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)_____

Розробка та дослідження методу управління великими даними в реальному часі

(тема)

Виконав:

студент __II__ курсу, групи ІТПм-22-2_

_____Гончаренко Т.А._____

(прізвище, ініціали)

Спеціальність _____

_____122 «Комп'ютерні науки»__

(код і повна назва спеціальності)

Тип програми ___освітньо-професійна_

(освітньо-професійна або освітньо-наукова)

Освітня програма _____

_Інформаційні технології проектування

(повна назва освітньої програми)

Керівник: _____проф. Перова І.Г.____

(посада, прізвище, ініціали)

Допускається до захисту

Зав.кафедри _____ _____Гребеннік І.В._____

(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Системотехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 122 «Комп'ютерні науки»
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційні технології проектування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"_____" _____20___р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гончаренку Тарасу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методу управління великими даними в реальному часі
затверджена наказом університету від " ___ " _____ 2023 р. № _____.
2. Термін здачі студентом роботи до екзаменаційної комісії _____
3. Вихідні дані до роботи Мова програмування – Python. Бібліотеки – BeautifulSoup, Flask, Jango. Мова програмування – JavaScript, CSS, Хмарний сервіс – Google Cloud. Командна оболонка для обчислень – Visual Studio Code. Стандартизована мова розмітки документів для перегляду вебсторінок у браузері – HTML. Спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду – CSS. Вбудована СУБД – SQLite.
4. Перелік питань, що потрібно опрацювати у роботі
1. Опис сучасного стану розвитку інформаційних систем та технологій управління великими даними в реальному часі
2. Аналіз застосування досліджуваних методів та технологій в існуючих системах.
3. Постановка задачі на дослідження.
4. Удосконалення методу управління великими даними в режимі реального часу на прикладі вебзастосунку.

5. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1).

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН
09 жовтня 2023 р. - 03 січня 2024 р.

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Опис сучасного стану розвитку інформаційних систем та технологій управління великими даними в реальному часі	09.10.23-16.10.23	
2.	Аналіз застосування досліджуваних методів та технологій в існуючих системах.	17.10.23-24.10.23	
3.	Постановка задачі на дослідження.	25.10.23-01.11.23	
4.	Загальні положення удосконалення методу управління великими даними на прикладі вебзастосунку	02.11.23-09.11.23	
5.	Опис методів рішення задачі.	10.11.23-17.11.23	
6.	Визначення основних функцій та вимог вебзастосунку.	18.11.23-25.11.23	
7.	Розробка методу збору та підготовки даних для алгоритму обробки.	26.11.23-08.12.23	
8.	Оформлення висновків.	09.12.23-11.12.23	
9.	Оформлення матеріалів кваліфікаційної роботи	12.12.23-22.12.23	
10.	Подання кваліфікаційної роботи керівникові та її попередній захист	23.12.23-29.12.23	
11.	Подання кваліфікаційної роботи на рецензування	03.01.24	

Дата видачі завдання ___ _____ 20___ р.

Студент _____
(підпис)

Керівник роботи _____проф.Перова І.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 7 рис., 26 джерел.

ВЕЛИКІ ДАНІ, ІНФОРМАЦІЙНІ СИСТЕМИ, ІНТЕРНЕТ РЕЧЕЙ, GOOGLE CLOUD, PYTHON, ВЕБЗАСТОСУНОК, МЕТОД УПРАВЛІННЯ ВЕЛИКИМИ ДАНИМИ, ХМАРНІ СЕРВІСИ, МЕРЕЖА, ВЕБСАЙТ

Метою кваліфікаційної роботи є ознайомлення з галуззю великих даних, огляд та порівняння методів та інструментів управління великими даними, а також представлення способів застосування методу управління великими даними в режимі реального часу.

У ході виконання роботи був проведений детальний огляд особливостей галузі великих даних, її основних положень та проблем, а також методів та інструментів методів та інструментів управління великими даними в режимі реального часу.

Далі було проаналізовано корисність та наведено перелік популярних програмних засобів для використання у досліджуваному методі. Також було представлено концепцію використання хмарних сервісів, на прикладі Google Cloud, його інструментів.

В практичній частині була проведена програмна реалізація удосконалення методу управління великими даними в режимі реального часу із застосуванням хмарних технологій на прикладі вебзастосунку.

Програмна частина розроблялася з використанням мов програмування Python (бібліотеки BeautifulSoup, Django, Flask), HTML, JavaScript, CSS.

ABSTRACT

Master's thesis: 75 pages, 7 figures, 26 sources.

BIG DATA, INFORMATION SYSTEMS, INTERNET OF THINGS, GOOGLE CLOUD, PYTHON, WEB APPLICATION, BIG DATA MANAGEMENT METHOD, CLOUD SERVICES, NETWORK, WEBSITE

The purpose of the qualification work is to familiarize with the field of big data, review and compare methods and tools for managing big data, as well as presenting ways to apply the big data management method in real-time.

In the course of the work, a detailed review of the peculiarities of the big data industry, its main provisions and problems, as well as methods and tools for managing big data in real-time, was conducted.

Then, the utility was analyzed, and a list of popular software tools for use in the researched method was presented. The concept of using cloud services, exemplified by Google Cloud and its tools, was also presented.

In the practical part, a program implementation of improving the method of managing big data in real-time with the use of cloud technologies was conducted using the example of a web application.

The software part was developed using Python programming languages (libraries BeautifulSoup, Django, Flask), HTML, JavaScript, CSS.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

СУБД – система управління базами даних

ER-модель – модель «сутність-зв’язок» (англ., Entity-Relationship model)

SQL – мова структурованих запитів (англ., Structured Query Language)

URL – уніфікований локатор ресурсів (англ., Uniform Resource Locator) IP – інтернет протокол (англ., Internet Protocol)

API – прикладний програмний інтерфейс (англ., Application Programming Interface)

CSS – каскадні таблиці стилів (англ., Cascading Style Sheets)

CMS – система керування вмістом (англ., Content Management System)

ВСТУП

В сучасному світі, де щодня генерується безліч даних, управління великими обсягами інформації в реальному часі стає вирішальним для компаній та організацій. Використання методів управління великими даними дозволяє не лише зберігати та обробляти великі набори даних, але й отримувати з них цінні знання та розуміння.

Великі дані - це не просто масиви інформації, але й складні структури, які постійно змінюються та оновлюються. Із зростанням обсягів інформації, яку щодня створюють користувачі, соціальні мережі, сенсори, транзакції та інші джерела, з'являється потреба в ефективному методі управління цими даними в реальному часі.

Методи управління великими даними включають збір, зберігання, аналіз, візуалізацію та управління даними. Ці методи дозволяють компаніям визначати шаблони поведінки, прогнозувати тренди, робити обґрунтовані рішення та підвищувати ефективність.

Об'єктом дослідження є системи управління великими даними, які забезпечують можливість обробки та аналізу даних в режимі реального часу.

Предметом дослідження є удосконалення методу управління великими даними в режимі реального часу на прикладі вебзастосунку.

Метою роботи є аналіз сучасних методів управління великими даними та розробка рекомендацій щодо їх використання для підвищення продуктивності та ефективності в реальному часі.

У світі, де швидкість інформаційних потоків вимірюється в петабайтах за секунду, здатність швидко обробляти та аналізувати великі обсяги даних стає критичною для успіху. Швидкість реакції на зміни ринку, здатність прогнозувати поведінку споживачів, ефективне управління ресурсами - все це залежить від методів управління великими даними. Таким чином, розробка та впровадження ефективних методів управління даними стає актуальним завданням для багатьох організацій.

Реалізація ефективних методів управління великими даними може принести значні переваги для організацій, включаючи:

- підвищення продуктивності: автоматизація процесу збору та аналізу даних зменшує потребу в ручній роботі та підвищує ефективність процесів;
- поліпшення прийняття рішень: доступ до своєчасної та точної інформації дозволяє краще розуміти ситуацію та робити обґрунтовані рішення;
- прогнозування та аналіз трендів: використання алгоритмів машинного навчання для аналізу великих обсягів даних допомагає виявляти шаблони та прогнозувати майбутні тренди.

Управління великими даними в режимі реального часу відкриває нові можливості для організацій у всіх сферах діяльності. Від фінансового сектору до охорони здоров'я, від роздрібної торгівлі до телекомунікацій - здатність швидко обробляти та аналізувати великі обсяги даних може стати ключовим фактором успіху. Розробка та імплементація ефективних методів та інструментів управління великими даними стає важливим завданням для досягнення конкурентної переваги в сучасному бізнесі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис сучасного стану розвитку інформаційних систем та технологій

Покращення якості життя людини на сьогодні є одним із рушіїв бізнесу. Від рядового менеджера з продажів до транснаціональної корпорації - всі працюють на збільшення розмаїття засобів задоволення зростаючих потреб людей. Виграють обидві сторони: перші отримують доходи за рахунок розширення спектру послуг та товарів, потреба у яких спрогнозована на основі аналізу великих даних, останні - розширюють свою зону комфорту. Змінюються методи ведення бізнесу, поведінка споживачів, самі споживачі. Для збереження конкурентоспроможності підприємства прагнуть в реальному часі дізнаватися, коли клієнти щось купують, де вони купують, і навіть що вони думають перед тим, як зайти в магазин або відвідати Web-сайт [1]. Інформація постійно змінюється, і обсяг її зростає, внаслідок чого слід використовувати засоби, пристосовані для роботи з великими обсягами даних.

Сучасний етап розвитку інформаційних систем (ІС) характеризується інтенсивною інтеграцією технологій, що обробляють великі обсяги даних, та широким розповсюдженням елементів штучного інтелекту, а також включають апаратні засоби, програмне забезпечення, комунікаційні технології, бази даних та інтерфейси для взаємодії з користувачем, тощо. Слід зазначити, що інформаційні системи - це комплексні системи, які об'єднують людей, процеси, дані та технології для збору, обробки, зберігання та розповсюдження інформації, що необхідна для підтримки прийняття рішень, координації та контролю. Все більшого розповсюдження набуває практика використання ІС у дуже різних сферах життя - у бізнесі, управлінні, операційній діяльності, навчанні та багатьох інших. Якщо ще не так давно використання ІС могли дозволити собі лише власники або користувачі потужних комп'ютерних ресурсів, то на сьогодні навіть портативні, карманні, а подекуди, і більш мініатюрні пристрої, вже спроможні збирати та обробляти великі обсяги

інформації. Але мініатюризація не є єдиними шляхом розвитку ІС на сучасному етапі. Перспективним є напрямок хмарних сервісів, завдяки яким більший обсяг ресурсоємних завдань ІС беруть на себе віддалене устаткування та застосунки із використанням штучних нейронних мереж (ШНМ).

У І кварталі 2023 року витрати на послуги хмарних інфраструктур зросли в річному обчисленні на 19% до $66,4 млрд, повідомляється в новому дослідженні Canalys. Як зазначили аналітики, хмарні технології залишаються одним із найшвидше зростаючих сегментів ІТ-ринку.

За оцінками Canalys, лідирувала у звітному періоді платформа Amazon Web Services (AWS), на частку якої припало 32% загальних витрат при зростанні на 16% у річному обчисленні, що вперше нижче за позначку в 20%. На тлі тренду на впровадження генеративного ШІ у існуючі та нові продукти, AWS запустила Amazon Bedrock, набір хмарних інструментів ШІ.

Друге місце - у Microsoft Azure з часткою ринку 23% при зростанні 27% у річному обчисленні. Microsoft оголосила про доступність сервісів Azure OpenAI, що дозволило більшій кількості клієнтів створювати високопродуктивні ШІ-сервіси у хмарі Azure. Компанія отримала у звітному періоді на порядок більше (2500+) нових клієнтів Azure OpenAI, включаючи Coursera, Mercedes-Benz та Shell.

Третє місце у Google Cloud Platform, що займає 9% ринку при зростанні у річному обчисленні 30%.[2]

Інформаційні системи та технології є фундаментальною частиною сучасного суспільства, відіграючи ключову роль в інтеграції комп'ютерних технологій із завданнями організації для ефективного виконання бізнес-процесів, підтримки прийняття рішень та надання послуг.

Інформаційні технології охоплюють широкий спектр інструментів, методів і рішень, починаючи від комп'ютерного апаратного забезпечення до програмного забезпечення, баз даних, мереж і хмарних рішень. Вони дозволяють організаціям збирати дані з множини джерел, включно з датчиками і мобільними пристроями, інтернетом речей та іншими електронними

інструментами. ІС використовують ці технології для того, щоб перетворити зібрані дані в корисну інформацію, яка може бути використана для аналізу та оптимізації процесів. Системи управління базами даних, інструменти аналітики великих даних і комплексні програмні рішення уможливлюють цю трансформацію, надаючи можливості для автоматизації рутинних завдань та збільшення продуктивності.

Окрім того, інформаційні системи служать платформами для співпраці та комунікації не тільки в межах окремої організації, але й для зв'язку з глобальними партнерами та клієнтами, що розширює можливості для бізнесу і відкриває нові ринки.

Таким чином, інформаційні системи та технології є не лише набором інструментів, але й стратегічним активом, який дозволяє організаціям пристосовуватися до змін умов ринку, інновацій та еволюційних зрушень в суспільстві та техніці.

Сучасні інформаційні системи відіграють ключову роль у підтримці різноманітних процесів, що відбуваються в організаціях, установах та щоденному житті людей. Ці системи охоплюють широкий спектр функціональності, починаючи від простих задач обробки даних до складних аналітичних вирішень, що залучають великі обсяги інформації та їх інтенсивний обмін. З огляду на швидкість збору та акумуляції даних, важливо відзначити, що розвиток технологій великих даних значною мірою сприяє збільшенню потенціалу інформаційних систем у різних галузях.

Основними функціями сучасних інформаційних систем є збір, зберігання, обробка, аналіз, розподіл та візуалізація даних. Ці системи спроектовані таким чином, що вони можуть ефективно працювати з великими масивами даних, що надходять з різних джерел та в різноманітних форматах. Використання сучасних баз даних, обчислювальних хмар та інших технологій, таких як машинне навчання та штучний інтелект, дозволяє інформаційним системам не лише швидко обробляти великі дані або "Big Data", але й виконувати складні аналітичні процедури для виведення цінних знань і прогнозів.

Деякі бізнес-аналітики тепер вважають дані "новим золотом". Фраза "нове золото - дані" відображає ідею про те, що в сучасному цифровому світі інформація, а точніше здатність збирати, аналізувати та використовувати великі обсяги даних, є надзвичайно цінним ресурсом. Великі дані стали ключовим елементом у багатьох галузях, від бізнесу до науки, від культури до медицини, і забезпечують можливості для глибокого аналізу, кращого прийняття рішень та інновацій.

