimizing out-of-stock situations.

Instacart: Instacart is a popular AI-powered grocery delivery platform that partners with local grocery stores. Customers can use the Instacart app or website to place orders, and AI algorithms optimize the route for shoppers who pick and deliver the items.

3.2 DRAWBACKS OF EXISTING SYSTEM:

Lack of Physical Inspection: With AI grocery shopping, customers do not have the opportunity to physically inspect and select fresh produce or evaluate the quality of perishable items. Relying solely on online descriptions and images may lead to dissatisfaction if the received products do not meet expectations.

Limited Product Selection: Although AI grocery shopping platforms aim to provide a wide range of products, there might still be limitations in terms of available brands or specific niche items. Some customers may prefer unique or specialty products that are not readily available through AI grocery shopping platforms.

Delivery Challenges: While delivery is a convenient aspect of AI grocery shopping, it can also pose challenges. Factors like delivery fees, time slots availability, and potential delays or errors in the delivery process can impact customer satisfaction. Additionally, certain remote areas may have limited or no delivery coverage.

Impersonal Experience: Despite attempts to provide personalized recommendations, AI grocery shopping can lack the personal touch of interacting with a knowledgeable store associate. Some customers may miss the human interaction, personalized advice, and the ability to ask questions or get immediate clarifications.

3.3 PROPOSEDSYSTEM:

User Interface:

A user-friendly interface accessible through web or mobile applications. It allows customers to browse products, search for specific items, view details such as prices and nutritional information, and add products to their virtual shopping carts.

Product Recommendation Engine:

An AI-powered recommendation engine that suggests products based on customer preferences, purchase history, and browsing behavior. It can employ collaborative filtering techniques, content-based filtering, or a combination of both to provide personalized recommendations.

Inventory Management System:

A backend system that manages the inventory, tracks stock levels, and updates product availability in real-time. It integrates with the point of sale (POS) system to ensure accurate inventory information and avoid out-of-stock situations.

Virtual Assistant:

A chatbot or virtual assistant powered by natural language processing (NLP) capabilities. It assists customers with queries, provides product information, helps with order placement and payment processing, and offers support throughout the shopping process.

Payment and Security Integration:

Integration with secure payment gateways to facilitate seamless and secure transactions. It should support various payment options such as credit/debit cards, mobile wallets, and online banking.

Delivery Management System:

A system that manages the logistics of product delivery. It includes features such as order tracking, delivery scheduling, and coordination with delivery personnel or third-party delivery services.

Data Analytics and Insights:

A backend analytics system that collects and analyzes customer data, purchase patterns, and other relevant metrics. It helps retailers gain insights into customer behavior, preferences, and trends, enabling them to make data-driven decisions for inventory management, marketing strategies, and personalized promotions.

Integration with Physical Stores:

Optionally, the system can integrate with physical stores to provide options like click-and-collect or curbside pickup. This allows customers to order online and collect their groceries from a nearby store location at their convenience.

Feedback and Ratings:

An integrated feedback and rating system that enables customers to provide feedback on products and overall shopping experience. It helps improve customer satisfaction and provides valuable insights for retailers.

3.4 ADVANTAGES:

Time-saving:

AI grocery shopping eliminates the need for physically visiting a store and waiting in long checkout lines. Customers can browse and purchase products online, saving valuable time and effort.

Personalized recommendations:

AI algorithms can analyze customer preferences, purchase history, and browsing patterns to provide personalized product recommendations. This helps customers discover new products and ensures they find items that align with their tastes and dietary requirements.

Improved inventory management:

Retailers can use AI to optimize their inventory management processes. By analyzing historical data, AI systems can accurately predict demand, ensuring that the right products are stocked in the right quantities. This minimizes the chances of out-of-stock situations and reduces waste due to overstocking.

Streamlined shopping experience:

AI-powered grocery shopping platforms often feature intuitive interfaces that make it easy for customers to search for products, compare prices, and place orders. Additionally, AI chatbots and virtual assistants can provide real-time assistance, answering queries and guiding customers through the shopping process.

Enhanced efficiency and accuracy:

AI algorithms can automate various aspects of grocery shopping, such as order processing, payment verification, and delivery scheduling. This reduces the chances of human errors and ensures a smoother overall experience for both customers and retailers.

Improved accessibility:

AI grocery shopping can benefit individuals with mobility challenges or those living in remote areas with limited access to physical stores. Online platforms enable them to conveniently purchase groceries from the comfort of their homes, ensuring they have access to a wide range of products.

Cost savings:

AI-driven promotions can be tailored to individual customers based on their preferences and purchasing behavior. This enables retailers to offer personalized deals, leading to potential cost savings for customers.

CHAPTER 4

SYSTEM DESCRIPTION

4.1 OVERVIEW OF THE PROJECT

AI grocery shopping refers to the integration of artificial intelligence technologies into the process of purchasing groceries. This innovative approach aims to enhance the shopping experience by providing personalized recommendations, streamlining the purchasing process, and improving overall efficiency. By leveraging AI, the project seeks to revolutionize the way people shop for groceries, offering convenience, cost-effectiveness, and tailored solutions.

Personalized Recommendations:

Develop an AI system that analyzes individual preferences, dietary restrictions, and shopping patterns to provide personalized product recommendations to users.

Smart Shopping Assistance:

Create an intelligent virtual assistant capable of guiding users through the shopping process, answering queries, and offering suggestions based on real-time inventory and pricing information.

Seamless User Experience:

Design a user-friendly mobile application or web platform that integrates AI capabilities to provide a seamless and intuitive grocery shopping experience.

Efficient Inventory Management:

Implement AI algorithms to optimize inventory management for grocery retailers, ensuring sufficient stock levels, minimizing waste, and predicting demand accurately. Cost Optimization: Utilize AI algorithms to identify cost-saving opportunities for shoppers, such as suggesting alternative brands or offering discounts and coupons based on user preferences.

Integration with Delivery Services:

Collaborate with local delivery services or establish partnerships to integrate seamless delivery options within the AI grocery shopping platform, facilitating a complete end-to-end solution. Continuous Improvement: Implement mechanisms for gathering user feedback and leveraging data analytics to continuously enhance the AI system's accuracy, performance, and relevance.

Data Privacy and Security:

Address concerns related to user data privacy and implement robust security measures to protect personal information. Accuracy and Relevance: Continuously improve AI algorithms to deliver accurate product recommendations and relevant information, considering factors like user preferences, dietary restriction sand budget.

CHAPTER 5

DIAGRAMS

5.1 Architecture Block Diagram:

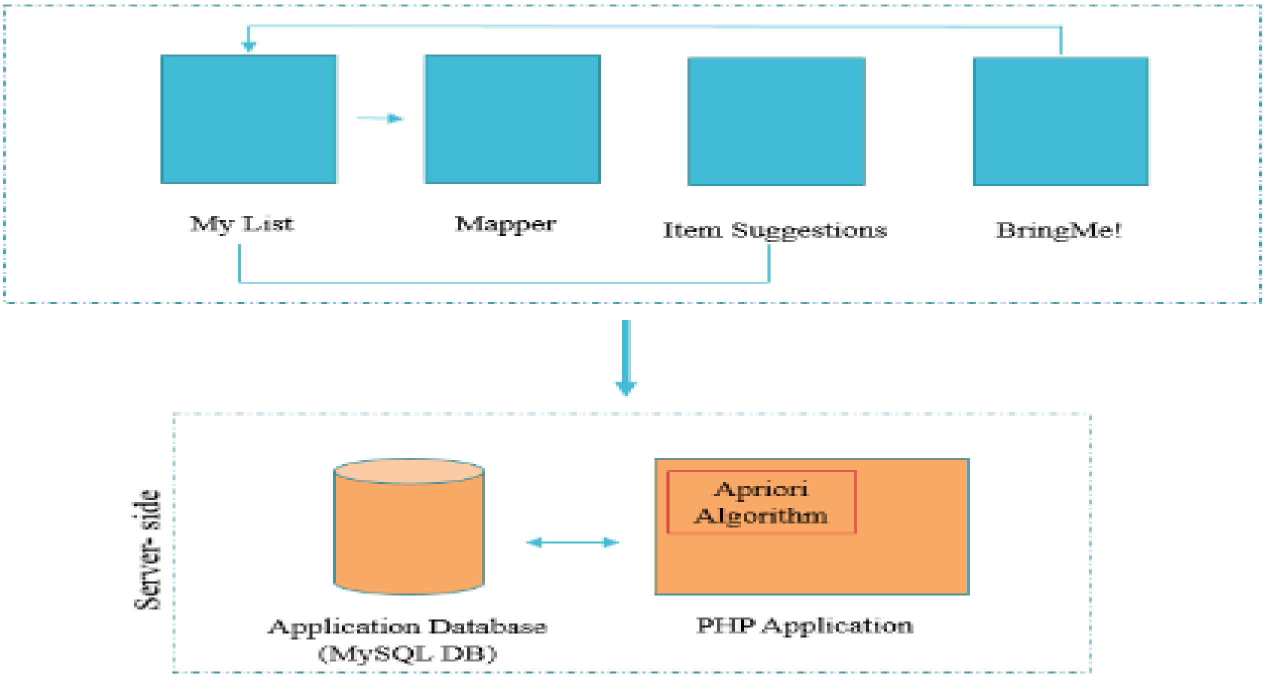


Fig 5.1.1 This figure shows the Architecture for developing AI assistant for grocery shopping

