Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти.

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Сторчак В.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Ясній О.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Сторчак Василь Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка сервера "Щоденник харчування" на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти.

Керівник роботи Ясній Олег Петрович, доктор технічних наук, професор кафедри МН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-470*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Наукові та інформаційні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання. 1.1 Аналітичний огляд існуючих рішень. 1.2 Технічне завдання. 1.3 Найменування та область застосування. 1.4 Призначення розробки. 1.5 Вимоги до програмного забезпечення. 1.6 Вимоги до програмної документації. 1.7 Техніко-економічні показники. 1.8 Вимоги до середовища виконання. 1.9 Висновок до першого розділу. 2. Проектна частина. 2.1 Проєктування структури програми. 2.2 Проєктування структури серверної частини програми. 2.3 Проєктування структури бази даних програми. 2.4 Проєктування структури моделі для виявлення об'єктів. 2.5 Проєктування структури моделі для прогнозу калорій. 2.6 Проєктування програмного модуля для складання раціону. 2.7 Висновок до другого розділу. 3. Практична частина. 3.1 Розробка серверної частини програми. 3.2 Створення моделі для виявлення об'єктів. 3.3 Створення моделі для прогнозу калорій. 3.4 Створення програмного модуля для складання рекомендацій раціону. 3.5 Тестування серверної частини. 3.6 Тестування моделі для виявлення об'єктів. 3.7 Тестування моделі для прогнозу калорій. 3.8 Тестування програмного модуля для складання раціону. 3.9 Висновок до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титулка. 2 Актуальність. 3 Мета, задачі дослідження. 4 Використання GitHub у навчальному процесі. 5 Актори та функціональні вимоги до системи. 6 Модель предметної області. 7 Типи проектів. 8 Вибір інструментів створення системи. 9 Про серверну частину. 10 Про базу даних. 11 Модуль ШІ для виявлення об'єктів. 12 Модуль ШІ для прогнозу калорій. 13 Модуль для створення рекомендацій раціону. 14 Приклад використання. 15 Результати дослідження.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин Віктор Степанович		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про створення штучного інтелекту та	31.01.2024-03.02.2024	*Виконано*
	розробку серверів		
4.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
5.	Виконання дослідження, що до створення штучного	07.02.2024-11.02.2024	*Виконано*
	інтелекту		
6.	Виконання дослідження, що до створення датасетів	11.02.2024-17.02.2024	*Виконано*
7.	Виконання дослідження, що до створення серверів	17.02.2024-30.02.2024	*Виконано*
8.	Виконання дослідження, що до створення баз даних	30.02.2024-10.03.2024	*Виконано*
9.	Виконання дослідження, що до тестування сервера	10.03.2024-30.03.2024	*Виконано*
10.	Оформлення розділу «Аналіз предметної області	03.06.2024-05.06.2024	*Виконано*
	та постановка завдання»		
11.	Оформлення розділу «Проектна частина»	06.06.2024-08.06.2024	*Виконано*
12.	Оформлення розділу «Практична частина»	09.06.2024-11.06.2024	*Виконано*
13.	Виконання завдання до підрозділу «Безпека	12.06.2024-13.06.2024	*Виконано*
	Життєдіяльності»		
14.	Виконання завдання до підрозділу «Основи охорони		
	праці»		
15.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	*Виконано*
16.	Нормоконтроль	18.06.2024-19.06.2024	*Виконано*
17.	Перевірка на плагіат	20.06.2024	*Виконано*
18.	Попередній захист кваліфікаційної роботи	21.06.2024	*Виконано*
19.	Захист кваліфікаційної роботи	30.06.2024	

Студент _____ Сторчак В.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ясній О.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти // Кваліфікаційна робота освітнього рівня «Бакалавр» // Сторчак Василь Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНс-42 // Тернопіль, 2024 // С. 64 , рис. – 11, табл. – 2, кресл. – 0, додат. – 6, бібліогр. – 35.

Ключові слова: штучний інтелект, Python, сервер, щоденник їжі, база даних, REST API, дієта

Кваліфікаційна робота присвячена дослідженню на тему "Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти" охоплює процес створення системи, здатної аналізувати харчові звички користувачів та надавати персоналізовані рекомендації. У першому розділі проведено детальний аналіз існуючих рішень та технологій, що використовуються в галузі дієтології та штучного інтелекту. Другий розділ присвячений проєктуванню архітектури сервера, включаючи моделі баз даних, алгоритми машинного навчання та інтерфейс REST API. В третьому розділі описано реалізацію розробленого серверного рішення, включаючи інтеграцію з базами даних та налаштування алгоритмів штучного інтелекту. Проведено тестування системи на реальних даних, щоб перевірити точність та ефективність рекомендацій. Отримані результати демонструють потенціал використання даного сервера для покращення харчових звичок користувачів.

ANNOTATION

Development of the "Diary of food" server based on artificial intelligence and REST API for the development of recommendations for creating an individual diet. // Qualification work of the educational level "Bachelor" // Storchak Vasyl // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-42 // Ternopil, 2024 // P. 64, fig. - 11, tabl. - 2, chair. - 0, annexes. – 6, references - 35.

Keywords: artificial intelligence, python, server, food diary, database, REST API, diet.

The qualification work is devoted to the research on the topic "Development of the "Diary of food" server based on artificial intelligence and REST API for it to make recommendations for making individual diets" covers the process of creating a system capable of analyzing the eating habits of users and providing personalized recommendations. The first chapter provides a detailed analysis of existing solutions and technologies used in the field of dietetics and artificial intelligence. The second chapter is devoted to the design of the server architecture, including database models, machine learning algorithms, and the REST API interface. The third chapter describes the implementation of the developed server solution, including integration with databases and setting up artificial intelligence algorithms. The system was tested on real data to verify the accuracy and effectiveness of the recommendations. The obtained results demonstrate the potential of using this server to improve the eating habits of users.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – Штучний інтелект.

БД – База даних

OD (англ. Object Detection) — виявлення об'єкта

GPU (англ. Graphics Process Unit) — графічний обчислювальний компонент

YOLOv5 (англ. You Only Look Once version 5) — ви дивитися тільки один раз версія 5

CRUD (англ. Create Read Update Delete) — створення читання оновлення видалення

БЖД - безпеку життєдіяльності

ПЗ – програмне забезпечення

ТЗ – технічне завдання

ЗМІСТ

ВСТУП

Останні десятиліття відзначилися стрімким розвитком технологій штучного інтелекту через розвиток обчислювальних технологій, таких як сучасні відеокарти від компанії Nvidia, що в свою чергу дозволяє використовувати сучасні архітектури нейронних мереж, таких як революційна архітектура Transformers, що в свою чергу стало одним з найважливіших напрямків сучасних наукових досліджень та інновацій.

ШІ проник у різні сфери життя, від медицини й фінансів до транспорту і освіти, що свідчить про його універсальні можливості і необхідність дослідження вивчення.

Актуальність досліджень у галузі ШІ обумовлена його потенціалом автоматизувати часто повторювальних й подібних завдань з різною важкістю, покращуючи продуктивність та ефективність у різних галузях.

В умовах глобалізації та цифровізації суспільства попит на технології ШІ зростає, сприяючи економічному розвитку та соціальним змінам. Впровадження ШІ у виробничі процеси дозволяє підвищити точність, швидкість і якість виконання завдань, а головне зробить більш дешевим виконання роботи, що має позитивний вплив на конкурентоспроможність підприємств, які масово будуть впроваджувати дану технологію.

Системи ШІ здатні аналізувати великі обсяги даних та виявляти складні закономірності, що робить їх незамінними в дослідженнях і розробках нових продуктів, послуг чи сервісів.

Інтернет сервіси приносять користь людям, надаючи зручний доступ до інформації, спілкування та покупок, а для розробників відкривають нові можливості створення інноваційних рішень та розвитку професійних навичок.

Розвиток ШІ відкриває нові можливості для інклюзивної освіти, дозволяючи створювати персоналізовані навчальні програми, які відповідають індивідуальним потребам людини в сфері планування здорового раціону. Етичні аспекти використання ШІ викликають значні дискусії, адже важливо

забезпечити, щоб розвиток цих технологій сприяв загальному благу, а не створював загрози для людства.

ШІ є ключовим елементом у розробці нових підходів до охорони здоров'я, включаючи діагностику, лікування та прогнозування захворювань, що значно підвищує ефективність медичної допомоги. З огляду на все вищезазначене, дослідження та розробка технологій ШІ є критично важливими для забезпечення сталого розвитку суспільства в умовах сучасних викликів та можливостей.

Кінцевою метою даного дослідження є розробка та впровадження системи штучного інтелекту для автоматичного розпізнавання їжі на фотографіях та ведення щоденника харчування. Використання таких систем допоможе користувачам легше стежити за своїм раціоном, забезпечуючи точне відстеження спожитих продуктів та калорій. Основним завданням є створення алгоритму розпізнавання їжі, який буде здатен ідентифікувати різноманітні страви та інгредієнти з високою точністю.

Для досягнення цієї мети необхідно зібрати та обробити великий обсяг навчальних даних, що включатимуть зображення їжі з різних джерел. Наступним завданням є розробка програмного інтерфейсу користувача, що дозволить легко завантажувати фотографії та отримувати детальну інформацію про склад страви.

Одним із завдань дослідження є оцінка точності та ефективності розробленої системи через серію тестувань з реальними користувачами.

Практичне значення одержаних результатів: в кінцевому результаті, дана система має на меті не лише спростити ведення щоденника харчування, але й сприяти здоровому способу життя та підвищенню обізнаності користувачів про їх харчові звички.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналітичний огляд існуючих рішень

Розпочинаючи будь-який проект чи роботу, важливо скласти перелік усіх можливих варіантів, завдань та ресурсів, що можуть сприяти його реалізації. У цьому проекті, який передбачає створення програмного продукту, необхідно володіти специфічними уміннями, навичками та досвідом роботи з програмним забезпеченням і його внутрішніми компонентами. Крім того, використання сучасних технологій, таких як штучний інтелект, може значно підвищити ефективність, дозволить додавати новий функціонал та якість розробки ПЗ, автоматизуючи рутинні завдання, оптимізуючи процеси та покращуючи кінцевий програмний продукт.

Аналітичний огляд існуючих рішень на тему "Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти" включає розгляд різноманітних підходів та інструментів, що використовуються для створення таких серверів. Одним з найпоширеніших рішень є використання фреймворку Flask чи FastAPI[1], який забезпечує простоту та гнучкість у розробці REST API[2]. Flask дозволяє легко інтегрувати моделі штучного інтелекту, зокрема ті, що розроблені з використанням бібліотек Tensorflow[3] або PyTorch[4]. Крім того, фреймворк FastAPI набирає популярність завдяки своїй високій продуктивності та можливостям автоматичного створення документації API. Для реалізації складних AI моделей також часто використовується Django, який забезпечує надійну інфраструктуру для масштабованих рішень. У випадку необхідності обробки великих обсягів даних та високої швидкості відповіді, застосовуються асинхронні бібліотеки, такі як aiohttp. Також варто звернути увагу на популярні сервіси, на кшталт Google Cloud AI та AWS AI Services, що пропонують інтегровані рішення для розгортання AI моделей через REST API. Огляд сучасних рішень демонструє широкий спектр можливостей для

розробників, що дозволяє обирати оптимальний підхід залежно від конкретних вимог продукту, також демонструє велику гнучкість мови програмування Python для роботи з нейронними мережами, від тренування до розгортання на якомусь пристрої для роботи з користувачами.

1.2 Технічне завдання

Технічне завдання (ТЗ) є офіційним документом, затвердженим усіма зацікавленими сторонами, в даному випадку між студентом і керівником дипломної роботи, документ який слугує основою для виконання розробки. У цьому документі надзвичайно точно та детально викладено вимоги замовника щодо компонентів та характеристик майбутнього програмного продукту.

Також, завдяки технічному завданню, замовник має можливість оцінити роботу виконавця, використовуючи критерії відповідності кінцевого програмного продукту вимогам, описаним у технічному завданні. Основна мета технічного завдання полягає у забезпеченні взаєморозуміння між виконавцем та замовником, з урахуванням усіх необхідних критеріїв, вартості робіт, етапів виконання та термінів здачі кінцевого продукту. В даному випадку замовником слугує приймальна комісія, яка буде оцінювати знання та навички які були набуті під час навчання у ВНЗ методом оцінки дипломної роботи бакалавра.

Тема даного дипломного проекту – це “Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти”.

Програма буде розроблена для інтеграції та координації роботи двох моделей штучного інтелекту, модуля створення рекомендацій раціону, серверу та бази даних. Перша модель ШІ відповідає за обробку зображень і виконує завдання OD (Object Detection), визначаючи об'єкти на фото. Друга модель спеціалізується на передбаченні кількості калорій у визначених об'єктах на основі отриманих зображень. Модуль створення рекомендацій повинен створити відповідно рекомендацію в раціону базуючись на спожитій їжі. Сервер

забезпечує безперебійну взаємодію між компонентами системи, обробляючи запити і керуючи мережею. База даних зберігає всі необхідні дані, включаючи історію користувацьких запитів та результати роботи моделей ШІ. Завдяки цій комплексній системі, програма здатна швидко і ефективно обробляти великі обсяги даних, надаючи користувачам точні і релевантні результати.

Перш за все, мета дипломного проєкту – це саме набути практичних навичок та вмінь по:

– проєктуванні;
– плануванні;
– розробці;
– тестуванні;
– випуску програмного продукту.

Проєктування. На кшталт проєктування комп’ютерних додатків, серверів, інтернет-сервісів, системних програм, сайтів та інших цифрових проєктів – це один з ключових етапів розробки будь-якого завдання, що вимагає розумової активності та поетапності, а особливо це стосується цифрового програмного продукту. На цьому етапі життєвого циклу продукту можна візуально уявити і, при необхідності, скоригувати всі вимоги технічного завдання під реалії розробки, або скоригувати програмний продукт під технічне завдання. Як і в будівництві, проєктування програмного забезпечення вимагає створити схему архітектури продукту згідно технічного завдання замовника. Тому й фахівців з сфери проєктування програмного забезпечення часто називають і мають відповідні робочі міста, тобто називаються - програмними або системними архітекторами. Важливою складовою проєктування програмного забезпечення є прототипування. Прототипування трансформує формалізовані та неформалізовані побажання клієнта чи власні побажання до продукту, що виражені у вигляді текстових описів, фото, діаграм, записок, приміток, нотаток і ілюстрацій, у добре продуманий і візуально зрозумілий прототип програмного продукту, який можна буде аналізувати.

