

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ТА
ЗАСОБІВ ГЕНЕРАТИВНОГО AI ДЛЯ ПОЛЕГШЕННЯ ПОБУТУ ЛЮДИНИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF GENERATIVE AI
METHODS AND TOOLS TO FACILITATE EVERYDAY HUMAN LIFE

Виконав: студент(ка) 2 курсу, групи КНТ-212м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

БЕЗКОРОВАЙНИЙ М. Р.

(прізвище та ініціали)

Керівник ШИТІКОВА О. В.

(прізвище та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій Субботін

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БЕЗКОРОВАЙНОГО Микити Романовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів та засобів генеративного AI для полегшення побуту людини

Research and Software Implementation of Generative AI Methods and Tools to Facilitate Everyday Human Life

керівник проєкту (роботи) к.т.н., доцент ШИТІКОВА Олена Вікторівна

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року
№ 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити 1. Аналіз предметної області 2. Характеристики об'єкта проектування

3. Вибір і обґрунтування структури та методів розпізнавання проектованої системи

4. Керівництво програміста 5 Керівництво користувача

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ШИТІКОВА О.В к.т.н., доцент кафедри ПЗ		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 14 ” листопада 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Вибір засобів розробки програмного продукту	4-5 тижні	Розділ 2
	Розробка архітектури програми	6-7 тижні	Розділ 3
4	Розробка програмного продукту	8-9 тижні	Розділ 4-5
5	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	10-11 тижні	Додатки
6	Захист роботи	12 тиждень	

Студент(ка)

Керівник проєкту (роботи)

Микита БЕЗКОРОВАЙНИЙ

(підпис)

(прізвище та ініціали)

Олена ШИТІКОВА

(підпис)

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
95 с., 2 табл., 38 рис., 2 дод., 27 джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗРОБКА ВЕБ ДОДАТКУ, OPENAI

Об'єктом дослідження є процес застосування методів та засобів генеративного штучного інтелекту (генеративний AI).

Предмет дослідження – програмні системи на базі генеративного штучного інтелекту (генеративний AI). Це включає вивчення та аналіз різноманітних програмних продуктів, які використовують методи та алгоритми генеративного AI для створення нових контентів або розв'язання завдань у сферах, таких як обробка природної мови, обробка зображень, аудіогенерація, та інші.

Метою дослідження є розробка програмного забезпечення, яке допоможе простим користувачам зручно будувати рецепти із продуктів які є під рукою.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування Angular, середовище розробки IntelliJ IDEA, API OpenAI, комп'ютер з шестидерним процесором Intel Core i7, оперативною пам'яттю на 16 Гб, монітором, клавіатура.

Виконано аналіз предметної області, розглянуто аналоги на ринку, що зараз існують, на основі цього були виявлені переваги та недоліки кожного рішення та було розроблено модель, яка розв'язує найголовніші проблеми

аналогів, що було розглянуто. Після чого за допомогою розробленої моделі було реалізовано програмне рішення.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 95 pages, 2 tables, 38 figures, 2 appendixes, 27 sources.

ARTIFICIAL INTELLIGENCE, SOFTWARE, WEB APPLICATION DEVELOPMENT, OPENAI.

The object of research is the process of applying methods and means of generative artificial intelligence (generative AI).

The subject of research is software systems based on generative artificial intelligence (generative AI). This includes the study and analysis of various software products that use generative AI methods and algorithms to create new content or solve tasks in areas such as natural language processing, image processing, audio generation, and others.

Tasks – development of software that will help ordinary users conveniently build recipes from products that are at hand.

Materials, methods and technical means: structural and object-oriented programming, Angular programming language, IntelliJ IDEA development environment, OpenAI API, a computer with a six-core Intel Core i7 processor, 16 GB of RAM, a monitor, a keyboard.

An analysis of the subject area was performed, analogues on the market that currently exist were considered, on the basis of this, the advantages and disadvantages of each solution were identified, and a model was developed that solves the main problems of the analogues that were considered. After that, a software solution was implemented using the developed model.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	9
Вступ	10
1 Аналіз предметної області	14
1.1 Аналіз методів та засобів генеративного AI	14
1.2 Аналіз сучасних систем на базі генеративного AI в гастрономії	16
1.3 Приклади систем	17
1.3.1 Система "ГастроКреатив"	18
1.3.2 ChefBot 3000.....	18
1.3.3 TasteCraft	18
1.4 Постановка завдань роботи.....	19
1.5 Висновок по першому розділі.....	20
2 Матеріали та методи	22
2.1 Платформа chatGPT: Можливості та обмеження	22
2.2.1 Представники генеративного AI	22
2.2.2 ChatGPT	28
2.2 Фреймворк Angular: Можливості	34
2.2.1 Angular.....	34
2.2.3 RxJS	38
2.2.4 Jasmine	40
2.3 Технічні аспекти інтеграції Angular з генеративним AI.....	41
2.4 Висновки за розділом 2	42
3 Розробка та реалізація інтелектуального асистента для генерації рецептів	44
3.1 Розробка алгоритму генерації рецептів.....	44
3.2 Програмна реалізація та інтеграція з chatGPT	47
3.3 Створення Angular додатку для введення продуктів.....	49
3.4 Висновки за розділом 3	58

4 Експлуатація, тестування та експериментальне дослідження	60
4.1 Призначення й умови застосування додатку	60
4.2 Характеристики додатку	61
4.3 Інструкція по експлуатації додатку	61
4.4 Тестування програмного додатку	62
4.5 Висновки за розділом 4	64
Висновки	66
Перелік джерел посилання	67
Додаток А Текст програми	70
Додаток Б Слайди презентації	90

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	– Artificial intelligence;
API	– Application programming interface;
BERT	– Bidirectional Encoder Representations from Transformers;
CNN	– convolutional neural network;
CSS	– Cascading Style Sheets;
CTRL	– conditional transformer language model;
DAN	– Do anything now;
DOM	– Document Object Model;
ES6	– ECMAScript 6;
GAN	– Generative Adversarial Networks;
GB	– gigabyte;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
JS	– JavaScript;
NLP	– Natural Language Processing;
RAM	– Random Access Memory;
RNN	– Recurrent neural network;
SVG	– Scalable Vector Graphics;
TS	– TypeScript;
UI	– User interface;
URL	– Uniform Resource Locator;
VAE	– Variational Autoencoders;
ООП	– Об'єктно-орієнтоване програмування;
ШІ	– Штучний інтелект.

ВСТУП

В сучасному інформаційному суспільстві, де технології активно змінюють наше повсякденне життя, поєднання штучного інтелекту та інтерактивних інтерфейсів стає ключовим напрямком для розвитку інноваційних рішень. Однією з найцікавіших областей використання цих технологій є кулінарія та готування їжі.

У світлі стрімкого технологічного розвитку та зростаючого впливу штучного інтелекту в різних сферах життя, важливо акцентувати увагу на застосуванні інноваційних підходів для покращення щоденного життя людей. Зокрема, в галузі готування їжі виникає проблема ефективного використання доступних продуктів. Використання методів генеративного AI для створення персоналізованих рецептів може суттєво полегшити цей процес та забезпечити новий рівень зручності та креативності в готуванні.

Поєднання потужності генеративного ШІ та можливостей платформи chatGPT відкриває широкі перспективи для створення інтелектуального асистента у кулінарній сфері. Здатність взаємодіяти з системою, вводячи лише наявні продукти в холодильник, і отримувати індивідуалізовані рецепти забезпечить не тільки високий рівень сервісу, але й сприятиме раціональному використанню продуктів та уникненню їхнього марнування.

Враховуючи зростаючий інтерес до галузі генеративного ШІ та його практичного використання, дослідження, присвячене створенню інноваційного інструменту для кулінарної творчості за допомогою генеративного ШІ та chatGPT, стає надзвичайно актуальним та перспективним напрямком наукових досліджень.

Об'єкт дослідження – процес застосування методів та засобів генеративного штучного інтелекту (генеративний AI). Генеративний AI є складною областю, яка базується на використанні глибоких нейронних мереж

для створення нових, не існуючих раніше об'єктів, таких як текст, зображення чи звук.

Дослідження включає в себе аналіз ефективності різних алгоритмів генеративного AI, їх застосування в конкретних галузях, а також вивчення можливостей оптимізації та покращення результатів. Особлива увага приділяється розумінню процесів навчання генеративних моделей, їхнім взаємодіям із зовнішнім середовищем та адаптації до різноманітних завдань.

Дослідження також спрямоване на вивчення можливостей вдосконалення якості генерованих об'єктів, уникнення недоліків та покращення швидкості роботи генеративних систем. Одним із ключових аспектів є визначення етичних аспектів використання генеративного AI, таких як уникнення виникнення алгоритмічного упередження та забезпечення безпеки при взаємодії з користувачами.

Узагальнюючи, об'єкт дослідження охоплює весь життєвий цикл генеративного AI – від навчання моделей до їхнього застосування в конкретних сценаріях та подальшого вдосконалення.

Предмет дослідження – програмні системи на базі генеративного штучного інтелекту (генеративний AI). Це включає вивчення та аналіз різноманітних програмних продуктів, які використовують методи та алгоритми генеративного AI для створення нових контентів або розв'язання завдань у сферах, таких як обробка природної мови, обробка зображень, аудіогенерація, та інші.

Дослідження зосереджене на вивченні архітектур програмних систем, які використовують генеративний AI, та їхніх технічних характеристик. Важливо враховувати якість та результативність програм, їхню можливість адаптації до різних завдань, а також ступінь їхньої взаємодії з користувачем та іншими програмними системами.

Дослідження також включає аналіз та порівняння різних підходів до реалізації генеративного AI у програмних системах, оцінюючи їхню ефективність, стабільність та можливості розширення. Звертається увага на етапи життєвого циклу програм, включаючи навчання моделей, впровадження та підтримку.

Узагальнюючи, предмет дослідження охоплює весь спектр програмних продуктів, що базуються на генеративному AI, і дозволяє глибше зрозуміти їхню роль, можливості та перспективи у сучасному програмному середовищі.

У сучасному інформаційному суспільстві, де технології розвиваються на шалену швидкість, важливо визначити нові шляхи застосування штучного інтелекту для вирішення повсякденних проблем. Однією із суттєвих задач є оптимізація процесу готування їжі, зокрема, врахування наявних продуктів у холодильнику.

Мета даного дослідження полягає в створенні та впровадженні ефективного інструменту на основі генеративного штучного інтелекту та платформи chatGPT для автоматизації процесу генерації рецептів. Головною метою є надання користувачам простого та інтуїтивного інтерфейсу для введення наявних продуктів та отримання персоналізованих рецептів на їх основі.

Для досягнення поставленої мети були визначені наступні конкретні завдання:

а) аналіз методів генеративного ШІ: Провести комплексний огляд існуючих методів генеративного штучного інтелекту з акцентом на їх застосування в галузі генерації рецептів;

б) розробка алгоритму генерації рецептів: Розробити ефективний алгоритм, який здатний адаптуватися до введених продуктів та генерувати смачні та інноваційні рецепти;

в) програмна реалізація: Реалізувати розроблений алгоритм у вигляді програмного додатку, що інтегрується з платформою chatGPT для забезпечення продуктивної та динамічної взаємодії з користувачем;

г) оцінка ефективності: Провести об'єктивну оцінку розробленого інструменту на основі критеріїв ефективності, зручності використання та задоволення користувача.

Це дослідження спрямоване на створення інноваційного рішення, яке не лише забезпечить конкретний функціонал, а й відкриє нові перспективи використання генеративного штучного інтелекту в побутових задачах.

Використовуючи методи генеративного штучного інтелекту та платформу chatGPT, спрямовані на створення інноваційного інструменту, який полегшить процес приготування їжі та забезпечить користувача унікальними та смачними рецептами.

Дослідження охоплює аналіз сучасних методів генеративного штучного інтелекту, вивчення можливостей платформи chatGPT та розробку алгоритму генерації рецептів. Особлива увага приділяється інтеграції інтелектуального асистента з Angular, з метою створення зручного та інтуїтивно зрозумілого інтерфейсу для користувача.

Це дослідження присвячене не лише технічним аспектам створення програмного продукту, а й вивченню практичних застосувань генеративного штучного інтелекту у побутовому середовищі. Його результати можуть сприяти не лише удосконаленню процесу готування, але й розвитку нових підходів до використання інтелектуальних технологій в щоденному житті

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз методів та засобів генеративного AI

Генеративний штучний інтелект (генеративне AI) є важливою галуззю дослідження в сфері комп'ютерних наук та інформаційних технологій. Цей розділ присвячений аналізу різних методів та інструментів, які використовуються у галузі генеративного штучного інтелекту. Для досягнення цієї мети проводиться огляд основних напрямків та технологій, що застосовуються для генерації контенту та вирішення завдань, пов'язаних з генеративним AI [1].

Одним із ключових напрямків в генеративному AI є застосування методів обробки природної мови. Алгоритми NLP (рисунок 1.1) використовуються для розуміння та генерації текстової інформації, що робить їх ефективними для створення текстового контенту, такого як рецепти, описи тощо.

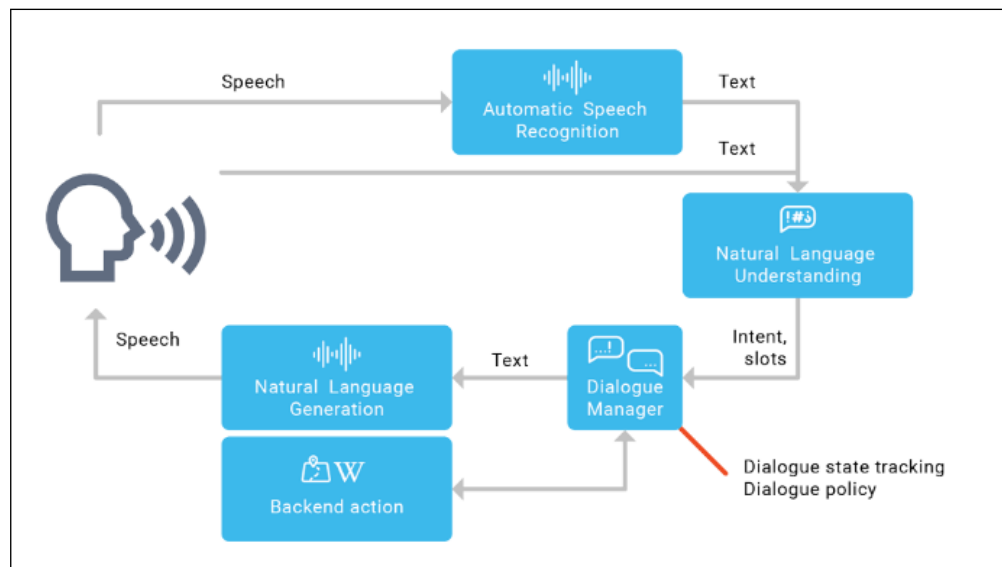


Рисунок 1.1 – Алгоритм NLP [2]

Використання глибокого навчання та нейронних мереж є ключовим аспектом в генеративному AI. Рекурентні нейронні мережі (RNN), варіації генеративних засобів, таких як Variational Autoencoders (VAE) та Generative Adversarial Networks (GAN), забезпечують можливість створення нових контентів на основі введених даних (рисунок 1.2).

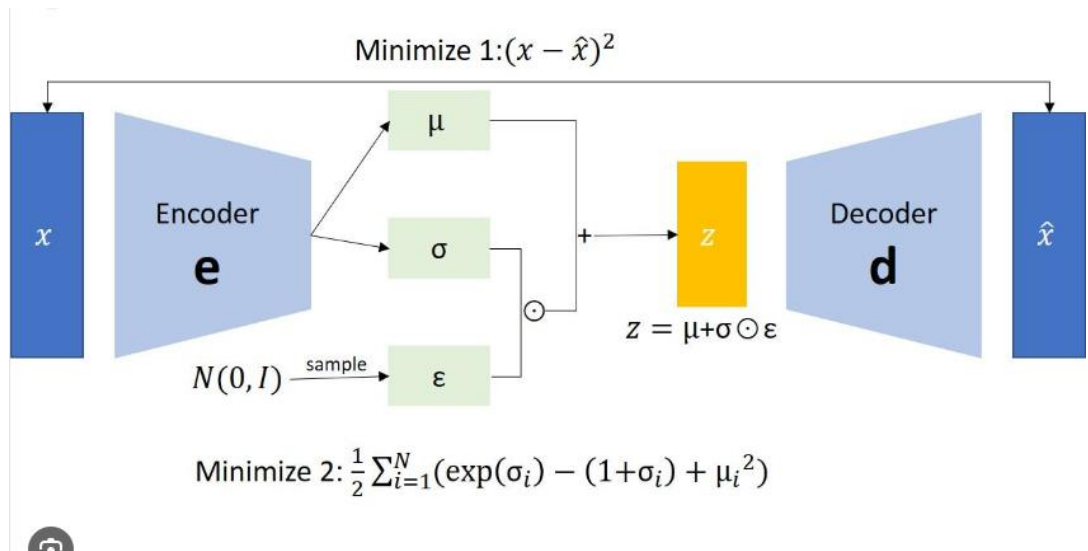


Рисунок 1.2 – Variational Autoencoders [2]

Для генерації візуального контенту, такого як фотографії страв чи кулінарні композиції, використовуються методи обробки зображень та комп'ютерного бачення. Застосування згорткових нейронних мереж (CNN) та технік передобробки зображень дозволяє створювати вражаючі візуальні елементи.

У випадках, коли генерація контенту пов'язана з аудіо або музичними композиціями, застосовуються методи обробки сигналів та алгоритми, спрямовані на створення аудіальних вражень.

Деякі системи генеративного AI можуть інтегруватися з зовнішніми сервісами, такими як чат-боти, що дозволяє взаємодіяти з користувачами та отримувати замовлення або додаткові вказівки.

Цей аналіз методів та засобів генеративного AI надає змогу зрозуміти ключові технічні аспекти, які можуть бути використані у створенні інтелектуального асистента для генерації рецептів на основі наявних продуктів.

В сучасній історії досліджується можливість створення інтелектуального асистента для готування, який враховує не лише персональні смакові вподобання користувача, але й наявність конкретних інгредієнтів. Це відкриває шлях до нових можливостей у сфері кулінарії та забезпечує унікальний підхід до вирішення повсякденних задач.

Застосування генеративного ШІ у створенні рецептів на основі наявних продуктів має велике значення для розвитку сучасної кулінарії. Воно не лише спрощує процес готування, але й стимулює творчість, пропонуючи унікальні та інноваційні способи комбінування продуктів.

Такий інтегрований підхід також відкриває можливості для майбутнього вдосконалення та розвитку сучасних систем генеративного ШІ в побутовому використанні. Результати даного дослідження можуть служити підґрунтям для подальших досліджень у сфері штучного інтелекту та його застосування в різних аспектах повсякденного життя.

1.2 Аналіз сучасних систем на базі генеративного AI в гастрономії

Гастрономічна галузь востаннє отримала великий імпульс з розвитком генеративного штучного інтелекту. Системи на базі генеративного AI дозволяють створювати інноваційні та персоналізовані рішення в галузі гастрономії. У цьому розділі ми розглянемо аналіз сучасних систем, які використовують генеративний AI в гастрономії.