Для великих даних сьогодні немає строгого визначення. Спочатку ідея полягала в тому, що обсяг інформації настільки зріс, що кількість, що розглядається, вже фактично не містилася в пам'яті комп'ютера, що використовується для обробки, тому інженерам знадобилося модернізувати інструменти аналізу всіх даних. Так з'явились нові технології обробки, наприклад модель MapReduce компанії Google та її аналог з відкритим вихідним кодом - Hadoop від компанії Yahoo. Вони дали можливість керувати набагато більшою кількістю даних, ніж раніше [3].

Під Big Data розуміються обсяги інформації, що є настільки великими та складними, що традиційні методи обробки даних стають недостатньо ефективними для їхнього аналізу та управління. Ця концепція включає не тільки обсяги даних, але й різноманітність і швидкість, більш відомі як три "V": обсяг (Volume), різноманітність (Variety) та швидкість (Velocity), що вперше було сформульовано Дагом Лейні (Doug Laney) у 2001 році в його доповіді для Gartner, де він описав виклики та можливості, пов'язані зі зростанням обсягу, швидкості та різноманітності даних. Однак концепція "великих даних" з часом розвивалася, і до цих трьох характеристик були додані ще дві: достовірність (Veracity) та цінність (Value). Ці дві характеристики є результатом загальної еволюції дискусії про великі дані у професійному співтоваристві. Достовірність стосується надійності та точності даних, що є критично важливим на фоні швидкого зростання обсягу інформації. Цінність акцентує на важливості перетворення великих обсягів даних на корисні інсайти, які можуть приносити реальну вартість бізнесу чи іншим зацікавленим сторонам. У цьому контексті

можна знайти багато різних джерел, які розширюють і доповнюють оригінальну модель "трьох V", включаючи різні наукові статті, аналітичні звіти та книги, які обговорюють різні аспекти великих даних.

Концепція Big Data стала особливо актуальною у час цифрової ери, де щохвилини генеруються терабайти даних. Великі дані трансформують спосіб прийняття рішень у бізнесі, науці, урядових інституціях та інших галузях.

У контексті бізнесу, великі дані використовуються для аналізу споживацького поведінки, оптимізації ланцюгів поставок та підвищення ефективності.

Великі дані відіграють революційну роль у сучасному світі, надаючи можливості для глибшого інсайту та кращого розуміння складних процесів. Водночас, окрім величезного потенціалу та користі від Big Data, вони ставлять перед усіма нові виклики та проблеми, пов'язані з обробкою, зберіганням та захистом цих даних.

Одним з важливих викликів у роботі з великими даними є забезпечення безпеки та конфіденційності, а також питання приватності та етики у контексті збору та аналізу великих обсягів даних.

Великі дані стали особливо актуальними у сучасному цифровому світі завдяки зростанню кількості інформації, що генерується через IoT, соціальні медіа, мобільні технології та інші цифрові платформи. Вони використовуються в різноманітних сферах, включаючи бізнес-аналітику, медицину, наукові дослідження, урядові системи, і, звісно, в області розвитку програмного забезпечення.

Обробка великих даних вимагає застосування складних технологій, таких як хмарні обчислення, алгоритми машинного навчання, штучний інтелект та передові методи статистичного аналізу. Ці технології дозволяють не тільки зберігати та обробляти величезні обсяги даних, але й витягувати з них цінну інформацію, виявляти закономірності, робити прогнози та приймати обґрунтовані рішення.

З огляду на їх здатність до обробки та аналізу величезних обсягів даних, великі дані стають невід'ємною частиною розвитку сучасних технологій і відіграють ключову роль у створенні більш інтелектуальних, ефективних і інноваційних рішень у всіх сферах людської діяльності.

Важливою тенденцією у розвитку інформаційних систем є їх інтеграція з технологіями IoT, яка відкриває нові можливості для автоматизації процесів та оптимізації роботи системи. IoT-пристрої збирають різноманітну інформацію з фізичного світу та перетворюють її в цифрові дані, які потім можуть бути аналізовані інформаційними системами для прийняття обґрунтованих рішень.

Термін "Інтернет речей" був вперше озвучений у 1999 році дослідником RFID-технології Кевіном Ештоном. Сама ж концепція Інтернету речей бере свій початок ще з кінця XIX століття.

Найпоширенішою є думка, що відлік появи Інтернету речей як масового явища потрібно вести з моменту, коли кількість підключених до Інтернету пристроїв переважила кількість населення Землі. Це сталося приблизно між 2008 та 2009 роками.

Основну суть Інтернету речей дуже образно та змістовно описав Блейк Бурнетте на конференції IoT World у 2016 році: "Уявіть собі Facebook для пристроїв. Кожен із них має власну сторінку в соцмережі та може розмістити на стіні інформацію, яку прочитають інші пристрої".

У цьому простому визначенні досить чітко описується комплексність IoT. Адже це і пристрої, які вийшли в Інтернет та взаємодіють між собою, і спосіб підключення – M2M (машина до машини), і великі дані, які генерують ці пристрої та надалі можуть стати основою для прийняття рішень.[4]

Окрім того, сучасні інформаційні системи стають все більш інтерактивними та здатними до самонавчання завдяки застосуванню алгоритмів штучного інтелекту, що дозволяє їм адаптуватися до змінних умов та потреб користувачів. Ці алгоритми можуть аналізувати поведінку користувачів, їхні вподобання та звички, а також забезпечувати персоналізовану інтеракцію.

Однією із актуальних тем покращення життя людини в останній час є ведення здорового способу життя на основі збалансованого харчування. У боротьбі за правильні калорії, білки, вуглеці, жири часто перемагає відсутність вільного часу та достатніх знань для планування закупки продуктів, формування меню та підрахунку кількості спожитого. Продовольча і сільськогосподарська організація (ФАО) в якості спеціалізованої установи ООН у Керівних принципах "Стійке здорове харчування" визначило: стійке здорове харчування – це такий раціон харчування, який допомагає всім аспектам здоров'я і благополуччя людей; не завдає значного тиску на навколишнє середовище; є доступним, недорогим, безпечним і справедливим; а також прийнятним з культурної точки зору [5].

В контексті цієї роботи особлива увага приділяється дослідженню потенціалу інформаційних систем у вирішенні задач пов'язаних з "розумним" управлінням. Подібні системи мають використовувати великі дані для підтримки рішень в реальному часі, що може включати, наприклад, оптимізацію запасів харчових продуктів, планування закупівель, ідентифікацію зіпсованих продуктів або автоматизацію замовлень. Така функціональність вимагає інтеграції різноманітних технологій обробки даних та їх аналітики, а також розвитку спеціалізованих алгоритмів для обробки та аналізу інформації в реальному часі.

З огляду на описані проблему, можна запропонувати вебзастосунок, який її вирішить, дял демонстрації способу удосконалення методу управління великими даними в режимі реального часу. Цікавим є напрямок розробки та впровадження вебзастосунку "Розумний холодильник". І, хоча, є й ті, хто скептично ставиться до розвитку цього напрямку [6], але той факт, що компанія Amazon вже тривалий час займається розробкою подібного продукту [7], свідчить про його високу перспективність.

Вебзастосунок "Розумний холодильник" представляє собою комплексний інструмент, який розширює функціональні можливості традиційного холодильника завдяки використанню передових технологій. Він забезпечує

автоматизацію багатьох процесів, пов'язаних зі зберіганням продуктів, плануванням харчування та покупками. З його допомогою користувачі можуть в реальному часі відстежувати запаси продуктів у своєму холодильнику, отримувати нагадування про необхідність поповнення запасів або про те, що певні продукти незабаром можуть зіпсуватися. Цей продукт також може рекомендувати рецепти на основі того, що зараз є в холодильнику, допомагаючи таким чином мінімізувати відходи і спонукаючи до більш розумного споживання.

"Розумний холодильник" може синхронізуватися з онлайн-магазинами та сервісами доставки продуктів, автоматизуючи процес покупок та планування бюджету на харчування. Використання даних, зібраних з поведінки користувачів, дозволяє програмі адаптувати рекомендації та пропозиції, стаючи з часом все більш персоналізованою. Застосункові функції, такі як підтримка голосових команд та інтеграція з іншими розумними пристроями в домі, роблять взаємодію з холодильником ще більш інтуїтивно зрозумілою та ефективною.

Актуальність дослідження полягає в тому, що із застосуванням методу, який пропонується удосконалити, вебзастосунок "Розумний холодильник" перетворює звичайний побутовий пристрій на інтерактивного помічника, що допомагає користувачам вести здоровий спосіб життя, економити час і гроші, та зменшувати свій екологічний слід, оптимізуючи процес споживання продуктів.

1.2 Аналіз застосування досліджуваних методів та технологій в існуючих системах

Застосування сучасних методів управління великими даними в реальному часі в ІС демонструє різноманіття та глибину їх впливу на сучасні технології та бізнес-процеси. Ці методи знайшли своє застосування в широкому спектрі галузей, починаючи від електронної комерції та закінчуючи охороною здоров'я,

демонструючи свою здатність ефективно обробляти великі обсяги інформації, забезпечуючи при цьому високий рівень продуктивності та точності.

У сфері електронної комерції методи великих даних використовуються для створення глибоко персоналізованих користувацьких досвідів. Компанії аналізують великі масиви даних про покупки клієнтів, їхні перегляди вебсторінок та взаємодію з рекламою, щоб краще розуміти потреби та переваги своїх клієнтів. Це дозволяє пропонувати товари та послуги, які найбільше відповідають інтересам певної групи споживачів, підвищуючи тим самим рівень задоволення клієнтів та збільшуючи продажі.

В галузі охорони здоров'я, великі дані відіграють критичну роль у підвищенні якості медичних послуг. Використання даних з медичних записів, датчиків та інших джерел допомагає медичним фахівцям у діагностиці, плануванні лікування та моніторингу стану пацієнтів. Аналітика великих даних дозволяє виявляти закономірності, які не завжди очевидні при традиційному підході, і використовувати ці знання для розробки ефективніших стратегій лікування та профілактики.

У логістиці та управлінні ланцюгами поставок, методи великих даних дозволяють значно підвищити ефективність процесів. Вони використовуються для аналізу та оптимізації транспортних маршрутів, управління запасами та прогнозування попиту. Це дозволяє компаніям зменшити витрати, знизити час доставки та підвищити задоволеність клієнтів, пропонуючи більш надійні та швидкі послуги.

В контексті IoT великі дані відіграють вирішальну роль у створенні інтелектуальних та автоматизованих систем. Це стосується як розумних будинків, де системи управління використовують дані з датчиків для автоматизації побутових процесів, так і промисловості чи смарт-міст, де аналіз великих даних дозволяє оптимізувати різні процеси і підвищити ефективність роботи обладнання та інфраструктури.

Врешті, стає очевидним, що великі дані та їх обробка в реальному часі відіграють критичну роль у персоналізації та оптимізації взаємодій з клієнтами,

особливо в таких сферах, як електронна комерція. Завдяки глибокому аналізу поведінкових даних, компанії здатні створювати значно більш цільові та ефективні стратегії маркетингу та обслуговування клієнтів. Аналіз показує, що інтеграція методів управління великими даними в реальному часі стає ключовою для інноваційних рішень у різних галузях, відкриваючи нові можливості для розвитку, оптимізації та персоналізації продуктів та послуг.

У рамках розробки вебзастосунку "Розумний холодильник" методи управління великими даними можуть бути застосовані для відстеження та аналізу інформації про харчові продукти, їх терміни придатності, частоту використання та звички споживання користувачів. Такий підхід дозволяє оптимізувати запас продуктів, попереджати про можливе псування та автоматизувати процес замовлення продуктів.

Тенденції у світі, що пов'язані з пандемією COVID-19 та розвитком сучасних технологій в галузі дистанційної торгівлі, мікро-доставки та доставки готової їжі, лише доводять актуальність створення такого продукту.

Розглянемо деякі з тенденцій та переваги продукту.

Під час пандемії COVID-19 однією з ключових рекомендацій стала необхідність мінімізувати особистий контакт. Вебзастосунок "Розумний холодильник" дозволить користувачам уникати походів до магазину та зменшити ризик зараження.

Умови пандемії зробили нас більш свідомими щодо споживання продуктів та мінімізації відходів їжі. Вебзастосунок допоможе контролювати та оптимізувати запаси продуктів, зменшуючи втрати та зайві закупівлі.

Споживачі все частіше обирають онлайн-покупки, включаючи продукти харчування. Вебзастосунок може бути інтегрований з онлайн-сервісами доставки продуктів, що спрощує процес замовлення та доставки, забезпечуючи користувачам точну інформацію про їхні запаси та характеристики.

Мікро-доставка набула популярності завдяки своїй швидкості та зручності. Вебзастосунок може автоматично надсилати замовлення на необхідні

продукти до мікро-доставки, щоб користувач отримував свіжі продукти за декілька годин.

Запит на доставку готової їжі зріс під час пандемії, і вебзастосунок може підтримувати користувачів у замовленні готової їжі без необхідності наявності продуктів для приготування їжі вдома.

Отже, вебзастосунок "Розумний холодильник" має численні переваги і може принести користь як окремій людині, так і суспільству в цілому, сприятиме ефективному управлінню запасами продуктів, зменшенню витрат, збереженню природних ресурсів і споживчій відповідальності. Ця технологія має потенціал покращити якість життя окремих людей і сприяти створенню більш стійкого та екологічно свідомого суспільства.

Слід зазначити, що виклики застосування досліджуваних методів включають забезпечення конфіденційності та безпеки даних, обробку великих обсягів інформації в реальному часі та інтеграцію різноманітних джерел даних. Втім, постійне вдосконалення інформаційних систем і технологій великих даних сприяє вирішенню цих викликів і відкриває нові можливості для розробки інноваційних рішень в області управління даними.

Вебзастосунок "Розумний холодильник" є частиною широкої категорії "розумних" пристроїв та застосунків, які інтегруються з Інтернетом речей та використовують великі дані для покращення повсякденного досвіду користувачів. Існують різні продукти та застосунки, які можна вважати подібними до "Розумного холодильника" за функціональністю або підходом. Оглянемо деякі з них.

"Розумні термостати", наприклад, Nest або Ecobee, використовують дані про звички користувачів, погоду та інші фактори для оптимізації опалення та охолодження домашнього простору.[8]

Освітлювальні Системи Philips Hue, дозволяють користувачам контролювати освітлення через мобільний застосунок, створювати налаштування освітлення відповідно до настрою або часу доби.

Продукти Arlo, пропонують розумні дверні дзвінки, камери безпеки та інші пристрої, які користувачі можуть контролювати та моніторити через мобільні застосунки.

