Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)
Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(освітньо-кваліфікаційний рівень)

Рекомендаційна система з приготування страв для набору розпізнаних та класифікованих продуктів харчування
(тема роботи)

Виконав:
студент 2 курсу, групи *СПРм-22-1*

Варламов М. Д.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(шифр і назва спеціальності)

Тип програми освітньо-наукова

Освітня програма Системне проєктування

Керівник *доц. Коваленко А.І*
(прізвище, ініціали)

Допускається до захисту

Зав. кафедри СТ _____ *проф. Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2024 р

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

13.06.2024  Варламов М.Д.

Атестаційна робота не містить відомостей заборонених до відкритого опублікування.

Атестаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведений «13» червня 2024 р.

Керівник кваліфікаційної роботи доц. каф. СТ Коваленко А.І.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____

Кафедра_____*Системотехніки*_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____122 – Комп'ютерні науки_____

Тип програми _____освітньо-професійна_____

Освітня програма__Інформаційні технології проектування_____

ЗАТВЕРДЖУЮ:

Зав. кафедри _____СТ_____

проф. Гребеннік І.В

(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Варламову Микиті Денисовичу*

(прізвище, ім'я, по батькові)

1. Тема роботи *Рекомендаційна система з приготування страв для набору розпізнаних та класифікованих продуктів харчування*

затверджена наказом по університету від " *01* **"** *квітня* **2024р. №** *582Ст*

2. Термін подання студентом роботи *13 червня 2022 р.*

3. Вихідні дані до роботи (проекту) *Розробити рекомендаційну систему з приготування страв для набору розпізнаних та класифікованих продуктів харчування. Система має являти собою проект користувацького веб-додатка з бек-енд сервісом виконання логіки програми. Перелік використовуваних програмних засобів: ОС Microsoft Windows 11, Pycharm 2022.2, Anaconda Navigator, Python 3.9, Tensorflow 2.6, Linux Ubuntu 22.04 LTSC для запуску на сервері, СУБД MongoDB з інтерфейсом MongoDB Atlas, Python Flask Server, React. Технічне забезпечення: IBM-сумісний ПК з 16 Гб ОЗУ або вище. Відеокарта з CUDA-ядрами, процесор з 4 або більше ядрами. Вебкамера. Потрібне технічне забезпечення для запуску на сервері: ARM-сумісний Raspberry Pi 4 або вище. Вебкамера сумісна с Raspberry Pi.*

4. Перелік питань, що потрібно опрацювати в роботі

4.1 Вступ. 4.2 Аналіз предметної області та постановка задачі. 4.3 Порівняння існуючих систем розпізнавання та класифікації продуктів харчування. 4.4 Розробка методів розпізнавання та класифікації продуктів для рекомендаційної системи. 4.4.1 Розробка методів класифікації продуктів з використанням IDE Jupiter Lab. 4.4.2 Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування. 4.4.3 Застосування моделі глибокого навчання для розпізнавання та класифікації продуктів харчування в рекомендаційній системі. 4.5 Аналіз рекомендаційних методів для реалізації функцій системи з приготування страв. 4.5.1 Неперсоналізовані рекомендаційні функції за модифікованим методом спільної фільтрації. 4.5.2 Рекомендаційні функції за базовим методом спільної фільтрації. 4.5.3 Удосконалена формула колаборативної фільтрації з урахуванням середнього значення. 4.5.4 Порівняння зміненої формули Жаккара, базова

формула колаборативної фільтрації (user-based) та удосконалена формула з урахуванням середнього значення. 4.6 Реалізація рекомендаційної системи з приготування страв. 4.6.1 Зберігання даних рекомендаційної системи за допомогою СУБД MongoDb. 4.6.2 Обране технічне рішення для апаратної реалізації системи з приготування страв. 4.6.3 Вибір технологій та фреймворків для серверної частини системи. 4.6.4 Реалізація серверної частини системи з приготування страв. 4.6.5 Огляд структури веб-інтерфейсу клієнтської частини системи. 4.6.6 Проблеми, що виникали під час реалізації системи. 4.7 Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, плакатів)

***5.1** Холодильник LG InstaView ThinQ **5.2** Встановлене віртуальне оточення **5.3** Діаграма роботи методу для копіювання директорій **5.4** Діаграма послідовності роботи завантаження та зміни розміру зображень **5.5** Приклад зображення для тренування **5.6** Діаграма послідовності завантаження тренувальних та тестових даних **5.7** Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування **5.8** Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування (продовження) **5.9** Вивід результатів завантаження **5.10** Класифіковані зображення **5.11** Діаграма активності роботи процесу розпізнання продуктів харчування **5.12** Діаграма послідовності роботи збереження та передбачення класу **5.13** Діаграма роботи функції getObjects() **5.14** Коефіцієнт спільноти. **5.15** Формула індексу Жаккара 5.16 Змінена формула індексу Жаккара. **5.17** Змінена формула індексу Жаккара. **5.18** Формула можливості вподобання. **5.19** Базова формула колаборативної фільтрації (user-based) **5.20** Удосконалена формула колаборативної фільтрації з урахуванням середнього значення **5.21** Вид зберігання об'єктів продуктів 5.22 Приклад зберігаємого зображення 5.23 Зібраний міні-ПК Raspberry Pi 5 з камерою 5.24 Послідовність дій основної функці додатку 5.25 Діаграма послідовності запису продукту в базу даних 5.26 Діаграма послідовності роботи функції generateFoodsDict() 5.27 Діаграма роботи функції sortRecipesByAvabilityDesc(). 5.28 Послідовність процесу виконання рецепту 5.29 Інтерфейс відображення карток рецептів 5.30 Інтерфейс відображення рецепту з фотографіями найстаріших продуктів, потрібних для приготування рецепту*

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів атестаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Отримання завдання атестаційної роботи*	*10.01.2024*	*виконано*
2.	*Аналіз завдання, літератури та аналогів з теми*	*24.03 — 15.04.24*	*виконано*
3.	*Розробка модуля комп'ютерного зору*	*16.04 — 18.04.24*	*виконано*
6.	*Розробка алгоритмів рекомендацій*	*02.05 — 14.05.24*	*виконано*
7.	*Розробка інтерфейсу користувача*	*12.06 — 18.06.24*	*виконано*
8.	*Оформлення п. з., підготовка презентації*	*за 5 днів*	*виконано*
9.	*Представлення роботи на рецензування*	*за 3 дні*	*виконано*
10	*Захист роботи*	*за 2 дні*	*виконано*

Дата видачі завдання __10_ _____01_____20_24_ р.

Студент __Варламов М.Д._____
(підпис)

Керівник роботи _____ ____доцент Коваленко А.І._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота: __82__ стор., __33__ рис., __2__ табл., __26__ джерел, __1__ додаток

БАЗА ДАНИХ, КАМЕРА, МІКРОКОМП’ЮТЕР, РЕЦЕПТИ, MONGODB, PYTHON, RASPBERRY PI, TENSORFLOW, REACT, LAMBDA, PYTHON FLASK REST-API

Об’єктом дослідження роботи є процес розпізнавання продуктів харчування та підбір рецептів для приготування страв з наявних продуктів у місці їх зберігання.

Предметом досліджень є методи ідентифікації продуктів та їхньої класифікації за категоріями, рекомендаційні методи, а також інформаційні технології зі створення рекомендаційної системи з приготування страв для набору розпізнаних та класифікованих продуктів харчування.

Мета дослідження – розробка рекомендаційної системи для приготування страв на основі розпізнаних та класифікованих продуктів харчування. Система має автоматично ідентифікувати продукти за зображеннями, класифікувати їх за категоріями, надавати користувачам персоналізовані рецепти та рекомендації щодо приготування страв, враховуючи їхні вподобання та наявні інгредієнти.

Методи дослідження включають аналіз літератури та існуючих систем, використання методів машинного навчання та комп'ютерного зору для розпізнавання продуктів харчування за допомогою нейронних мереж. Розробка алгоритмів класифікації та бази даних рецептів і продуктів харчування забезпечить ефективний пошук рецептів на основі наявних інгредієнтів. Реалізація прототипу з використанням камер та міні-комп'ютера дозволить ефективно інтегрувати систему у місця зберігання продуктів.

Область застосування – в побуті для інтеграції з "нерозумними" холодильниками та мобільними додатками, в закладах громадського харчування для оптимізації використання інгредієнтів, а також у рітейлі для допомоги покупцям у виборі продуктів та зменшення втрат та харчових відходів внаслідок закінчення термінів придатності. Соціальні проекти можуть використовувати систему для оптимального розподілу харчових продуктів серед нужденних.

ABSTRACT

Qualification work: __82__ p., __33__ pics., __2__ table, __26__ sources, , __1__ application.

DATABASE, CAMERA, MICROCOMPUTER, RECIPES, MONGODB, PYTHON, RASPBERRY PI, TENSORFLOW, REACT, LAMBDA, PYTHON FLASK REST-API

The object of the study is the process of recognizing food products and selecting recipes for cooking meals based on available stored products.

The subject of the research is a recommendation system for cooking dishes using recognized and classified food products.

The aim of the study is the development and implementation of a recommendation system for cooking meals based on recognized and classified food products. The system aims to automatically identify products from images, classify them into categories, provide users with personalized recipes and cooking recommendations tailored to their preferences and available ingredients.

Research methods include literature analysis, study of existing systems, application of machine learning methods and computer vision for food product recognition using neural networks. Developing classification algorithms and a database of recipes and food products will facilitate efficient recipe searching based on available ingredients. Prototyping with cameras and mini-computers will enable effective integration of the system into food storage locations.

Application areas include household integration with "non-smart" refrigerators and mobile applications, public catering establishments for ingredient usage optimization, and retail to assist customers in product selection and reduce losses and food waste due to expiry. Social projects can utilize the system for optimal distribution of food products among those in need.

ЗМІСТ

ВСТУП

У сучасному світі, де темп життя постійно зростає, а час стає одним з найцінніших ресурсів, пошук ефективних та зручних методів швидкого приготування їжі набуває особливої актуальності. З урахуванням індивідуальних уподобань, дієтичних обмежень та наявних продуктів у холодильнику, контролю свіжості продуктів, вибір відповідного рецепту може бути складним завданням. Ця проблема стимулює розвиток рекомендаційних систем для приготування страв, які пропонують оптимальні рішення, враховуючи доступні інгредієнти, час приготування та особисті вподобання.

Усе розглянуте вище обумовлює актуальність теми кваліфікаційної роботи. Робота присвячена аналізу методів та технологій з розробці системи розпізнавання продуктів харчування та підбору рецептів для приготування страв з наявних продуктів у місці їх зберігання (холодильнику), з урахуванням терміну зберігання.

Така рекомендаційна система допомагає користувачам готувати смачні та різноманітні страви з наявних продуктів в холодильнику, враховуючи їхні індивідуальні вподобання та обмеження. Система також вирішує задачу обліку термінів придатності продуктів, для їх ефективного використання та вибору рецептів для приготування їжі.

Об’єктом дослідження роботи є процес розпізнавання продуктів харчування та підбір рецептів для приготування страв з наявних продуктів у місці їх зберігання.

Предметом досліджень є методи ідентифікації продуктів та їхньої класифікації за категоріями, рекомендаційні методи, а також інформаційні технології зі створення рекомендаційної системи з приготування страв для набору розпізнаних та класифікованих продуктів харчування.

Мета дослідження – розробка рекомендаційної системи для приготування страв на основі розпізнаних та класифікованих продуктів харчування. Система має автоматично ідентифікувати продукти за зображеннями, класифікувати їх за категоріями, надавати користувачам персоналізовані рецепти та рекомендації щодо приготування страв, враховуючи їхні вподобання та наявні інгредієнти.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області застосування рекомендаційної системи з приготування страв для набору розпізнаних та класифікованих продуктів харчування

Сучасні технології розпізнавання образів та машинного навчання відкривають нові можливості для створення інноваційних кулінарних додатків. Однією з таких можливостей є розробка рекомендаційної системи, яка допомагає користувачам готувати страви з наявних у них продуктів що були розпізнані 2D камерами вбудованими в холодильник.

Рекомендаційна система з приготування страв може бути корисною для широкого кола користувачів, включаючи:

– домогосподарок. Система може допомогти їм знайти нові рецепти та ідеї для страв з продуктів, які вони вже мають;

– людей, які дотримуються дієти. Система може рекомендувати страви, які відповідають їхнім дієтичним обмеженням;

– людей, які хочуть зменшити харчові відходи. Система може допомогти їм використовувати продукти, які вони вже мають, перш ніж вони зіпсуються.

Розробка рекомендаційної системи з приготування страв вимагає комплексного підходу, який включає наступні напрями.

Розпізнавання та класифікація продуктів харчування. Система повинна вміти розпізнавати та класифікувати продукти харчування за допомогою технологій комп'ютерного зору. Це включає в себе ідентифікацію:

– типу продукту – рукти, овочі, м'ясо, молочні продукти, крупи тощо;

– кількості – система повинна вміти оцінювати кількість продуктів, наприклад, за допомогою розпізнавання розміру або ваги;

– стану – система повинна вміти визначати, чи є продукт свіжим, зіпсованим або придатним до вживання.

База даних рецептів. Система повинна мати доступ до бази даних рецептів, яка містить інформацію про:

– інгредієнти – список інгредієнтів з їх кількістю та одиницями вимірювання;

– інструкції з приготування – покрокові інструкції з приготування страви;

– метадані – час приготування, складність, країна походження, тип страви (наприклад, сніданок, обід, вечеря), дієтичні обмеження тощо.

Алгоритми рекомендацій. Система повинна використовувати алгоритми машинного навчання, щоб рекомендувати страви на основі:

– наявних продуктів – система повинна враховувати типи, кількість та стан продуктів, які є у користувача;

– вподобань користувача – система може враховувати попередні вибори користувача, його дієтичні обмеження, алергії та інші фактори;

– популярності рецептів – истема може враховувати популярність рецептів серед інших користувачів;

– складності приготування – система може враховувати рівень кулінарних навичок користувача.

Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє:

– вводити інформацію про наявні продукти – це можна зробити вручну, за допомогою розпізнавання образів або сканування штрих-кодів;

– отримувати рекомендації щодо страв: рекомендації повинні бути чіткими, детальними та містити інформацію про інгредієнти, інструкції з приготування та час приготування;

– фільтрувати та сортувати рекомендації – користувач повинен мати можливість фільтрувати та сортувати рекомендації за різними параметрами, такими як час приготування, складність, дієтичні обмеження тощо.

Розробка рекомендаційної системи з приготування страв пов'язана з певними викликами та такими обмеженнями.

Точність розпізнавання продуктів:

– технології розпізнавання образів постійно вдосконалюються, але вони все ще не є ідеальними;

– система повинна враховувати можливі помилки розпізнавання.

Якість та повнота бази даних рецептів:

– база даних рецептів повинна бути великою, різноманітною та містити точну інформацію;

– необхідно постійно оновлювати базу даних новими рецептами та інформацією.

Складність алгоритмів рекомендацій:

– алгоритми машинного навчання можуть бути складними для розробки та налаштування.

– необхідно знайти баланс між точністю рекомендацій та обчислювальними ресурсами.

Індивідуальні вподобання:

– смаки та вподобання людей дуже різняться.

– системі може бути важко рекомендувати страви, які сподобаються всім користувачам.

Розробка рекомендаційної системи з приготування страв є складною, але перспективною задачею. Така система може бути корисною для широкого кола користувачів та має потенціал для покращення кулінарного досвіду.

1.2 Порівняння існуючих систем розпізнавання та класифікації продуктів харчування

На ринку існує кілька систем розумних холодильників, які пропонують різні функції. Розглянемо деякі з найбільш популярних систем.

Система Samsung Family Hub:

– має великий сенсорний екран, який можна використовувати для керування холодильником, перегляду рецептів, перевірки погоди тощо.

– Family Hub має вбудовані камери, які дозволяють користувачам переглядати вміст холодильника віддалено.

– Family Hub пропонує систему рекомендацій рецептів яка враховує вміст холодильника та уподобання користувача. Але вміст продуктів потрібно вносити власноруч.

Система LG InstaView ThinQ має скляну панель, яка дозволяє користувачам переглядати вміст холодильника, не відкриваючи дверцята. Система InstaView ThinQ має вбудовані камери, які дозволяють користувачам переглядати вміст холодильника віддалено.

InstaView ThinQ пропонує систему рекомендацій рецептів, яка враховує вміст холодильника та уподобання користувача. Але вміст продуктів потрібно вносити власноруч.

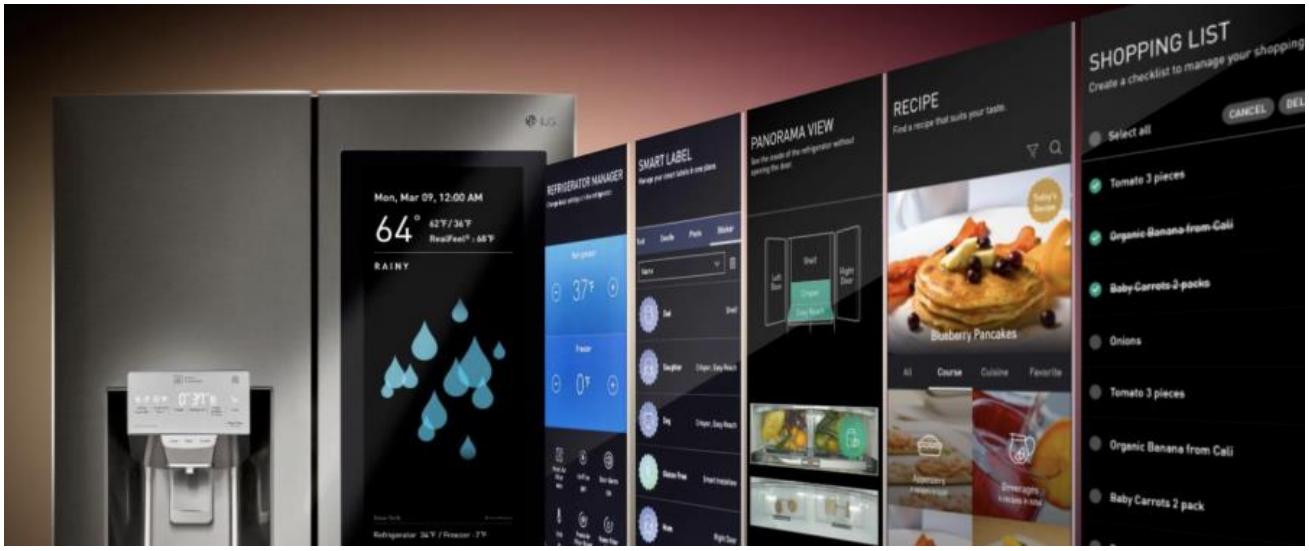

Рисунок 1.2 – Холодильник LG InstaView ThinQ

Система Bosch Home Connect

– дозволяє користувачам керувати холодильником віддалено за допомогою смартфону;

– connect не має вбудованих камер, не пропонує систему рекомендацій рецептів.

Whirlpool Smart Appliances:

– дозволяє користувачам керувати холодильником віддалено за допомогою смартфону;

– whirlpool Smart Appliances не має вбудованих камер;

– whirlpool Smart Appliances не пропонує систему рекомендацій рецептів.

З цього можна зробити такі висновки.

Вбудовані камери:

– не всі розумні холодильники мають вбудовані камери;

– на даний момент цю функцію пропонують лише Samsung Family Hub та LG InstaView ThinQ.

Система рекомендацій рецептів:

– не всі розумні холодильники пропонують систему рекомендацій рецептів. А ті що пропонують, потребують мануального вводу продуктів харчування;

– на даний момент цю функцію пропонують лише Samsung Family Hub та LG InstaView ThinQ.

1.3 Постановка задачі на проєктування та розробку рекомендаційної системи з приготування страв

Потрібно розробити рекомендаційну систему, яка допомагає користувачам готувати страви з наявних продуктів, враховуючи їхні індивідуальні уподобання та обмеження. Система повинна розпізнавати та класифікувати продукти харчування з високою точністю, рекомендувати страви, які відповідають наявним продуктам, вподобанням користувача та іншим заданим критеріям, а також мати зручний та інтуїтивно зрозумілий інтерфейс користувача.

Вимоги до системи складаються з наступних пунктів:

– система повинна розпізнавати та класифікувати продукти харчування з високою точністю.

– система повинна рекомендувати страви, які відповідають наявним продуктами, вподобанням користувача та іншим заданим критеріям.

– інтерфейс користувача повинен бути зручним та інтуїтивно зрозумілим

– система повинна бути масштабованою.

За проведеним аналізом та обраною предметною областю потрібно:

– порівняти існуючі системи розпізнавання та класифікації продуктів харчування;

– сформулювати задачі на проєктування та розробку рекомендаційної системи з приготування страв;

– проаналізувати предметну область застосування рекомендаційної системи з приготування страв для набору розпізнаних та класифікованих продуктів харчування.

