НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" червня 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Комп'ютерна система моніторингу ресурсів об'єкту»

Виконав (-ла):

студент (-ка) IV курсу, групи КВ-11
(шифр групи)

Пильова Діана Миколаївна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент каф. СПіСКС, к.т.н. Морозов К.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент ст. викладач каф. ПМА ФПМ Мальчиков В.В. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____ 202_р.

ЗАВДАННЯ
на дипломний проєкт студента

Пильової Діани Миколаївни
(прізвище, ім'я, по батькові)

1. Тема проєкту «Комп'ютерна система моніторингу ресурсів об'єкту»
_____,
керівник проєкту Доцент каф. СПіСКС, к.т.н. Морозов К.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету № 1808-С від «29» травня 2025 р.

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: див. ТЗ.

4. Зміст пояснювальної записки:

Розглянуто аналіз існуючих рішень, що дозволило оцінити переваги та недоліки аналогічних систем і обґрунтувати доцільність розробки власного рішення. Проведено огляд обраних апаратних і програмних технологій, які використовувалися під час реалізації проєкту, із поясненням причин їх вибору. Описано архітектуру комп'ютерної системи, процес її розробки, принципи роботи окремих компонентів, а також способи взаємодії між ними.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

1) Логічна модель.Схема бази даних
2) Аналіз зображення. Схема алгоритму
3) Розрахунок відсотка заповнення полиць. Схема алгоритму
4) Зовнішній модуль управління. Схема алгоритму

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц. каф. СПіСКС, к.т.н		

7. Дата видачі завдання ___ _____ 202_ р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Видача завдання на дипломне проєктування	10.10.2024	ВИК
2.	Вивчення літератури за тематикою проекту	10.10.2024	ВИК
3.	Розроблення та узгодження технічного завдання	13.03.2025	ВИК
4.	Аналіз існуючих рішень	21.03.2025	ВИК
5.	Підготовка матеріалів першого розділу дипломного проекту	01.04.2025	вик
6.	Підготовка матеріалів другого розділу дипломного проекту	08.04.2025	вик
7.	Підготовка матеріалів третього розділу дипломного проекту	23.04.2025	вик
8.	Підготовка матеріалів четвертого розділу дипломного проекту	15.05.2025	вик
9.	Підготовка графічної частини дипломного проекту	27.05.2025	вик
10.	Оформлення документації дипломного проекту	29.05.2025	вик

Студент _____ Діана ПИЛЬОВА
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проєкту _____ Костянтин МОРОЗОВ
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (52 с., 21 рис., 3 додатки).

Об’єкт розробки – створення комп’ютерної системи моніторингу ресурсів об’єкту, яка дозволяє здійснювати контроль стану холодильника, зокрема заповненості його полиць та дотримання термінів придатності продуктів.

Комп’ютерна система дозволяє: здійснювати обробку фотографій полиць холодильника, отриманих із камер. Система аналізує зображення для визначення рівня заповненості полиць продуктами.

Основною метою розробки є створення системи, яка дозволяє аналізувати заповненість полиць холодильника, автоматично відслідковувати терміни придатності продуктів та надсилати повідомлення користувачу про необхідність використання продуктів, термін придатності яких добігає кінця.

В ході розробки:

- Проведено аналіз методів побудови існуючих комп’ютерних систем для моніторінгу холодильника;
- Сформульовано вимоги до комп’ютерної системи;
- Ознайомлено з документацією бібліотеки `telegram.ext`
- Створено власного чат-бота Telegram;
- Розроблено алгоритм аналізу фотографій для визначення заповненості полиць;
- Спроектовано архітектуру системи, визначено основні компоненти та їх взаємодію;
- Підготовлено повну документацію з розробки, включаючи опис архітектури системи, алгоритмів та коду;
- Написано висновки до кожного розділу, а також загальний висновок, що підсумовує результати роботи
- Розроблена система "Fridge Control" забезпечує зручний контроль за продуктами, дозволяє мінімізувати їх втрати та покращує взаємодію

користувача із холодильником.

Ключові слова: КОМП'ЮТЕРНА СИСТЕМА, МОНІТОРИНГ РЕСУРСІВ, ХОЛОДИЛЬНИК, ОБРОБКА ЗОБРАЖЕНЬ, КОНТРОЛЬ ЗАПОВНЕНОСТІ, ТЕРМІН ПРИДАТНОСТІ, TELEGRAM-БОТ, АНАЛІЗ ДАНИХ, ОПОВІЩЕННЯ КОРИСТУВАЧА.

ABSTRACT

The qualification work includes an explanatory note (52 pages, 21 figures, 3 appendices).

The object of development is creating a computer system for monitoring the resources of an object, which allows controlling the state of the fridge, specifically the occupancy of its shelves and compliance with the expiration dates of products.

The computer system allows: processing photos of the fridge's shelves obtained from cameras. The system analyzes the images to determine the level of shelf occupancy with products.

The main goal of the development is to create a system that allows analyzing the occupancy of the fridge's shelves, automatically tracking the expiration dates of products, and sending notifications to the user about the necessity of using products whose expiration dates are approaching.

In the course of development:

- An analysis of methods for building existing computer systems for fridge monitoring was carried out;
- Requirements for the computer system were formulated;
- Familiarization with the documentation of the`telegram.ext` library;
- A custom Telegram bot was created;
- An algorithm for analyzing photos to determine shelf occupancy was developed;
- The system architecture was designed, and the main components and their interaction were defined;
- Full development documentation was prepared, including descriptions of the system's architecture, algorithms, and code;
- Conclusions were written for each section, as well as a general conclusion that summarizes the results of the work.
- The developed system "Fridge Control" provides convenient control over products, minimizes their waste, and improves user interaction with the

fridge.

Keywords: COMPUTER SYSTEM, RESOURCE MONITORING, FRIDGE, IMAGE PROCESSING, OCCUPANCY CONTROL, EXPIRATION DATE, TELEGRAM BOT, DATA ANALYSIS, USER NOTIFICATIONS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Комп’ютерна система	4		
			моніторингу ресурсів			
			об’єкту			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Комп’ютерна система	2		
			моніторингу ресурсів			
			об’єкту			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Комп’ютерна система	52		
			моніторингу ресурсів			
			об’єкту			
			Пояснювальна записка			
	А1	ІАЛЦ.045490.005 Д1	Логічна модель.	1		
			Схема бази даних			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Пильова Д. М.			Комп’ютерна система моніторингу ресурсів об’єкту **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Морозов К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ.045490.006 Д2	Аналіз зображення.	1		
			Схема алгоритму			
	А1	ІАЛЦ.045490.007 Д3	Розрахунок відсотка	1		
			заповнення полиць.			
			Схема алгоритму			
	А1	ІАЛЦ.045490.008 Д4	Зовнішній модуль	1		
			управління.			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп’ютерна система моніторингу ресурсів об’єкта».

Галузь застосування: автоматизація домашнього господарства, зокрема облік продуктів харчування, контроль їхнього терміну придатності та оптимізація використання вмісту холодильника.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення комп’ютерної системи моніторингу ресурсів холодильникаСистема використовує технології комп’ютерного зору та обробки зображень для аналізу фотографій вмісту холодильника, визначення рівня заповненості полиць, збереження інформації про наявні продукти та нагадування користувачу про наближення терміну їх придатності. Такий застосунок спрощує облік харчових продуктів у побуті та сприяє зменшенню кількості харчових відходів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерела інформації включають технічну й науково-технічну літературу, технічну документацію та статті з періодичних і онлайн-видань.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- підтримка взаємодії через месенджер Telegram;
- можливість обробки зображень холодильника;
- аналіз заповнення полиць;
- збереження даних про продукти;
- нагадування про продукту термін яких спливає через сповіщення;
- наявність зручної системи сповіщень ініціатора та виконавця;
- підтримка багатокористувацької взаємодії;
- прив'язки файлів ініціатора та виконавця до заявок;
- зберігання заявок;
- аналітичні можливості.

5.2 Вимоги до апаратного забезпечення

- смартфон або комп'ютер з доступом до Telegram;
- Оперативна пам’ять: 2 Гб;
- Доступу до Інтернету (GPRS, EDGE, 3G, 4G).

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Android, iOS, Linux, ChromeOS.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.10.2024
2.	Розроблення та узгодження технічного завдання	13.03.2025
3.	Аналіз існуючих рішень	21.03.2025
4.	Підготовка матеріалів першого розділу дипломного проекту	01.04.2025
5.	Підготовка матеріалів другого розділу дипломного проекту	08.04.2025
6.	Підготовка матеріалів третього розділу дипломного проекту	23.04.2025
7.	Підготовка матеріалів четвертого розділу дипломного проекту	15.05.2025
8.	Підготовка графічної частини дипломного проекту	27.05.2025
9.	Оформлення документації дипломного проекту	29.05.2025
10.	Попередній огляд матеріалів диплому на кафедрі	30.05.2025

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.004 ПЗ	Комп’ютерна система	52		
			моніторингу ресурсів			
			об’єкту			
			Пояснювальна записка			
	А1	ІАЛЦ.045490.005 Д1	Логічна модель.	1		
			Схема бази даних			
	А1	ІАЛЦ.045490.006 Д2	Аналіз зображення.	1		
			Схема алгоритму			
	А1	ІАЛЦ.045490.007 Д3	Розрахунок відсотка	1		
			заповнення полиць.			
			Схема алгоритму			
	А1	ІАЛЦ.045490.008 Д4	Зовнішній модуль	1		
			управління.			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Пильова Д. М.			Комп’ютерна система моніторингу ресурсів об’єкта **Відомість технічного проєту**	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Морозов К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Комп’ютерна система моніторингу ресурсів об’єкта

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – Програмне забезпечення;

SDK — Набір із засобів розробки (software Development Kit);

JDBC — З'єднання з базами даних на Java (Java DataBase Connectivity);

СУБД — Система управління базами даних

API — Інтерфейс програмування додатків (Application Programming Interface).

ВСТУП

У сучасному світі інформаційні технології стрімко інтегруються в усі сфери життєдіяльності людини, сприяючи підвищенню продуктивності та зручності життя. Одним із перспективних напрямів є розробка комп'ютерних систем моніторингу, які забезпечують ефективний контроль за станом різноманітних об'єктів та їхніх ресурсів у режимі реального часу.

Особливої актуальності набуває проблема управління харчовими ресурсами через зростаючу усвідомленість суспільства щодо необхідності раціонального споживання, зменшення харчових відходів та дбайливого ставлення до навколишнього середовища. Зіпсовані продукти, надмірне заповнення холодильного обладнання або несвоєчасне виявлення прострочених товарів призводять до харчових відходів і фінансових втрат як у побуті, так і в комерційній сфері.

У зв'язку з цим виникає потреба у розробці комп'ютерної системи моніторингу ресурсів холодильника. Така система, що здійснює контроль заповненості полиць та термінів придатності продуктів, сприятиме оптимізації зберігання, зменшенню харчових відходів та підвищенню зручності користування.

Метою цього проекту є створення інструменту, який надаватиме інформацію про вміст холодильника та його поточний стан.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Вступ

Для обґрунтування вибору розробки системи моніторингу холодильника проведено детальний аналіз існуючих рішень на ринку. Сучасне суспільство, з його прискореним темпом життя, дедалі більше усвідомлює необхідність ефективного управління домашніми ресурсами. У контексті ведення домогосподарства, оптимізація витрат часу, фінансів та зменшення впливу на навколишнє середовище стають ключовими пріоритетами.

Управління запасами продуктів у холодильнику є одним з таких аспектів, який часто ігнорується, але має значний потенціал для покращення. Проблема полягає не лише у відстеженні термінів придатності, щоб уникнути споживання зіпсованих продуктів, а й у відсутності точної інформації про поточну заповненість холодильника. Це призводить до низки негативних наслідків, таких як:

1) Надмірні закупівлі: Люди часто купують продукти, які вже є в холодильнику, через відсутність чіткого уявлення про наявні запаси. Це не лише призводить до зайвих витрат, а й сприяє накопиченню продуктів, які згодом можуть зіпсуватися.
2) Збільшення харчових відходів: Наявність великої кількості невикористаних або забутих продуктів у холодильнику значно збільшує обсяг харчових відходів. Це є серйозною проблемою як з економічної, так і з екологічної точки зору.
3) Неефективне використання ресурсів: Час, витрачений на повторні поїздки до магазину за продуктами, які вже є вдома, або на утилізацію зіпсованої їжі, є втраченим ресурсом.
4) Відсутність планування: Без точної інформації про вміст холодильника складно ефективно планувати меню, що може призвести до незбалансованого харчування або непередбачених витрат на готову їжу.

Визначивши ці ключові проблеми, перейдемо до аналізу існуючих технологічних рішень, які розроблені для їх подолання. Розуміння переваг та недоліків наявних підходів дозволить нам обґрунтувати вибір власного рішення та його унікальні переваги.