Кухонні Пристрої, Instant Pot Smart WiFi або Anova Precision Cooker, дозволяють користувачам контролювати приготування їжі через мобільні застосунки, надаючи можливості для дистанційного управління та моніторингу.

Застосунки для управління домашнім бюджетом допомагають стежити за витратами на продукти та інші побутові потреби, аналізуючи витрати та рекомендуючи бюджетні рішення.

Ці пристрої та застосунки подібні до "Розумного холодильника" тим, що вони використовують сучасні технології, такі як IoT, машинне навчання та аналіз даних, для підвищення зручності, ефективності та особистої адаптації у повсякденному житті.

"Розумні" холодильники, в самому прямому значенні на сьогодні, – це пристрої, які інтегруються з інтернетом та використовують передові технології для підвищення зручності та ефективності. Вони часто оснащені функціями, такими як віддалене управління, моніторинг продуктів, інтеграція з іншими розумними пристроями в домі та навіть мають вбудовані екрани для перегляду рецептів або управління функціями холодильника.

Наприклад, Samsung Family Hub має великий сенсорний екран, який може використовуватися для перегляду рецептів, стрімінгу музики або відео, перегляду календаря сім'ї тощо. Він також дозволяє користувачам переглядати вміст холодильника за допомогою вбудованих камер.[9]

LG InstaView ThinQ включає функцію InstaView Door-in-Door, яка дозволяє користувачам подивитися всередину холодильника, подвійно постукавши по скляній панелі. Також він має функції розумного домашнього асистента та можливість управління голосом.

Холодильник GE Café Series від General Electric має вбудовану сенсорну панель та інтеграцію з Wi-Fi, що дозволяє користувачам керувати

температурою, отримувати сповіщення про відкриті двері та інші корисні функції.

Розумний холодильник Smart French Door Refrigerator від Whirlpool включає функції, такі як віддалене управління через мобільний застосунок, сповіщення про статус холодильника та заморожувача та спеціальні режими для зберігання певних типів продуктів.

Продукти Bosch Home Connect Series включають функції інтелектуального управління та моніторингу, дозволяючи користувачам контролювати та діагностувати свій прилад через мобільний застосунок.

Ці та інші моделі розумних холодильників демонструють, як технології Інтернету речей та інтелектуального дому можуть бути інтегровані в побутові прилади для підвищення їхньої функціональності та зручності використання.

Аналізуючи останні два переліка продуктів, можна виділити кілька ключових переваг, які можуть бути корисними для розробки вебзастосунку "Розумний холодильник". Також можна розглянути потенційні проблеми, які можуть бути вирішені за допомогою нових технологічних підходів.

Переваги для використання в "Розумному холодильнику":

1. Використання сенсорного екрану, подібного до Samsung Family Hub, дозволяє користувачам взаємодіяти з холодильником більш інтуїтивно, переглядаючи рецепти, управляючи запасами продуктів, стрімуючи музику чи відео.
2. Можливість віддаленого управління, як у LG InstaView ThinQ, дозволяє користувачам контролювати налаштування холодильника через мобільний застосунок, що є зручно для моніторингу та управління запасами продуктів.
3. Система сповіщень, як у GE Café Series, може надавати корисну інформацію про стан холодильника, наприклад, про необхідність заміни фільтра для води або про відкриті двері.

4. Інтеграція з іншими системами "Розумного дому" може забезпечити синхронізацію холодильника з іншими пристроями, створюючи єдину екосистему домашньої автоматизації.

Проблеми та їх рішення:

1. Розробка алгоритмів для ефективного управління енергоспоживанням може допомогти знизити електроенергію, яку споживає холодильник, тим самим зменшуючи витрати та вплив на довкілля. Різні продукти вимагають різних рівнів температури зберігання, що надає широкі можливості розташування продуктів.
2. Використання розширених технологій виявлення продуктів, таких як RFID або штучний інтелект для розпізнавання продуктів, може допомогти точніше відстежувати запаси та терміни придатності продуктів.
3. Збір та аналіз даних про звички та переваги користувачів може дозволити надавати персоналізовані рекомендації, наприклад, рецепти або поради щодо здорового харчування.
4. Розробка безпечних протоколів збору та зберігання даних є критично важливою для захисту інформації про споживачів та їхніх персональних даних.

Впровадження цих переваг та рішень у "Розумному холодильнику" може значно покращити його функціональність та користувацький досвід, роблячи його більш інтуїтивним, ефективним та корисним для повсякденного використання.

2 ПОСТАНОВКА ЗАДАЧІ НА ДОСЛІДЖЕННЯ

Об'єктом дослідження у цій роботі є удосконалення методу управління великими даними в реальному часі на прикладі вебзастосунку. Це включає розробку інноваційних підходів та алгоритмів для збору та підготовки до обробки і аналізу великих обсягів даних, що генеруються IoT, в режимі реального часу.

Дослідження охоплює вивчення та застосування технологій хмарного обчислення для забезпечення масштабованості, високої доступності та ефективності обробки великих даних в режимі реального часу. Основна увага приділяється вирішенню викликів, пов'язаних із швидкісною обробкою великих даних, їх зберіганням та аналітикою в умовах великого обсягу інформації, що постійно оновлюється. Це включає розробку ефективних механізмів для збору даних з ІС, аналізу споживацької поведінки, прогнозування користувацьких потреб, а також інтеграцію цих даних з іншими системами "розумного дому" для створення комплексного інтелектуального середовища.

Таким чином, центральним питанням дослідження є удосконалення методу, який не тільки спрощує управління великими даними, але й вносить інновації в способи їх застосування в рамках вебзастосунку, використовуючи переваги хмарних технологій.

Предметом дослідження в роботі виступає ІС, що представляє собою втілення концепції IoT в побутовій техніці. Головне завдання цієї системи полягає у виконанні комплексного управління запасами продуктів, моніторингу їх стану і взаємодії з користувачем для оптимізації процесу харчування та покупок.

Ця система включає в себе програмне забезпечення, яке здатне навчатися на засадах машинного навчання, удосконалюючи алгоритми прогнозування та персоналізації на основі зібраних даних і уподобань користувача. Також вебзастосунок має можливість синхронізації з мобільними пристроями та

іншими елементами розумного будинку, розширюючи свої функціональні можливості та стаючи невід'ємною частиною повсякденного життя користувача.

Завдяки такій багатофункціональності та інтегрованості, вебзастосунок відкриває перед дослідниками широке поле для вивчення методів управління великими даними в реальному часі та їх застосування в побуті, що сприятиме не тільки зручності та ефективності використання продуктів, а й екологічності споживання завдяки мінімізації відходів.

Загальна схема роботи вебзастосунку за допомогою вбудованих алгоритмів в розрізі управління великими даними у реальному часі полягає у зборі, обробці, збереженні, аналізі даних, створенні прогнозів, здійсненні синтезу і надані користувачу як результату - рекомендацій, сповіщень та пропозицій. Однією із максималістських задач подальшого розвитку застосунку є досягнення такого рівня прогнозування на основі візуалізації та аналізу даних, коли ІС має інформацію про час, місце та вид їжі, які обере споживач.

На перший погляд може здатись, що автоматизація, що ведеться ШІ, може призвести до значних змін на ринку праці, включаючи втрату робочих місць. Але повністю замінити обміркований вплив користувача на ІС на основі алгоритмів ШІ - це лише варіант подальшого розвитку ІС, який на сьогодні ще не має вирішення з огляду на існуючі занепокоєння. Наприклад, що ШІ може вийти з-під контролю або бути використаним у шкідливих цілях. Важливо розробляти ШІ таким чином, щоб він був сумісним з цінностями людства.

Для інформаційної системи вебзастосунку "Розумний холодильник", вхідні дані та їх якість мають важливу роль, як інформація, що необхідна для обробки та прийняття рішень. Основними типами вхідних даних для цієї системи є:

1. Інформація про продукти, що зберігаються у холодильнику, включаючи їх назви, кількість, терміни придатності, інформацію про вагу та інші специфічні характеристики.

2. Дані, введені користувачем або зібрані через інтерактивний інтерфейс, такі як персональні налаштування, історія покупок, переваги в продуктах та звички харчування.
3. Повідомлення, що надходять від інших пристроїв в рамках "розумного дому", які можуть включати дані про споживання енергії, вагу продуктів та інші сенсорні дані.
4. Дані про зовнішні умови, такі як погода, сезонність продуктів, інформація з мережі продуктових магазинів про акції, знижки або наявність конкретних продуктів.
5. Відомості про те, як користувачі взаємодіють з інтерфейсом застосунку, включаючи використання різних функцій, частоту входу в застосунок, відгуки та відзначення важливих функцій.

Ці вхідні дані становлять основу для всіх алгоритмів та функцій "Розумного холодильника", дозволяючи системі адекватно реагувати на потреби користувача, ефективно управляти запасами продуктів і надавати корисні рекомендації та інформацію.

Вихідні дані "Розумного холодильника" представляють собою комплексну інформацію, яка генерується на базі Big Data та їх аналітики з метою полегшити та оптимізувати процеси харчування та управління продуктовими запасами користувача. Ці дані включають в себе рекомендації, сповіщення, аналітичні звіти та інші корисні інструменти, що допомагають користувачу в повсякденному житті.

На основі аналізу зберігаємих продуктів, їхнього стану і термінів придатності, а також з урахуванням історичних даних про споживання, система формує персоналізовані рекомендації щодо замовлення нових продуктів. Це не просто списки потрібних для покупки товарів, а інтелектуальні пропозиції, які враховують переваги користувача, його дієтичні обмеження та навіть сезонність продуктів.

Сповіщення включають в себе інформування користувача про необхідність вжиття продуктів, які наближаються до кінця терміну їх

придатності, та потребу заміни тих, що вже не можуть бути вжиті. Це дозволяє зменшити кількість відходів та зекономити кошти.

Аналітичні звіти дозволяють користувачу оцінити свої звички у споживанні продуктів, виявляти тенденції та відповідно до них коригувати своє харчування або покупки. Звіти можуть включати інформацію про найбільш і найменш вживані продукти, частоту покупок, сезонні зміни у споживанні, а також порівняння цінових трендів.

Оптимізовані списки покупок генеруються з урахуванням не лише персональних переваг користувача, а й історії попередніх покупок, що робить процес вибору продуктів більш простим та швидким. Система може навіть враховувати актуальні пропозиції від магазинів або онлайн-сервісів доставки їжі, щоб користувач міг скористатися найвигіднішими знижками.

Таким чином, вихідні дані ІС — це динамічний набір інструментів, які допомагають користувачам вести більш здоровий, економний та екологічно свідомий спосіб життя.

Метою дослідження є розробка та дослідження ефективного методу управління великими даними в реальному часі на прикладі вебзастосунку "Розумний холодильник". Основна увага в дослідженні приділяється створенню інноваційного підходу до збору, обробки та аналізу даних, які генеруються інтелектуальними побутовими пристроями, з подальшою метою підвищення комфорту та ефективності їх використання користувачами.

Дослідження має описати процес розробки ІС, здатної адаптуватися до змінних умов та вимог користувачів, забезпечуючи при цьому швидкість та точність обробки даних. Використання методів машинного навчання та штучного інтелекту для прогнозування поведінки користувачів та оптимізації їхніх покупок та споживання продуктів є додатковим аспектом дослідження.

Цільова система повинна включати в себе здатність ідентифікувати та категоризувати продукти, відстежувати терміни їх придатності, аналізувати звички користувачів, і надавати персоналізовані рекомендації. Також важливою метою є забезпечення інтеграції "Розумного холодильника" з іншими

компонентами розумного будинку та зовнішніми сервісами для створення єдиної інформаційної екосистеми.

У процесі дослідження слід використати передові методики в області Big Data та IoT, зокрема, для оптимізації процесів збору даних, їх передачі, зберігання та захисту. Це дозволить виявити найбільш значущі параметри для підвищення ефективності ІС "Розумного холодильника" та створити модель, яка може бути масштабована та адаптована для використання в інших пристроях та середовищах.

Таким чином, метою дослідження є не лише розробка функціонального "Розумного холодильника", а й внесок в загальний розвиток галузі управління великими даними, що має велике значення для прогресу інформаційних систем та технологій у цілому.

Критерії оцінювання в дослідженні визначають ключові показники успіху розробленої системи управління великими даними в реальному часі для "Розумного холодильника". Основна увага при оцінюванні зосереджується на якості та ефективності алгоритмів обробки даних, швидкості реакції системи на зміни в даних, точності аналізу та рекомендацій, здатності до інтеграції з іншими системами та безпеки даних.

При оцінюванні алгоритмів обробки даних важливим є здатність коректно та своєчасно обробляти великі обсяги інформації, що отримується в режимі реального часу, та на адаптивності цих алгоритмів до змінних умов та нових вимог. Ефективність системи вимірюється через співвідношення корисної дії алгоритму до використаної ним обчислювальної потужності та часу.

Швидкість реакції системи на зміни в даних є важливим критерієм, оскільки система повинна оперативно реагувати на нові дані, що надходять, наприклад, інформацію про нові продукти чи зміну їхнього стану. Система також має демонструвати здатність до швидкого внесення змін у рекомендації та адаптації до змін у користувацьких уподобаннях.

Точність аналізу та рекомендацій оцінюється через відсоток правильних прогнозів, здійснених системою, та її здатності мінімізувати помилки. Точність

рекомендацій включає в себе якість підбору продуктів для списку покупок, коректність вказівок щодо термінів споживання продуктів, та здатність адаптуватися до персональних дієтичних потреб користувача.

Здатність до інтеграції з іншими системами, зокрема з іншими елементами "розумного будинку" та зовнішніми сервісами, також є важливим критерієм, який оцінює гнучкість та відкритість системи для майбутнього розширення та сумісності.

Безпека даних оцінюється через здатність системи забезпечувати захист інформації від несанкціонованого доступу, втрати даних, їх пошкодження чи зловмисного використання. Це включає в себе ефективність шифрування, аутентифікації, аудиту доступу та інших механізмів захисту інформації.

При використанні вебзастосунку "Розумний холодильник" можуть виникнути декілька проблем, пов'язаних із захистом інформації, які необхідно врахувати, дотримуючись локального законодавства. Основні аспекти на прикладі законодавства України [10]:

- застосунок може збирати персональні дані користувачів, такі як імена, адреси, інформацію про споживані продукти - згідно з законодавством України, потрібно отримати згоду користувачів на збір та обробку їхніх персональних даних;
- необхідно забезпечити належне зберігання та захист персональних даних, вживати заходів проти їх несанкціонованого доступу, зміни, розголошення або знищення;
- інформація про харчові звички та вибір продуктів може вважатися конфіденційною.

Важливо забезпечити, щоб ця інформація не була розкрита без згоди користувача.