Багато систем фокусуються на генерації унікальних та смачних рецептів. Вони використовують алгоритми обробки природної мови та

глибокого навчання для аналізу існуючих рецептів, ідентифікації трендів та створення нових кулінарних комбінацій.

Деякі системи використовують генеративний AI для створення персоналізованих рекомендацій щодо харчових вподобань користувачів. Вони аналізують історію замовлень, враховують дієтичні обмеження та пропонують індивідуально адаптовані рецепти.

Системи генеративного AI можуть діяти як віртуальні кулінарні асистенти, які надають поради щодо приготування страв, вказівки по збереженню продуктів та інші корисні рекомендації.

Системи генеративного AI в гастрономії розвивають новаторські кулінарні технології майбутнього. Це може включати в себе автоматизоване створення рецептів, аналіз харчових трендів та навіть створення смакових профілів для персонального підходу.

Деякі системи поєднують генеративний AI з віртуальною реальністю для іммерсивного гастрономічного досвіду. Вони можуть створювати віртуальні кулінарні лабораторії чи відтворювати страви віртуально.

Аналіз сучасних систем на базі генеративного AI в гастрономії свідчить про те, що ця технологія не лише трансформує спосіб приготування їжі, а й впливає на весь гастрономічний простір, надаючи нові можливості та розширюючи кулінарні горизонти.

1.3 Приклади систем

У цьому розділі проводиться аналіз реально існуючих систем, що базуються на генеративному штучному інтелекті в гастрономічній сфері. Переважна увага приділяється різним критеріям, таким як розробник, використана технологія, мета, наявний функціонал та інші значущі параметри.

1.3.1 Система "ГастроКреатив"

Розробник: Компанія "FoodInnovateTech"

Використана технологія: Глибокі нейронні мережі, основані на генеративних моделях.

Мета: Створення унікальних рецептів та кулінарних ідей, враховуючи індивідуальні смакові уподобання користувача.

Функціонал: Генерація рецептів, створення нових поєднань інгредієнтів, поради щодо стилю подачі страв.

1.3.2 ChefBot 3000

Розробник: Інститут Штучного Інтелекту "CulinariaTech"

Використана технологія: Основано на глибоких засобах машинного навчання та еволюційних алгоритмах.

Мета: Автоматизація процесу створення ресторанних меню, враховуючи витрати і популярні тенденції.

Функціонал: Розробка і оптимізація меню, прогнозування популярних страв, оптимізація складу страв для покращення маржі.

1.3.3 TasteCraft

Розробник: Спільнота хобі-кухарів та програмістів

Використана технологія: Відкриті алгоритми машинного навчання, базовані на відгуках користувачів.

Мета: Створення платформи для обміну та рецензування гастрономічних ідей та рецептів.

Функціонал: Взаємодія та обговорення ідей, колективна генерація рецептів, оцінювання та рецензування страв.

На таблиці 1.1 відображається порівняння систем, що базуються на генеративному штучному інтелекті в гастрономії.

Таблиця 1.1 – Порівняння систем на базі ШІ в гастрономії

Критерій	ГастроКреатив	ChefBot 3000	TasteCraft
Розробник	FoodInnovateTech	CulinariaTech	Спільнота
Використана технологія	Глибокі НМ, ГА	Глибокі ЗМН, ЕА	Відкриті алгоритми
Мета	Унікальні рецепти	Автоматизація меню	Обмін ідеями
Функціонал	Генерація рецептів	Оптимізація меню	Колективна генерація

1.4 Постановка завдань роботи

Метою даної дипломної роботи є дослідження, аналіз та розробка програмних рішень з використанням генеративного штучного інтелекту для оптимізації повсякденного життя людини. Робота спрямована на вивчення сучасних методів генеративного мистецтва, а також їхню реалізацію в програмному середовищі.

Спираючись на проведений в попередньому розділі аналіз існуючих систем, що базуються на генеративному штучному інтелекті в гастрономії, були сформовані функціональні вимоги до розроблюваної системи.

Система повинна забезпечувати:

- а) створення продукту (назва і вага);
- б) редагування продукту;
- в) видалення продукту;
- г) перегляд своїх продуктів;
- д) можливість вибору певних продуктів для створення рецепта;
- е) можливість генерації рецептів на основі обраних продуктів;
- ж) перегляд готового рецепта який згенерував AI.

1.5 Висновок по першому розділі

У першому розділі роботи було висвітлено важливі аспекти, пов'язані із впровадженням генеративного штучного інтелекту (генеративного ШІ) сферу готування та створення рецептів. Актуальність дослідження обумовлена потребою в ефективних та креативних рішеннях для покращення процесів щоденного готування.

Мета дослідження визначалася розробкою та реалізацією інноваційного інструменту, який базується на генеративному ШІ. Основні завдання, поставлені перед роботою, включають аналіз існуючих методів генеративного ШІ, розробку ефективного алгоритму генерації рецептів, програмну реалізацію та оцінку ефективності створеного інструменту.

Огляд літератури засвідчив актуальність обраного напрямку дослідження та визначив ключові технологічні аспекти, які слід врахувати при розробці інтелектуального асистента для готування. Детальний аналіз генеративного ШІ надав зрозуміння можливостей та переваг кожної технології.

Напрямки подальшого дослідження включають розширення функціоналу інтелектуального асистента, врахування додаткових параметрів користувача та оптимізацію алгоритмів для більш точної генерації рецептів.

Загальний висновок з першого розділу вказує на обґрунтованість обраного шляху дослідження та визначає контекст для подальших етапів розробки та валідації інтелектуального асистента для готування, що базується на генеративному ШІ.

2 МАТЕРІАЛИ ТА МЕТОДИ

2.1 Платформа chatGPT: Можливості та обмеження

2.2.1 Представники генеративного AI

Проведемо порівняльний аналіз кількох провідних платформ генеративного штучного інтелекту, розглянемо їхні унікальні особливості та можливості.

Алгоритм BERT (Bidirectional Encoder Representations from Transformers) - це метод глибокого навчання, що ґрунтується на технології обробки природної мови (NLP) і використовує нейронну мережу [2]. Вона була розроблена командою вчених з Google Research і вперше представлена в 2018 році. Основна ідея BERT полягає в тому, щоб навчати модель для передбачення слова в контексті, використовуючи обидва напрямки (зліва направо і справа наліво) тексту (рисунок 2.1). Це відрізняє BERT від багатьох попередніх моделей, які використовували лише однонаправковий підхід. Більше того, BERT використовує трансформатори, які є потужними блоками для обробки послідовностей даних. Однією з ключових переваг BERT є здатність розуміти контекст і семантику слів в розмовному контексті. Це дозволяє йому вирішувати завдання, такі як визначення сенсу слів, відповіді на питання, аналіз тональності та інші завдання NLP з вражаючою точністю. BERT широко використовується у великому спектрі застосувань, таких як пошукові системи, чат-боти, аналіз текстів, машинний переклад, виправлення помилок та інші завдання, пов'язані з обробкою природної мови. Модель також виявилася ефективною у попередженні перекладачних атак та розгортанні відповідних завдань NLP у багатьох галузях.

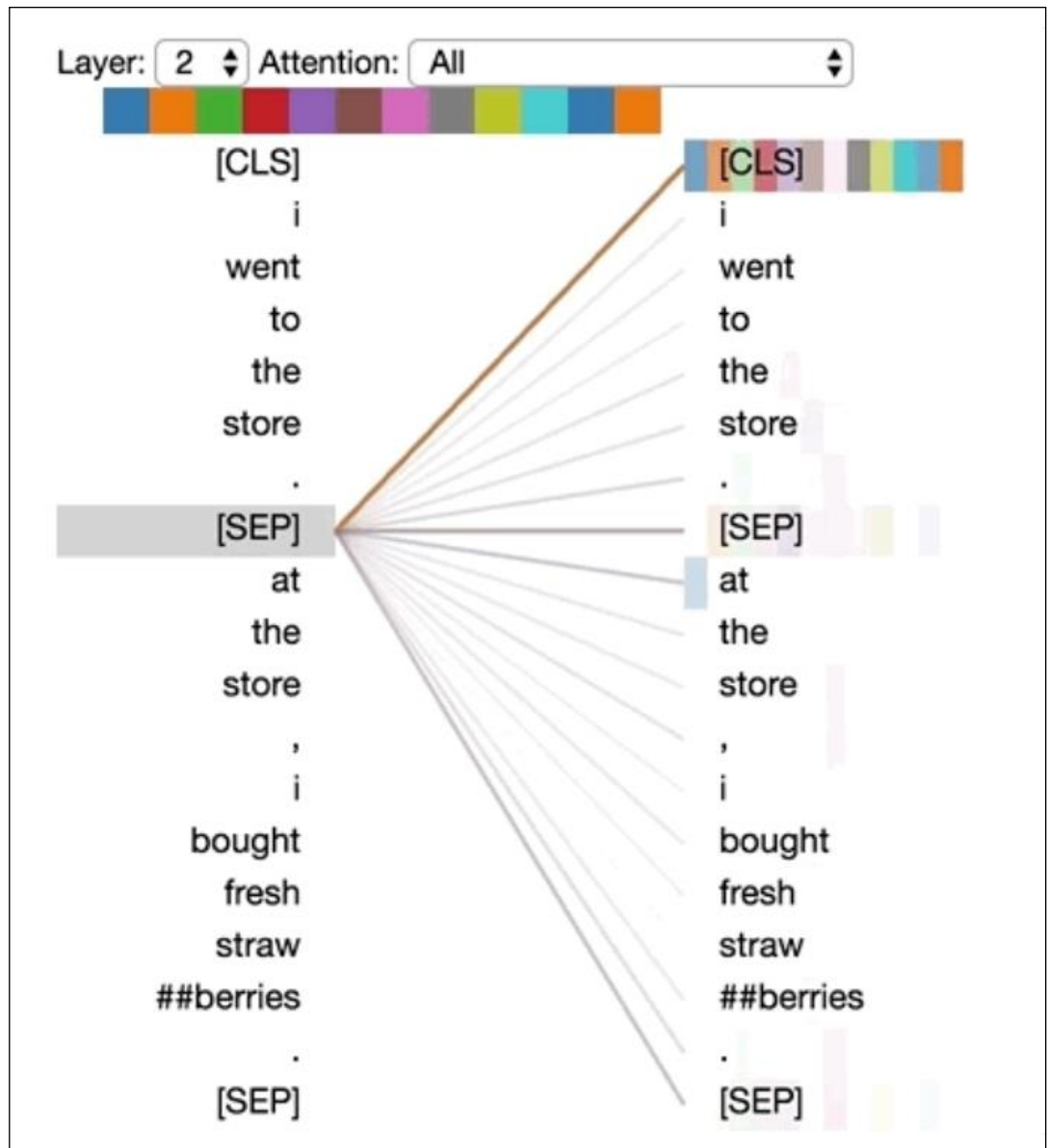


Рисунок 2.1 – Робота алгоритма передбачення слів [10]

Dialogflow - це інструмент розроблений Google, який пропонує зручність у створенні розмовних інтерфейсів та чат-ботів. Заснований на технології обробки природної мови (NLP), він володіє здатністю розпізнавання інтенцій та сутностей у тексті, що дозволяє розуміти користувацькі наміри та витягувати ключову інформацію. Розробники можуть легко визначати діалогові потоки та налаштовувати відповіді на різні запитання, спрощуючи взаємодію з користувачем. Dialogflow також легко

інтегрується з різними платформами, підтримує мультимовність та надає засоби для аналізу взаємодії користувачів, що робить його популярним інструментом для розробки розмовних інтерфейсів та чат-ботів. За допомогою Dialogflow додаток або бот надає методи введення та виведення і реагує на дані, які спонукають до дій. Також можна використовувати webhook для підключення Dialogflow до свого веб-сервісу (рисунок 2.2), який, в свою чергу, буде звертатися до сховищ даних, реалізовувати бізнес-логіку та виконувати виклики зовнішніх API. Приймаючи запит у вигляді тексту на природній мові або певної події, Dialogflow узгоджує запит з найбільш відповідним шаблоном. При цьому він базується на інформації, що міститься в шаблоні (приклади, сутності, контекст, параметри) та машинному навчанні. Dialogflow формує відповідний запит і повертає дані у вигляді об'єкта відповіді JSON.

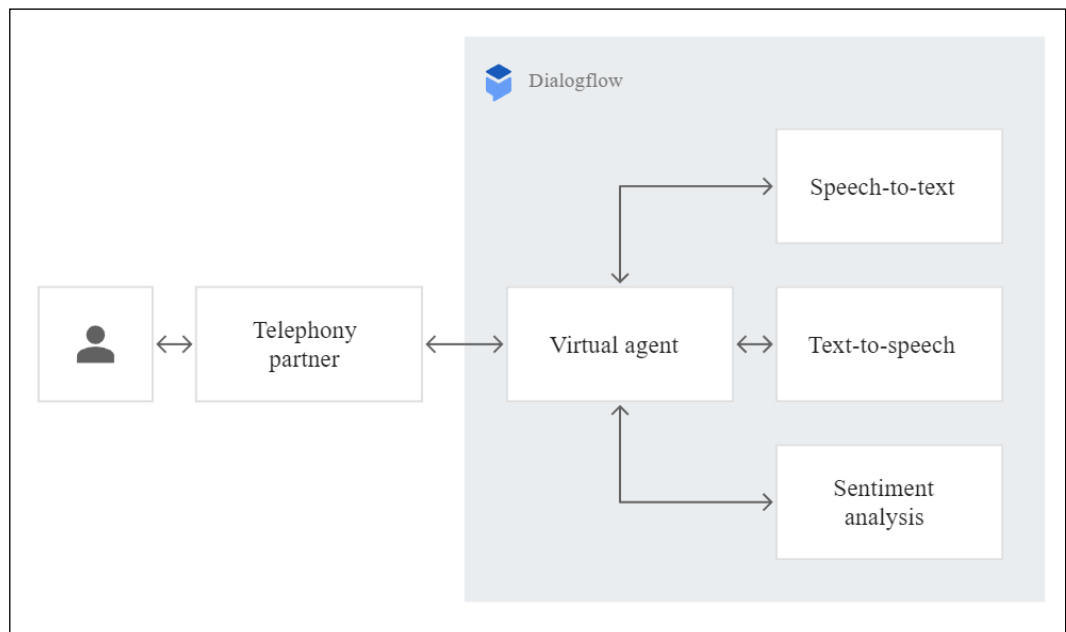


Рисунок 2.2 – Приклад використання Dialogflow у саппорт-боті

CTRL - це мовна модель з 1,6 мільярда параметрів, яка має потужну та управляючу систему штучного генерування тексту, що може передбачити,

який піднабір навчальних даних найбільше вплинув на згенеровану послідовність тексту. Вона надає потенційний метод для аналізу великої кількості згенерованого тексту, визначаючи найбільш впливовий джерело навчальних даних у моделі. Навчена з використанням понад 50 різних контрольних кодів, модель CTRL дозволяє поліпшити взаємодію людини з штучним інтелектом, оскільки користувачі можуть контролювати згенерований зміст і стиль тексту, а також тренувати її для генерації текстів багатозадачно.

Її можна використовувати для поліпшення інших застосувань обробки природної мови (NLP), якщо використовувати її для налаштування для конкретного завдання або для передачі репрезентацій, які модель вивчила [3]. За допомогою спеціальних ключових слів, які називаються контрольними кодами, люди можуть більш явно впливати на стиль, жанр, сутності, взаємини між сутностями та датами (рисунок 2.3). Це може покращити генерацію наративів, таку як генерація текстів з орієнтацією на ціль у застосунках, таких як відповіді на питання, машинний переклад або звичайний діалог у системах взаємодії людини з текстом, а також автоматичну генерацію художніх творів чи новинних звітів. Ці контрольні коди також роблять відносину між намірами людини та цими великими, складними моделями більш явною. Модель менше схильна до генерації випадкових послідовностей слів, ніж раніше випущені технології.

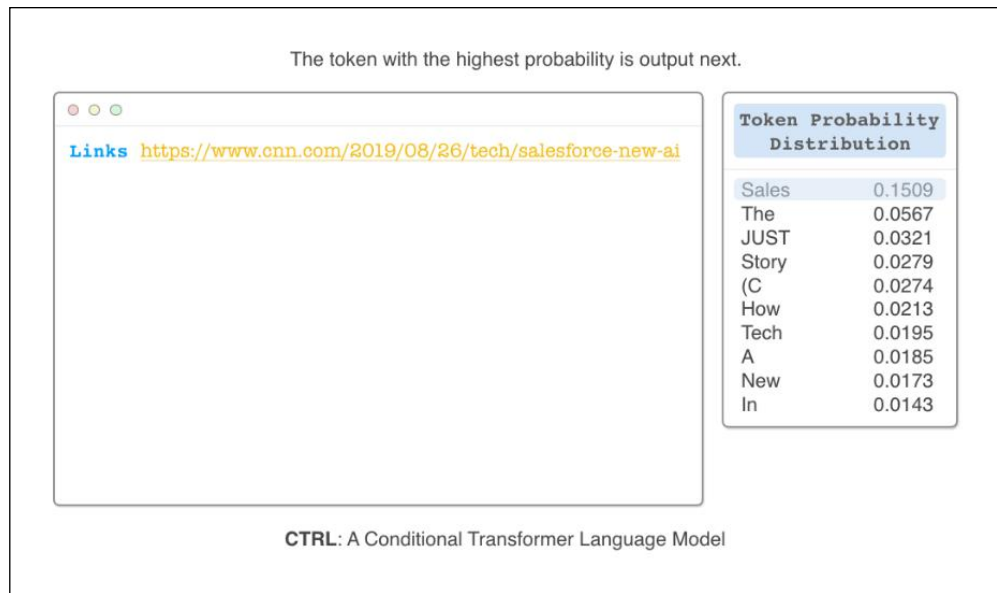


Рисунок 2.3 – Приклад аналізу даних

Приклад аналізу запитів представлено у таблиці 2.1

Таблиця 2.1 - Приклад аналізу запитів

Глобальне потепління - це брехня	r/unpopularopinion, r/conspiracy, r/science
Глобальне потепління - це брехня	r/eli5, r/science, r/unpopularopinion
Глобальне потепління - реальне явище	r/eli5, r/science, r/changemyview
Глобальне потепління - реальне явище	OpenWebText, r/changemyview, r/science

Порівняння вищезгаданих моделей у таблиці 2.2

Таблиця 2.2 – Порівняння моделей генеративного AI

Критерії	BERT	Dialogflow	CTRL
Призначення	Розуміння природної мови, NLU	Розробка чат-ботів та віртуальних асистентів	Генерація текстів, контрастивне навчання
Архітектура	Трансформатор	Різні технології під капотом для обробки природної мови	Трансформатор
Використання в галузях	Пошукові системи, відповіді на питання	Чат-боти, віртуальні асистенти	Генерація текстів, варіанти контекстів
Навчання	Масковане мовлення, великі корпуси текстів	Велика кількість текстових та голосових даних	Контрастивне навчання, генерація текстів
Видатки	Великі витрати обчислювальних ресурсів	Залежить від обсягу використання та обчислювальних потреб	Великі витрати обчислювальних ресурсів

2.2.2 ChatGPT

18 лютого 2022 року найгучніша технологічна новинка світу - чат-бот на основі штучного інтелекту ChatGPT став доступний для реєстрації в Україні. Минуло кілька місяців, відколи ChatGPT (Generative Pre-trained Transformer, «генеративний попередньо навчений трансформатор») вийшов на широкий загал, а хвиля інфоприводів навколо нього не вщухає [4].