1.1 Розумні холодильники. Огляд ринку

З розвитком Інтернету речей (IoT) та поширенням концепції "розумного" будинку, побутова техніка трансформується, пропонуючи нові рівні автоматизації та інтеграції. Серед таких інновацій особливе місце посідають розумні холодильники, які виходять за рамки традиційного зберігання продуктів, надаючи користувачам розширені можливості для моніторингу вмісту холодильника.

Вони оснащені внутрішніми камерами, які дозволяють користувачам візуально перевіряти вміст холодильника дистанційно через спеціальний застосунок на смартфоні. Деякі моделі, наприклад, певні лінійки Samsung Family Hub (Рис. 1), використовують технології штучного інтелекту для розпізнавання продуктів, що може допомогти у складанні списків покупок та нагадуваннях про наявність певних інгредієнтів [1].

Однак, більшість сучасних розумних холодильників не надають кількісної оцінки заповненості полиць (наприклад, у відсотках). Хоча візуальний доступ є корисним, він не забезпечує автоматизованого контролю заповненості та термінів придатності, що є метою нашої розробки. Розумні холодильники, що оснащені функцією зчитування інформації з упаковок, на перший погляд здаються зручним і технологічно досконалим рішенням для контролю за термінами придатності продуктів. Вони здатні автоматично сканувати штрих-коди або етикетки, зчитувати дати з упаковок та надсилати користувачеві сповіщення про наближення закінчення терміну зберігання[1].

Однак на практиці така система часто виявляється недосконалою й далекою від ідеалу. По-перше, ефективність цієї функції безпосередньо залежить від якості друку на упаковках і точності камер. У реальних умовах багато

продуктів мають нечитабельні або нечітко надруковані дати, які складно розпізнати навіть людині, не кажучи вже про автоматизовану систему. Камери можуть не зафіксувати важливу інформацію через відблиски, зміну кута огляду чи погане освітлення. Унаслідок цього система або не розпізнає термін придатності зовсім, або фіксує його некоректно, що може призвести до хибних сповіщень або, навпаки, до їх відсутності.

По-друге, далеко не всі продукти мають штрих-коди чи QR-коди з термінами придатності. Багато свіжих товарів, як-от фрукти, овочі або м'ясо, часто купуються без упаковки або маркуються вручну. У таких випадках холодильник втрачає здатність ефективно відстежувати терміни зберігання, що значно знижує практичну користь заявленої функції.

Окремо слід відзначити залежність таких систем від стабільної роботи камер, програмного забезпечення та синхронізації із зовнішнім мобільним застосунком. У разі збою з’єднання або оновлення ПЗ функціональність холодильника може бути частково або повністю втрачена.

Таким чином, попри загальне зростання технологічного рівня побутової техніки, наявні реалізації розумних холодильників демонструють певні функціональні обмеження, зокрема у сфері точного контролю за заповненістю полиць та обліку термінів зберігання харчових продуктів. Ці недоліки створюють передумови для розробки альтернативного підходу — такого, що дозволяє реалізувати зазначені функції без повної апаратної інтеграції у холодильник, наприклад, шляхом використання комп’ютерного зору на рівні окремого застосунку.

Рис. 1. Приклад холодильника Samsung Family Hub

Переваги розумних холодильників:

1. Інтеграція: функціональність моніторингу вбудована безпосередньо в пристрій;
2. Зручність перегляду: візуальний доступ до вмісту через мобільний застосунок;

Недоліки розумних холодильників:

1. Висока вартість: розумні холодильники є значно дорожчими за звичайні.
2. Обмежений контроль заповненості: існуючі моделі не надають автоматизованої оцінки рівня заповненості полиць.
3. Залежність від екосистеми виробника: функціональність може бути обмежена сумісністю з іншими пристроями та сервісами виробника.

1.2 Мобільні застосунки для обліку продуктів

Ще одним поширеним підходом до управління харчовими запасами є використання мобільних застосунків, таких як Kitchen Pal (Рис.2) та NoWaste, які орієнтовані на зменшення харчових відходів та покращення планування закупівель. Ці додатки надають користувачам можливість самостійно фіксувати інформацію про наявні продукти, їх кількість, терміни придатності та категорії.

На основі введених даних користувачі можуть отримувати сповіщення про наближення кінцевих строків зберігання, що допомагає уникнути псування продуктів та економити кошти.

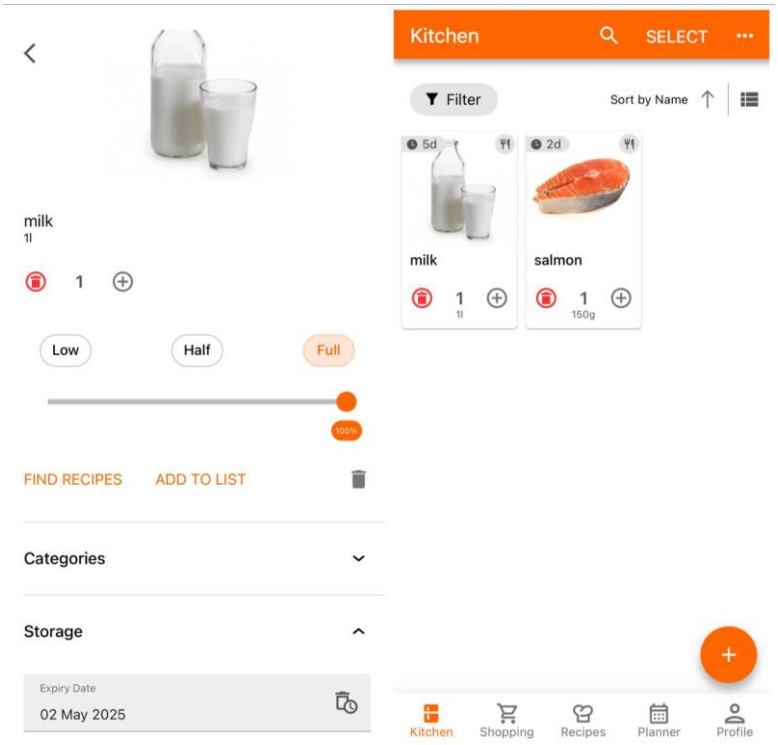

Рис. 2. Застосунок KitchenPal

Основні функції подібних застосунків:

1. Інвентаризація продуктів: користувачі можуть вносити назви товарів, їх кількість, категорію та термін придатності;
2. Автоматичні нагадування: система надсилає повідомлення про продукти, які скоро зіпсуються;
3. Генерація списків покупок: ґрунтуючись на залишках у холодильнику;
4. Сканування штрихкодів: для прискорення процесу додавання нових товарів;
5. Підтримка рецептів: деякі програми пропонують рецепти на основі наявних продуктів;
6. Мультикористувацький доступ: дозволяє членам сім'ї синхронізувати інформацію між кількома пристроями.

Переваги мобільних застосунків для обліку продуктів

1. Доступність: широке розповсюдження серед користувачів смартфонів.
2. Економічність: багато таких застосунків є безкоштовними або мають низьку вартість.

Недоліки мобільних застосунків для обліку продуктів

1. Відсутність візуального контролю: немає можливості оцінити реальну заповненість полиць.
2. Вимога встановлення застосунку: деякі користувачі не бажають встановлювати додаткові програми через обмеження пам’яті пристрою або безпекові міркування.
3. Мовний бар’єр: інтерфейс багатьох популярних застосунків недоступний українською мовою, що може створювати незручності для користувачів в Україні.

1.3 Переваги власного рішення

На відміну від дорогих розумних холодильників, які часто поставляються з безліччю функцій, що можуть ніколи не бути використані, "Fridge Control" зосереджується на двох критично важливих аспектах: ефективному моніторингу вмісту холодильника та автоматичному відстеженні термінів придатності продуктів. Такий цілеспрямований підхід дозволяє користувачам уникнути значних фінансових витрат на придбання нового обладнання, при цьому надаючи дійсно корисні інструменти для оптимізації їхніх запасів їжі.

Порівняно з мобільними застосунками для обліку продуктів, які не інтегровані з фактичним станом холодильника, "Fridge Control" пропонує більш тісну взаємодію з вмістом завдяки обробці фотографій. Це забезпечує об’єктивнішу оцінку заповненості полиць.

Використання Telegram-бота як основного інтерфейсу користувача усуває необхідність встановлення додаткових застосунків, роблячи систему

максимально доступною та зручною для широкої аудиторії в Україні, де цей месенджер є популярним. Користувачі отримуватимуть своєчасну та актуальну інформацію про стан своїх продуктів безпосередньо у звичному каналі комунікації, що робить взаємодію з системою простою та інтуїтивно зрозумілою.

Підсумовуючи, розроблена система "Fridge Control" являє собою економічно вигідне, зручне та ефективне рішення для управління харчовими продуктами в холодильнику. Вона сприяє значному зменшенню харчових відходів, допомагає оптимізувати витрати на продукти харчування та покращує загальну взаємодію користувача зі своїм холодильником, не вимагаючи при цьому придбання дорогого розумного холодильника або виконання складних та тривалих налаштувань додаткового обладнання.

Висновки

Аналіз існуючих рішень показав, що наявні на ринку технології для моніторингу стану холодильників мають як переваги, так і суттєві недоліки.

Таким чином розроблена комп’ютерна система "Fridge Control" демонструє низку важливих переваг:

1. Вона поєднує в собі можливості автоматичного аналізу заповненості холодильника через обробку фотографій;
2. Забезпечує контроль термінів придатності продуктів із автоматичними повідомленнями користувачу;
3. Використовує популярну платформу Telegram як інтерфейс взаємодії, що забезпечує доступність і простоту використання;
4. Є економічно вигідною альтернативою дорогим смарт-рішенням. Не потребує складної інтеграції або дорогого спеціалізованого обладнання.

На відміну від комплексних систем із надмірною кількістю функцій, "Fridge Control" залишається сконцентрованим на реальній побутовій задачі — зменшенні харчових втрат через ефективну організацію простору холодильника. Простота рішень і чіткий фокус роблять систему практичною, стабільною та

придатною для повсякденного використання.

2 ОГЛЯД ТЕХНОЛОГІЙ

2.1 Вибір мови програмування

Вибір мови програмування є одним з ключових етапів у розробці будь-якої програмної системи, оскільки він суттєво впливає на ефективність розробки, продуктивність системи, її масштабованість та легкість підтримки. Для проєкту "Fridge Control" було розглянуто кілька провідних мов програмування, кожна з яких має свої унікальні переваги та недоліки: Java, Python та Go.

2.1.1 Java

Java вже десятиліттями залишається однією з найпопулярніших і найпотужніших мов програмування, відомою своєю філософією "напиши один раз, запускай всюди" завдяки віртуальній машині Java JVM [2]. Ця платформа надає високу продуктивність та масштабованість, що робить її привабливим вибором для ботів, які мають обробляти значні обсяги запитів або працювати з мінімальною затримкою. JVM ефективно використовує системні ресурси, дозволяючи боту легко розширюватися для обслуговування великої кількості одночасних користувачів.

Однією з ключових переваг Java є її надійність та стабільність. Завдяки суворій типізації та зрілому механізму обробки винятків, Java сприяє створенню високостабільних та стійких до збоїв ботів. Більшість потенційних помилок виявляються вже на етапі компіляції, що суттєво зменшує ймовірність збоїв під час виконання, що є критично важливим для систем, які функціонують цілодобово. Величезна та зріла екосистема Java надає розробникам доступ до безлічі бібліотек та фреймворків: від інструментів для роботи з мережевими протоколами (HTTP, WebSockets) до інтеграції з базами даних (JDBC) та готових SDK для популярних месенджерів (включно Telegram). Це значно прискорює розробку навіть найскладніших ботів. Крім того, вбудована підтримка багатопоточності в Java дозволяє ефективно обробляти паралельні запити від користувачів, не блокуючи основну роботу бота, що є незамінним для інтерактивних систем. Завдяки широкому використанню в корпоративному

секторі, Java є відмінним вибором для ботів, яким потрібна глибока інтеграція з існуючою корпоративною інфраструктурою [3].

Однак, Java має і свої недоліки. Її синтаксис є більш багатослівним порівняно з Python, що може призвести до збільшення обсягу коду для виконання тих самих завдань. Це, в свою чергу, може подовжити час розробки, особливо для невеликих проєктів або на етапах швидкого прототипування. Крім того, додатки на Java можуть споживати більше оперативної пам'яті через роботу JVM, що може бути фактором для систем з обмеженими апаратними ресурсами.

2.1.2 Go

Go, або Golang, розроблена компанією Google, швидко здобула визнання завдяки своїй орієнтації на високу продуктивність, ефективність та особливо на конкурентне програмування [4]. Ці характеристики роблять Go надзвичайно привабливим для розробки ботів, які вимагають швидкої обробки запитів та ефективного використання ресурсів. Go компілюється безпосередньо в нативний код, що забезпечує надзвичайно високу швидкість виконання та мінімальне споживання системних ресурсів. Це є ідеальною властивістю для ботів, які повинні оперативно реагувати на дії користувачів або працювати на апаратних платформах з обмеженими можливостями.

