Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ВЕБ-ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ. ЧАСТИНА 3. СЕРВЕРНА ЧАСТИНА»

Виконав: студент 2 курсу, групи 1КІ-22м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Рижков А.К.
(прізвище та ініціали)
Керівник к.т.н., доцент каф. ОТ
Войцеховська О.В.
(прізвище та ініціали)
«07» 12 2023 р.
Опонент к.т.н., доцент каф. ПЗ
Майданюк В.П.
(прізвище та ініціали)
«08» 12 2023 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н проф. Азаров О.Д.
«08» 12 2023 р.

Вінниця ВНТУ — 2023 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень другий (магістерський)

Галузь знань — 12 — Інформаційні технології

Спеціальність — 123-«Комп'ютерна інженерія»

Освітньо — професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ, д.т.н, проф.

Азаров О.Д.

"26" 09 2023року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Рижкову Андрію Костянтиновичу

1 Тема роботи: Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина

керівник роботи Войцеховська Олена Валеріївна, к.т.н., доцент,

затверджені наказом вищого навчального закладу від «26» 09 2023 року № 247

2 Строк подання студентом роботи 9 грудня

3 Вихідні дані до роботи: контент для наповнення бази даних веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту, технології реалізації серверної частини

4 Зміст текстової частини: аналіз технологій для проектування серверної частини веб застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту, метод інтеграції веб-застосунку зі штучним інтелектом, розробка серверної частини веб-застосунку для організації дієт та тренувань, проектування бази даних веб-застосунку для організації дієт та тренувань

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових кресл структурна схема серверної частини веб-застосунку, діаграма класів компон інтеграції зі штучним інтелектом, UML-діаграма діяльності модулю інтеграції застосунку зі штучним інтелектом, UML-діаграма діяльності серверної частини застосунку зі штучним інтелектом

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Войцеховська О. В., к.т.н., доцент каф. ОТ		
5	Небава М.І. к.е.н, професор каф ЕПВМ		

7 Дата видачі завдання 19.09.2023

8 Календарний план виконання КМКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз завдання	26.09.23	27.09.23	вик.
2	Аналіз використання багаторівневої архітектури у серверній частині	28.09.23	05.10.23	вик.
3	Аналіз технологій для із базою даних та авторизацією	05.10.23	20.10.23	вик.
4	Аналіз методів інтеграції штучного інтелекту із веб-застосунком	20.10.23	30.10.23	вик.
5	Покращення методу інтеграції зі штучним інтелектом. Розробка модулю інтеграції	31.10.23	12.11.23	вик.
6	Проектування бази даних серверного додатку	12.11.23	15.11.23	вик.
7	Розробка та програмна реалізація рівня доступу до даних серверної частини	15.11.23	20.11.23	вик.
8	Розробка та програмна реалізація рівня бізнес-логіки серверної частини	20.11.23	25.11.23	вик.
9	Перевірка працездатності серверної частини веб-додатку	25.11.23	26.11.23	вик
10	Оформлення пояснювальної записки та ілюстративного матеріалу	26.11.23	05.12.23	вик.
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	05.12.23	08.12.23	вик.

Студент **Рижков А. К.**

Керівник роботи **Войцеховська О**

АНОТАЦІЯ

УДК 004.4

Рижков А.К. Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина. Комплексна магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна Інженерія, Вінниця: ВНТУ, 2023 — 116 с. На укр. Мові. Біблогр. 21 назв; рис: 27;

У магістерській кваліфікаційній роботі проведено аналіз технологій, які використовуються для проектування та розробки серверної частини веб-застосунку для організації дієт та систем тренувань. Розглянуто технології для зв'язку з базою даних, авторизації, інтеграції зі штучним інтелектом.

Вдосконалено метод інтеграції серверної частини веб-застосунку зі штучним інтелектом, що дало змогу покращити User Experience та збільшити швидкодію обробки запитів штучним інтелектом.

Розроблено серверну частину веб-застосунку для організації тренувань та дієти мовою C#, з урахуванням принципів об'єктно-орієнтовного програмування. Під час розробки було використано підхід для розподілення різних функціональних можливостей між різними модулями серверної частини.

Описано проектування бази даних для веб-застосунку для організації тренувань та дієти. Налаштований зв'язок серверної частини веб-додатку з базою даних для отримання та збереження інформації.

Ключові слова: веб-застосунок, .NET, C#, Entity Framework, Access Token, ChatGpt, OpenAI, Web-API.

ABSTRACT

УДК 004.4

Ryzhkov A.K. Web application for organizing training and diet with integration of artificial intelligence. Part 3. Server-side. Comprehensive master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — ... p. In Ukrainian. Bibliography: 21 titles; figures: 27;

The master's thesis conducts an analysis of technologies used in the design and development of the server-side of a web application for organizing diets and training systems. Technologies for database communication, authorization, and integration with artificial intelligence are considered.

Improved the method of integrating the server-side of the web application with artificial intelligence, which improved User Experience and increased the processing speed of requests by artificial intelligence.

Developed the server-side of the web application for organizing training and diets using C#, considering the principles of object-oriented programming. A modular approach was used in the development to distribute various functional capabilities among different modules of the server-side.

Described the design of the database for the web application for organizing training and diets. Configured the connection of the server-side of the web application to the database for obtaining and storing information.

Keywords: web application, .NET, C#, Entity Framework, Access Token, ChatGPT, OpenAI, Web API.

ЗМІСТ

ВСТУП

Сучасний розвиток інформаційних технологій відкриває перед користувачами можливості для розробки інноваційних рішень, які можуть поліпшити якість життя та розвинути галузі, що стикаються з комп'ютерними науками. Окрім цього зараз відбувається дуже швидкий розвиток штучного інтелекту. Зараз чат-боти можуть виконувати функцію співрозмовника, вчителя, тренера та навіть дієтологу.