Планування спрямоване на розвиток внутрішнього відчуття часу, що дозволяє ефективно розподіляти часові ресурси під час створення продукту. Це забезпечує систематичний підхід до виконання завдань і допомагає уникнути затримок. Важливим аспектом планування є вміння прогнозувати часові витрати на різні етапи проєкту. Крім того, чітке планування сприяє кращій координації команди та підвищенню продуктивності роботи.

Розробка, як етап, назва даного етапу говорить сама за себе – це процес створення програмного забезпечення. Основною метою цього етапу є освоєння навичок написання складних програм у форматі вихідного коду, на прикладі даного проєкту: вихідного коду на мові програмування Python. Важливо не лише створювати функціональні програми, але й забезпечувати їхню оптимальність і ефективність. Опанування цих навичок дозволяє розробникам реалізовувати проєкти будь-якої складності, забезпечуючи їхню стабільну роботу. Крім того, цей процес вимагає постійного вдосконалення технічних знань та вміння працювати в команді.

Тестування є безсумнівно критичним етапом життєвого циклу розробки, оскільки дозволяє виявити всі потенційні проблеми продукту як під час його створення, так і перед запуском. Цей процес допомагає забезпечити високу якість програмного забезпечення та уникнути можливих збоїв. Ретельне тестування гарантує, що кінцевий продукт буде надійним і відповідатиме очікуванням користувачів.

Етап випуск програмного продукту включає в себе офіційне впровадження розробленого програмного забезпечення для кінцевих користувачів. На цьому етапі відбувається розгортання системи, її налаштування і можливе проведення навчання користувачів.

Цей програмний продукт має використовувати мережу Інтернет для своєї роботи незалежно від платформи пристрою, на якому він виконується, що гарантує його платформонезалежність. Програма повинна забезпечувати якісне виконання запитів до штучного інтелекту, сервера та бази даних. Важливо, щоб запити оброблялися швидко і точно, що дозволить користувачам отримувати

необхідну інформацію без затримок. Продукт має мати оптимізований алгоритм для роботи з великими обсягами даних, забезпечуючи надійність і безперервність функціонування. Також програмний продукт має використовувати можливості відеокарти для прискорення обробки графічних та обчислювальних задач, що покращить загальну продуктивність і ефективність роботи програми.

Розробка даного програмного продукту для дипломного проєкту на тему “Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти” буде виконувати студент 4-го курсу, Сторчак Василь Юрійович за індивідуальним завданням від керівника дипломного проєкту Ясній Олег Петрович.

1.3 Найменування та область застосування

Кожна створена або майбутня програма має свою унікальну назву та сферу застосування. Дана програма на тему “ Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти ” має найменування “FoodNoteAI”, це відображає суть програми та її цільову аудиторію.

Ця програма-сервер використовує штучний інтелект для детектування їжі на фотографіях і визначення її калорійності. Користувачі можуть завантажувати зображення їжі, а сервер обробляє їх, аналізуючи вміст за допомогою алгоритмів комп'ютерного зору. Після аналізу система надає інформацію про калорійність виявлених продуктів. Вся отримана інформація зберігається в базі даних для подальшого використання та аналізу. Це дозволяє користувачам відстежувати споживання калорій і поліпшувати свої харчові звички.

1.4 Призначення розробки

Кожна дія в нашому житті має свій сенс чи мету, так само і кожна програма створюється з певною метою.

Основною метою створення цього програмного продукту є виконання завдання, поставленого керівником дипломного проекту, що необхідно для здобуття диплому молодшого спеціаліста, який є кінцевою ціллю навчання в закладі освіти.

Ця програма-сервер може допомогти користувачам точніше відстежувати споживання калорій, покращуючи їхні харчові звички. Вона зменшує час і зусилля, необхідні для підрахунку калорій вручну, завдяки автоматичному аналізу фотографій їжі. Крім того, програма забезпечує зберігання та аналіз даних про харчування, що може бути корисним для дієтологів і медичних фахівців у наданні рекомендацій.

Дана програма може допомогти автору – тим, що допоможе завершити навчання в закладі вищої освіти, а одже підвищити рівень кваліфікації та збільшити свою конкурентоспроможність серед інших працівників у даній сфері.

1.5 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення – це сукупність загальних вимог до властивостей, якості, функціональних і нефункціональних характеристик програмного забезпечення, що розробляється або вже знаходиться в процесі розробки. Зазвичай, вимоги до програмного забезпечення формує замовник у вигляді технічного завдання, і цей розділ є невід'ємною частиною технічного завдання програмного продукту. Вони визначаються під час аналізу вимог і документуються у вигляді специфікації вимог, метрики моделей, необхідної інфраструктури, швидкості окупності та інших елементів.

Розробка технічних умов, що до програмного продукту має бути розподілена певні етапи, таких як:

– пошук вимог;

– аналіз вимог;

– специфікація;

– перевірка вимог.

Типи вимог за характером можна поділити, а саме на:

– вимоги до поведінки бізнес системи та вимоги відносно цільового користувача;

– вимоги до очікуваної поведінки системи та наданих бізнес-правила які описують обмеження, які є дотичними до предметної області, системні вимоги, тобто вимоги до програмного інтерфейсу, відповідних характеристик, надійності та безпеки даних, характеристика якості результату.

Для програмної реалізації потрібно скласти перелік кінцевих вимог до програмного забезпечення. Із переліком вимог можна детальніше ознайомитися нижче по змісту:

– відповідність технічному завданню;

– відповідна швидкість роботи з використанням GPU[5];

– документація;

– безпека даних[6];

– стабільна робота.

Дані пункти відповідають ТЗ дипломної роботи.

1.6 Вимоги до програмної документації

Технічна документація являє собою систему графічних та текстових документів, які є необхідними та достатніми для повного і безперешкодного використання на всіх етапах життєвого циклу продукції.

Існують різні типи технічної документації, до так званої технічної документації можна віднести:

– конструкторська документація, що в свою чергу відповідає за конструкторські документів, які мають загальні дані, згідно якими можна продовжувати розробку, постачання, контроль, підтримку та інше;

– технічна документація, що в свою чергу являє перелік структурованих документів які мають визначення технологічного процесу виготовлення та покращення продукту;

– програмна документація, що являє собою в собі сукупність документів, які необхідно для розробки, підтримки, супроводу та підтримки цифрового продукту.

Даний дипломний проєкт вимагає наявності програмної документації. При цьому, під час отримання завдання не було встановлено жодних специфічних або особливих вимог до технічної документації. Вона має включати стандартний набір документів, а саме:

– інструкція інсталяції;
– вимоги до апаратної підтримки;
– інструкція використання;
– описана структура програми.

Даний список вимог до документації цілком відповідають завданню з документації програмного продукту.

1.7 Техніко-економічні показники

Техніко-економічні показники програми серверу з ШІ є критично важливими для оцінки її ефективності та вартості. Основним показником є продуктивність, яка визначається швидкістю обробки даних і здатністю обслуговувати велику кількість користувачів одночасно. Енергоспоживання програми є ще одним важливим аспектом, оскільки ефективне використання енергії може значно знизити експлуатаційні витрати. Масштабованість програми дозволяє їй адаптуватися до зростаючих потреб, забезпечуючи можливість розширення функціональності без значних додаткових витрат. Зрештою,

технічна підтримка та обслуговування також є важливими показниками, оскільки вони забезпечують безперервну роботу системи та своєчасне усунення можливих проблем.

Ресурси які були використані для розробки дипломного проєкту наведенні нижче:

– відповідне обладнання для розробки;
– необхідне програмне забезпечення;
– необхідні знання;
– відповідна кількість часу.

Для виконання програмної реалізації буде використано орієнтовно приблизно від ста годин роботи. Відповідно для підрахунку техніко-економічний показників необхідно мати визначення термінів:

– собівартість реалізованої продукції;
– чистий прибуток;
– час окупності.

Собівартість є одним з ключових економічних показників продукції будь-якого підприємства. У процесі розрахунку собівартості можуть брати участь не лише бухгалтери, а й інші зацікавлені особи.

Чистий прибуток є кінцевим фінансовим результатом діяльності підприємства після вирахування всіх витрат, включаючи податки і збори. Він розраховується як різниця між загальним доходом і сумою всіх витрат. Його значення використовується для оцінки рентабельності, планування інвестицій та прийняття управлінських рішень.

Актуальність цього дипломного проекту обумовлена високим попитом на ресурси, пов'язані з даною тематикою, що надає йому потенційні шанси на успіх. Наявність такого попиту дозволяє припустити, що проект може бути економічно вигідним. Таким чином, є підстави очікувати, що інвестиції в цей програмний продукт будуть виправданими та принесуть прибуток.

1.8 Вимоги до середовища виконання

Середовище виконання серверу з штучним інтелектом на основі PyTorch забезпечує високопродуктивну платформу для розгортання та обслуговування моделей глибокого навчання[7]. Використання PyTorch на сервері забезпечує зручні інструменти для відлагодження та моніторингу моделей. Безпека і стабільність середовища виконання досягається через регулярні оновлення та патчі. Висока гнучкість і ефективність PyTorch роблять його оптимальним вибором для розробки та впровадження AI-рішень на серверному рівні. Тобто в даному випадку можна створити вимоги до середовища виконання, а саме:

– операційна система Linux[8] з Python[9] інтерпретатором;

– відповідне обладнання у вигляді відеокарти з підтримкою OpenCL чи CUDA.

Дані вимоги цілком відповідають поставленому завдання з розробки та роботи програмного продукту.

1.9 Висновок до першого розділу

В першому розділі кваліфікаційної роботи було досліджено актуальність розробки серверу для ведення щоденника харчування з використанням штучного інтелекту. Визначено основні вимоги до системи, зокрема точність розпізнавання продуктів, зручність використання та забезпечення конфіденційності даних. На основі цього сформульовано основні завдання проекту, включаючи розробку алгоритмів обробки даних та інтеграцію з базами харчових продуктів. Отже, запропонований проект має потенціал для задоволення потреб користувачів та заповнення ніші на ринку цифрових рішень для ведення харчового щоденника.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА

2.1 Проєктування структури програми

Даний абзац дипломного проєкту, присвячений вибору мови програмування Python для написання сервера та роботи з ІШ для комп'ютерного зору та БД, є ключовим елементом технічного опису дослідження.

Python обрано через його універсальність, ефективність та велику спільноту розробників, що забезпечує доступ до багатої екосистеми бібліотек та інструментів. Мова програмування Python ідеально підходить для написання серверної частини проєкту через свою простоту та високий рівень абстракції, що дозволяє розробникам швидко реалізовувати функціонал та ефективно керувати ресурсами сервера.

Крім того, наявність різноманітних фреймворків, таких як FastAPI, сприяє швидкому розгортанню серверних додатків та забезпечує стабільність роботи системи.

Щодо роботи з штучним інтелектом для комп'ютерного зору, Python є популярним вибором завдяки багатству спеціалізованих бібліотек, таких як OpenCV[10], TensorFlow та PyTorch, які надають потужні інструменти для обробки зображень, навчання моделей та інтеграції із системами комп'ютерного зору.

Управління базою даних також здійснюється з використанням Python завдяки широкому вибору об'єктно-реляційній моделі, наприклад таких як SQLAlchemy, що спрощує взаємодію з реляційними базами даних і дозволяє легко виконувати операції з даними.

Використання Python в проєкті забезпечує його високу продуктивність, зручність у розробці та підтримці, а також забезпечує інтеграцію з іншими технологіями, що робить його ідеальним вибором для комплексного розробки серверної частини, штучного інтелекту та роботи з базою даних у даному дипломному проєкті.

Для продовження розробки потрібно дати відповіді на наступні питання та етапи проектної частини, а саме на:

– архітектурний підхід;
– тип структури;
– обробка запитів;
– безпека та автентифікація;
– моніторинг та логування;
– інтеграція з іншими сервісами;
– масштабування.

Архітектурний підхід сервера, базується на використанні FastAPI для реалізації серверної частини системи. Вибір цього фреймворку обумовлений його високою продуктивністю, швидкістю відгуку та підтримкою асинхронного програмування, що особливо важливо для обробки великого обсягу запитів. FastAPI забезпечує просту та зрозумілу синтаксичну структуру для створення RESTful API, що сприяє швидкому розгортанню та масштабуванню серверних додатків. Цей підхід дозволяє забезпечити ефективне взаємодію з клієнтами системи та зручну розробку нового функціоналу без значних витрат часу на налаштування інфраструктури.

Тип структури полягає у тому, що потрібно зробити розподілення сервера на логічні модулі, наприклад, модуль для обробки запитів користувачів, модуль для роботи з базою даних, модуль для обробки алгоритмів штучного інтелекту тощо.

Дана структура частково подібна на мікросервісну архітектуру, де одна велика програма розподілена на декілька окремих значно менших програм, які є незалежними між собою та є більш простими з боку підтримки, масштабування та розробки програмного забезпечення. Детальніше можна глянути на рисунку 2.1.