5.2 Data Flow Diagram 1:

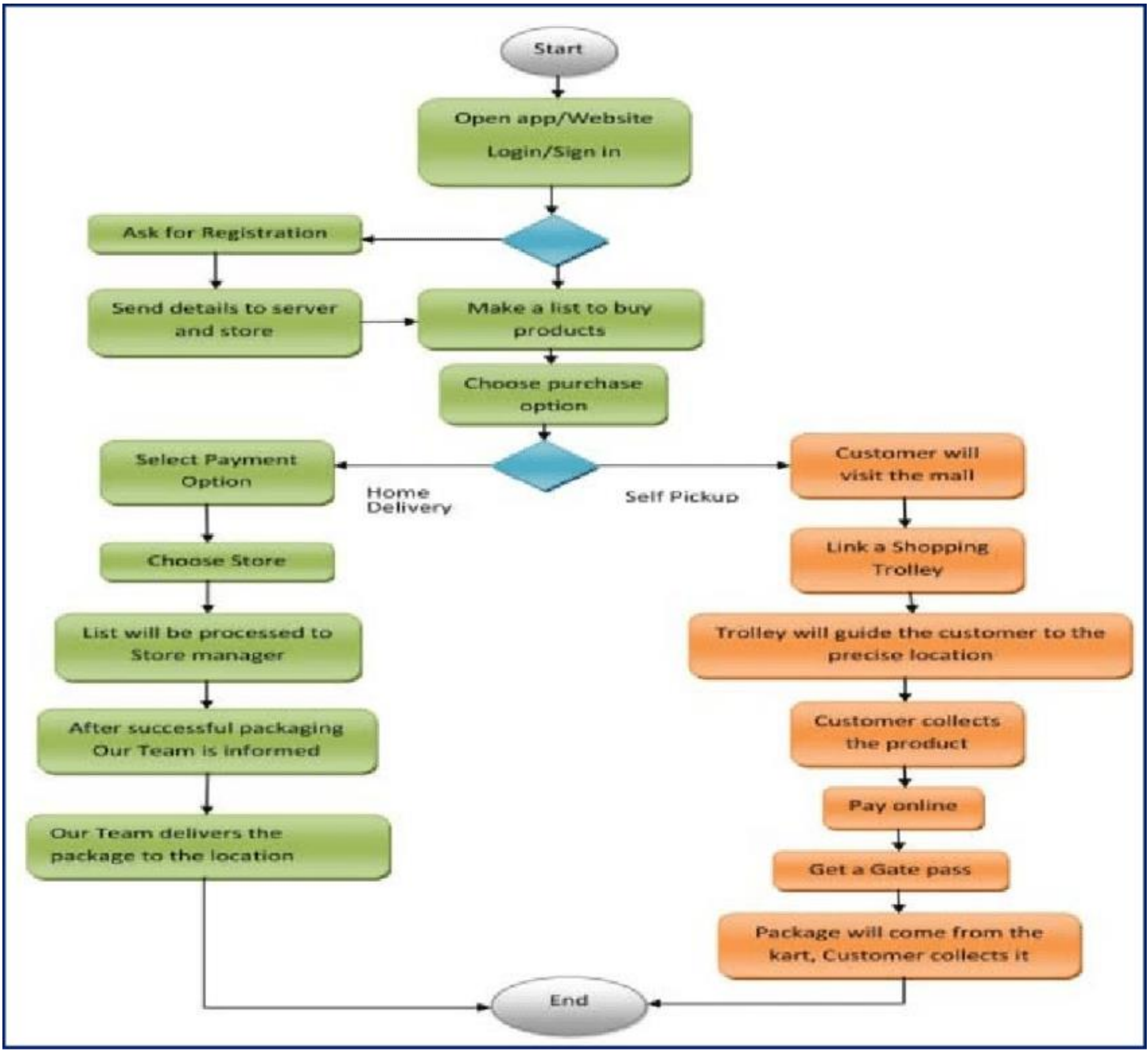

Start
Open app/Website
Login/Sign in
Ask for Registration
Send details to server and store
Make a list to buy products
Choose purchase option
Select Payment Option
Home Delivery
Self Pickup
Customer will visit the mall
Choose Store
Link a Shopping Trolley
List will be processed to Store manager
Trolley will guide the customer to the precise location
After successful packaging Our Team is informed
Customer collects the product
Pay online
Our Team delivers the package to the location
Get a Gate pass
Package will come from the kart, Customer collects it
End

Fig 5.2.11This figure shows the Live Data Flow for developing AI for shopping assistant

Data Flow Diagram 2:

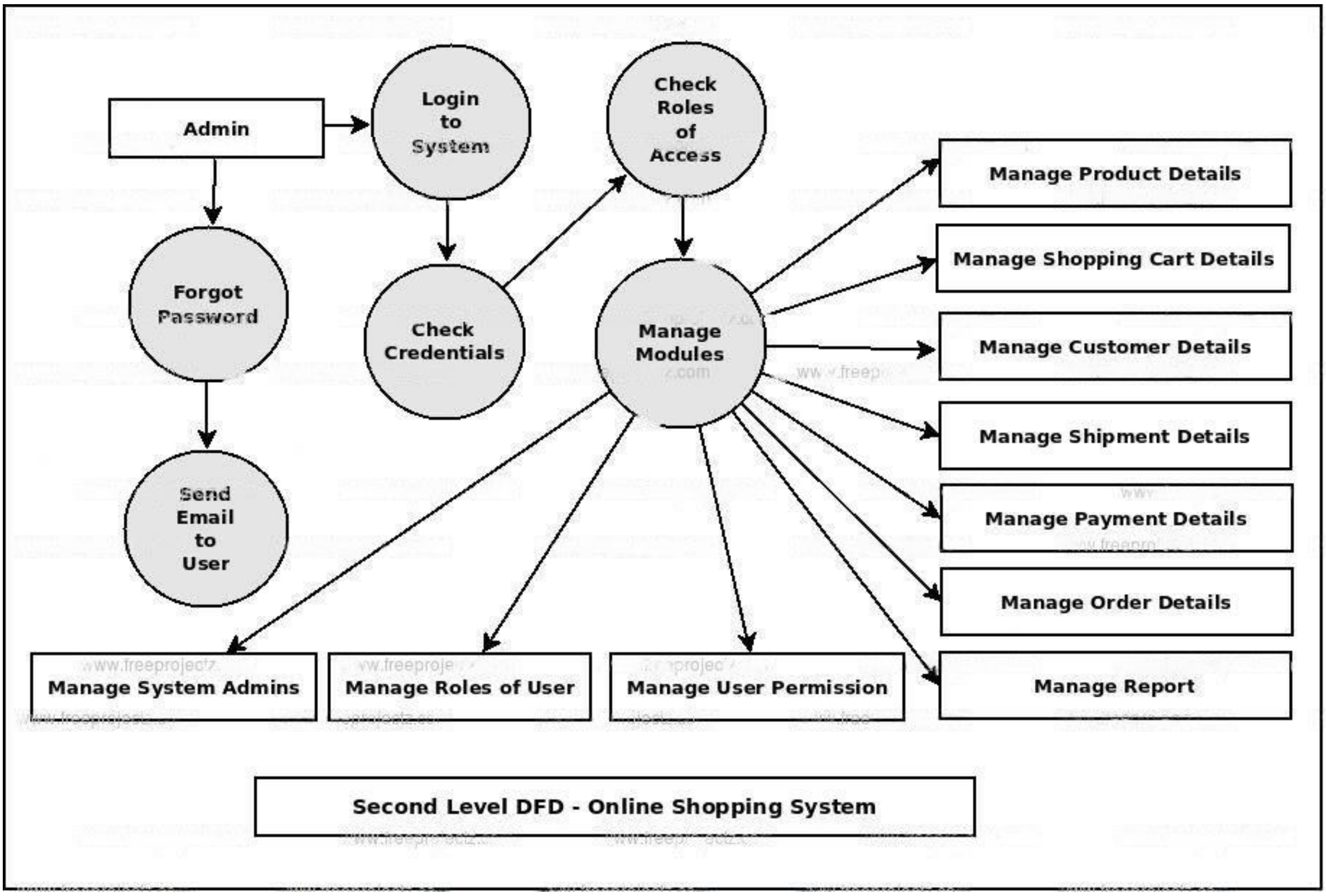


Fig 5.2.21This figure shows the Collected Information From developing AI for shopping assistant

CHAPTER 6

TECHNOLOGIES USED

6.1 PYTHON:

Developing an AI shopping assistant using Python involves creating a program that can understand and respond to user queries, provide product recommendations, and facilitate the shopping experience. Here's a high-level description of the development process:

Data Collection:

Gather relevant data for training the AI shopping assistant. This includes product information, user preferences, customer reviews, and any other relevant data sources.