Однією з найсильніших сторін Go є її вбудована підтримка конкурентності через механізми горутин та каналів. Горутини – це легковагові потоки, які дозволяють легко писати паралельний код, а канали спрощують безпечний обмін даними між ними [5]. Ця архітектура робить Go ідеальною мовою для створення ботів, які повинні одночасно обробляти велику кількість вхідних з'єднань або виконувати фонові завдання без блокування основного циклу роботи. Швидкість компіляції Go також є значною перевагою, прискорюючи цикл розробки та тестування навіть для великих проєктів. Простий, зрозумілий синтаксис Go є мінімалістичним, що сприяє написанню "чистого" коду, який легко підтримувати та масштабувати. Відсутність складних абстракцій робить Go легкою для вивчення та розуміння. Крім того, компіляція в один статичний

бінарний файл значно спрощує розгортання ботів, оскільки не потрібно встановлювати додаткові залежності або рантайми на цільовій системі. Природна орієнтація Go на мережеві застосунки та архітектуру мікросервісів робить її чудовим вибором для розробки ботів, які часто є частиною більшої розподіленої системи.

Незважаючи на значні переваги, Go має і деякі обмеження. Хоча екосистема Go динамічно розвивається, вона все ще пропонує меншу кількість готових бібліотек для дуже специфічних інтеграцій, порівняно з Python або Java. Це може вимагати більшої ручної розробки для певних функціональних можливостей бота. Крім того, Go має менш гнучку систему об'єктно-орієнтованого програмування, не підтримуючи успадкування класів у традиційному сенсі, а використовуючи натомість композицію та інтерфейси. Це може вимагати адаптації для розробників з досвідом у Java або Python. До версії Go 1.18 мова також не мала підтримки загальних типів (Generics), що інколи ускладнювало написання гнучкого та повторно використовуваного коду. Хоча зараз дженерики доступні, деякі старіші бібліотеки можуть їх не використовувати.

2.1.3 Python

Python є інтерпретованою, високорівневою мовою програмування, яка підтримує різні парадигми програмування, включаючи об'єктно-орієнтований, імперативний та функціональний стилі. Її популярність стрімко зростає завдяки унікальному поєднанню простоти, потужності та величезної гнучкості, що робить її винятково привабливою для широкого спектру застосунків, включаючи розробку ботів.

Величезна екосистема Python є, мабуть, її найсильнішою стороною, особливо для розробки ботів. Існує безліч високоякісних бібліотек, які значно прискорюють процес розробки, надаючи готові рішення для багатьох типових завдань. Це включає специфічні бібліотеки для взаємодії з різними API месенджерів (наприклад, python-telegram-bot). Це дозволяє розробникам

зосередитись на унікальних аспектах проєкту "Fridge Control", уникаючи необхідності "винаходити велосипед".

Після ретельного порівняльного аналізу, було прийнято рішення використовувати Python як основну мову програмування для розробки комп'ютерної системи "Fridge Control". Цей вибір був зумовлений кількома ключовими факторами, які найкращим чином відповідали цілям та вимогам проєкту.

По-перше, швидкість розробки та прототипування була критичним пріоритетом для "Fridge Control". Можливість швидко реалізовувати новий функціонал та оперативно вносити зміни є надзвичайно важливою для проєкту, що постійно розвивається. Простий синтаксис Python та багата бібліотека, особливо ті, що призначені для взаємодії з різними API, дозволяють значно прискорити цей процес. Це дає змогу швидко створювати та тестувати інтерактивні елементи бота, що є суттєвою перевагою. Як зазначає Ерік Меттес у своїй книзі "Python Crash Course" [6], однією з головних особливостей Python є можливість виконувати значні обсяги роботи за допомогою меншої кількості рядків коду порівняно з багатьма іншими мовами. Ця лаконічність значно прискорює ітерації розробки, що є особливо цінним для швидкого створення мінімально життєздатних продуктів (MVP) або для тестування нових функцій бота.

По-друге, читабельність та легкість підтримки коду є фундаментальними для "Fridge Control". Python чудово відповідає цим вимогам, забезпечуючи легку масштабованість проєкту в майбутньому. Його зрозумілий та лаконічний синтаксис допомагає створювати "чистий" код, який легко підтримувати, тестувати та розширювати в майбутньому.

По-третє, доступність ресурсів та спеціалізованих бібліотек для розробки ботів є неперевершеною в екосистемі Python. Величезна кількість готових рішень для роботи з базами даних, комп’ютерним зором та інтерфейсами користувача, а також специфічні бібліотеки для створення ботів, значно

спрощують інтеграцію всіх необхідних компонентів системи "Fridge Control", включаючи її інтерактивну складову. Зокрема, для обробки зображень, які є ключовим елементом системи для аналізу заповненості полиць, використовується бібліотека OpenCV [7]. Ця бібліотека надає широкий спектр функцій для комп'ютерного зору, включаючи завантаження, обробку, аналіз та розпізнавання зображень, що є необхідним для визначення рівня заповненості полиць холодильника. Для реалізації взаємодії з користувачем через Telegram було обрано бібліотеку python-telegram-bot (telegram.ext), яка значно спрощує роботу з Telegram Bot API, дозволяючи легко створювати та керувати чат-ботом для обміну повідомленнями та сповіщеннями. Для роботи з базою даних, де зберігається інформація про продукти та розклади сповіщень, застосовується стандартна бібліотека sqlite3, яка дозволяє ефективно взаємодіяти з базами даних SQLite. Це дозволяє зосередитися на унікальній логіці проєкту, а не на базовій інфраструктурі.

І, нарешті, велика та активна спільнота Python є безцінним ресурсом. Вона забезпечує легкий доступ до рішень типових проблем, надає підтримку та сприяє постійному розвитку мови та її інструментів, що є вкрай важливим для будь-якого нового проєкту.

Хоча Java та Go пропонують видатну продуктивність та чудові можливості для конкурентності, що робить їх привабливими для ботів з екстремально високим навантаженням, їхні переваги не є критичними для специфіки "Fridge Control". Основний акцент у цьому проєкті робиться на логіці управління, ефективній взаємодії з зображеннями та зручності розробки. Таким чином, Python є найкращим вибором, забезпечуючи ідеальний баланс між швидкістю розробки, легкістю підтримки та всеосяжними функціональними можливостями для реалізації всіх запланованих функцій системи "Fridge Control".

2.2 Бібліотека telegram.ext

Для інтеграції з Telegram Bot API було використано бібліотеку python-

telegram-bot [8], яка є однією з найпопулярніших і найбільш підтримуваних бібліотек для створення Telegram-ботів мовою програмування Python. Зокрема, застосовується її модуль telegram.ext, який надає зручний фреймворк з високим рівнем абстракції для побудови логіки бота, обробки подій, управління розмовами та взаємодії з користувачем.

Модуль telegram.ext базується на архітектурі диспетчеризації (dispatcher pattern), де всі вхідні події (оновлення, повідомлення, команди тощо) обробляються за допомогою спеціальних об’єктів — обробників (handlers), які реагують лише на ті події, до яких вони прив'язані. Завдяки цьому досягається модульність, зручність у підтримці коду та можливість масштабування функціоналу бота.

Серед ключових можливостей модуля можна виокремити:

1) Асинхронне виконання запитів за допомогою async/await, що забезпечує ефективну обробку великої кількості користувачів одночасно без блокування основного потоку.
2) Фільтрацію вхідних повідомлень через зручну систему filters, що дозволяє обмежити обробку певними типами (текст, фото, команди, документи тощо).
3) Станову машину для діалогів (через ConversationHandler), яка спрощує побудову багатокрокових сценаріїв взаємодії з користувачем.
4) Гнучку структуру контексту (ContextTypes), яка дозволяє зберігати інформацію між кроками розмови або окремими оновленнями.

Використання telegram.ext значно спростило розробку інтерфейсу Telegram-бота в системі "Fridge Control", дозволивши зосередитись на логіці обробки даних, роботі з базою та взаємодії з користувачем, замість рутинної реалізації обробки HTTP-запитів до Telegram API.

2.2.1 Обробка команд

Бібліотека telegram.ext пропонує різноманітні типи обробників, кожен з яких має своє чітке призначення. Це дозволяє розділити логіку обробки

повідомлень за категоріями та легко масштабувати бот.

Серед основних використовуваних обробників:

1) CommandHandler — для обробки команд користувача, які починаються зі слешу (/start, /cancel). Вони ініціюють або завершують основні сценарії взаємодії з ботом.
2) MessageHandler — для обробки звичайних повідомлень:
   - filters.TEXT — текстові повідомлення, що використовуються для вводу назв продуктів, дат, часу тощо.
   - filters.PHOTO — дозволяє отримувати фотографії від користувача, які в подальшому можуть бути оброблені для виявлення вмісту холодильника.
3) ConversationHandler — застосовується для реалізації діалогів із кількома станами, таких як додавання/видалення продуктів або налаштування сповіщень.
4) CallbackQueryHandler — призначений для обробки натискань інлайн-кнопок, хоча в даній реалізації ще не використовується, але потенційно може бути інтегрований для покращення взаємодії через клавіатури та меню.

Ці обробники організовані у структурі, що дозволяє легко контролювати перебіг взаємодії, уникати конфліктів між командами та повідомленнями, а також реагувати на події лише в потрібних контекстах.

2.2.2 Реалізація розмов

Клас ConversationHandler є ключовим елементом для реалізації складних сценаріїв діалогу. Він дозволяє визначити послідовність станів, кожен з яких відповідає за певний етап взаємодії. Наприклад, при додаванні продукту користувач спочатку вводить назву, потім дату, а після цього підтверджує введені дані.

Кожен стан обробляється окремим функціональним блоком, що підвищує зручність відлагодження та повторного використання коду. При цьому

ConversationHandler також дозволяє обробляти переривання діалогу (/cancel), переходи до наступного стану, а також некоректне введення — з відповідними повідомленнями та підказками для користувача.

Завдяки підтримці context.user_data у кожному кроці зберігається введена інформація, що дозволяє повністю контролювати перебіг діалогу без необхідності постійного звернення до бази даних.

2.2 Вибір бази даних

Вибір системи управління базами даних (СУБД) є критично важливим етапом у проєктуванні будь-якої комп'ютерної системи, оскільки він визначає спосіб зберігання, доступу та управління даними. Для проєкту "Fridge Control", що передбачає зберігання інформації про продукти, їх терміни придатності, розташування та налаштування сповіщень, необхідно було обрати СУБД, яка б забезпечувала оптимальний баланс між простотою розгортання, продуктивністю для очікуваних обсягів даних та легкістю інтеграції з основною мовою розробки – Python.

Для обґрунтування вибору було проведено порівняльний аналіз трьох основних категорій баз даних, які могли б потенційно відповідати вимогам проєкту: SQLite, PostgreSQL/MySQL та MongoDB.

2.2.1 MongoDB

MongoDB є одним з найпопулярніших представників класу NoSQL баз даних, а саме документоорієнтованих СУБД. Вона зберігає дані у форматі BSON (Binary JSON), що забезпечує високу гнучкість схеми даних – це означає, що кожен запис (документ) може мати свою унікальну структуру, не прив'язану до жорстко визначеної таблиці [9]. Така гнучкість є надзвичайно корисною для швидкого прототипування або для додатків, де структура даних постійно змінюється або є надзвичайно різноманітною. MongoDB також розроблена з акцентом на горизонтальну масштабованість, дозволяючи легко розподіляти дані між кількома серверами для обробки величезних обсягів інформації. Однак, для "Fridge Control", де дані мають чітко визначену реляційну структуру (продукти,

їх атрибути, прив'язка до полиць, користувачі, сповіщення), гнучкість схеми MongoDB є скоріше недоліком, ніж перевагою, оскільки вона може ускладнити моделювання та забезпечення цілісності даних. Крім того, MongoDB, як і клієнт-серверні реляційні СУБД, вимагає розгортання та адміністрування окремого серверного процесу, що є надмірним для автономного додатка, а її переваги в масштабуванні для великих, неструктурованих даних не є актуальними для поточних вимог проєкту.

2.2.2 PostgreSQL / MySQL

PostgreSQL та MySQL представляють клас повнофункціональних клієнт-серверних реляційних систем управління базами даних (СУБД), які стали основою для безлічі корпоративних та веб-додатків по всьому світу. Їхня головна перевага полягає у високій масштабованості та продуктивності, які досягаються за рахунок оптимізованої внутрішньої архітектури та розвинених механізмів паралелізму. Це дозволяє їм ефективно обробляти величезні обсяги даних та численні одночасні запити, що є критично важливим для великих систем з високим навантаженням, таких як соціальні мережі, банківські системи або електронна комерція [10].