ChatGPT – це чат-бот на основі мовної моделі штучного інтелекту (ШІ), розроблений американською лабораторією досліджень OpenAI. Був запущений як прототип у листопаді 2022 року і викликав неабиякий резонанс серед світової спільноти. Популярність до нього прийшла практично одразу: 1 млн користувачів за п'ять днів.

Звичайні пошукові системи видають зазвичай найбільш релевантний пошуковий запит. ChatGPT натомість прямо відповідає на запитання, тому не потрібно відвідувати безліч сайтів і фільтрувати гігабайти інформації. А ще ChatGPT піддається персоналізації. Тобто можна задати певний тон і вказати теми, які він обговорюватиме. Окрім написання текстів, бот може видавати базовий програмний код, генерувати фінансовий аналіз, статті, прогнози, поради.

Мовна модель (рисунок 2.4), завдяки якій функціонує чат-бот, навчалася за величезною кількістю даних. Серед них – різноманітні тексти, матеріали Вікіпедії, публікації користувачів у Twitter та пости в Reddit. Загалом робота з моделлю ШІ проходила у два етапи. Перший – навчання за допомогою діалогів, коли людина-тренер одночасно виступала у ролі потенційного користувача й асистента штучного інтелекту під час складання відповідей. А другий етап – навчання з підкріпленням. Він полягав в оцінюванні людиною різних варіантів відповіді, згенерованих мовною

моделлю, з метою розмістити їх за рейтингом і вибудувати таким чином систему пріоритетів.

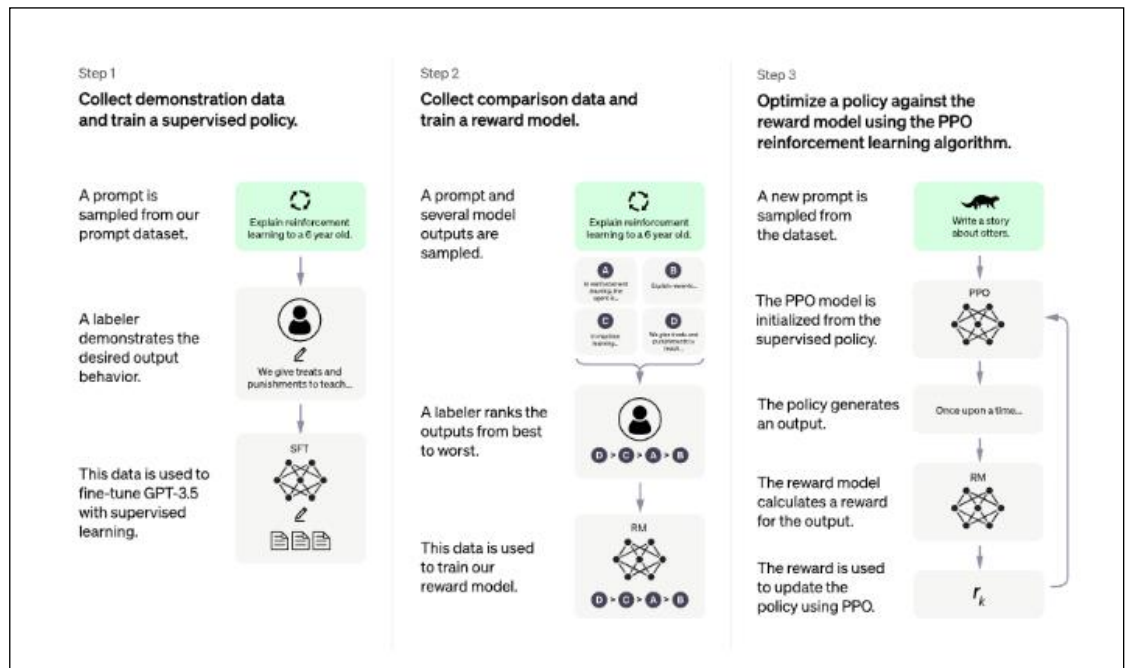


Рисунок 2.4 – Процес навчання chatGPT

ChatGPT здатний генерувати великі за обсягом тексти, наближені до стилю письма людини. У відповідь на правильно сформульований запит чат-бот може надати вам доволі повну та якісну інформацію з певної теми, написати нескладний програмний код, згенерувати перелік ідей, створити статтю, подати дані у вигляді зручної таблиці. Способи практичного використання технології надають чимало простору для фантазії. Так, американському продуктовому дизайнеру Аммару Реші спало на думку написати за допомогою ChatGPT дитячу книжку. Результат взаємодії людського та штучного інтелекту отримав назву «Аліса і Спаркл» і зараз продається на просторах інтернету. Якщо ж говорити про ключові характеристики ChatGPT, то він здатний підтримувати діалог, визнавати свої помилки, а також запам'ятовувати і

враховувати інформацію, згадану у розмові раніше. Крім цього, за допомогою спеціальних запобіжників чат-бот може уникати неетичних відповідей і відхиляти неприйнятні запити, що містять вияви жорстокості, расизму чи сексизму.

Попри те, що сьогодні ChatGPT є найбільш “просунутою” технологією з використанням штучного інтелекту, він дуже далекий від ідеалу. Ще у грудні користувачам вдалося знайти спосіб, як обходити фільтри та знімати етичні обмеження, задані розробниками. Виявилося, що можна змусити чат-бот вжитися у роль іншого ШІ, який буде брехати, спілкуватися на заборонені теми і використовувати нецензурну лексику. Такий метод називається DAN («Do anything now») і перекладається як «Тепер роби що завгодно».

Самі розробники в описі продукту попереджають, що іноді чат-бот може виконувати неприйнятні вказівки і демонструвати ознаки упередженої поведінки. А ще надавати, на перший погляд, правдоподібні, та насправді хибні чи безглузді відповіді. Частково це пояснюється тим, що ChatGPT володіє даними лише до 2022 року, поза цим проблеми стосуються недосконалості самої технології. ChatGPT має обмеження в кількості слів, що становить майже 25 тис., хоч інші звіти натякають на обмеження в 8 тис. Тобто існує ймовірність, що розмова з ChatGPT буде перервана посередині. Платформа просто створює стиснене повідомлення самостійно для кращого розуміння контексту [5].

Крім того, ChatGPT уже встиг посіяти зерна паніки в академічних та професійних колах. Працівники навчальних закладів непокояться, що учні та студенти використовують нову технологію для написання домашніх робіт, творчих завдань і дослідницьких проєктів. На допомогу тут приходять програми, які дозволяють відрізнити текст, написаний людиною, від того, що був згенерований штучним інтелектом. Однак в окремих країнах і регіонах

чат-бот уже заборонили, а в інших – напевно шукатимуть нові рішення та підходи у відповідь на існуючі виклики.

ChatGPT був модифікований на основі GPT-3.5 з використанням методів навчання як з учителем, так і з підсиленням. В обох підходах люди-тренери використовувалися для поліпшення продуктивності моделі. У випадку навчання з учителем модель отримувала бесіди, де тренери виконували обидві ролі: користувача і асистента з штучним інтелектом. На етапі підсилення інструктори-люди спочатку оцінювали відповіді, які модель створила у попередньому розмові. Ці оцінки використовувалися для створення моделей винагород, на яких модель додатково піддавалася модифікації за допомогою кількох ітерацій методу Proximal Policy Optimization. Алгоритми Proximal Policy Optimization мають вигоди щодо витрат порівняно з алгоритмами Region Policy Optimization; вони скасовують багато обчислювально витратних операцій із вищою ефективністю.

Моделі були навчені в співпраці з Microsoft на їхній суперкомп'ютерній інфраструктурі Azure. Порівняно зі своїм попередником, InstructGPT, ChatGPT намагається зменшити кількість шкідливих та вводячих в оману відповідей. Наприклад, тоді як InstructGPT розглядає повідомлення про те, що Христофор Колумб прибув в США в 2015 році як правдиве, ChatGPT використовує інформацію про подорожі Колумба та сучасний світ, включаючи уявлення про особистість Колумба, щоб побудувати відповідь, припускаючи, що сталося б, якщо б Колумб приплив до США в 2015 році. Дані, які доступні у ChatGPT, включають справочні сторінки, інформацію про інтернет-феномени та інформацію про мови програмування.

Навіть ChatGPT відстежує стан користувача, пам'ятаючи попередні питання і відповіді, дані, надані в тому ж самому розмові. Це, за думкою деяких журналістів, може дозволити використовувати ChatGPT як персоналізованого терапевта. У спробі запобігти видаванню образливих

результатів від запитань користувачів і отриманню образливих відповідей, в ChatGPT запити проходять через API модерації, а потенційно расистські або сексистські підказки відхиляються.

У ChatGPT є численні обмеження. Модель винагороди ChatGPT, розроблена з урахуванням контролю людини, може бути перевантаженою та, таким чином, погіршувати продуктивність згідно з законом Goodhart's. Під час навчання рецензенти надавали перевагу більш довгим відповідям, незалежно від фактичного розуміння чи фактичного змісту. Дані у ChatGPT можуть страждати від алгоритмічної упередженості; відповіді, які включають розмиті описи осіб, таких як генеральний директор, можуть генерувати відповідь, що передбачає, що така людина, наприклад, є білим чоловіком.

ChatGPT може бути використаний для написання коду або створення плагінів, але код повинен бути сумісний з GPL. Спочатку сервіс не мав можливості отримання актуальної інформації. Однак з березня 2023 року ChatGPT почав надавати користувачам доступ до Інтернету і надав плагіни у версії ChatGPT Plus. 28 серпня 2023 року OpenAI представила корпоративну модель чат-бота ChatGPT Enterprise з передовими можливостями захисту даних і необмеженим доступом до GPT-4. 25 вересня 2023 року у ChatGPT ввели функції розпізнавання та генерації голосу, а також розпізнавання зображень. Крім того, сервіс отримав можливість пошуку інформації в Інтернеті з прямими посиланнями на джерела і перестав обмежуватися даними до вересня 2021 року.

ChatGPT отримав загалом позитивні відгуки. Нью-Йорк таймс назвав його «найкращим чат-ботом зі штучним інтелектом, що коли-небудь ставав доступним для широкого загалу»[6]. Саманта Лок із Гардіан зазначила, що він зумів згенерувати «вражаюче детальний» і «подібний до створеного людиною» текст[7]. Технічний автор Ден Гілмор використав ChatGPT для виконання студентського завдання і, виявивши, що згенерований текст був на

рівні з текстом, що міг би створити хороший студент, висловив думку, що «академічним колам доведеться зіткнутися з дуже серйозними проблемами». Алекс Кантровіц з журналу Slate похвалив відповіді ChatGPT на питання, що стосувались Третього Рейху, зокрема те, що твердження про будівництво Гітлером автобанів у Німеччині він парирував інформацією про використання нацистами примусової праці.

У грудні 2022 року економіст Пол Кругман опублікував статтю, де висловив думку, що ChatGPT вплине на попит на інформаційних працівників[4]. Джим Вінсент із The Verge вбачає в вірусному успіху ChatGPT свідчення того, що штучний інтелект став мейнстримом. Стівен Марке відмітив у журналі The Atlantic, що вплив ChatGPT на академічні кола й особливо на прикладні есе ще потрібно буде зрозуміти[7]. Вчитель середньої школи в Каліфорнії і письменник Деніел Герман написав, що ChatGPT ознаменує «кінець англійської мови в середній школі».

Фактична точність ChatGPT, окрім іншого, була поставлена під сумнів. Майк Перл із Mashable тестував ChatGPT багатьма питаннями. В одному з прикладів він попросив модель назвати «найбільшу країну в Центральній Америці, яка не є Мексикою», без уточнення за яким саме параметром слід порівнювати країни. ChatGPT відповів, що це Гватемала, хоча, якщо порівнювати за площею країни, а не за населенням або ВВП, то правильна відповідь буде — Нікарагуа. У грудні 2022 року сайт Stack Overflow заборонив використання ChatGPT для генерації відповідей на запитання, посиляючись на фактично неоднозначний характер відповідей ChatGPT. Економіст Тайлер Коуен висловив занепокоєність стосовно впливу ChatGPT на демократію, адже в будь-якої людини буде можливість створювати автоматичні коментарі в спробі повпливати на процес прийняття нових нормативних актів. Акс Шарма з Bleeping Computer відмітив, що ChatGPT здатен створювати шкідливі програми і фішингові електронні листи[10].

2.2 Фреймворк Angular: Можливості

2.2.1 Angular

Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) — написаний на TypeScript (рисунок 2.5) front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який був переосмислений та перероблений тією ж командою розробників [1].

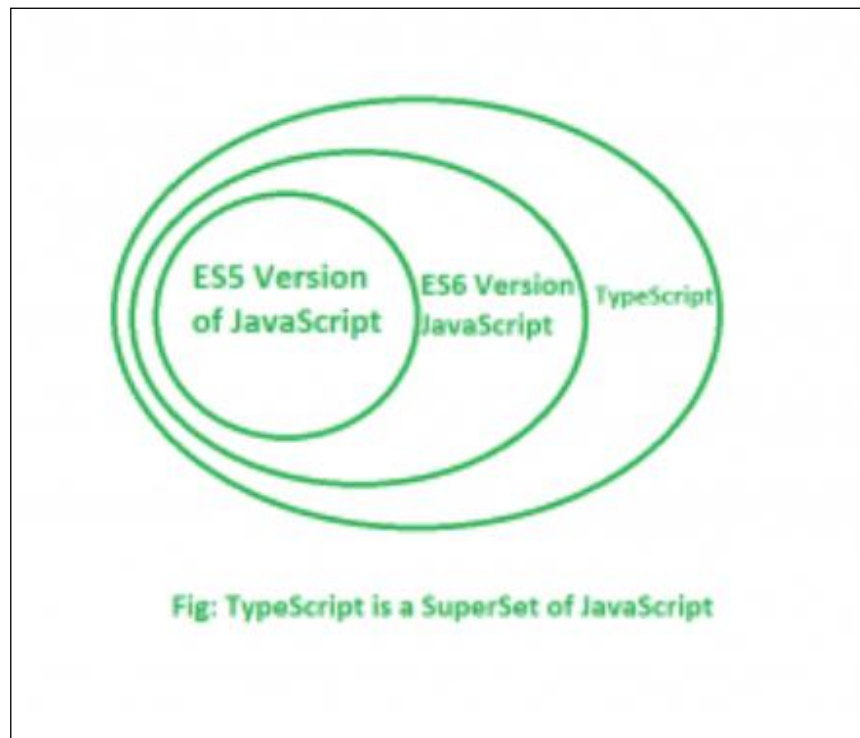


Рисунок 2.5 – Вигляд покриття TS над JS [10]

Angular вирізняється своєю компонентною архітектурою, що є фундаментальним аспектом фреймворку. Ця архітектура дозволяє будувати додатки як набори невеликих, автономних компонентів [12], кожен з яких

відповідає за конкретну частину функціоналу. Кожен компонент Angular має визначений клас, шаблон та метадані, що визначають його поведінку та вигляд. Компоненти можуть використовуватися незалежно один від одного, дозволяючи створювати модульні та перевикористовувані частини коду. Компоненти Angular пройшли багатоступінчастий життєвий цикл, що надає можливість втручатися в різні етапи їхнього існування. Цей фреймворк надає механізми для взаємодії між компонентами через властивості, події та спільні сервіси. Він надає зручні інструменти для створення компонентів, включаючи використання Angular CLI. Фреймворк підтримує модуль роутінгу, що дає можливість керувати навігацією та виводити відповідні компоненти для різних URL-адрес.

Angular виділяється використанням TypeScript [13], що вносить суттєві переваги в розробку front-end додатків. Однією з головних переваг TypeScript є статична типізація, що дозволяє визначати типи змінних та параметрів функцій під час компіляції. Це робить код більш надійним та полегшує виявлення помилок ще до виконання програми. TypeScript підтримує об'єктно-орієнтовану парадигму програмування, надаючи можливість використовувати класи, спадкування, інтерфейси та інші концепції ООП. Це поліпшує читабельність та організацію коду. TypeScript базується на стандарті ECMAScript 6 (ES6) і включає його функціональність, що робить його потужним та сучасним інструментом розробки. TypeScript інтегрується з багатьма популярними інструментами рефакторингу та аналізу коду, що спрощує роботу з великими проектами та дозволяє швидко виявляти та виправляти проблеми в коді. TypeScript - це розширення JavaScript, що дозволяє використовувати існуючий код JavaScript в проекті без проблем. Також, TypeScript компілюється в JavaScript, що гарантує сумісність з усіма браузерами та середовищами виконання.

Angular вирізняється потужною системою сервісів та ефективним управлінням залежностями між компонентами. Сервіси в Angular представляють собою продовження логіки додатку, яка не пов'язана безпосередньо з представленням чи взаємодією з користувачем. Вони використовуються для організації спільної логіки та надають можливість компонентам обмінюватися даними. Однією з ключових особливостей сервісів є їхня ін'єкція в компоненти [14]. Ін'єкція забезпечує доступ до функціональності сервісу у будь-якому компоненті, дозволяючи легко об'єднувати та розділяти логіку. Залежності в Angular виражають стосунки між різними частинами додатку та дозволяють ефективно використовувати та організовувати код. Механізм ін'єкції залежностей в Angular дозволяє визначити, як і коли ін'єктити залежності в компоненти. Ця архітектура сприяє зменшенню повторення коду та полегшує розширення та модифікацію додатку. Крім того, вона підтримує легкість тестування, оскільки логіка розділена на окремі частини, що дозволяє ефективно тестувати кожен компонент. Система сервісів та залежностей в Angular є однією з причин, чому цей фреймворк є потужним та гнучким інструментом для розробки великих та складних додатків.

Екземпляр компонента має життєвий цикл (рисунк 2.6), який починається, коли Angular ініціалізує клас компонента і рендерить вид компонента разом із його дочірніми видами. Життєвий цикл продовжується з обнаруженням змін, коли Angular перевіряє, чи змінилися прив'язані до даних властивості, і в разі потреби оновлює як вид, так і екземпляр компонента. Життєвий цикл завершується, коли Angular знищує екземпляр компонента і видаляє його рендерний шаблон з DOM. Директиви мають схожий життєвий цикл, оскільки Angular створює, оновлює та знищує екземпляри в ході виконання.

Ваш додаток може використовувати методи "перехоплення" життєвого циклу, щоб взаємодіяти з ключовими подіями у життєвому циклі компонента чи директиви, щоб ініціалізувати нові екземпляри, ініціювати об'явлення змін при необхідності, реагувати на оновлення під час об'явлення змін та видаляти екземпляри перед їхнім видаленням.