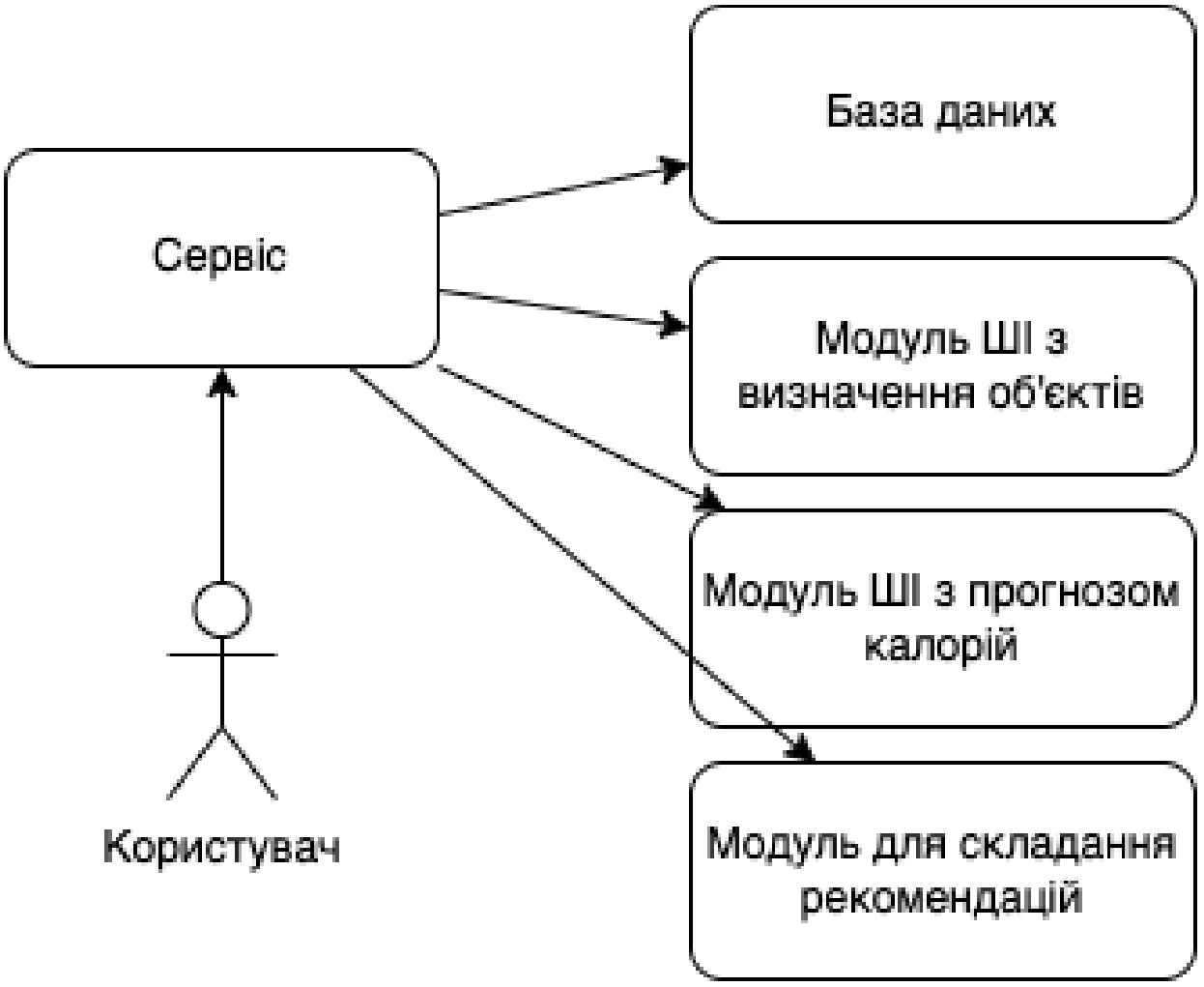

Рисунок 2.1 – Опис структури модулів серверної частини

Обробка запитів в Python FastAPI сервері з використанням двох модулів штучного інтелекту базується на високопродуктивній асинхронній архітектурі фреймворку.

FastAPI дозволяє зручно налаштовувати маршрути і контролери для взаємодії з першим модулем, що обробляє вхідні дані та виконує їх попередню обробку перед передачею другому модулю для більш глибокого аналізу.

Цей підхід забезпечує ефективну інтеграцію між модулями штучного інтелекту, забезпечуючи швидку обробку та відповідь на запити, що дозволяє системі зберігати високу продуктивність навіть при значному навантаженні. Детальніше можна ознайомитися на рисунку 2.2.

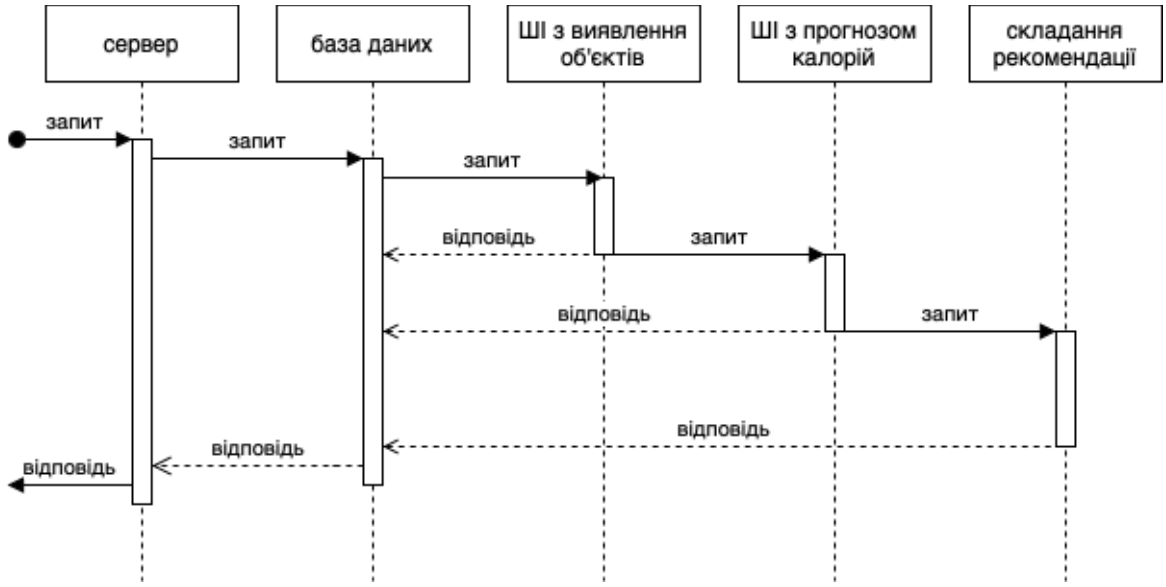

Рисунок 2.2 - Діаграма послідовностей обробки запитів користувача

Автентифікація та безпека даних на сервері FastAPI забезпечуються за допомогою різноманітних механізмів, таких як JWT, OAuth2, або базова автентифікація HTTP. Це дозволяє забезпечити захист ресурсів сервера шляхом перевірки ідентифікаційних даних користувача перед наданням доступу до API. FastAPI також підтримує вбудовані методи для криптографічного захисту паролів та інших конфіденційних даних, що знижує ризик витоку інформації через зловмисний доступ до сервера.

Моніторинг та логування на сервері FastAPI грають важливу роль у забезпеченні ефективності та безпеки додатків. FastAPI надає підтримку для інтеграції з різними системами моніторингу, такими як Prometheus, що дозволяє вести детальні метрики про продуктивність і стан сервера. Щодо логування, FastAPI дозволяє налаштовувати журналізацію подій через бібліотеки, такі як модуль логування Python, щоб відстежувати дії користувачів та інші важливі події для подальшого аналізу і відлагодження системи. Це допомагає забезпечити ефективне управління та моніторинг додатків на основі FastAPI.

FastAPI забезпечує просту і ефективну інтеграцію з іншими сервісами завдяки підтримці асинхронних запитів та стандартним інтерфейсам API, таким як RESTfull. Це дозволяє легко обмінюватися даними з іншими мікросервісами, зовнішніми API або базами даних, забезпечуючи гнучкість та масштабованість

архітектури вашого додатку. FastAPI також підтримує різноманітні бібліотеки та інструменти для реалізації комплексних сценаріїв інтеграції, що спрощує розробку розподілених систем.

Масштабування серверних додатків та систем штучного інтелекту є критичним аспектом у сучасному інформаційному середовищі. Використання Python FastAPI для масштабування сервера дозволяє ефективно обробляти великий обсяг запитів завдяки асинхронній обробці і вбудованій підтримці RESTful API. Цей фреймворк забезпечує високу швидкість відгуку і здатність до масштабування завдяки асинхронній природі та вбудованим інструментам для оптимізації продуктивності.

Щодо масштабування систем штучного інтелекту, Python використовується для розробки та імплементації алгоритмів машинного навчання і нейронних мереж. Використання бібліотек, таких як TensorFlow або PyTorch, дозволяє працювати з складними моделями і забезпечує масштабовану обробку даних завдяки можливості використання GPU або розподіленим обчисленням. Узагальнюючи, масштабування сервера на Python FastAPI та системи штучного інтелекту в Python відображає стрімкий розвиток технологій у напрямку забезпечення високої доступності, ефективності та відповідності до сучасних вимог до інформаційних систем.

2.2 Проєктування структури серверної частини програми

Вибір мови програмування Python для сервера з широким застосуванням ШІ обґрунтовується кількома ключовими перевагами. По-перше, Python відомий своєю чистотою коду та легкістю в освоєнні, що робить його ідеальним для швидкого розроблення та підтримки складних систем. Другим аспектом є широкий вибір сторонніх бібліотек і фреймворків для розробки ШІ, що спрощує інтеграцію з різноманітними протоколами та системами. Крім того, підтримка асинхронного програмування у Python забезпечує ефективну обробку великого

обсягу запитів, що є важливим для розгортання серверних додатків з високою відповідальністю і швидкодією.

FastAPI є сучасним фреймворком для створення веб-додатків на Python, який здобуває все більшу популярність завдяки своїм перевагам у продуктивності. Перш за все, FastAPI використовує сучасні підходи до асинхронного програмування, що дозволяє обробляти багато запитів одночасно і ефективно використовувати ресурси сервера. Це особливо корисно для високонавантажених застосунків, де кожна мілісекунда має значення. FastAPI базується на стандартних інструментах Python, таких як Type Hints, що полегшують розробку і підтримку коду, а також роблять його більш прогнозованим і зрозумілим для команди розробників.

Другий аспект, який підвищує продуктивність FastAPI, - це його автоматична генерація документації. FastAPI автоматично генерує інтерактивну OpenAPI документацію зі схемами запитів та відповідей, що дозволяє розробникам швидко ознайомитися з API та тестувати його без зайвих зусиль. Це зменшує час, потрібний для розгортання інтерфейсу API, та сприяє ефективній комунікації між командами розробників і замовниками проектів. Крім того, FastAPI підтримує декларативний підхід до визначення маршрутів API, що спрощує їх розширення та модифікацію, забезпечуючи гнучкість і швидкість в управлінні проектами різної складності.

Проєктування структури сервера на FastAPI включає кілька ключових аспектів, починаючи з визначення маршрутів API за допомогою декораторів Python, які вказують URL-шляхи та HTTP-методи кожного ендпоінту. Для організації бізнес-логіки і обробки запитів рекомендується використовувати модульну структуру, розділяючи код на різні модулі або пакети для кожної логічної частини додатку. Детальніше з публічними для користувача ендпоінтами можна ознайомитися у списку, а саме:

– ping;
– login;
– logout;

– getresult;
– getresultbyid.

Також важливо створити ендпоінти для адміністративного використання, а саме таких ендпоінтів, як:

– createuser;
– readuser;
– updateuser;
– deleteuser.

Відповідно до кожної таблиці, потрібно застосувати відповідні методи до паттерну проєктування CRUD[11].

– create;
– read;
– update;
– delete.

Важливо враховувати асинхронність в FastAPI, що дозволяє оптимізувати ресурси сервера та підтримувати багатокористувацьке обслуговування. Налаштування бази даних через ORM, наприклад, SQLAlchemy або звичайні SQL-запити дозволяє забезпечити ефективний доступ до даних.

Забезпечення безпеки здійснюється за допомогою вбудованих інструментів FastAPI для аутентифікації та авторизації, таких як OAuth2 або JWT. Логування подій і моніторинг стану сервера забезпечуються через налаштування систем логування та інтеграцію з іншими інструментами моніторингу, такими як Prometheus.

Врахування принципів RESTful дизайну API, які включають у себе використання належних HTTP-методів та статус-кодів для повідомлення результатів операцій, є ключовим аспектом при проєктуванні структури сервера на FastAPI. Детальніше з статус кодами ендпоінтів можна ознайомитися у списку, а саме:

– 1XX, інформаційні коди;
– 2XX коди успіху;

– 3XX коди переадресації;
– 4XX коди помилок із боку користувача;
– 5XX коди помилок сервера.

Структура проєкту в FastAPI як правило має включати також додаткові компоненти, такі як робота з файлами, завантаження та зберігання даних, а також реалізацію асинхронних фонових задач через Celery або інші вбудовані асинхронні бібліотеки, але в даному проєкті перевага буде надана на роботу з базою даних для зберігання даних.

2.3 Проєктування структури бази даних програми

Для створення структури бази даних, потрібно підготувати середовище виконання. Середовище виконання бази даних для сервера з Python включає в себе ряд ключових складових. Воно визначається вибором СУБД[12], такими як SQLite[13], PostgreSQL, MySQL або MongoDB, залежно від вимог проекту до масштабованості, продуктивності та типів даних. Це середовище також включає налаштування з'єднання з базою даних через Object-Relational Mapping або прямі SQL-запити, що забезпечують взаємодію з даними сервера, що працює під управлінням Python. Важливою частиною є також забезпечення безпеки даних, включаючи механізми автентифікації та авторизації, а також захист від несанкціонованого доступу і збереження конфіденційності інформації.

Проєктування бази даних для сервера на Python включає кілька ключових етапів:

– вимоги до бази даних;
– аналіз даних;
– моделювання;
– вибір СУБД.

При проектуванні БД для сервера з FastAPI Python важливо враховувати ряд вимог для забезпечення ефективної роботи додатку, що було сформовано з технічного завдання, а саме такі пункти як:

– підтримка асинхронних операцій;
– зручність у використанні;
– можливість зберігання файлів, включаючи зображення;
– масштабованість і продуктивність;
– підтримка транзакцій;
– безпека даних.

Врахування цих вимог допоможе обрати відповідну базу даних для даного сервера на FastAPI Python, забезпечуючи високу ефективність, надійність і зручність у використанні для продовження розробки програмного забезпечення.

Для аналізу даних, які будуть використовуватися в базі даних з таблицями користувачів та результатами роботи ШІ у форматі фото, необхідно врахувати кілька ключових аспектів. По-перше, важливо збирати і зберігати основну інформацію про користувачів, таку як ідентифікатори, ім'я, електронну пошту і додаткові дані. Другим аспектом є ефективне зберігання і керування фотографіями, включаючи їхні бінарні дані, метадані та спеціальні характеристики, такі як розмір, формат і дата завантаження. Третім аспектом є забезпечення взаємозв'язку між цими двома сутностями через унікальний ідентифікатор або відповідні зовнішні ключі для зручного доступу та управління даними. Нарешті, врахування аспектів безпеки, таких як конфіденційність особистих даних користувачів і захист фотографій від несанкціонованого доступу, є критичним для забезпечення надійності та захисту інформації в системі.

Моделювання бази даних є ключовим етапом при розробці інформаційних систем. Воно включає в себе процес створення логічної структури даних, що відображає бізнес-процеси та взаємозв'язки між різними сутностями. Основні аспекти моделювання включають визначення сутностей (таблиць), атрибутів і їх типів даних, а також відносин між таблицями для забезпечення цілісності і нормалізації даних. Врахування вимог до продуктивності, масштабованості та безпеки даних є критичним для успішного моделювання бази даних. Детальніше з ER діаграмою бази даних можна ознайомитися на рисунку 2.3.