– розробити код методів розпізнавання та класифікації продуктів для рекомендаційної системи з приготування страв;

– розробити код методів класифікації продуктів з використанням IDE Jupiter Lab;

– створити модель глибокого навчання для розпізнавання та класифікації продуктів харчування;

– застосувати модель глибокого навчання для розпізнавання та класифікації продуктів харчування в рекомендаційній системі;

– застосувати модель для класифікації продуктів харчування в рекомендаційній системі;

– розробити і використати функції для захоплення і розпізнавання зображень.

Проаналізувати рекомендаціййні методи для реалізації функцій системи з приготування страв з таких питань:

– розглянути можливість використання неперсоналізованих рекомендацій, що реалізуються за модифікованим методом спільної фільтрації;

– розглянути можливість використання рекомендацій, що реалізуються за базовим методом спільної фільтрації;

– розглянути можливість використання удосконаленої формули колаборативної фільтрації з урахуванням середнього значення;

– порівняти змінену формулу Жаккара, базову формулу колаборативної фільтрації (user-based) та удосконалену формулу з урахуванням середнього значення.

Реалізувати рекомендаційну систему з приготування страв та визначити:

– можливість зберігання даних рекомендаційної системи з приготування страв за допомогою СУБД MongoDb;

– технічне рішення для апаратної реалізації рекомендаційної системи з приготування страв;

– технології та фреймворки для реалізації серверної частини рекомендаційної системи з приготування страв;

– архитектуру реалізації серверної частини рекомендаційної системи з приготування страв;

– варіант реалізації веб-інтерфейсу клієнтської частини рекомендаційної системи з приготування страв;

– визначити проблеми, що можуть виникнути під час реалізації рекомендаційної системи з приготування страв.

2 ВИЗНАЧЕННЯ І РОЗРОБКА МЕТОДІВ РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЇ ПРОДУКТІВ ДЛЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ З ПРИГОТУВАННЯ СТРАВ

2.1 Визначення і розробка коду методів класифікації продуктів з використанням IDE Jupiter Lab

У рамках роботи, яка присвячена створенню рекомендаційної системи для приготування страв на основі розпізнаних та класифікованих продуктів харчування, важливо забезпечити відтворюваність експериментів та стабільність роботи системи. Використання requirements.txt для управління залежностями допомагає досягти цього, оскільки:

– відтворюваність – інші дослідники або розробники можуть легко встановити ті самі версії бібліотек, що були використані в оригінальному проєкті;

– сумісність – вказівка конкретних версій бібліотек запобігає конфліктам між різними версіями пакетів;

– простота налаштування: Автоматизація процесу встановлення залежностей робить налаштування середовища швидким і простим.

Включення команди pip install -r requirements.txt у проект забезпечує легке встановлення всіх необхідних залежностей для коректної роботи системи розпізнавання та класифікації продуктів харчування. Це є важливим кроком для створення надійної рекомендаційної системи, яка може бути використана як в наукових дослідженнях, так і в практичних застосуваннях;

Для старту Jupiter Notebook потрібно встановити Anaconda Navigator. В навігаторі потрібно встановити віртуальне оточення пайтон з файлу конфігурації.

Спочатку, вся програма була натренована на готовому наборі зображень блюд з відкритої баз food101 яка має близько 75000 фотографій їжі. Далі, модель була перетренована для розпізнавання конкретно продуктів харчування, таких як помідор, огірок, тощо.

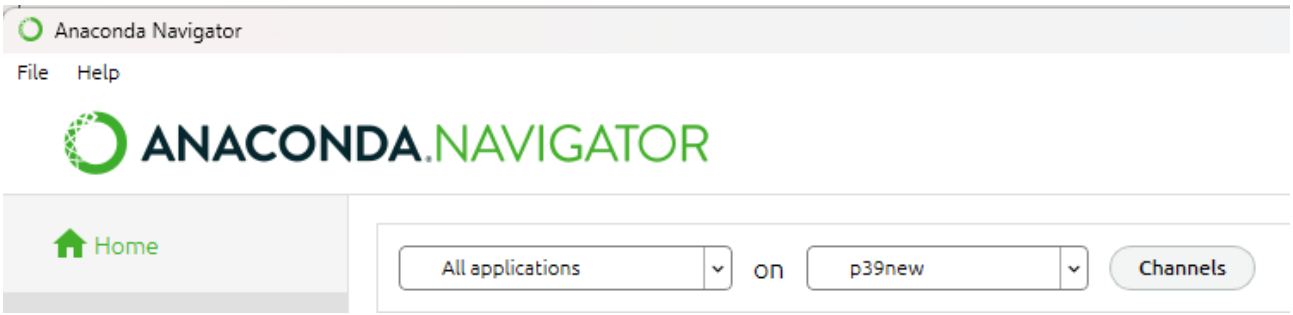

Рисунок 2.1 – Встановлене віртуальне оточення

Фрагмент коду (лістинг 2.1) генерує необхідні словники та методи для обробки зображень із набору даних Food-101. Це спрощує процес завантаження, підготовки та обробки зображень для моделі машинного навчання.

Лістинг 2.1 – Програмний код процесу завантаження та обробки зображень для моделі

```
class_N = {}
N_class = {}
with open('food-101/meta/classes.txt', 'r') as txt:
    classes = [i.strip() for i in txt.readlines()]
    class_N = dict(zip(classes, range(len(classes))))
    N_class = dict(zip(range(len(classes)), classes))
    class_N = {i: j for j, i in N_class.items()}
class_N_sorted = collections.OrderedDict(sorted(class_N.items()))
print(class_N)
```

Пояснення до лістингу 2.1:

– class_N та N_class – словники, що зіставляють назви класів (категорій їжі) з їхніми індексами і навпаки;

– завантаження класів. Імена класів завантажуються з файлу classes.txt, потім створюються відповідні словники для швидкого доступу;

– сортування класів. Словник class_N сортується для зручного перегляду.

Метод для генерації мапи файлів подається в лістингу 2.2.

Лістинг 2.2 – Метод для генерації мапи файлів

```
def gen_dir_file_map(path):
    dir_files = defaultdict(list)
    with open(path, 'r') as txt:
        files = [i.strip() for i in txt.readlines()]
```

```
        for f in files:
            dir_name, id = f.split('/')
            dir_files[dir_name].append(id + '.jpg')
    return dir_files
```

gen_dir_file_map – створює мапу, де ключами є назви директорій (категорій їжі), а значеннями – списки файлів зображень.

Метод для для копіювання директорій подається в лістингу 2.3.

Лістинг 2.3 – Метод для копіювання директорій:

```
def copytree(source, target, symlinks = False, ignore = None):
    if not os.path.exists(target):
        os.makedirs(target)
        shutil.copystat(source, target)
    data = os.listdir(source)
    if ignore:
        exclude = ignore(source, data)
        data = [x for x in data if x not in exclude]
    for item in data:
        src = os.path.join(source, item)
        dest = os.path.join(target, item)
        if symlinks and os.path.islink(src):
            if os.path.lexists(dest):
                os.remove(dest)
            os.symlink(os.readlink(src), dest)
            try:
                st = os.lstat(src)
                mode = stat.S_IMODE(st.st_mode)
                os.lchmod(dest, mode)
            except:
                pass
        elif os.path.isdir(src):
            copytree(src, dest, symlinks, ignore)
        else:
            shutil.copy2(src, dest)
```

copytree – рекурсивно копіює вміст директорій із джерела в цільову директорію, з можливістю ігнорувати певні файли або директорії.

Діаграма роботи методу для копіювання директорій подається на рис. 2.2.

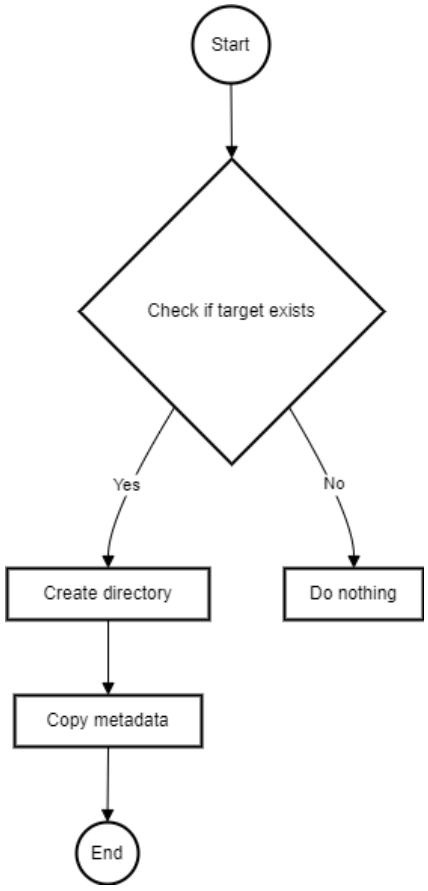

Рисунок 2.2 – Діаграма роботи методу для копіювання директорій

Метод ігнорування файлів для тренування і тестування, якщо вони існують, подається в лістингу 2.4.

Лістинг 2.4 – Ігнорування файлів для тренування і тестування, якщо вони існують

```
    def ignore_train(d, filenames):
        subdir = d.split('/')[-1]
        train_dir_files = gen_dir_file_map('food-101/meta/train.txt')
        to_ignore = train_dir_files[subdir]
        return to_ignore

    def ignore_test(d, filenames):
        subdir = d.split('/')[-1]
test_dir_files = gen_dir_file_map('food-101/meta/test.txt')
to_ignore = test_dir_files[subdir]
return to_ignore
```

ignore_train та ignore_test – Визначають файли, які слід ігнорувати під час копіювання тренувальних та тестових даних відповідно.

Метод завантаження та зміни розміру зображень подається в лістингу 2.4.

Лістинг 2.5 – Завантаження та зміна розміру зображень:

```
    resize_count = 0
    invalid_count = 0
    all_imgs = []
    all_classes = []
    for i, subdir in enumerate(listdir(path_to_imgs)):
        imgs = listdir(join(path_to_imgs, subdir))
        classN = class_N[subdir]
        for img_name in imgs:
            img_arr = cv2.imread(join(path_to_imgs, subdir, img_name))
            img_arr_rs = img_arr
            img_arr_rs        =        cv2.resize(img_arr,        (200,200),
interpolation=cv2.INTER_AREA)
            resize_count += 1
            im_rgb = cv2.cvtColor(img_arr_rs, cv2.COLOR_BGR2RGB)
            all_imgs.append(im_rgb)
            all_classes.append(classN)
    return np.array(all_imgs), np.array(all_classes)
load_images: Завантажує зображення з зазначеної директорії, змінює їх розмір
до 200x200 пікселів, конвертує в формат RGB та зберігає в масиви.
```

Діаграма послідовності роботи завантаження та зміни розміру зображень подається на рис. 2.3

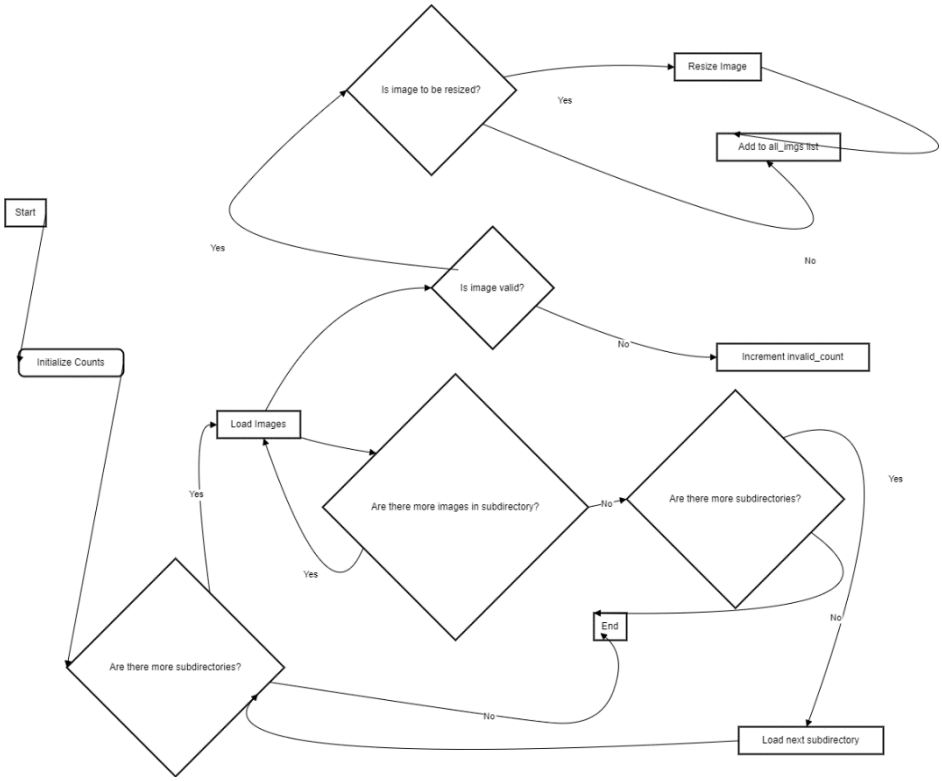

Рисунок 2.3 – Діаграма послідовності роботи завантаження та зміни розміру зображень

Метод генерація тренувальних і тестових директорій подається в лістингу 2.6.

Лістинг 2.6 – Генерація тренувальних і тестових директорій

```
def  gen_train_test_split(path_to_imgs='food-101/images',  target_path='food-101'):
    copytree(path_to_imgs, target_path + '/train', ignore=ignore_test)
    copytree(path_to_imgs, target_path + '/test', ignore=ignore_train)
gen_train_test_split: Генерує тренувальні та тестові набори даних, копіюючи відповідні файли в окремі директорії.
```

Код процесу завантаження тренувальних і тестових даних:

```
def load_train_test_data(path_to_train_imgs, path_to_test_imgs):
X_train, y_train = load_images(path_to_train_imgs)
X_test, y_test = load_images(path_to_test_imgs)
return X_train, y_train, X_test, y_test
load_train_test_data: Завантажує зображення та їхні класи для тренувального і тестового наборів даних.
```

Ця клітинка коду спрощує підготовку даних для моделі розпізнавання їжі, забезпечуючи ефективну організацію файлів, завантаження зображень, їх обробку та створення словників для класифікації.

Результат виконання клітинки:

```
{'potato': 0, 'tomato': 1, 'cucumber': 2}.
```

Цей результат показує відображення індексів класів для деяких категорій їжі. У цьому словнику ключами є назви класів (у цьому випадку це "potato", "tomato" і "cucumber"), а значеннями є відповідні числові індекси. Наприклад, "potato" має індекс 0, "tomato" - 1, а "cucumber" - 2.

Це корисно для подальшої обробки даних, такої як класифікація зображень: замість роботи з назвами класів, які можуть бути неоднозначними або складними для обробки, можна використовувати числові індекси, що спрощує роботу з даними.

Метод генерування тестових та тренувальних файлів подається в лістингу 2.7.

Лістинг 2.7 – Генерування тестових та тренувальних файлів

```
if  not  os.path.isdir('./food-101/test')  and  not  os.path.isdir('./food-101/train'):
    gen_train_test_split()
```

```
    len_train = len(os.listdir('./food-101/train'))
    len_test = len(os.listdir('./food-101/test'))
    print(len_train,len_test)
else:
    print('train and test folders already exists.')
    len_train = len(os.listdir('./food-101/train'))
    len_test = len(os.listdir('./food-101/test'))
    print(len_train,len_test)
```

Цей метод (лістинг 2.7) перевіряє наявність папок для тренувальних та тестових даних. Якщо ці папки не існують, вони генеруються за допомогою функції gen_train_test_split(), що забезпечує необхідну структуру каталогів для подальшої обробки даних. Метод дозволяє виводити кількість файлів у папках тренування та тестування, що надає інформацію про розподіл даних між цими двома наборами. У випадку, якщо папки вже існують, код просто повідомляє про їх існування і виводить кількість файлів у папках тренування та тестування.

Код генерування списоку усіх класів їжі забезпечує організацію та виведення алфавітно впорядкованого списку файлів та папок у заданій директорії:

```
foods_sorted = sorted(os.listdir('food-101/images'))
print('ok')
foods_sorted
```

Результат виконання цього коду: `['cucumber', 'potato', 'tomato'].`

Приклад зображення для тренування подається на рис. 2.4.

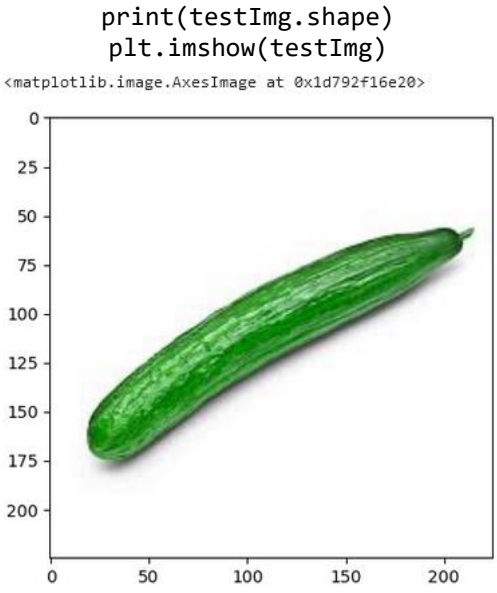

Рисунок 2.4 – Приклад зображення для тренування

Наступна клітинка виконує операцію завантаження тренувальних та тестових даних із вказаних шляхів та зберігає їх у змінних для подальшого використання в аналізі та навчанні моделей.

```
X_train, y_train, X_test, y_test = load_train_test_data('./food-
                    101/train','./food-101/test')
```

Вивід: `Fitting the data on the Inception-v3 model.`

Діаграма послідовності завантаження тренувальних та тестових даних подається на рис. 2.5.

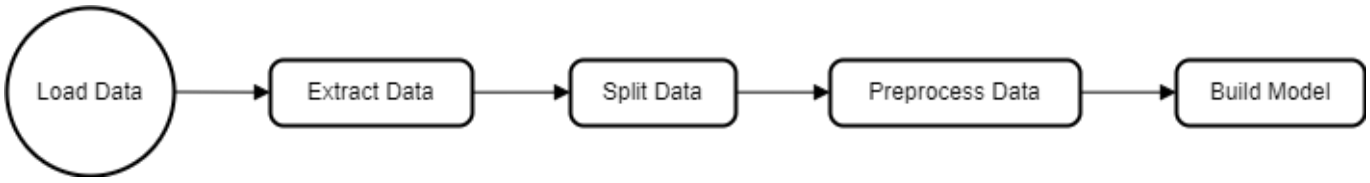

Рисунок 2.5 – Діаграма послідовності завантаження тренувальних та тестових даних

За діагрмою (рис. 2.5) викликається функція load_train_test_data, передаючи шляхи до тренувальних та тестових зображень як аргументи. Функція load_train_test_data завантажує дані зображень та відповідні мітки класів з вказаних шляхів. Після завершення виконання функції, результати завантаження, тобто зображення та відповідні мітки для навчання та тестування, призначаються змінним X_train, y_train, X_test та y_test.

2.2 Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування

Проведено створення та налаштування моделі глибинного навчання для розробки рекомендаційної системи з приготування страв на основі розпізнаних та класифікованих продуктів харчування.

Перша частина коду створенної моделі глибокого навчання для розпізнавання та класифікації продуктів харчування подається в лістингу 2.8.

Лістинг 2.8 – Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування

```
//Очищення поточної сесії Keras, щоб уникнути проблем з пам'яттю та
конфліктів в моделях
K.clear_session()
//Визначення параметрів моделі та даних
n_classes = 3  # Кількість класів для класифікації
batch_size = 16  # Розмір батча для навчання
width, height = 200, 200  # Розміри зображень (ширина і висота)
train_data = './food-101/train'  # Шлях до тренувальних даних
test_data = './food-101/test'  # Шлях до тестових даних
train_samples = 35  # Кількість тренувальних зразків
test_samples = 35  # Кількість тестових зразків
//Створення генератора даних для навчання з різними перетвореннями зображень
train_data_gen = ImageDataGenerator(
    rescale=1. / 255,  # Масштабування значень пікселів від 0 до 1
    shear_range=0.2,  # Застосування зсуву
    zoom_range=0.2,  # Застосування масштабування
    horizontal_flip=True  # Застосування горизонтального відображення
)
//Створення генератора даних для тестування з масштабуванням зображень
test_data_gen = ImageDataGenerator(rescale=1. / 255)
//Створення генератора зображень з директорії для навчання
train_gen = train_data_gen.flow_from_directory(
    train_data,  # Шлях до директорії з тренувальними даними
    target_size=(height, width),  # Розміри зображень
    batch_size=batch_size,  # Розмір батча
    class_mode='categorical'  # Режим класів (категоріальна класифікація)
)
//Створення генератора зображень з директорії для тестування
test_gen = test_data_gen.flow_from_directory(
    test_data,  # Шлях до директорії з тестовими даними
    target_size=(height, width),  # Розміри зображень
    batch_size=batch_size,  # Розмір батча
    class_mode='categorical'  # Режим класів (категоріальна класифікація)
)
```

Порядок створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування подвється на рис. 2.6.