Ці СУБД пропонують розширений функціонал, який виходить за рамки базового зберігання даних. Він включає підтримку складних аналітичних запитів (SQL), збережених процедур для виконання логіки безпосередньо на стороні бази даних, тригерів для автоматичного виконання дій при зміні даних, а також можливості реплікації та кластеризації. Останні функції є незамінними для забезпечення високої доступності та відмовостійкості, дозволяючи системі продовжувати роботу навіть у разі збою одного з серверів. Завдяки багаторічному існуванню та активному розвитку, PostgreSQL та MySQL мають зрілу екосистему з величезною спільнотою розробників, безліччю офіційних та сторонніх інструментів для адміністрування, моніторингу та розробки, а також великою кількістю документації та навчальних матеріалів. Це робить їх надзвичайно надійним та підтримуваним вибором для масштабних проєктів, де

стабільність та можливість розширення є ключовими.

Однак, для цілей проєкту "Fridge Control", де основний акцент робиться на простоті та швидкості розробки для автономної системи, вимоги до розгортання та адміністрування PostgreSQL або MySQL є надмірними. Встановлення окремого серверного програмного забезпечення, його складна конфігурація та потреба у регулярному обслуговуванні (наприклад, резервне копіювання, оптимізація продуктивності, оновлення) створюють зайвий оверхед для невеликого, автономного застосунку. Більша частина їхнього розширеного функціоналу є зайвою для простої системи обліку продуктів та сповіщень. Крім того, їхнє споживання ресурсів (оперативної пам'яті та процесорного часу) значно вище порівняно з легкою SQLite, що не відповідає вимогам до легкої системи, яка може бути розгорнута на пристроях з обмеженими апаратними можливостями. Таким чином, хоча вони є потужними інструментами, їхні переваги не є критичними для "Fridge Control", а недоліки створюють невиправдану складність.

2.2.3 SQLite

SQLite — це унікальна реляційна система управління базами даних (СУБД), яка вирізняється своєю легкістю та вбудованою архітектурою. На відміну від традиційних клієнт-серверних баз даних, SQLite не функціонує як окремий серверний процес, а замість цього зберігає всю базу даних у єдиному файлі на диску [11]. Це означає, що програма взаємодіє з базою даних безпосередньо через бібліотеку, що вбудована в сам застосунок. Така архітектура значно спрощує розгортання та адміністрування, оскільки відсутня необхідність у встановленні, конфігурації чи підтримці окремого серверного програмного забезпечення. Для проєкту "Fridge Control", який розробляється як автономна система і не передбачає високих навантажень або багатокористувацької взаємодії у масштабах великого підприємства, безсерверна природа SQLite є значною перевагою, дозволяючи розробникам зосередитися виключно на бізнес-логіці програми, а не на інфраструктурних аспектах.

Однією з ключових переваг SQLite для Python-проєктів є те, що бібліотека sqlite3 є частиною стандартної бібліотеки Python. Це забезпечує безшовну інтеграцію з мовою, дозволяючи розробникам легко отримувати доступ до функцій бази даних та зручно працювати з нею без потреби у встановленні сторонніх драйверів чи адаптерів. Така нативна підтримка скорочує час на налаштування та потенційні проблеми сумісності. Крім того, файловий формат SQLite робить її надзвичайно портативною. Оскільки база даних зберігається як звичайний файл, її легко переміщувати між різними системами, копіювати для створення резервних копій або версіонування, а також розгортати на пристроях з обмеженими ресурсами. Це спрощує процес розробки, тестування та потенційне розширення системи на різні апаратні платформи.

Хоча SQLite не призначена для екстремально високих навантажень або великої кількості одночасних запитів, її продуктивності цілком достатньо для обсягів даних, характерних для домашнього використання системи "Fridge Control", де необхідно зберігати інформацію про кілька десятків або сотень продуктів, забезпечуючи оперативну взаємодію з даними та швидке виконання запитів.

2.3 Бібліотека sqlite3

Для зберігання та керування даними про продукти в холодильнику та розклади сповіщень було обрано вбудовану в Python бібліотеку sqlite3.

SQLite є легкою та самодостатньою реляційною базою даних, яка не потребує окремого сервера та ідеально підходить для локального зберігання даних застосунку. Бібліотека sqlite3 надає простий та ефективний API для виконання SQL-запитів, що використовується для створення таблиць, додавання, видалення та отримання даних про продукти та налаштування розкладів.

2.4 Бібліотека OpenCV

У ході розробки комп'ютерної системи "Fridge Control" для автоматизованого аналізу вмісту холодильника, ключовим компонентом є обробка та інтерпретація зображень. Для реалізації цієї функціональності було

використано бібліотеку OpenCV (Open Source Computer Vision Library), яка надає широкий спектр інструментів для вирішення завдань комп'ютерного зору.

OpenCV є кросплатформною бібліотекою з відкритим вихідним кодом, що розроблена для прискорення впровадження комп'ютерного зору в реальних застосунках. Вона підтримує мови програмування C++, Python, Java та інші, що забезпечує гнучкість у виборі середовища розробки. Для проєкту "Fridge Control", де основною мовою є Python, інтеграція з OpenCV є безшовною завдяки наявності потужних Python-біндінгів

2.3.1 Зчитування та представлення зображень

Функція cv2.imread() дозволяє легко зчитувати зображення з файлу у форматі, зручному для подальшої обробки. Зображення зчитується у вигляді багатовимірного масиву NumPy, що є стандартним форматом представлення зображень у Python і забезпечує ефективну роботу з піксельними даними. OpenCV забезпечує інтеграцію з NumPy, що дозволяє використовувати потужні інструменти для маніпулювання зображеннями.

2.3.2 Перетворення колірних просторів

Для спрощення певних етапів аналізу, код використовує функцію cv2.cvtColor() для перетворення зображення з колірного простору BGR (Blue, Green, Red) у Grayscale та HSV (Hue, Saturation, Value).

Перетворення у Grayscale зменшує розмірність даних, спрощуючи виявлення меж полиць за допомогою алгоритму Canny.

Перетворення у HSV дозволяє більш ефективно виділяти певні кольорові діапазони, що може бути корисним для сегментації продуктів харчування.

OpenCV надає великий набір функцій для перетворення між різними колірними просторами, що забезпечує гнучкість при обробці зображень для різних завдань.

2.3.3 Виявлення полиць

Для визначення меж полиць у холодильнику в даному проекті застосовується комбінація класичних алгоритмів комп’ютерного зору,

реалізованих за допомогою бібліотеки OpenCV.

Алгоритм виявлення контурів Canny, який реалізується функцією cv2.Canny() є одним з найпопулярніших методів виявлення меж завдяки його ефективності та надійності.

Перетворення Хафа для прямолінійних об’єктів, яке реалізується через функцію cv2.HoughLinesP() дозволяє знаходити на зображенні прямі лінії на основі результатів попереднього детектування меж. У нашому випадку це дає змогу ефективно виявити горизонтальні лінії, які відповідають фізичним полицям холодильника.

3 АРХІТЕКТУРА КОМП'ЮТЕРНОЇ СИСТЕМИ

Архітектура розробленого застосунку побудована за принципами трирівневої моделі (Рис.3), що передбачає розділення функціональних частин системи на логічні слої [12]. Такий підхід дозволяє досягти високої гнучкості, модульності та полегшує підтримку програмного забезпечення в подальшому. Архітектура складається з рівня представлення, рівня бізнес-логіки та рівня даних. Така структура полегшує модифікацію, тестування та розширення функціоналу в майбутньому.

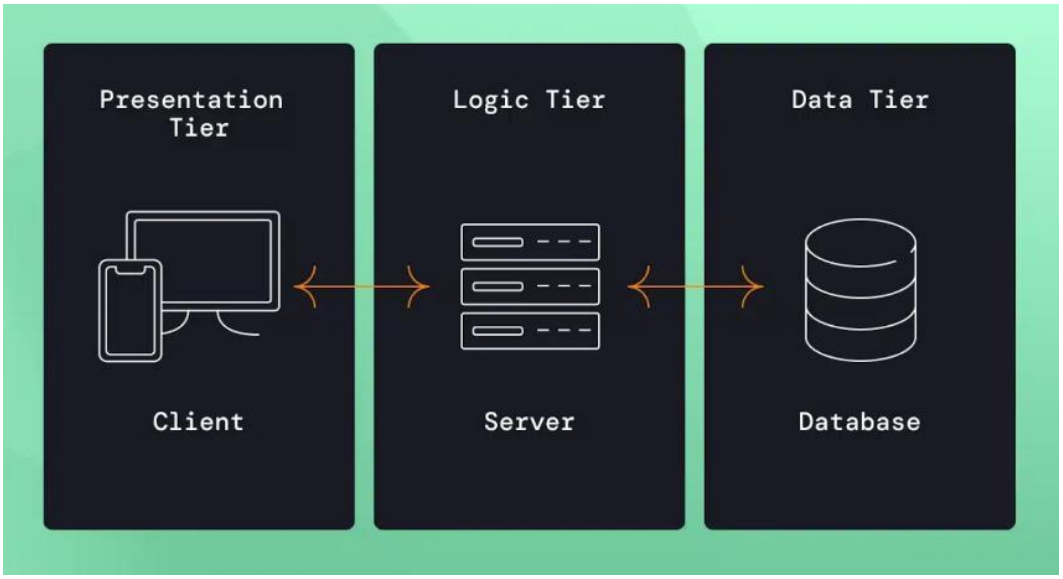

Рис. 3. Ілюстрація від vFunction

3.1 Загальна структура системи

Розроблена система моніторингу вмісту холодильника, що використовує аналіз зображень та інтегрований інтерфейс Telegram-бота, функціонує завдяки взаємодії кількох ключових компонентів. Основним елементом взаємодії з користувачем виступає Telegram-бот. Він відповідає за отримання від користувача різноманітних команд, а також зображень полиць холодильника, які є первинними даними для подальшого аналізу.

Через цього бота користувач вводить структуровану інформацію про продукти, що розміщуються в холодильнику, включаючи їхні назви та терміни придатності. Ці текстові дані передаються до системи збереження даних, де вони надійно реєструються та зберігаються для подальшого використання.

Окремо від процесу введення текстових даних, користувач має можливість надсилати фотографії полиць холодильника через той самий інтерфейс Telegram-бота. Отримані фотографії полиць передаються до модуля комп’ютерного зору. Цей модуль є ключовим для обробки візуальної інформації. Застосовуючи спеціальні алгоритми, модуль аналізує зображення з метою визначення поточного рівня заповненості кожної полиці холодильника продуктами. Результати цього аналізу є важливими для загального розуміння рівня заповненості холодильника.

Для забезпечення своєчасного інформування користувача про продукти, термін придатності яких добігає кінця, в системі передбачено механізм планування сповіщень. Цей компонент періодично звертається до системи збереження даних, де знаходиться інформація про назви продуктів та їхні терміни придатності. На основі цих даних система виявляє продукти, які потребують уваги користувача, та ініціює надсилання відповідних сповіщень через Telegram-бота.

Таким чином, архітектура системи "Fridge Control" організована так, що система збереження даних виступає централізованим сховищем структурованої інформації про харчові продукти та їхні терміни придатності, отриманої безпосередньо від користувача. Натомість, модуль комп'ютерного зору обробляє візуальні дані у вигляді фотографій для оцінки рівня заповненості полиць, не зберігаючи самі зображення, а результати його роботи оперативно відображаються користувачеві. Такий підхід дозволяє ефективно розподілити задачі обробки структурованих та неструктурованих даних, забезпечуючи при цьому необхідну функціональність моніторингу та інформування.

3.2 Три рівні системи

3.2.1 Рівень представлення (Presentation Layer)

Рівень представлення є безпосередньою точкою контакту між користувачем та системою, відповідаючи за забезпечення зручної взаємодії та відображення всієї необхідної інформації. У розробленій системі цю ключову

роль виконує Telegram-бот. Він надає інтуїтивно зрозумілий користувацький інтерфейс, через який користувач може вводити різноманітні команди та отримувати своєчасні відповіді від системи.

Рівень представлення відображає важливу текстову інформацію, включаючи деталі про збережені продукти, їхні критичні терміни придатності, а також наочні результати аналізу рівня заповненості полиць холодильника. Він забезпечує механізм прийому від користувача структурованих даних, таких як точна назва продукту та його термін придатності у встановленому форматі.

Також рівень представлення отримує від користувача неструктуровані дані у вигляді цінних фотографій полиць холодильника, які є основою для подальшого візуального аналізу. Він здійснює візуалізацію результатів складного аналізу заповненості полиць у вигляді чітких текстових звітів, доповнених інформативними емодзі для кращого сприйняття.

Крім того, рівень представлення надає користувачеві зручний механізм для персоналізації розкладу отримання автоматичних сповіщень про продукти, термін придатності яких наближається.