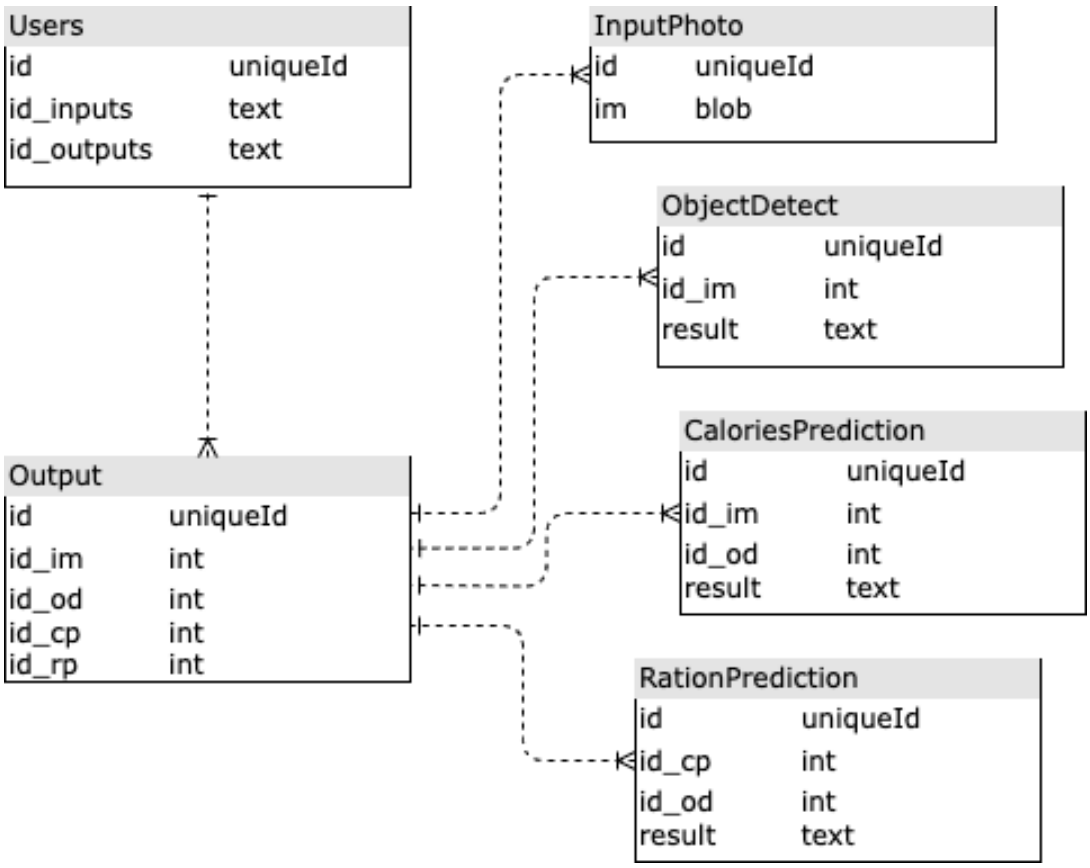

Рисунок 2.3 – ER діаграма бази даних

Вибір СУБД, а саму вибір SQLite3 для сервера на Python обґрунтовується кількома ключовими факторами. По-перше, SQLite3 є легко використовуваною вбудованою базою даних, яка не вимагає окремого сервера і працює без залучення зовнішніх процесів, що спрощує розгортання та підтримку. Другим аспектом є підтримка транзакцій і операцій ACID, що забезпечує надійність і цілісність даних. Крім того, SQLite3 підтримує велику кількість типів даних, включно з фото, також відмінно працює з Python завдяки вбудованій підтримці, що робить його ідеальним вибором для невеликих і середніх проектів з обмеженими вимогами до масштабованості та одночасних підключень.

Ці етапи дозволяють ефективно проектувати і впроваджувати базу даних для сервера на Python, забезпечуючи надійність, продуктивність і безпеку обробки даних у вашому додатку.

2.4 Проєктування структури моделі для виявлення об’єктів

Проєктування моделі штучного інтелекту є багатоступеневим процесом, який охоплює різні аспекти від визначення цілей до впровадження і тестування. Ось основні пункти, які допоможуть зрозуміти ключові етапи та аспекти цього процесу:

– визначення цілей та вимог;
– збір та підготовка даних[14];
– вибір та підготовка моделі.

Вище згаданий список охоплює основні кроки, необхідні для успішного проєктування та впровадження моделі ШІ. Кожен з цих етапів є критично важливим і потребує ретельного планування та виконання.

Ціллю програмного модуля на Python з ШІ є автоматизація процесу OD їжі на фото, подібним що були використовувалися у великому обсязі даних, тобто датасеті. Вимоги включають розробку алгоритмів комп'ютерного зору для точного визначення об'єктів їжі з різноманітних кутів зору та у різних умовах освітлення. Додатковими вимогами є ефективність алгоритмів у виявленні об'єктів різних типів їжі, наприклад таких як фрукти, овочі, м'ясо і так далі. Необхідно також забезпечити можливість ідентифікації і класифікації об'єктів їжі на основі їхньої форми, кольору та текстури. Особливу увагу слід звернути на оптимізацію продуктивності модуля для швидкого оброблення великого обсягу даних без втрати точності виявлення.

Збір та підготовка даних. Пошук зображень їжі та створення датасету для OD. У даному пункті описано процес пошуку зображень їжі та створення з них датасету для задачі OD. Збір та підготовка датасету є критично важливими етапами для побудови ефективних моделей машинного навчання. Особлива увага приділяється джерелам зображень, методам збору даних, а також процесу їх обробки та анотації.

Пошук зображень їжі проводився за допомогою декількох джерел, що дозволило отримати різноманітний та багатий набір даних. Основними джерелами стали Google Images[15], Kaggle[16] та HuggingFace[17].

Google Images є одним з найбільш доступних та зручних інструментів для пошуку зображень. Процес пошуку включав наступні кроки:

– визначення ключових слів;
– фільтрація зображень;
– завантаження зображень.

Kaggle є відомою платформою для спільного використання датасетів та проведення змагань з машинного навчання. Для пошуку зображень їжі на Kaggle використовувалися наступні методи:

– пошук готових датасетів;
– завантаження та перевірка.

HuggingFace є ще однією важливою платформою для обміну датасетами. Процес збору даних включав:

– вибір відповідних датасетів;
– завантаження та перевірка.

Після збору зображень було здійснено їх підготовку та анотацію для використання в задачі Object Detection.

– попередня обробка;
– аналіз якості.

Для задачі Object Detection необхідно було анотувати зображення, зазначивши положення об'єктів у вигляді їжі на зображеннях:

– інструменти для анотації, в даному випадку сервіс Makesense[18];
– процес анотації, або так званий лейблінг даних.

Процес пошуку та створення датасету для Object Detection виявився трудомістким, але надзвичайно важливим етапом. Використання різноманітних джерел, таких як Google Images, Kaggle та HuggingFace, дозволило отримати багатий та різноманітний набір зображень. Ретельна обробка та анотація

зображень забезпечили високу якість підготовленого датасету, що є ключовим для подальших етапів роботи з моделями машинного навчання.

Вибір та підготовка моделі. Для завдання визначення об’єктів, була обрана вже готова архітектура моделі YOLOv5[19] від компанії Ultralytics[20], її використання для визначення об'єктів обумовлені її високою продуктивністю та точністю в задачах комп'ютерного зору. YOLOv5 вирізняється швидкістю роботи та здатністю в реальному часі детектувати об'єкти на зображеннях, що робить її ідеальною для застосувань, де потрібна оперативна обробка даних. Підготовка моделі включає етапи попередньої обробки даних, таких як аугментація зображень та нормалізація. Відповідно після цього проводиться налаштування гіперпараметрів і тренування моделі на спеціально підібраному наборі даних, що забезпечує високу якість розпізнавання об'єктів. Для підвищення точності, модель може бути донавчена на нових даних, що відповідають специфіці конкретного завдання. Оцінка результатів проводиться за допомогою метрик, таких як точність, повнота та середнє значення показника IoU (Intersection over Union).

2.5 Проєктування структури моделі для прогнозу калорій

Як було згадано у попередньому розділі, проєктування моделі штучного інтелекту є багатоступеневим процесом та подібним не залежно від типу моделі чи складності, що охоплює різні кроки від визначення цілей до впровадження і тестування. Ось основні пункти, які допоможуть зрозуміти ключові етапи та аспекти цього процесу:

– визначення цілей та вимог;
– збір та підготовка даних;
– вибір та підготовка моделі.

Вище згаданий список охоплює основні кроки, необхідні для успішного проєктування та впровадження моделі ШІ. Кожен з цих етапів є критично важливим і потребує ретельного планування та виконання.

Визначення цілей та вимог. Метою даного розділу є розробка моделі на базі штучного інтелекту, здатної прогнозувати кількість калорій в їжі за фото. Основними вимогами до моделі відповідно до технічного завдання є висока точність і швидкість розпізнавання, що дозволить користувачам отримувати миттєві результати в режимі реального часу.

Для досягнення цих цілей необхідно забезпечити великий і різноманітний набір даних зображень їжі з відповідною інформацією про калорійність для тренування моделі. Крім того, модель повинна бути здатна розпізнавати різні типи страв та їх складові, щоб надавати користувачам точну інформацію про їхній раціон.

Збір та підготовка даних. Даний пункт включає використання датасету для моделі OD, що містить зображення різних страв. До цього датасету додаються додаткові файли, які містять інформацію про калорійність кожного зображеного блюда. Ці додаткові дані дозволяють моделі не тільки визначати наявність та тип їжі на фото, але й прогнозувати її калорійність з високою точністю.

Підготовка даних також включає аугментацію зображень для підвищення загальної продуктивності та стійкості моделі до різних умов зйомки.

Вибір та підготовка моделі. Вибір моделі для прогнозування калорій по фото зупинився на ResNet-50[21] завдяки її здатності ефективно розпізнавати складні візуальні патерни.

ResNet-50, як глибока нейронна мережа з 50 шарами, забезпечує високу точність у класифікації зображень, що є критичним для точного визначення калорійності страв. Підготовка моделі включає попереднє тренування на великому наборі зображень їжі з анотованою інформацією про калорійність, що дозволяє моделі навчитися зв'язку між візуальними характеристиками їжі та її енергетичною цінністю.

Після цього модель проходить етап тонкого налаштування на специфічному наборі даних, що відповідає задачам користувачів, з метою покращення її продуктивності в реальних умовах.

2.6 Проєктування програмного модуля для складання рекомендацій раціону

Аналіз завдання. Завдання для програмного модуля для складання раціону на мові Python полягає в аналізі даних про спожиту їжу для складання персоналізованого раціону людини.

Модуль має обробляти інформацію про кількість спожитих калорій, макро та мікроелементів, порівнювати ці дані з рекомендованими нормами та надавати рекомендації щодо коригування дієти.

Крім того, передбачається можливість відстежування прогресу користувача і генерація звітів про стан харчування.

Пошук можливих рішень та опис концепції. Розробка програмного модуля на мові Python для аналізу даних спожитої їжі та складання раціону людини вимагає інтеграції з вже існуючими ШІ технологіями, такими як ChatGPT[22], для автоматизації цього процесу.

Використання саме стороннього модуля ШІ для складання рекомендацій полягає в тому, що дана технологія вирішує багато проблем, як мінімум задовольняє технічне завдання та зменшує тривалість розробки програмного продукту.

Було обрано ChatGPT замість Gemini[23] через те, що останній має обмежену підтримку мов, що ускладнює його використання для багатомовних проєктів. Крім того, розробникам значно складніше працювати з Gemini через відсутність зручних інструментів і ресурсів для розробки.

ChatGPT забезпечує ширшу мовну підтримку та більш дружнє середовище для розробників, що полегшує інтеграцію та використання у різних проєктах. ChatGPT дозволяє обробляти великі обсяги даних про споживану їжу, аналізувати їх та генерувати персоналізовані рекомендації щодо раціону. Це зменшує необхідність ручної обробки даних та забезпечує високу точність і релевантність результатів.

Крім того, ChatGPT може враховувати індивідуальні особливості користувачів, такі як алергії, вподобання та дієтичні обмеження, що робить процес складання раціону більш гнучким та адаптованим. Дана велика мовна модель, є оптимальним вибором відповідно ТЗ дипломної роботи.

Впровадження такого підходу сприяє ефективнішому управлінню харчуванням, економить час та ресурси, а також підвищує загальну якість життя користувачів.

2.7 Висновок до другого розділу

В другому розділі кваліфікаційної роботи було розглянуто проектування серверної архітектури на мові програмування Python.

Були створені два ключові модулі з використанням штучного інтелекту: модуль для об'єктного виявлення та модуль для прогнозування калорійного споживання.

Модуль об'єктного виявлення базується на алгоритмах глибокого навчання для точного розпізнавання продуктів на зображеннях користувача.

Модуль прогнозу калорій використовується для автоматичного обчислення калорійності їжі на основі розпізнаних об'єктів.

Реалізація цих модулів відбулася з використанням передових бібліотек та фреймворків штучного інтелекту, що забезпечило високу точність та швидкодію алгоритмів.

Інтеграція цих компонентів у серверну архітектуру забезпечила стабільну та ефективну роботу системи, яка готова до практичного застосування в побуті та сфері здорового способу життя.