Якщо застосунок інтегрований з онлайн-магазинами для замовлення продуктів, необхідно враховувати законодавчі вимоги щодо електронної комерції, зокрема умови обробки платіжних даних та забезпечення безпеки транзакцій.

Оскільки "Розумний холодильник" є частиною екосистеми IoT, важливо враховувати законодавчі вимоги до безпеки та приватності в цій сфері.

При розробці та впровадженні "Розумного холодильника" важливо консультуватися з юристами, які спеціалізуються на захисті даних та конфіденційності, щоб забезпечити повне дотримання відповідного місцевого законодавства.

У сукупності, ці критерії дозволяють оцінити роботу "Розумного холодильника" як комплексну систему, що включає в себе елементи швидкодії, точності, адаптивності, інтеграції та безпеки, забезпечуючи при цьому високий рівень задоволеності користувачів та ефективності використання ресурсів.

3 УДОСКОНАЛЕННЯ МЕТОДУ УПРАВЛІННЯ ВЕЛИКИМИ ДАНИМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ НА ПРИКЛАДІ ВЕБЗАСТОСУНКУ

3.1 Опис методів рішення задачі

Вебзастосунок взагалі складається з кількох ключових компонентів, які спільно працюють для забезпечення його функціональності. Ці компоненти охоплюють як фронтенд (клієнтська частина), так і бекенд (серверна частина) застосунку, а також базу даних та інфраструктуру для розгортання та обслуговування застосунку.

Для повноцінної реалізації ІС "Розумний холодильник" необхідно використовувати різноманітні методи та технології. Вибір конкретних продуктів та мов програмування залежатиме від конкретних вимог до функціональності та архітектури системи. Кілька ключових технологій та продуктів, які можуть бути використані можна визначити як загальні.

В процесі розробки вебзастосунку "Розумний холодильник", використання професійних інструментів моделювання, таких як BPWIN та ERWIN, відіграє ключову роль у створенні ефективної та оптимізованої системи. Ці інструменти допомагають аналізувати, проектувати та оптимізувати бізнес-процеси та структуру бази даних, що є фундаментальними аспектами будь-якого сучасного вебзастосунку.[11]

BPWIN (Business Process Workbench) є могутнім інструментом для візуалізації та аналізу бізнес-процесів.

В контексті "Розумного холодильника", BPWIN може бути використаний для:

- моделювання робочих процесів - визначення та візуалізації ключових процесів, таких як управління запасами, обробки даних з сенсорів холодильника, взаємодії з користувачем та інші.
- оптимізації процесів - аналізу існуючих процесів на предмет ефективності та виявлення можливостей для поліпшення,

наприклад, скорочення часу відгуку системи або зменшення кількості кроків у процесі взаємодії.

ERWIN (Entity-Relationship WINdow) є інструментом для моделювання баз даних, що дозволяє ефективно спроектувати структуру бази даних та управляти даними.

У випадку з "Розумним холодильником", ERWIN використовується для:

- створення схеми БД - моделювання таблиць, зв'язків, ключів та індексів, що забезпечують зберігання та ефективний доступ до даних про продукти, користувацькі вподобання та історію взаємодій;
- аналізу та нормалізації даних - перевірки, що структура бази даних є оптимальною, відсутній зайвий дублювання даних, та забезпечена їх цілісність.

Використання BPWIN та ERWIN у проекті "Розумний холодильник" дозволяє не тільки оптимізувати окремі аспекти системи, але й гарантує, що весь проект розроблено з урахуванням кращих практик та стандартів. Це забезпечує високий рівень якості, ефективність в роботі та гнучкість у масштабуванні та подальшому розвитку проекту.

Мова програмування Python [12] є відмінним вибором для розробки ІС завдяки своїй універсальності та багатій екосистемі бібліотек. Він може бути використаний для розробки бекенду, обробки даних та машинного навчання.

Бекенд для вебзастосунку, реалізований на мові програмування Python, може включати наступні ключові складові та аспекти.

Перш за все, Python пропонує декілька потужних вебфреймворків для розробки бекенда. Найпопулярнішими є Django та Flask, які мають деякі відмінності, в залежності від вирішуваних задач та наявних апаратних і програмних ресурсів. Django є високорівневим фреймворком, який слідує принципу "батарейки включені", пропонуючи вбудовані рішення для багатьох задач веброзробки. А ось Flask вже є більш легковісним та гнучким

фреймворком, що дозволяє розробникам мати більший контроль над компонентами та архітектурою застосунку.

Інтерфейс користувача з базою даних (БД) на основі інтеграції з СУБД, такими як PostgreSQL, MySQL, SQLite або MongoDB (для NoSQL), здійснюється через ORM (Object-Relational Mapping) або прямі запити до бази даних. Django надає свій власний ORM, що дозволяє легко взаємодіяти з базою даних. Flask часто використовує SQLAlchemy як ORM, але також дозволяє використовувати прямі SQL-запити.

Для реалізація систем аутентифікації (перевірка ідентичності користувача) та авторизації (визначення прав доступу користувача) Django пропонує вбудовану систему аутентифікації, а Flask може використовувати розширення, такі як Flask-Login та Flask-Principal.

Розробка API для взаємодії з фронтендом, мобільними застосунками або іншими сервісами базується на використанні бібліотек, таких як Django REST framework для Django або Flask-RESTful для Flask, для створення RESTful API.

Python надає також програмні можливості для створення інструментів обробки HTTP-запитів та формування відповідей у вигляді HTML, JSON, XML тощо, застосування заходів безпеки, таких як валідація та санітізація вхідних даних, захист від CSRF-атак, використання HTTPS, налаштування логування для відстеження помилок та надзвичайних ситуацій у застосунку.

Серверне програмне забезпечення є не менш важливою складовою будь-якого вебзастосунку чи інформаційної системи. Цей тип програмного забезпечення відповідає за прийняття запитів від клієнтських пристроїв (наприклад, веббраузерів) та надання відповідних відповідей, які можуть включати вебсторінки, зображення, дані та інше. Серверне програмне забезпечення також управляє вебсерверами та іншою інфраструктурою, необхідною для функціонування вебзастосунків. Apache HTTP Server є одним із найпоширеніших вебсерверів, який відомий своєю надійністю, гнучкістю та широким спектром можливостей, які можна налаштовувати за допомогою численних модулів.

Фронтенд (клієнтська частина) може бути створена у поєднанні можливостей HTML, CSS та JavaScript.

HTML (HyperText Markup Language) використовується для структурування вмісту на вебсторінках, є основою для розробки фронтенду будь-якого вебзастосунку. Він відповідає за структурування та організацію вмісту на вебсторінках, що включає текст, зображення, посилання, форми та інші елементи.

За допомогою HTML можна створювати інтерактивні форми, які збирають дані від користувачів. Використовуються елементи форми, такі як input, textarea та button.

CSS (Cascading Style Sheets) використовується для стилізації вебсторінки, а JavaScript - для додавання інтерактивності. Вони інтегруються з HTML через відповідні теги (наприклад, link для CSS та script для JavaScript).

Семантичний HTML використовується для покращення доступності та SEO (пошукової оптимізації) вебсторінки. Це включає використання тегів, які чітко описують їх зміст та призначення (наприклад, article, section, footer, header).

Сучасний вебдизайн вимагає, щоб сторінки були адаптивними, тобто коректно відображалися на різних пристроях. Для цього в HTML використовуються метатеги для визначення в'юпорту та медіа-запити в CSS.

HTML дозволяє інтегрувати мультимедійний контент, такий як відео, аудіо та зображення, за допомогою відповідних тегів (img, video, audio).

Важливо розробляти HTML-код з урахуванням стандартів доступності, щоб забезпечити однакову функціональність вебсторінки для всіх користувачів, включаючи людей з обмеженими можливостями.

Використання HTML, як основи для фронтенду, забезпечує стабільний та універсальний підхід до створення вебсторінок, які є основою користувацького інтерфейсу будь-якого вебзастосунку.[13]

Ефективним доповненням CSS та HTML є мова програмування JavaScript, що дозволяє робити вебсторінки інтерактивними, наприклад, реагуючи на дії користувачів.

JavaScript є надзвичайно вдалим доповненням до HTML та CSS у процесі розробки вебсторінок, особливо коли мова йде про управління великими даними в режимі реального часу. Якщо HTML відповідає за структурування вмісту вебсторінки, а CSS за її візуальний стиль, то JavaScript додає необхідну інтерактивність, роблячи вебсторінки не просто статичними документами, а динамічними, багатофункціональними застосунками. Це особливо корисно не тільки з погляду візуальної привабливості застосунку, так і з точки зору можливостей застосування широкого набору функцій візуалізації та аналізу інформації, отриманої від користувача в ході використання застосунку .

JavaScript може реагувати на дії користувача, такі як кліки мишкою, введення з клавіатури або жести на сенсорних екранах. Це дозволяє створювати інтерактивний користувацький досвід, наприклад, реагування на вибір користувача, анімацію елементів інтерфейсу та динамічну зміну вмісту без необхідності перезавантаження сторінки.

Для управління великими даними, особливо в реальному часі, JavaScript є незамінним. Він дозволяє вебзастосункам отримувати, обробляти та відображати дані майже миттєво, наприклад, для відображення оновлень даних на дашбордах або у вигляді інтерактивних графіків та діаграм.

Технології, як AJAX (Asynchronous JavaScript and XML) та WebSockets, дозволяють JavaScript взаємодіяти з сервером у фоновому режимі, отримуючи нові дані або відправляючи запити без перезавантаження сторінки. Це особливо важливо для застосунків, які працюють з великими даними в реальному часі, де швидкість відгуку та актуальність інформації мають критичне значення.

Існує багато JavaScript-фреймворків та бібліотек, таких як React, Angular, Vue.js, які полегшують розробку складних інтерактивних вебзастосунків. Вони пропонують розширені можливості для управління інтерфейсом, станом застосунку та взаємодією з сервером.

Враховуючи ці аспекти, JavaScript є невід'ємною частиною сучасної веброзробки, особливо у контексті розробки динамічних, багатофункціональних вебзастосунків, які вимагають швидкої та ефективної обробки великих обсягів даних.[14]

Системи Управління Базами Даних (СУБД), наприклад, MySQL, PostgreSQL, MongoDB (для NoSQL), SQLite та інші, використовуються для зберігання та керування даними.

Різні типи СУБД використовуються в залежності від потреб проекту та типу даних.

MySQL – це відкрита СУБД, яка використовує SQL (Structured Query Language) для управління даними. Вона є однією з найпопулярніших систем для вебзастосунків завдяки своїй надійності, швидкості та простоті використання. MySQL підходить для широкого спектру застосунків, від малих до великих корпоративних систем.

PostgreSQL є потужною, відкритою об'єктно-реляційною СУБД. Вона підтримує розширені функції SQL і відома своєю стабільністю, масштабованістю та підтримкою складних запитів, що робить її вибором для складних і великих систем.

MongoDB – це провідна NoSQL база даних, яка використовується для зберігання даних у вигляді документів. Вона пропонує високу гнучкість і є відмінним вибором для застосунків, що потребують швидкого розвитку, а також для роботи з великими обсягами неструктурованих або напівструктурованих даних.

SQLite – це легковісна вбудована СУБД, яка не вимагає окремого сервера. Вона чудово підходить для мобільних застосунків, невеликих застосунків або як компонент більших застосунків, де потрібна проста і надійна система зберігання даних.

Кожна з цих СУБД має свої особливості та оптимальні сценарії використання. Вибір конкретної СУБД для проекту залежить від різних факторів, включаючи тип даних, які потрібно обробляти, вимоги до

масштабованості та доступності, а також індивідуальні переваги розробника або команди.

Сервіси хмарного хостингу, наприклад, Amazon Web Services [15], Google Cloud Platform [16], які можуть використовуватися для розгортання та масштабування застосунку. Ці платформи пропонують широкий спектр послуг та інструментів, які допомагають розробникам швидко розгортати, масштабувати та управляти своїми застосунками. Вони дозволяють легко масштабувати застосунки, збільшуючи або зменшуючи обчислювальні ресурси відповідно до потреб. Це особливо корисно для впорання зі змінним навантаженням, наприклад, піковими навантаженнями під час рекламних кампаній або особливих подій. Приваблює різноманітність сервісів таких платформ, включаючи обчислювальні потужності (наприклад, EC2 в AWS), бази даних (наприклад, Amazon RDS, Google Cloud SQL), засоби для роботи з великими даними, інструменти для машинного навчання, функції безпеки та багато іншого.

Хмарні провайдери пропонують високий рівень доступності та надійності, розподіляючи дані та ресурси через численні географічно розподілені центри обробки даних. З іншого боку, за їх допомогою можна отримати розширені засоби управління та моніторингу, які дозволяють відстежувати стан застосунку, аналізувати продуктивність та виявляти можливі проблеми.

Однією з ключових переваг хмарного хостингу є гнучка модель ціноутворення, де платиш лише за ті ресурси, які фактично використовуєш. Це дозволяє оптимізувати витрати, особливо для стартапів та малих проектів.

Загалом, хмарні сервіси сприяють спрощенню процесів розгортання та неперервної інтеграції (CI/CD), надаючи інструменти та сервіси для автоматизації цих процесів, підтримують широкий спектр технологій та мов програмування, що робить їх ідеальними для різноманітних вебзастосунків і мікросервісної архітектур, особливо з огляду на необхідність роботи з великими даними в режимі реального часу.

Використання хмарних сервісів для розгортання та масштабування вебзастосунків дозволяє розробникам зосередитися на розробці функціональності, не турбуючись про управління фізичною інфраструктурою серверів.

В результаті оцінки описаних методів, можна визначити той набір методів, який дозволить створити вебзастосунок "Розумний холодильник" для реалізації поставлених перед ним завдань.

Для розробки вебзастосунку "Розумний холодильник", важливо вибрати програмні засоби, які найкраще відповідають потребам проекту, забезпечуючи ефективність, гнучкість і масштабованість.

Перелік та обґрунтування вибору ключових технологій і бібліотек:

Python є відмінним вибором для бекенду завдяки своїй читабельності, ефективності та великому співтовариству. Він підтримує численні вебфреймворки та бібліотеки, що полегшують розробку.

Бібліотеки Django або Flask ,як основні вебфреймворки, SQLAlchemy (для Flask) або Django ORM для взаємодії з базою даних.

HTML/CSS/JavaScript для фронтенду. Стандартний набір технологій для створення користувацьких інтерфейсів. HTML відповідає за структуру сторінок, CSS - за стиль, а JavaScript - за інтерактивність.

Бібліотеки/Фреймворки React або Vue.js для більш інтерактивного та динамічного UI, Bootstrap або Tailwind CSS для стилізації.