Natural Language Processing (NLP):

Utilize NLP techniques to process and understand user queries. This involves tasks such as tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis.

Intent Recognition:

Develop an intent recognition system to identify the user's intent from their queries.

Product Recommendation Engine:

Implement a recommendation engine that suggests relevant products based on user preferences and historical data.

User Interface:

Design and develop a user-friendly interface for the shopping assistant. This could be a chatbot interface or a voice-controlled interface, depending on the target platform and user interaction preferences.

6.2 PYTHON FLASK:

Set up Flask:

Begin by installing Flask, a lightweight web framework in Python. Create a new Flask project and configure the necessary dependencies.

Design User Interface:

Plan and design the user interface for the shopping assistant. This can include a chatbot-like interface or a form-based interface where users can input their queries and receive responses. Flask provides flexible templating options to create dynamic HTML templates for the interface.

NLP Processing:

Implement natural language processing (NLP) techniques to process and understand user queries. Utilize Python libraries like NLTK or spaCy for tasks such as tokenization, part-of-speech tagging, and named entity recognition. These libraries enable you to extract important information from user queries.

Intent Recognition:

Develop an intent recognition system using machine learning techniques. Train a model to classify user queries into different intents, such as searching for products, checking prices, or adding items to the cart. You can use libraries like scikit-learn or TensorFlow to build and train the intent recognition model.

Product Recommendation Engine:

Integrate a product recommendation engine into the shopping assistant. This engine should provide personalized recommendations based on user preferences and historical data. Implement collaborative filtering or content-based filtering.

Backend Development:

Implement the backend logic of the shopping assistant using Flask. Define routes and endpoints that handle user requests and interact with the AI models. Use Flask's routing capabilities to map URLs to specific functions that process user queries and generate responses.

Integration with E-commerce Platforms:

Integrate the shopping assistant with e-commerce platforms or APIs to fetch real-time product information, prices, and inventory. Use Flask's ability to make HTTP requests to communicate with external APIs or web scraping libraries to extract data from websites

6.3 SQL LITE:

Database Design:

Design the database schema to represent the necessary entities and relationships. Identify the tables needed to store information such as products, user profiles, user preferences, order history, and any other relevant data. Define the appropriate columns and data types for each table.

Database Creation:

Set up an SQLite database using Python's SQLite module or an ORM (Object-Relational Mapping) library such as SQLAlchemy. Create the necessary tables and define any constraints or indexes required for efficient data retrieval.

Data Population:

Populate the database with initial data. This can include product information, user profiles, and sample data to test and develop the AI shopping assistant's functionalities. You can either manually insert the data or create scripts to import

Data Access and Manipulation:

Develop functions or methods to interact with the SQLite database. Use SQL queries or ORM methods to retrieve, update, insert, and delete data as needed. Implement the necessary CRUD (Create, Read, Update, Delete) operations to manage the data.

User Profile Management:

Implement features for managing user profiles, such as creating new profiles, updating preferences, and storing user-specific information. Use SQL queries or ORM methods to handle user-related operations and maintain the integrity of the data.

Product Catalog Management:

Build functionality to manage the product catalog. Implement features such as adding new products, updating product details, and retrieving product information based on user queries. Utilize SQL queries or ORM methods to handle product-related operations efficiently.

Integration with AI Components:

Integrate the SQLite database with AI components such as natural language processing (NLP) models, intent recognition systems, and recommendation engines. Retrieve relevant data from the database to enhance the assistant's responses and recommendations based on user queries and preferences.

Security and Performance Optimization:

Implement security measures to protect the SQLite database from unauthorized access. Utilize techniques like parameterized queries to prevent SQL injection attacks. Optimize database performance by creating appropriate indexes, optimizing queries, and using transactions when necessary.

Testing and Deployment:

Test the AI shopping assistant to ensure proper functionality and data retrieval from the SQLite database. Perform various test scenarios, including querying for products, updating user preferences, and verifying data consistency. Once testing is complete, deploy the application with the SQLite database to a suitable production environment.

Continuous Improvement:

Collect user feedback, monitor the assistant's performance, and make iterative improvements based on user needs and preferences. Consider periodically updating the SQLite database with new product data, user feedback, and any other relevant updates.

6.4 XAMPP

To develop an AI shopping assistant using XAMPP, you would typically follow these steps:

Install XAMPP:

XAMPP is a popular software package that includes Apache, MySQL, PHP, and Perl. It provides a local development environment for web applications. Start by downloading and installing XAMPP on your computer.

Set up a web server:

Once XAMPP is installed, start the Apache web server from the XAMPP control panel. This will allow you to run your shopping assistant application locally.

Create a database:

Use the phpMyAdmin tool included with XAMPP to create a MySQL database for your shopping assistant. Define the necessary tables and fields to store product information, user data, and any other relevant information.

Develop the AI component:

You will need to integrate an AI system into your shopping assistant. There are various options available, such as using pre-trained models or developing your own using machine learning frameworks like TensorFlow or PyTorch. Train the AI model on relevant data to enable it to provide recommendations and assist with shopping decisions.

Design the user interface:

Create a web-based user interface using HTML, CSS, and JavaScript. This interface will allow users to interact with the shopping assistant. Implement features like product search, browsing categories, adding items to a cart

Connect to the database:

Use PHP or another server-side scripting language to establish a connection between your web application and the MySQL database. Implement database queries to fetch and store data as needed.

Implement AI functionality:

Integrate the AI component into your application. Depending on your chosen approach, this may involve making API calls to a pre-trained model or running the model directly on your server. Implement the necessary logic to process user queries and generate AI-powered responses and recommendations.

Test and debug:

Thoroughly test your shopping assistant application, both for functionality and usability. Identify and fix any bugs or issues that may arise during testing.

Deploy and refine:

Once your shopping assistant is working as expected, deploy it to a production environment. Continuously monitor and gather user feedback to improve the AI algorithms and user experience over time.Remember to consider legal and ethical considerations when developing an AI shopping assistant, such as privacy concerns, data security, and transparency in AI recommendations. Make sure you have the necessary components and tools installed for AI development.

6.5 Black Box AI with Visual Studio

Set up Visual Studio:

Install Visual Studio, a popular integrated development environment (IDE) for creating applications. Make sure you have the necessary components and tools installed for AI development.

Define requirements:

Clearly define the requirements and functionalities of your shopping assistant. Determine what AI capabilities you want to integrate, such as natural language processing (NLP) for user queries, recommendation systems, or image recognition for product identification.

Choose the AI framework:

Select an AI framework that is compatible with Visual Studio. Some popular options include TensorFlow, PyTorch, and Microsoft Cognitive Toolkit (CNTK). Consider the specific requirements of your project and choose the framework that best suits your needs.

Prepare training data:

If your shopping assistant requires machine learning or deep learning algorithms, you'll need to gather and preprocess relevant training data. This may include product information, user preferences, historical transaction data, or image datasets for object recognition.

Develop AI models:

Use the chosen AI framework within Visual Studio to develop and train your AI models. This involves designing the architecture of the neural network, defining loss functions, and optimizing hyperparameters. Train the models using the prepared training data to make them learn and generalize from the provided examples**.**

Integrate the AI models:

Once your AI models are trained, integrate them into your shopping assistant application. Use the appropriate APIs or libraries provided by the AI framework to load and utilize the trained models. Implement the necessary logic to process user inputs, make predictions or recommendations, and generate responses.

Design the user interface:

Create a user-friendly interface for your shopping assistant using Visual Studio's design tools and programming languages like C# or VB.NET. Design the layout, implement search functionality, display product information, and incorporate features such as cart management and checkout.

Connect to external data sources:

If your shopping assistant needs access to external data sources, such as product catalogs or user profiles, establish connections and implement the necessary data retrieval and integration mechanisms. This could involve working with APIs, web scraping, or database integration.