Даний розділ є ключовим для продовження проробки будь якого програмного продукту.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Розробка серверної частини програми

Конфігурація сервера на Python з використанням FastAPI є ключовим етапом розробки сучасних веб-застосунків. FastAPI дозволяє створювати швидкі, асинхронні і високопродуктивні API завдяки використанню стандартних анотацій Python для валідації даних і генерації документації. Даний сервер має працювати цілком повністю на стороні сервера, що унеможливлює зворотної розробки, що в чергу не потребує використання методів по типу обфускації[24]. Для початку, встановлюємо FastAPI та Uvicorn, сервер для запуску нашого додатку, використовуючи pip, детальніше можна ознайомитися на рисунку 3.1. Далі, створюємо основний файл додатку відповідно спроєктованої структури додатку з розділу 2.1, де імпортуємо необхідні модулі та визначаємо ендпойнти, детальніше з основним файлом можна ознайомитися на лістингу 3.1, для більш детальнішого огляду можна звернутися в додаток А.

```
vasil.storchak@MacBook-Pro-Vasil ~ % pip install uvicorn fastapi
Requirement already satisfied: uvicorn in /usr/local/lib/python3.11/site-packages (0.25.0)
Requirement already satisfied: fastapi in /usr/local/lib/python3.11/site-packages (0.109.0)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.11/site-packages (from uv
icorn) (8.1.7)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.11/site-packages (from uvic
orn) (0.14.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4 i
n /usr/local/lib/python3.11/site-packages (from fastapi) (2.5.3)
Requirement already satisfied: starlette<0.36.0,>=0.35.0 in /usr/local/lib/python3.11/site-pa
ckages (from fastapi) (0.35.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/site-pac
kages (from fastapi) (4.8.0)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.11/site-packa
ges (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4->fastapi) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.6 in /usr/local/lib/python3.11/site-packag
es (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0,>=1.7.4->fastapi) (2.14.6)
Requirement already satisfied: anyio<5,>=3.4.0 in /usr/local/lib/python3.11/site-packages (fr
om starlette<0.36.0,>=0.35.0->fastapi) (4.2.0)
Requirement already satisfied: idna>=2.8 in ./Library/Python/3.11/lib/python/site-packages (f
rom anyio<5,>=3.4.0->starlette<0.36.0,>=0.35.0->fastapi) (3.6)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/site-packages (from
anyio<5,>=3.4.0->starlette<0.36.0,>=0.35.0->fastapi) (1.3.0)
vasil.storchak@MacBook-Pro-Vasil ~ %
```

Рисунок 3.1 – Встановлення програмних пакетів для сервера

Лістинг 3.1 – Основний файл сервера

```
@app.on_event("startup")
```

```
async def startup():
    logger.info("Server started")
@app.on_event("shutdown")
async def shutdown():
    logger.info("Server shutdown")
@app.get("/ping", status_code=status.HTTP_200_OK)
def ping():
    return {"ping": "pong"}
@app.post('/login', status_code=status.HTTP_200_OK)
async def ai_login(_request: Request, user: str = Form(...), password: str = Form(...)):
    pass
@app.post('/getresult', status_code=status.HTTP_200_OK)
async def ai_getresult(_request: Request, image: UploadFile, user: str = Form(...), password: str = Form(...)):
    pass
@app.post('/getresultbyid', status_code=status.HTTP_200_OK)
async def ai_getresultbyid(_request: Request, res_id: int, user: str = Form(...), password: str = Form(...)):
    pass
```

Розробка бази даних. Створення бази даних SQLite для FastAPI сервера є важливим етапом, що забезпечує збереження та обробку даних. SQLite є легковажною і простою у використанні базою даних, що ідеально підходить для прототипування та невеликих проектів. Для інтеграції SQLite з FastAPI використовуємо SQLAlchemy, ORM-бібліотеку для роботи з базами даних у Python. Спочатку встановлюємо необхідні пакети (рис. 3.2), створюємо модель бази даних відповідно спроєктованої структури із розділу 2.3 з визначенням таблиць та їхніх полів, а потім налаштовуємо сесію для взаємодії з базою даних. Нарешті, створюємо функції для виконання CRUD операцій, детальніше з

основною частино реалізації можна ознайомитися на лістингу 3.2, для більш повного ознайомлення можна звернутися в додаток Б.

```
vasil.storchak@MacBook-Pro-Vasil ~ % pip install databases SQLAlchemy
Requirement already satisfied: databases in /usr/local/lib/python3.11/site-packages (0.8.0)
Requirement already satisfied: SQLAlchemy in /usr/local/lib/python3.11/site-packages (1.4.51)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.11/site-packages (fr
om SQLAlchemy) (3.0.3)
vasil.storchak@MacBook-Pro-Vasil ~ %
```

Рисунок 3.2 – Встановлення пакетів для бази даних

Лістинг 3.2 – Основний файл реалізації бази даних сервера

```
import io
import databases
import sqlalchemy
import src.database.release as rel
from PIL import Image
from fastapi import APIRouter
from fastapi import UploadFile, Form
from src.config import *
database_app = APIRouter()
database_connector = databases.Database(DATABASE_URL)
metadata = sqlalchemy.MetaData()
rel.config_table(DATABASE_URL, DATABASE_TABLES)
metadata.create_all(engine)
@database_app.on_event("startup")
async def startup():
    await database_connector.connect()
@database_app.on_event("shutdown")
async def shutdown():
    await database_connector.disconnect()
rel.make_crud_for_table(DATABASE_URL, DATABASE_TABLES)
```

Розробка частини сервера для інтеграції з модулями ШІ є важливим кроком у створенні сучасних високотехнологічних рішень. Цей крок присвячений опису процесу налаштування серверної частини, яка забезпечить ефективну взаємодію з різними модулями ШІ, що будуть детально розглянуті у наступних розділах.

Також налаштовуємо механізм обробки помилок і логування, щоб забезпечити надійну роботу серверу та швидке виявлення можливих проблем, детальні про створену систему логування можна ознайомитися в додатку В. Всі ці аспекти створюють основу для інтеграції з модулями ШІ, які будуть детально описані в наступних розділах, включаючи архітектуру модулів, алгоритми ШІ та їхню реалізацію.

3.2 Створення моделі для виявлення об’єктів

Для тренування моделі штучного інтелекту на базі архітектури YOLOv5. від компанії Ultralytics будуть використані підготовлені дані, що включають зображення та відповідні анотації для OD. Ці дані були ретельно зібрані та оброблені для забезпечення високої якості та релевантності для задачі OD відповідно рекомендації, що до створення датасетів. Основна частина роботи полягала в аналізі різних середовищ виконання для тренування моделі, таких як Google Colab[25], Kaggle та власний GPU. Після детального порівняння цих платформ було вирішено віддати перевагу платформі Kaggle.

Основними критеріями для вибору були продуктивність, надійність та зручність використання. Kaggle. надає потужні обчислювальні ресурси, включаючи доступ до високопродуктивних GPU, що значно скорочує час тренування моделі. Крім того, Kaggle. забезпечує стабільне середовище з можливістю легкого доступу до широкого спектру додаткових інструментів і бібліотек для аналізу та візуалізації даних. Використання Kaggle. також дозволяє зберігати та легко ділитися результатами експериментів, що сприяє кращій відтворюваності та колаборації.

Загалом, рішення використовувати Kaggle. для тренування моделі YOLOv5. було прийнято завдяки його перевагам у продуктивності та надійності, що дозволяє ефективно реалізувати завдання даної дипломної роботи.

Для зручного та швидкого налаштування системи для тренування моделі був створений документ Jupyter Notebook[26], детальніше з документом можна ознайомитися на додатку Г.

Цей документ містить усі необхідні кроки, починаючи від завантаження та підготовки даних, і закінчуючи тренуванням моделі.

Використання Jupyter Notebook дозволяє легко виконувати код покроково та вносити зміни в налаштування в режимі реального часу.

В рамках цього проекту була натренована модель YOLOv5 на 100 епохах, що дозволило досягти високої точності OD. Процес тренування включав динамічний моніторинг показників продуктивності, таких як втрата (loss) та точність (accuracy).

3.3 Створення моделі для прогнозу калорій

Для тренування моделі штучного інтелекту на базі архітектури ResNet50 для прогнозування калорій будуть використані підготовлені дані, які включають зображення продуктів харчування та відповідні значення калорій. Вище згадані дані були старанно відібрані та попередньо оброблені, щоб гарантувати високу якість і відповідність поставленому завданню. Основна частина роботи полягала в аналізі різних середовищ виконання для тренування моделі, таких як Google Colab, Kaggle та власний GPU. Після детального порівняння цих платформ було вирішено віддати перевагу платформі Kaggle.

Основними критеріями для вибору Kaggle стали висока продуктивність, надійність та зручність використання. Kaggle надає потужні обчислювальні ресурси, включаючи доступ до високопродуктивних GPU, що значно скорочує час тренування моделі та забезпечує стабільне середовище. Крім того, Kaggle забезпечує легкий доступ до широкого спектру інструментів і бібліотек для

аналізу та візуалізації даних, а також можливість зберігати та ділитися результатами експериментів. Це робить платформу ідеальною для реалізації завдань даної дипломної роботи, дозволяючи ефективно тренувати модель ResNet50 та отримувати точні прогнози калорій.

У рамках створення моделі для прогнозу калорій було розроблено документ Jupyter Notebook, який забезпечує швидке та зручне налаштування системи для тренування моделі. Цей документ містить покрокові інструкції, які дозволяють швидко підготувати середовище для роботи з глибокими нейронними мережами. Використовуючи Jupyter Notebook, можна легко організувати та виконувати необхідні обчислення, а також візуалізувати результати. Детальніше з даним документом можна в додатку Ґ.

Однією з головних задач було тренування моделі ResNet50, яка є однією з найефективніших архітектур для задач класифікації зображень. Модель була натренована на 1000 епохах, що дозволило досягти високої точності та стабільності.

3.4 Створення програмного модуля для складання рекомендацій раціону

У рамках дипломної роботи було розглянуто та детально проаналізовано ТЗ щодо створення модуля складання рекомендацій.

У ході аналізу ТЗ були визначені ключові вимоги до системи, функціональні та нефункціональні характеристики, а також очікувані результати роботи модуля. На основі цього аналізу було сформовано план розробки, який на сам перед включав етапи проектування, реалізації та тестування модуля рекомендацій, в даному розділі описано етап реалізації.

Особливу увагу було приділено алгоритмам, які забезпечують персоналізовані рекомендації, та методам оцінки їхньої ефективності. Було враховано особливості вхідних даних і вибрано найбільш оптимальні моделі для їх обробки. Після завершення аналізу та планування, розробка модуля була

успішно здійснена відповідно до вимог ТЗ. Детальніше з основною частиною модуля можна ознайомитися на лістингу 3.3.

Лістинг 3.3 – Основний частина реалізації модуля складання рекомендацій раціону

```
import os
from openai import OpenAI
def setup(content, key="XXXX-XXXX-XXXX-XXXX"):
    os.environ['OPENAI_API_KEY'] = key
    client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY"),)
    return client
def pra(content):
    client = setup(content)
    chat_completion = client.chat.completions.create(
        messages=[
            {"role": "user","content": content,}
        ],
        temperature=0.1,
        model="gpt-3.5-turbo-16k",
    )
    return chat_completion['text']
```

Окремо було створено скрипт на основі ChatGPT, який інтегрувався в систему для забезпечення генерації рекомендацій на основі природної мови. Цей скрипт був оформлений як незалежний модуль сервера, що дозволило забезпечити його автономну роботу та легку інтеграцію з іншими компонентами системи. Такий підхід забезпечив гнучкість і масштабованість розробленого рішення, дозволяючи легко адаптувати його під різні потреби користувачів.

3.5 Тестування серверної частини

Опис очікуваного результату. У результаті тестування серверної частини програми на Python очікується підтвердження її стабільності та продуктивності. Перш за все, сервер має безперебійно обробляти запити клієнтів, навіть при значному навантаженні. Очікується, що програма коректно управляє ресурсами, включаючи пам'ять і процесорний час, не допускаючи їх витоків. Сервер повинен забезпечувати швидке виконання запитів та мінімальні затримки у відповідях. Безпека також є важливим аспектом: сервер має захищати дані користувачів від несанкціонованого доступу та можливих атак. Тестування включатиме перевірку правильності обробки різних типів даних та відповідей на некоректні запити. Важливо, щоб всі функціональні вимоги були виконані, а серверна частина надійно інтегрувалася з іншими компонентами системи.

Перевірка працездатності сервера можлива після запуску самого сервера. Для перевірки працездатності, можна використати вбудований ендпоінти Ping та Docs. Ендпоінт Ping повинен вертати просте текстове повідомлення (див. рисунок 3.3), ендпоінт Docs в свою чергу повинен вертати веб сторінку з усіма ендпоінтами та формами для тестування чи виклику, детальніше з усіма ендпоінтами можна на рисунку 3.4.

Рисунок 3.3 – Приклад використання ендпоінта Ping

Рисунок 3.4 – Веб сторінка усіх ендпоінтів сервера

Перевірка працездатності бази даних являє собою створення тестового запиту до сервера, наприклад в даному випадку запит на авторизацію користувача, так як дані користувачів взаємодіють з базою даних, також даний приклад може слугувати тестуванням авторизації, детальніше з тестовим запитом авторизації можна ознайомитися на рисунку 3.5.

Рисунок 3.5 – Тестовий запит авторизації на сервері

У рамках дипломної роботи було проведено тестування сервера та бази даних, реалізованих на Python. Зокрема, були створені тести для перевірки функціональності REST API та інтеграційні тести для забезпечення коректної взаємодії між сервером і базою даних. Використання вбудованих функцій у FastAPI дозволило пришвидшити процес тестування та виявити можливі помилки на ранніх етапах розробки. Усі тести успішно пройшли, що підтвердило стабільність і надійність розробленого програмного забезпечення.

3.6 Тестування моделі для виявлення об’єктів

Очікуваний результат роботи нейронної мережі з пошуку їжі на фото з використанням архітектури YOLOv5 полягає в точній та швидкій ідентифікації об'єктів їжі на зображеннях. YOLOv5, завдяки своїй ефективній архітектурі, здатний в режимі реального часу визначати та класифікувати різні типи їжі, такі як фрукти, овочі, м'ясо, десерти та інші об’єкти що були використані для тренування. Мережа виводить координати рамок, що окреслюють знайдені об'єкти, а також ймовірність того, що виявлений об'єкт належить до певної категорії їжі. Завдяки попередньому навчанні на достатній кількості зображень, нейронна мережа повинна забезпечувати високу точність виявлення елементів відповідно до технічного завдання, навіть в умовах складного фону або різного освітлення.

У рамках тестування моделі DO було підготовлено дані для тестування моделі штучного інтелекту YOLOv5. Спочатку було зібрано достатній набір зображень та виділено для тестування 20% усього датасету, що охоплюють різноманітні сценарії та умови освітлення. Потім дані були розмічені вручну, щоб забезпечити точність виявлення елементів моделлю. Далі зображення та їх розмітки були структуровані у відповідний формат, прийнятний для YOLOv5. Підготовлені дані успішно завантажено у модель, і початкові результати тестування показали високу точність розпізнавання об'єктів. Для більш

детального ознайомлення з метриками під час тренування можна переглянути рисунок 3.6.