NoSQL для бази даних, такі як MongoDB, є ідеальними для обробки великих обсягів неструктурованих даних, які можуть бути згенеровані "Розумним холодильником". Серед переваг - гнучкість схеми, масштабованість та ефективна робота з JSON-подібними документами.

RESTful API для зв'язку між фронтендом і бекендом дозволить організувати ефективний зв'язок між клієнтом і сервером, що є ключовим для вебзастосунків, що працюють з великими даними.

Інструменти Swagger або Postman для тестування та документації API.

Додаткові інструменти та бібліотеки:

D3.js або Chart.js для візуалізації даних, особливо якщо потрібно відображати складні графіки або діаграми з даними холодильника.

WebSocket для реалізації взаємодії в реальному часі, наприклад, для отримання миттєвих оновлень стану холодильника.

OAuth або JWT (JSON Web Tokens) для аутентифікації та забезпечення безпеки.

Вибір цих технологій забезпечує баланс між гнучкістю, продуктивністю та легкістю розробки, що є ключовими для створення сучасного, надійного та ефективного вебзастосунку "Розумний холодильник".

Для ефективної розробки вебзастосунку "Розумний холодильник" важливо вибрати відповідне середовище розробки та систему контролю версій. Ці інструменти забезпечують продуктивність розробників, організацію коду та спрощення співпраці в команді.

Visual Studio Code - легкий, але потужний редактор від Microsoft, популярний серед розробників JavaScript/Node.js та Python. Він підтримує безліч розширень, включаючи інтеграцію з Git, підсвічування синтаксису, інтелектуальне автозавершення коду та відладку.[17]

Git - найпопулярніша система контролю версій на сьогоднішній день. Git дозволяє розробникам вести версіонування коду, легко відстежувати зміни, співпрацювати з іншими розробниками і розгалужувати проекти. GitHub [18] або GitLab - веббазовані сервіси, які доповнюють Git, надаючи графічний інтерфейс для управління репозиторіями, а також інструменти для спільної роботи, включаючи відстеження проблем, код-рев'ю та CI/CD (Continuous Integration/Continuous Deployment).

Для забезпечення високої якості та надійності вебзастосунку "Розумний холодильник", важливо інтегрувати в процес розробки ефективні засоби тестування. Тестування допомагає виявити та виправити помилки, перевірити відповідність функціональності вимогам та забезпечити стабільну роботу застосунку.

Для тестування фронтенду підійде Jest, популярний тестовий фреймворк для JavaScript, який часто використовується разом з бібліотекою React.

Mocha/Chai є гнучким тестовим фреймворком для JavaScript, що дозволяє легко писати та виконувати тести. Cypress або Selenium - інструменти для автоматизації браузерного тестування, які дозволяють імітувати взаємодії користувачів з вебзастосунком.

За допомогою PyTest можна реалізувати тестування бекенду. Це міцний та гнучкий фреймворк для написання та виконання тестів в Python. Стандартний фреймворк JUnit для юніт-тестування в Java-орієнтованих проектах. Postman дозволяє тестувати API, відправляючи запити до сервера та перевіряючи відповіді.

Інтеграційні тести перевіряють, як різні частини застосунку працюють разом. Це може включати тестування взаємодії між фронтендом та бекендом, а також між бекендом та базою даних.

Jenkins, Travis CI, GitHub Actions автоматизують процес тестування та розгортання, виконуючи тести на кожен коміт у репозиторій.

New Relic або Datadog, можуть використовуватися для моніторингу продуктивності застосунку та виявлення "вузьких місць".

3.2 Визначення основних функцій та вимог вебзастосунку

Основні вимоги щодо функціональності включають:

- управління запасами продуктів,
- моніторинг термінів придатності,
- автоматичне планування покупок,
- інтеграція з онлайн-магазинами для замовлення продуктів,
- відстежування харчових звичок користувача,
- надання рекомендації щодо рецептів на основі наявних продуктів та статистики використання продуктів.

Вимоги до користувацького інтерфейсу:

- простота та інтуїтивність навігації,
- сучасний дизайн.

Для отримання даних для аналізу потреб користувачів за допомогою опитування у вебзастосунку проводиться збір даних про їхні харчові звички, частоту покупок та типові проблеми, з якими вони стикаються під час управління домашніми запасами продуктів.

Список необхідних функцій:

- управління запасами продуктів: можливість вводити інформацію про куплені продукти, їх кількість, термін придатності.
- моніторинг термінів придатності: автоматичне сповіщення користувачів про продукти, що незабаром можуть зіпсуватися.
- автоматичне планування покупок: система може пропонувати список покупок на основі історії споживання та поточного запасу продуктів.
- інтеграція з онлайн-магазинами: можливість замовлення продуктів безпосередньо через вебзастосунок.

Проектування користувацького інтерфейсу займає важливе місце в контексті візуальної привабливості та зручності використання будь-якого вебзастосунку. При створенні прототипів інтерфейсів для кожної функції вебзастосунку увага повинна бути сконцентрована на простоті, інтуїтивності використання та естетиці дизайну.

Оцінка ефективності може бути проведена після тестування прототипів з використанням методів UX/UI, таких як user testing, для забезпечення високої якості користувацького досвіду.

Розробка технічної специфікації, яка включає детальний опис технологій, інструментів та платформ, що будуть використовуватися для реалізації вебзастосунку.

Ключовим аспектом є розуміння того, що вебзастосунок "Розумний холодильник" повинен не тільки вирішувати конкретні задачі користувачів, але й пропонувати зручний та приємний досвід використання.

При розробці концепції інтерфейсу користувача можна визначити, що головні екрани вебзастосунку, включають:

- головну панель управління,
- сторінки запасів продуктів,
- історії покупок,
- рекомендацій щодо рецептів,
- кабінет користувача.

Розробка верхньорівневих макетів полягає у створенні верхньорівневих макетів (wireframes) для кожного з ключових екранів, використовуючи інструмент вебдизайну Figma.

Головна панель управління містить інформативний огляд стану запасів продуктів, швидкий доступ до функцій додавання нових продуктів та огляду активних нагадувань про терміни придатності.

Інтеграція інтерактивних елементів, таких як сповіщення та рекомендації, для забезпечення ефективної взаємодії користувача з вебзастосунком.

На сторінці управління запасами передбачається створення візуалізації детального списку усіх наявних продуктів, з можливістю сортування за різними критеріями: за терміном придатності або категорією.

Сторінка історії покупок дозволяє користувачам переглядати свою історію покупок, аналізувати витрати та виявляти тенденції у своїх харчових звичках.

Сторінка рекомендацій рецептів пропонує рецепти на основі наявних продуктів, з наданням можливості користувачу змінювати параметри пошуку.

Сторінка "кабінет користувача" надає можливість для персоналізації вебзастосунку, включаючи налаштування сповіщень, керування обліковим записом та інші параметри користувацького досвіду.

Створення стартової сторінки вебзастосунку "Розумний холодильник" вимагає використання HTML для структурування контенту, CSS для стилізації та JavaScript для додавання інтерактивності:

```
<!DOCTYPE html>
```

```
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Розумний Холодильник</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f0f0f0;
        }
        .header {
            background-color: #4CAF50;
            color: white;
            padding: 10px 0;
        }
        .content {
            margin: 20px;
        }
        .footer {
            background-color: #333;
            color: white;
            position: fixed;
            bottom: 0;
            width: 100%;
            padding: 10px 0;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Ласкаво просимо до Розумного Холодильника</h1>
    </div>

    <div class="content">
```

```
        <p>Оптимізуйте ваше харчування та управління продуктами з
Розумним Холодильником.</p>
        <button onclick="startApp()">Запустити
вебзастосунок</button>
    </div>

    <div class="footer">
        <p>Розумний Холодильник &copy; 2023</p>
    </div>

    <script>
        function startApp() {
            // Реалізація коду для запуску вебзастосунку або
перенаправлення на іншу сторінку
            alert("Запуск вебзастосунку Розумний Холодильник!");
        }
    </script>
</body>
</html>
```

Цей HTML-код створює просту структуру сторінки з заголовком, основним контентом та футером. CSS використовується для базової стилізації сторінки, а JavaScript додає базову інтерактивність - у цьому випадку, просте сповіщення, яке з'являється при натисканні на кнопку "Запустити вебзастосунок".

Для додавання сторінки реєстрації та авторизації до вебзастосунку "Розумний холодильник", розширемо базовий HTML-код, з використанням HTML та невеликої кількості JavaScript для обробки подій форми (без реальної серверної логіки аутентифікації):

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Розумний Холодильник - Вхід/Реєстрація</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f0f0f0;
        }
        .header {
            background-color: #4CAF50;
            color: white;
            padding: 10px 0;
        }
        .content {
            margin: 20px;
        }
        .form-container {
            background-color: white;
            margin: 20px auto;
            padding: 20px;
            border-radius: 8px;
            width: 300px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        input[type=text], input[type=password] {
            width: 90%;
            padding: 12px 20px;
            margin: 8px 0;
            display: inline-block;
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
        }
        button {
            background-color: #4CAF50;
```

```
            color: white;
            padding: 14px 20px;
            margin: 8px 0;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            width: 95%;
        }
        button:hover {
            background-color: #45a049;
        }
        .footer {
            background-color: #333;
            color: white;
            position: fixed;
            bottom: 0;
            width: 100%;
            padding: 10px 0;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Розумний Холодильник</h1>
    </div>

    <div class="content">
        <div class="form-container">
            <form id="loginForm">
                <h2>Вхід</h2>
                <input type="text" placeholder="Ім'я користувача"
name="username" required>
                <input type="password" placeholder="Пароль"
name="password" required>
                <button type="submit">Увійти</button>
            </form>
```

```
        </div>
        <div class="form-container">
            <form id="registerForm">
                <h2>Реєстрація</h2>
                <input type="text" placeholder="Ім'я користувача"
name="newUsername" required>
                <input type="password" placeholder="Пароль"
name="newPassword" required>
                <button type="submit">Зареєструватися</button>
            </form>
        </div>
    </div>

    <div class="footer">
        <p>Розумний Холодильник &copy; 2023</p>
    </div>

    <script>

document.getElementById('loginForm').addEventListener('submit',
function(event) {
            event.preventDefault();
            // Код для обробки входу
            alert('Форма входу оброблена!');
        });


document.getElementById('registerForm').addEventListener('submit',
function(event) {
            event.preventDefault();
            // Код для обробки реєстрації
            alert('Форма реєстрації оброблена!');
        });
    </script>
</body>
</html>
```

Цей код створює дві окремі форми для входу та реєстрації користувачів. Кожна форма має поля для введення імені користувача та пароля та кнопку для відправки форми. JavaScript використовується для відміни стандартної поведінки відправки форми та показу сповіщень, що імітують обробку даних.

Додаємо сторінку відображення списку продуктів із зазначенням у кожного з них:

- кількості,
- термін придатності,
- короткий опис,
- білки, жири, вуглеці, клітковина, сіль, цукор, інші параметри,
- прогноз витрачання

Для створення сторінки відображення списку продуктів із детальною інформацією використаємо HTML для структури сторінки та невеликий блок CSS для базової стилізації:

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Список продуктів - Розумний Холодильник</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }
        .header {
```

```
            background-color: #4CAF50;
            color: white;
            text-align: center;
            padding: 10px 0;
        }
        .product-list {
            width: 80%;
            margin: 20px auto;
        }
        .product {
            background-color: white;
            border: 1px solid #ddd;
            padding: 10px;
            margin-bottom: 10px;
        }
        .product h3 {
            margin-top: 0;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Список Продуктів</h1>
    </div>

    <div class="product-list">
        <!-- Продукт -->
        <div class="product">
            <h3>Назва продукту</h3>
            <p>Кількість: 1 кг</p>
            <p>Термін придатності: 01.01.2024</p>
            <p>Опис: Яблука</p>
            <p>Живильна цінність (на 100г): Білки: 0.3г, Жири:
0.2г, Вуглеводи: 14г, Клітковина: 2.4г, Сіль: 0г, Цукор: 10г</p>
            <p>Прогноз витрачання: 200г на день</p>
        </div>
```

```
        <!-- Інші продукти -->

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

Цей HTML-код створює сторінку зі списком продуктів, де кожен продукт представлений у вигляді окремого блоку з інформацією про кількість, термін придатності, описом, живильною цінністю та прогнозом витрачання. CSS використовується для базової стилізації сторінки, роблячи її читабельною та візуально привабливою.

Додаємо сторінку для формування корзини замовлення продуктів із використанням парсингу даних з вебсторінок з використанням логіки, яка включає бекенд-процеси для парсингу та збору даних:

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Формування корзини замовлення - Розумний
Холодильник</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }
        .header {
            background-color: #4CAF50;
            color: white;
```

```
            text-align: center;
            padding: 10px 0;
        }
        .shopping-cart {
            width: 80%;
            margin: 20px auto;
        }
        .product {
            background-color: white;
            border: 1px solid #ddd;
            padding: 10px;
            margin-bottom: 10px;
        }
        .product input[type="checkbox"] {
            margin-right: 10px;
        }
        button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 15px;
            margin: 20px 0;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Формування корзини замовлення</h1>
    </div>

    <div class="shopping-cart">
```

```
        <div class="product">
            <input type="checkbox" id="product1" name="product1">
            <label for="product1">Продукт 1</label>
        </div>

        <div class="product">
            <input type="checkbox" id="product2" name="product2">
            <label for="product2">Продукт 2</label>
        </div>

        <!-- Додаткові продукти... -->

        <button onclick="submitOrder()">Замовити вибрані
продукти</button>
    </div>

    <script>
        function submitOrder() {
            // Код для обробки замовлення
            alert('Замовлення відправлене!');
        }
    </script>
</body>
</html>
```

Ця сторінка включає список продуктів із чекбоксами для кожного продукту, що дозволяє користувачам обрати, які саме продукти вони хочуть замовити. Натискання на кнопку "Замовити Вибрані Продукти" ініціює функцію submitOrder(), яка може бути реалізована для обробки вибраних продуктів та відправлення замовлення.

3.3. Збір та підготовка даних для методу

Збір та підготовка даних є критично важливими етапами в процесі розробки алгоритмів обробки, особливо у сферах, де використовуються методи машинного навчання та аналізу даних. Якісні, добре структуровані та чисті дані значно підвищують ефективність та точність моделей та алгоритмів. Спершу слід визначити джерела даних. Це можуть бути бази даних, веб-сайти, API, сенсори, опитування, журнали транзакцій тощо. Наступним кроком є збір різних типів даних, які можуть бути корисними для реалізації. Це можуть бути текст, зображення, відео, звукові дані, структуровані записи тощо. Необхідно переконатися, що збір даних відповідає всім юридичним та етичним нормам, включаючи згоду на обробку особистих даних.