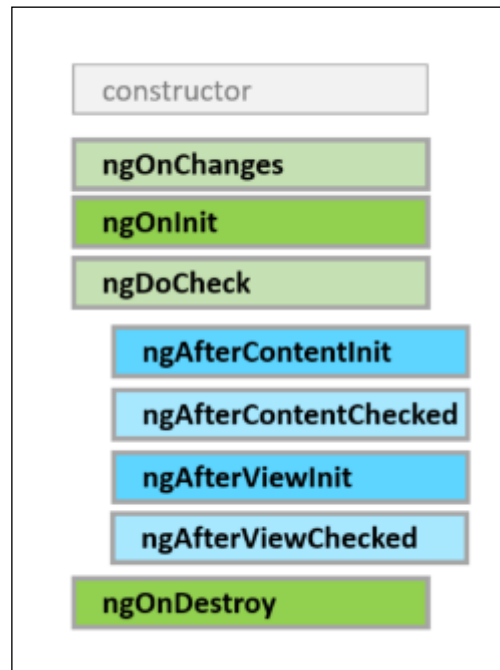


Рисунок 2.6 – Життєві цикли компонента

В Angular є певні сутності, ключові елементи, які дозволяють побудувати додаток та визначають його поведінку.

Компоненти відповідають за відображення і поведінку частини користувацького інтерфейсу. Кожен компонент має свою логіку, власні дані та вигляд, що може бути вкладеним у інші компоненти.

Директиви дозволяють змінювати або розширювати DOM та його поведінку. Це може бути зміна вигляду елементів DOM, додавання/видалення елементів, чи взагалі зміна структури DOM.

Модулі групують пов'язані компоненти, директиви та сервіси в логічні блоки. Вони допомагають організовувати код та роблять його більш масштабованим та легшим для управління.

Сервіси надають загальні функціональності та логіку, які можна використовувати в різних частинах додатка. Сервіси дозволяють виконувати операції, обмінюватися даними між компонентами та забезпечувати реюзабельність коду.

Пайпи використовуються для обробки та форматування даних, які виводяться в шаблоні. Angular має вбудовані пайпи, а також можливість створювати власні.

Роути дозволяють навігацію між різними відображеннями та компонентами в додатку. Вони дозволяють визначати, який компонент має бути відображений при зміні URL.

Ці сутності дозволяють розбити додаток на модульні та перевикористовувані частини, полегшуючи розробку, тестування та підтримку додатків на основі Angular.

2.2.3 RxJS

RxJS (Reactive Extensions for JavaScript) - це бібліотека для роботи з асинхронним програмуванням та реактивним програмуванням в JavaScript. У контексті Angular, RxJS використовується для роботи з реактивним підходом до програмування, де дані і події розглядаються як потоки, а не статичні значення.

RxJS дозволяє зручно працювати з асинхронним кодом та подіями. Всі асинхронні операції можна розглядати як потоки подій, що робить код більш читабельним та підтримуваним. RxJS вбудовує принципи реактивного програмування, що полегшує обробку та взаємодію зі змінними значеннями

відразу після їх зміни. Бібліотека містить велику кількість потужних операторів для маніпулювання та обробки потоків даних, таких як `map`, `filter`, `merge`, і багато інших (рисунок 2.7). RxJS дозволяє ефективно обробляти та передавати помилки в потоках, полегшуючи налагодження та виправлення [15].

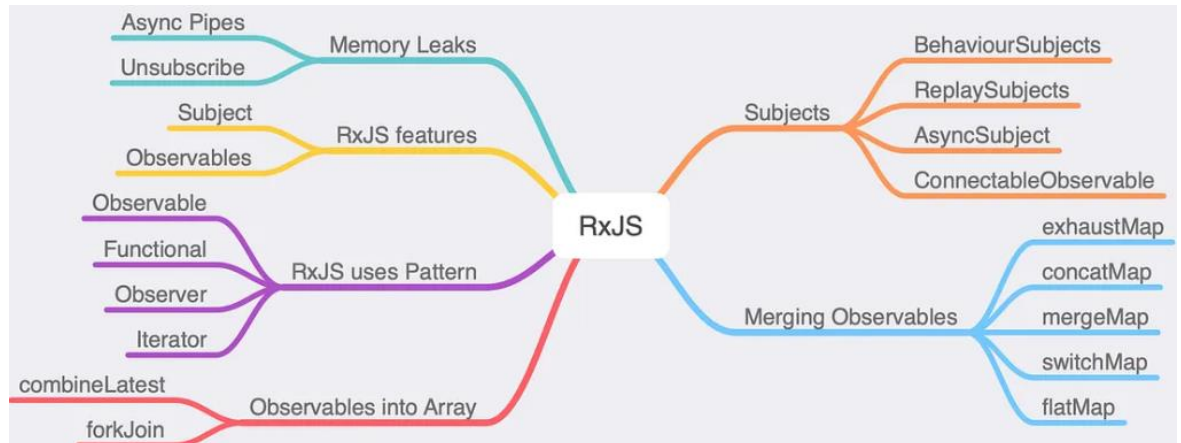


Рисунок 2.7 – Методи RxJS [5]

Вивчення RxJS може виявитися складним для новачків, особливо тих, хто не має досвіду з реактивним програмуванням. Він вимагає часу для вивчення та розуміння концепцій. В деяких випадках використання RxJS може привести до зайвого обсягу коду, особливо для простих асинхронних завдань, де стандартні проміси можуть бути зручнішими. У деяких випадках може виникнути ситуація, коли визначення потоку занадто докладне та складне для простих завдань.

В Angular RxJS використовується для роботи з подіями, асинхронними викликами, HTTP-запитами та іншими аспектами розробки. Наприклад, використання `Observable` та операторів RxJS для підписки на події компонентів, взаємодії з сервісами, та обробки HTTP-відповідей.

2.2.4 Jasmine

Jasmine - це дуже популярний фреймворк для розробки на мові JavaScript у стилі Behavior-Driven Development (BDD, де тести пишуться перед написанням фактичного коду) для тестування одиниць (unit testing) JavaScript-додатків. Він надає засоби, які можна використовувати для автоматизованого виконання тестів як для синхронного, так і для асинхронного коду [16].

Jasmine має багато функцій, таких як:

- а) він швидкий та має низький рівень накладань і не має зовнішніх залежностей;
- б) це бібліотека, яка містить все необхідне для тестування вашого коду;
- в) він доступна як для середовища Node, так і для браузера;
- г) він може використовуватися з іншими мовами, такими як Python та Ruby;
- д) для своєї роботи вона не вимагає DOM;
- е) він надає чистий та легко зрозумілий синтаксис, а також багатий і прямий API;
- ж) можна використовувати природну мову для опису тестів та очікуваних результатів;
- з) jasmine — це інструмент із відкритим вихідним кодом, який доступний за дозвоільною ліцензією MIT. На момент написання цієї статті останньою основною версією є Jasmine 3.0 , яка надає нові функції та деякі критичні зміни.

2.3 Технічні аспекти інтеграції Angular з генеративним AI

Підключення Angular до систем генеративного AI вимагає ретельного розгляду технічних аспектів для ефективної інтеграції та оптимального використання обох технологій. Нижче представлено ключові етапи [17] і технічні аспекти інтеграції Angular з генеративним AI:

- а) вибір технології генеративного ШІ;
- б) створення RESTful API;
- в) аутентифікація та безпека;
- г) управління запитами та обробка відповідей;
- д) інтеграція з UI;
- е) оптимізація динамічності;
- ж) тестування та налагодження.

Ретельний відбір технології генеративного ШІ є критичним етапом, оскільки від цього вибору залежить якість та ефективність роботи системи. Наприклад, OpenAI GPT може бути обрано для генерації текстового контенту [18]. OpenAI GPT є потужним інструментом з великим обсягом доступної інформації, що дозволяє створювати якісний та різноманітний текст

Для взаємодії між Angular та генеративним ШІ, ключовим є створення RESTful API. Ця архітектура полегшує обмін даними між клієнтською та серверною стороною [19]. RESTful підхід дозволяє легко та ефективно керувати ресурсами, що є критичним у взаємодії з генеративним ШІ.

Забезпечення безпеки API ключів та реалізація механізмів аутентифікації є важливим елементом. Докладні настанови щодо цього можна знайти у джерелі [20]. Надійна аутентифікація важлива для захисту системи від несанкціонованого доступу.

Використання Angular сервісів та HTTP клієнта є фундаментальним для ефективної взаємодії з RESTful API [21]. Angular HTTP клієнт дозволяє

здійснювати HTTP запити, а сервіси обробляють отримані відповіді. Це забезпечує ефективну обробку запитів та обмін даними з сервером.

Розробка Angular компонентів та сервісів - ключовий етап для створення інтерфейсу користувача [22]. Це включає введення продуктів, отримання рецептів та інші взаємодії. Використання компонентної архітектури Angular дозволяє створити модульний та ефективний інтерфейс.

Використання функціоналу Angular для динамічного оновлення контенту сторінок є важливим для коректної взаємодії з генеративним ШІ [23]. Angular дозволяє ефективно оновлювати вміст сторінок без перезавантаження, що поліпшує враження від використання додатку.

Реалізація єдиної системи тестування - це ключовий етап для забезпечення якості та стабільності системи [24]. Тестування повинно охоплювати якість API, так і коректність відображення та взаємодії в Angular додатку. Посилання на Angular Testing Guide може служити основою для розробки системи тестування.

2.4 Висновки за розділом 2

У другому розділі було розглянуто ключові технічні компоненти, які визначають можливості та обмеження розробки інтелектуального асистента для генерації рецептів з використанням платформи chatGPT та фреймворку Angular.

Платформа chatGPT, надаючи потужність генеративного ШІ, відкриває широкі можливості для реалізації інтелектуального асистента. Її здатність генерувати текстовий контент з великою мірою природності дозволяє створювати персоналізовані та цікаві рецепти. Однак, важливо враховувати обмеження, такі як можливість генерації неправдоподібних чи неситуаційних рецептів, що виникає з натренованості моделі на доступних даних.

Фреймворк Angular, завдяки своїй компонентній архітектурі та зручним інструментам розробки, виявляється відмінним вибором для створення клієнтської частини інтелектуального асистента. Можливості Angular забезпечують легку інтеграцію та ефективне управління візуалізацією та взаємодією з користувачем, що є ключовим елементом функціональності асистента.

Взаємодія платформи chatGPT із фреймворком Angular відкриває шлях до створення динамічного та інтуїтивно зрозумілого інтерфейсу для введення продуктів та отримання рецептів. Використання Angular дозволяє ефективно керувати компонентами, що взаємодіють із штучним інтелектом, та забезпечує оптимальну реалізацію функцій асистента.

Узагальнюючи, об'єднання потужностей платформи chatGPT та фреймворку Angular відкриває перспективи для створення інтелектуального асистента, який забезпечить користувачів персоналізованими та креативними рецептами, а також інтуїтивно зрозумілим і зручним інтерфейсом взаємодії.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО АСИСТЕНТА ДЛЯ ГЕНЕРАЦІЇ РЕЦЕПТІВ

3.1 Розробка алгоритму генерації рецептів

Першим етапом є визначення критеріїв генерації рецептів. Це включає в себе врахування доступних продуктів, обмежень користувача, вподобань щодо стилів кулінарії, та можливість врахування етнічних особливостей. Для отримання текстового контенту рецептів використовується платформа chatGPT. Описується процес інтеграції, включаючи передачу запитань, обробку відповідей та врахування контексту для забезпечення логічності та зв'язку між кроками генерації. Алгоритм включає в себе механізми оптимізації та фільтрації результатів. Це дозволяє підтримувати якість та відповідність рецептів вимогам користувача.

Головна мета цього етапу розробки полягає в створенні ефективного та зручного інтерфейсу, який не лише дозволяє користувачам легко взаємодіяти з системою, але й генерує максимально релевантні запити до штучного інтелекту. Центральний етап - формування prompt для взаємодії з API chatGPT. Особлива увага приділяється точному та релевантному сформулюванню запитів, щоб отримати бажані результати. Підходи до цього можуть включати уточнення інгредієнтів, кулінарних уподобань та інших факторів, що визначають унікальність рецепту. Початок розробки зазвичай включає в себе визначення базової архітектури проекту. Це охоплює створення компонентів, сервісів, роутінгу та інтерфейсів. Розділення HTML на компоненти слідує принципам ООП, забезпечуючи легке розширення та підтримку. Створення користувацького інтерфейсу здійснюється за допомогою HTML та SCSS. Використання методології BEM в стилізації полегшує розуміння коду та сприяє підтримці проекту. Розділення різних частин UI на компоненти дозволяє зберігати код чистим та легко розширюваним. Для взаємодії з

платформою chatGPT створюється окремий сервіс, який формує та виконує запити. Підключення вимагає отримання API ключа, який забезпечує безпечний та автентичний обмін даними між додатком та платформою штучного інтелекту.

На рисунку 3.1 зображений загальний алгоритм роботи програми.



Рисунок 3.1 – Алгоритм роботи програми

3.2 Програмна реалізація та інтеграція з chatGPT

Структура проекту (рисунок 3.2) складається з декількох компонентів, загального модулю, сервісу інтеграції та файлів стилізації.

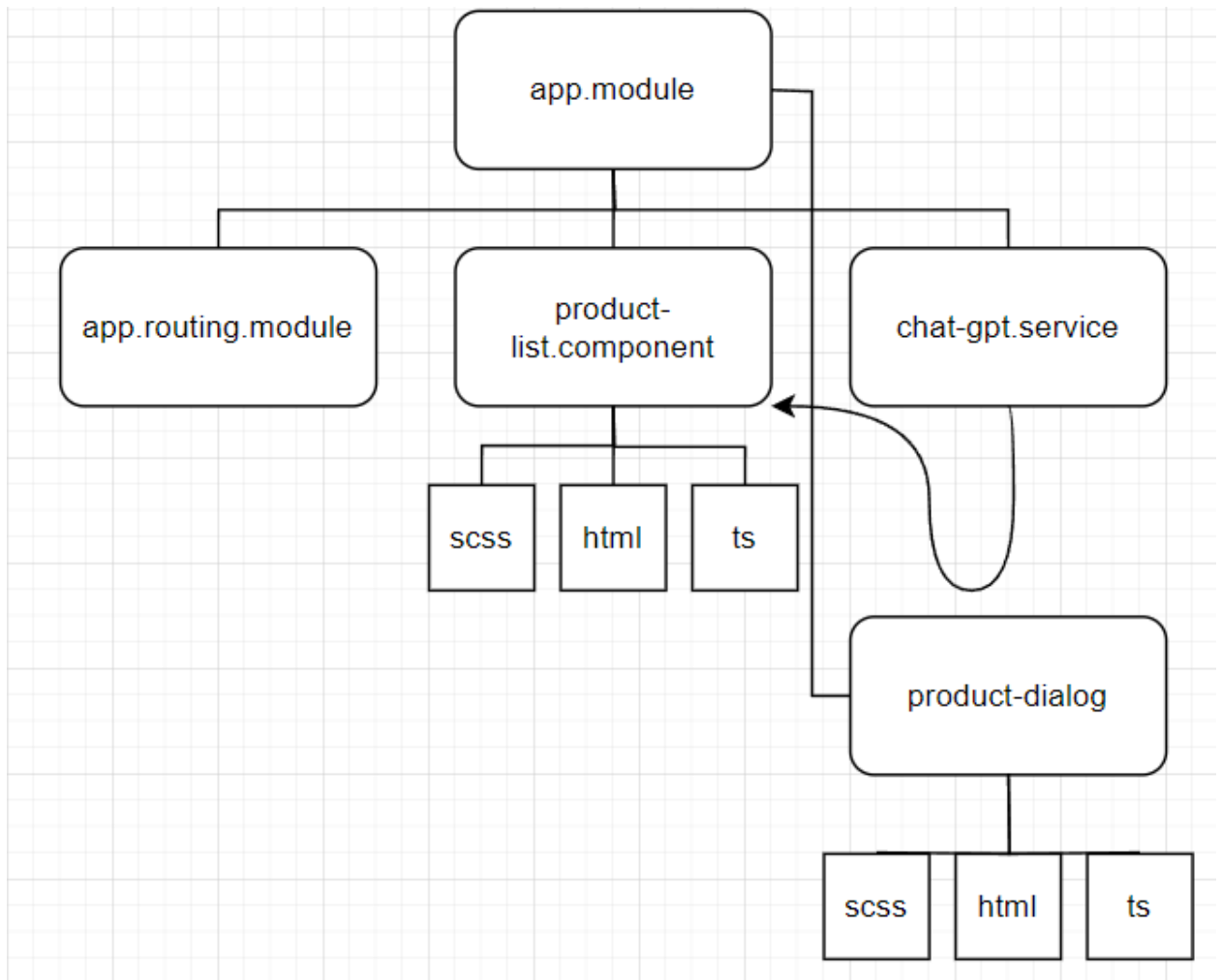


Рисунок 3.2 - Структура проекту

Для того щоб інтегрувати API chatGPT нам необхідно встановити NPM пакет який полегшує цю задачу. Команда "npm i openai" встановить останню версію пакета [14]. Після успішного встановлення ця бібліотека додається у файл залежностей (рис. 3.2) та встановлюється в node_modules.

Після цього необхідно створити сервіс (рисунок 3.3) де буде формуватися запит.

```
"dependencies": {
  "@angular/animations": "^16.0.0",
  "@angular/common": "^16.0.0",
  "@angular/compiler": "^16.0.0",
  "@angular/core": "^16.0.0",
  "@angular/forms": "^16.0.0",
  "@angular/platform-browser": "^16.0.0",
  "@angular/platform-browser-dynamic": "^16.0.0",
  "@angular/router": "^16.0.0",
  "@fortawesome/angular-fontawesome": "^0.13.0",
  "openai": "^4.0.0",
  "rxjs": "~7.8.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.13.0"
},
"devDependencies": {
```

Рисунок 3.3 - Залежності проекту

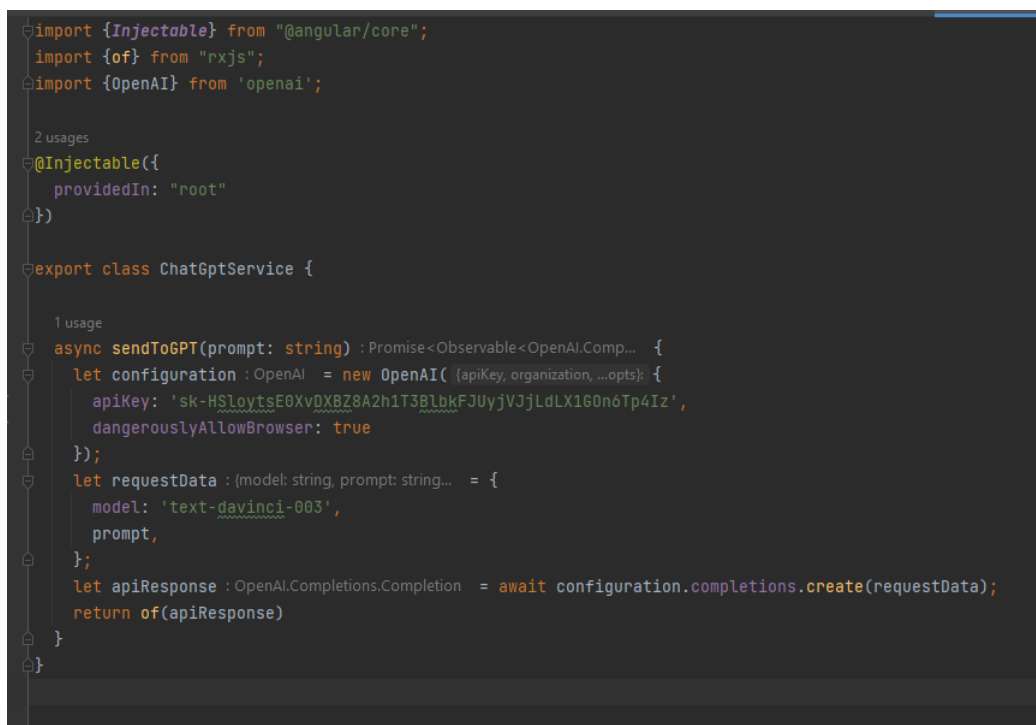
В данному випадку сервіс створено як root. Це означає що він буде завантажуватись одразу із завантаженням проекту. У даній реалізації це не впливає ніяк на оптимізацію проекту, але у великих застосунках необхідно виносити сервіси в окремі модулі, так як він створюється як singleton, що говорить про єдиний екземпляр. Якщо користувач не зайшов до того розділу де необхідний цей сервіс, його і не потрібно завантажувати. В даній реалізації існує тільки один app.module, тому цим можна знехтувати.