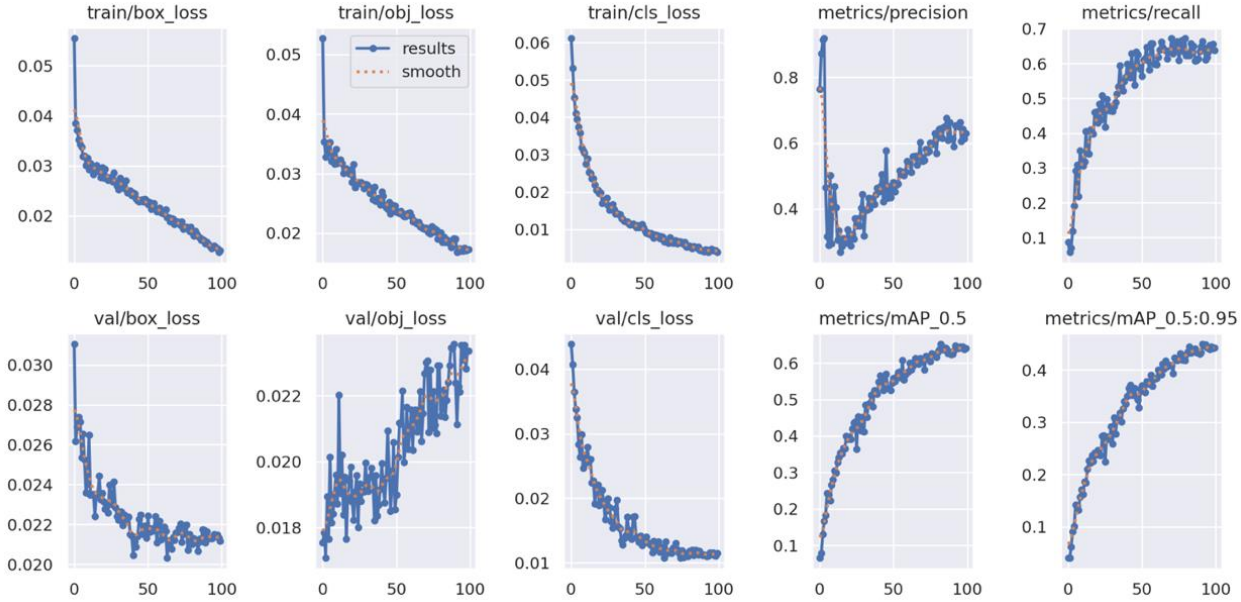

Рисунок 3.6 – Метрики моделі YOLOv5

Результати прогону тестових даних для моделі YOLOv5 під моделі Extra, були отримані і записані в таблицю 3.1 для ознайомленні.

Таблиця 3.1 – Результати метрики тестування моделі YOLOv5

-	Recall	Precision	mAP 0.5	mAP 0.5:0.95	cls_loss
YOLOv5 Extra	0.68	0.67	0.65	0.46	0.01

Під час тестування відповідно модуля ШІ сервера, середній час відповіді сервера складав менше 0.01 секунди з використанням GPU та в середньому 0.22 секунди з використанням CPU.

У рамках тестування моделі OD було проведено тестування моделі штучного інтелекту YOLOv5 під моделі Extra.

Модель була перевірена на підготовленому наборі даних, що включав різноманітні зображення та розмітки.

Результати тестування показали відповідно технічному завданню точність і швидкість OD.

Проведений аналіз результатів підтвердив ефективність і надійність моделі для практичного застосування.

3.7 Тестування моделі для прогнозу калорій

Очікуваний результат роботи нейронної мережі з прогнозування калорій їжі на фото з використанням архітектури ResNet-50 полягає в точній оцінці калорійності зображених страв.

ResNet-50, завдяки своїм глибоким резидуальним блокам, здатний ефективно витягати характеристики зображень, що дозволяє точно розпізнавати різні види їжі. Після попереднього навчання на великому наборі даних зображень їжі з відомими даними калорій, мережа може передбачати калорійність нових зображень.

Модель виводить числове значення калорійності на основі виявлених об'єктів та їх характеристик.

Це значення може бути використане в додатках для моніторингу харчування, здорового способу життя або у фітнес-програмах. Точність прогнозування повинна залежити від якості навчальних даних і повинна бути відповідною до технічного завдання.

Для тестування було підготовлено дані для моделі ResNet-50 з метою передбачення калорій. Було зібрано достатній набір зображень різних страв, кожне з яких супроводжувалося інформацією про калорійність. Дані були ретельно розмічені та відформатовані для навчання моделі ResNet-50.

Цей підготовчий етап забезпечив достатню точність відповідно технічному завданню моделі у визначенні калорійності на основі зображень їжі.

Для більш детального ознайомлення з результату тренування моделі ResNet-50 можна звернутися до рисунку 3.7. Для ознайомлення з метриками можна звернутися до таблиці 3.2.

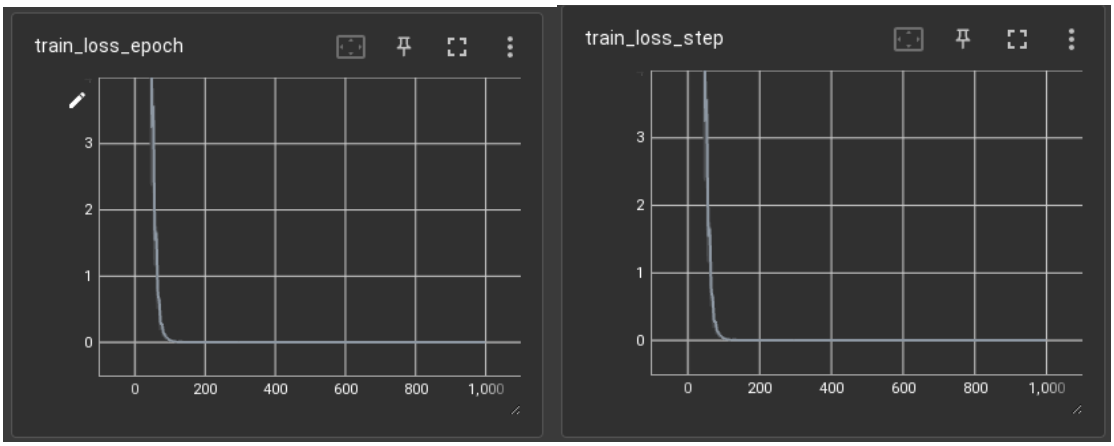

Рисунок 3.7 – Тренувальні метрики моделі ResNet-50

Таблиця 3.2 – Результати метрики тестування моделі ResNet-50

-	Recall	Precision	Accuracy	Specificity	F1 score
ResNet-50	0.65	0.62	0.68	0.9	0.62

Під час тестування відповідно модуля ШІ сервера, середній час відповіді сервера складав менше 0.001 секунди з використанням GPU та в середньому 0.01 секунди з використанням CPU.

Під час проведення тестування моделі штучного інтелекту ResNet-50 для передбачення калорійності страв. Модель було перевірено на підготовленому наборі зображень їжі з відомою калорійністю. Результати тестування показали високу точність передбачень моделі. Отримані результати були детально описані, що підтвердило ефективність ResNet-50 для цієї задачі.

3.8 Тестування програмного модуля для складання раціону

Очікуваний результат роботи програмного модуля зі складання рекомендацій раціону на основі з'їденої їжі полягає в наданні індивідуалізованих порад щодо харчування. Модуль аналізує введені дані про з'їдені страви, включаючи їх калорійність та інші дані. На основі цього аналізу він генерує рекомендації, що допомагають досягти заданих цілей користувача, таких як схуднення, підтримка ваги або нарощування м'язової маси. Програмний модуль

також враховує персональні дані, такі як вік, стать, рівень фізичної активності та медичні обмеження. Результати включають пропозиції щодо змін у раціоні, збалансованих страв та продуктів, яких слід уникати або споживати більше.

Було підготовлено вхідні дані для тестування модуля складання рекомендацій індивідуальної дієти. Дані були ретельно підготовлені, щоб відповідати вимогам модуля рекомендацій згідно технічному завданню. Також були складені критерії для очікуваних відповідей згідно з технічним завданням, включаючи точність відповідності дієтичних рекомендацій індивідуальним потребам користувачів. Ці критерії забезпечили об'єктивність та прозорість процесу тестування. Детальніше з відповіддю модуля складання рекомендацій індивідуальної дієти, можна ознайомитися на рисунку 3.8.

```
storchak.vasil@localhost: ~/Dropbox> python3.10 pra.py <<< "я звичайна здорова людина, я зїв \
    сьогодні 300 калорій борщу, 200 калорій чізкейку, \
    300 калорій рису, дай рекомендації мого раціон"
Answer:
Згідно з вашими споживаннями їжі, ви спожили загалом 800 калорій, які включали висококалорійну, \
солодку та жирну їжу. У повсякденному житті рекомендується їсти більше продуктів, що містять \
білок, вітаміни та мінерали, наприклад:

* Білок: курка, риба, бобові, молочні продукти
* Вітаміни: овочі, фрукти, цільнозернові каші, горіхи
* Мінерали: горіхи, бобові, зернові

При цьому рекомендується контролювати споживання калорій і уникати надмірного споживання цукру \
і жиру. Рекомендована добова норма споживання калорій становить 2000-2500 калорій, яка \
регулюється відповідно до потреб вашого організму та рівня активності.

Крім того, якщо ви хочете зменшити споживання калорій, спробуйте наступне:

* Зменшіть або виключіть солодощі та жирну їжу
* Їжте більше овочів і фруктів
* Вибирайте продукти з низьким вмістом жиру та цукру
*Пийте більше води, щоб підтримувати водний баланс організму

Пам’ятайте, що кожна людина має різні фізичні потреби та рівні активності, а також потребує \
персоналізованого плану харчування відповідно до ситуації.
```

Рисунок 3.8 – Відповідь модуля складання рекомендацій індивідуальної дієти

Проведені випробування показали, що модуль успішно генерує персоналізовані дієтичні плани.

3.9 Висновок до третього розділу

Розділ про розробку та тестування дипломної роботи "Розробка сервера ”Щоденник харчування” на основі штучного інтелекту та REST API для нього для вироблення рекомендацій при складанні індивідуальної дієти" є ключовим для розуміння процесу створення й ефективності реалізації проекту. Також даний розділ є кінцевою метою при розробці будь якого програмного забезпечення.

Під час розробки сервера було реалізовано архітектуру програми з використанням мови програмування Python та спеціалізованих бібліотек для обробки зображень та машинного навчання.

В процесі тестування було перевірено цілісність та відповідності висунутим вимогам результат роботи модулів штучного інтелекту та інших важливих компонентів даного програмного продукту, що дозволило підвищити стабільність та якість сервера.

Основним етапом розробки був ШІ модуль для OD, що використовує передові методи комп'ютерного зору та нейронних мереж для точного аналізу зображень.

Також був імплементований модуль для прогнозування калорій, який базується на аналізі фотографій страв і використовує алгоритми машинного навчання для точності прогнозування.

В додаток було розроблено і протестована програмна модель для складання рекомендацій користувачу на основі з’їденої їжі та особливостей раціону кожного конкретного користувача

Загальний висновок полягає в тому, що розроблений Python сервер з ШІ модулями є ефективним інструментом для автоматизації процесів OD на зображеннях та прогнозування калорій за фото, що відкриває нові можливості у галузі комп'ютерного зору та здорового способу життя.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Навчання працюючих і інструктажі з охорони праці

Однією із складових ефективної роботи з профілактики виробничого травматизму є належна підготовка, навчання та підвищення кваліфікації працівників з питань охорони праці. Загальний порядок проведення навчання з питань охорони праці встановлений Законом України «Про охорону праці» (ст. 18. «Навчання з питань охорони праці»).

Виконання вимог Закону України «Про охорону праці»[27] в частині проведення навчання та перевірки знань з питань охорони праці здійснюється відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держкомітету України з нагляду за охороною праці 26 січня 2005 р. №15 (далі — Типове положення).

Нагляд за дотриманням вимог Типового положення здійснюють органи державного нагляду за охороною праці, а координацію та методичний супровід — Головний навчально-методичний центр та навчальні підрозділи експертно-технічних центрів Держгірпромнагляду [28].

Вивчення предмета «Охорона праці» при підготовці, перепідготовці та підвищенні кваліфікації працівників, які залучаються до виконання робіт з підвищеною небезпекою, на підприємстві регламентується п. 2.3 Типового положення. На підприємствах згідно з п. 1.1 Додатку 3 Типового положення навчання та перевірку знань з питань охорони праці повинні проходити керівники, заступники керівників, головні спеціалісти, керівники основних виробничих та технічних служб, безпосередньо пов’язані з організацією безпечного ведення робіт. Крім цього, згідно з п. 5 Додатку 3, навчання та перевірку знань з питань охорони праці мають проходити керівники, спеціалісти служб охорони праці, члени комісій з перевірки знань з питань охорони праці, особи, відповідальні за технічний стан і безпечну експлуатацію об’єктів підвищеної небезпеки підприємств [29].

Типове положення встановлює порядок та місце проведення навчання та перевірки знань з питань охорони праці посадових осіб (п. 5.2 та п. 5.3). Посадові особи, перелік яких наведено в п. 5.2, проходять навчання у Головному навчально-методичному центрі Держнаглядохоронпраці. Перевірка знань цієї категорії посадових осіб проводиться комісією, створеною наказом Держгірпромнагляду.

Організацію навчання та перевірки знань з питань охорони праці працівників на підприємстві здійснюють працівники служби кадрів або інші спеціалісти, яким роботодавець доручив організацію цієї роботи. Навчання та перевірка знань з питань охорони праці працівників (виконавців і посадових осіб), які не залучаються до виконання робіт підвищеної небезпеки, проводиться не рідше ніж один раз на три роки. Посадові особи та працівники, які виконують роботи підвищеної небезпеки, проходять спеціальне навчання та перевірку знань відповідних нормативно-правових актів з охорони праці не рідше одного разу на рік.

Спеціальне навчання з питань охорони праці може проводитись безпосередньо на підприємстві або навчальним закладом, який має відповідний дозвіл. При проведенні такого навчання на підприємстві навчальні плани та програми розробляються з урахуванням конкретних видів робіт, виробничих умов і функціональних обов’язків працівників і затверджуються наказом керівника підприємства.