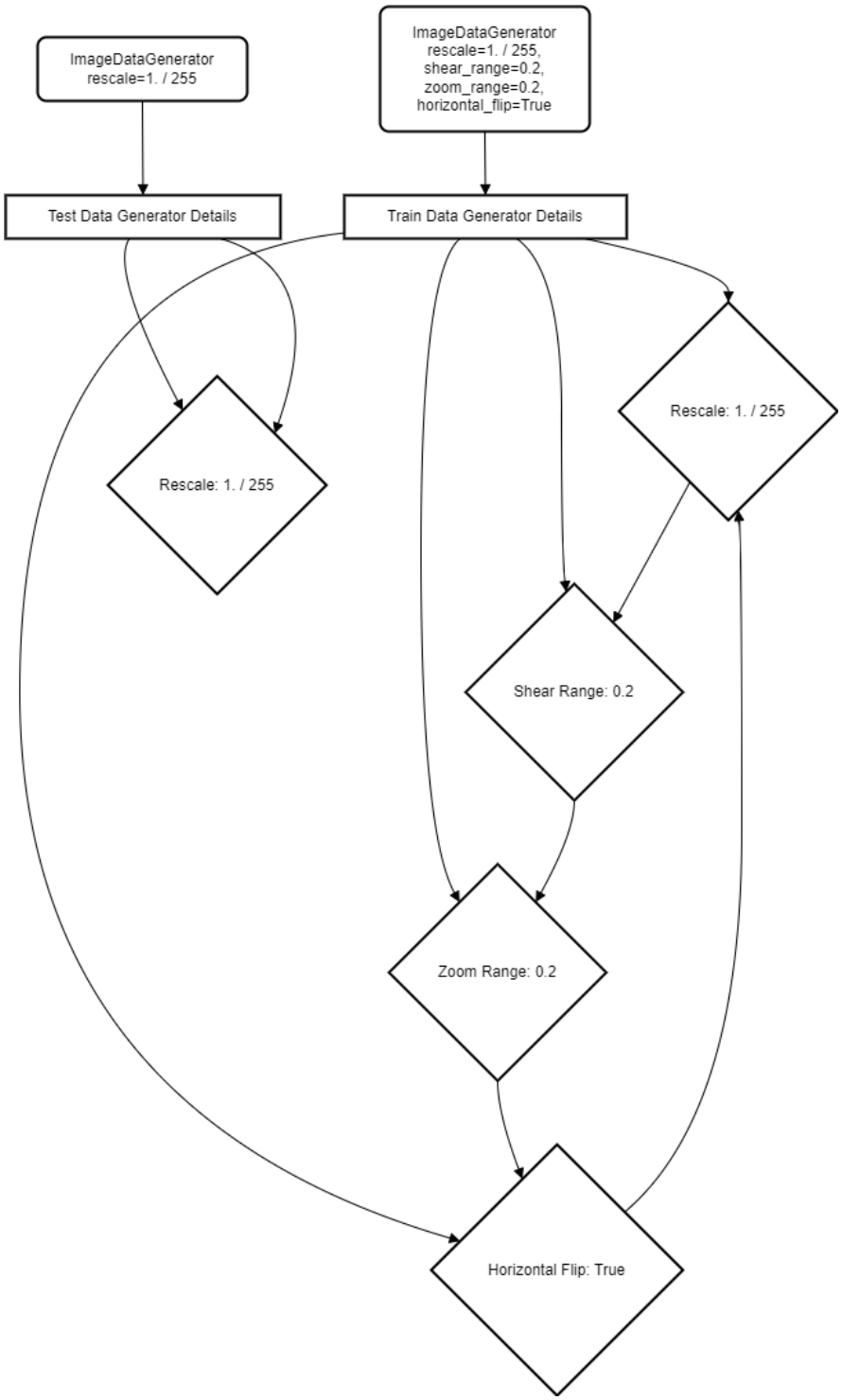

Рисунок 2.6 – Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування

Друга частина коду створенної моделі глибокого навчання для розпізнавання та класифікації продуктів харчування подається в лістингу 2.9.

Лістинг 2.9 – Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування

```
//Завантаження попередньо навченої моделі InceptionV3 без верхнього шару (fully connected)
inception = InceptionV3(weights='imagenet', include_top=False)
//Створення додаткових шарів для моделі
layer = inception.output
layer = GlobalAveragePooling2D()(layer)  # Додавання глобального середнього пулінгу
layer = Dense(128, activation='relu')(layer)  # Додавання повнозв'язного шару з активацією ReLU
layer = Dropout(0.2)(layer)  # Додавання Dropout для зменшення перенавчання
//Додавання вихідного шару з кількістю класів і активацією softmax
predictions = Dense(
    n_classes,
    kernel_regularizer=regularizers.l2(0.005),  # Додавання регуляризації L2
    activation='softmax'
)(layer)
//Створення повної моделі
model = Model(inputs=inception.input, outputs=predictions)

//Компіляція моделі з оптимізатором SGD, функцією втрат і метрикою
model.compile(
    optimizer=SGD(lr=0.0001, momentum=0.9),   # Оптимізатор SGD з низькою швидкістю навчання та моментумом
    loss='categorical_crossentropy',  # Функція втрат
    metrics=['accuracy']  # Метрика точності
)
//Визначення колбеків для збереження найкращої моделі та логування історії навчання
checkpointer      =      ModelCheckpoint(filepath='best_model_101class.hdf5', save_best_only=True)
csv_logger = CSVLogger('history_101class.log')

//Навчання моделі з генератором даних
history_101class = model.fit(
    train_gen,  # Генератор тренувальних даних
    steps_per_epoch=train_samples // batch_size,  # Кількість кроків на епоху
    validation_data=test_gen,  # Генератор тестових даних
    validation_steps=test_samples  //  batch_size,   # Кількість кроків для валідації
    epochs=30,  # Кількість епох навчання
callbacks=[csv_logger, checkpointer]  # Колбеки для логування та збереження моделі
)
```

```
Збереження навченої моделі
model.save('model_trained_101class.hdf5')
```

Порядок створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування подвється на рис. 2.7.

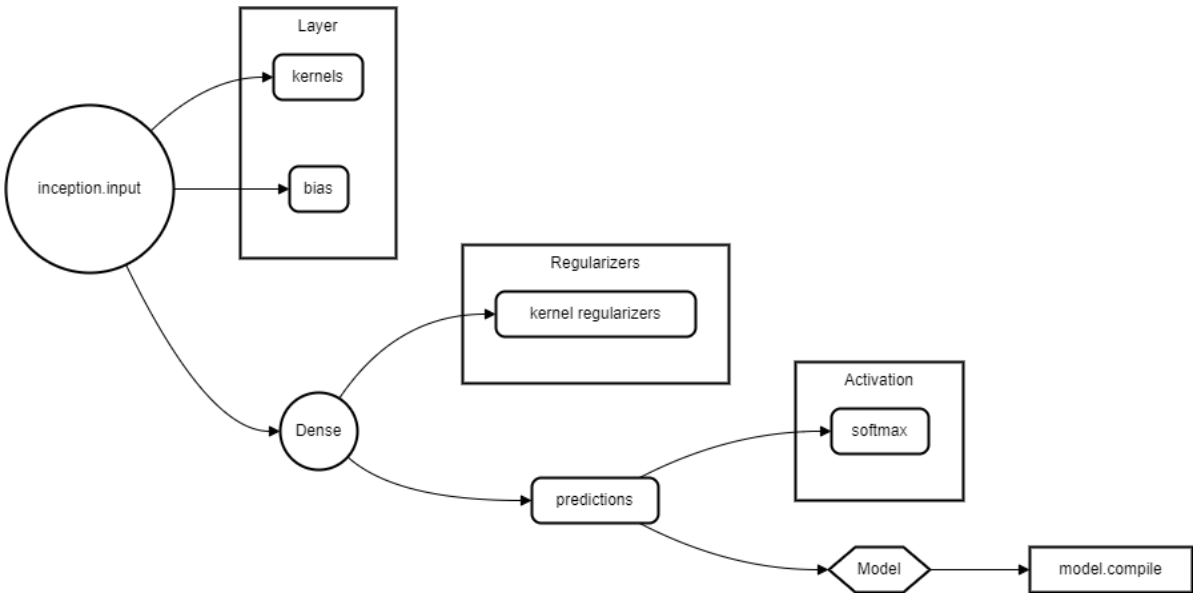

Рисунок 2.7 – Створення моделі глибокого навчання для розпізнавання та класифікації продуктів харчування (продовження)

Даний процес (рис. 2.7) можна розділити на кілька етапів:

Етап 1. Очищення сесії та налаштування параметрів моделі. Починаючи з очищення сесії TensorFlow/Keras для звільнення пам'яті та попередніх обчислень, було встановлено параметри моделі, такі як кількість класів, розмір пакету та розмір зображень.

Етап 2. Підготовка генераторів даних. Застосовано генератори даних для підготовки тренувальних та тестових зображень. Додаткові перетворення були застосовані для підвищення різноманітності та покращення ефективності навчання моделі. Налаштування параметрів моделі:

– n_classes: Кількість класів або категорій продуктів харчування (у цьому випадку 3);

– batch_size: Розмір пакету для навчання моделі;

– width та height: Розмір зображення (ширина та висота);

– train_data та test_data: Шляхи до тренувальних та тестових даних;

– train_samples та test_samples: Кількість прикладів тренувальних та тестових даних.

Підготовка генераторів даних:

– створюється генератор даних для тренувальних та тестових зображень з використанням ImageDataGenerator;

– генератори обробляють дані, масштабуючи їх значення до діапазону [0, 1] та застосовуючи додаткові перетворення (збільшення, обрізання тощо) для підвищення різноманітності.

Етап 3. Створення та конфігурація моделі. Використовуючи архітектуру InceptionV3, була створена модель для витягування ознак зображень. Додавалися глобальний шар пулінгу та повнозв'язний шар для класифікації, а також використання функції активації softmax для отримання ймовірностей належності до кожного класу.

Етап 4. Компіляція та навчання моделі. Модель була скомпільована з використанням оптимізатора SGD та функції втрат категоріальної хрест-ентропії. Для навчання моделі використовувалися тренувальні та тестові генератори даних, а також використовувалися callback-функції для збереження найкращої моделі та запису історії навчання. Викликається метод fit для навчання моделі з використанням тренувальних та тестових генераторів даних. Використовуються callback-функцій для збереження найкращої моделі та запису історії навчання.

Етап 5. Збереження навченої моделі. Після завершення навчання модель була збережена у файл model_trained_101class.hdf5 для подальшого використання в рекомендаційній системі з приготування страв.

Процес створення моделі глибинного навчання забезпечує необхідність та ефективність для розробки рекомендаційної системи, яка забезпечить користувачам персоналізовані рекомендації щодо приготування страв на основі їхніх вподобань та доступних інгредієнтів.

Послідовний опис коду:

```
//Звільнення пам'яті та очищення поточної сесії TensorFlow/Keras:
tensorflow.keras.backend.clear_session()
```

```
//Визначення параметрів моделі та набору даних:
n_classes = 3
batch_size = 16
width, height = 200, 200
train_data = './food-101/train'
test_data = './food-101/test'
train_samples = 35
test_samples = 35
```

Етап 6. Створення генераторів даних для тренувального та тестового наборів. Програма створює спеціальні об'єкти, які дозволяють нам робити певні операції з даними, такі як зміна розміру зображень, масштабування їх значень, а також здійснення додаткових операцій для покращення якості та різноманітності навчальних даних:

– ImageDataGenerator – цей клас дозволяє нам автоматично створювати партії даних зображень з вказаними налаштуваннями аугментації та попередньої обробки;

– rescale=1./255: Ця опція встановлює коефіцієнт масштабування для зображень, щоб значення пікселів перебували в діапазоні від 0 до 1. Це допомагає покращити стабільність та швидкість навчання моделі;

– shear_range=0.2, zoom_range=0.2, horizontal_flip=True – ці параметри встановлюють додаткові операції аугментації зображень, такі як обертання, збільшення та горизонтальне відображення, відповідно. Це допомагає моделі навчатися на більш різноманітних та реалістичних даних, що зазвичай призводить до кращих результатів;

– flow_from_directory – цей метод генератора даних створює партії зображень з директорії, вказаної в аргументі train_data або test_data. Він автоматично зчитує та обробляє зображення з цієї директорії та їхні мітки класів. Аргумент target_size встановлює розмір, до якого будуть змінені всі зображення, а batch_size визначає кількість зображень у кожній партії даних.

Цей підхід дозволяє нам автоматизувати обробку та підготовку даних для тренування моделі, що допомагає покращити якість та ефективність навчання.

Етап 7. Створення та налаштування моделі глибинного навчання. Програма створює та налаштовує модель глибинного навчання, використовуючи архітектуру InceptionV3:

– `inception = InceptionV3(weights='imagenet', include_top=False)`. Створюємо модель InceptionV3 без верхніх (повністю з'єднаних) шарів із завантаженими вагами, навченими на наборі даних ImageNet;

– `layer = inception.output`. Отримуємо вихідний шар (тензор) моделі InceptionV3;

– `layer = GlobalAveragePooling2D()(layer)`. Додаємо глобальний шар пулінгу для зменшення розмірності даних та отримання важливих ознак зображень;

– `layer = Dense(128,activation='relu')(layer)`. Додаємо повнозв'язний шар з 128 нейронами та функцією активації ReLU для вилучення корисних ознак;

– `layer = Dropout(0.2)(layer)`. Додаємо шар Dropout для регуляризації та запобігання перенавчанню;

– `predictions = Dense(n_classes,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(layer)`. Додаємо повнозв'язний шар з функцією активації softmax для класифікації зображень на основі передбачених ймовірностей для кожного класу. `n_classes` визначає кількість класів, а `regularizers.l2(0.005)` використовується для регуляризації моделі.

– `model = Model(inputs = inception.input, outputs=predictions)`. Створюємо модель, вказуючи вхідний тензор (вхідні зображення) та вихідний тензор (передбачені ймовірності класів);

– `model.compile(optimizer=SGD(lr=0.0001,momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])`: Компілюємо модель, вказуючи оптимізатор, функцію втрат та метрики для оцінки її ефективності під час навчання.

Архітектура InceptionV3 є продуктом досліджень у галузі глибинного навчання та комп'ютерного зору. Вона базується на ідеях глибоких нейронних мереж, що використовуються для автоматичного витягування корисних ознак зображень. Ця архітектура використовує концепцію модулів, які дозволяють ефективно аналізувати зображення на різних рівнях абстракції.

Інцепція, як ключовий компонент InceptionV3, представляє собою набір фільтрів з різними розмірами ядер, які одночасно застосовуються до вхідного зображення. Це дозволяє моделі виявляти ознаки з різних масштабів і розмірів. Крім того, InceptionV3 використовує пулінг з усередненням на останньому шарі, щоб зменшити розмір зображення перед передачею його до повнозв'язного шару.

Ще однією важливою особливістю InceptionV3 є використання попередньо навчених ваг, отриманих на великому наборі даних ImageNet. Ці ваги містять корисні патерни, які модель вивчила під час навчання на великому обсязі даних. Використання попередньо навчених ваг дозволяє використовувати здобуті знання для розв'язання нових завдань класифікації зображень.

Крім того, InceptionV3 є ефективним інструментом для трансферного навчання. Це означає, що модель може бути використана для вирішення конкретного завдання класифікації, але з можливістю адаптації до нових даних безпосередньо під час навчання. Такий підхід дозволяє досягти високої ефективності навчання навіть при обмеженому обсязі даних.

Обираючи архітектуру InceptionV3, ми можемо скористатися передовими досягненнями в галузі глибинного навчання та використати їх для успішного розв'язання завдань класифікації зображень у нашому дослідженні.

Етап 8. Навчання моделі та збереження найкращої моделі. Проводилося навчання на 35 зображеннях. На відеокарті GTX 1080 Ti з 11 Гб відеопам’яті. Проходження кожної епохи зайняло приблизно 1 хвилину.

Розглянемо кожну функцію:

```
checkpointer = ModelCheckpoint(filepath='best_model_101class.hdf5',
                               save_best_only=True)
```

Цей рядок створює об'єкт ModelCheckpoint, який дозволяє автоматично зберігати найкращу версію моделі під час навчання. Аргумент filepath вказує шлях для збереження моделі, а save_best_only=True означає, що буде збережена лише модель з найкращою точністю на валідаційному наборі даних.

Функція `csv_logger = CSVLogger('history_101class.log')` створює об'єкт CSVLogger, який дозволяє записувати історію метрик моделі в текстовий файл у форматі CSV. Це допомагає візуалізувати та аналізувати процес навчання після завершення.

Код, який починає процес навчання моделі:

```
history_101class = model.fit(train_gen, steps_per_epoch= train_samples
// batch_size, validation_data= test_gen, validation_steps= test_samples //
batch_size, epochs=30, callbacks=[csv_logger, checkpointer])
```

Виклик методу fit об'єкта моделі (model) запускає цикл навчання. Аргументи методу визначають дані для навчання та валідації, кількість епох навчання та зворотні виклики (callbacks), такі як збереження найкращої моделі та журналювання історії метрик.

Ці рядки показують кількість зображень, які були знайдені для навчання та валідації відповідно. У цьому випадку є 35 зображень, які належать до трьох різних класів. Вивід:

```
Found 35 images belonging to 3 classes.
Found 35 images belonging to 3 classes.
Epoch 1/30
2/2 [==============================] - 55s 6s/step - loss: 1.3484 - accuracy: 0.1875 - val_loss: 1.7422 - val_accuracy: 0.2188
Epoch 2/30
2/2 [==============================] - 14s 176ms/step - loss: 1.2562 - accuracy: 0.2632 - val_loss: 1.7392 - val_accuracy: 0.1875
…
…
…
Epoch 30/30
2/2 [==============================] - 1s 1s/step - loss: 0.6511 - accuracy: 0.7895 - val_loss: 0.6957 - val_accuracy: 0.7188
Found 35 images belonging to 3 classes. та Found 35 images belonging to 3 classes.
```

Кожен рядок представляє одну ітерацію навчання моделі, так звану «епоху». Це означає, що модель пройшла через весь навчальний набір даних один раз. У цьому випадку, модель навчалась протягом 30 епох:

```
Epoch 1/30, Epoch 2/30, ..., Epoch 30/30:
2/2 [==============================] - 55s 6s/step - loss: 1.3484 - accuracy: 0.1875 - val_loss: 1.7422 - val_accuracy: 0.2188:
```

Цей рядок показує метрики моделі після кожної епохи. loss відображає значення функції втрат на навчальних даних, а accuracy показує точність моделі на навчальних даних. Аналогічно, val_loss та val_accuracy відображають відповідно значення функції втрат та точність на валідаційних даних.

Кожен Epoch підраховується на основі кількості кроків (steps) у навчальному генераторі (steps_per_epoch) та кількості кроків у валідаційному генераторі

(validation_steps). Ці метрики допомагають оцінити ефективність моделі під час навчання та визначити, чи відбувається перенавчання (overfitting).

Етап 9. Збереження навченої моделі. Збереження навченої моделі глибинного навчання у файлі з розширенням ".hdf5". Після завершення навчання ваги моделі, а також її архітектура та конфігурація, будуть збережені у цьому файлі. Це дозволить використовувати модель пізніше для передбачень без необхідності повторного навчання. Також цей файл можна буде використовувати для відновлення навченої моделі для подальших досліджень або вирішення задач класифікації.

Створення словнику class_map_101, який містить відображення між іменами класів та їхніми числовими індексами. Ключі цього словника - це назви класів, які визначені під час навчання моделі, а значення - це відповідні числові індекси класів.

Цей словник дозволяє зручно відстежувати відповідність між текстовими мітками класів і числовими значеннями, які використовуються під час навчання моделі:

```
class_map_101 = train_gen.class_indices
class_map_101
…
{'cucumber': 0, 'potato': 1, 'tomato': 2}
```

Етап 10. Створення інфраструктури для тестування моделі та виконання передбачень. Для ефективного тестування навченої моделі глибинного навчання і виконання передбачень на нових зображеннях необхідно підготувати відповідну інфраструктуру. В цьому розділі розглянемо кроки, необхідні для створення такої інфраструктури.

Очищення сесії TensorFlow

Першим кроком є очищення попередньо використаної сесії TensorFlow за допомогою наступного коду:

```
tensorflow.keras.backend.clear_session()
```

Цей крок допомагає уникнути конфліктів з попередніми об'єктами та моделями під час виконання наступних операцій.