Для реалізації автоматичного інформування користувачів про продукти, термін придатності яких наближається, у розробленій системі використовується механізм запланованих завдань, наданий бібліотекою python-telegram-bot через об'єкт job_queue. Цей механізм дозволяє виконувати певні функції через задані проміжки часу або в конкретний час.

В контексті розробленого Telegram-бота, основним елементом системи запланованих сповіщень є асинхронна функція send_scheduled_notification.

```
async def send_scheduled_notification(context: ContextTypes.DEFAULT_TYPE):
    current_time = datetime.now().strftime('%H:%M')
    with get_db_connection() as conn:
        schedules = conn.execute('SELECT user_id FROM schedules WHERE notification_time = ?',
                                 (current_time,)).fetchall()
    for user_id in schedules:
        try:
            await check_products_for_user(context, user_id[0])
        except Exception as e:
            logging.error(f"Не вдалося надіслати сповіщення для {user_id[0]}: {e}")
```

Рис. 4. Реалізація функції планування сповіщень

Як видно з коду в основному файлі (Рис. 5), ця функція реєструється для періодичного виконання кожну хвилину за допомогою рядка job_queue.run_repeating(send_scheduled_notification, interval=60).

```
job_queue = application.job_queue
job_queue.run_repeating(send_scheduled_notification, interval=60)
```

Рис. 5. Перевірка сповіщень

Під час кожного виклику send_scheduled_notification отримує поточний час у форматі HH:MM за допомогою datetime.now().strftime('%H:%M'). Потім здійснюється підключення до бази даних SQLite за допомогою функції get_db_connection() і виконується SQL-запит до таблиці schedules. Цей запит вибирає user_id усіх користувачів, для яких встановлено час сповіщення, що співпадає з поточним часом.

Для кожного user_id, отриманого в результаті запиту, функція send_scheduled_notification викликає іншу асинхронну функцію – check_products_for_user, передаючи їй об'єкт context та ідентифікатор користувача.

Функція check_products_for_user відповідає за формування та надсилання сповіщення конкретному користувачеві. Вона також встановлює з'єднання з базою даних та виконує SQL-запит до таблиці products. Цей запит вибирає назви продуктів (product_name) та їхні терміни придатності (expiry_date) для заданого user_id, де термін придатності знаходиться в межах наступних семи днів (порівнюючи з поточною датою за допомогою date(expiry_date) <= date(?) AND date(expiry_date) >= date(?)). Отримані продукти сортуються за датою закінчення терміну придатності, як показано на рисунку нижче.

```python
async def check_products(update: Update, context: ContextTypes.DEFAULT_TYPE):
    today = datetime.now()
    seven_days = today + timedelta(days=7)
    with get_db_connection() as conn:
        products = conn.execute('''
            SELECT product_name, expiry_date
            FROM products
            WHERE user_id = ?
              AND date(expiry_date) <= date(?)
              AND date(expiry_date) >= date(?)
            ORDER BY date(expiry_date)
        ''', (update.effective_user.id, seven_days.strftime('%Y-%m-%d'), today.strftime('%Y-%m-%d'))).fetchall()

    if not products:
        await update.message.reply_text("Немає продуктів, у яких термін придатності спливає протягом 7 днів.")
        return

    message = "Продукти, термін придатності яких спливає протягом 7 днів:\n\n"
    for product in products:
        expiry = datetime.strptime(product[1], '%Y-%m-%d')
        days_left = (expiry - today).days
        icon = '🟢' if days_left >= 5 else '🔴' if days_left >= 3 else '🟠'
        message += f"{icon} {product[0]}: {expiry.strftime('%d.%m.%Y')}\n"

    await update.message.reply_text(message)
```

Рис. 6. Перевірка продуктів

Якщо для користувача знайдено продукти, термін придатності яких спливає, функція формує інформативне повідомлення, що включає назву кожного продукту, дату його закінчення та відповідний емодзі (зелений, помаранчевий або червоний) залежно від кількості днів, що залишилися. Якщо таких продуктів не знайдено, користувачеві надсилається повідомлення про відсутність продуктів, що потребують уваги. Нарешті, сформоване повідомлення надсилається користувачеві безпосередньо в чат Telegram за допомогою context.bot.send_message(chat_id=user_id, text=message).

Рівень представлення є критично важливим каналом зв'язку, що безпосередньо з'єднує користувача з усією функціональністю системи. Важливо підкреслити, що цей рівень не містить жодної бізнес-логіки; його єдина відповідальність полягає у коректній передачі запитів користувача на наступний рівень – рівень бізнес-логіки – та у відображенні отриманих від нього оброблених результатів у зручному для користувача форматі.

3.2.2 Рівень бізнес-логіки (Business Logic Layer)

Рівень бізнес-логіки є центральним елементом системи, відповідаючи за обробку даних, реалізацію всіх основних функціональних можливостей та координацію взаємодії між рівнем представлення та рівнем даних. У розробленій системі цей критично важливий рівень представлений основним скриптом

Python (main.py) та спеціалізованим модулем аналізу зображень (fridge_analysis.py). Основний скрипт (main.py) здійснює ретельну обробку всіх команд користувача, які надходять від інтерфейсу Telegram-бота, а також проводить обов'язкову валідацію всіх введених користувачем даних, включаючи перевірку відповідності формату (наприклад, коректність формату дати терміну придатності – Рис.4).

```python
async def expiry_date(update: Update, context: ContextTypes.DEFAULT_TYPE):
    try:
        date_str = update.message.text
        expiry_date = datetime.strptime(date_str, '%d.%m.%Y')
        with get_db_connection() as conn:
            conn.execute(
                'INSERT INTO products (user_id, product_name, expiry_date) VALUES (?, ?, ?)',
                (update.effective_user.id, context.user_data['product_name'], expiry_date.strftime('%Y-%m-%d'))
            )
        reply_markup = ReplyKeyboardMarkup(DEFAULT_KEYBOARD, resize_keyboard=True)
        await update.message.reply_text('Продукт успішно додано!', reply_markup=reply_markup)
        return ConversationHandler.END
    except ValueError:
        await update.message.reply_text('Неправильний формат дати. Будь ласка, використовуйте формат дд.мм.рррр:')
        return EXPIRY_DATE
```

Рис. 7. Приклад валідації введених даних

Він ініціює виклик відповідних функцій модуля рівня даних для виконання операцій зі збереження, отримання та оновлення інформації про продукти харчування та налаштований розклад сповіщень. Крім того, основний скрипт реалізує складну логіку планування сповіщень, яка включає періодичне визначення продуктів, термін придатності яких наближається до завершення, та автоматичне ініціювання процесу надсилання відповідних попереджувальних повідомлень користувачеві. Він відповідає за передачу отриманих від користувача зображень полиць холодильника до спеціалізованого модуля аналізу зображень для подальшої обробки та отримує оброблені результати аналізу рівня заповненості полиць від модуля аналізу зображень, здійснюючи їх форматування у зручний для сприйняття користувачем формат.

Модуль аналізу зображень (fridge_analysis.py) містить реалізацію передових алгоритмів комп'ютерного зору, які використовуються для детальної обробки отриманих фотографій полиць холодильника, забезпечуючи точне виявлення меж та розташування окремих полиць на аналізованому зображенні (Рис.5).

```
# Detect shelves
edges = cv2.Canny(image_gray, 50, 150, apertureSize=3)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=200, minLineLength=150, maxLineGap=15)

if lines is None:
    return "No shelves detected in the image."

shelf_candidates = []
for line in lines:
    x1, y1, x2, y2 = line[0]
    if abs(y2 - y1) < 10:
        shelf_candidates.append(y1)

# Filter and cluster shelves
min_distance = 100
filtered_shelves = []
for y in shelf_candidates:
    if all(abs(y - shelf) > min_distance for shelf in filtered_shelves):
        filtered_shelves.append(y)

if not filtered_shelves:
    return "No clear shelf structure detected."

filtered_shelves = np.array(filtered_shelves).reshape(-1, 1)
kmeans = KMeans(n_clusters=min(5, len(filtered_shelves)), random_state=0).fit(filtered_shelves)
shelf_levels = sorted(kmeans.cluster_centers_.flatten())
```

Рис. 8. Виявлення меж полиць

Він виконує сегментацію продуктів, розташованих на полицях, для їхнього подальшого аналізу та здійснює розрахунок відсоткового співвідношення площі, зайнятої продуктами, до загальної площі кожної окремої полиці, повертаючи оброблені результати аналізу, що включають відсоток заповненості для кожної виявленої полиці, назад до основного скрипту для подальшого відображення користувачеві.

Рівень бізнес-логіки є ключовим елементом, де виконується вся основна обробка даних та реалізується вся функціональність розробленого застосунку. Він залежить від рівня даних для отримання та збереження необхідної інформації, але при цьому залишається повністю незалежним від конкретної реалізації інтерфейсу користувача, представленого рівнем представлення.

3.2.3 Рівень даних (Data Layer)

Рівень даних відповідає за критично важливі функції зберігання, надійного отримання та ефективного керування всіма даними, що використовуються в системі. У розробленій системі цю ключову роль виконує локальна база даних SQLite (products.db). Вона забезпечує надійне зберігання структурованої інформації про всі харчові продукти, включаючи їхні точні назви, терміни придатності та унікальні ідентифікатори користувачів, які їх додали, а також

зберігає важливу інформацію про персоналізований розклад сповіщень для кожного окремого користувача, включаючи встановлений час сповіщення та відповідний ідентифікатор користувача.

Рівень даних надає чітко визначену структуру для виконання операцій додавання нових даних, оновлення існуючих записів, видалення застарілої інформації та отримання необхідних даних за запитами, що надходять з рівня бізнес-логіки.

Схема бази даних (Таблиця 1) включає таблицю products з полями id (унікальний ідентифікатор кожного продукту), user_id (ідентифікатор користувача, який додав продукт), product_name (назва продукту) та expiry_date (дата закінчення терміну придатності продукту), а також таблицю schedules з полями user_id (унікальний ідентифікатор користувача) та notification_time (час щоденного сповіщення для користувача).

Таблиця 1. Бази Даних

Сутність	Атрибут	Тип атрибуту
products	id	INTEGER PRIMARY KEY AUTOINCREMENT
	user_id	INTEGER
	product_name	TEXT
	expiry_date	DATE
schedules	user_id	INTEGER PRIMARY KEY TIME
	notification_time	

Для зберігання даних використовується легка для розуміння та надійна система керування реляційною базою даних SQLite. Ефективна взаємодія з базою даних SQLite забезпечується за допомогою вбудованого модуля sqlite3 мови програмування Python. Різноманітні операції з даними, включаючи їхнє

додавання, вибірку, оновлення та видалення, виконуються за допомогою мови структурованих запитів SQL.

```python
async def remove_schedule(update: Update, context: ContextTypes.DEFAULT_TYPE):
    with get_db_connection() as conn:
        conn.execute('DELETE FROM schedules WHERE user_id = ?', (update.effective_user.id,))
    reply_markup = ReplyKeyboardMarkup(DEFAULT_KEYBOARD, resize_keyboard=True)
    await update.message.reply_text('Розклад успішно видалено!', reply_markup=reply_markup)
```

Рис. 9. Приклад виконання SQL-запиту видалення

Рівень даних є критично важливим для забезпечення незалежності бізнес-логіки від конкретної реалізації зберігання даних. Це означає, що зміни у способі фізичного зберігання даних (наприклад, у випадку переходу на іншу систему керування базами даних) не повинні мати значного впливу на роботу рівня бізнес-логіки, за умови збереження узгодженого інтерфейсу взаємодії між цими двома рівнями.

Висновок

Застосування такої трирівневої архітектури забезпечує ефективне розділення відповідальностей між різними компонентами розробленої системи, що значно сприяє її кращій організації, спрощує процеси розробки та тестування, а також значно полегшує подальшу підтримку та масштабування. Кожен рівень виконує свою чітко визначену специфічну функцію, ефективно взаємодіючи з іншими рівнями через ретельно розроблені та зрозумілі інтерфейси.

4 РОЗРОБЛЕНА СИСТЕМА

4.1 Telegram-бот

4.1.1 Створення та налаштування бота (описание результата створення)

У результаті реалізації першого етапу проекту було створено Telegram-бота з назвою FridgeControlBot, доступного за посиланням t.me/FridgeControlBot.