Підготовка даних та очищення даних полягає у видаленні або коригуванні неправильних, відсутніх або аномальних значень. Це може включати виправлення помилок, заповнення пропущених значень та видалення дублікатів. Після цього можна привести дані до консистентного формату - перетворення текстових файлів у табличні дані, перекодування категорійних даних, нормалізацію числових значень тощо. Для оптимізації використовуються методи зменшення розміру, такі як вибір особливостей та розмірностей, щоб скоротити обсяг даних, зберігаючи при цьому їх корисність.

Для зручності використання - дані поділяються на набори для навчання, перевірки та тестування. Це допоможе в оцінці та валідації ефективності алгоритму.

На основі головних вхідних даних - уподобань користувача та історії використання ним продуктів, у тому числі параметрів: білки, вуглеці, жири, клітковина, цукор, сіль та інших, за допомогою функціонала вебзастосунку "Розумний холодильник" здійснюється візуалізація та аналіз отриманих даних для створення рекомендацій щодо придбання користувачем продуктів по закладеним критеріям, основними серед яких є вимога отримання користувачем

збалансованого харчування з оптимальним використанням ресурсів. Оскільки процес споживання відбувається безперервно, а саме змінюються:

- уподобання користувача,
- набір доступних продуктів, строки їх доставки, термін придатності,
- все це вимагає коригування в режимі реального часу.

Складовими алгоритма є:

- функціонал для аналізу термінів придатності продуктів, який буде автоматично нагадувати користувачам про продукти, що незабаром можуть зіпсуватися.
- функціонал системи рекомендацій рецептів, яка аналізує наявні запаси продуктів та користувацькі переваги для надання персоналізованих рекомендацій.
- реалізація функціональності планування покупок, що враховує історію споживання продуктів, їх наявність та планові потреби користувача.

Для збирання та підготовки даних для алгоритму обробки і надання рекомендацій у вебзастосунку "Розумний холодильник", використовуючи можливості Google Cloud, реалізовано наступні кроки:

1) Створення проекту в Google Cloud Platform.
2) Збирання та зберігання даних.
3) Розробка функціональності для збору даних від користувачів через веб-інтерфейс та інтеграція збору даних із бекендом вебзастосунку.
4) Інтеграція з Google BigQuery з можливістю виконання запитів SQL над зібраними даними, аналізу харчових звичок користувачів, історії покупок та уподобань.
5) Візуалізація даних.

Використання Google Cloud Storage для збору та зберігання великих обсягів даних, включаючи історію покупок користувачів, детальну інформацію про продукти та харчові звички забезпечує масштабованість та надійність

зберігання даних з використанням хмарних технологій, не задіюючи фізичні носії інформації користувача.[19]

Здійснимо налаштування Google Cloud Environment. Для створення проекту в Google Cloud Platform (GCP) проведемо реєстрацію аккаунту та створимо новий проект у GCP для вебзастосунок.

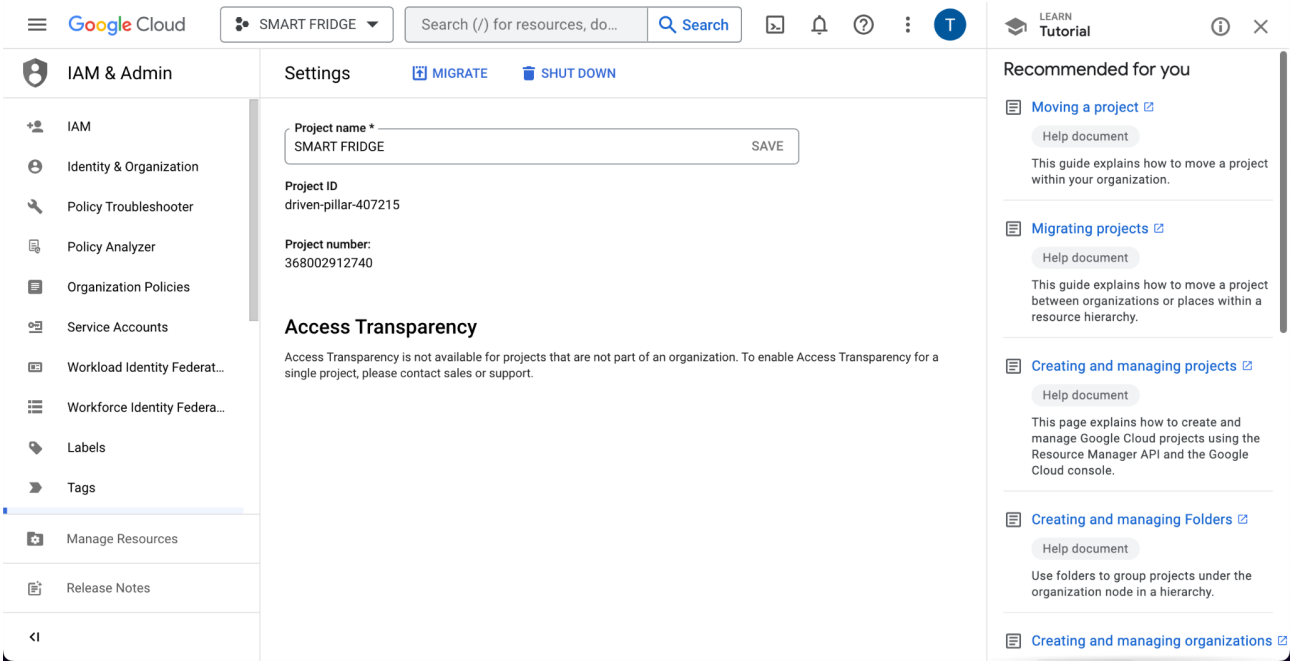

Рисунок 3.3.1 - Скріншот реєстрації аккаунту на Google Cloud Platform

Створимо кошик у Google Cloud Storage для зберігання даних.

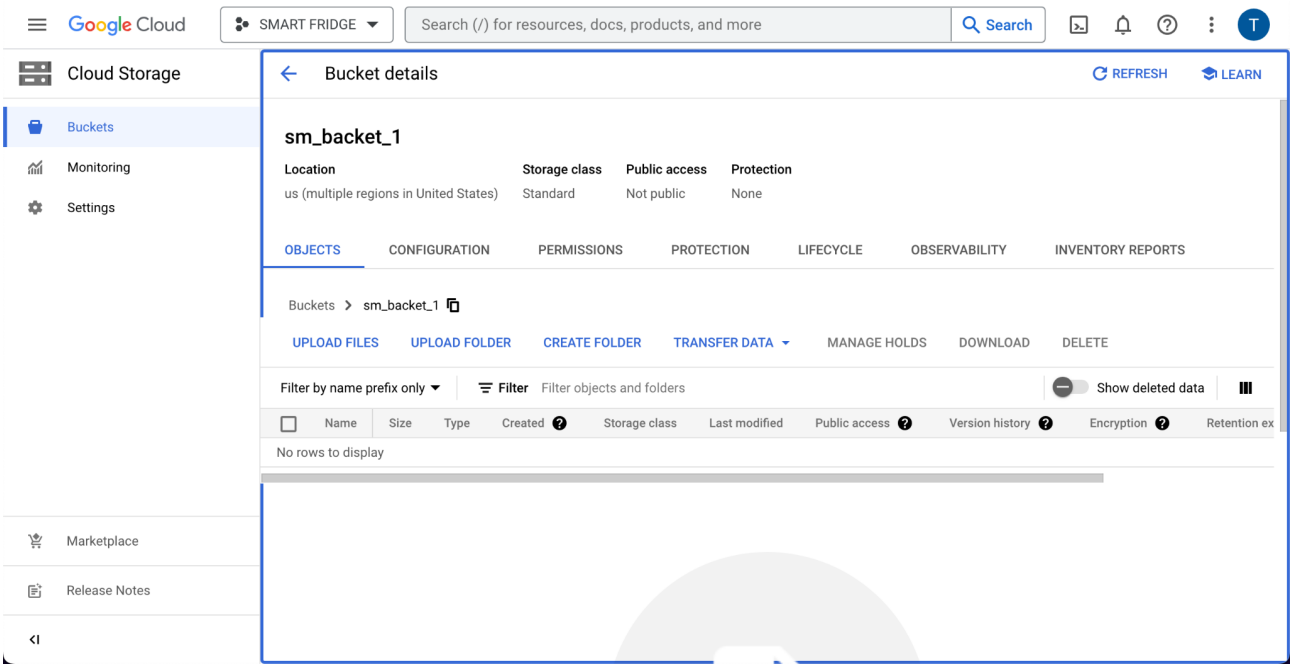

Рисунок 3.3.2 - Скріншот створення кошику

Налаштуємо дозволи та доступ до кошику.

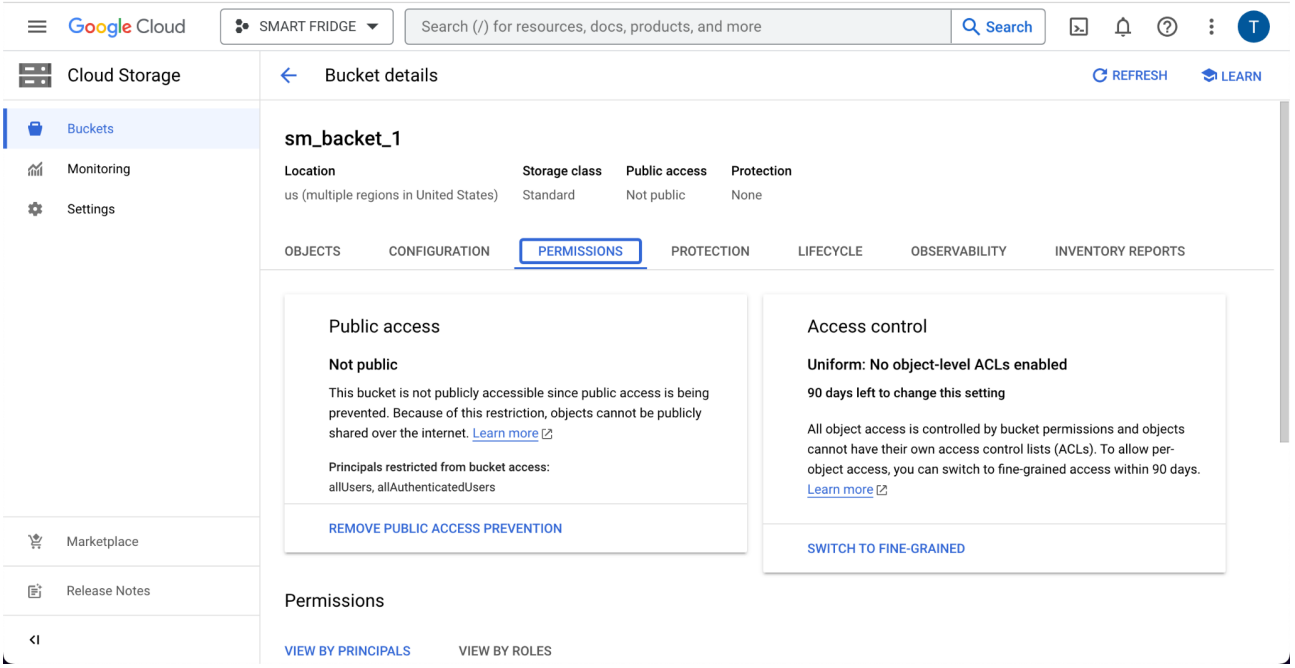

Рисунок 3.3.3 - Скріншот налаштування дозволів та доступу до кошика

Для інтеграції API Google Cloud з Python встановимр клієнтську бібліотеку GCP - google-cloud-storage.

Інтеграція API Google Cloud з Python дозволяє вебзастосунку взаємодіяти з сервісами Google Cloud, такими як Google Cloud Storage.

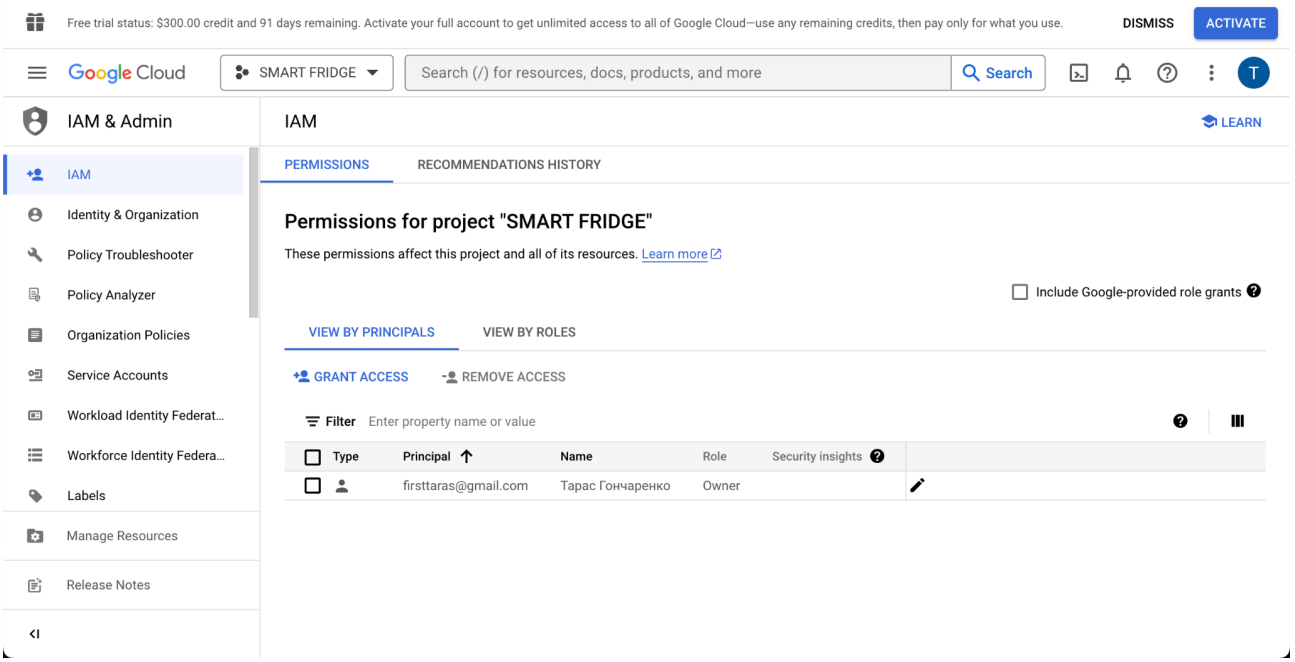

Рисунок 3.3.4 - Скріншот списку аккаунтів із доступом

Для використання сервісів Google Cloud у вебзастосунку, Google потребує автентифікації запиту. Реалізуємо це за допомогою сервісного облікового запису в розділі "IAM & Admin" > "Service accounts".