Завантаження найкращої збереженої моделі:

Далі ми завантажуємо найкращу збережену модель з файлу best_model_101class.hdf5. Ми використовуємо наступний код для цього:

```
model_best = load_model('best_model_101class.hdf5', compile=False)
```

Параметр compile=False вказує, що модель не повинна бути скомпільована після завантаження, оскільки вона вже була скомпільована раніше.

Завантаження тестових зображень. Завантаження тестових зображень з Інтернету та їх збереження на локальному комп'ютері.

Наприклад, для завантаження зображення огірка використовується наступний код:

```
!wget --no-check-certificate -O cucumber.jpg https://greenplace.by/wp-content/uploads/2021/06/%D0%9E%D0%B3%D1%83%D1%80%D1%86%D1%8B.jpg
```

Аналогічно, завантажуються зображення помідора та картоплі (рис. 2.8). Ці зображення будуть використані для тестування моделі та виконання передбачень на них.

```
--2024-06-02 23:26:12--  https://greenplace.by/wp-content/uploads/2021/06/%D0%9E%D0%B3%D1%83%D1%80%D1%86%D1%8B.jpg
Reusing existing connection to greenplace.by:443.
HTTP request sent, awaiting response... 200 OK
Length: 31098 (30K) [image/jpeg]
Saving to: 'cucumber.jpg'
```

Рисунок 2.8 – Вивід результатів завантаження

Функція для передбачення класу зображення predict_class

```
def predict_class(model, images, show=True):
```

Ця функція приймає три аргументи:

– model – навчена модель глибинного навчання, яка буде використовуватися для передбачень;

– images – список шляхів до зображень, для яких ми хочемо здійснити передбачення.

– Show – булеве значення, яке вказує, чи потрібно відображати передбачення разом із зображеннями (за замовчуванням True):

```
pred = model.predict(img)
index = np.argmax(pred)
foods_sorted.sort()
pred_value = foods_sorted[index]
```

Після цього виконується передбачення класу для зображення за допомогою навченої моделі. Отриманий результат перетворюється у відповідний клас за допомогою відображення foods_sorted, яке містить індекси класів:

```
if show:
    plt.imshow(img[0])
    plt.axis('off')
    plt.title(pred_value)
    plt.show()
```

Нарешті, якщо параметр show встановлено на True, функція відображає зображення разом із передбаченим класом за допомогою бібліотеки matplotlib.pyplot (рис. 2.9).

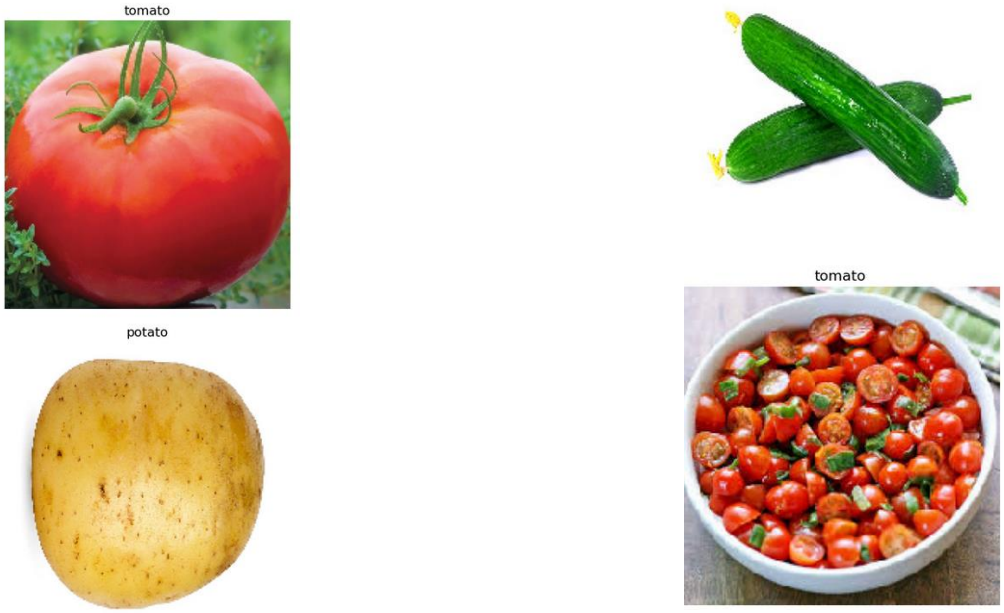

Рисунок 2.9 – Класифіковані зображення

2.3 Застосування моделі глибокого навчання для розпізнавання та класифікації продуктів харчування в рекомендаційній системі

2.3.1 Застосування моделі для класифікації продуктів харчування в рекомендаційній системі

Для використання отриманих даних у роботі, потрібно якось використовувати створену модель не у Jupiter Lab, а у Python додатку.

Розглянемо використання моделі глибинного навчання для класифікації об'єктів на зображеннях, отриманих з веб-камери. Розглянемо основні аспекти коду:

Потрібні функції використовують такі бібліотеки, як csv, math, os, time, cv2, tensorflow та інші для обробки зображень.

Завантаження моделі глибинного навчання. Програма завантажує найкращу збережену модель для передбачень. Модель завантажується з файлу 'best_model_101class.hdf5'.

Функція predict(img) приймає шлях до зображення, оброблює його та здійснює передбачення за допомогою завантаженої моделі. Вона повертає передбачений клас об'єкта на зображенні.

Ця функція призначена для передбачення класу об'єкта на зображенні та повертає назву передбаченого класу:

```
def predict(img):
  filename = img
  …
  pred = model_best.predict(pred_img)
  print("Pred")
  print(pred)
  top = pred.argsort()[0][-3:]
  label.sort()
  _true = label[top[2]]
  print(_true)
  return _true
```

_true – назва розпізнаного класу

– вхідні параметри: Функція отримує шлях до зображення img;

– завантаження зображення: Спочатку шлях до зображення зберігається у змінній filename, а потім завантажується зображення за допомогою функції image.load_img, і змінює його розмір на (200, 200). Після цього зображення перетворюється у масив NumPy за допомогою image.img_to_array;

– нормалізація зображення: Зображення нормалізується, діленням на 255, щоб значення пікселів були у діапазоні [0,1];

– передбачення класу: Зображення подається на вхід навченої моделі, щоб здійснити передбачення. Результат передбачення зберігається у змінній pred;

– перевірка на NaN: Якщо результат передбачення містить значення NaN, то вони замінюються на значення за замовчуванням;

– визначення передбаченого класу: Функція визначає три найбільш вірогідних класу, використовуючи функцію argsort(). Потім відсортовані класи зберігаються в змінній _true;

– повернення результату: Назва передбаченого класу повертається як результат роботи функції.

2.3.2 Розробка і використання функцій для захоплення і розпізнавання зображень

Для того, щоб використовувати модель у цілях розпізнавання, потрібно передавати їй зображення, коли буде помічено рух на камері.

У даному розділі розглядається функція startCaptureProcess(), яка відповідає за початок процесу захоплення зображень з камери. Вона використовує функцію getObjects(), а далі функцію predict()

Процес роботи нескінченого процесу розпізнання зображено на діаграмі послідовності:

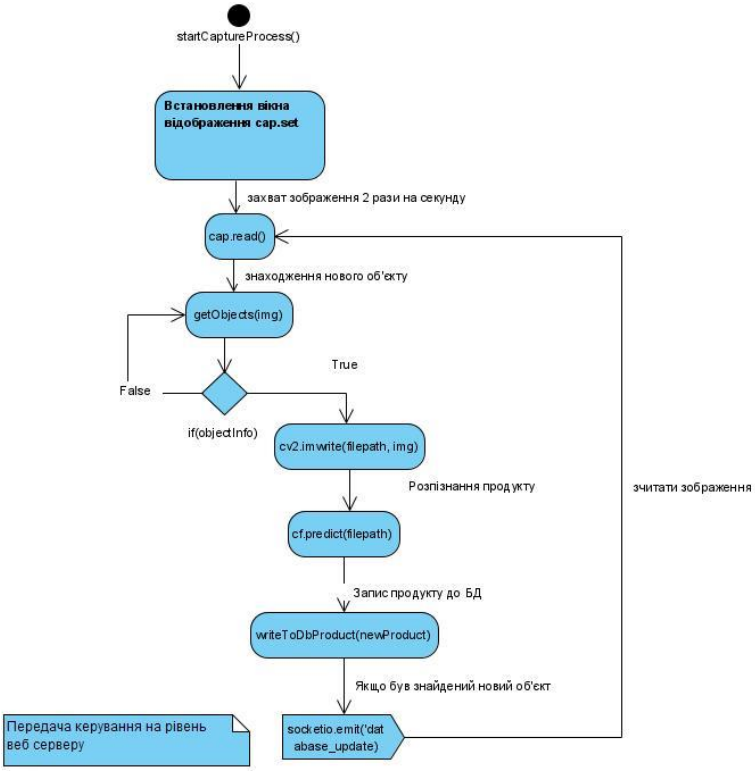

Рисунок 2.10 – Діаграма активності роботи процесу розпізнання продуктів харчування

Відкриття камери

Відкриття вбудованої камери пристрою зійснюється за допомогою функції cv2.VideoCapture(0). Якщо камера не вдалося відкрити, виводиться повідомлення про помилку і програма завершується:

```
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open camera.")
    exit()
```

Встановлюється бажана кількість кадрів на секунду за допомогою функції cap.set(cv2.CAP_PROP_FPS, desired_fps). Потім перевіряється, чи вдалося це зробити. Це потрібно для зниження навантаження. Для захвату зображень продуктів вистачає і 2 кадрів за секунду:

```
desired_fps = 2
cap.set(cv2.CAP_PROP_FPS, desired_fps)
actual_fps = cap.get(cv2.CAP_PROP_FPS)
print(f"FPS set to: {actual_fps}")
```

Встановлення розмірів відеокадру: Розмір відеокадру встановлюється на ширину 640 та висоту 480 пікселів. Для використання на сервері цю опцію можна вимкнути щоб зменшити навантаження:

```
cap.set(3,640)
cap.set(4,480)
```

Захоплення кадрів з камери: У безкінечному циклі захоплюються кадри з камери за допомогою функції cap.read():

```
while True:
    success, img = cap.read()
```

Розпізнавання об'єктів: Зображення, яке було захоплено, подається на вхід функції getObjects() для розпізнавання об'єктів:

```
result, objectInfo = getObjects(img, 0.5, 0.2)
```

Збереження та передбачення класу: Якщо об'єкт було розпізнано на зображенні, зображення зберігається, а потім за допомогою функції predict() передбачається клас цього об'єкта:

```
if(objectInfo):
    currentTime = datetime.now()
    filename = f'{currentTime:%d-%m-%Y-%H-%M-%S}.jpg'
```

```
filepath = os.path.join(UPLOAD_FOLDER,filename)
cv2.imwrite(filepath, img)
predictedName = cf.predict(filepath)
newProduct = Product()
newProduct.setVals(predictedName, filepath, datetime.utcnow())
writeToDbProduct(newProduct)
```

Діаграма послідовності роботи збереження та передбачення класу подвється на рис. 2.11.

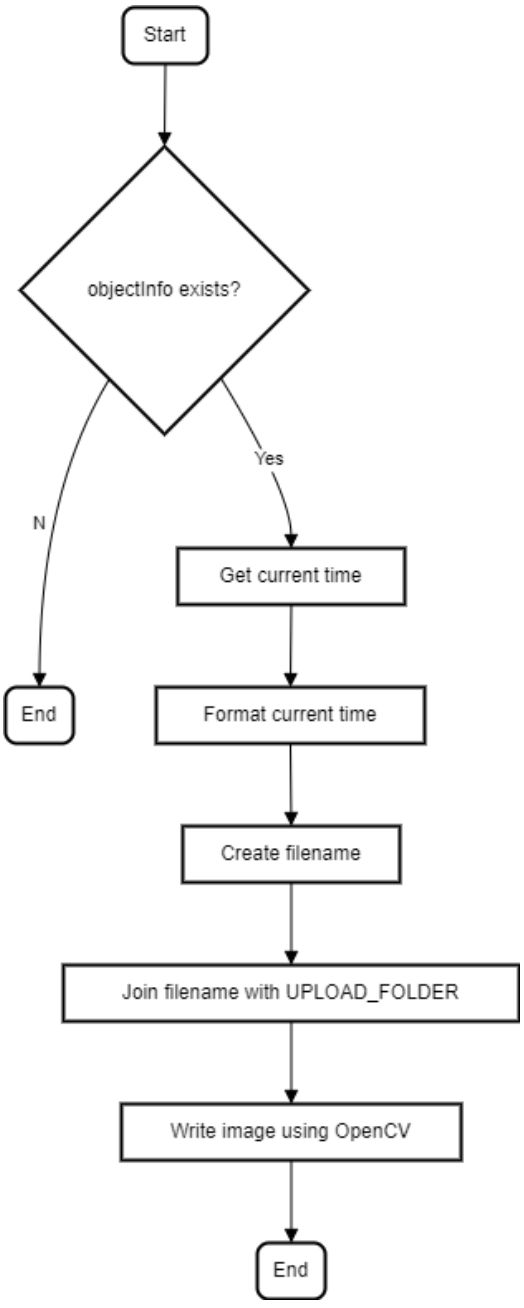

Рисунок 2.11 – Діаграма послідовності роботи збереження та передбачення класу

Пояснення до діаграми:

– filename = f'{currentTime:%d-%m-%Y-%H-%M-%S}.jpg' – створення рядка filename, який містить поточну дату та час у форматі "день-місяць-рік-година-хвилина-секунда", призначений для імені файлу;

– filepath = os.path.join(UPLOAD_FOLDER,filename) – створення повного шляху до файлу шляхом об'єднання шляху до завантажувальної теки UPLOAD_FOLDER та імені файлу filename;

– cv2.imwrite(filepath, img) – збереження зображення img у файл зі шляхом filepath у форматі JPEG.

– predictedName = cf.predict(filepath) – виклик функції predict для передбачення класу об'єкта на зображенні, що збережене у файлі;

– newProduct = Product() – Створення нового об'єкта класу Product, який призначений для зберігання інформації про виявлений продукт;

– newProduct.setVals(predictedName, filepath, datetime.utcnow()) – встановлення значень атрибутів об'єкта newProduct на основі передбаченого імені класу predictedName, шляху до зображення filepath та поточного часу у форматі UTC;

– writeToDbProduct(newProduct) – Запис об'єкта newProduct у базу даних продуктів;

– відображення зображення – Зображення виводиться на екран за допомогою функції cv2.imshow("Output",img), cv2.imshow("Output",img), cv2.waitKey(1).

Цей процес продовжується в циклі, доки програма не буде вручну зупинена користувачем.

Для швидкого виявлення нових об’єктів використовується алгоритм Non-Maximum Suppression. Алгоритм (NMS) використовується для підвищення точності об'єктного визначення шляхом прибирання зайвих областей, що перекриваються або дублюються. Основні кроки алгоритму NMS такі:

Крок 1. Вибір об'єктів – після виявлення об'єктів на зображенні визначаються їх межі та ймовірності наявності.

Крок 2. Сортування за ймовірністю – об'єкти сортуються в порядку спадання за ймовірністю наявності. Це означає, що перший об'єкт у списку має найвищу ймовірність.

Крок 3. Видалення перекриваючихся областей – для кожного об'єкта у списку перевіряється, чи перекривається він з іншими об'єктами, які мають більшу ймовірність. Якщо таке перекриття виявлене, то поточний об'єкт видаляється.

Крок 4. Залишення лише найбільш впевнених об'єктів – після завершення перевірки залишаються лише найбільш впевнені об'єкти, які не перекриваються з більш впевненими об'єктами.

Крок 5. Повернення результатів – остаточний список об'єктів, що лишилися після застосування NMS, вважається остаточним результатом розпізнавання об'єктів.

Алгоритм NMS допомагає покращити точність об'єктного визначення, видаляючи зайві області та залишаючи лише найбільш впевнені об'єкти:

```
def getObjects(img, thres, nms, draw=True, objects=[]):
    classIds,                  confs,                  bbox                  =
net.detect(img,confThreshold=thres,nmsThreshold=nms)
    #print(classIds,bbox)
    if len(objects) == 0: objects = classNames
    objectInfo =[]
    if len(classIds) != 0:
        for              classId,              confidence,box              in
zip(classIds.flatten(),confs.flatten(),bbox):
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box,className])
    return img,objectInfo
```

Пояснення до коду:

– `img` – вхідне зображення, на якому виконується пошук об'єктів;

– `thres` – поріг впевненості для відфільтрування слабких детекцій. Об'єкти з впевненістю нижче цього порогу будуть відкинуті;

– `nms` – поріг Non-Max Suppression (NMS), який визначає ступінь перекриття детекторів. Чим вище значення, тим більше детекторів буде допущено до виходу, навіть якщо вони перекриваються;

– `draw` (за замовчуванням `True`) – Прапорець, що вказує, чи потрібно малювати прямокутники та мітки на зображенні для позначення виявлених об'єктів. Якщо `True`, області з виявленими об'єктами будуть намальовані;

– `objects` (за замовчуванням `[]`) – Список імен об'єктів, які потрібно виявити на зображенні. Якщо цей список не вказано, функція буде використовувати всі доступні класи.

Діаграма роботи функції getObjects() подається на рис. 2.12.

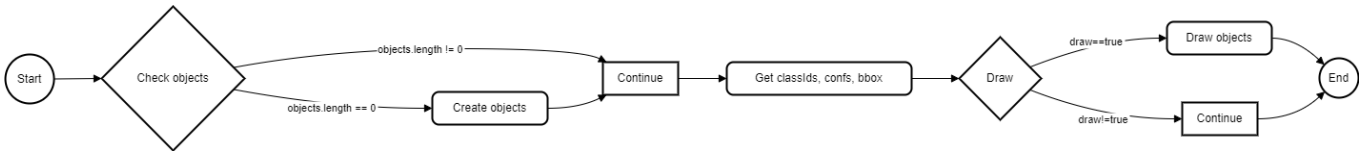

Рисунок 2.12 – Діаграма роботи функції getObjects()

Кожен крок роботи функції getObjects():

– отримання даних від нейронної мережі – функція викликає метод detect з об'єкта net для отримання класів, впевненості та межі рамок об'єктів на зображенні. Параметри confThreshold та nmsThreshold встановлюють порог впевненості та поріг NMS відповідно.

– фільтрація об'єктів – функція фільтрує отримані класи та рамки об'єктів згідно з переданими в аргументі objects іменами класів. Якщо список objects порожній, то вважається, що всі класи допустимі.

– створення вихідного зображення – Якщо аргумент draw встановлений в True, то функція малює прямокутники та написи на вихідному зображенні для кожного розпізнаного об'єкта. Прямокутники навколо об'єктів малюються зеленим кольором, а текст з назвою класу та впевненістю малюється зеленим кольором

– повернення результатів – функція повертає вихідне зображення з нанесеними прямокутниками та написами для кожного розпізнаного об'єкта, а також інформацію про об'єкти у форматі [рамка, клас].

3 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ МЕТОДІВ ДЛЯ РЕАЛІЗАЦІЇ ФУНКЦІЙ СИСТЕМИ З ПРИГОТУВАННЯ СТРАВ

Механізм рекомендацій, часто називаний системою рекомендацій, є інструментом, який дозволяє розробникам алгоритмів передбачати, чи сподобається користувачеві від певного списку елементів. Він представляє собою цікаву альтернативу пошуковим системам, сприяючи знаходженню користувачами продуктів або вмісту, які вони, можливо, не знайшли б інакше. Механізми рекомендацій стають важливою складовою багатьох веб-сайтів і сервісів, таких як Facebook, YouTube, Amazon та інші.

Вони також стали на нагоді при розробці у системі рекомендацій для досліджуваної системи.

3.1 Неперсоналізовані рекомендаційні функції, що реалізуються за модифікованим методом спільної фільтрації

Ідеально механізми рекомендацій функціонують за одним із двох підходів. Вони можуть базуватися на властивостях елементів, які сподобалися користувачеві, аналізуючи їх для визначення подальших рекомендацій. Або вони можуть враховувати симпатії та антипатії інших користувачів, використовуючи їх для обчислення індексу схожості та відповідно рекомендуючи елементи.

Неперсоналізовані ці методи тому, що не адаптовані до конкретного користувача і не враховують його унікальних вподобань або історії. Замість цього, використовується більш загальний підхід, який може бути застосований для всіх користувачів або для широкого кола сценаріїв.

Також можна комбінувати обидва ці підходи для створення більш надійного механізму рекомендацій. Проте, як і в усіх інших випадках, пов'язаних з

інформацією, вибір відповідного алгоритму для вирішення конкретної проблеми має вирішальне значення.