Test and debug:

Thoroughly test your shopping assistant application, both for functionality and AI performance. Validate that the AI models provide accurate recommendations, handle user queries correctly, and behave as expected. Debug any issues or errors that arise during testing

Deploy and refine:

Once your shopping assistant is functioning properly, deploy it to a production environment. Monitor user interactions, collect feedback, and analyze performance metrics to identify areas for improvement. Continuously refine and update your AI models to enhance the shopping assistant's capabilities.

Remember to adhere to ethical and legal considerations when developing an AI shopping assistant, such as privacy protection, data security, and transparency in AI-driven recommendations.

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENTS

7.1 CONCLUSION

In conclusion, developing an AI assistant for grocery shopping can greatly enhance the shopping experience for users. By leveraging artificial intelligence technologies, such as natural language processing, recommendation systems, and image recognition, an AI assistant can provide personalized and efficient assistance throughout the grocery shopping process.

The AI assistant can help users create shopping lists, suggest relevant products based on preferences and dietary restrictions, and even offer recipe recommendations based on available ingredients. It can assist in locating products within the store, provide real-time price comparisons, and offer alternatives for out-of-stock items.

By utilizing machine learning algorithms, the AI assistant can learn from user interactions and continuously improve its recommendations and predictions. It can adapt to individual shopping patterns, adapt to changing preferences, and provide tailored suggestions that align with the user's needs.

Furthermore, the integration of visual recognition capabilities allows the AI assistant to identify products through images, making it easier for users to find and select items, especially when they are unsure of the exact name or description. Overall, an AI assistant for grocery shopping can save users time, reduce decision-making stress, and improve overall shopping efficiency. It can streamline the process, offer personalized assistance, and enhance the overall shopping experience. With advancements in AI technologies and the availability of development tools like Black Box AI and Visual Studio, creating such an AI assistant is within reach for developers and can significantly benefit both shoppers and retailers alike.

7.2 FUTURE ENHANCEMENTS

Voice Interface:
Implement voice recognition capabilities to enable users to interact with the AI assistant using natural language voice commands. This allows for hands-free operation and makes the assistant more accessible and user-friendly.

Integration with Smart Home Devices: Connect the AI assistant to smart home devices, such as smart refrigerators or pantry systems. This integration enables the assistant to monitor inventory levels, suggest items for restocking, and even place automated orders for frequently used products.

Personalized Recommendations:
Enhance the recommendation system by leveraging user data and preferences. Incorporate machine learning techniques to understand individual shopping habits, dietary restrictions, and favorite brands. This allows the assistant to provide more accurate and personalized recommendations tailored to each user's specific needs.

Integration with Loyalty Programs:
Integrate the AI assistant with grocery store loyalty programs. This enables users to access personalized discounts, offers, and promotions based on their shopping history. The assistant can automatically apply relevant discounts or suggest alternative products with better deals.

Nutritional Information and Allergen Alerts:
Provide detailed nutritional information for products and offer allergen alerts for users with specific dietary restrictions or allergies. The AI assistant can analyze product labels or ingredient lists to identify potential allergens and suggest alternatives when necessary.

Real-time Store Information:

Integrate with store APIs or databases to provide real-time information about product availability, current pricing, and promotions. This ensures that users receive up-to-date information and can make informed decisions while shopping.

Integration with Delivery Services: Enable users to order groceries directly through the AI assistant and integrate with popular delivery services. This allows for a seamless transition from shopping assistance to home delivery, providing a complete end-to-end solution for users.

Continuous Learning and Feedback:

Implement mechanisms for continuous learning and feedback. Allow users to provide ratings and feedback on suggested products, recipes, or overall user experience. Utilize this feedback to improve the AI assistant's performance and accuracy over time.

Multi-platform Support:

Develop applications or extensions for different platforms, such as mobile devices, smart speakers, or web browsers. This ensures that users can access the AI assistant from various devices and platforms, providing flexibility and convenience.

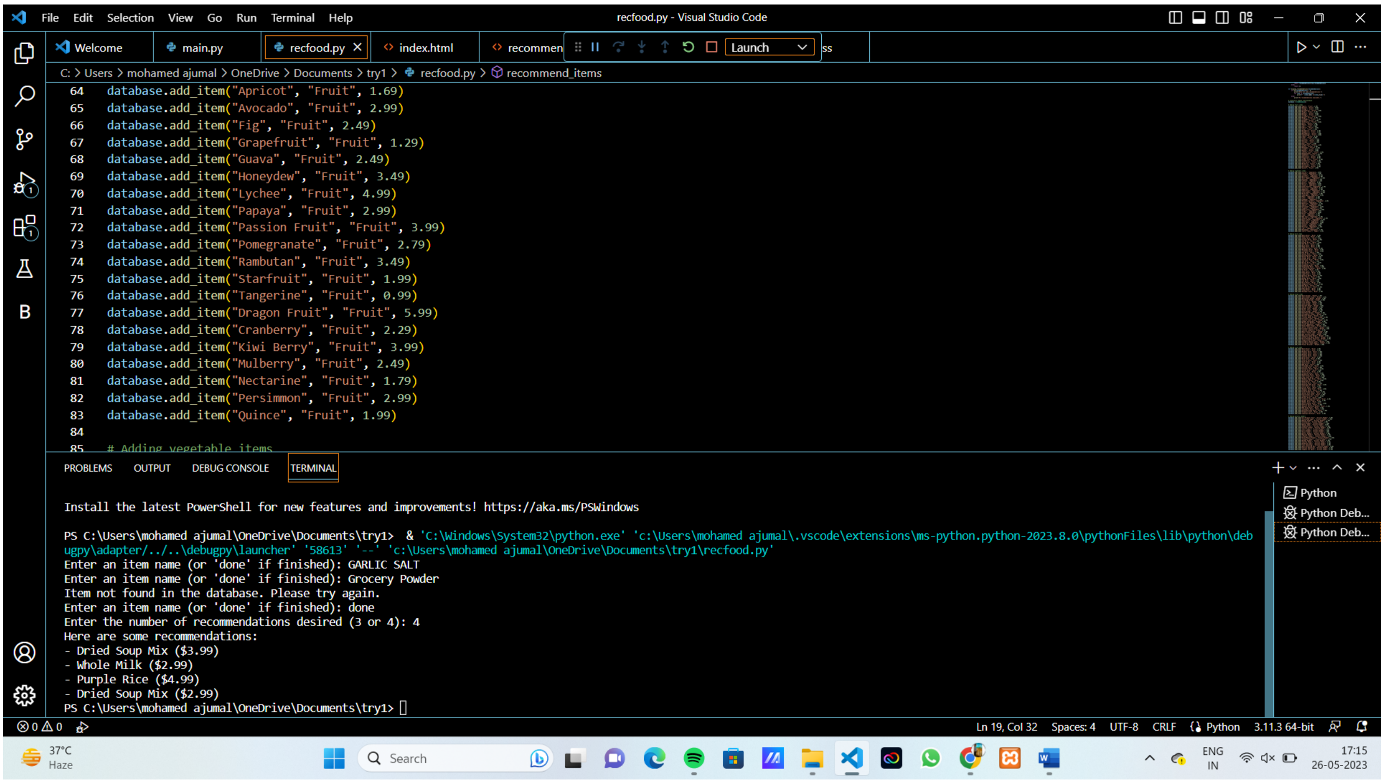
Integration with Payment Systems:

Enable users to make secure payments directly through the AI assistant. Integrate with popular payment gateways or digital wallet platforms to streamline the checkout process and provide a seamless end-to-end shopping experience.

By incorporating these enhancements, an AI assistant for grocery shopping can become even more valuable and indispensable for users, offering personalized recommendations, streamlined operations, and a seamless shopping experience both in-store and online.