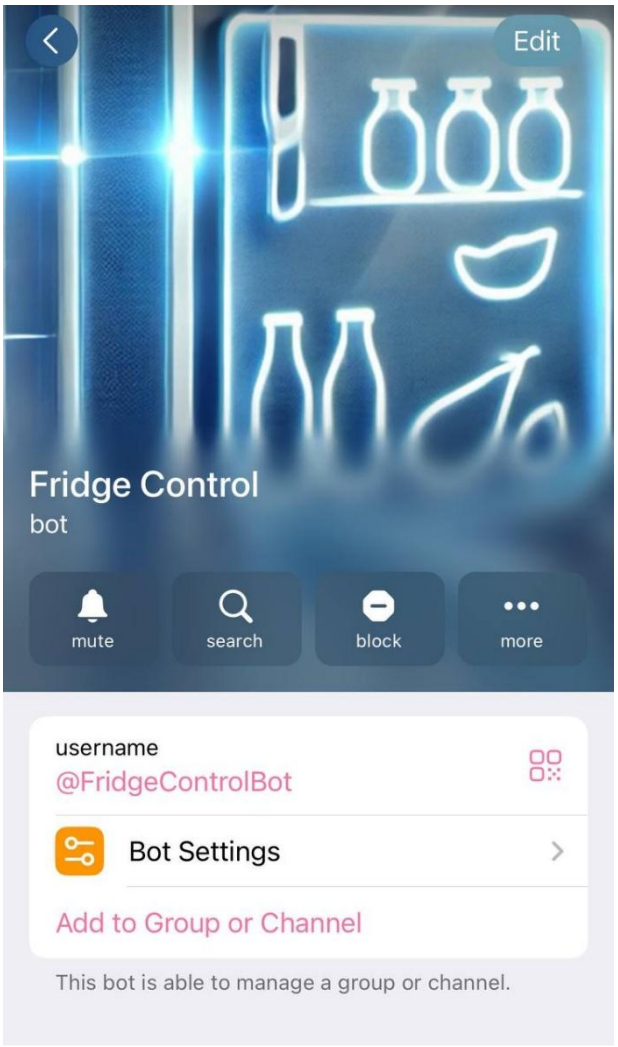

Рис. 10. Створений бот

Ім’я користувача та назва були обрані під час реєстрації за допомогою сервісу BotFather, а згенерований токен успішно інтегровано в програмну частину системи. Отриманий токен API було використано для ініціалізації бібліотеки python-telegram-bot, яка лягла в основу архітектури взаємодії з користувачем

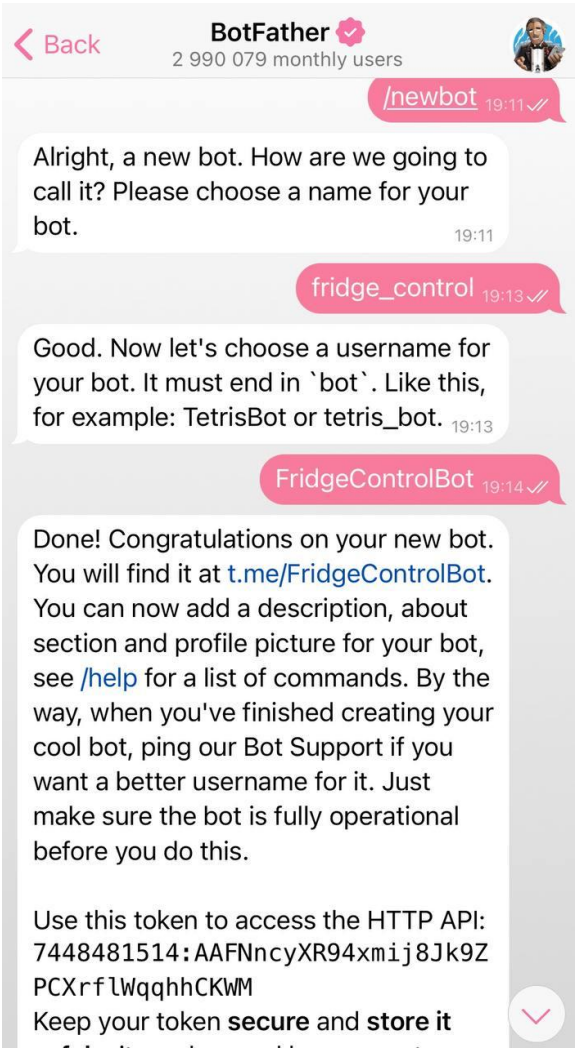

Рис. 11. Створення боту

4.1.2 Обробка команд

Розроблений Telegram-бот FridgeControlBot оснащений широким спектром функцій для ефективного управління продуктами та моніторингу холодильника. Реалізовано обробники команд (CommandHandler) та повідомлень (MessageHandler) для забезпечення безперебійної взаємодії з користувачем.

Після отримання команди /start користувачеві надсилається вітальне повідомлення. Це повідомлення містить опис основних можливостей бота та пропонує скористатися основним меню (Рис. 12)

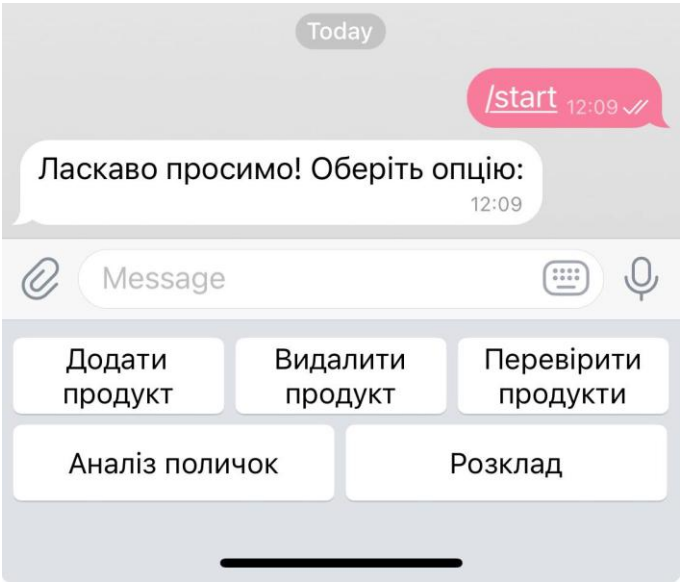

Рис. 12. Запуск бота

Під час взаємодії з ботом, користувач може ініціювати різні операції, які вимагають послідовних кроків або введення додаткової інформації. Наприклад, це може бути процес додавання нового продукту до списку вмісту холодильника, видалення існуючої позиції, або ж налаштування розкладу сповіщень про термін придатності продуктів. Кожна з цих операцій, будучи багатоетапною, передбачає, що бот очікує на певну відповідь від користувача або подальші інструкції.

Однак, у динамічному середовищі взаємодії, може виникнути ситуація, коли користувач вирішить перервати поточну операцію. Це може статися з різних причин: можливо, користувач передумав додавати продукт, випадково ініціював неправильну дію, або просто бажає повернутися до головного меню без завершення розпочатого процесу. Для таких випадків у системі "Fridge Control" був розроблений інтуїтивно зрозумілий механізм скасування, реалізований за допомогою команди /cancel.

Ця команда слугує універсальним інструментом для переривання будь-якої активної багатоетапної операції. Як тільки бот отримує команду /cancel від користувача, що ілюструється на Рис. 8, він негайно припиняє виконання поточного сценарію. У відповідь бот надсилає чітке та лаконічне повідомлення, яке інформує користувача про успішне скасування поточної операції. Це

повідомлення слугує підтвердженням дії користувача та усуває будь-яку невизначеність. Після підтвердження скасування, система автоматично повертає користувача до основного меню бота. Це забезпечує плавний перехід та дозволяє користувачеві швидко і безперешкодно обрати наступну дію або завершити взаємодію з ботом. Такий механізм підвищує зручність використання та робить навігацію по функціоналу бота більш гнучкою та контрольованою для користувача.

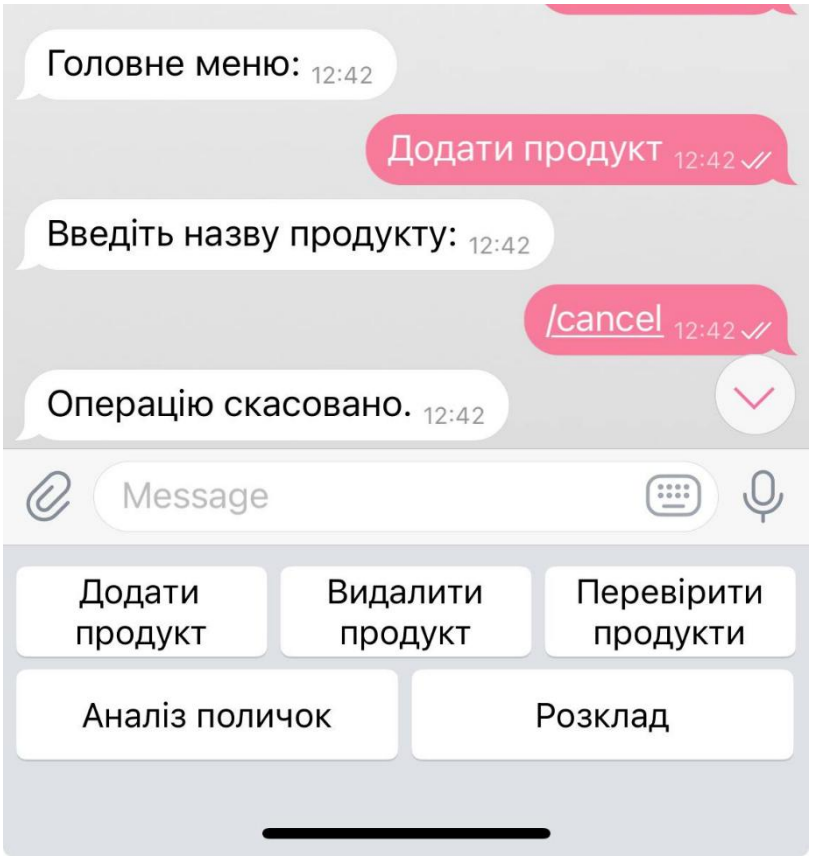

Рис. 13. Приклад скасування операції

Однією з центральних функцій, яка забезпечує зручність та ефективність системи "Fridge Control", є можливість легкого додавання нових продуктів до її внутрішньої бази даних. Цей ключовий процес ініціюється користувачем за допомогою інтуїтивно зрозумілого натискання кнопки "Додати продукт" у головному меню бота. Після активації цієї опції, система розпочинає послідовну взаємодію з користувачем, спрямовану на збір усієї необхідної інформації.

Бот не просто запитує всю необхідну інформацію одразу, що могло б бути незручним і схильним до помилок. Натомість, він організовує керований діалог,

який крок за кроком збирає всі потрібні дані. Для успішного внесення нового продукту до інвентарю бота, системі потрібні дві ключові одиниці інформації від користувача: назва продукту та термін його придатності.

Для реалізації такого структурованого діалогу система "Fridge Control Bot" ефективно використовує патерн ConversationHandler з бібліотеки python-telegram-bot. Цей потужний інструмент дозволяє організувати складні багатоетапні розмови з користувачем, керуючи станами діалогу та забезпечуючи логічний потік інформації.

При натисканні кнопки "Додати продукт", бот переходить у спеціальний режим введення даних, де ConversationHandler бере на себе управління. Першим кроком у цьому діалозі є чіткий запит на введення назви продукту. Після того, як цей запит надсилається користувачеві, внутрішній стан розмови бота змінюється на PRODUCT_NAME. Це означає, що бот тепер цілеспрямовано очікує на текстове повідомлення від користувача, яке буде інтерпретоване саме як назва нового продукту.

Щойно назва продукту успішно отримана та оброблена, ConversationHandler автоматично переводить діалог на наступний, логічно послідовний етап. Бот надсилає користувачеві новий, конкретний запит — цього разу щодо терміну придатності продукту. Відповідно, внутрішній стан розмови змінюється на EXPIRY_DATE. Протягом цього етапу бот очікує на введення дати у певному форматі, забезпечуючи її коректне розпізнавання та валідацію. Такий поетапний підхід, як ілюструє Рис. 9, робить процес введення даних максимально простим та інтуїтивним для користувача і значно зменшує ймовірність помилок. Кожен запит чітко формулюється, направляючи користувача та забезпечуючи збір коректної та повної інформації.

Важливою особливістю цього процесу є те, що вся введена користувачем інформація – як назва продукту, так і його термін придатності – тимчасово зберігається у спеціальному об'єкті context.user_data. Цей об'єкт слугує тимчасовим сховищем даних, пов'язаних виключно з поточною сесією взаємодії

користувача. Він дозволяє боту "пам'ятати" попередні введення під час багатоетапної розмови, забезпечуючи її безперервність та логічність.

Лише після успішного введення та валідації дати терміну придатності – тобто коли всі необхідні дані зібрано та підтверджено їхню коректність – інформація про новий продукт зберігається в основній базі даних системи "Fridge Control". Після цього фінального кроку діалог з користувачем завершується. Бот може або автоматично повернути користувача до головного меню, або надати чітке підтвердження про успішне додавання продукту до інвентарю. Такий структурований та керований підхід до збору даних не тільки забезпечує точність введення інформації, але й мінімізує потенційні помилки з боку користувача, роблячи процес додавання продуктів інтуїтивно зрозумілим.