Перш ніж запроваджувати механізм рекомендацій на основі спільної пам’яті, потрібно спочатку зрозуміти основну ідею такої системи. Для цього механізму кожен елемент і кожен користувач є не що інше, як ідентифікатори. Ми будемо брати до уваги атрибути рецептів (наприклад, складності приготування, час приготування, калорійність, відповідність до національної кухні, тощо) під час формування рекомендацій. Подібність між двома користувачами представлена за допомогою десяткового числа від -1,0 до 1,0. Назвемо це число індексом подібності. Нарешті, можливість того, що користувачеві сподобався рецепт, буде представлена за допомогою іншого десяткового числа від -1,0 до 1,0. Тепер, коли ми змоделювали світ навколо цієї системи за допомогою простих термінів, ми можемо розв’язати декілька математичних рівнянь, щоб визначити зв’язок між цими ідентифікаторами та числами.

В алгоритмі рекомендацій має підтримуватись кілька наборів. Кожен користувач матиме два набори – набір рецептів, які йому подобаються, і набір рецептів, які не подобаються. Кожен рецепт також матиме два набори, пов’язані з ним – набір користувачів, яким рецепт сподобався, і набір користувачів, яким рецепт не сподобався. Під час етапів, на яких генеруються рекомендації, буде створено кілька наборів – переважно об’єднань або перетинів інших наборів. Ми також матимемо впорядковані списки пропозицій і схожих користувачів для кожного користувача.

Для розрахунку індексу подібності ми скористаємося різновидом формули індексу Жаккара. Спочатку відома як «coefficient de communauté» (придумана Полем Жаккаром), формула порівнює два набори та створює просту десяткову статистику від 0 до 1,0:

$$J(A,B) = |A \cap B| \div |A \cup B|$$

Формула включає в себе ділення кількості спільних елементів у будь-якій множині на кількість усіх елементів (рахованих тільки один раз) в обох множинах.

Індекс Жаккара двох однакових наборів завжди дорівнюватиме 1, тоді як індекс Жаккара двох наборів без спільних елементів завжди даватиме 0. Потрібно обрати стратегію, яку можливо використовувати для порівняння двох користувачів. Користувачі, з точки зору системи, це три речі: ідентифікатор, набір рецептів, які сподобалися, і набір рецептів, які не сподобалися. Якби ми мали визначити індекс схожості наших користувачів лише на основі набору рецептів, які їм сподобалися, ми могли б безпосередньо використати формулу індексу Жаккара:

$$S(U_1, U_2) = \left|L_1 \cap L_2\right| \div \left|L_1 \cup L_2\right|$$

Тут U1 і U2 — це два користувачі, яких ми порівнюємо, а L1 і L2 — набори рецептів, які сподобалися U1 і U2 відповідно. Два користувачі, яким подобаються однакові рецепти, схожі, тоді два користувачі, яким не подобаються однакові рецепти, також повинні бути схожими. Тут потрібно змінити рівняння:

$$S(U_1, U_2) = (\left|L_1 \cap L_2\right| + \left|D_1 \cap D_2\right|) \div \left|L_1 \cup L_2 \cup D_1 \cup D_2\right|$$

Замість того, щоб просто враховувати загальні оцінки «подобається» в чисельнику формули, тепер ми також додаємо кількість загальних оцінок «не подобається». У знаменнику ми беремо кількість усіх елементів, які сподобалися або не сподобалися користувачам. Тепер, розглянуто як «подобається», так і «не подобається» окремо, потрібно подумати про випадок, коли два користувачі є полярними протилежностями у своїх уподобаннях. Індекс подібності двох користувачів, де одному рецепт подобається, а іншому він не подобається, не повинен дорівнювати 0:

$$S(U_1, U_2) = (\left|L_1 \cap L_2\right| + \left|D_1 \cap D_2\right| - \left|L_1 \cap D_2\right| - \left|L_2 \cap D_1\right|) \div \left|L_1 \cup L_2 \cup D_1 \cup D_2\right|$$

Формула схожа на попередню формулу з невеликою різницею в чисельнику. Віднімаємо кількість суперечливих оцінок "подобається" та "не подобається" двом користувачам із кількості їхніх спільних оцінок "подобається" та "не подобається". Це призводить до того, що формула індексу подібності має діапазон значень від -1,0

до 1,0. Два користувачі з ідентичними смаками матимуть індекс подібності 1,0, тоді як два користувачі з абсолютно суперечливими смаками у рецептах матимуть індекс подібності -1,0.

Тепер, коли ми знаємо, як порівнювати двох користувачів на основі їхніх уподобань у рецептах, потрібно розглянути ще одну формулу:

$$P(U,M) = (Z_L - Z_D) \div (\left|M_L\right| + \left|M_D\right|)$$

Під P(U,M) маємо на увазі можливість того, що користувачеві U сподобається рецепт. M_L і M_d є сумою індексів схожості користувача U з усіма користувачами, яким сподобався або не сподобався рецепт M відповідно. $|M_L|+|M_D|$ представляють загальну кількість користувачів, яким сподобався або не сподобався рецепт M. Результат P(U,M) дає число від -1,0 до 1,0.

3.2 Рекомендаційні функції, що реалізуются за базовим методом спільної фільтрації

Алгоритми механізму спільних рекомендацій на основі пам’яті можуть бути досить потужною річчю. Спільна фільтрація — це спосіб, у який системи рекомендацій фільтрують інформацію, використовуючи переваги інших людей. Він використовує припущення, що якщо особа А має такі ж уподобання, як особа Б, щодо елементів, які вони обидва переглянули, то особа А, ймовірно, матиме подібні переваги до особи Б щодо елемента, який переглянула лише особа Б.

З розвитком Інтернету та швидшими комп’ютерами компанії по всьому світу зберігають великі обсяги даних. Це призвело до появи підгалузі інформатики під назвою «великі дані» , яка зосереджена на проблемах зберігання та аналізу величезних масивів даних. Спільна фільтрація – це спосіб отримання корисної інформації з цих даних у загальному процесі, який називається фільтрацією інформації. Алгоритм порівнює користувача з іншими подібними користувачами (з точки зору переваг) і рекомендує конкретний продукт або дію на основі цих подібностей.

Зокрема, у схемі спільної фільтрації використовуються такі кроки:

– користувач виражає переваги рецептів, зазвичай оцінюючи їх;

– алгоритм знаходить інших користувачів зі смаками, схожими на даного користувача;

– система рекомендує елементи, які користувач ще не оцінював (тому, ймовірно, є новими для користувача), і які високо оцінені користувачами, схожими на даного користувача;

– застереження в цьому процесі включає активну участь користувачів сервісу. Для того, щоб отримати рекомендацію рецепту, користувач повинен вже лайкати або дизлайкати рецепти в минулому; інакше система не надаватиме хороших рекомендацій.

Методи спільної фільтрації (CF) збирають уподобання у формі оцінок або сигналів від багатьох користувачів (звідси і назва), а потім рекомендують товари користувачеві на основі взаємодії людей із подібними смаками, як у цього користувача в минулому. Іншими словами, ці методи припускають, що якщо особі X подобається підмножина предметів, які подобаються особі Y, то X, швидше за все, матиме ту саму думку, що й Y щодо певного елемента, порівняно з випадковою особою, яка може мати або не мати уподобання.

Основна ідея методів, заснованих на сусідстві, полягає в тому, щоб використовувати або подібність між користувачами, або подібність між елементами для надання рекомендацій. Ці методи припускають, що подібні користувачі, як правило, мають однакову поведінку під час оцінювання елементів. Ми також можемо поширити це припущення на предмети: подібні предмети, як правило, отримують однакові оцінки від одного користувача.

У цих методах взаємодія між користувачами та елементами, як правило, представлена матрицею «користувач-елемент», де кожен рядок представляє користувача, а кожен стовпець — елемент, тоді як клітинки представляють взаємодію між двома, які в більшості випадків є оцінки товарів, зроблені користувачами. У цьому контексті ми можемо визначити два типи методів на основі сусідства.

Базова спільна фільтрація на основі користувачів:

– оцінки, надані користувачами, як-от користувач U, використовуються для надання рекомендацій. Точніше, щоб передбачити рейтинг U для певного елемента I, ми обчислюємо середньозважену оцінку r k схожих користувачів (сусідів) на U, де ваги визначаються подібністю між U та кожним із подібних користувачів (рис. 3.6).

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u, v)}$$

3.3 Удосконалена формула колаборативної фільтрації з урахуванням середнього значення

Колаборативна фільтрація є одним із найбільш розповсюджених методів рекомендаційних систем. Вона базується на схожості між користувачами або елементами. Основна ідея полягає у прогнозуванні оцінок, які користувач може дати невідомим елементам, на основі інформації про оцінки, зроблені іншими користувачами.

Для покращення точності прогнозування використовують удосконалену формулу, яка враховує середнє значення оцінок користувачів.

Розрахунок подібності між користувачами для прогнозування оцінок може ввести в оману, оскільки користувачі можуть оцінювати елементи по-різному. Коли ви надаєте користувачеві діапазон значень, він/вона може інтерпретувати їх по-різному. Наприклад, у 5-зірковій системі оцінювання користувач може оцінити елемент як 3, оскільки він виконує те, що від нього очікується, і нічого більше, тоді як інші можуть використовувати 3, щоб оцінити елемент, який майже не працює. Деякі користувачі оцінюють елементи високо, а інші – менш прихильно. Щоб вирішити цю проблему, оцінки мають бути зосереджені на користувачеві, тобто середня оцінка користувача віднімається від необробленої оцінки, а середня оцінка цільового користувача додається до розрахунку, як у прикладі нижче:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

3.4 Порівняння зміненої формули Жаккара, базова формула колаборативної фільтрації (user-based) та удосконалена формула з урахуванням середнього значення

Розглянемо три методи колаборативної фільтрації, які використовуються для прогнозування оцінок у системах рекомендацій: змінена формула Жаккара з урахуванням суперечливих оцінок, базова формула колаборативної фільтрації (user-based) та удосконалена формула з урахуванням середнього значення.

Змінена формула Жаккара з урахуванням суперечливих оцінок

Цей метод використовує коефіцієнт Жаккара для оцінки схожості між користувачами з урахуванням не тільки спільних позитивних оцінок, але й суперечливих оцінок. Підходить для систем, де важливо враховувати як позитивні, так і негативні оцінки користувачів. Вона забезпечує високу точність, але має високу обчислювальну складність і середню складність реалізації.

Базова формула колаборативної фільтрації (user-based)

Є простою у реалізації та використовується в простих системах рекомендацій. Вона не враховує індивідуальні упередження користувачів, що може знизити точність прогнозів. Удосконалена формула з урахуванням середнього значення забезпечує більш високу точність прогнозів за рахунок врахування середніх оцінок користувачів. Вона підходить для систем, де важлива висока точність рекомендацій, але має вищу складність реалізації та високу обчислювальну складність.

За результатами порівняльного аналізу методів колаборативної фільтрації можна зробити такі висновки:

– для простих систем рекомендацій з обмеженими ресурсами найкраще використовувати базову формулу колаборативної фільтрації;

– для систем, де важливо враховувати як позитивні, так і негативні оцінки, варто розглянути змінену формулу Жаккара;

– для систем, де потрібна висока точність прогнозів, рекомендується використовувати удосконалену формулу з урахуванням середнього значення.

Порівняльний аналіз методів колаборативної фільтрації подається в табл. 3.4.

Таблиця 3.4 – Порівняльний методів колаборативної фільтрації

Характеристика	Змінена формула Жаккара	Базова формула (user-based)	Удосконалена формула з урахуванням середнього значення
Складність реалізації	Середня	Легка	Середня
Урахування суперечливих оцінок	Так	Ні	Ні
Урахування індивідуальних особливостей	Частково	Не враховує	Враховує
Точність прогнозів	Висока	Середня	Вища
Обчислювальна складність	Висока	Висока	Висока
Застосування	Рекомендаційні системи з високою точністю	Прості системи рекомендацій	Системи з високою точністю рекомендацій

Для рекомендаційної системи з приготування страв для набору розпізнаних та класифікованих продуктів харчування обраний метод прогнозування з використанням удосконаленої формули з урахуванням середнього значення.

4 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ З ПРИГОТУВАННЯ СТРАВ

4.1 Зберігання даних рекомендаційної системи з приготування страв за допомогою СУБД MongoDb

Для зберігання рецептів, даних про продукти та рецептів використовується СУБД MongoDb.

Усі розпізнані продукти зберігаються як окремі JSON об’єкти. Нам не потрібно Зберігати зв’язки рецептів з продуктами. Кожного разу, коли користувач хоче щось приготувати, до рецепту підтягуються топ старих об’єктів продуктів. Ми вкладаємо один об’єкт у другий.

Нові рецепти додаються не так часто як продукти. При запуску додатку, у програму завантажується колекція рецептів. Кожного разу, коли до бази додається новий продукт, словник типу продукту, кількості наявних продуктів та потрібних продуктів оновлює своє значення. При цьому, об’єкти рецептів у БД не оновлюються. Таке рішення було обрано з точки зору швидкості оновлення. Адже оновлення кожного рецепту у БД зайняло б дуже багато часу.

Також, оскільки веб частина додатку розроблена на React, який динамічно оновлює картки з рецептами, використовує в якості об’єктів саме JSON документи, дуже ефективно використовувати сховище, яке також зберігає дані у такому форматі. Таким чином, ми нічого не конвертуємо, а працюємо напряму з «сирими» даними. Усі прийняті рішення позитивно впливають на ефективність роботи на слабкому апаратному забезпечені. Вид форми зберігання об’єктів продуктів подається на рис. 4.1.

NoSQL бази даних є більш гнучкими, ніж реляційні бази даних, що робить їх ідеальними для зберігання даних, які мають складну або мінливу структуру.

Головними даними будуть рецепти, що мають мінливу структуру, а отже, NoSQL база даних є найдоречнішою.

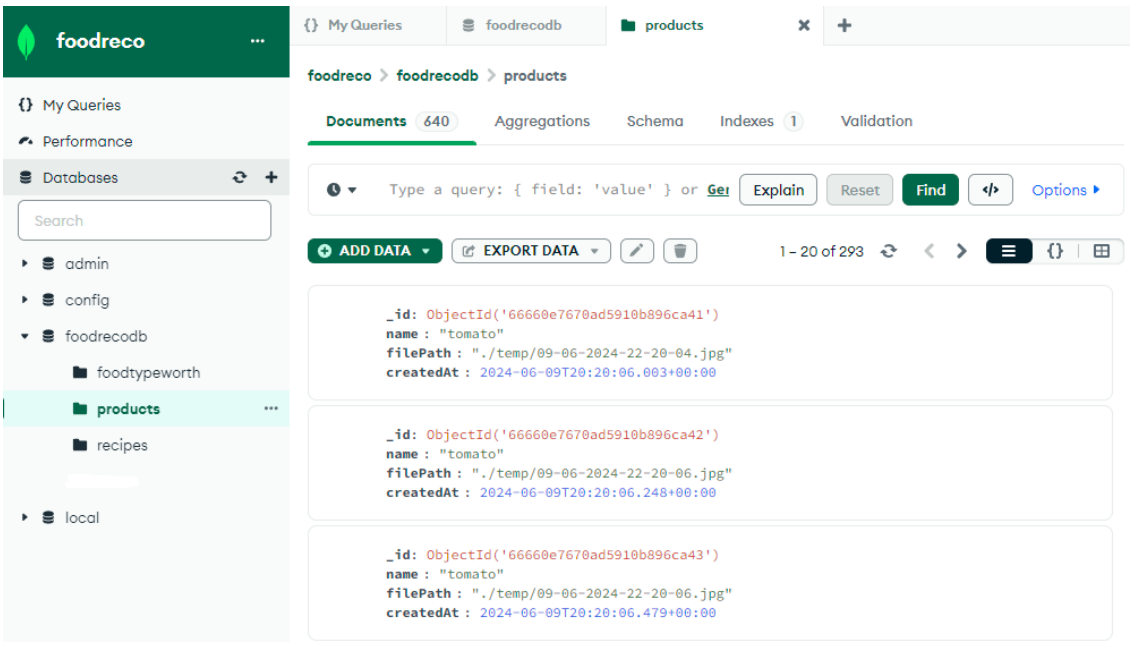

Рисунок 4.1 – Вид зберігання об’єктів продуктів

MongoDB — це база даних NoSQL, розроблена MongoDB inc, яка не містить схем. Він був розроблений і створений з використанням C і javascript, що забезпечує кращі можливості підключення. Він використовує колекцію документів і також має опцію для створення схем. Він не відповідає структурі традиційної бази даних, де дані зберігаються у формі рядків.

Оскільки загальні реляційні СУБД легші у використанні, це стосується і MongoDB. MongoDB використовує платформу NoSQL, що полегшує роботу особам, які мають менше або зовсім не мають попередніх знань програмування. MongoDB обробляє дані в напівструктурованому форматі, що дозволяє обробляти великі обсяги даних за один раз. Його можна розмістити майже на всіх хмарних платформах, будь то Google Cloud, Microsoft Azure або навіть веб-сервіси Amazon. MongoDB використовує Binary JSON і MQL як альтернативу SQL. BSON дозволяє використовувати такі типи даних, як число з плаваючою комою, long, дата та багато інших, які не підтримуються звичайним JSON. MQL пропонує додаткові можливості порівняно зі звичайним SQL, що робить його більш актуальним для MongoDB, оскільки він обробляє документи типу JSON. MongoDB — це сервер NoSQL, на якому дані зберігаються в документах BSON (Binary JSON), і кожен документ, по суті, побудований на структурі пари ключ-значення. Оскільки MongoDB легко

зберігає дані без схеми, зробіть його придатним для захоплення даних, структура яких невідома. Цей підхід, орієнтований на документ, створений для того, щоб запропонувати багатший досвід роботи з сучасними техніками програмування.

Зберігання інформації про рецепти страв та інформації про продукти в MongoDb. В якості клієнту СУБД використано MongoDb Atlas. MongoDB Atlas є хмарною версією MongoDB, яка пропонує безліч переваг, що робить її дуже зручною системою управління базами даних (СУБД). Ось основні причини:

MongoDB Atlas є хмарною версією MongoDB, яка пропонує безліч переваг, що робить її дуже зручною системою управління базами даних (СУБД). Ось основні причини:

– керована хмара (Managed Cloud Service):

– Mongodb Atlas є повністю керованим сервісом, що означає, що користувачі не повинні турбуватися про налаштування, конфігурацію, моніторинг та оновлення бази даних. Всі ці аспекти обробляються автоматично;

– швидкий розгортання та масштабування;

– atlas дозволяє легко і швидко розгорнути нові кластери та масштабувати їх за потребою. Ви можете налаштувати автоматичне масштабування відповідно до навантаження, що забезпечує оптимальну продуктивність і витрати;

– висока доступність;

– Mongodb Atlas забезпечує високу доступність за допомогою автоматичного резервування даних у кількох регіонах та відмовостійких механізмів. Це гарантує безперебійну роботу додатків, навіть у випадку збоїв апаратного забезпечення.

– інтеграція з основними хмарними провайдерами:

– atlas підтримує інтеграцію з основними хмарними провайдерами, такими як AWS, Google Cloud Platform та Microsoft Azure. Це дозволяє користувачам обрати хмарну платформу, яка найкраще відповідає їхнім потребам;

– безпека. MongoDB Atlas надає передові функції безпеки, включаючи шифрування даних в русі і на місці зберігання, а також інтеграцію з ключовими керуючими системами. Крім того, Atlas підтримує складні механізми контролю доступу і автентифікації користувачів;

– автоматичне резервне копіювання. Сервіс пропонує автоматичне резервне копіювання, що забезпечує безпеку даних та можливість швидкого відновлення у разі необхідності.

– моніторинг та аналітика – atlas включає вбудовані інструменти для моніторингу та аналітики, що дозволяє користувачам відстежувати продуктивність, виявляти і усувати проблеми в режимі реального часу.

– глобальна реплікація. Atlas підтримує глобальну реплікацію даних, що дозволяє розміщувати дані ближче до користувачів, зменшуючи затримки і покращуючи швидкодію додатків.

MongoDB Atlas є зручною і потужною СУБД, яка забезпечує просте управління, високу продуктивність та безпеку, масштабованість і надійність. Всі ці переваги роблять її відмінним вибором для різних типів додатків, від веб-додатків до аналітичних систем і IoT-проектів.