В результаті створено новий сервісний обліковий запис.

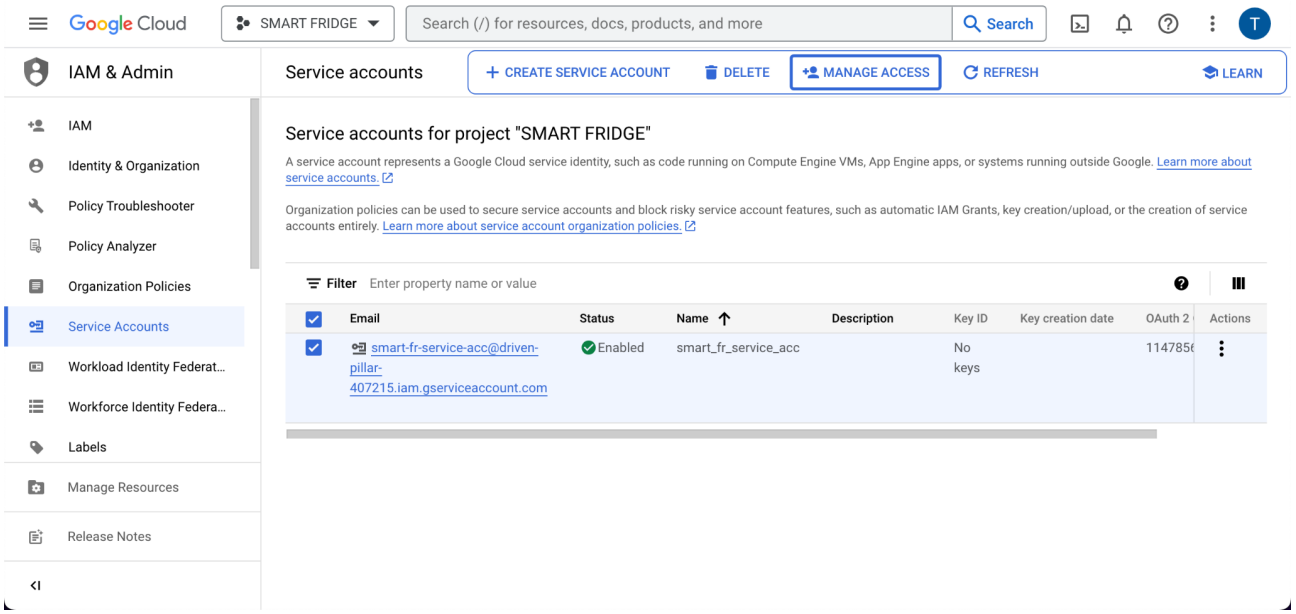

Рисунок 3.3.5 - Скріншот створення сервісного облікового запису

Під час створення сервісного облікового запису вибрано відповідні ролі, для вебзастосунку.[20]

Для захисту сервісного облікового запису створено ключ у форматі JSON.

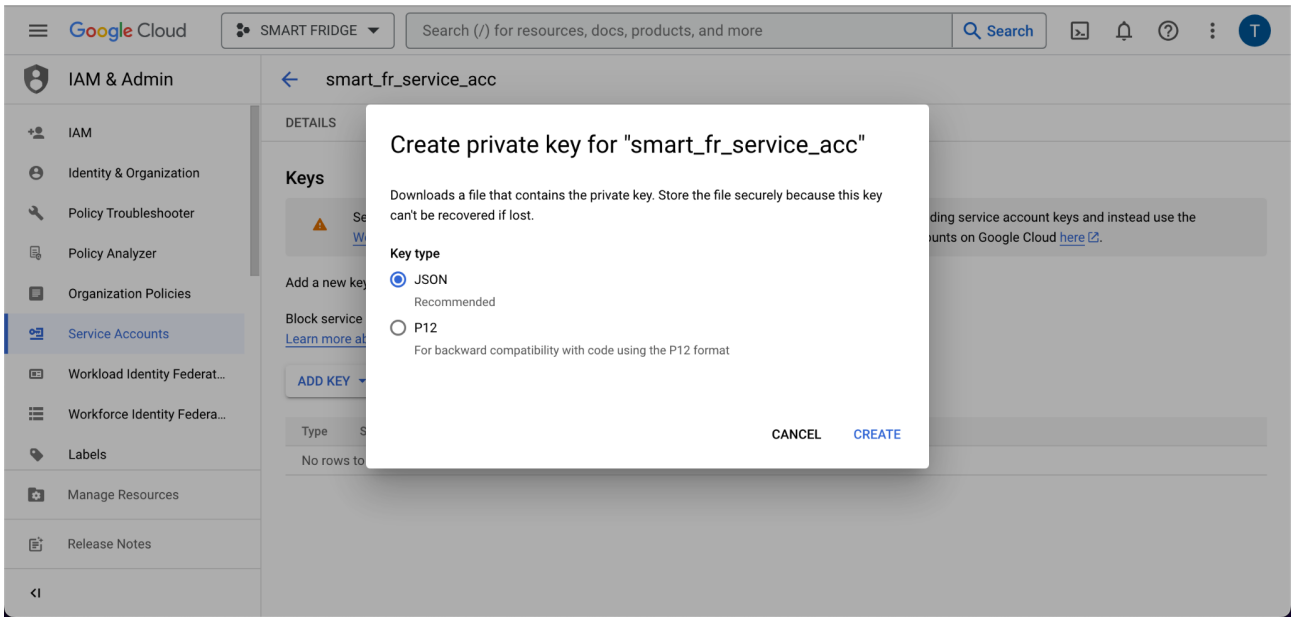

Рисунок 3.3.6 - Скріншот створення ключа

та завантажено JSON-файл з ключем на комп'ютер.

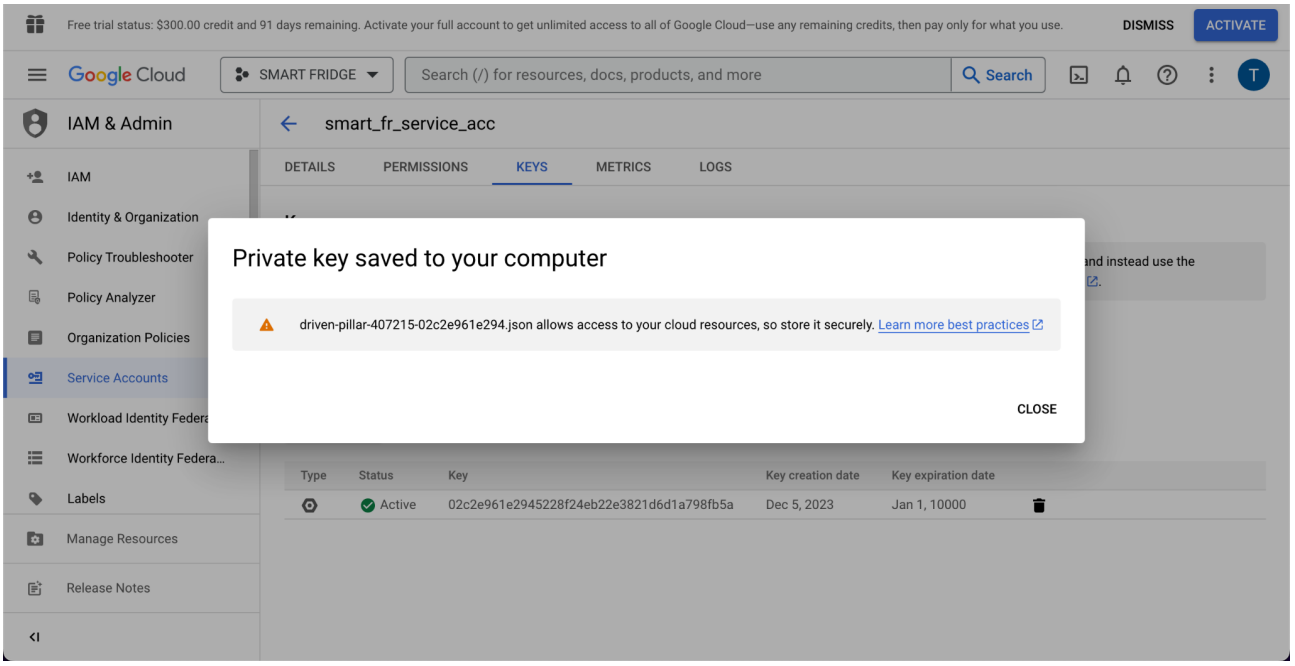

Рисунок 3.3.7 - Скріншот збереження ключа

Внаслідок виконання цих кроків реалізовано можливість вебзастосунку використовувати API Google Cloud для зберігання та обробки даних.

Для зберігання даних у Google Cloud Storage (GCS) в контексті вебзастосунку "Розумний холодильник", використано Python SDK від Google. Нижче наведено лістинг коду, який демонструє, як зберігаються та отримується доступ до даних з GCS.

Перед виконанням коду додатково встановлено бібліотеку google-cloud-storage та налаштовано автентифікацію за допомогою сервісного облікового запису Google Cloud:

```
from google.cloud import storage
import json

def upload_to_gcs(bucket_name, data, destination_blob_name):
    """Завантажує дані у Google Cloud Storage."""
```

```
    # Ініціалізація клієнта GCS
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)

    # Створення об'єкта Blob у бакеті
    blob = bucket.blob(destination_blob_name)

    # Завантаження даних у об'єкт Blob
    blob.upload_from_string(json.dumps(data),
content_type='application/json')
    print(f"Data uploaded to {destination_blob_name} in bucket
{bucket_name}.")

def download_from_gcs(bucket_name, source_blob_name):
    """Завантажує дані з Google Cloud Storage."""
    # Ініціалізація клієнта GCS
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)

    # Отримання об'єкта Blob із бакету
    blob = bucket.blob(source_blob_name)

    # Завантаження даних з об'єкта Blob
    data = json.loads(blob.download_as_string())
    print(f"Data downloaded from {source_blob_name} in bucket
{bucket_name}.")
    return data

# Використання
bucket_name = 'your-bucket-name'
data_to_upload = {
    'user_id': '123',
    'product_name': 'Apple',
    'quantity': 5
}
destination_blob_name = 'user_purchase_history.json'
```

```
# Завантаження даних у GCS
upload_to_gcs(bucket_name, data_to_upload, destination_blob_name)

# Завантаження даних з GCS
downloaded_data = download_from_gcs(bucket_name,
destination_blob_name)
print(downloaded_data)
```

Функція upload_to_gcs приймає назву кошика, дані та ім'я об'єкта, до якого потрібно завантажити дані. Вона перетворює дані у формат JSON та завантажує їх у GCS.

Функція download_from_gcs приймає назву кошика та ім'я об'єкта, з якого потрібно завантажити дані. Вона зчитує дані у форматі JSON з GCS.

Збирання та зберігання даних складний та кропіткий процес, який можна оптимізувати із використанням сучасних надійних методів. Парсинг сторінок інтернет-магазинів є досить ефективним методом для збору даних про продукти, ціни, характеристики товарів, відгуки користувачів та іншу інформацію, яка розміщується на вебсайтах. Ця техніка широко застосовується у сферах електронної комерції, маркетингового аналізу, конкурентної розвідки та автоматизації бізнес-процесів.[21]

Парсинг дозволяє автоматизувати процес збору даних, значно зменшуючи потребу в ручній роботі. Наприклад, замість того, щоб вручну переглядати сотні сторінок інтернет-магазину для збору інформації про продукти, парсер може швидко витягувати ці дані, зберігаючи час та ресурси. Завдяки парсингу можна забезпечити актуальність та точність зібраної інформації. Програмні скрипти можна налаштувати на регулярне оновлення даних, що забезпечує доступ до найсвіжішої інформації. Компанії можуть використовувати парсинг для моніторингу цін та асортименту товарів конкурентів, що є ключовим елементом у стратегії ціноутворення та управлінні асортиментом. Аналіз даних, зібраних з різних інтернет-магазинів, може допомогти виявити загальні тенденції та

переваги споживачів, що є важливим для маркетингового аналізу та планування асортименту.[22]

Наведений спосіб збору інформації також має свої обмеження та виклики. Важливо зазначити, що парсинг вебсайтів може порушувати умови користування вебсайту, а також може бути обмежений захистом від ботів. Не всі вебсайти дозволяють парсинг своїх даних. Необхідно дотримуватися умов використання сайту та місцевого законодавства. Багато сайтів мають заходи проти парсингу, такі як CAPTCHA, що ускладнює автоматичний збір даних. Також, структура веб-сайтів може змінюватися, тому скрипт парсинга може вимагати постійної адаптації.

Методики вилучення даних із вебсайтів класифікуються на три основні категорії: ручні, напівавтоматичні та повністю автоматизовані. Ручні методи включають процес, де оператор (людина) активно моніторить відповідні сторінки і переносить потрібні дані до бази вручну. Такий підхід може похвалитися високоякісним збором даних та простими вимогами до навичок оператора. Проте, цей метод є дуже затратним по часу та ефективності, і велика ймовірність людської помилки може негативно вплинути на точність даних, особливо в завданнях, де потрібна висока точність.

Напівавтоматичні методи передбачають використання спеціалізованого програмного забезпечення, як-от Web Info Extractor, Web Data Extractor, Mozenda, яке після первинної настройки дозволяє автоматизувати збір даних. Це може включати розмітку елементів сторінки оператором через графічний інтерфейс або складання складніших сценаріїв за допомогою регулярних виразів чи XQuery запитів. Перевагою такого підходу є збереження високої якості збору даних при значно вищій швидкості. Однак, людський фактор все ще залишається, а кваліфікація оператора має відповідати високим вимогам для створення ефективних налаштувань.

Автоматичні (інтелектуальні) методи передбачають повне відокремлення людини від процесу вилучення даних. Ці методи автоматично аналізують веб-сторінки та витягують інформацію без втручання оператора. Зі зростом

інтересу до технологій обробки великих даних та дата-майнінгу, саме цей підхід набуває найбільшої популярності. Повністю автоматизований збір даних є найбільш ефективним, але також може супроводжуватись власним набором проблем та складнощів, які потребують уваги.