Використовуючи бібліотеку openai, потрібно створити новий екземпляр класу OpenAI, передавши туди apiKey та у даному випадку прапор про те що виконання цього запиту через фронт є небезпечним. Після цього необхідно відправити запит передавши в нього модель AI та prompt. Результат буде повертатися як Observable.

3.3 Створення Angular додатку для введення продуктів

В початкових етапах розробки було надано особливу увагу створенню інтуїтивно зрозумілого та естетичного інтерфейсу користувача (UI). Для досягнення цієї мети використовувався препроцесор SCSS. Обрана технологія є значно зручнішою порівняно з нативним CSS, що дозволяє ефективно організувати та підтримувати стилі великих проектів. Методологія BEM використовувалась для створення читабельних та структурованих назв класів, сприяючи правильній організації стилів.

У цьому контексті важливим є головний файл стилів `styles.scss` (рисунок 3.4), який містить набір стилів, що використовуються у всьому застосунку. Цей файл є центральним елементом для забезпечення стилізації та єдності в зовнішньому вигляді додатку.



```
import {Injectable} from "@angular/core";
import {of} from "rxjs";
import {OpenAI} from 'openai';

2 usages
@Injectable({
  providedIn: "root"
})

export class ChatGptService {

1 usage
  async sendToGPT(prompt: string) : Promise<Observable<OpenAI.Comp...  {
    let configuration : OpenAI = new OpenAI( {apiKey, organization, ...opts}: {
      apiKey: 'sk-HSloytsE0XvDXBZ8A2h1T3BlbkFJUyJlLdLX160n6Tp4Iz',
      dangerouslyAllowBrowser: true
    });
    let requestData : {model: string, prompt: string... = {
      model: 'text-davinci-003',
      prompt,
    };
    let apiResponse : OpenAI.Completions.Completion = await configuration.completions.create(requestData);
    return of(apiResponse)
  }
}
```

Рисунок 3.4 – Файл `chat-gpt.service.ts`

Крім того, створено імпорт файлу стилів для кнопок (рисунок 3.5).

```
@import "assets/scss/buttons";

html, body, * {
  padding: 0;
  margin: 0;
  font-family: 'Roboto', sans-serif;
}

.container {
  margin: 4rem 10rem;
}

.ml-2 {
  margin-left: 1rem;
}

.mr-2 {
  margin-right: 1rem;
}

.mt-4 {
  margin-top: 2rem;
}

.mx-2 {
  margin-right: 1rem;
  margin-left: 1rem;
}

.my-2 {
  margin-top: 1rem;
  margin-bottom: 1rem;
}

.text-center {
  text-align: center;
}

.d-flex {
  display: flex;
}
```

Рисунок 3.5 – Файл styles.scss

Це рішення спрощує читання та управління кодом, оскільки стилі для кнопок винесено в окремий файл. Цей підхід до оформлення забезпечує чистоту та структурованість коду, дозволяючи легко вносити зміни та додавати новий функціонал. Поєднання SCSS і BEM створює основу для ефективного та гнучкого дизайну інтерфейсу користувача.

Структура мого проекту базується на двох основних компонентах, які взаємодіють для забезпечення функціоналу. Перший з них – це головний екран, де користувач може визначити, які продукти він має, і які з них повинні брати участь у запитах. На цьому екрані розташовані кнопка для взаємодії з

товарами та кнопка для створення рецепту. Деталі верстки подані у вказаному коді на рисунку 3.6. Цей код визначає основну структуру головного екрану, який забезпечує зручний вибір продуктів та надає користувачеві можливість взаємодіяти з додатком.

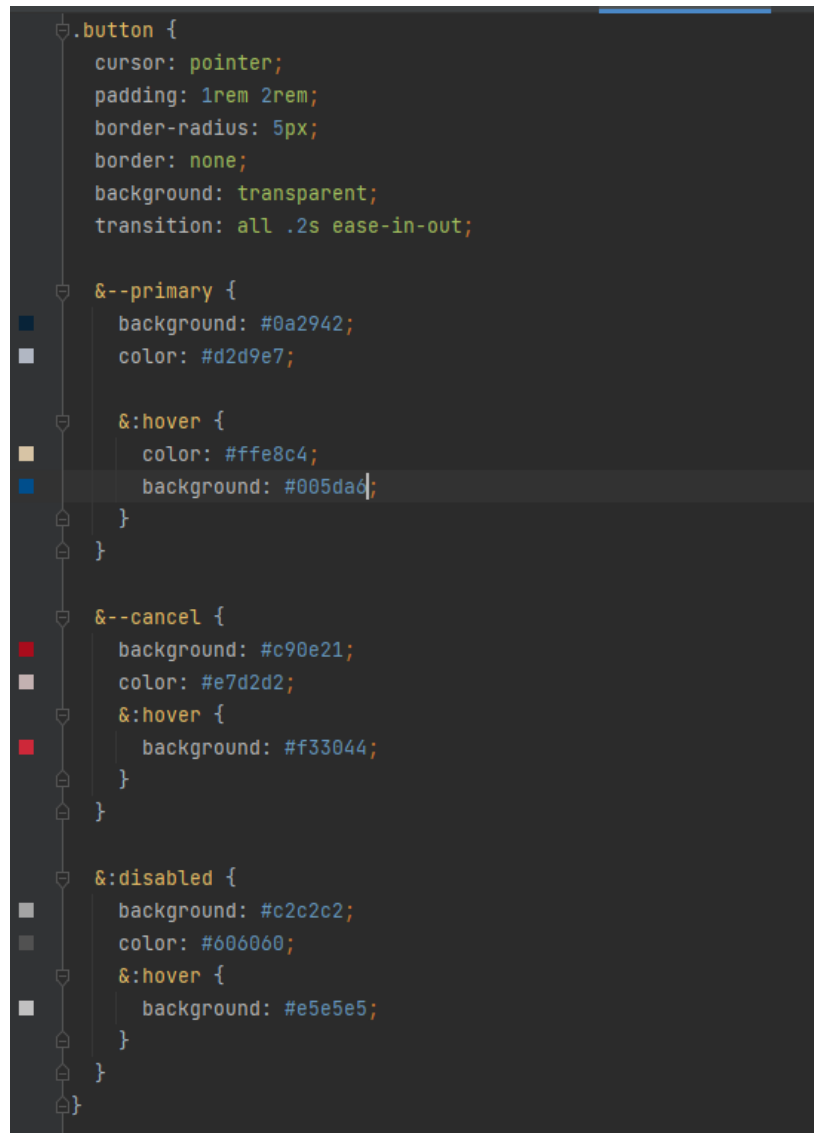


Рисунок 3.6 – Файл стилізації кнопок

Другий компонент (рисунок 3.7) - це модальне вікно, призначене для редагування та створення продуктів. Цей функціонал було винесено у вигляді окремого компоненту для полегшення перевикористання та уникнення

дублювання коду. Цей компонент визначає інтерфейс модального вікна, яке використовується для зручного редагування та додавання нових продуктів.

```

<div class="container">
  <div *ngFor="let product of productList" class="product">
    <span>{{product.title}}</span>
    <span>{{product.weight}} кг</span>
    <div class="d-flex">
      <button
        [ngClass]="{'button--primary': checkProductsForRequest(product), 'button--cancel': !checkProductsForRequest(product)}"
        class="button mr-2"
        (click)="addToRequest(product)">{{checkProductsForRequest(product) ? 'Обрати' : 'Прибрати'}}</button>
      <button class="button action mr-2" (click)="openModal( mode: 'edit', product)"><i class="fa fa-cog"></i></button>
      <button (click)="removeFromProductList(product)" class="button action"><i class="fa fa-trash-o"></i></button>
    </div>
  </div>
  <div class="mt-4">
    <button class="button button--primary mr-2" (click)="openModal( mode: 'add')">Додати продукти</button>
    <button class="button button--primary" [disabled]="!productsForRequest.length" (click)="sendRequest()">Згенерувати
      рецепт
    </button>
  </div>
</div>

<div *ngIf="showModal">
  <app-product-dialog [form]="form" [mode]="mode" (isClosed)="closeModal($event)"></app-product-dialog>
</div>

```

Рисунок 3.7 – Верстка компоненту головної сторінки

Щодо основної логіки, то вона виконується в компонентах. Перш за все при ініціалізації сторінки створюється форма. Ця форма буде використовуватись для взаємодії з продуктами. До форми належить 3 поля: id, title та weight. Для генерації форми використовується спеціальний сервіс angular який називається FormBuilder (рисунок 3.8).

```

<div class="modal-container">
  <p class="text-center my-2">{{modalTitle}}</p>
  <form [formGroup]="form">
    <input formControlName="title" type="text" placeholder="Назва">
    <input formControlName="weight" type="text" placeholder="Вага">

    <div class="modal__actions my-2">
      <button (click)="save()" class="button button--primary mr-2">Зберегти</button>
      <button (click)="close()" class="button button--cancel">Закрити</button>
    </div>
  </form>
</div>
<div (click)="close()" class="backdrop"></div>

```

Рисунок 3.8 – Верстка компоненту модального вікна

Також компонент містить в собі методи які виконуються для взаємодії з відкриття/закриттям модального вікна. В цих методах перевіряється тип дії, тобто це створення чи редагування і в залежності від цього форма або очищається, або заповнюється (рисунок 3.9).

```

1 usage
initForm(): void {
  this.form = this.fb.group( controls: {
    id: [null, Validators.required],
    title: [null, [Validators.required]],
    weight: [null, [Validators.required]]
  })
}

```

Рисунок 3.9 – Створення форми

Методи взаємодії з продуктами можна глянути на рисунку 3.10.

```

2 usages
openModal(mode: string, product?: any) :void {
  mode === 'edit' ? this.form.setValue(product) : this.form.reset()
  this.mode = mode
  this.showModal = true;
}

1 usage
closeModal(event: any) :void {
  if (event && event.form) {
    const existingProductIndex : number = this.productList.findIndex(
      (product : {id: number, title: string, we... } : boolean => product.id === event.form.id
    );

    if (existingProductIndex !== -1) {
      this.productList[existingProductIndex] = event.form;
    } else {
      this.productList.push({
        ...event.form,
        id: this.productList[this.productList.length - 1].id + 1
      });
    }
  }

  this.showModal = false;
}

```

Рисунок 3.10 – Методи взаємодії з модальними вікнами

Компонент з модальним вікном містить в собі дуже просту логіку (рисунок 3.11). Він повинен прийняти певні дані, повзаємодіяти з формою та віддати назад у головний компонент. Цей підхід забезпечує легку розширюваність та читабельність коду, роблячи додаток більш підтримуваним та ефективним.

```

1 usage
addToRequest(product: any): void {
  const index : number = this.productsForRequest.findIndex((p) : boolean => p.title === product.title);

  if (index !== -1) {
    this.productsForRequest.splice(index, deleteCount: 1);
  } else {
    this.productsForRequest.push(product);
  }
}

3 usages
checkProductsForRequest(product: any): boolean {
  return !this.productsForRequest?.find((item: any) : boolean => item.title === product.title)
}

1 usage
removeFromProductList(product: any): void {
  const index : number = this.productList.findIndex((p : {id: number, title: string, we... } : boolean => p.title === product.title);

  this.productList.splice(index, deleteCount: 1);
}

```

Рисунок 3.11 – Методи взаємодії з продуктами

На рисунку 3.12 показана логіка компонента з модальними вік.

```

@Input() form!: FormGroup
@Input() mode!: string
@Output() isClosed: EventEmitter<any> = new EventEmitter<any>()

modalTitle: string = ''

no usages
ngOnChanges(changes: SimpleChanges) : void {
  this.mode === 'edit' ? this.modalTitle = 'Edit product' : this.modalTitle = 'Add new product'
}

2 usages
close(): void {
  this.isClosed.emit()
}

1 usage
save(): void {
  this.isClosed.emit( value: {
    form: this.form.value,
    mode: this.mode
  })
}

```

Рисунок 3.12 – Логіка компонента з модальними вік

На виході можна побачити готовий застосунок (рисунки 3.13 – 3.17), який взаємодіє з chatGPT та здатен генерувати рецепти з тих продуктів які ми вказали