Розглянемо структуру на прикладі рецепту борщу. Об'єкт типу Recipe у колекції MongoDB має наступну структуру. Унікальний ідентифікатор об'єкту _id в колекції MongoDB генерується автоматично при вставці нового документу:

```
"_id": {
"$oid": "665dbc7de340aae969a2306e"
}
name: Назва рецепту, яка дає уявлення про страву.
"name": "Borsch classich"
servings: Кількість порцій, на які розрахований рецепт.
"servings": "8"
timeToPrepare: Час, необхідний для приготування страви, вказаний в хвилинах.
"timeToPrepare": "120"
description: Опис рецепту, що містить загальну інформацію про страву і її
особливості.
"description": "Every housewife is sure that she knows exactly how to cook
delicious borscht correctly. And among the many recipes there is borscht with
prunes. All other ingredients are the same as for regular, classic borscht
with beets, but the addition of prunes makes this first dish more
interesting."
productsNeeded: Перелік інгредієнтів, необхідних для приготування страви, з
вказанням кількості кожного інгредієнта в грамах.
```

Кількість окремого продукта зберігається у штуках. Вага кожної позиції дорівнює середньому значенню ваги пачки, штуки, тощо. Наприклад, для яловичини (beef) це значення дорівнює 400 грамам. Для томата це 100 грам. Для Огірка це 120 грам. Тому, кожен рецепт можна адаптувати до штук, просто розділивши потрібну вагу на середню вагу позиції.

```
"productsNeeded": {
    "beef": "1",
```

```
    "potatoes": "3",
    "white cabbage": "1",
    "beetroot": "2",
    "carrots": "2",
    "onions": "2",
    "tomato paste": "1",
    "salt": "1",
    "sun oil": "0.1",
    "water": "4"
  }
```


Покрокова інструкція з приготування страви instruction:


```
"instruction": "1. Prepare all necessary products. If the prunes are dry, pour boiling water over them for 10-15 minutes.\n2. The meat is cut into small pieces.\n3. Place the meat in a saucepan and fill with cold water. Bring to a boil, skim off the foam and cook the meat for 1.5 hours over low heat, covered.\n4. The potatoes are peeled and cut into large pieces.\n5. The beets are peeled and cut into strips. Usually beets are stewed in a frying pan with the addition of lemon juice or vinegar - you can do just that. But in this recipe, beets are added raw to the broth and cooked along with potatoes.\n6. The cabbage is chopped into strips.\n7. The onion is peeled and finely chopped.\n8. The carrots are peeled and grated on a coarse grater.\n9. Prunes are pitted and finely chopped.\n10. Heat the frying pan and add vegetable oil. Place onions and carrots in hot oil, fry the vegetables, stirring, for 3-4 minutes over medium heat.\nAdd tomato paste, stir and fry the vegetables and tomatoes for another 1 minute.\n11. Potatoes, beets, cabbage, fried onions and carrots are added to the meat broth. Add prunes, salt and cook the borscht with prunes until the potatoes are ready, 20-25 minutes.\n12. The greens are washed and finely chopped.\n13. The finished borscht with prunes is poured into plates and sprinkled with chopped parsley.\nServe borscht with sour cream.\nBon appetit!"
```


Ця структура зберігання забезпечує всі необхідні дані для збереження, пошуку та відображення рецептів страв у додатку.

На прикладі об’єкту розпізнаного томату розглянемо структуру об’єкту. Об'єкт розпізнаного продукту в базі даних MongoDB має таку структуру:

Опис структури об'єкта: id, Тип: Об'єкт, Поле: $oid, Тип значення: Рядок.

Опис унікального ідентифікатора об'єкта в базі даних. Використовується для унікального визначення кожного документа в колекції MongoDB. Значення генерується автоматично MongoDB.


```
"_id": {
  "$oid": "66660e7670ad5910b896ca41"
}
```

Назва розпізнаного продукту – name. У даному прикладі це "tomato", що вказує на те, що розпізнаний об'єкт є помідором.

```
"name": "tomato"
```

Шлях до зображення розпізнаного продукту. Це шлях до файлу на сервері, де зберігається зображення, використане для розпізнавання.

```
"filePath": "./temp/09-06-2024-22-20-04.jpg"
```

Дата і час створення запису в базі даних – $date (рядок за форматом дати та часу ISO 8601). Це час, коли об'єкт був розпізнаний і збережений у базі даних. Значення автоматично генерується під час створення документа.

```
"createdAt": {
  "$date": "2024-06-09T20:20:06.003Z"
}
```

Переваги використання MongoDB:

– гнучкість. MongoDB дозволяє зберігати дані в гнучкому форматі, що робить її ідеальною для зберігання даних про рецепти та продукти, які можуть мати складну або мінливу структуру;

– масштабованість. MongoDB легко масштабується, що робить її ідеальною для зберігання великої кількості даних про рецепти та продукти;

– продуктивність. MongoDB забезпечує високу продуктивність, що робить її ідеальною для рекомендаційних систем, які потребують швидкого доступу до даних.

4.2 Обране технічне рішення для апаратної реалізації рекомендаційної системи з приготування страв

Для роботи системи був обраний міні-комп’ютер Raspberry Pi 5.

Raspberry Pi 5 – це найновіша модель Raspberry Pi, яка пропонує ряд переваг, що роблять її ідеальною для запуску рекомендаційної системи з приготування страв:

– висока продуктивність: Raspberry Pi 5 оснащений чотириядерним процесором ARM Cortex-A76 з тактовою частотою 2,4 ГГц, що забезпечує високу продуктивність для запуску моделей машинного навчання;

– графічний процесор: Raspberry Pi 5 має вбудований графічний процесор VideoCore VI, який може бути використаний для прискорення обробки зображень та машинного навчання;

Його висока продуктивність, достатня кількість пам'яті, підтримка швидкого зберігання даних, різноманітні порти, підтримка різних операційних систем, низька вартість, енергоефективність та велике співтовариство роблять його ідеальним вибором для розробників, які хочуть створювати комп'ютерний зір та рекомендаційні системи.

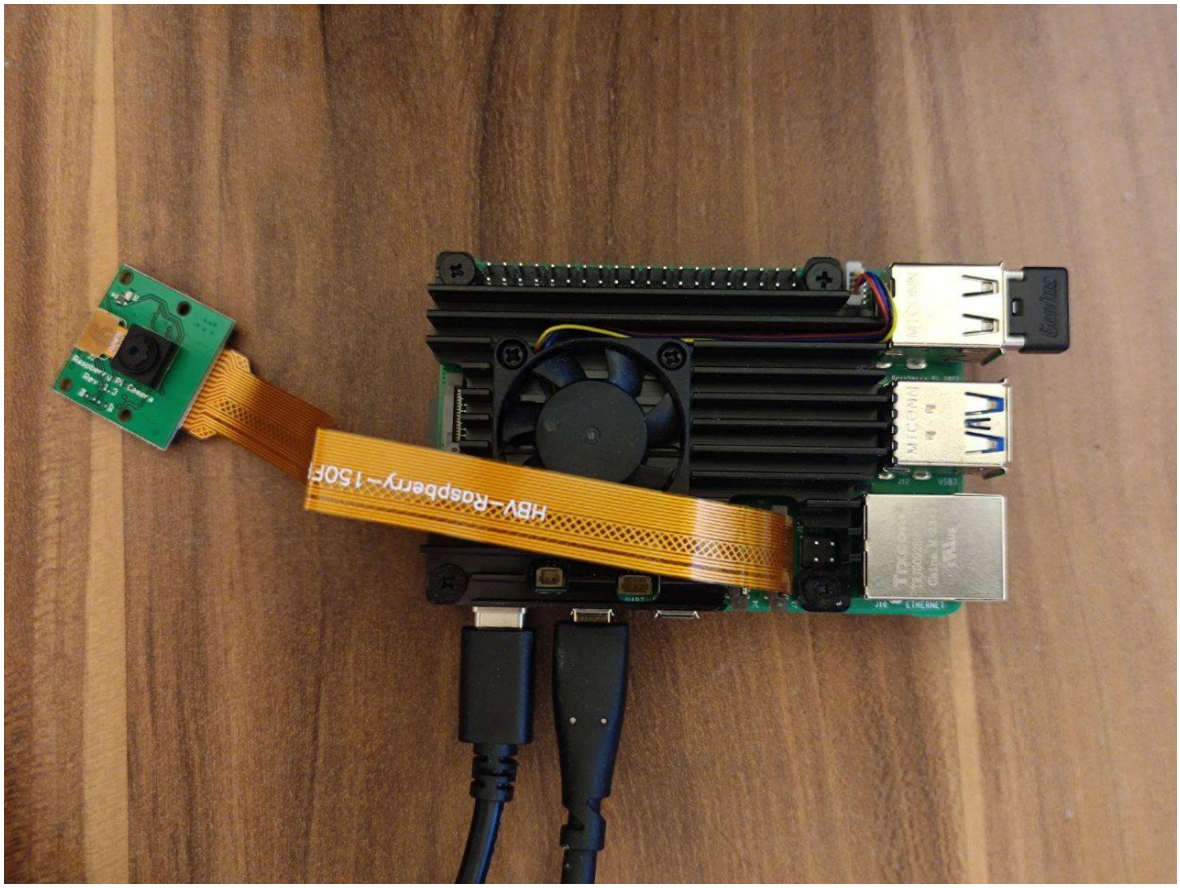

Рисунок 4.3 – Зібраний міні-ПК Raspberry Pi 5 з камерою

Проведено порівняльні тести з використанням фреймворку ncnn на Raspberry Pi 4 8 ГБ і Raspberry Pi 5 8 ГБ, щоб оцінити продуктивність висновків. ncnn — це ефективний і зручний фреймворк глибокого навчання, який підтримує різні моделі нейронних мереж, заокрема, TensorFlow. Ncnn було розроблено з урахуванням мобільного розгортання та пропонує прискорення графічного процесора через

Vulkan API з низькими витратами. При використанні можливостей графічного процесора Raspberry Pi 5 результати перевершують результати, отримані лише за допомогою процесора. Ось порівняння продуктивності двох моделей Raspberry Pi.

Таблиця 4.2 – Порівняння швидкості роботи Raspberry Pi 5 в різних алгоритмах фільтрації

Модель	Raspberry Pi 4 / 1 потік середня швидкість висновку, мс	Raspberry Pi 4 / 4 потоки середня швидкість висновку, мс	Raspberry Pi 5 / 1 потік середня швидкість висновку, мс	Raspberry Pi 5 / 4 потоки середня швидкість висновку, мс
mobilenet_v2	93,48	61,88	19.24	13.63
mobilenet_v3	72,81	47,53	13.25	9.48
efficientnet_b0	129,73	74.19	26.24	14.87

4.3 Вибір технологій та фреймворків для реалізації серверної частини рекомендаційної системи з приготування страв

Розробка бекенд-частини системи є критичним аспектом загальної архітектури програмного забезпечення. Вибір технологій та фреймворків для реалізації бекенд-логіки впливає на продуктивність, масштабованість та зручність розробки.

В якості вебсервера обраний додаток python Flask. Flask – це мікрофреймворк, що базується на Python, який відомий своєю простотою і легкістю використання. Його мінімалістичний підхід дозволяє розробникам швидко розпочати роботу, зосередившись лише на необхідних компонентах.

Хоча Flask є мікрофреймворком, його можна використовувати для побудови масштабованих додатків. Використання WSGI-сумісних серверів, таких як Gunicorn, дозволяє масштабувати додаток для обробки великої кількості запитів.

Python відомий своєю багатою екосистемою бібліотек для машинного навчання, обробки даних та наукових обчислень, таких як TensorFlow, NumPy,

Pandas. Flask легко інтегрується з цими бібліотеками, що робить його ідеальним вибором для додатків, які потребують складної обробки даних або інтеграції з моделями машинного навчання.

Розглянемо, як вищезазначені переваги реалізуються у конкретному додатку на основі наданого коду.

Легкість інтеграції з MongoDB. Flask дозволяє легко інтегруватися з MongoDB, використовуючи бібліотеки, такі як pymongo. У коді MongoDB використовується для зберігання даних про користувачів, продукти та рецепти:

```
from pymongo import MongoClient
MONGO_URI = 'your_mongo_uri'
mongo = MongoClient(MONGO_URI)['foodrecodb']
```

Обробка зображень і машинне навчання. Flask може бути інтегрований з OpenCV для обробки зображень та класифікації об'єктів. У нашому додатку використовується OpenCV для обробки зображень з камери, а результати обробки передаються через Flask:

```
import cv2
import classifyobject as cf
def startCaptureProcess():
    cap = cv2.VideoCapture(0)
```

Легкість розширення функціоналу. Функціонал додатку легко розширюється за рахунок додавання нових маршрутів та обробників запитів у Flask. Це дозволяє створювати нові API для роботи з користувачами, продуктами та рецептами. Приклад можливого додання реалізації маніпуляції колекціями користувачів додатка:

```
@app.route("/users", methods=['POST'])
def createUser():
    id = usersCollection.insert_one({
        'name': request.json['name'],
        'email': request.json['email'],
        'contact': request.json['contact'],
        'address': request.json['address']
    })
    return jsonify({'id': str(id.inserted_id), 'msg': "User added
succefully"})
```

Підтримка реального часу. Flask легко інтегрується з SocketIO для підтримки функціоналу реального часу, що важливо для миттєвого оновлення даних в додатку:

```
from flask_socketio import SocketIO
socketio = SocketIO(app, cors_allowed_origins="*")
```

4.4 Реалізація серверної частини рекомендаційної системи з приготування страв

Кожного разу, коли до бази даних додається новий продукт, за допомогою веб сокет надсилається сигнал до веб додатку. У цей момент, веб додаток виконує оновлення карток рецептів, оновлюючи для кожної картки список досупних продуктів.

У сучасних веб-додатках важливо забезпечити можливість оновлення інформації в режимі реального часу, особливо в тих випадках, коли дані змінюються часто і повинні відображатися для користувачів без затримок. В цьому контексті, інтеграція WebSocket забезпечує ефективний спосіб для досягнення такого результату. Далі буде детально пояснено, як саме це реалізовано в нашому додатку.

Огляд послідовності дій основної функці додатку:

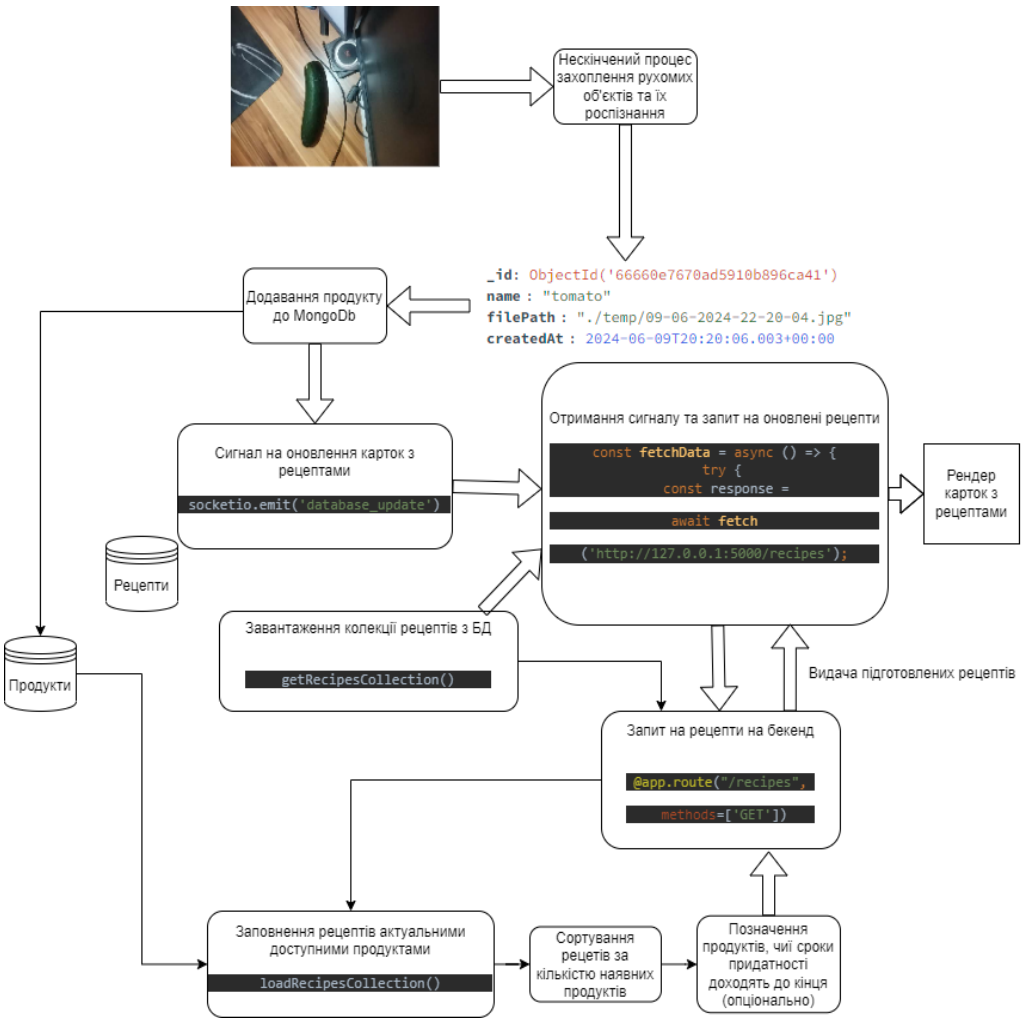

Рисунок 4.5 – Послідовність дій основної функці додатку

Механізм оновлення даних у базі даних і повідомлення через WebSocket. Код для запису продукту в базу даних і надсилання сигналу:

```
def writeToDbProduct(product: Product):
    getFoodsCollection().insert_one({
        "name": product.name,
        "filePath": product.filePath,
        "createdAt": product.createdAt
    })
    socketio.emit('database_update', broadcast=True)
```

Діаграма послідовності запису продукту в базу даних подається на рис. 4.6.

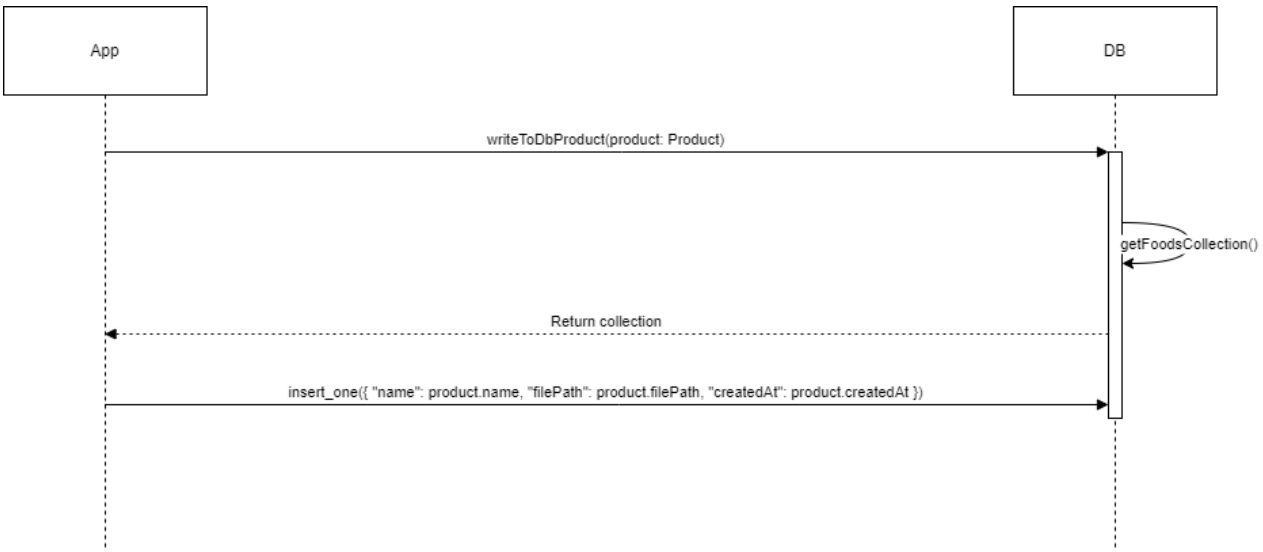

Рисунок 4.6 – Діаграма послідовності запису продукту в базу даних

Опис функцій діаграми:

– getFoodsCollection().insert_one(): Цей рядок коду відповідає за вставку нового продукту в колекцію products бази даних MongoDB. Він приймає об'єкт продукту, що містить наступні поля: name – назва продукту; filePath – шлях до зображення продукту; createdAt – дата і час створення запису.

– socketio.emit('database_update', broadcast=True) – після успішного додавання продукту в базу даних, відправляється сигнал через WebSocket з подією database_update. Параметр broadcast=True забезпечує, що цей сигнал буде відправлений всім підключеним клієнтам.

Процес оновлення інтерфейсу користувача. Веб-додаток, створений на React, підписується на подію database_update, яка надходить від серверу. Після отримання цього сигналу, додаток викликає функцію для отримання оновлених даних рецептів з сервера і відповідно оновлює відображення карток рецептів.

API-метод для отримання рецептів. Цей метод обробляє HTTP-запити до шляху /recipes з методом GET, повертаючи список рецептів у форматі JSON. Далі наведено детальне пояснення кожного рядка коду.

```
@app.route("/recipes", methods=['GET'])
def getRecipes():
```

Опис методу:

– @app.route("/recipes", methods=['GET']): Цей декоратор визначає маршрут для обробки HTTP-запитів. Він вказує, що функція getRecipes буде викликана при отриманні запиту на /recipes з методом GET;

– def getRecipes():: Це визначення функції getRecipes, яка обробляє запит.