APPENDDIX 1

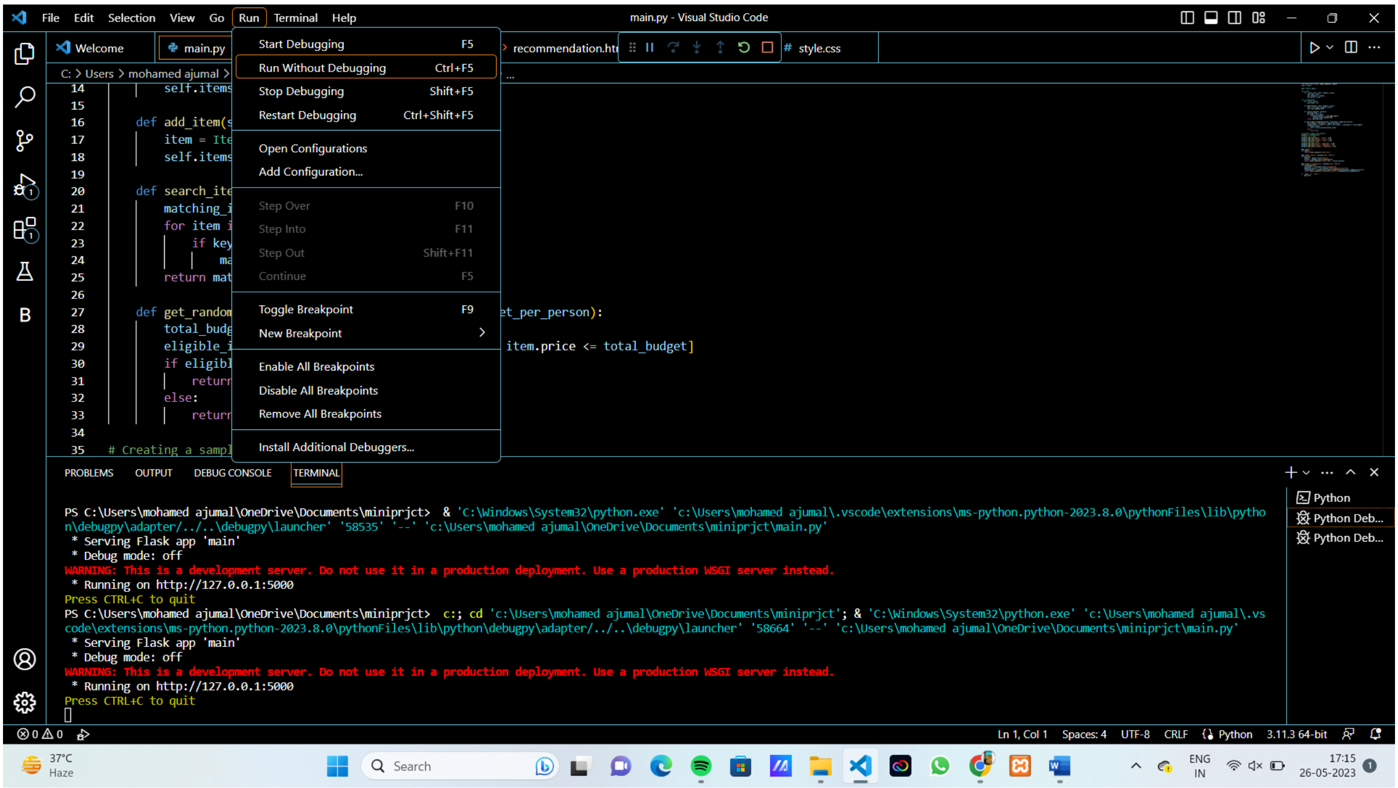
OUTPUT



File Edit Selection View Go Run Terminal Help
recfood.py - Visual Studio Code
Welcome
main.py
recfood.py
index.html
recommen
Launch
C: > Users > mohamed ajumal > OneDrive > Documents > try1 > recfood.py > recommend_items
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Python
Python Deb...
Python Deb...
Ln 19, Col 32 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit
37°C Haze
Search
ENG IN
17:15 26-05-2023

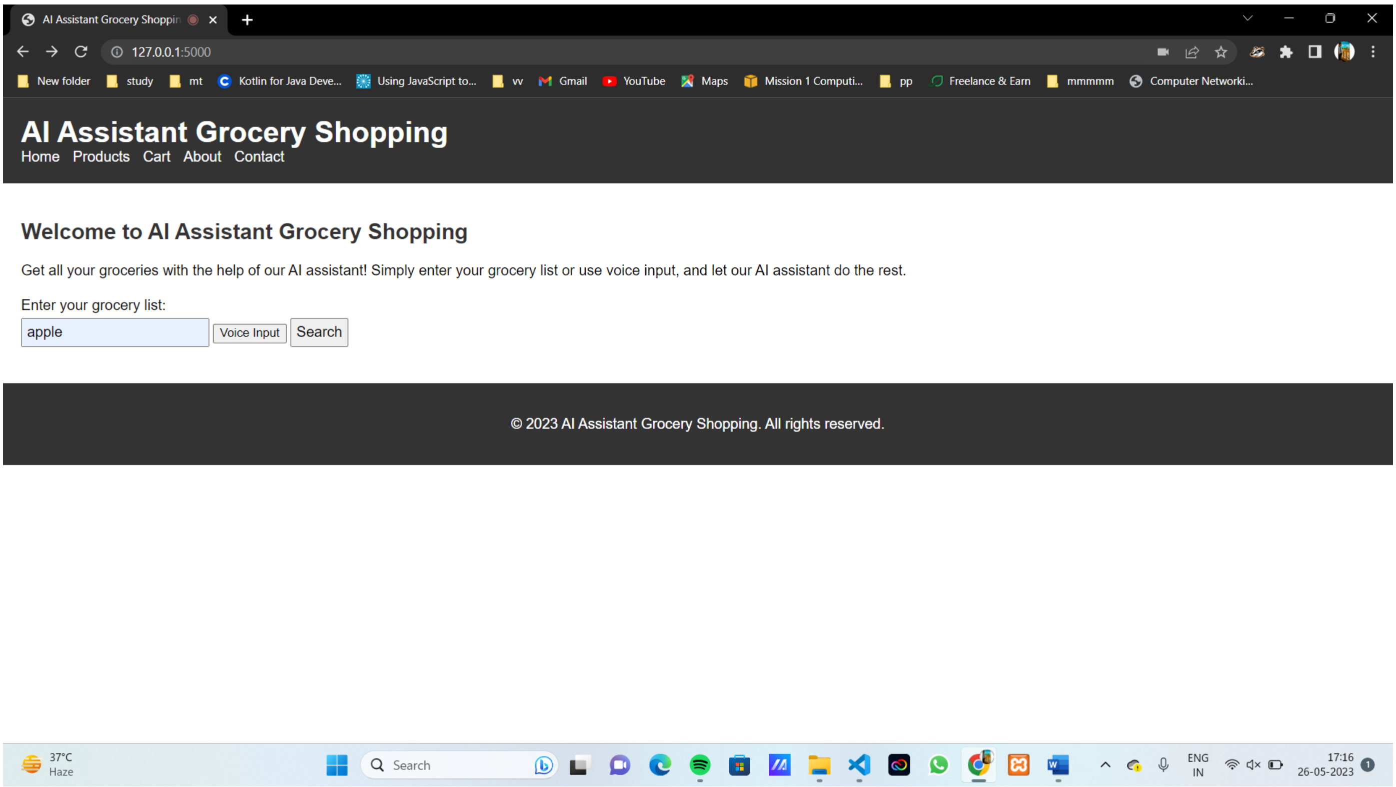
```
database.add_item("Apricot", "Fruit", 1.69)
database.add_item("Avocado", "Fruit", 2.99)
database.add_item("Fig", "Fruit", 2.49)
database.add_item("Grapefruit", "Fruit", 1.29)
database.add_item("Guava", "Fruit", 2.49)
database.add_item("Honeydew", "Fruit", 3.49)
database.add_item("Lychee", "Fruit", 4.99)
database.add_item("Papaya", "Fruit", 2.99)
database.add_item("Passion Fruit", "Fruit", 3.99)
database.add_item("Pomegranate", "Fruit", 2.79)
database.add_item("Rambutan", "Fruit", 3.49)
database.add_item("Starfruit", "Fruit", 1.99)
database.add_item("Tangerine", "Fruit", 0.99)
database.add_item("Dragon Fruit", "Fruit", 5.99)
database.add_item("Cranberry", "Fruit", 2.29)
database.add_item("Kiwi Berry", "Fruit", 3.99)
database.add_item("Mulberry", "Fruit", 2.49)
database.add_item("Nectarine", "Fruit", 1.79)
database.add_item("Persimmon", "Fruit", 2.99)
database.add_item("Quince", "Fruit", 1.99)

# Adding vegetable items
```

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\mohamed ajumal\OneDrive\Documents\try1>  & 'C:\Windows\System32\python.exe' 'c:\Users\mohamed ajumal\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\deb
ugpy\adapter/../..\debugpy\launcher' '58613' '--' 'c:\Users\mohamed ajumal\OneDrive\Documents\try1\recfood.py'
Enter an item name (or 'done' if finished): GARLIC SALT
Enter an item name (or 'done' if finished): Grocery Powder
Item not found in the database. Please try again.
Enter an item name (or 'done' if finished): done
Enter the number of recommendations desired (3 or 4): 4
Here are some recommendations:
- Dried Soup Mix ($3.99)
- Whole Milk ($2.99)
- Purple Rice ($4.99)
- Dried Soup Mix ($2.99)
PS C:\Users\mohamed ajumal\OneDrive\Documents\try1> 
```



```
        self.items

    def add_item(s
        item = Ite
        self.items

    def search_ite
        matching_i
        for item i
            if key
                ma
        return mat

    def get_random                              et_per_person):
        total_budg
        eligible_i                              item.price <= total_budget]
        if eligibl
            returr
        else:
            returr