Для реалізації збору даних про продукти реалізуємо метод парсингу даних з інтернетмагазину. На мові Python використаємо бібліотеки requests для здійснення HTTP-запитів та BeautifulSoup [23] для аналізу HTML-сторінок. Це дозволяє отримати потрібну інформацію та зберегти її у форматі, зручному для обробки та зберігання - у форматі CSV:

```
import requests
from bs4 import BeautifulSoup
import csv

def parse_product_page(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')

    # Дані про продукт містяться у тегах <div class='product'>
    products = soup.find_all('div', class_='product')

    data = []
    for product in products:
        # Отримання даних продукту
        product_name = product.find('h2',
class_='product-name').get_text().strip()
        protein = product.find('span',
class_='protein').get_text().strip()
        carbs = product.find('span',
class_='carbs').get_text().strip()
        # Аналогічні рядки для витягування жирів, цукру, солі,
клітковини, терміну придатності
        # ...
```

```
        product_url = product.find('a',
class_='product-link')['href']

        data.append([product_name, protein, carbs, product_url])

    return data

# Використання функції та зберігання даних у файл CSV
url = 'http://example.com/products'  # Замінюється на URL
веб-сторінки інтернет-магазину
product_data = parse_product_page(url)

# Запис даних у файл CSV
with open('products.csv', 'w', newline='', encoding='utf-8') as
file:
    writer = csv.writer(file)
    writer.writerow(['Name', 'Protein', 'Carbs', 'URL'])
    writer.writerows(product_data)

print("Data saved to products.csv")
```

Цей код адаптується відповідно до структури HTML вебсайту, з якого буде здійснено парсинг.

Розробка функціональності для збору даних від користувачів через веб-інтерфейс та інтеграція збору даних із бекендом вебзастосунку.

Для реалізації збору даних від користувачів у вебзастосунку та інтеграції цього процесу з бекендом, використаємо можливості HTML, JavaScript та Google Cloud.

Створення форми на вебсторінці, яка дозволяє користувачам вводити свої дані, реалізуємо за допомогою HTML та JavaScript:

```
<form id="product-form">
    <input type="text" id="product-name" name="product-name"
placeholder="Назва продукту">
```

```
    <!-- Інші поля форми за необхідності -->
    <button type="submit">Надіслати</button>
</form>
```

Для обробки даних на клієнтській стороні - надсилання форми та збору даних з полів форми - використаємо JavaScript:

```
document.getElementById('product-form').addEventListener('submit',
function(e) {
    e.preventDefault();
    const productName =
document.getElementById('product-name').value;
    // Значення інших полів
    sendDataToServer({ productName: productName });
});
```

Реалізуємо відправку зібраних даних на сервер за допомогою AJAX-запиту:

```
function sendDataToServer(data) {
    fetch('http://your-backend-endpoint.com/api/products', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(data),
    })
    .then(response => response.json())
    .then(data => console.log('Success:', data))
    .catch((error) => console.error('Error:', error));
}
```

Для обробки вхідних запитів на бекенді використаємо Flask, Python вебфреймворк:

```
from flask import Flask, request, jsonify
app = Flask(__name__)

@app.route('/api/products', methods=['POST'])
def add_product():
    data = request.json
    print(data)  # Обробіть дані
    # Тут виконуються додаткові дії, такі як зберігання даних
    return jsonify({'status': 'success', 'data': data})

if __name__ == '__main__':
    app.run(debug=True)
```

Для зберігання даних використаємо Google Cloud Storage:

```
from google.cloud import storage

def upload_data_to_gcs(bucket_name, data, destination_blob_name):
    """Завантажує дані у Google Cloud Storage."""
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_string(data)
    print(f"Data uploaded to {destination_blob_name} in bucket
{bucket_name}.")

# Використання функції у Flask-маршруті
upload_data_to_gcs('your-bucket-name', json.dumps(data),
'product-data.json')
```

Описаний процес включає створення веб-форми для збору даних від користувачів, обробку цих даних на бекенді з використанням Flask, та зберігання даних у Google Cloud Storage.

Для обробки та аналізу зібраних даних реалізуємо інтеграцію з Google BigQuery з можливістю виконання запитів SQL над зібраними даними, аналізу харчових звичок користувачів, історії покупок та уподобань.

Реалізація включає налаштування середовища BigQuery, завантаження даних та виконання SQL-запитів для аналізу.

Використання Google BigQuery є потужним інструментом для швидкого SQL-аналізу великих даних, що дозволяє легко обробляти запити, пов'язані з продуктами, харчовими звичками користувачів та їхніми перевагами.

Для виконання аналізу даних за допомогою Google BigQuery для вебзастосунок, створимо код на Python, який демонструє, як виконано запит до BigQuery для аналізу харчових звичок користувачів. Перед виконанням коду встановимо Google Cloud SDK і налаштуємо автентифікацію для доступу до проекту Google Cloud.

Вихідні дані - таблиця user_food_preferences, яка містить історію покупок користувачів:

```
from google.cloud import bigquery

# Створення клієнта BigQuery
client = bigquery.Client()

# Створення запиту SQL
query = """
    SELECT user_id, product_name, sum(quantity) as total_quantity
    FROM `your_project.your_dataset.user_food_preferences`
    GROUP BY user_id, product_name
    ORDER BY total_quantity DESC
"""

# Виконання запиту
query_job = client.query(query)

# Обробка результатів
```

```
for row in query_job:
    print(f"User ID: {row.user_id}, Product: {row.product_name},
Total Quantity: {row.total_quantity}")
```

Цей код виконує наступні дії:

- імпортує необхідний модуль від Google Cloud BigQuery.
- створює клієнта BigQuery для взаємодії з сервісом.
- створює SQL-запит, який обирає дані про покупки користувачів, групуючи їх за user_id та product_name та розраховуючи загальну кількість купленого продукту.
- виконує запит і друкує результати.[24]

Таблиця user_food_preferences містить поля user_id (ідентифікатор користувача), product_name (назва продукту) і quantity (кількість купленого продукту).

Використаємо Google Cloud SDK для виконання запиту BigQuery:

```
from google.cloud import bigquery

# Ініціалізація клієнта BigQuery
client = bigquery.Client()

# Формулювання SQL-запиту
query = """
SELECT product_name, COUNT(*) as purchase_count
FROM `your_project.your_dataset.your_table`
GROUP BY product_name
ORDER BY purchase_count DESC
"""

# Виконання запиту
query_job = client.query(query)
```

```
# Виведення результатів
for row in query_job:
    print(f"Product: {row.product_name}, Purchases:
{row.purchase_count}")
```

Код виконує запит SQL до BigQuery, який агрегує дані про покупки продуктів та виводить найбільш популярні продукти. [25]

Інтеграція з інструментами візуалізації реалізована із використанням Google Data Studio, для створення дашбордів та графіків на основі результатів BigQuery допомагає в наочному представленні аналітики для кращого розуміння тенденцій та шаблонів.

Інтеграція Google Data Studio для візуалізації даних у вебзастосунок включає вбудовування інтерактивних дашбордів чи звітів безпосередньо у вебсторінки.

Перш ніж інтегрувати звіт в ваш вебзастосунок, потрібно створити звіт у Google Data Studio та отримати URL для вбудовування.

Створимо звіт у Google Data Studio, налаштувавши його з даними з BigQuery чи іншого джерела. Отримаємо URL для вбудовування. Відкриємо створений звіт у Google Data Studio.

Для інтеграції звіту в вебзастосунок, використовуючи отримане посилання, вбудуємо звіт у вебзастосунок:

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard Integration</title>
</head>
<body>
    <h1>Мій Аналітичний Дашборд</h1>
```

```
            <iframe width="600" height="450" src="EMBED_URL"
frameborder="0" style="border:0" allowfullscreen></iframe>
</body>
</html>
```

У цьому коді "EMBED_URL" замінюється на URL, який скопійовано з Google Data Studio. Присутня можливість налаштувати розміри вбудовування (width та height) відповідно до дизайну вебсторінки.

Завантажимо оновлений HTML-файл на хостинговий сервіс. Тепер відвідувачі вебсайту можуть переглядати вбудований дашборд Google Data Studio прямо на вебсторінці. Існує можливість додатково налаштувати відображення та доступ до звіту в Google Data Studio, включаючи управління дозволами доступу та вбудовування інтерактивних елементів.

ВИСНОВКИ

Удосконалення методу управління великими даними на прикладі вебзастосунку демонструє важливість напрямок у розвитку технологій управління. Із застосуванням описаного методу, вебзастосунок реалізує покращення побуту користувачів, ефективно реагуючи на постійні зміни набору даних у режимі реального часу - уподобання користувачів, набір послуг та інше.

З практичної точки зору, індивідуальні переваги для користувачів полягають у забезпеченні оптимального управління запасами їжі, що допомагає зменшити кількість зіпсованих продуктів, забезпечує рекомендації зі здорового харчування, та оптимізує витрати на покупки. Водночас, на більш глобальному рівні, вебзастосунок може сприяти формуванню культури розумного споживання, надаючи інструменти для зменшення відходів їжі та спонукаючи до осмисленого підходу до покупок.

Загальною метою цієї роботи є не лише демонстрація потенціалу досліджуваного методу управління великими даних для покращення щоденного життя людей, але й підкреслення важливості інновацій у досягненні сталого розвитку суспільства. Використання аналітики великих даних в реальному часі для управління, на прикладі вебзастосунку, є одним із прикладів, як технології можуть служити людям та довкіллю [26].

Крім того, використання сучасних програмних методів та підходів в процесі розробки застосунку "Розумний холодильник" має вирішальне значення для забезпечення його надійності, безпеки та користувальницького досвіду. Розробка базується на принципах агільної методології, що дозволяє гнучко реагувати на зміни вимог та швидко адаптуватися до нових технологій. Використання автоматизованих тестів, неперервної інтеграції та розгортання сприяє підвищенню якості продукту та скороченню часу на його впровадження.

У цілому, досліджуваний метод являє собою приклад того, як можна використовувати сучасні технології для створення інноваційних рішень, що

вносять позитивний вклад у життя індивідуумів та суспільства в цілому. Використання великих даних в реальному часі, на прикладі управління продуктами та запасами, із застосуванням хмарних технологій, відкриває нові можливості для підвищення ефективності та сталості у нашому повсякденному житті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Верес О. М., Р. М. Оливко. Класифікація методів аналізу Великих даних. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі. 2017. № 872.
2. У І кварталі 2023 року зростання світового ринку хмарних технологій уперше склало менше 20%. Senior.ua [Електронний ресурс] – Режим доступу до ресурсу: https://senior.ua/news/u-i-kvartal-2023-roku-zrostannya-svtovogo-rinku-hmarnih-tehnology-upershe-sklalo-menshe-20 – Дата доступу: 21.12.2023.
3. Mayer-Schönberger V., Cukier K. Big Data: A Revolution that Will Transform how We Live, Work and Think. John Murray, 2013.
4. Kirk A. Data Visualization: a successful design process. Packt Publishing, 2012.
5. The Food and Agriculture Organization. Sustainable healthy diets – Guiding principles. [Електронний ресурс] – Режим доступу до ресурсу: https://www.fao.org/3/ca6640en/ca6640en.pdf – Дата доступу: 25.10.2023.
6. Arthur C. Internet fridges: the zombie idea that will never, ever happen. The Guardian. [Електронний ресурс] – Режим доступу до ресурсу: https://www.theguardian.com/technology/2014/jan/07/internet-fridge-lg-ces-2014 – Дата доступу: 25.10.2023.
7. Wolf M. Called It: Just as We Predicted, Amazon is Building a Smart Fridge. The Spoon [Електронний ресурс] – Режим доступу до ресурсу: https://thespoon.tech/called-it-just-as-we-predicted-amazon-is-building-a-smart-fridge/ – Дата доступу: 25.10.2023.
8. The Best Smart Thermostat. Wirecutter. [Електронний ресурс] – Режим доступу до ресурсу: https://www.nytimes.com/wirecutter/reviews/the-best-thermostat/ – Дата доступу: 22.12.2023.

9. Samsung. [Електронний ресурс] – Режим доступу до ресурсу: https://www.samsung.com/us/explore/family-hub-refrigerator/overview/ – Дата доступу: 20.11.2023.
10. Закон України "Про захист персональних даних". [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2297-17#Text – Дата доступу: 15.12.2023.
11. Маклаков С.В. Bpwin и Erwin. CASE-средства разработки информационных систем. ДИАЛОГ–МИФИ, 2000.
12. The official home of the Python Programming Language. [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/ – Дата доступу: 15.11.2023.
13. HTML and CSS: Design and Build Websites. [Електронний ресурс] – Режим доступу до ресурсу: https://www.perlego.com/book/2761923/html-and-css-design-and-build-websites?utm_source=google&utm_medium=cpc&campaignid=17287656381&adgroupid=134138478662&gclid=CjwKCAiAp5qsBhAPEiwAP0qeJlIHEd1JX89H6iOhRqmL8sVNfa7rNAbYF8rjHKSakyFiCJueEk5qOxoCikcQAvD_BwE – Дата доступу: 18.12.2023.
14. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. Random House Publishing Group, 2011.
15. Amazon Web Services. [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/?nc1=h_ls – Дата доступу: 15.11.2023.
16. Google Cloud. [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/ – Дата доступу: 15.11.2023.
17. Johnson B. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. Wiley, 2019.
18. GitHub. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ – Дата доступу:15.11.2023.

19. Будуємо сховище для даних з Google Cloud Platform. [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/46394/ – Дата доступу: 25.11.2023.
20. Srinivasan V., Ravi J., Raj J. Google Cloud Platform for Architects: Design and manage powerful cloud solutions. Packt Publishing, 2018.
21. Мітчелл P. Web Scraping with Python: Collecting More Data from the Modern Web". O'Reilly Media, 2018.
22. Мендельсон Х. Web Scraping for Data Science: Practical Examples and Best Practices. O'Reilly Media, 2020.
23. Selenium vs. Beautiful Soup: A Full Comparison [Електронний ресурс] – Режим доступу до ресурсу: https://www.blazemeter.com/blog/seleniumvs-beautiful-soup-python. – Дата доступу: 21.11.2023.
24. Lakshmanan V., Tigani J. Google BigQuery: The Definitive Guide. O'Reilly Media, 2021.
25. Lakshmanan V. Data Science on the Google Cloud Platform, 2nd Edition. O'Reilly Media, 2022.
26. Гончаренко Т., Ситніков Д. *Розробка та дослідження методу управління великими даними в реальному часі*. Молодіжна наукова ліга: тези доп. IV Міжнародна студентська наукова конференція «КОНЦЕПТ НАУКИ XXI: СТРАТЕГІЇ, МЕТОДИ ТА НАУКОВІ ІНСТРУМЕНТИ». м.Вінниця, 3 лист. 2023 р., С. 158—160, [Електронний ресурс] – Режим доступу до ресурсу: https://archive.liga.science/index.php/conference-proceedings/issue/view/inter-03.11.2023?utm_source=eSputnik-promo&utm_medium=email&utm_campaign=MNL_Konferenc%D1%96ja_%7C_Status_:_opubl%D1%96kovano&utm_content=2293651120 – Дата доступу: 28.11.2023.