– getRecipesCollection(): Виклик функції, яка отримує колекцію рецептів з бази даних;

– loadRecipesCollection(): Виклик функції, яка завантажує і обробляє дані з колекції рецептів, оновлюючи глобальну змінну recipesCollection;

– listRecipesCollection = list(recipesCollection): Перетворює глобальну змінну recipesCollection, що містить рецепти, у список для подальшої обробки;

– recipesData = []: Ініціалізація порожнього списку, який буде містити об'єкти рецептів у форматі, готовому до серіалізації у JSON;

– for doc in listRecipesCollection:: Початок циклу, який ітерує по кожному рецепту у списку listRecipesCollection;

– recipe = { ... }: Створює словник, що містить дані про рецепт, з полями, конвертованими у зручний для серіалізації формат;

– 'productsNeeded': {key: doc['productsNeeded'][key] for key in doc['productsNeeded']}: Створює словник з продуктами, необхідними для приготування, і їх кількістю;

– 'foodsAvability': {key: doc['foodsAvability'][key] for key in doc['foodsAvability']}: Створює словник з доступними продуктами і їх кількістю;

– return jsonify(recipesData): Серіалізує список recipesData у формат JSON і повертає його як відповідь на HTTP-запит.

Функція generateFoodsDict виконує важливу роль у підготовці даних про продукти для подальшого використання в додатку. Її завдання полягає у створенні словника, де ключами є типи продуктів, а значеннями – списки продуктів цього типу, наявних у базі даних. Функція виконує наступні кроки:

– викликає метод getFoodsCollection().find(), щоб отримати всі документи з колекції продуктів у базі даних;

– ініціює цикл, що проходить по кожному продукту (документу) у колекції;

– отримує значення поля name з кожного документа, яке представляє тип продукту;

– перевіряє, чи існує цей тип продукту у глобальному словнику groupedFoodsDict. Якщо ні, то створює новий ключ у словнику для цього типу продукту, значенням якого є порожній список.

– додає поточний продукт до відповідного списку у словнику groupedFoodsDict:

```
def generateFoodsDict():
    foods = getFoodsCollection().find()
    for food in foods:
        foodType = food["name"]
        if foodType not in groupedFoodsDict:
            groupedFoodsDict[foodType] = []
        groupedFoodsDict[foodType].append(food)
```

Діаграма послідовності роботи функції generateFoodsDict() подається на рис. 4.7.

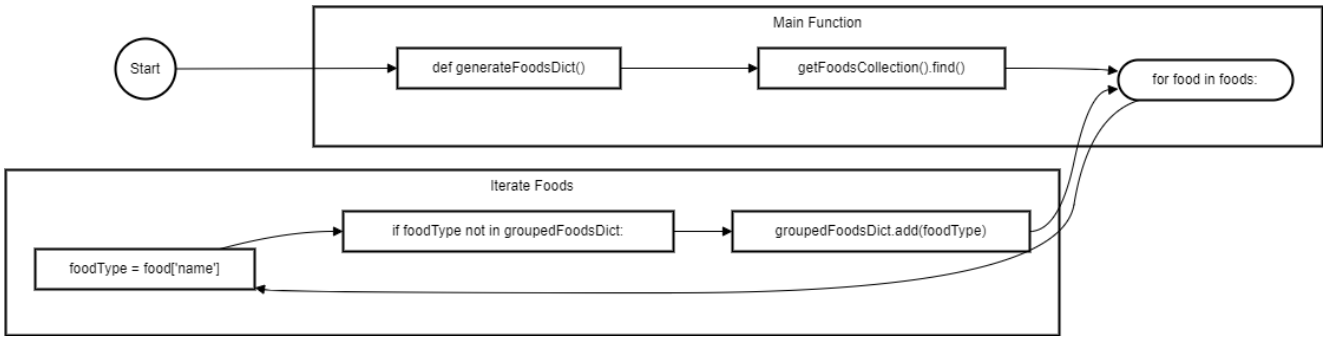

Рисунок 4.7 – Діаграма послідовності роботи функції generateFoodsDict()

Таким чином, ця функція створює структуру даних, яка дозволяє швидко отримати всі продукти певного типу з бази даних, що значно полегшує доступ до цієї інформації під час роботи додатку. Це особливо корисно при обробці рецептів, де необхідно визначити наявність та кількість потрібних інгредієнтів.

Функція `loadRecipesCollection` відіграє ключову роль у підготовці та оновленні даних про рецепти, включаючи інформацію про доступність необхідних інгредієнтів. Ця функція викликається щоразу, коли потрібно оновити картки рецептів на вебсайті через вебсокет. Її робота складається з кількох важливих етапів.

Ініціалізація циклу по всіх рецептах:

```
for r in recipes:
    ...
```

Функція починає з ітерації по всіх рецептах, які зберігаються у глобальній змінній `recipes`. Кожен рецепт представлений як словник. Ініціалізація поля доступності інгредієнтів:

```
r["foodsAvability"] = {}
```

Для кожного рецепту створюється нове поле `foodsAvability`, яке буде зберігати інформацію про наявність інгредієнтів для цього рецепту. Ітерація по необхідних продуктах для кожного рецепту:

```
for key, value in r["productsNeeded"].items():
    ...
```

Для кожного рецепту відбувається ітерація по всіх інгредієнтах, які необхідні для його приготування. Ключ `key` представляє назву продукту, а значення `value` - необхідну кількість цього продукту.

Перевіряється, чи існує продукт у глобальному словнику `groupedFoodsDict`, який був згенерований функцією `generateFoodsDict`:

```
if key in groupedFoodsDict.keys():
```

Розраховується кількість доступних та необхідних продуктів:

```
countOfProduct = len(groupedFoodsDict[key])
countOfneededProduct = float(r["productsNeeded"][key])
```

Якщо продукт існує у базі даних, то обчислюється кількість доступних продуктів (`countOfProduct`) і кількість необхідних продуктів (`countOfneededProduct`).

Оновлюється інформація про доступність продуктів:

```
if countOfProduct < countOfneededProduct:
    r["foodsAvability"][key] = [countOfProduct, countOfneededProduct]
else:
    r["foodsAvability"][key] = [countOfneededProduct, countOfneededProduct]
```

Якщо кількість доступних продуктів менша за необхідну, то в `foodsAvability` додається запис з фактичною кількістю доступних продуктів та необхідною кількістю. Якщо ж доступних продуктів достатньо, то обидва значення будуть рівні необхідній кількості продукту.

Оновлюється інформація про відсутні продукти:

```
else:
```

```
r["foodsAvability"].update({key:[0,float(r["productsNeeded"][key])]})
```

Якщо продукт відсутній у базі даних, то в `foodsAvability` додається запис з нульовою кількістю доступних продуктів та необхідною кількістю.

Додається оновлений рецепт до колекції рецептів:

```
recipesCollection.append(r)
```

Після обробки всіх інгредієнтів для поточного рецепту, оновлений рецепт додається до глобального списку `recipesCollection`.

Ця функція забезпечує актуалізацію інформації про рецепти, включаючи дані про доступність необхідних інгредієнтів, що дозволяє вебдодатку відображати користувачам актуальні рецепти з урахуванням наявних продуктів.

Функція `getProductsFilteredByTypeSortedByDateAsc` виконує важливу роль у відборі продуктів з бази даних. Її основне завдання полягає в тому, щоб отримати продукти певного типу та відсортувати їх за датою створення у порядку зростання. Ця функція часто використовується для відображення найсвіжіших продуктів у рецептах. Ось детальний опис роботи функції:

Параметри функції:

```
def getProductsFilteredByTypeSortedByDateAsc(type: str):
```

Функція приймає один параметр `type` – це рядок, що представляє тип продукту, який потрібно знайти у базі даних.

Отримуються колекції продуктів:

```
return getFoodsCollection().find({"name":type})
```

Функція `getFoodsCollection()` повертає колекцію продуктів з бази даних. Потім виконується пошук продуктів, що відповідають заданому типу, використовуючи фільтр `{"name": type}`. Цей фільтр означає, що з бази даних будуть вибрані лише ті документи, у яких поле `name` відповідає значенню змінної `type`.

Сортуються продукти за датою створення:

```
.sort("createdAt", pymongo.ASCENDING) ...
```

Після фільтрації продуктів за типом, результати сортуються за датою створення у порядку зростання. Поле `createdAt` містить дату та час створення

продукту у базі даних. Використання `pymongo.ASCENDING` забезпечує сортування від найстаріших до найновіших записів.

Порівнюється реезультат. Функція повертає курсор, який містить відсортовані та відфільтровані документи з бази даних. Курсор дозволяє виконувати ітерації по результатах запиту та обробляти кожен знайдений документ.

У підсумку, функція `getProductsFilteredByTypeSortedByDateAsc` надає зручний спосіб отримання продуктів певного типу з бази даних, відсортованих за датою створення у порядку зростання. Це дозволяє легко отримати найсвіжіші продукти для використання у рецептах або інших частинах додатку.

Функція `loadRecipesCollectionWithProducts` виконує обробку рецептів для їх зв'язування з наявними продуктами.

Ось подробний опис роботи цієї функції. Ітерація по рецептах:

```
for recipe in recipesCollection:
```

Цикл перебирає кожен рецепт у колекції `recipesCollection`.

Створення словника для зберігання продуктів:

```
productObjectsDict = recipe["productObjects"] = {} ...
```

Для кожного рецепту створюється порожній словник `productObjectsDict`, який буде використовуватися для зберігання продуктів, необхідних для цього рецепту.

Ітерація по продуктам, необхідним для рецепту:

```
for foodType, valueCollection in recipe["foodsAvability"].items():
```

Цикл перебирає кожен тип продукту та його кількість, які необхідні для даного рецепту, зі словника `foodsAvability`.

Створення списку для кожного типу продукту:

```
productObjectsDict[foodType] = [] ...
```

Для кожного типу продукту створюється порожній список у `productObjectsDict`.

Отримання необхідних продуктів з бази даних:

```
foods = getProductsFilteredByTypeSortedByDateAsc(foodType) ...
```

Викликається функція `getProductsFilteredByTypeSortedByDateAsc`, яка повертає відсортований список продуктів певного типу з бази даних.

Ітерація по кількості необхідних продуктів:

```
for i in range(math.ceil(valueCollection[0])): ...
```

Цикл перебирає кількість продуктів, необхідних для рецепту, використовуючи округлення вгору за допомогою `math.ceil`.

Додавання продуктів до списку:

```
productObjectsDict[foodType].append(foods[i]) ...
```

Кожен знайдений продукт додається до відповідного списку у `productObjectsDict`. Ця функція дозволяє зв'язати кожен рецепт із списком продуктів, які необхідні для його приготування. Вона забезпечує актуальність даних про наявні продукти та їх кількість, що є важливим для користувачів, які шукають рецепти веб-додатку.

Функція `sortRecipesByAvabilityDesc` сортує колекцію рецептів за наявністю продуктів для приготування:

```
def sortRecipesByAvabilityDesc():
    sortedColl = sorted(recipesCollection, key=lambda recipe: sum(food[0] for food in recipe["foodsAvability"].values()), reverse=True)
    return sortedColl
```

Діаграма роботи функції sortRecipesByAvabilityDesc() подається на рис. 4.8.

Короткий опис її роботи:

– використовується функція `sorted`, яка сортує колекцію рецептів за вказаним ключем;

– ключ сортування визначається за допомогою lambda-функції, яка обчислює суму кількостей наявних продуктів для кожного рецепту;

– рецепти сортуються в порядку спадання суми наявних продуктів;

– відсортована колекція рецептів повертається як результат функції.

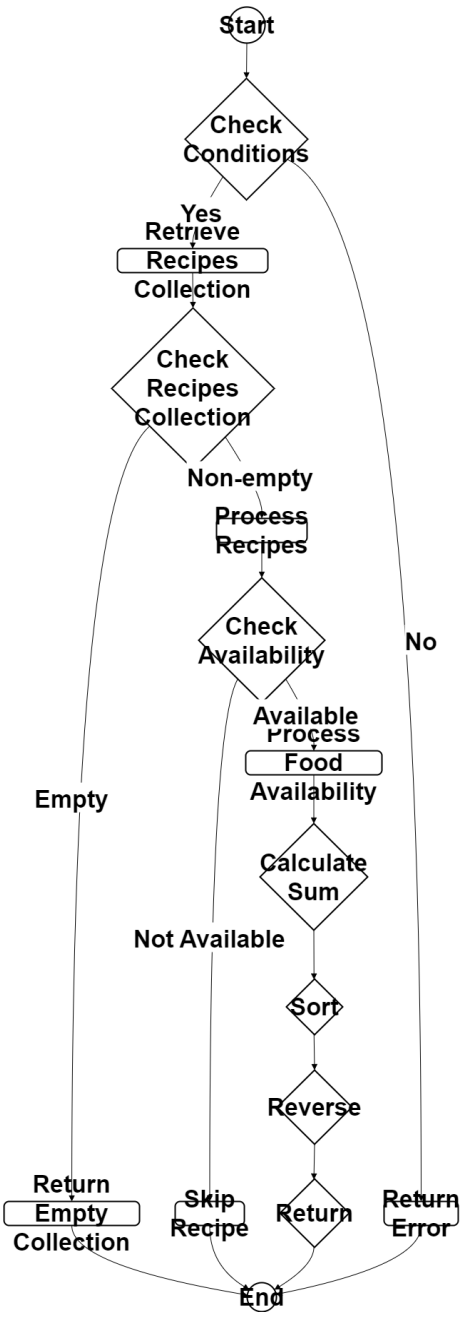

Рисунок 4.8 – Діаграма роботи функції sortRecipesByAvabilityDesc()

Ця функція допомагає забезпечити, щоб найбільш доступні для приготування рецепти відображалися першими у списку:

```
if __name__ == '__main__':
    thread = threading.Thread(target=startCaptureProcess)
    hread.start()
    generateFoodsDict()
    loadRecipesCollection()
    app.run(debug=True)
```

Порядок виконання головного скрипта:

– створюється новий потік `thread`, в якому запускається функція `startCaptureProcess`. Це потрібно для паралельного виконання процесу захоплення відео та розпізнавання об'єктів;

– викликається функція `generateFoodsDict()`, яка створює словник `groupedFoodsDict`, що містить інформацію про наявні продукти в базі даних;

– запускається веб-додаток за допомогою функції `app.run(debug=True)`. В цьому режимі додаток запускається у режимі налагодження, що дозволяє отримувати додаткову інформацію про помилки та відлагоджувати додаток під час розробки.

Функція getRecipe(id) забезпечує можливість отримувати детальну інформацію про конкретний рецепт за його унікальним ідентифікатором через HTTP GET запити до маршруту /recipe/<id>.

– отримує ідентифікатор рецепту з параметру запиту URL;

– фільтрує колекцію recipesCollection для знаходження рецепту з відповідним _id;

– для кожного типу продукту, необхідного для рецепту, визначає найстаріші екземпляри цього продукту з бази даних;

– вибирає потрібну кількість найстаріших продуктів кожного типу, які потрібні для приготування рецепту;

– конвертує кожний _id продукту у строковий формат для зручності обробки на клієнтській стороні;

– підготовлює об'єкт recipeObj, який містить всю інформацію про рецепт у необхідному для відображення форматі;

– повертає об'єкт recipeObj у вигляді JSON через Flask API для подальшого відображення на сторінці.

Функція ініціалізація productObjects – створює порожній словник productObjects, який буде містити інформацію про продукти, використані у рецепті:

```
loadedRecipe["productObjects"] = {}
```

Отримання продуктів за типом і сортування. Для кожного типу продукту з foodsAvability викликає функцію getProductsFilteredByTypeSortedByDateAsc, яка повертає відсортовані за датою продукти з бази даних:

```
    for foodType, valueCollection in loadedRecipe["foodsAvability"].items():
        loadedRecipe["productObjects"][foodType] = []
```

```
        foods = getProductsFilteredByTypeSortedByDateAsc(foodType)
        for i in range(math.ceil(valueCollection[0])):
            loadedRecipe["productObjects"][foodType].append(foods[i])
```

Підготовка вихідного об'єкту рецепту. Створює об'єкт recipeObj, який містить всю необхідну інформацію про рецепт для відображення:

```
    recipeObj = {
        '_id': str(ObjectId(loadedRecipe['_id'])),
        'name': loadedRecipe['name'],
        'servings': loadedRecipe['servings'],
        'timeToPrepare': loadedRecipe['timeToPrepare'],
        'description': loadedRecipe['description'],
        'productsNeeded': {key: loadedRecipe['productsNeeded'][key] for key
in loadedRecipe['productsNeeded']},
        'foodsAvability': {key: loadedRecipe['foodsAvability'][key] for key
in loadedRecipe['foodsAvability']},
        'instruction': loadedRecipe['instruction'],
        'filePath': loadedRecipe['filePath'],
        'productObjects': {key: loadedRecipe['productObjects'][key] for key
in loadedRecipe['productObjects']},
    }
```

Послідовність процесу виконання рецепту подається на рис. 4.9.

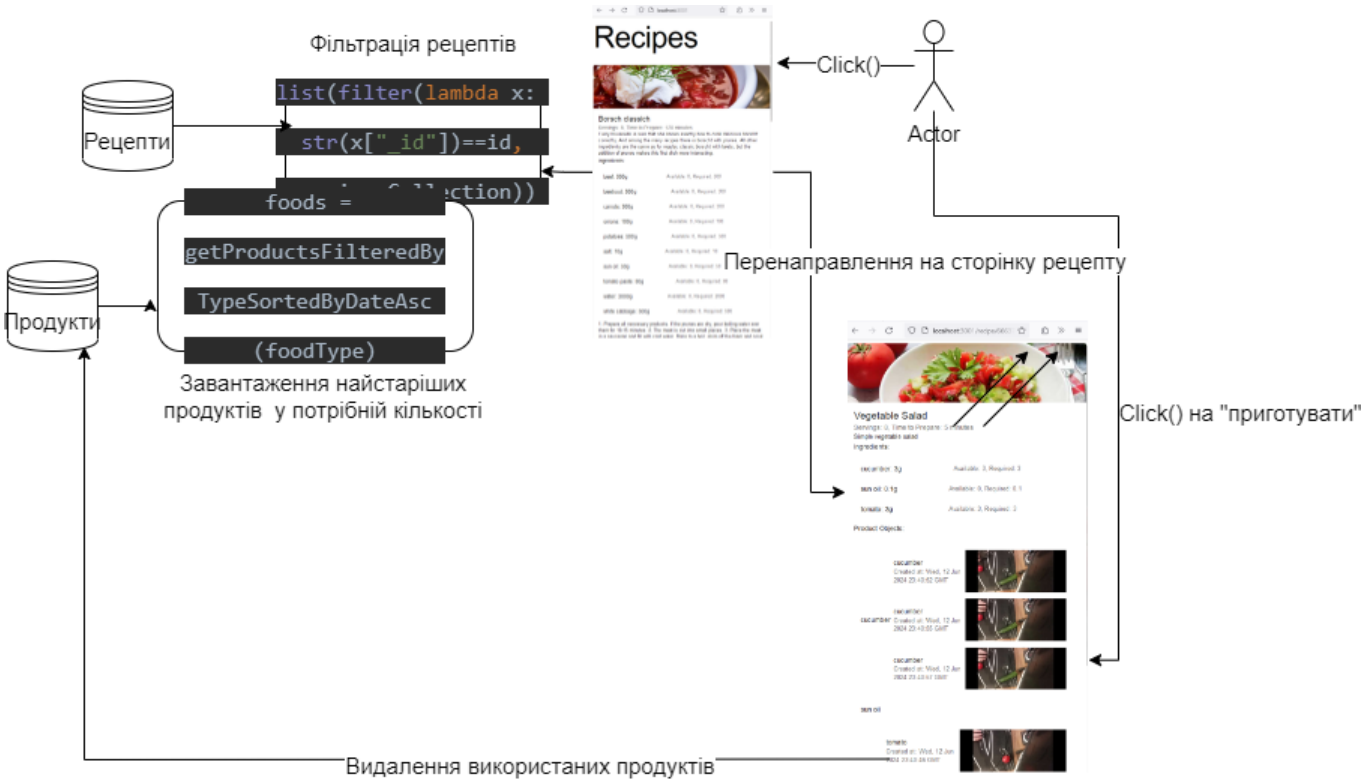

Рисунок 4.9 – Послідовність процесу виконання рецепту

Ця функція можливість отримувати детальну інформацію про конкретний рецепт за його унікальним ідентифікатором через HTTP GET запити до маршруту /recipe/<id>.