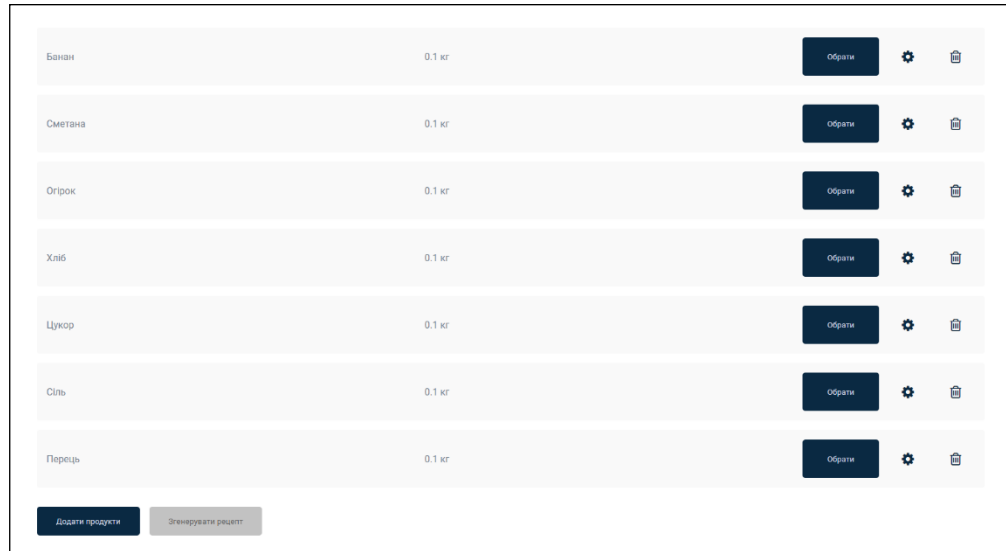


Рисунок 3.13 – UI готового застосунку

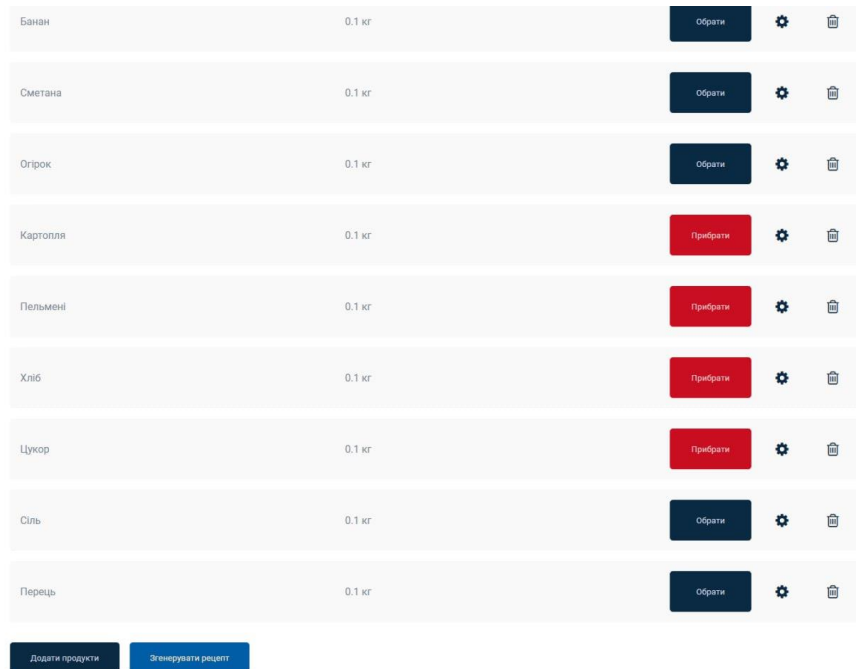


Рисунок 3.14 – Вибір потрібних товарів для генерації рецепта

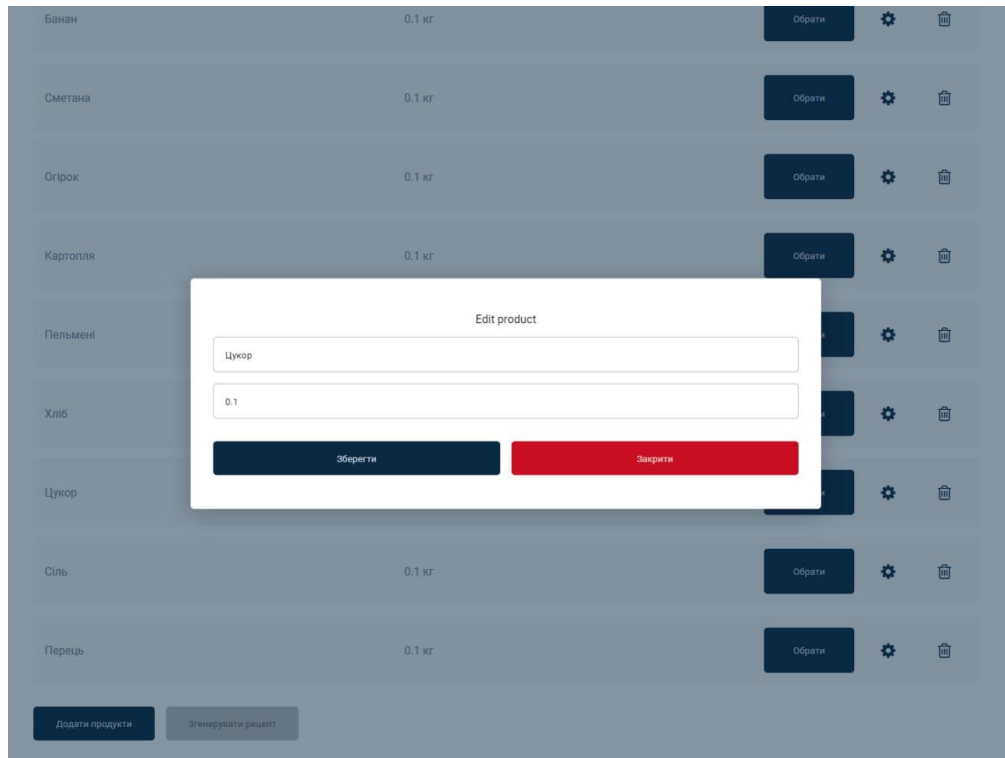


Рисунок 3.15 – Редагування товару

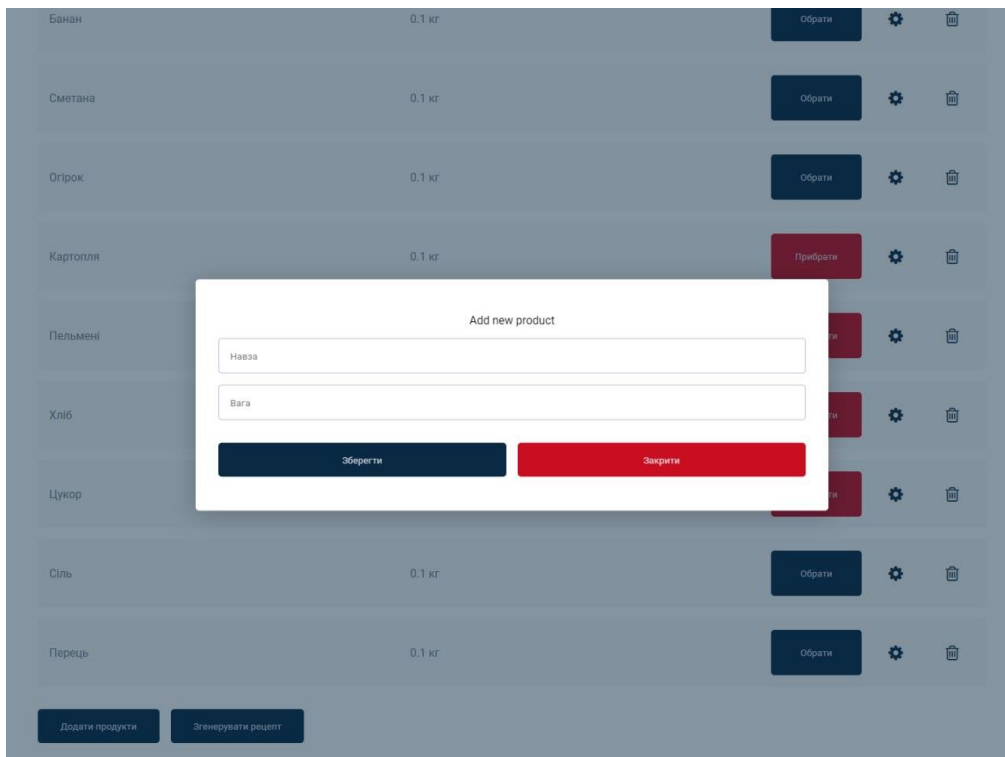


Рисунок 3.16 – Створення товару

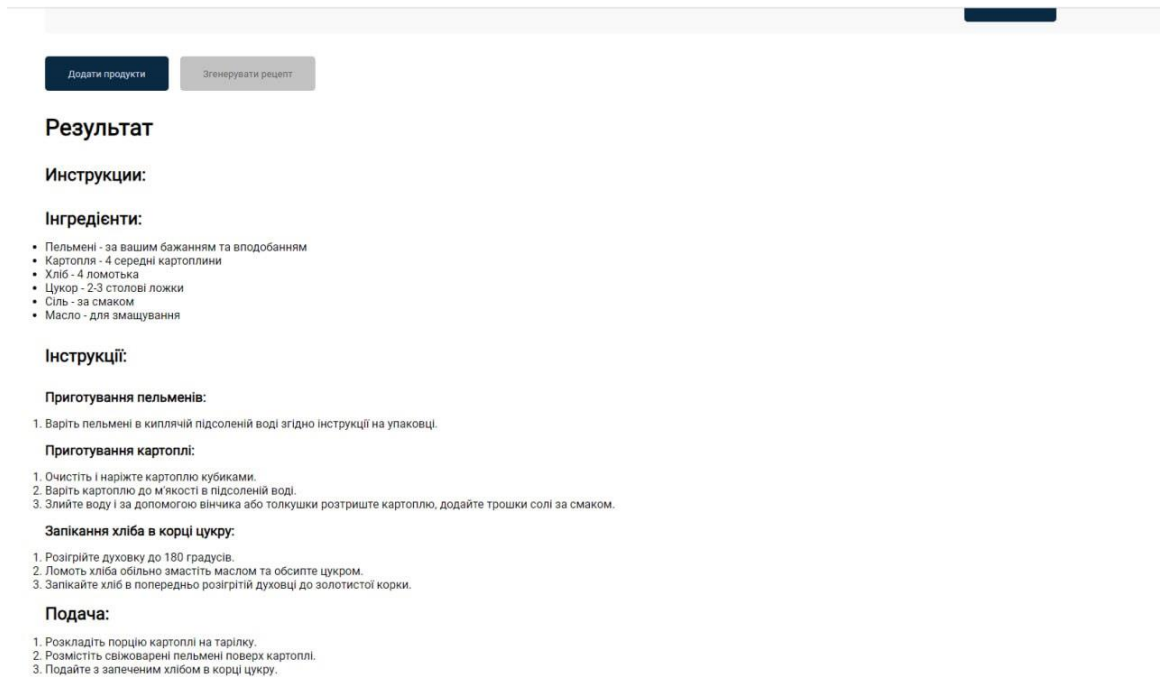


Рисунок 3.17 – Результат генерації після натискання кнопки "Згенерувати рецепт"

3.4 Висновки за розділом 3

У данному розділі було розглянуто докладну програмну реалізацію та інтеграцію Angular додатку з ChatGPT для генерації рецептів. На початкових етапах розробки було надано особливу увагу оформленню інтерфейсу користувача (UI) з використанням препроцесора SCSS та методології BEM для створення чистого та структурованого коду.

Структура проекту включає два ключові компоненти: головний екран та модальне вікно для редагування та створення продуктів. Головний екран дозволяє користувачу взаємодіяти з продуктами та створювати рецепти, а модальне вікно спрощує введення даних про продукти.

З використанням бібліотеки `openai`, був створений сервіс для взаємодії з API ChatGPT, що дозволяє отримувати рецепти на основі введених користувачем продуктів.

Кожен компонент має свою логіку та методи для взаємодії з іншими компонентами, а також з сервісом ChatGPT. Готовий застосунок може взаємодіяти з користувачем, генерувати рецепти та дозволяє легко редагувати та додавати нові продукти.

Всі ці кроки дозволяють створити ефективний та зручний застосунок, який може використовуватися для генерації рецептів на основі вказаних продуктів. Готовий UI додатку демонструє його гнучкість та здатність взаємодіяти з потужним ChatGPT для надання користувачам персоналізованих рецептів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Призначення й умови застосування додатку

Цей додаток розроблено для полегшення процесу генерації рецептів на основі продуктів, які користувач має в наявності. Основна мета застосунку - створення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачам швидко та ефективно отримувати персоналізовані рецепти.

Основні функції додатку полягають у тому що користувач може обрати продукти, які він має в своєму розпорядженні, використовуючи головний екран застосунку. На основі обраних продуктів додаток використовує ChatGPT для генерації рецептів. Користувач отримує рекомендації щодо того, як готувати страви з вказаних інгредієнтів. Додаток дозволяє користувачам легко редагувати інформацію про продукти та додавати нові, забезпечуючи актуальність та різноманіття вибору. Застосунок має інтуїтивно зрозумілий та дружелюбний інтерфейс, що дозволяє користувачам з легкістю навігувати та взаємодіяти з усіма його функціями.

Додаток призначено для любителів кулінарії, які хочуть експериментувати з приготуванням їжі та використанням наявних продуктів. Рекомендації додатку спрямовані на оптимізацію використання наявних продуктів, дозволяючи користувачам максимально раціонально використовувати складники при готуванні. Додаток підтримує творчий підхід до готування, дозволяючи створювати різноманітні та смачні страви навіть з обмеженого асортименту продуктів. За допомогою застосунку кожен може стати шеф-кухарем у власній кухні. Спробуйте нестандартні поєднання та створюйте свої гастрономічні шедеври. Застосунок призначений для широкого кола користувачів, забезпечуючи зручний та простий інтерфейс, який може використовувати будь-яка людина, незалежно від досвіду в готуванні. Навіть

якщо ви не є професійним кухарем, зручний та простий інтерфейс дозволить вам насолоджуватися процесом готування та відкривати нові гастрономічні горизонти.

4.2 Характеристики додатку

Frontend реалізовано на Angular і TypeScript, використовуючи SCSS для стилізації. Спеціальний сервіс для взаємодії з OpenAI API, що забезпечує комунікацію з ChatGPT. База даних не використовується, оскільки головна функціональність заснована на отриманні рецептів від ChatGPT.

Функціональність додатку включає вибір продуктів на головному екрані, генерацію рецептів через ChatGPT, можливість редагування та додавання продуктів. Інтерфейс стилізований за допомогою SCSS, з дотриманням методології ВЕМ для чіткості та зрозумілості коду.

Технічні переваги включають гнучкість та адаптивність для різних пристроїв, використання сучасних технологій штучного інтелекту, простоту та зручність інтерфейсу. Застосунок також передбачає можливості розширення бази даних, інтеграції з іншими ресурсами, покращення AI моделі та мультимовність для більшого охоплення аудиторії.

Всі ці характеристики роблять додаток передовим інструментом для експериментів у готуванні та надають можливість користувачам насолоджуватися творчим підходом до кулінарії.

4.3 Інструкція по експлуатації додатку

Інструкція по експлуатації додатку розроблена для забезпечення зручного та ефективного користування усіма його функціями. Перед початком використання додатку, рекомендується ознайомитися з наступними кроками:

- а) запустіть додаток на своєму пристрої, перейшовши за відповідним посиланням або відкривши його ікону на домашньому екрані;
- б) після запуску вас вітатиме головний екран додатку. Тут ви можете вибрати продукти, які ви маєте в наявності, та обрати опції для генерації рецептів;
- в) натискайте на кнопки або використовуйте доступні опції для вибору продуктів, які ви хочете використовувати при готуванні;
- г) після вибору продуктів активуйте опцію створення рецепту. Додаток використає ChatGPT для генерації персоналізованих рекомендацій;
- д) щоб редагувати або додавати нові продукти, скористайтесь відповідними функціями на головному екрані або в модальному вікні;
- е) за бажанням, збережіть отриманий рецепт або надрукуйте його для подальшого використання.

Ця інструкція призначена для того, щоб забезпечити зручне та інтуїтивно зрозуміле користування додатком.

4.4 Тестування програмного додатку

Тестування програмного додатку включало в себе використання різних методів та інструментів для забезпечення його якості та надійності. Основні аспекти тестування включають автоматизовані тести за допомогою Jasmine, ручне тестування та здійснення функціональних перевірок.

Для автоматизації тестування використовувався фреймворк Jasmine. Написані тести покривали різні аспекти додатку, такі як взаємодія зі службами, обробка вхідних даних та функціональні можливості. Тестування забезпечувало перевірку очікуваних результатів та виявлення можливих дефектів.

Ручне тестування використовувалося для глибокої перевірки функціональності та взаємодії з додатком. Було перевірено сценарії взаємодії з інтерфейсом, перевірено коректність введення даних та переконано у правильності реакцій на введені команди.

Для перевірки безпеки додатку використовувалися автоматизовані засоби та проводилось мануальне тестування з метою виявлення потенційних вразливостей та вжиття заходів щодо їх усунення.

Здійснювалася перевірка сумісності додатку на різних пристроях та браузерах, включаючи популярні веб-переглядачі. Використання різних конфігурацій гарантувало коректне відображення та функціональність.

У ході тестування акцентувалася увага на валідацію введених даних. Всі форми та поля перевірялись на коректність обробки введених користувачем даних.

Всі вищезазначені методи тестування спільно забезпечили високий рівень якості та надійності програмного додатку, готового до успішного використання користувачами.

На рисунку 4.1 представлено вікно програми Капта v 6.4.2 для тестування, на рисунку 4.2 – результати проходження успішного тестування.

Chrome is being controlled by automated test software.

Karma v 6.4.2 - connected; test: complete;

Chrome 119.0.0.0 (Windows 10) is idle

 **Jasmine** 4.6.0
.....

8 specs, 0 failures, randomized with seed 29337

```
AppComponent
  • should have as title 'diploma'
  • should create the app

ProductListComponent
  • should initialize form on ngOnInit
  • should remove a product from the product list
  • should create
  • should open the modal in edit mode
  • should add a product to the request list
  • should open the modal in add mode
```

Рисунок 4.1 – Програма для тестування

```
PS C:\Users\nicks\Documents\diploma> ng test
✓ Browser application bundle generation complete.
13 12 2023 14:48:27.522:WARN [karma]: No captured browser, open http://localhost:9876/
13 12 2023 14:48:27.536:INFO [karma-server]: Karma v6.4.2 server started at http://localhost:9876/
13 12 2023 14:48:27.537:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
13 12 2023 14:48:27.551:INFO [launcher]: Starting browser Chrome
13 12 2023 14:48:28.254:INFO [Chrome 119.0.0.0 (Windows 10)]: Connected on socket h9jKrn8e6mu-RGwNAAAB with id 89173449
Chrome 119.0.0.0 (Windows 10): Executed 8 of 8 SUCCESS (0.008 secs / 0.113 secs)
TOTAL: 8 SUCCESS
```

Рисунок 4.2 – Показ успішності проходження тестів

4.5 Висновки за розділом 4

Розділ надає повний обзор процесів, пов'язаних із створенням, тестуванням та використанням розробленого додатку. Додаток розроблено для полегшення генерації рецептів на основі наявних продуктів. Його основна мета - створення зручного та інтуїтивно зрозумілого інтерфейсу для отримання персоналізованих рецептів. Додаток володіє високою

функціональністю, забезпечуючи користувачам можливість вибору продуктів, генерації рецептів та редагування інформації про продукти. Застосунок побудований із застосуванням сучасних технологій, що гарантує його ефективність та високу продуктивність. Надана інструкція дозволяє користувачам з легкістю взаємодіяти з додатком. Зокрема, вона включає в себе розгортання, введення продуктів, генерацію рецептів та редагування інформації. Всі кроки чітко пояснені, що сприяє швидкому освоєнню функціоналу. Розділ відображає використання різних методів тестування, включаючи автоматизовані тести з використанням Jasmine, мануальне тестування та тестування безпеки. Результати підтверджують високу якість та надійність програмного забезпечення. Загальний висновок за розділом підтверджує, що додаток готовий до ефективного використання, надійний та відповідає вимогам користувачів. Процес тестування гарантує високу якість та оптимальну продуктивність.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено аналіз сучасних методів та засобів генеративного штучного інтелекту, зокрема в контексті їх використання в гастрономії. Виявлено, що генеративний інтелект може стати потужним інструментом у створенні нових рішень для гастрономічної сфери.

На основі аналізу було визначено завдання дослідження, а саме розробка та реалізація інтелектуального асистента для генерації рецептів. Для досягнення цієї мети були використані платформа chatGPT та фреймворк Angular.

У розділі "Матеріали та методи" розглянуті можливості та обмеження використання платформи chatGPT, а також важливі аспекти роботи з фреймворком Angular та його компонентами.

Розділ "Розробка та реалізація інтелектуального асистента для генерації рецептів" включає в себе опис розробленого алгоритму генерації рецептів, програмної реалізації та інтеграції з chatGPT, а також створення Angular додатку для введення продуктів.

У розділі "Експлуатація, тестування та експериментальне дослідження" розглянуті практичні аспекти використання розробленого додатку, його характеристики та інструкції з експлуатації. Проведене тестування дозволило перевірити функціональність додатку та зробити висновки щодо його ефективності.

В цілому, результати дослідження підтверджують актуальність використання генеративного інтелекту в гастрономії та показують практичні можливості розробленого інтелектуального асистента. Робота внесе важливий внесок у розвиток області та може бути використана як основа для подальших досліджень та розробок у сфері гастрономії та штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про Angular [Електрон. ресурс]. – Режим доступу: [https://uk.wikipedia.org/wiki/Angular_\(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA\)](https://uk.wikipedia.org/wiki/Angular_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA))
2. OpenAI library installation command [Electronic resource]. – Access mode: <https://www.npmjs.com/package/openai>
3. Все про BERT [Електрон. ресурс]. – Режим доступу: <https://q-seo.com.ua/blog/absolyutno-vse-o-novom-algoritme-bert-i-chem-on-mozhet-vas-udivit/>
4. Про ChatGPT в Україні [Електрон. ресурс]. – Режим доступу: <https://www.ukrinform.ua/rubric-technology/3672046-chatgpt-epoha-novih-zagroz-ci-novih-mozlivostej.html>
5. Обмежена запитів ChatGPT [Електрон. ресурс]. – Режим доступу: <https://glavcom.ua/techno/hitech/chatgpt-stvoriv-vlasnu-movu-uzhe-spilkujetsja-z-koristuvachami.html>
6. All about ChatGPT [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/ChatGPT>
7. About ShatGPT integration with Angular [Electronic resource]. – Access mode: <https://bootcamp.uxdesign.cc/integrating-chatgpt-api-into-angular-a-3-step-guide-for-conversational-ai-development-39bee9ddef8>
8. Advantages of ChatGPT [Electronic resource]. – Access mode: <https://www.nytimes.com/2022/12/05/technology/chatgpt-ai-twitter.html>
9. About Angular Component [Electronic resource]. – Access mode: <https://angular.io/api/core/Component>
10. What is TypeScript [Electronic resource]. – Access mode: <https://www.typescriptlang.org/>

11. What is dependency injection [Electronic resource]. – Access mode:
<https://angular.io/guide/architecture-services>
12. Advantages of SCSS [Electronic resource]. – Access mode:
<https://highload.today/scss/>
13. About RxJS [Electronic resource]. – Access mode:
<https://levelup.gitconnected.com/rxjs-a-deep-dive-with-angular-8-7fe9fd675aaa>
14. About Jasmine [Electronic resource]. – Access mode:
<https://www.freecodecamp.org/news/jasmine-unit-testing-tutorial-4e757c2cbf42/>
15. API ChatGPT [Electronic resource]. – Access mode:
<https://beta.openai.com/docs/>
16. About HTTP services in Angular [Electronic resource]. – Access mode:
<https://angular.io/guide/http>
17. About RESTful [Electronic resource]. – Access mode:
<https://www.restapitutorial.com/>
18. All about security in Angular [Electronic resource]. – Access mode:
<https://angular.io/guide/security>
19. About change detection in Angular [Electronic resource]. – Access mode:
<https://angular.io/guide/change-detection>
20. What does a component in Angular consist of? [Electronic resource]. – Access mode:
<https://angular.io/guide/component-overview>
21. Testing methods in Angular [Electronic resource]. – Access mode:
<https://angular.io/guide/testing>
22. All about CTRL [Electronic resource]. – Access mode:
<https://blog.salesforceairesearch.com/introducing-a-conditional-transformer-language-model-for-controllable-generation/>
23. Disadvantages of ChatGPT [Electronic resource]. – Access mode:
<https://www.bleepingcomputer.com/news/technology/openais-new-chatgpt-bot-10-dangerous-things-its-capable-of/>