# Creating a sampl
```

```
PS C:\Users\mohamed ajumal\OneDrive\Documents\miniprjct>  & 'C:\Windows\System32\python.exe' 'c:\Users\mohamed ajumal\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\pytho
n\debugpy\adapter/../..\debugpy\launcher' '58535' '--' 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct\main.py'
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
PS C:\Users\mohamed ajumal\OneDrive\Documents\miniprjct>  c:; cd 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct'; & 'C:\Windows\System32\python.exe' 'c:\Users\mohamed ajumal\.vs
code\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '58664' '--' 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct\main.py'
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



AI Assistant Grocery Shopping

Home Products Cart About Contact

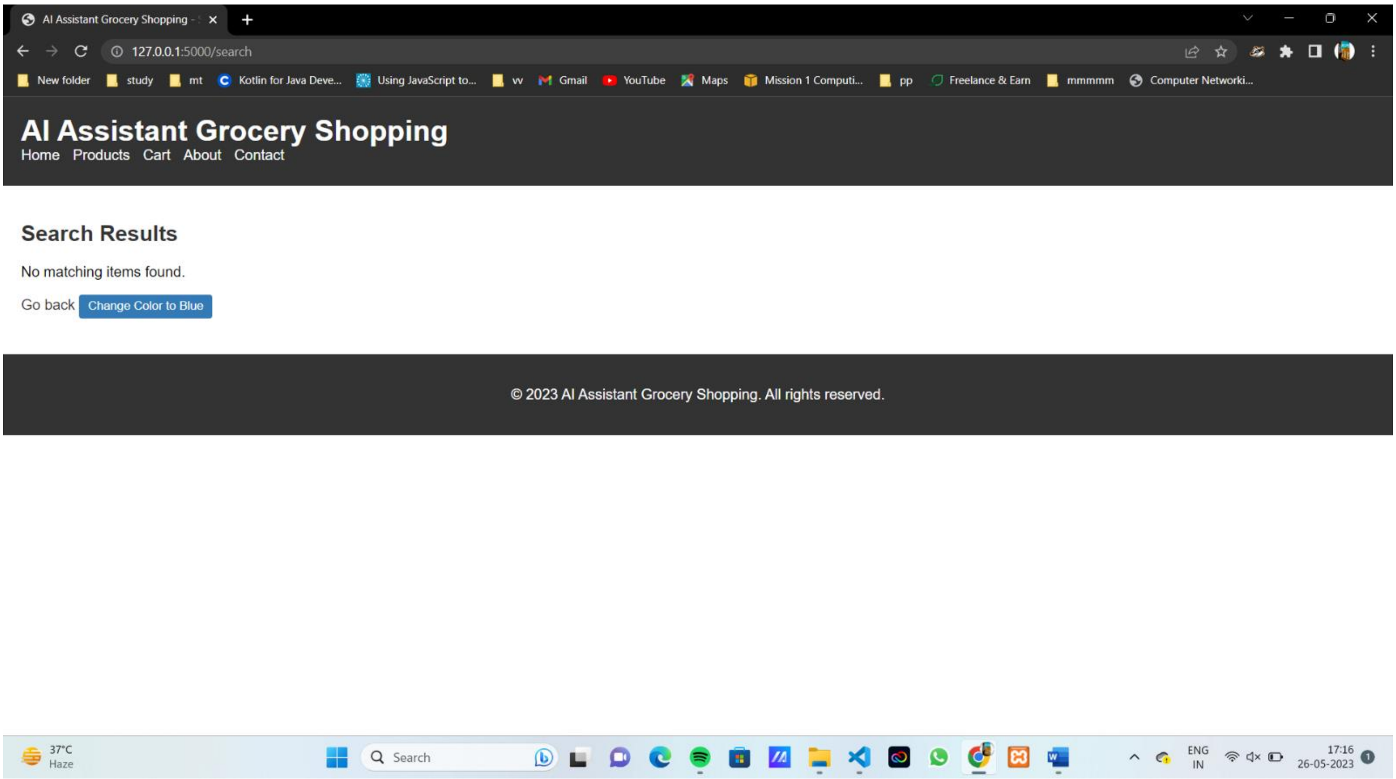
Welcome to AI Assistant Grocery Shopping

Get all your groceries with the help of our AI assistant! Simply enter your grocery list or use voice input, and let our AI assistant do the rest.

Enter your grocery list:

apple

© 2023 AI Assistant Grocery Shopping. All rights reserved.



AI Assistant Grocery Shopping

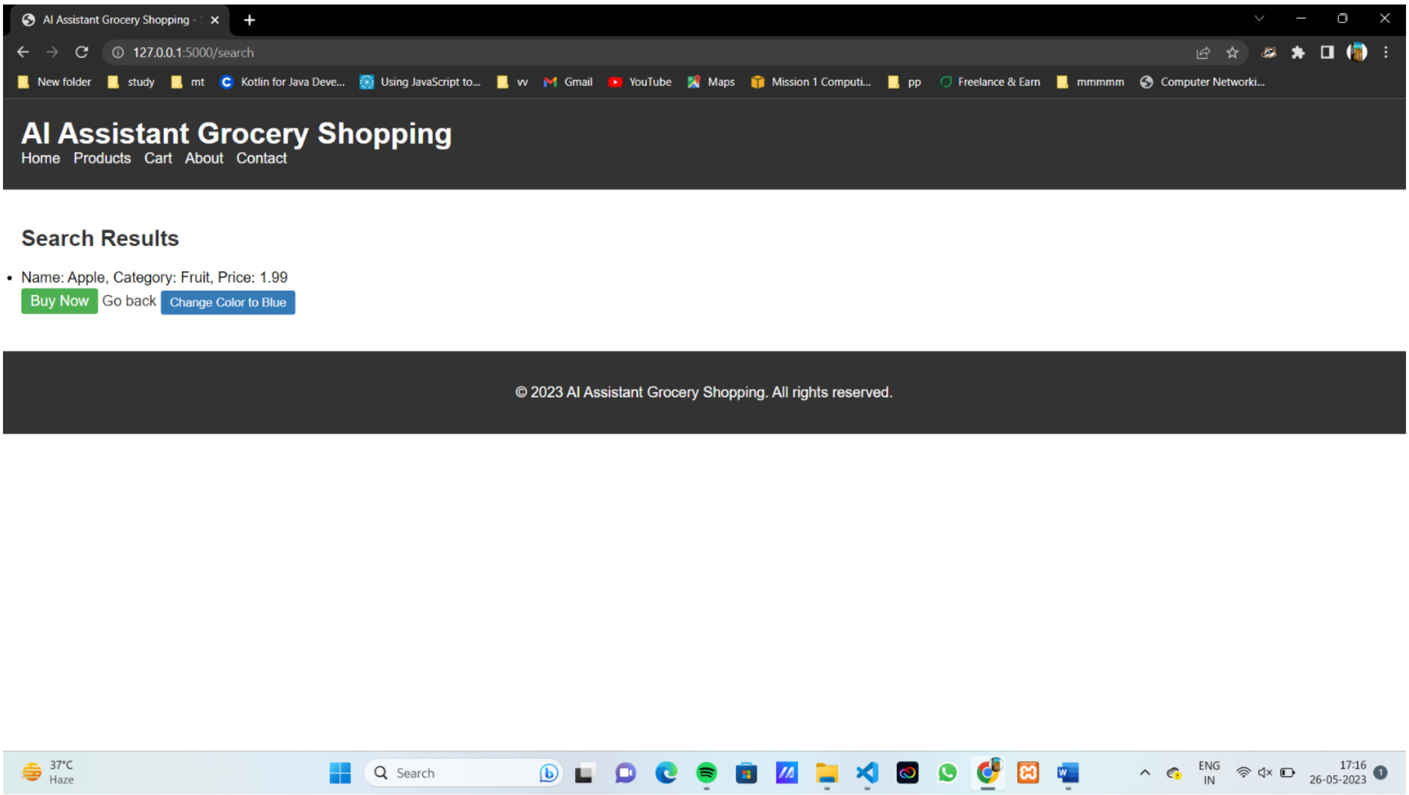
Home Products Cart About Contact

Search Results

No matching items found.

Go back

© 2023 AI Assistant Grocery Shopping. All rights reserved.