Якщо на підприємстві неможливо створити комісію з перевірки знань з питань охорони праці (п. 4.4 Типового положення), перевірка знань проводиться комісією спорідненого підприємства або Теруправління Держгірпромнагляду. Всі працівники та посадові особи підприємства, включаючи посадових осіб, відповідальних за виконання робіт підвищеної небезпеки (крім зазначених в п. 5.2 та п. 5.3 Типового положення), проходять навчання та перевірку знань з питань охорони праці на підприємстві. Типове положення не зобов’язує, але й не забороняє проводити навчання всіх виконавців та посадових осіб (особливо тих, що виконують роботи підвищеної небезпеки) в навчальних закладах. У нашій

країні є багато підприємств, де таке навчання проводиться, і це має позитивні наслідки. Ті витрати, які при цьому несуть підприємства, перекриваються створенням більш безпечних умов праці і в результаті збереженням життя та здоров’я працівників.

Також в навчальних закладах проходять навчання та перевірку знань із загальних питань охорони праці всі посадові особи та фахівці, які проводять інструктажі або навчання підлеглих працівників з питань охорони праці, виконують роботи з проектування об’єктів, а також інші працівники, незалежно від того, передбачено таке навчання Типовим положенням чи ні.

4.2 Розробка раціональної діяльності та створення сприятливих умов трудового колективу

Чим більше людей залучено до спільної діяльності в організації, тим складнішим і відповідальнішим є процес організації та регулювання цієї діяльності. Наприклад, у офісі де створюються веб-застосунки, є край важливо створити сприятливі умови для трудового колективу. Навіть у малочисельній організації спільна діяльність персоналу є різноманітною і складною: відбувається у просторі й часі, супроводжується використанням різноманітних ресурсів, потребує великих витрат управлінської праці на нормування, планування, мотивацію, координацію, облік і контроль тощо.

Організація праці – приведення трудової діяльності людей в певну систему, що характеризується внутрішньою впорядкованістю, узгодженістю і спрямованістю взаємодії для реалізації спільної програми та цілі. Організація праці має як статичний аспект, тобто структуру праці, так і динамічний, тобто функціонування, поведінку, взаємодію людей в процесі трудової діяльності [30].

Організація праці на даному рівні передбачає:

– поділ і кооперацію праці;

– розробку методів праці, за допомогою яких може бути виконаний то й чи інший вид роботи;

– організацію робочих місць;

– створення умов праці;

– підбір і розстановку кадрів;

– визначення переліку функцій та обов`язків кожного працівника;

– встановлення міри праці за допомогою нормування;

– організацію оплати праці;

– встановлення правил дисципліни праці, які забезпечують порядок і узгодженість в роботі, режим праці і відпочинку працівників.

Регулювання трудової діяльності персоналу – це системно організований процес забезпечення і підтримання високоефективної праці кожного окремого працівника і трудового колективу в цілому; це найважливіше і постійне завдання менеджменту персоналу в організації. Регулювання трудової діяльності персоналу за метою, змістом і значенням має багато спільного з організацією праці, тому їх нелегко розвести, відділити одне від одного, та в цьому немає й потреби. Головне – чітко визначити, які лінійні керівники, функціональні підрозділи, допоміжні виробництва за що відповідають.

Основними завданнями регулювання трудової діяльності персоналу є:

– забезпечення продуктивної праці всіх категорій персоналу;

– підтримання якості продукції та послуг на конкурентоспроможному рівні;

– створення безпечних умов праці;

– підтримання у трудовому колективі сприятливого соціально-психологічного клімату.

Ефективність трудової діяльності персоналу залежить і формується під впливом багатьох різнопланових чинників. Усі їх з певною мірою умовності можна поєднати у дві групи. В першій групі варто виділити заходи, у процесі виконання яких створюються необхідні умови для діяльності персоналу, без чого взагалі неможлива нормальна робота структурного підрозділу, а про якийсь рівень ефективності й мови немає [31]. До першої групи чинників доцільно зарахувати наступні заходи:

– визначення мети діяльності;

– планування діяльності;

– кадрове забезпечення;

– нормативно-правове забезпечення;

– організаційно-економічне забезпечення;

– інженерно-технічне забезпечення.

Наступні чинники спрямовані на підтримання установленого ритму виробництва, виконання календарних графіків поставок замовникам, забезпечення руху транспортних засобів за розкладом. До другої групи належать такі заходи:

– ресурсне забезпечення;

– поточне обслуговування виробництва, трудових, транспортних та інших процесів;

– оперативне регулювання виробничих і трудових процесів;

– облік, контроль та оцінювання результатів діяльності.

Фахівці з управління персоналом виявили: одним з основних джерел незадоволеності роботою в організації – неефективна організація праці. Наприклад, не розмежовано зони відповідальності, співробітники виконують непотрібні дії, порушують технологічну послідовність операцій тощо. У даному випадку причина незадоволеності працівників – управлінські помилки, а саме: неправильно спроектовані робочі процеси, які можуть негативно впливати на продуктивність праці, створювати в підрозділі конфліктні ситуації, приводити до звільнення ключових співробітників, погіршувати задоволеність клієнтів [32].

Для регулювання трудової діяльності в організації розробляють та використовують таки документи:

– колективний договір організації[33];

– правила внутрішнього трудового розпорядку;

– положення про структурні підрозділи;

– посадові інструкції;

– правила експлуатації технологічного та іншого устаткування;

– норми і правила з охорони праці та техніки безпеки;

– штатний розпис організації в цілому та її структурних підрозділів (формування малих і великих трудових колективів: ланок, бригад, виробничих дільниць, цехів, відділів, секторів, бюро, лабораторій);

– графіки змінності;

– норми і ліміти витрачання всіх видів ресурсів (праці, зарплати, основних і допоміжних матеріалів, інструментів, енергетичних ресурсів тощо);

– календарні графіки виробництва продукції, відвантаження її замовникам;

– графіки і регламенти обслуговування робочих місць[34], потокових і конвеєрних ліній, складних технічних систем;

– графіки планово-попереджувального ремонту устаткування;

– положення щодо систем матеріального і морального заохочення працівників.

Звичайно, на ефективність діяльності організації справляють свій вплив різні чинники – внутрішні і зовнішні, об’єктивні й суб’єктивні. Адже, як відомо, саме кадри все вирішують. Не дарма японські менеджери найбільше уваги приділяють питанням розвитку людських ресурсів.

4.3 Висновок до четвертого розділу

Висновок до розділу про безпеку життєдіяльності. Аналіз різних видів небезпек та відповідних заходів захисту демонструє необхідність постійної обізнаності та підготовки особистої безпеки під час роботи та інших умов. Систематичний підхід до управління ризиками допомагає зменшити негативні наслідки для здоров'я та безпеки людей. Впровадження сучасних технологій і методів у сфері БЖД підвищує ефективність захисту. Отже, забезпечення безпеки життєдіяльності є невід'ємною складовою сталого розвитку суспільства.

ВИСНОВКИ

В даній дипломній роботі було розроблено Python REST API сервер Щоденник харчування на основі штучного інтелекту і також вироблення рекомендацій при складанні індивідуальної дієти та збережено в репозиторій[35].

Сервер складається з наступних компонентів:

– база даних, що містить інформацію про користувачів, продукти харчування, їхні поживні властивості та рецепти;

– модуль штучного інтелекту, що використовує машинне навчання для аналізу наданої фотографії та й для аналізу щоденника харчування користувача та вироблення рекомендацій щодо покращення його раціону;

– REST API, що забезпечує інтерфейс для взаємодії сервера з веб-додатками або також із іншими мобільними пристроями.

Розроблений сервер має наступні функції:

– зберігання інформації про щоденник харчування користувача;

аналіз фотографії з допомогою штучного інтелекту;

– аналіз щоденника харчування користувача за допомогою штучного інтелекту;

– вироблення рекомендацій щодо покращення раціону користувача сервісу;

– надання інформації про продукти харчування та їхні поживні властивості.

Сервер був протестований на широкій вибірці даних і показав відповідну точність та якість рекомендацій відносно технічного завдання.

Практичне значення даної роботи полягає в тому, що сервер Щоденник харчування може бути використаний для:

– допомоги людям у складанні здорового та збалансованого раціону;

– схуднення та контролю ваги;

– лікування та профілактики захворювань, пов'язаних з неправильним харчуванням;

– сервер може бути інтегрований з веб-додатками та мобільними пристроями для зручного використання користувачами.

Дана дипломна робота є дослідженням можливостей штучного інтелекту в галузі здорового харчування, а також кроком до створення комплексної системи для покращення харчування людей.

Проаналізувавши пророблену роботу, можна зробити висновок про подальші дослідження, які можуть бути дальше спрямовані для розвитку ідеї інтеграції штучного інтелекту у повсякденне життя, а саме такі напрямки як:

– розширення можливостей сервера, наприклад, додавання функцій планування їжі та моніторингу фізичної активності;

– покращення точності класифікації та корисності рекомендацій штучного інтелекту;

– проведення більш масштабних досліджень для оцінки ефективності сервера.

Впевнений, що дана робота матиме достатній вплив на життя людей та допоможе їм покращити своє здоров'я за допомогою правильного харчування з допомогою штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ

1. Сайт бібліотеки FastAPI. *fastapi.tiangolo.com.* URL: https://fastapi.tiangolo.com/ (дата звернення: 05.03.2024).

2. Опис технології REST API. *ibm.com*. URL: https://www.ibm.com/topics/rest-apis (дата звернення: 05.04.2024).

3. Фреймворк для машинного навчання TensorFlow. *tensorflow.org*. URL: https://www.tensorflow.org/ (дата звернення: 07.04.2024).

4. Технічна документація Python фреймворку PyTorch для розробки ШІ. *pytorch.org.* URL: https://pytorch.org/. (дата звернення: 07.04.2024).

5. Стаття про використання GPU для нейронних мереж. *geeksforgeeks.org.* URL: https://www.geeksforgeeks.org/how-to-use-gpu-acceleration-in-pytorch/. (дата звернення: 01.05.2024).

6. Ясній О. П. Захист програмного забезпечення на апаратному та програмному рівнях / О. П. Ясній, В. І. Карплюк // Матеріали VIII науково-технічної конференції „Інформаційні моделі, системи та технології“, 9-10 грудня 2020 року. — Т. : ТНТУ, 2020. — С. 105. — (Комп’ютерні системи та мережі).

7. Ясній О. П. Застосування методу нейронних мереж в машинному навчанні / О. Ясній, І. Дідич, С. Гречківський // Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології“, 1-2 лютого 2018 року. — Т. : ТНТУ, 2018. — С. 55. — (Секція 2. Інформаційні системи).

8. Технічна документація операційної системи Linux. *rocklylinux.org*. URL: https://docs.rockylinux.org/. (дата звернення: 03.04.2024).

9. Технічна документація мови програмування Python. *python.org.* URL: https://docs.python.org/3.10/. (дата звернення: 17.05.2024).

10. Сайт OpenCV проєкту. *opencv.org.* URL: https://opencv.org/ (дата звернення: 05.05.2024).

11. Стаття про CRUD. *sumologic.com.* URL: https://www.sumologic.com/ (дата звернення: 04.06.2024).

12. Вікіпедія. СУБД. *uk.wikipedia.org.* URL: https://uk.wikipedia.org/wiki/Система_керування_базами_ даних (дата звернення: 07.06.2024).

13. Вікіпедія. SQLite. *uk.wikipedia.org.* URL: https://uk.wikipedia.org/wiki/SQLite (дата звернення: 08.06.2024).

14. Стаття про створення датасетів для тренування нейронних мереж. *dou.ua.* URL: https://dou.ua/forums/topic/48983/ (дата звернення: 01.06.2024).

15. Сервіс google image. *google.com.* URL: https://google.com/ (дата звернення: 01.06.2024).

16. Сервіс для розробників нейронних мереж Kaggle. *kaggle.com.* URL: https://www.kaggle.com/ (дата звернення: 08.06.2024).

17. Сервіс для розробників нейронних мереж. *huggingface.co*. URL: https://huggingface.co/ (дата звернення: 08.06.2024).

18. Сервіс для створення датасетів ШІ. *makesense.ai*. URL: https://www.makesense.ai (дата звернення: 07.06.2024).

19. Документація архітектури YOLO для нейронних мереж. *ultralytics.com.* URL: https://docs.ultralytics.com/models/yolo-world (дата звернення: 01.06.2024).

20. Розробники архітектури YOLO для нейронних мереж. *ultralytics.com*. URL: https://www.ultralytics.com (дата звернення: 02.06.2024).

21. Технічна документація архітектури ResNet50 для ШІ. *pytorch.org*. URL: https://pytorch.org/resnet50 (дата звернення: 01.06.2024).

22. Велика мовна модель ChatGPT. *chatgpt.com*. URL: https://chatgpt.com/ (дата звернення: 01.06.2024).

23. Велика мовна модель Gemini. *gemini.google.com*. URL: https://gemini.google.com (дата звернення: 01.06.2024).

24. Ясній О. П. Методи обфускації програмного коду в комп'ютерних системах / О. П. Ясній, В. І. Карплюк // Збірник тез доповідей IX Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі

сучасних технологій“, 25-26 листопада 2020 року. — Т. : ТНТУ, 2020. — Том 2. — С. 77. — (Компютерно-інформаційні технології та системи зв’язку).

25. Універсальне робоче середовище для Data Science розробників. *colab.research.google.com*.
URL: https://colab.research.google.com (дата звернення: 04.06.2024).

26. Веб сайт проєкту Jupyter Notebook. *https://jupyter.org/*. URL: https://jupyter.org/community (дата звернення: 03.06.2024).

27. Закон України «Про охорону праці». *zakon.rada.gov.ua*. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 03.06.2024).

28. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.

29. Козлов С.С. Методичні вказівки до виконання розділу “Охорона праці та безпека в надзвичайних ситуаціях” в дипломних проектах для підготовки студентів факультету електроніки за освітньо-кваліфікаційним рівнем “Спеціаліст” та ”Магістр”. "Вимоги безпеки під час експлуатації обчислювальної техніки" / К.:НТУУ ”КПІ”, 2015, - 30 с.

30. Данюк В. М. Управління персоналом / В. М. Данюк, А. М. Колот. – Харків, 2017. – 664 с.

31. Створення безпечних і нешкідливих умов праці. Державний нагляд за охороною праці. *is.gd*. URL: https://is.gd/tPRqTr (дата звернення: 03.06.2024).

32. Створення сприятливих умов праці на виробництві. *oppb.com.ua*. URL: https://oppb.com.ua/news/stvorennya-spryyatlyvyh-umov-praci-na-vyrobnyctvi (дата звернення: 03.06.2024).