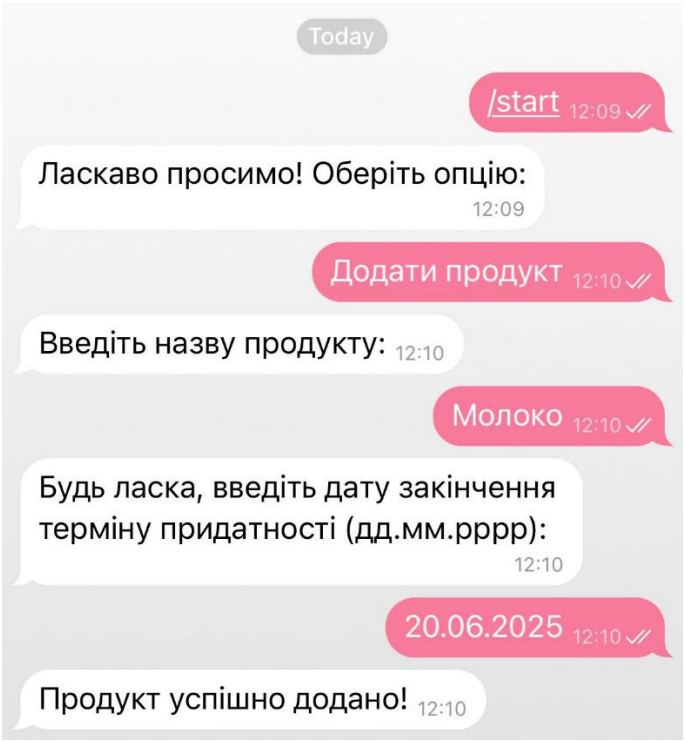

Рис. 14. Приклад додавання продукту

Можливість ефективного управління вмістом холодильника передбачає не тільки додавання, а й видалення продуктів, які вже були спожиті або зіпсувалися. Ця функція є однією з ключових для підтримки актуальності інвентарю в системі "Fridge Control".

Процес видалення ініціюється користувачем шляхом натискання кнопки "Видалити продукт" у головному меню. Після активації цієї опції, бот

звертається до внутрішньої бази даних. Його завдання — отримати повний список усіх продуктів, які були раніше додані саме цим конкретним користувачем.

Отриманий перелік інформації система відображає користувачеві у вигляді чіткого, нумерованого списку. Кожен елемент цього списку містить не тільки назву продукту, але й його термін придатності, що допомагає користувачеві легко ідентифікувати потрібну позицію. Після відображення списку, бот пропонує користувачеві ввести номер продукту, який він бажає видалити.

Введений користувачем номер продукту обробляється у спеціальному стані діалогу, названому REMOVE_PRODUCT. Цей стан гарантує, що система очікує саме числовий ідентифікатор зі списку. Після успішного отримання та валідації номера, бот виконує запит до бази даних, видаляючи відповідний запис про продукт. Цей інтуїтивно зрозумілий та послідовний підхід значно спрощує процес управління інвентарем, дозволяючи користувачеві швидко й безпомилково оновлювати вміст свого холодильника.

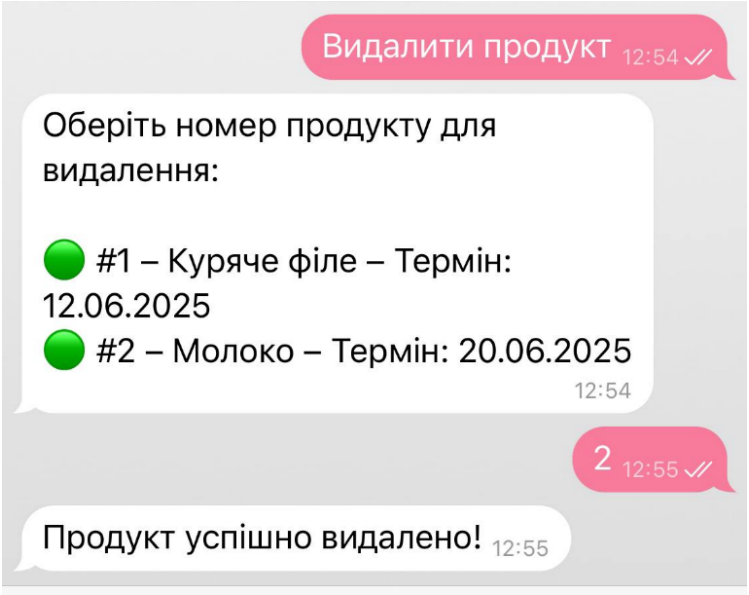

Рис. 15. Приклад видалення продукту

Особливо важливою функцією для ефективного управління вмістом холодильника є можливість швидкого отримання інформації про продукти, термін придатності яких спливає найближчим часом. Ця функціональність реалізована в "Fridge Control" через кнопку "Перевірити продукти". Натиснувши

її, користувач ініціює запит, що дозволяє уникнути псування продуктів та оптимізувати їх споживання.

Після активації цієї кнопки, бот звертається до внутрішньої бази даних. Його мета — отримати список усіх продуктів, що належать поточному користувачеві, термін придатності яких закінчується протягом наступних семи днів. Цей часовий проміжок обрано для надання користувачеві достатнього часу для реагування.

Отриманий список продуктів відображається користувачеві у вигляді інформативного та візуально привабливого повідомлення. Для підвищення наочності та інтуїтивності, як показано на Рис. 11, кожен продукт у списку супроводжується відповідним емоджі. Колір цього емоджі прямо вказує на кількість днів, що залишилися до закінчення терміну придатності:

Зелений емоджі (🟢) використовується для продуктів, термін придатності яких спливає більше ніж через 5 днів. Це сигналізує про відсутність термінової потреби у використанні продукту.

Помаранчевий емоджі (🟠) позначає продукти, термін придатності яких складає від 3 до 4 днів. Це є попередженням, що продукт варто використати найближчим часом.

Червоний емоджі (🔴) використовується для продуктів, термін придатності яких спливає менше ніж за 3 дні. Це є чітким закликом до негайного споживання продукту, щоб уникнути його псування.

Така візуальна індикація значно спрощує сприйняття інформації та дозволяє користувачеві швидко оцінити стан своїх запасів, приймаючи своєчасні рішення.

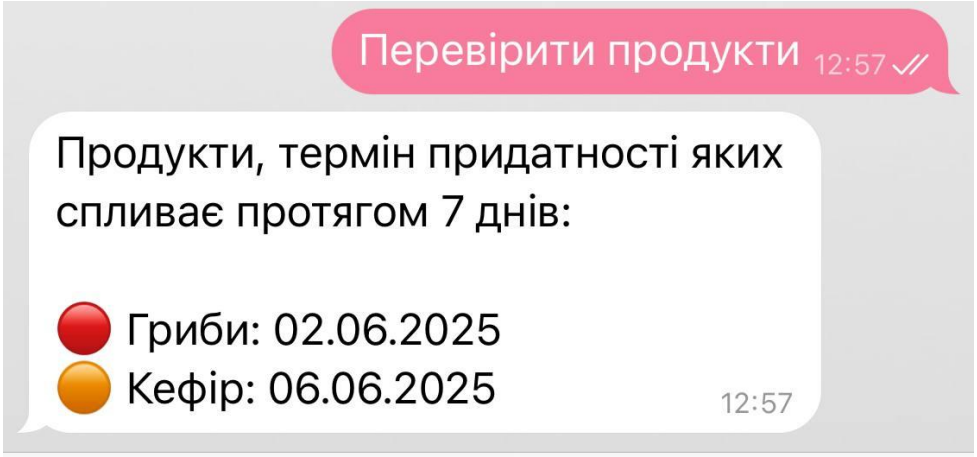

Рис. 16. Перевірка продуктів

У разі якщо після запиту до бази даних виявиться, що немає жодних продуктів, термін придатності яких закінчується протягом наступних семи днів, бот "Fridge Control" надасть відповідне інформативне повідомлення. Цей сценарій передбачено для уникнення будь-якої невизначеності для користувача.

Як ілюструє Рис. 12, у такій ситуації бот відправить повідомлення, яке чітко інформуватиме користувача про відсутність продуктів, що потребують термінової уваги. Це запевняє користувача, що його холодильник під контролем і що всі продукти в безпеці на найближчий тиждень.

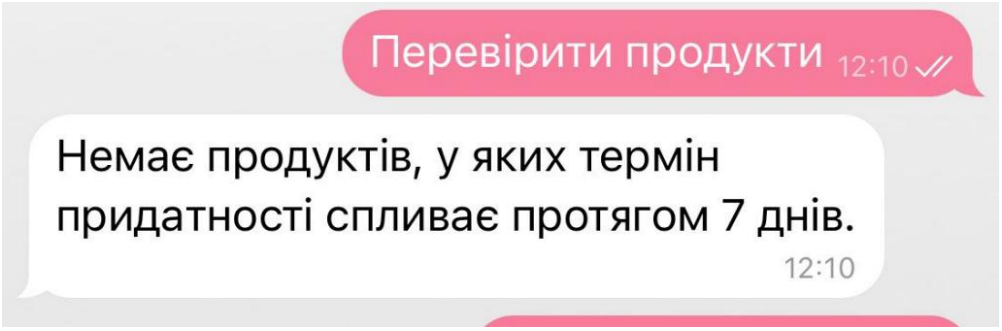

Рис. 17. Перевірка продуктів

Для аналізу рівня заповненості полиць холодильника користувач натискає кнопку "Аналіз поличок". У відповідь бот просить користувача надіслати фотографію полиць холодильника для подальшої обробки.

Після того як користувач надсилає фотографію полиць холодильника, спрацьовує обробник handle_photo. Ця функція завантажує отримане зображення на сервер, тимчасово зберігає його та передає шлях до файлу у зовнішній модуль fridge_analysis для проведення аналізу. Модуль fridge_analysis обробляє

зображення та визначає рівень заповненості кожної виявленої полиці у відсотках.

Після завершення аналізу, отримані результати – відсоток заповненості для кожної полиці – надсилаються користувачеві у вигляді текстового повідомлення. Для покращення візуалізації та швидкого сприйняття інформації, повідомлення доповнюється відповідними емоджі, які візуально відображають рівень заповненості:

Зелений емоджі (🟢) вказує на те, що на полиці багато вільного місця.

Жовтий емоджі (🟡) сигналізує про середній рівень заповненості.

Червоний емоджі (🔴) означає, що на полиці мало вільного місця або вона майже повністю заповнена.

Така візуальна індикація дозволяє користувачеві швидко оцінити ситуацію та прийняти рішення про необхідність поповнення запасів або перерозподілу продуктів. Після того як аналіз завершено, і результати відправлені користувачеві, тимчасово збережена фотографія видаляється з сервера, забезпечуючи конфіденційність даних та ефективне використання ресурсів.

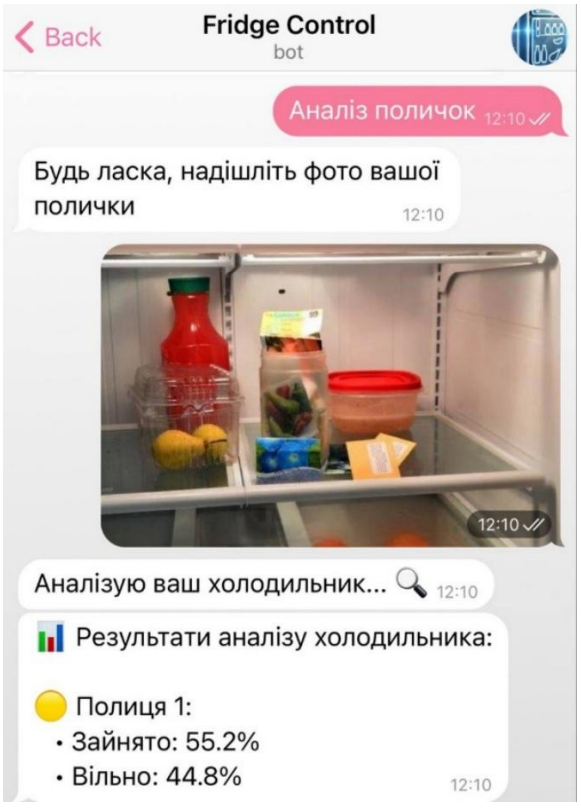

Рис.18. Приклад аналізу поличок

Однією з найзручніших функцій у "Fridge Control Bot" є можливість керування розкладом щоденних сповіщень. Це дозволяє користувачам отримувати регулярні нагадування про стан їхніх продуктів, забезпечуючи своєчасне споживання та зменшуючи харчові відходи. Доступ до цієї функції здійснюється натисканням кнопки "Розклад" у головному меню бота.

Після натискання цієї кнопки, бот відображає спеціалізоване меню налаштування розкладу, відоме як SCHEDULE_KEYBOARD. Це меню розроблено для забезпечення легкого та інтуїтивно зрозумілого управління сповіщеннями і містить три основні кнопки:

"Встановити розклад": для визначення нового часу отримання сповіщень.