AI Assistant Grocery Shopping

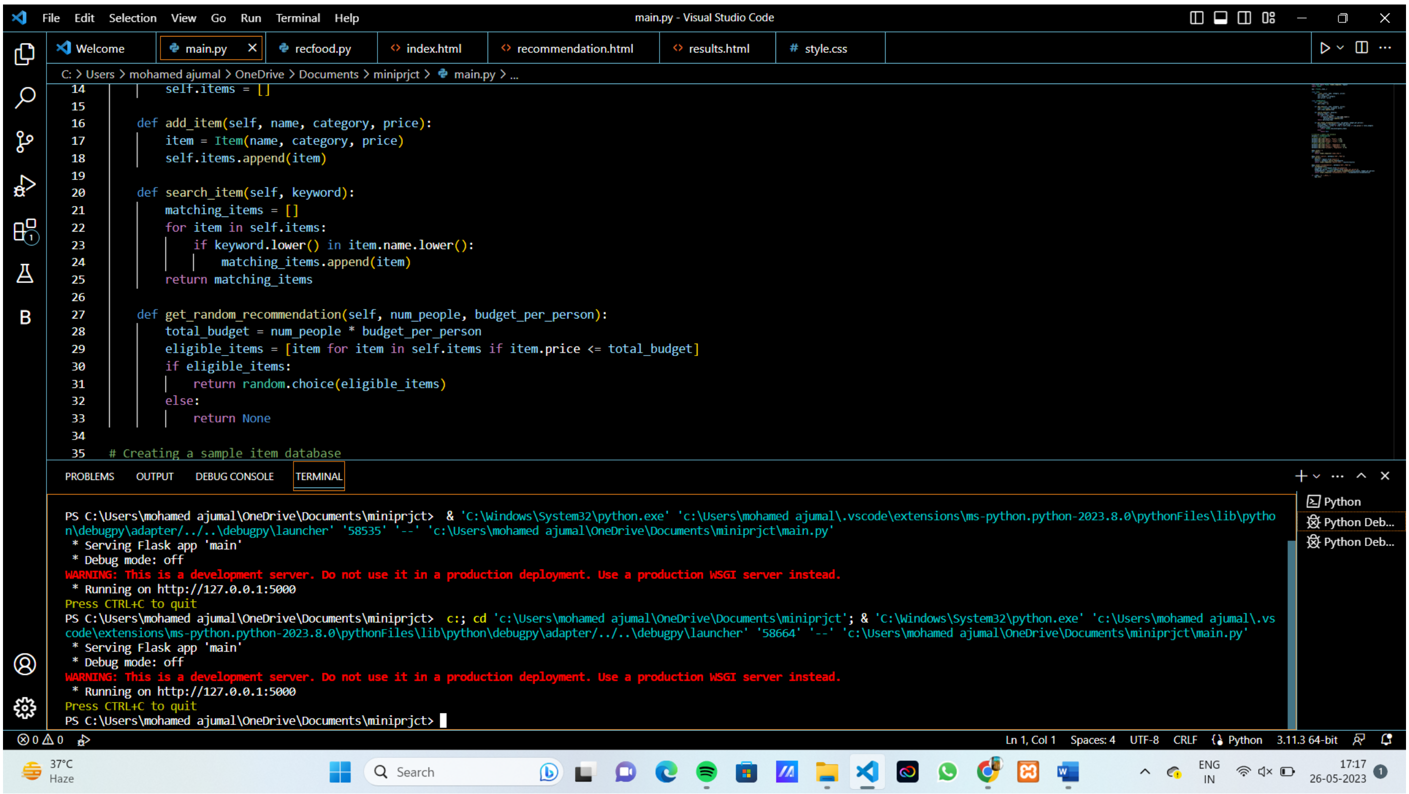
Home Products Cart About Contact

Search Results

- Name: Apple, Category: Fruit, Price: 1.99

Buy Now Go back Change Color to Blue

© 2023 AI Assistant Grocery Shopping. All rights reserved.



```python
        self.items = []

    def add_item(self, name, category, price):
        item = Item(name, category, price)
        self.items.append(item)

    def search_item(self, keyword):
        matching_items = []
        for item in self.items:
            if keyword.lower() in item.name.lower():
                matching_items.append(item)
        return matching_items

    def get_random_recommendation(self, num_people, budget_per_person):
        total_budget = num_people * budget_per_person
        eligible_items = [item for item in self.items if item.price <= total_budget]
        if eligible_items:
            return random.choice(eligible_items)
        else:
            return None

# Creating a sample item database
```

```
PS C:\Users\mohamed ajumal\OneDrive\Documents\miniprjct>  & 'C:\Windows\System32\python.exe' 'c:\Users\mohamed ajumal\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\pytho
n\debugpy\adapter/../..\debugpy\launcher' '58535' '--' 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct\main.py'
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
PS C:\Users\mohamed ajumal\OneDrive\Documents\miniprjct>  c:; cd 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct'; & 'C:\Windows\System32\python.exe' 'c:\Users\mohamed ajumal\.vs
code\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '58664' '--' 'c:\Users\mohamed ajumal\OneDrive\Documents\miniprjct\main.py'
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
PS C:\Users\mohamed ajumal\OneDrive\Documents\miniprjct>
```

APPENDIX 2

SOURCE CODE

Main.py

```
from flask import Flask, render_template, request
import random
app = Flask(__name__)
class Item:
    def __init__(self, name, category, price):
        self.name = name
        self.category = category
        self.price = price
class ItemDatabase:
    def __init__(self):
        self.items = []
def add_item(self, name, category, price):
        item = Item(name, category, price)
        self.items.append(item)
def search_item(self, keyword):
        matching_items = []
        for item in self.items:
            if keyword.lower() in item.name.lower():
                matching_items.append(item)
        return matching_items
def get_random_recommendation(self, num_people, budget_per_person):
        total_budget = num_people * budget_per_person
        eligible_items = [item for item in self.items if item.price <= total_budget]
        if eligible_items:
            return random.choice(eligible_items)
        else:
 return None
# Creating a sample item database
database = ItemDatabase()
# Adding fruit items
database.add_item("Apple", "Fruit", 1.99)
database.add_item("Banana", "Fruit", 0.99)
database.add_item("Orange", "Fruit", 1.49)
# Adding vegetable items
database.add_item("Carrot", "Vegetable", 0.99)
database.add_item("Tomato", "Vegetable", 1.49)
database.add_item("Cucumber", "Vegetable", 0.79)
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/search', methods=['GET','POST'])
def search():
    keyword = request.form['keyword']
    results = database.search_item(keyword)
    return render_template('results.html', results=results)
@app.route('/recommendation', methods=['GET','POST'])
def recommendation():
    num_people = int(request.form['num_people'])
    budget_per_person = float(request.form['budget_per_person'])
            recommendation = database.get_random_recommendation(num_people,
budget_per_person)
    return render_template('recommendation.html', recommendation=recommendation)
if __name__ == '__main__':
    app.run()
```

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>AI Assistant Grocery Shopping</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
 header {
      background-color: #333;
      color: #fff;
      padding: 20px;
    }
h1 {
      margin: 0;
    }
 nav ul {
      list-style: none;
      padding: 0;
      margin: 0;
    }
 nav ul li {
      display: inline-block;
      margin-right: 10px;
```

```
   }
 nav ul li a {
      color: #fff;
      text-decoration: none;
   }
main {
      padding: 20px;
   }
 section {
      margin-bottom: 20px;
   }
 h2 {
      color: #333;
   }
 form {
      margin-top: 20px;
   }
 label {
      display: block;
      margin-bottom: 5px;
   }
input[type="text"],
    button[type="submit"] {
      padding: 5px;
      font-size: 16px;
   }
 footer {
      background-color: #333;
      color: #fff;
      padding: 20px;
      text-align: center;
 }
  </style>
  <script>
                  window.SpeechRecognition = window.SpeechRecognition ||
window.webkitSpeechRecognition;
function startSpeechRecognition() {
      const recognition = new SpeechRecognition();
      recognition.interimResults = true;
recognition.addEventListener('result', (event) => {
        const transcript = Array.from(event.results)
          .map((result) => result[0].transcript)
          .join('');
document.getElementById('keyword').value = transcript;
      });
 recognition.start();
    }
  </script>
</head>
<body>
  <header>
    <h1>AI Assistant Grocery Shopping</h1>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
        <li><a href="#">Products</a></li>
        <li><a href="#">Cart</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
 </header>
  <main>
    <section>
      <h2>Welcome to AI Assistant Grocery Shopping</h2>
      <p>Get all your groceries with the help of our AI assistant! Simply enter your grocery list or use voice input, and let our AI assistant do the rest.</p>
      <form action="/search" method="POST">
        <label for="keyword">Enter your grocery list:</label>
          <input type="text" id="keyword" name="keyword" placeholder="Enter your grocery list">
        <button type="button" onclick="startSpeechRecognition()">Voice Input</button>
        <button type="submit">Search</button>
      </form>
    </section>
  </main>
  <footer>
    <p>© 2023 AI Assistant Grocery Shopping. All rights reserved.</p>
  </footer>
</body>
</html>
```

Recommendation.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>AI Assistant Grocery Shopping - Recommendation</title>
  <style>
    body {
font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
 header {
      background-color: #333;
      color: #fff;
      padding: 20px;
    }
h1 {
      margin: 0;
    }
 nav ul {
      list-style: none;
      padding: 0;
      margin: 0;
    }
 nav ul li {
      display: inline-block;
      margin-right: 10px;
    }
 nav ul li a {
      color: #fff;
      text-decoration: none;
    }
 main {
      padding: 20px;
    }
 section {
      margin-bottom: 20px;
}
h2 {
      color: #333;
    }
 ul {
      padding: 0;
      margin: 0;
    }
 li {
      margin-bottom: 5px;
    }
 a {
      color: #333;
      text-decoration: none;
    }
  footer {
      background-color: #333;
      color: #fff;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <header>
```

```
    <h1>AI Assistant Grocery Shopping</h1>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
        <li><a href="#">Products</a></li>
<li><a href="#">Cart</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section>
      <h2>Recommended Item</h2>
      {% if recommendation %}
        <ul>
          <li>Name: {{ recommendation.name }}</li>
          <li>Category: {{ recommendation.category }}</li>
          <li>Price: {{ recommendation.price }}</li>
        </ul>
      {% else %}
        <p>No recommendations available.</p>
      {% endif %}
      <a href="/">Go back</a>
    </section>
  </main>
  <footer>
    <p>© 2023 AI Assistant Grocery Shopping. All rights reserved.</p>
  </footer>
```

```
</body>
</html>
```