4.5 Реалізація веб-інтерфейсу клієнтської частини рекомендаційної системи з приготування страв

Веб-інтерфейс клієнтської частини є односторінковим додатком (SPA), реалізованим за допомогою фреймворку React та бібліотеки Material-UI для створення сучасного інтерфейсу користувача. Додаток відображає рецепти у вигляді карток на головній сторінці та автоматично оновлюється при зміні даних у базі даних за допомогою WebSocket (Socket.IO).

Проект складається з таких основних компонентів і файлів:

– головний компонент додатку (App.jsx). Відповідає за отримання та відображення списку рецептів. Підключається до WebSocket для отримання оновлень бази даних в режимі реального часу. Використовує компонент RecipeCard для відображення окремих рецептів у вигляді карток;

– компонент RecipeCard.jsx. Відповідає за відображення інформації про рецепт у вигляді картки. Включає зображення, назву, кількість порцій, час приготування, опис, інгредієнти та інструкцію;

– компонент FullRecipeCard. Додано новий компонент FullRecipePage, який відображає повну інформацію про рецепт при переході за маршрутом /recipe/:id. Це дозволяє відкривати конкретний рецепт на окремій сторінці;

– файли стилів та конфігурацій. Використовуються для налаштування зовнішнього вигляду додатку та збірки проекту.

Бібліотеки та інструменти:

– React – Фреймворк для побудови користувацьких інтерфейсів;

– Material-UI – Бібліотека компонентів для створення сучасних інтерфейсів;

– Socket.IO – Бібліотека для реалізації WebSocket з'єднання;

– Fetch API –Використовується для отримання даних з бекенду.

Головний компонент додатку:

```
const RecipePage = () => {
```

```
  const [recipes, setRecipes] = useState([]);
  const socket = io('http://127.0.0.1:5000'); // Встановлюємо з’єднання вебсокет
```

Головний компонент додатку містить:

– RecipePage: Функціональний компонент, який відповідає за відображення сторінки з рецептами;

– useState: Використовується для створення стану recipes, який зберігає масив рецептів;

– socket: Створює WebSocket з'єднання з сервером за допомогою socket.io-client.

Використання useEffect для отримання даних та підписки на WebSocket:

```
useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('http://127.0.0.1:5000/recipes');

        const dataaa = await response.json();
        setRecipes(data);
      } catch (error) {
        console.error('Error fetching recipes:', error);
      }
    };

    fetchData();
```

Коспоненти useEffect:

– useEffect – використовується для виконання побічних ефектів, таких як отримання даних з серверу після першого рендеру компонента;

– fetchData – асинхронна функція для отримання рецептів з серверу. Вона використовує fetch для відправки HTTP-запиту на сервер;

– setRecipes – оновлює стан recipes отриманими даними.

Підписка на події WebSocket:

```
    socket.on('database_update', () => {
      console.log('Received database update');
      fetchData(); // Вызываем fetchData при получении события databaseUpdate
    });
    return () => {
      socket.off('database_update');
    };
  }, []);
```

– socket.on('database_update') – підписується на подію database_update, яка надходить від сервера при оновленні бази даних;

– fetchData: Викликається при отриманні події database_update для оновлення списку рецептів;

– socket.off('database_update') – відписується від події при розмонтуванні компонента, щоб уникнути утворення непотрібних підписок.

Рендеринг компоненту:

```
return (
    <div>
      <Typography variant="h1" gutterBottom>
        Recipes
      </Typography>
      <Grid container spacing={3}>
        {recipes.map(recipe => (
          <Grid item key={recipe._id} xs={12} sm={6} md={4}>
            <RecipeCard recipe={recipe} />
          </Grid>
        ))}
      </Grid>
    </div>
  );
};

export default RecipePage;
```

– return – повертає JSX, який визначає інтерфейс компонента;

– Typography – омпонент Material-UI, використовується для відображення заголовка сторінки;

– Grid: Компонент Material-UI, використовується для створення сітки з картками рецептів;

– recipes.map: Ітерує через масив рецептів та створює RecipeCard для кожного рецепту;

– RecipeCard: Компонент для відображення інформації про окремий рецепт.

На рис. 4.10 зображено інтерфейс відображення карток рецептів.

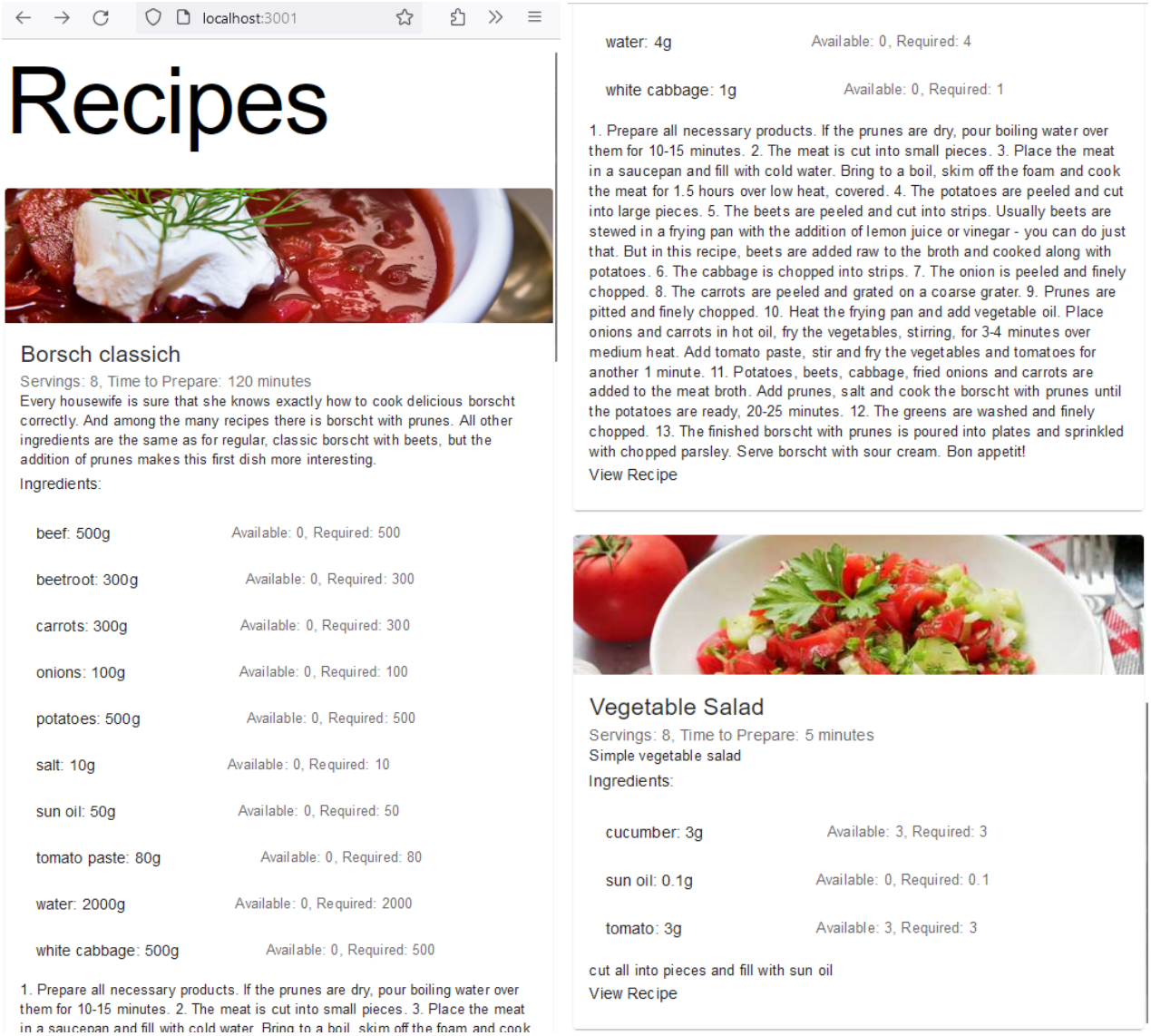

Рисунок 4.10 – Інтерфейс відображення карток рецептів

До було додано маршрутизацію – бібліотеку react-router-dom для управління маршрутизацією. Це дозволяє створювати динамічні маршрути і перемикатися між різними сторінками веб-додатку:

```
import { BrowserRouter, Route, Routes } from 'react-router-dom';
```

Створено сторінку FullRecipePage. Додано компонент FullRecipePage, який відображає повну інформацію про рецепт при переході за маршрутом /recipe/:id. Це дозволяє відкривати конкретний рецепт на окремій сторінці.

Ця сторінка також відображає топ старих продуктів, що підходять до обраного рецепту. Таким чином, користувач наочно зрозуміє, які продукти вже мають найкоротший строк придатності. Для цього використовуються дані з сервера про

продуктові об'єкти, що відповідають обраному рецепту. Кожен продукт в списку має назву, дату створення і зображення, які також завантажуються з сервера. Для завантаження даних про рецепт і його продуктові об'єкти використовуються асинхронні HTTP запити з використанням fetch.

Для переходу до конкретного рецепту, на нього потрібно клікнути. React Router використовується для переходу на FullRecipePage за допомогою динамічного маршруту, який включає ідентифікатор рецепту.

Продукти на сторінці рецепту відображаються таким чином:

```
<List>
          {recipe.productObjects[product].map((item) => (
            <ListItem key={item._id}>
              <ListItemText primary={item.name} secondary={`Created at:
${item.createdAt}`} />
              <CardMedia
                component="img"
                height="100"
                image={`/products/${item.filePath}`}
                alt={item.name}
              />
            </ListItem>
          ))}
        </List>
```

При кліці на кнопку «Приготувати», обрані продукти для рецепту видаляються з бази даних.

На рис. 4.11 наведено сторінку відображення рецепту салату.

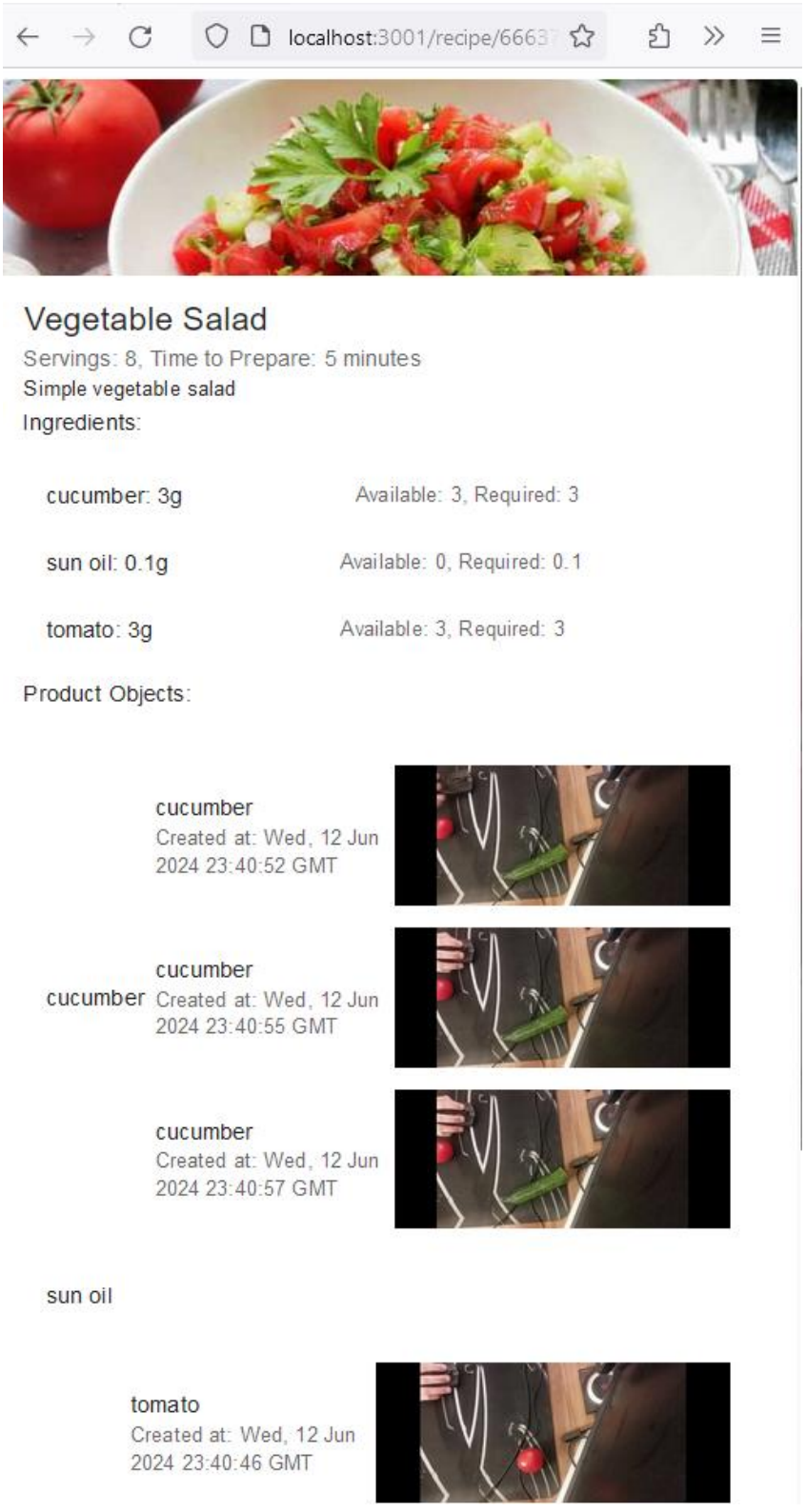

Рисунок 4.11 – Інтерфейс відображення рецепту з фотографіями найстаріших продуктів, потрібних для приготування рецепту

4.6 Проблеми, що виникли під час реалізації рекомендаційної системи з приготування страв

При розробці та дослідженні рооботи виникали різні проблем. Невеликий перелік цих пролем:

– проблема сумісності версій бібліотек. Оскільки для тренування моделі на великій кількості класів продуктів займає багато часу і ресурсів, провести її тренування у сервісі Google Colab було неможливо. При використанні близько 40 імпортів з різних бібліотек постійно виникали проблеми сумісності версій, оскільки бібліотеки для розробки додатків з використанням штучного інтелекту оновлюються. Також ці оновлення торкаються не тільки сумісності версій, а й зміни принципів роботи бібліотек, внаслідок чого документації постійно застарівають і стають неактуальними;

– проблема ресурсів. Відеокарта NVIDIA GTX 1080 Ti з 3,584 CUDA ядрами та 11 гігабайтами відеопам’яті GDDR5X обробляла 30 епох навчання близько 10 годин. Внслідок чого розробка дуже сильно сповільнюється. Також, живе розпізнавання зображень на тренованій моделі навантажує Raspberry Pi 5 на 99% відсотків;

– проблема обладнання. Для запуску проекту на реальному обладнанні придбані міні-комп’ютер Raspberry Pi 5, систему охолодження, блок живлення та оригінальну відеокамеру та Micro-HDMI кабель;

В процесі дослідження кваліфікаційної роботи з додаткового навчання моделі класифікації продуктів виникли значні труднощі, які вплинули на робочий процес та результати. Проблеми з сумісністю версій бібліотек, обмеженість ресурсів для тренування моделі та необхідність додаткового обладнання стали важливими факторами, що вплинули на швидкість розробки та якість отриманих результатів. Незважаючи на ці труднощі, дослідження дозволило набути значного досвіду у роботі зі штучним інтелектом та вирішити цільову задачу класифікації продуктів, хоча із значною затримкою в часі. Висновок полягає в тому, що вирішення складних завдань у галузі машинного навчання вимагає великих зусиль, обчислювальних ресурсів та вміння працювати зі складними технічними проблемами.

ВИСНОВКИ

У даній науковій роботі було проведено аналіз предметної області, пов’язаної з розумними холодильниками, методами класифікації продуктів та алгоритмами автоматичних рекомендацій. Робота мала на меті розробити та реалізувати систему, яка б забезпечувала класифікацію продуктів, а також надавала користувачам рекомендації щодо покупки продуктів на основі їхніх попередніх вподобань.

У першому розділі проведений детальний аналіз тематичного поля дослідження, включаючи огляд існуючих систем розумних холодильників та порівняння їхніх функціональних можливостей. Також сформульовано постановку задачі та визначено методи та інструменти для її вирішення.

У другому розділі були розроблені та навчені моделі глибинного навчання для класифікації продуктів на основі зображень у середовищі Jupiter Lab. Було проведено створення моделі, її навчання та використання у Python-додатку для класифікації об’єктів та запуску процесу захоплення зображень. Після підготовки даних розроблена модель глибинного навчання для класифікації продуктів. Для цього використана нейронна мережа з архітектурою, оптимізованою для розпізнавання об'єктів на зображеннях. Навчання моделі проводилося на наборі зображень продуктів, це дозволило реалізувати функцію автоматичного визначення продуктів на зображеннях.

У третьому розділі досліджені та обрані алгоритми для створення автоматичних рекомендацій, зокрема методи асоціацій та спільної фільтрації. Проведено порівняння їхнього використання для надання неперсоналізованих та персоналізованих рекомендацій користувачам. Початковий аналіз включав вивчення неперсоналізованих та персоналізованих методів рекомендацій. Зокрема, був розглянутий метод асоціацій для неперсоналізованих рекомендацій та метод спільної фільтрації для персоналізованих рекомендацій. Зазначено, що обидва методи мають свої переваги та недоліки, і вибір конкретного методу залежить від функцій та даних конкретної системи рекомендацій.

У четвертому розділі описано архітектурні особливості розробленого додатку, зокрема зберігання даних у СУБД MongoDB, вибір технічного рішення та реалізацію серверної частини веб-додатка з використанням Python Flask та React для створення бекенду та фронтенду відповідно. Визначено архітектуру серверної частини додатку, її компоненти та функції. Також визначена архітектура веб-частини системи, описано реалізацію відображення рецептів та їхніх складових для користувачів.

Отже, на основі проведеного аналізу та реалізації можна зробити висновок про досягнення поставленої мети з дослідження та розробки системи. Тестування функції системи класифікації продуктів та наданні користувачам рекомендацій підтвердило, що система готова до експлуатації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. MongoDB Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/.

2. Kaehler A., Bradski G. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, Incorporated, 2008.

3. Klein E., Bird S., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Incorporated, 2009.

4. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 p.

5. Aggarwal C. C. Recommender Systems. Cham : Springer International Publishing, 2016. URL: https://doi.org/10.1007/978-3-319-29659-3 (date of access: 28.03.2024).

6. Chakraborty A. Perceptron Collaborative Filtering. International Journal for Research in Applied Science and Engineering Technology. 2023. Vol. 11, no. 2. P. 437–447. URL: https://doi.org/10.22214/ijraset.2023.49044 (date of access: 28.03.2024).

7. Crompton D. Keras Python : Keras Incremental Training: Learning Rate Keras. Independently Published, 2021.

8. Deep learning based Food Recognition using Tensorflow / K. R. Abirami et al. Journal of Physics: Conference Series. 2021. Vol. 1916, no. 1. P. 012149. URL: https://doi.org/10.1088/1742-6596/1916/1/012149 (date of access: 28.03.2024).

9. Konstan J. A., Ekstrand M. D., Riedl J. T. Collaborative Filtering Recommender Systems. Now Publishers, 2011. 108 p.

10. RecipeDB: a resource for exploring recipes / D. Batra et al. Database. 2020. Vol. 2020. URL: https://doi.org/10.1093/database/baaa077 (date of access: 28.03.2024).

11. Szeliski R. Computer Vision. Cham : Springer International Publishing, 2022. URL: https://doi.org/10.1007/978-3-030-34372-9 (date of access: 28.03.2024).

12. Python tutorials for developers of all skill levels. [Електронний ресурс]. – Режим доступу: https://realpython.com/

13. Agarwal D. K., Chen B.-C. Statistical Methods for Recommender Systems. Cambridge University Press, 2015.

14. Ameisen E. Building Machine Learning Powered Applications: Going from Idea to Product. O'Reilly Media, Incorporated, 2020. 260 p.

15. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2016. 758 p.

16. Burkov A. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019. 160 p.

17. Crompton D. Keras Python : Keras Incremental Training: Learning Rate Keras. Independently Published, 2021.

18. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated, 2022.

19. Griffiths D., Griffiths D. React Cookbook: Recipes for Mastering the React Framework. O'Reilly Media, Incorporated, 2021. 500 p.

20. Grinberg M. Flask Web Development: Developing Advanced Web Applications With Python / ed. by A. MacDonald. 2nd ed. Beijing : O’Reilly Media, 2018. 316 p.

21. Lombardi A. WebSocket: Lightweight Client-Server Communications. O'Reilly Media, Incorporated, 2015.

22. McKinney W. Python for Data Analysis. O'Reilly Media, Incorporated, 2022.

23. Publishing N. Machine Learning Python: Beginner's Guide to Machine Learning with Python. Introduction to Machine Learning Using Python. Independently Published, 2019.

24. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow, 2nd Edition. Packt Publishing, 2017. 622 p.

25. Посібник з мови програмування Python. [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/python/tutorial/

26. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Incorporated, 2023.