"Видалити розклад": для скасування раніше встановлених нагадувань.

"Назад": для повернення до основного меню.

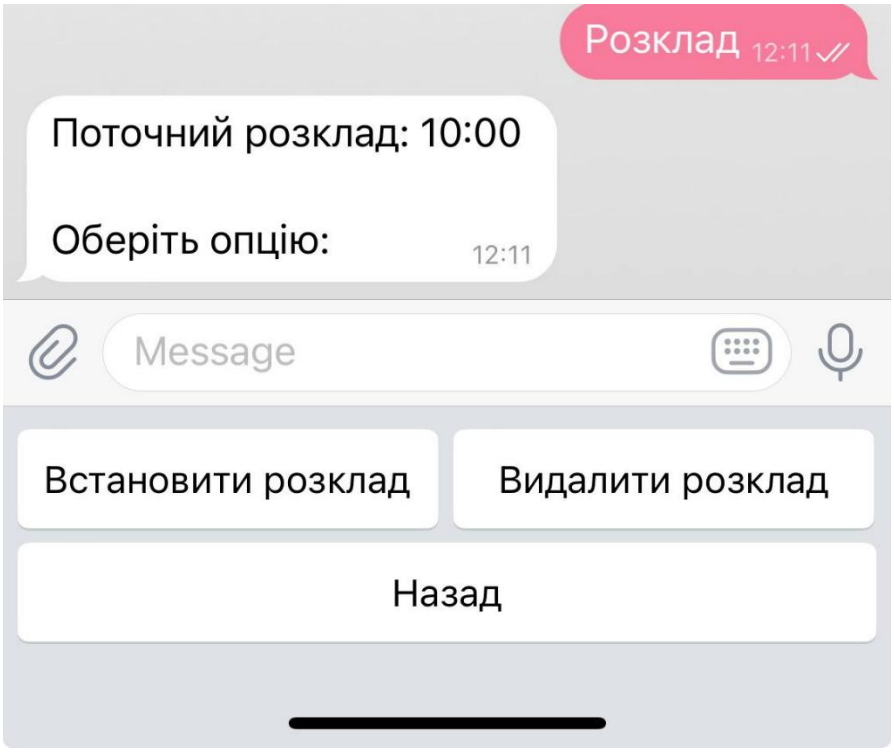

Рис. 19. Меню команд для розкладу

Натискання кнопки "Встановити розклад" ініціює процес визначення нового часу для щоденних сповіщень. Бот негайно надсилає користувачеві запит на введення бажаного часу у форматі HH:MM (наприклад, "18:30" для 18 години 30 хвилин). Після отримання цього часу, система перевіряє його коректність, а потім зберігає у базі даних у спеціальній таблиці schedules, асоціюючи з

ідентифікатором поточного користувача. Це забезпечує персоналізовані нагадування для кожного користувача.

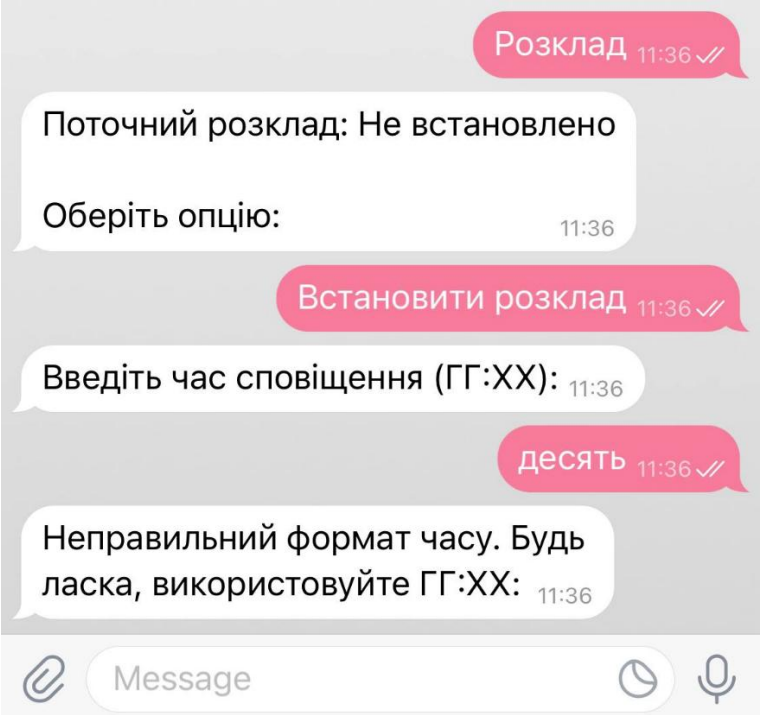

Рис. 20. Приклад некоректно введеного формата

Якщо користувач бажає скасувати щоденні сповіщення, він може скористатися кнопкою "Видалити розклад". Після її натискання, бот виконує запит до таблиці schedules у базі даних, знаходить та видаляє відповідний запис для поточного користувача. Успішне видалення підтверджується повідомленням, що надсилається користувачеві, інформуючи його про скасування розкладу сповіщень.

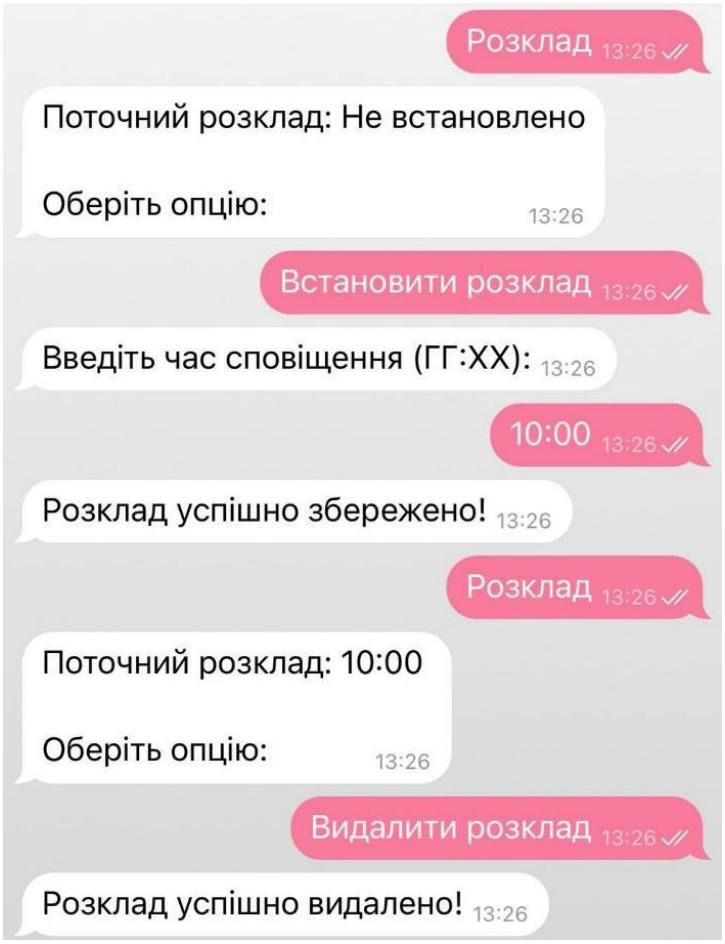

Рис. 21. Приклад видалення розкладу

Кнопка "Назад" є стандартним елементом навігації, що дозволяє користувачеві в будь-який момент повернутися з меню налаштування розкладу до основного меню бота. Це забезпечує гнучкість та зручність у навігації, дозволяючи користувачеві легко перемикатися між різними функціями системи "Fridge Control".

ВИСНОВКИ

У ході виконання цієї дипломної роботи було успішно розроблено та повноцінно реалізовано програмну систему "Fridge Control", що є комплексним рішенням для моніторінгу вмісту холодильника. Проєкт слугує прикладом практичного застосування сучасних підходів до розробки програмного забезпечення та демонструє можливість побудови корисних рішень у сфері "розумного дому" та побутової автоматизації.

Було проведено глибокий аналіз та обґрунтування вибору технологічної бази. Цей процес включав ретельне порівняння трьох провідних мов програмування: Java, Python та Go. Кожна з них була оцінена за низкою критеріїв, таких як продуктивність, складність синтаксису, наявність бібліотек та підтримка спільноти. У результаті цього всебічного аналізу було обґрунтовано, що Python є оптимальним вибором для даного проєкту. Його ключові переваги, такі як висока швидкість розробки, що є критично важливою для ітеративного створення можливостей; виняткова читабельність коду, яка сприяє легкій підтримці та масштабованості; багата екосистема бібліотек (зокрема, python-telegram-bot, що став основою для взаємодії з користувачем), а також велика та активна спільнота, яка забезпечує широку підтримку та доступ до готових рішень, виявилися вирішальними факторами для створення ефективного, гнучкого та зручного рішення.

В рамках першого етапу реалізації, який став фундаментом для системи, було розроблено та запущено інтерфейс взаємодії з користувачем (на базі Telegram-бота). Це дозволило забезпечити прямий та інтуїтивно зрозумілий доступ до усіх можливостей "Fridge Control".

Були створені ключові можливості, що забезпечують ефективне управління вмістом холодильника:

Додавання нових продуктів: Цей процес було організовано у формі поетапного діалогу, що дозволяє користувачеві зручно вводити назву продукту

та його термін придатності. Такий підхід мінімізує помилки та спрощує ведення обліку.

Видалення наявних продуктів: Для зручності користувача система відображає поточний інвентар у вигляді нумерованого списку, надаючи можливість легко вибрати та видалити продукт, який було спожито або зіпсовано.

Перевірка термінів придатності продуктів: Користувачі можуть швидко отримати інформацію про продукти, термін придатності яких спливає найближчим часом. Ця можливість включає візуальну індикацію за допомогою емоджі (🟢 для більше 5 днів, 🟠 для 3-4 днів, 🔴 для менше 3 днів), що дозволяє миттєво оцінити терміновість використання того чи іншого продукту.

Аналіз заповненості полиць холодильника: Користувач може завантажити фотографію полиць холодильника, а система автоматично аналізує зображення та надає відсоткову інформацію про заповненість кожної полиці, також візуалізовану кольоровими емоджі для швидкого розуміння вільного простору.

Управління розкладом щоденних сповіщень: Користувачі отримали можливість персоналізувати свій досвід, встановлюючи або видаляючи зручний час для отримання регулярних нагадувань про стан продуктів. Це сприяє своєчасному використанню запасів та мінімізує харчові відходи.

Усі зазначені можливості були створені з особливим акцентом на зручність використання, надійність та потенціал для подальшого розширення. Розроблена система "Fridge Control" яскраво демонструє величезний потенціал застосування сучасних програмних рішень для автоматизації рутинних завдань у побуті та покращення якості життя, надаючи користувачам ефективний та інтуїтивно зрозумілий інструмент для управління запасами продуктів у холодильнику.

Ця дипломна робота стала міцною і функціональною основою для подальшого розвитку проєкту. Вона відкриває широкі перспективи для майбутнього розширення системи, включаючи можливість інтеграції з іншими "розумними" пристроями в будинку (наприклад, сканерами штрих-кодів),

вдосконалення алгоритмів аналізу зображень для більш точного розпізнавання продуктів, додавання можливостей прогнозування закінчення продуктів на основі історичних даних споживання, а також розширення аналітичних можливостей системи для надання користувачам більш детальної статистики та рекомендацій щодо управління запасами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Samsung Family Hub™. Samsung US | Samsung USA. URL: Samsung Family Hub™ | Samsung US | Samsung USA

2. Java. IBM Think. URL: https://www.ibm.com/think/topics/java. White B. Internet of Things (IoT): Principles and Paradigms. – Wiley, 2021.

3. Багатопотоковість в Java: суть, «плюси» та часті пастки. URL: https://javarush.com/ua/groups/posts/uk.1992.bagatopotokovstjh-v-java-sutjh-pljusi-ta-chast-pastki

4. Go Programming Language. URL: https://go.dev/

5. Go by Example: Goroutines. URL: https://butuzov.github.io/gobyexample/goroutines.html

6. Matthes E. Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming. No Starch Press, 2019.

7. OpenCV. Офіційний сайт OpenCV. URL: https://opencv.org/

8. python-telegram-bot. Офіційна документація python-telegram-bot. URL: https://python-telegram-bot.org/

9. MongoDB. (б.д.). Why Use MongoDB? URL: https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb?msockid=14a5a794a1d46ece1062b555a0996f5c

10. Guru99. (б.д.). Що таке PostgreSQL? Вступ, переваги та недоліки. URL: https://www.guru99.com/uk/introduction-postgresql.html

11. SQLite. Офіційний сайт SQLite. URL: https://www.sqlite.org/

12. 3-Tier Application. VFunction Blog. URL: https://vfunction.com/blog/3-tier-application/