33. Колективний договір. *wiki.legalaid.gov.ua*. URL: https://wiki.legalaid.gov.ua/index.php/Колективний_договір (дата звернення: 02.06.2024).

34. Обслуговування робочих місць. *studfile.net*. URL: https://studfile.net/preview/5197678/page:4/ (дата звернення: 02.06.2024).

35. Сторчак В. Ю. GitHub репозиторій проєкту. *github.com*. URL: https://github.com/Who-are-me/FoodNoteApi (дата звернення: 01.06.2024).

ДОДАТКИ

Додаток А

Основний файл серверної частини

```
import logging
import multiprocessing
import platform
import gradio as gr
import time
import io
import src.ai_helper as ai
from src.dashboard.app import get_dashboard_app
from starlette.responses import FileResponse
from logging.config import dictConfig
from src.config import get_settings
from src.logger.LoggerConfig import LoggerConfig
from src.routers.api.v1 import router as api_router
from PIL import Image
from fastapi import FastAPI, UploadFile, HTTPException, status,
Form, Request
from fastapi.responses import HTMLResponse
from fastapi.middleware.wsgi import WSGIMiddleware
from src.database.database_sqlite import database_app
from src.gradio_setup import DEMO, GRADIO_ROUTE
from fastapi.templating import Jinja2Templates
settings = get_settings()
original_makeRecord = logging.Logger.makeRecord
templates = Jinja2Templates(directory="static/template")
def make_record_with_extra(self, name, level, fn, lno, msg, args,
exc_info, func=None, extra=None, sinfo=None):
    record = original_makeRecord(self, name, level, fn, lno, msg,
args, exc_info, func, extra, sinfo)
    record._extra = extra
    return record
logging.Logger.makeRecord = make_record_with_extra
dictConfig(LoggerConfig().model_dump())
```

```
logging.Formatter.converter = time.gmtime
logger = logging.getLogger(settings.logger)
app = FastAPI(title=settings.title, version=settings.version)
app = gr.mount_gradio_app(app, DEMO, path=GRADIO_ROUTE)
#                                          app.mount("/dashboard",
WSGIMiddleware(get_dashboard_app().server))
@app.on_event("startup")
async def startup():
    logger.info("Server started")
@app.on_event("shutdown")
async def shutdown():
    logger.info("Server shutdown")
@app.get("/ping", status_code=status.HTTP_200_OK)
def ping():
    return {"ping": "pong"}
@app.post('/login', status_code=status.HTTP_200_OK)
async def ai_login(_request: Request, user: str = Form(...),
password: str = Form(...)):
    try:
        res = ai.login(user, password)
    except Exception:
        raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,)
    return {'status': res}
@app.post('/logout', status_code=status.HTTP_200_OK)
async def ai_login(_request: Request, user: str = Form(...),
password: str = Form(...)):
    try:res = ai.logout(user, password)
    except Exception:
        raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,)
    return {'status': res}
@app.post('/getresult', status_code=status.HTTP_200_OK)
async def ai_getresult(_request: Request, image: UploadFile, user:
str = Form(...), password: str = Form(...)):
```

```
    try:
        ai_login(_request, user, password)
        contents = await image.read()
        with Image.open(io.BytesIO(contents)) as im:
            res, res_id = await ai.calculate(im, user, password)
            logger.info(f"status of AI work: {res}")
    except Exception:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail='There was an error uploading the file or
something else',
        )
    finally:
        await image.close()
    return {'status': 'successful', 'result_id': str(res_id)}
@app.post('/getresultbyid', status_code=status.HTTP_200_OK)
async def ai_getresultbyid(_request: Request, res_id: int, user: str
= Form(...), password: str = Form(...)):
    try:
        ai_login(_request, user, password)
        res, data = await ai.getbyid(res_id, user, password)
        logger.info(f"status of AI work: {res}")
    except Exception:
        raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,)
    return {'status': 'successful', 'data': str(data)}
app.include_router(api_router)
app.include_router(database_app)
```

Додаток Б

Файл конфігурації бази даних сервера

```
import io, databases, sqlalchemy
from PIL import Image
from fastapi import APIRouter
from fastapi import UploadFile, Form
from src.config import *
database_app = APIRouter()
database_connector = databases.Database(DATABASE_URL)
metadata = sqlalchemy.MetaData()
users_chemy = sqlalchemy.Table(
    DATABASE_USERS_TABLE, metadata,
    sqlalchemy.Column("id",                sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("true_answer", sqlalchemy.String),
    sqlalchemy.Column("pred_answer", sqlalchemy.String),
    sqlalchemy.Column("time_start", sqlalchemy.Float),
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),
)
output_chemy = sqlalchemy.Table(
    DATABASE_OUTPUT_TABLE,
    metadata,
    sqlalchemy.Column("id",                sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("true_answer", sqlalchemy.String),
    sqlalchemy.Column("pred_answer", sqlalchemy.String),
    sqlalchemy.Column("time_start", sqlalchemy.Float),
```

```
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),
)
input_chemy = sqlalchemy.Table(
    DATABASE_INPUT_TABLE,
    metadata,
    sqlalchemy.Column("id",                 sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("true_answer", sqlalchemy.String),
    sqlalchemy.Column("pred_answer", sqlalchemy.String),
    sqlalchemy.Column("time_start", sqlalchemy.Float),
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),
)
od_chemy = sqlalchemy.Table(
    DATABASE_OD_TABLE,
    metadata,
    sqlalchemy.Column("id",                 sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("true_answer", sqlalchemy.String),
    sqlalchemy.Column("pred_answer", sqlalchemy.String),
    sqlalchemy.Column("time_start", sqlalchemy.Float),
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),
)
pc_chemy = sqlalchemy.Table(
    DATABASE_PC_TABLE,
    metadata,
```

```
    sqlalchemy.Column("id",                 sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),)
pr_chemy = sqlalchemy.Table(
    DATABASE_PR_TABLE,metadata,
    sqlalchemy.Column("id",                 sqlalchemy.Integer,
primary_key=True),
    sqlalchemy.Column("user", sqlalchemy.String),
    sqlalchemy.Column("image", sqlalchemy.BINARY),
    sqlalchemy.Column("method", sqlalchemy.String),
    sqlalchemy.Column("time_end", sqlalchemy.Float),
    sqlalchemy.Column("time", sqlalchemy.Float),)
engine = sqlalchemy.create_engine(
    DATABASE_URL, connect_args={"check_same_thread": False})
metadata.create_all(engine)
@database_app.on_event("startup")
async def startup():
    await database_connector.connect()
@database_app.on_event("shutdown")
async def shutdown():
    await database_connector.disconnect()
@database_app.get("/log/{log_id}", description='-1 for last, 0
for all')  # , response_model=List[Log])
async def read_log(log_id: int):
    query = f"SELECT * FROM {DATABASE_OD_TABLE} WHERE id =
{log_id}"
    response = await database_connector.fetch_all(query=query)
    return response[0]
async def insert_into_log_db(image, method="", user="user",
true_answer="", pred_answer="", time_start=0., time_end=0.,
                             compress_image=(0, 0)):
```

```
        try:
            query = ("INSERT INTO log (user, image, method,
    true_answer, pred_answer, time_start, time_end, time) "
                     "VALUES (:user, :image, :method, :true_answer,
    :pred_answer, :time_start, :time_end, :time)")
            if compress_image[0] + compress_image[0] != 0:
                image.thumbnail((compress_image[0],
    compress_image[1]), Image.LANCZOS)
            buf = io.BytesIO()
            image.save(buf, format='JPEG')
            out = buf.getvalue()
            values = [{
                'user': user,
                'image': out,
                'method': method,
                'true_answer': true_answer,
                'pred_answer': pred_answer,
                'time_start': time_start,
                'time_end': time_end,
                'time': time_end - time_start}]
            await      database_connector.execute_many(query=query,
    values=values)
            return True
        except Exception as e:
            return False
    @database_app.post("/log/")
    async def create_log(image: UploadFile, method: str = Form(...),
    user: str = Form(...), true_answer: str = Form(...),
                         pred_answer: str = Form(...), time_start:
    float = Form(...), time_end: float = Form(...),
                         compress_image=(0, 0)):
        contents = await image.read()
        im = Image.open(io.BytesIO(contents))
    return await insert_into_log_db(im, method, user, true_answer,
pred_answer, time_start, time_end, compress_image)
```

Додаток В

Файл системи логування

```
from typing import Dict
from pydantic import BaseModel
from src.config import get_settings
class LoggerConfig(BaseModel):
    LOGGER_NAME: str = get_settings().logger
    LOG_FORMAT: str = ('%(asctime)s %(levelname)s [%(name)s]
[%(filename)s:%(lineno)d] - %(message)s')
    LOG_LEVEL: str = "DEBUG", version: int = 1
    disable_existing_loggers: bool = False
    formatters: Dict = {
        "default": {
            "()": "uvicorn.logging.DefaultFormatter",
            "fmt": LOG_FORMAT,
            "datefmt": "%Y-%m-%d %H:%M:%S",},
        "json": {
            "()": "pythonjsonlogger.jsonlogger.JsonFormatter",
            "fmt":      "%(asctime)s.%(msecs)03d      %(name)s
%(levelname)s %(message)s",
            "datefmt": "%Y-%m-%d %H:%M:%S"},}
    handlers: Dict = {
        "default": {
            "formatter": "default",
            "class": "logging.StreamHandler",
            "stream": "ext://sys.stderr",},
        "json": {"formatter": "json",
            "class": "logging.StreamHandler"},
        "datadog": {
            "formatter": "default",
            "class": "src.logger.datadog.DatadogHandler"}}
    loggers: Dict = {
        "app": {"handlers": ["default"], "level": LOG_LEVEL}, #
add datadog to handlers list "datadog"
```

```
        "uvicorn":    {"handlers":    ["default"],    "level":
LOG_LEVEL}}
```

Додаток Г

Скрипт тренування моделі YOLOv5

```
!pip install kaggle
kaggle_filename = 'kaggle.json'
key_to_kaggle = '{"username":"storchakvasil","key":"XXXXXX"}'

with open(kaggle_filename, 'w') as f:
    f.write(key_to_kaggle)
import os

root_path = '/kaggle/working'

# show all files, and check if exist kaggle config
for filename in os.listdir(root_path):
    if os.path.isfile(os.path.join(root_path, filename)):
      print(filename)

os.environ['KAGGLE_CONFIG_DIR'] = '/kaggle' #root_path + "/" +
kaggle_filename

!kaggle  datasets  download  -d  storchakvasil/foodnote-object-
detection
!unzip *.zip -d dataset
!cd dataset/valid/images && find . -type f | wc -l
!git clone https://github.com/ultralytics/yolov5.git
!cp -r dataset/* yolov5/
!pip install -r yolov5/requirements.txt
!cd  yolov5  &&  python  train.py  --data  data.yaml  --weights
runs/train/result_50x4/weights/last.pt --workers 4 --img 1280 -
-epochs 100 --batch-size 4 --name pre_result_50x --device 0
```

Додаток Ґ

Скрипт тренування моделі ResNet-50

```
import os
import pandas as pd
import numpy as np
import torch
import torchvision
import lightning as L
import torch.nn.functional as F
from PIL import Image
from torchvision import transforms, models
from torch.utils.data import Dataset, DataLoader
from torch import nn, optim
from lightning.pytorch.callbacks import ModelCheckpoint,
LearningRateMonitor
from lightning.pytorch.loggers import TensorBoardLogger
class FoodDataset(Dataset):
    def __init__(self, csv_file, root_dir, transform=None):
        self.df = pd.read_csv(csv_file)
        self.root_dir = root_dir
        self.transform = transform
        self.list_of_image =
self.list_images_in_directory(self.root_dir)
    def extract_class_name(self, path):
        parts = path.split('/')
        if len(parts) > 2:
            return parts[2]
        return None
    def list_images_in_directory(self, directory):
        image_extensions = ('.jpg', '.jpeg', '.png', '.bmp',
'.gif', '.tiff')
        image_files = []
        for root, dirs, files in os.walk(directory):
            for file in files:
```

```
                if file.lower().endswith(image_extensions):
                    image_files.append(os.path.join(root,
file))
        return image_files
    def __len__(self):return len(self.df)
    def __getitem__(self, idx):
        img_path = self.list_of_image[idx]
        img_class = self.extract_class_name(img_path)
        image = Image.open(img_path)
        calories     =     self.df.loc[self.df['class']     ==
img_class].cal.item()
        if self.transform:
            image = self.transform(image)
        return image, calories
class CaloriesPredictionModelCNNResnet(L.LightningModule):
    def __init__(self, output_dim):
        super(CaloriesPredictionModelCNNResnet,
self).__init__()                              #    self.model    =
torch.hub.load('pytorch/vision:v0.10.0',            'resnet50',
pretrained=True)
        self.model                                              =
torchvision.models.resnet50(pretrained=True)                    #
self.model.fc1 = nn.Linear(2048, 1024) # Replace last layer from
pretrained model
        self.model.fc = nn.Linear(2048, output_dim)
    def forward(self, x):
        return self.model(x)
    def criterion(self, target, prediction):
        loss = F.mse_loss(prediction, target)
        return loss
    def training_step(self, batch, batch_idx):
        image, target = batch
        prediction = self.model(image)
        loss          =          self.criterion(target.float(),
prediction.float())
```

```
        self.log("train_loss", loss, on_epoch=True)
        return loss
    def configure_optimizers(self):
      optimizer = torch.optim.AdamW(self.parameters(), lr=1e-2)
        return [optimizer]
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
0.225])])
dataset = FoodDataset(csv_file='./classes_dataset/cal.csv',
root_dir='./classes_dataset/', transform=transform)
dataloader = DataLoader(dataset, batch_size=64, shuffle=True,
num_workers=8)
model = CaloriesPredictionModelCNNResnet(1)
checkpoint_callback = ModelCheckpoint(
    monitor="val_loss",
    save_top_k=7,
    save_last=True,
    dirpath="./ckpt/",
    filename='{epoch}')
logger = TensorBoardLogger(save_dir=os.getcwd(), version=1,
name="./lightning_logs")
trainer = L.Trainer(accelerator="gpu",
                    precision=32,
                    max_epochs=1_000,
                    logger=logger,
                    log_every_n_steps=1,
                    accumulate_grad_batches=4,
                    callbacks=[checkpoint_callback,
LearningRateMonitor(logging_interval='epoch')])
trainer.fit(model, dataloader)
```