24. All about CSS [Electronic resource]. – Access mode:
<https://en.wikipedia.org/wiki/CSS>

25. Detailed CSS documentation [Electronic resource]. – Access mode:
<https://www.w3.org/Style/CSS/Overview.en.html>

26. What is artificial intelligence [Electronic resource]. – Access mode:
[https://www.ibm.com/topics/artificial-intelligence#:~:text=What%20is%20artificial%20intelligence%20\(AI\)%3F,capabilities%20of%20the%20human%20mind](https://www.ibm.com/topics/artificial-intelligence#:~:text=What%20is%20artificial%20intelligence%20(AI)%3F,capabilities%20of%20the%20human%20mind)

ДОДАТОК А
Текст програми

A.1 AppModule

```
import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { ProductListComponent } from './product-list/product-list.component';

import { ProductDialogComponent } from './product-dialog/product-dialog.component';

import { ReactiveFormsModule } from "@angular/forms";

import { HttpClientModule } from "@angular/common/http";

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
    ProductDialogComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
```

```
})
```

```
export class AppModule { }
```

A.2 AppRoutingModuleModule

```
import { NgModule } from '@angular/core';
```

```
import { RouterModule, Routes } from '@angular/router';
```

```
import { ProductListComponent } from "../product-list/product-list.component";
```

```
const routes: Routes = [{
```

```
  path: "", component: ProductListComponent
```

```
}];
```

```
@NgModule({
```

```
  imports: [RouterModule.forRoot(routes)],
```

```
  exports: [RouterModule]
```

```
})
```

```
export class AppRoutingModuleModule { }
```

A.3 style.scss

```
@import "assets/scss/buttons";
```

```
html, body, * {
```

```
  padding: 0;
```

```
  margin: 0;
```

```
    font-family: 'Roboto', sans-serif;
}
```

```
.container {
    margin: 4rem 10rem;
}
```

```
.ml-2 {
    margin-left: 1rem;
}
```

```
.mr-2 {
    margin-right: 1rem;
}
```

```
.mt-4 {
    margin-top: 2rem;
}
```

```
.mx-2 {
    margin-right: 1rem;
    margin-left: 1rem;
}
```

```
.my-2 {
    margin-top: 1rem;
    margin-bottom: 1rem;
}
```

```
.text-center {
    text-align: center;
}
```

```
.d-flex {
    display: flex;
```

```
}
```

A.4 buttons.scss

```
.button {  
  cursor: pointer;  
  padding: 1rem 2rem;  
  border-radius: 5px;  
  border: none;  
  background: transparent;  
  transition: all .2s ease-in-out;  
  
  &--primary {  
    background: #0a2942;  
    color: #d2d9e7;  
  
    &:hover {  
      color: #ffe8c4;  
      background: #005da6;  
    }  
  }  
  
  &--cancel {  
    background: #c90e21;  
    color: #e7d2d2;  
    &:hover {  
      background: #f33044;  
    }  
  }  
  
  &:disabled {  
    background: #c2c2c2;  
    color: #606060;  
    &:hover {  
      background: #e5e5e5;  
    }  
  }  
}
```

```

    }
  }

```

A.5 Product-list.component.html

```

<div class="container">

  <div *ngFor="let product of productList" class="product">
    <span>{{ product.title }}</span>
    <span>{{ product.weight }} кг</span>
    <div class="d-flex">
      <button
        [ngClass]="{'button--primary': checkProductsForRequest(product), 'button--cancel':
!checkProductsForRequest(product)}"
        class="button mr-2"
        (click)="addToRequest(product)">{{ checkProductsForRequest(product) ? 'Обрати' :
'Прибрати' }}</button>
      <button class="button action mr-2" (click)="openModal('edit', product)"><i class="fa fa-
cog"></i></button>
      <button (click)="removeFromProductList(product)" class="button action"><i class="fa fa-
trash-o"></i></button>
    </div>
  </div>

  <div class="mt-4">
    <button class="button button--primary mr-2" (click)="openModal('add')">Додати
продукти</button>
    <button class="button button--primary" [disabled]="!productsForRequest.length"
(click)="sendRequest()">Згенерувати
рецепт
  </button>
</div>

</div>

```

```

<div *ngIf="showModal">
  <app-product-dialog [form]="form" [mode]="mode" (isClosed)="closeModal($event)"></app-
product-dialog>
</div>

```

A.6 Product-list.component.scss

```

.product {
  border-radius: 5px;
  padding: 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  color: #72828f;
  background: #f9f9f9;
  margin-bottom: 1rem;

  button {
    width: 8rem;
  }

  span {
    width: 30%;
  }
}

.action {
  height: 4rem;
  color: #0a2942;
  width: 4rem !important;
  font-size: 24px;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 10rem;

```

```

    transition: all .2s ease-in-out;

    &:hover {
        background: #e0e1e5;
    }
}

```

A.7 Product-list.component.ts

```

import {Component, OnInit, TemplateRef, ViewChild} from '@angular/core';
import {FormBuilder, FormGroup, Validators} from "@angular/forms";
import {ChatGptService} from "../services/chat-gpt.service";

```

```

@Component({
    selector: 'app-product-list',
    templateUrl: './product-list.component.html',
    styleUrls: ['./product-list.component.scss']
})
export class ProductListComponent implements OnInit{
    productList = [
        {id: 1, title: 'Банан', weight: 0.1},
        {id: 2, title: 'Сметана', weight: 0.1},
        {id: 3, title: 'Огірок', weight: 0.1},
        {id: 4, title: 'Картопля', weight: 0.1},
        {id: 5, title: 'Пельмені', weight: 0.1},
        {id: 6, title: 'Хліб', weight: 0.1},
        {id: 7, title: 'Цукор', weight: 0.1},
        {id: 8, title: 'Сіль', weight: 0.1},
        {id: 9, title: 'Перець', weight: 0.1}
    ]
}

```

```
]

```

```
form!: FormGroup

```

```
showModal: boolean = false;

```

```
mode: string = "";

```

```
productsForRequest: any[] = []

```

```
@ViewChild('modal') modal!: TemplateRef<any>;

```

```
constructor(private fb: FormBuilder,
              private chatGptService: ChatGptService) {
}

```

```
ngOnInit() {
  this.initForm()
}

```

```
initForm(): void {
  this.form = this.fb.group({
    id: [null, Validators.required],
    title: [null, [Validators.required]],
    weight: [null, [Validators.required]]
  })
}

```

```

addToRequest(product: any): void {
    const index = this.productsForRequest.findIndex((p) => p.title === product.title);

    if (index !== -1) {
        this.productsForRequest.splice(index, 1);
    } else {
        this.productsForRequest.push(product);
    }
}

```

```

checkProductsForRequest(product: any): boolean {
    return !this.productsForRequest?.find((item: any) => item.title === product.title)
}

```

```

removeFromProductList(product: any): void {
    const index = this.productList.findIndex((p) => p.title === product.title);

    this.productList.splice(index, 1);
}

```

```

openModal(mode: string, product?: any) {
    mode === 'edit' ? this.form.setValue(product) : this.form.reset()

    this.mode = mode
    this.showModal = true;
}

```

```

closeModal(event: any) {

```

```

if (event && event.form) {
  const existingProductIndex = this.productList.findIndex(
    (product) => product.id === event.form.id
  );

  if (existingProductIndex !== -1) {
    this.productList[existingProductIndex] = event.form;
  } else {
    this.productList.push({
      ...event.form,
      id: this.productList[this.productList.length - 1].id + 1
    });
  }
}

this.showModal = false;
}

sendRequest(): void {
  this.chatGptService.sendToGPT("").then()

  this.productsForRequest = []
}
}

```

A.8 Product-dialog.component.html

```

<div class="modal-container">

  <p class="text-center my-2">{{modalTitle}}</p>

  <form [formGroup]="form">

    <input formControlName="title" type="text" placeholder="Назва">

    <input formControlName="weight" type="text" placeholder="Вага">


    <div class="modal__actions my-2">

      <button (click)="save()" class="button button--primary mr-2">Зберегти</button>

      <button (click)="close()" class="button button--cancel">Закрити</button>

    </div>

  </form>

</div>

<div (click)="close()" class="backdrop"></div>

```

A.9 Product-dialog.component.scss

```

.modal-container {
  padding: 2rem;
  width: 50%;
  position: fixed;
  z-index: 1;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
  overflow: auto;
}

```

```
background-color: white;

border-radius: 5px;

box-shadow: 0 5px 31px 0 rgba(0, 0, 0, 0.2);

}
```

```
.backdrop {

width: 100%;

height: 100%;

background: rgba(10,41,66, .5);

position: fixed;

left: 0;

top: 0;

z-index: 0;

overflow: hidden;

}
```

```
form {

display: flex;

flex-direction: column;

input {

height: 1rem;

padding: 1rem;

margin-bottom: 1rem;

border-radius: 5px;

border: 1px solid #b3bec7;

}

}
```

```
.modal {

  &__actions {

    display: flex;

    button {

      width: 100%;

    }

  }

}
```

A.10 Product-dialog.component.ts

```
import {Component, EventEmitter, Input, OnChanges, Output, SimpleChanges} from '@angular/core';
import {FormGroup} from "@angular/forms";
```

```
@Component({
  selector: 'app-product-dialog',
  templateUrl: './product-dialog.component.html',
  styleUrls: ['./product-dialog.component.scss']
})
export class ProductDialogComponent implements OnChanges{

  @Input() form!: FormGroup
  @Input() mode!: string
  @Output() isClosed: EventEmitter<any> = new EventEmitter<any>()

  modalTitle: string = "
```

```

ngOnChanges(changes: SimpleChanges) {
  this.mode === 'edit' ? this.modalTitle = 'Edit product' : this.modalTitle = 'Add new product'
}

```

```

close(): void {
  this.isClosed.emit()
}

```

```

save(): void {
  this.isClosed.emit({
    form: this.form.value,
    mode: this.mode
  })
}
}

```

A.11 Chat-gpt.service.ts

```

import {Injectable} from "@angular/core";
import {of} from "rxjs";
import {OpenAI} from 'openai';

@Injectable({
  providedIn: "root"
})

```

```

export class ChatGptService {

  async sendToGPT(prompt: string) {
    let configuration = new OpenAI({
      apiKey: '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',
      dangerouslyAllowBrowser: true
    });

    let requestData = {
      model: 'text-davinci-003',
      prompt,
    };

    let apiResponse = await configuration.completions.create(requestData);
    return of(apiResponse)
  }
}

```

A.12 Angular.json

```

{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "diploma": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:component": {

```

```
    "style": "scss"
  }
},
"root": "",
"sourceRoot": "src",
"prefix": "app",
"architect": {
  "build": {
    "builder": "@angular-devkit/build-angular:browser",
    "options": {
      "outputPath": "dist/diploma",
      "index": "src/index.html",
      "main": "src/main.ts",
      "polyfills": [
        "zone.js"
      ],
      "tsConfig": "tsconfig.app.json",
      "inlineStyleLanguage": "scss",
      "assets": [
        "src/favicon.ico",
        "src/assets"
      ],
      "styles": [
        "src/styles.scss"
      ],
      "scripts": []
    },
    "scripts": []
  },
}
```

```
"configurations": {  
  "production": {  
    "budgets": [  
      {  
        "type": "initial",  
        "maximumWarning": "500kb",  
        "maximumError": "1mb"  
      },  
      {  
        "type": "anyComponentStyle",  
        "maximumWarning": "2kb",  
        "maximumError": "4kb"  
      }  
    ],  
    "outputHashing": "all"  
  },  
  "development": {  
    "buildOptimizer": false,  
    "optimization": false,  
    "vendorChunk": true,  
    "extractLicenses": false,  
    "sourceMap": true,  
    "namedChunks": true  
  }  
},  
"defaultConfiguration": "production"  
},
```

```

"serve": {
  "builder": "@angular-devkit/build-angular:dev-server",
  "configurations": {
    "production": {
      "browserTarget": "diploma:build:production"
    },
    "development": {
      "browserTarget": "diploma:build:development"
    }
  },
  "defaultConfiguration": "development"
},
"extract-i18n": {
  "builder": "@angular-devkit/build-angular:extract-i18n",
  "options": {
    "browserTarget": "diploma:build"
  }
},
"test": {
  "builder": "@angular-devkit/build-angular:karma",
  "options": {
    "polyfills": [
      "zone.js",
      "zone.js/testing"
    ],
    "tsConfig": "tsconfig.spec.json",
    "inlineStyleLanguage": "scss",

```

```
"assets":  
  "src/favicon.ico",  
  "src/assets"  
],  
"styles": [  
  "src/styles.scss"  
],  
"scripts": []  
}  
}  
}  
}  
},  
"cli": {  
  "analytics": "6c139923-f2d7-4c79-ab5c-1c5b1786afe3"  
}  
}
```