Script.js

```
 function() {
  var width, height, largeHeader, canvas, ctx, points, target, animateHeader = true;
 // Main
    initHeader();
    initAnimation();
    addListeners();
 function initHeader() {
        width = window.innerWidth;
        height = window.innerHeight;
        target = {x: width/2, y: height/2};
 largeHeader = document.getElementById('large-header');
        largeHeader.style.height = height+'px';
 canvas = document.getElementById('demo-canvas');
        canvas.width = width;
        canvas.height = height;
        ctx = canvas.getContext('2d');
 // create points
        points = [];
        for(var x = 0; x < width; x = x + width/20) {
            for(var y = 0; y < height; y = y + height/20) {
                var px = x + Math.random()*width/20;
                var py = y + Math.random()*height/20;
                var p = {x: px, originX: px, y: py, originY: py };
                points.push(p);
            }
        }
    // for each point find the 5 closest points
        for(var i = 0; i < points.length; i++) {
            var closest = [];
            var p1 = points[i];
            for(var j = 0; j < points.length; j++) {
    var p2 = points[j]
                if(!(p1 == p2)) {
                    var placed = false;
                    for(var k = 0; k < 5; k++) {
                        if(!placed) {
                            if(closest[k] == undefined) {
                                closest[k] = p2;
                                placed = true;
                            }
                        }
                    }
    for(var k = 0; k < 5; k++) {
                        if(!placed) {
                            if(getDistance(p1, p2) < getDistance(p1, closest[k])) {
                                closest[k] = p2;
                                placed = true;
                            }
                        }
                    }
                }
            }
            p1.closest = closest;
        }
  // assign a circle to each point
        for(var i in points) {
            var c = new Circle(points[i], 2+Math.random()*2, 'rgba(255,255,255,0.3)');
            points[i].circle = c;
        }
    }
// Event handling
    function addListeners() {
        if(!('ontouchstart' in window)) {
            window.addEventListener('mousemove', mouseMove);
        }
        window.addEventListener('scroll', scrollCheck);
        window.addEventListener('resize', resize);
    }
function mouseMove(e) {
        var posx = posy = 0;
        if (e.pageX || e.pageY) {
            posx = e.pageX;
            posy = e.pageY;
        }
        else if (e.clientX || e.clientY)    {
                                  posx = e.clientX + document.body.scrollLeft +
document.documentElement.scrollLeft;
                                  posy = e.clientY + document.body.scrollTop +
document.documentElement.scrollTop;
        }
        target.x = posx;
        target.y = posy;
    }
 function scrollCheck() {
        if(document.body.scrollTop > height) animateHeader = false;
        else animateHeader = true;
    }
 function resize() {
        width = window.innerWidth;
        height = window.innerHeight;
        largeHeader.style.height = height+'px';
        canvas.width = width;
        canvas.height = height;
    }
 // animation
    function initAnimation() {
        animate();
        for(var i in points) {
            shiftPoint(points[i]);
        }
    }
function animate() {
        if(animateHeader) {
            ctx.clearRect(0,0,width,height);
            for(var i in points) {
                // detect points in range
                if(Math.abs(getDistance(target, points[i])) < 4000) {
                    points[i].active = 0.3;
                    points[i].circle.active = 0.6;
                } else if(Math.abs(getDistance(target, points[i])) < 20000) {
                    points[i].active = 0.1;
                    points[i].circle.active = 0.3;
                } else if(Math.abs(getDistance(target, points[i])) < 40000) {
                    points[i].active = 0.02;
                    points[i].circle.active = 0.1;
                } else {
                    points[i].active = 0;
                    points[i].circle.active = 0;
                }
 drawLines(points[i]);
                points[i].circle.draw();
            }
        }
        requestAnimationFrame(animate);
    }
function shiftPoint(p) {
        TweenLite.to(p, 1+1*Math.random(), {x:p.originX-50+Math.random()*100,
            y: p.originY-50+Math.random()*100, ease:Circ.easeInOut,
            onComplete: function() {
                shiftPoint(p);
            }});
    }
// Canvas manipulation
    function drawLines(p) {
        if(!p.active) return;
        for(var i in p.closest) {
            ctx.beginPath();
            ctx.moveTo(p.x, p.y);
            ctx.lineTo(p.closest[i].x, p.closest[i].y);
            ctx.strokeStyle = 'rgba(156,217,249,'+ p.active+')';
            ctx.stroke();
        }
    }
 function Circle(pos,rad,color) {
        var _this = this;
 // constructor
        (function() {
            _this.pos = pos || null;
            _this.radius = rad || null;
            _this.color = color || null;
        })();
this.draw = function() {
            if(!_this.active) return;
            ctx.beginPath();
            ctx.arc(_this.pos.x, _this.pos.y, _this.radius, 0, 2 * Math.PI, false);
            ctx.fillStyle = 'rgba(156,217,249,'+ _this.active+')';
            ctx.fill();
        };
    }
 // Util
    function getDistance(p1, p2) {
        return Math.pow(p1.x - p2.x, 2) + Math.pow(p1.y - p2.y, 2);
    }
})();
```

CHAPTER 8

REFRENCES

[1] Kamble S, Meshram S, Thokal R, Gakre R 2014 Developing a Multitasking Shopping Trolley Based On RFID Technology: International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-6

[2] S. Sainath, K. Surender, V. , Arvind V 2014 Automated Shopping Trolley for Super Market Billing System: International Journal of Computer Applications (0975 – 8887) International Conference on Communication, Computing and Information Technology (ICCCMIT)

[3] Shaw R, Akhil R, Radhakrishnan A, Senthil R 2017 Smart Kart: Advances in Computational Sciences and Technology, © Research India Publications http://www.ripublication.com ISSN 0973- 6107 Volume 10, pp 2273-2288

[4] Ng YenLeng, Lim Cheng Siong, Danapalasingam K A, Tan M L P, Tan C W: Automatic Human Guided Shopping Trolley with Smart Shopping System

[5] Shopping Trolley Underpayment Cases to Proceed. 2013, September 1. Australian Payroll Association. Retrieved September 16, 2014.

[6] Dawkhar K, Dhomase S, Mahabaleshwarkar 2015 S Electronic Shopping Cart For Effective Shopping based on RFID: International Journal OF Innovative Research in Electrical, Electronics, INSTRUMENTATION and Control Engineering Vol. 3, Issue 1

[7] Aryan P, Pise P, Tamhane S Smart Shopping Cart with Automatic Billing System through RFID and Bluetooth International Journal of Emerging Technology and Computer Science.

[8] Shih C, Liang B and Lin C 2011 “An Automatic Smart Shopping Cart Deployment Framework based on Pattern Design”, IEEE 15th International Symposium on Consumer Electronics.

[9] Dhokte S.S., Patere B.S. ,Magar M.T, Kulkarni V.S., Patil P.S. ,Patil R 2015 “Smart Shopping Trolley Using Rechargeable Smart Card “ in International Journal of Emerging Technology and Advanced Engineering, Volume 5, Issue 5.

[10] Bedi H.S., Goyal N, Kumar S and Gupta A 2017 Smart Trolley using Smart Phone and Arduino: Journal of Electrical & Electronic Systems.