Додаток Б
Слайди презентації

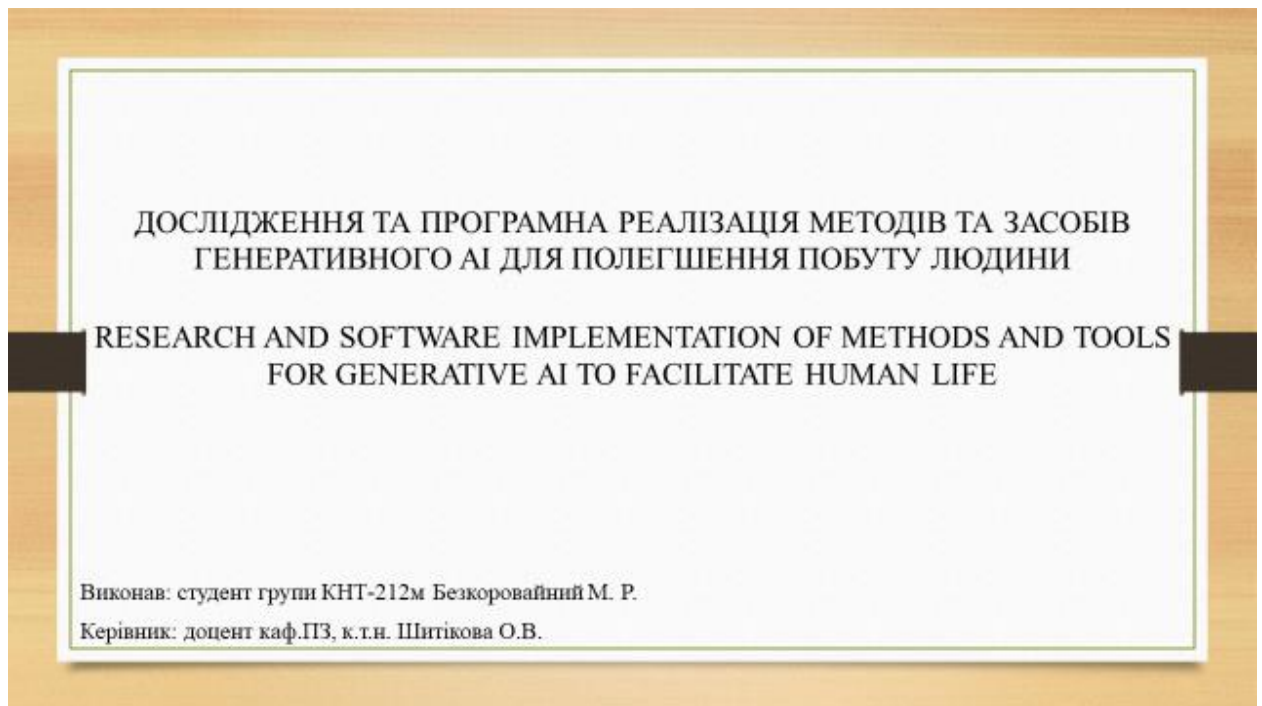


Рисунок Б.1 – Слайд № 1

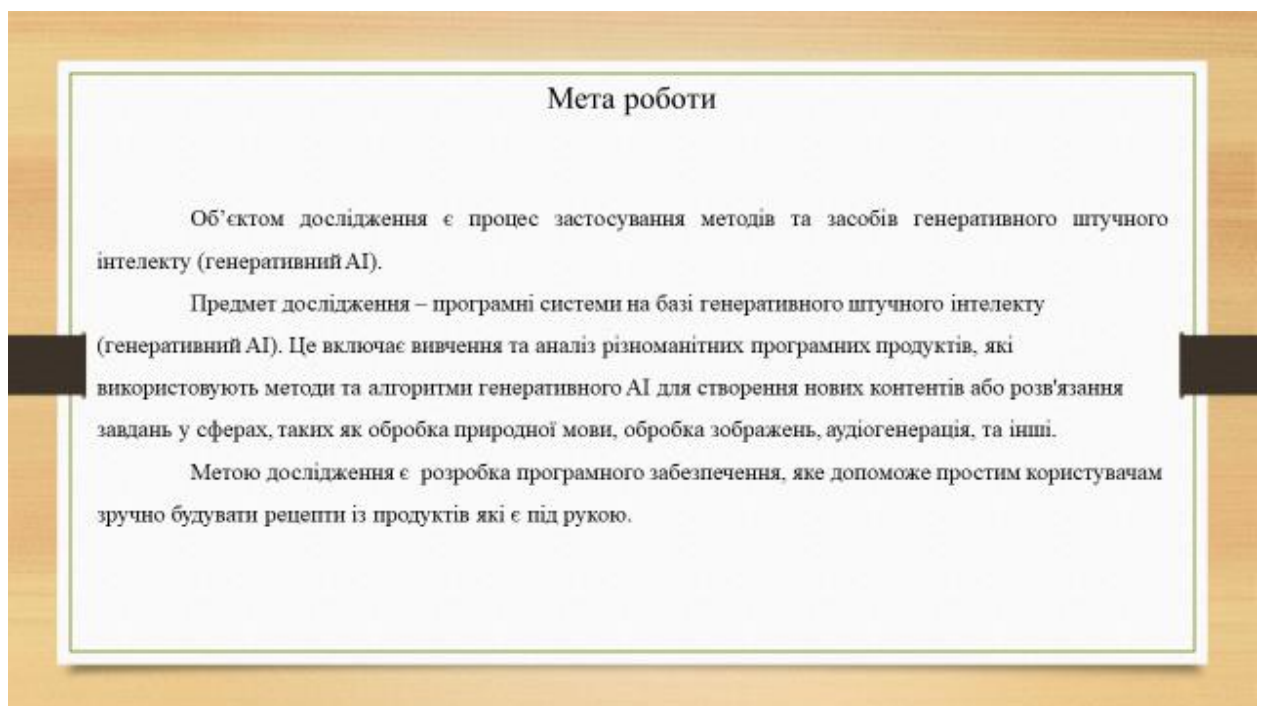


Рисунок Б.2 – Слайд № 2

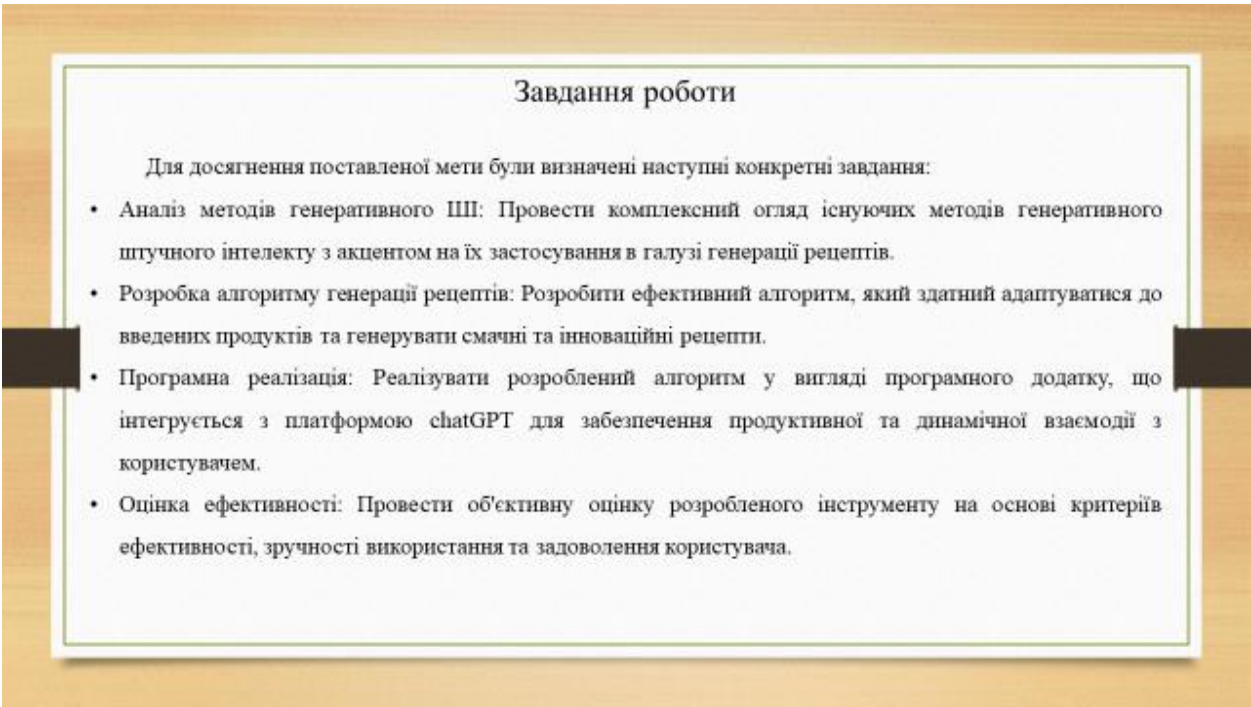


Рисунок Б.3 – Слайд № 3

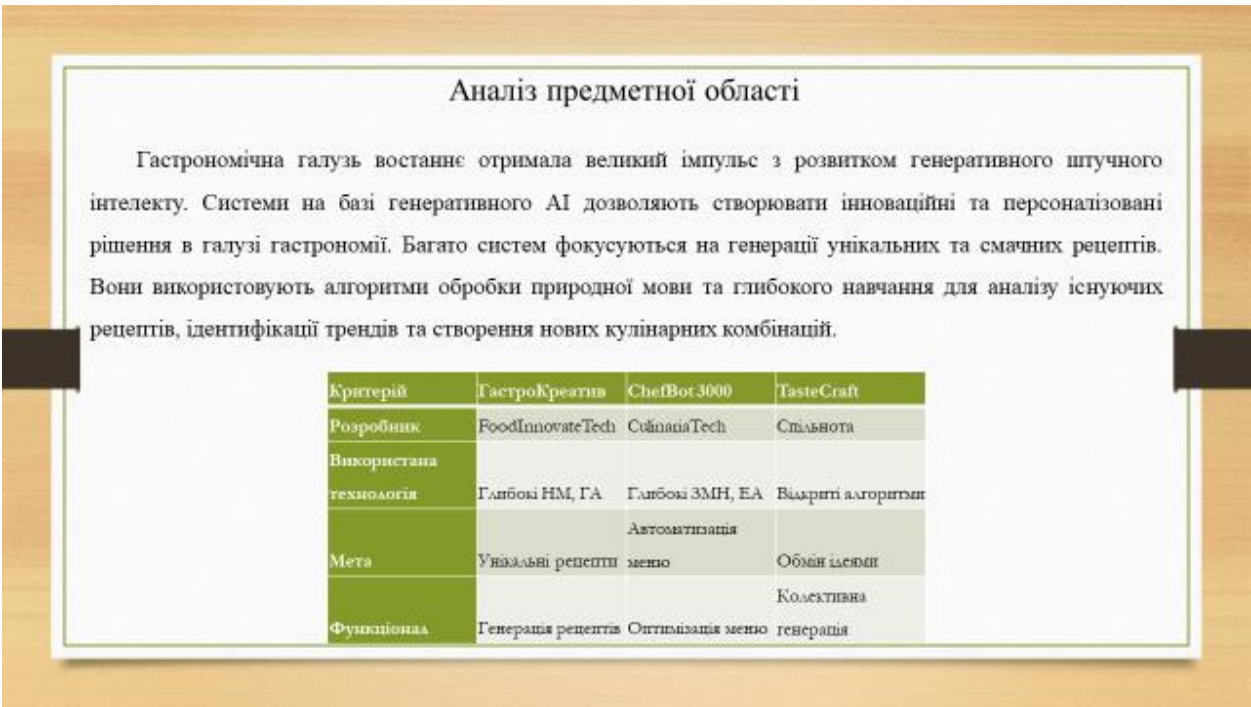


Рисунок Б.4 – Слайд № 4

Функціональні вимоги до розроблюваної системи

- Створення продукту (назва і вага)
- Редагування продукту
- Видалення продукту
- Перегляд своїх продуктів
- Можливість вибору певних продуктів для створення рецепта
- Можливість генерації рецептів на основі обраних продуктів

Перегляд готового рецепта який згенерував AI

Рисунок Б.5 – Слайд № 5

Інструменти розробки

Використання OpenAI було використано через значні переваги серед конкурентів

Критерій	React	Flutter	ReactNative	Swift
Примічання	Розуміння природної мови, NLU	Розуміння природної мови, генерація текстів, малювання	Розробка чат-бота та віртуальних асистентів	Генерація текстів, контрастне написання
Архітектура	Трансформатор	Трансформатор	Різні технології під капотом для обробки природної мови	Трансформатор
Використання в галузях	Пошукові системи, відповіді на питання	Пошукові системи, відповіді на питання, Генерація текстів, варіанти контекстів	Чат-боти, віртуальні асистенти	Генерація текстів, варіанти контекстів
Написання	Масштабне написання, великі корупції текстів	Масштабне написання, великі корупції текстів, Контрастне написання, генерація текстів	Велика кількість текстів та голосових даних	Контрастне написання, генерація текстів
Висновки	Великі витрати обчислювальних ресурсів	Мас обчислюв. базу даних, немає доступу до мережі інтернет	Залежить від обсягу використання та обчислювальних потреб	Великі витрати обчислювальних ресурсів

Використання Angular було використано через typescript, більшу захищеність та зручність написання.

Критерій	Angular	Next	Vue
Мова опису	Typescript	JavaScript	JavaScript
Архітектура	Компонентно-орієнтована, MVC (Model-View-Controller)	Компонентно-орієнтована, Virtual DOM	Компонентно-орієнтована, реактивні елементи
Оптимізація та продуктивність	Докорисливе використання даних, складні стратегії використання мови	Віртуальний DOM для ефективного оновлення інтерфейсу	Реактивні елементи, віртуальний DOM для оптимізації оновлень
Гнучкість та простота використання	Великий набір інструментів та функціональності, але з більш високим порогом входу	Простий API, але для деяких завдань може знадобитися використання додаткових бібліотек	Легко вивчити, мінімальна конфігурація, гнучкість в використанні

Рисунок Б.6 – Слайд № 6

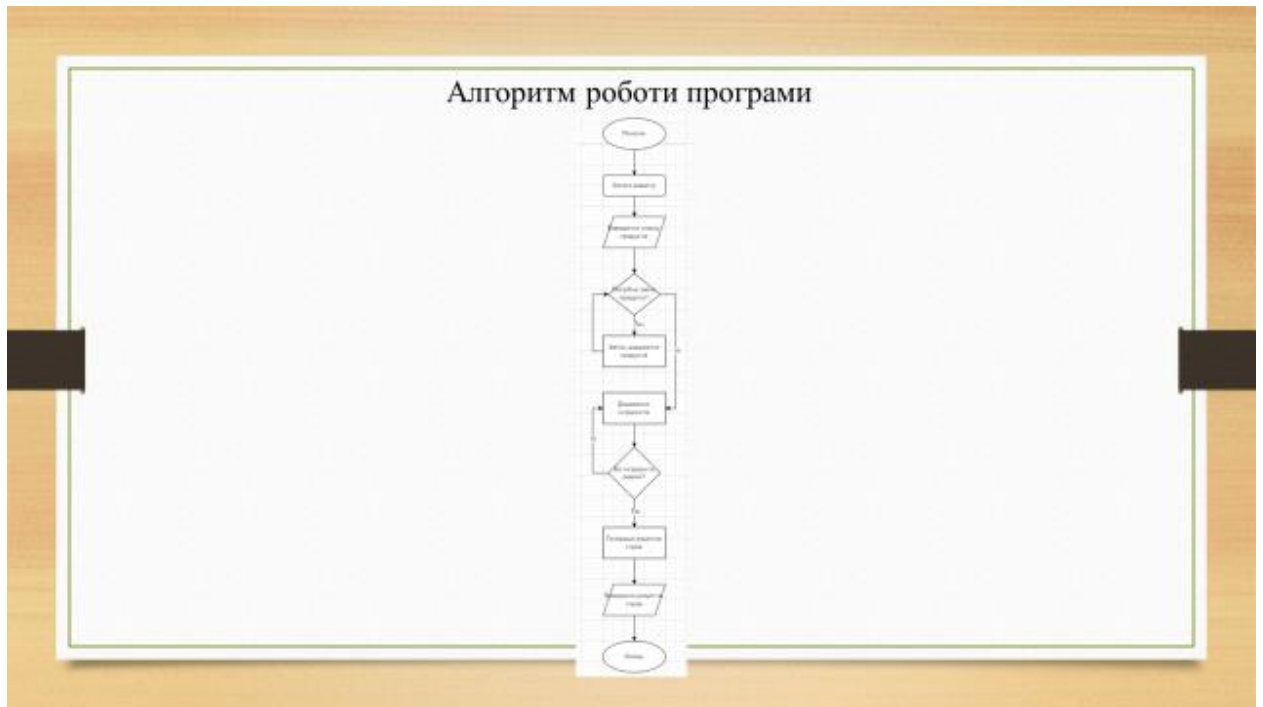


Рисунок Б.7 – Слайд № 7

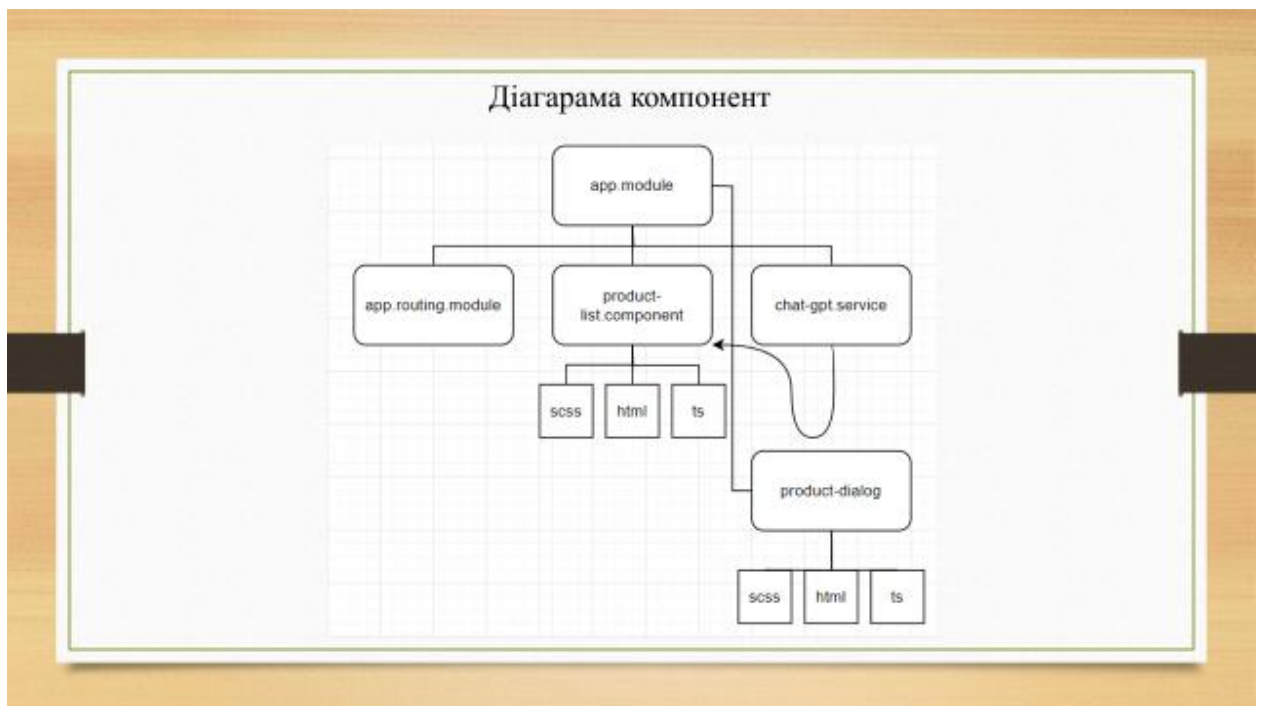


Рисунок Б.8 – Слайд № 8

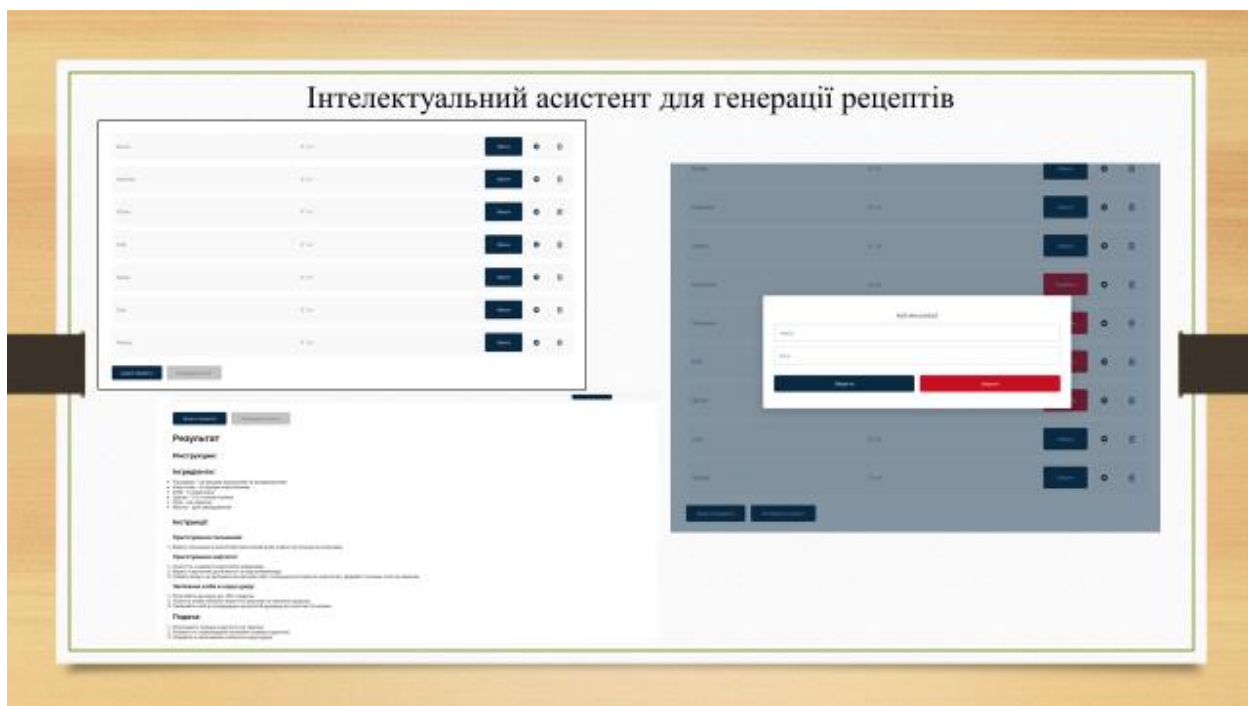


Рисунок Б.9– Слайд № 9

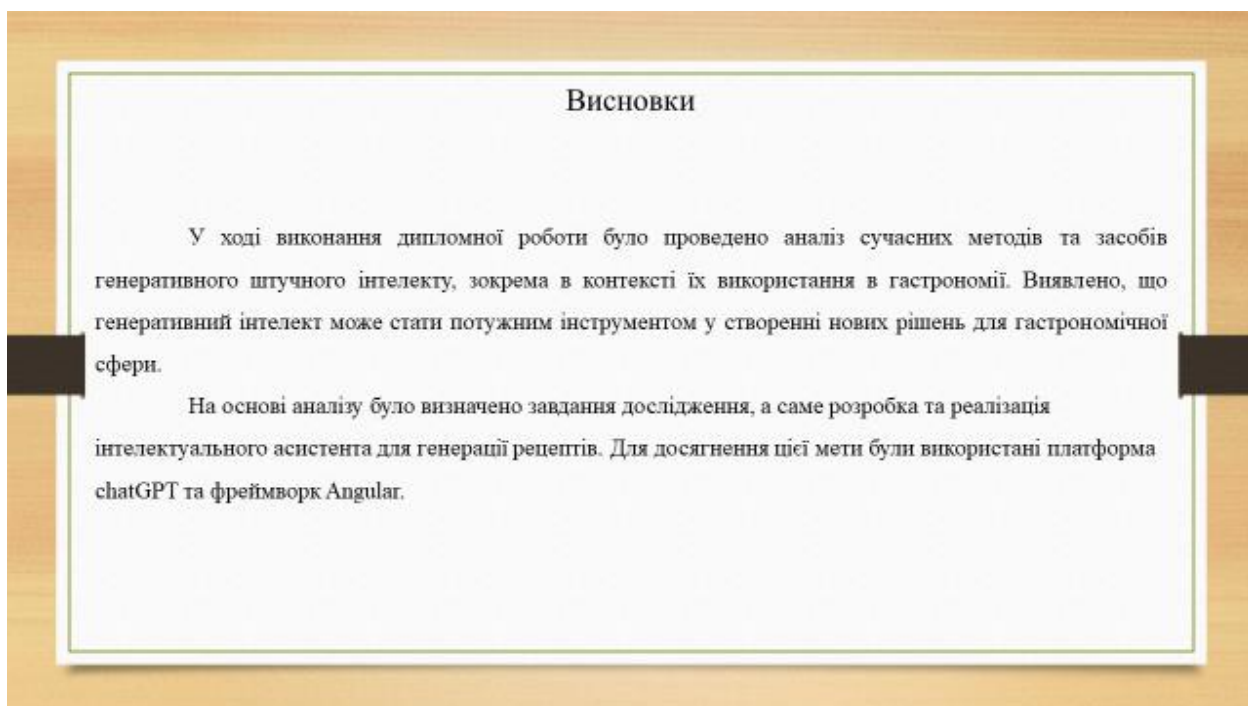


Рисунок Б.10– Слайд № 10