Більшість спортивних сайтів можуть запропонувати користувачам лише статті на тему харчування, тренувань та розрахунок денної норми калорій, в залежності від даних користувача. Але жоден із сайтів не може побудувати повноцінну дієту, виходячи із інформації, що надав користувач.

Отже, комплексна магістерська кваліфікаційна робота, присвячена розробці серверного додатку для веб-застосунку для організації тренувань та дієти з використанням штучного інтелекту, є актуальною та перспективною. Розробка програмного забезпечення мовою C# є актуальною завдяки високій продуктивності та доступності мови, а також великій кількості засобів, що допомагають створювати ефективні та надійні рішення. У даному веб-застосунку використовуються сучасні підходи до створення серверного додатку, такі як технологія ASP.NET Identity для авторизації користувачів та Entity Framework для взаємодії з базою даних. Це дозволяє створити стійке і функціональне серверне середовище для веб-додатку.

Також, використовуючи інтегрований механізм штучного інтелекту, веб-застосунок здатний аналізувати дані та будувати індивідуалізовані дієти для користувачів. Це відкриває перед користувачами нові перспективи в області здоров'я та харчування, а також сприяє здоровому способу життя.

Метою комплексної магістерської кваліфікаційної роботи є розширення функціональних можливостей серверного додатку, завдяки інтеграції з штучним інтелектом, що дасть можливість покращити User Experience (UX) та збільшити швидкодію обробки запитів штучним інтелектом.

Задачі дослідження:

— провести аналіз сучасних технологій для проектування серверної частини веб-додатку;

— вдосконалити метод інтеграції веб-застосунку із штучним інтелектом;

— розробити архітектуру модуля інтеграції веб-застосунку зі штучним інтелектом;

— провести аналіз вимог до запиту, що надсилається на серверну частину;

— виконати програмну реалізацію серверної частини веб-застосунку для організації тренувань та дієти;

— реалізувати авторизацію та оцінити безпеку використаного методу авторизації на серверній частині;

— спроектувати базу даних для веб-застосунку та провести її оптимізацію;

— виконати тестування модуля інтеграції зі штучним інтелектом.

Об’єкт дослідження — об’єктом дослідження є процес інтеграції веб-застосунку з штучним інтелектом.

Предмет дослідження — сучасні методи передавання інформації між серверною та клієнтською частинами веб-застосунку та методи інтеграції зі штучним інтелектом на серверній стороні веб-застосунку.

Новизна одержаних результатів — вдосконалено метод інтеграції серверної частини веб-застосунку із штучним інтелектом, в якому, на відміну від існуючих, реалізовано побудову та надсилання запитів, створених на основі користувацької інформації, до штучного інтелекту з використанням його API, що дозволило розширити функціональні можливості та збільшити швидкодію обробки запитів штучним інтелектом.

Практичним значенням одержаних результатів є результати дослідження, що можуть бути використані для покращення UX при роботі зі штучним інтелектом та отримання користувачем дієти та програми тренувань з урахуванням введених його антропометричних та фізіологічних даних.

Апробація результатів роботи — зроблено доповіді на LI науково-технічній конференції підрозділів ВНТУ та на Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» [1, 2].

Публікації:

— аналіз архітектури розподіленого слабкозв'язного серверного додатку інформаційної системи / Войцеховська О. В., Рижков А. К. //Матеріали LI наукової-технічної конференції підрозділів ВНТУ, 2022 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/allfitki/all-fitki-2022/paper/view/15103/12731 [1];

— аналіз методів авторизації при проектуванні серверної частини веб-додатку / Войцеховська О. В., Городецька О.С., Рижков А. К.// Матеріали міжнародної науково-практичної інтернет-конференції "Електронні інформаційні ресурси: створення, використання, доступ", 2023 р. Режим доступу: https://drive.google.com/file/d/1oVmxS3W_sEQPjes9S9AzWDaJxDxi6I0X/view [2];

— подано заявку на державну реєстрацію авторського права на твір: Комп'ютерна програма «Модуль інтеграції вебзастосунку для організації тренувань та дієти зі штучним інтелектом».

1 АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ВЕБ ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Аналіз використання багаторівневої архітектури при проектуванні серверної частини веб-застосунку організації тренувань та дієти

Багаторівнева архітектура є видом архітектурного підходу, який дозволяє розробникам розподілити програмний код серверної частини на різні рівні та шари. Кожен рівень виконує конкретні функції та відповідає за певні аспекти функціонування додатку [3].

Розглянемо кожен рівень багаторівневої архітектури серверної частини більш детально.

Рівень даних (Data Layer) є базовим рівнем багаторівневої архітектури і відповідає за збереження та управління даними. Цей рівень часто використовує технології доступу до даних, такі як Entity Framework або ADO.NET. Він містить такі елементи, як модель даних, база даних та операції збереження та вибірки даних.

Модель даних — спроектована модель даних, яка визначає структуру даних, їхні зв'язки та атрибути. В мові C#, Entity Framework дозволяє визначити класи, які відповідають сутностям бази даних.

Вибір бази даних — важливий крок, який потрібно виконувати з урахуванням потреб додатку. Як база даних може бути використана реляційна база даних, NoSQL сховище або інше джерело даних.

Рівень даних включає методи для збереження, оновлення, видалення та вибірки даних з бази даних.

Рівень доступу до даних (Data Access Layer) функціонує як посередник між рівнем даних та рівнем бізнес логіки. Він надає інтерфейс для звернення до даних та виконання операцій з ними. Зазвичай рівень включає такі складові:

— Data Access Objects (DAOs) — це класи або компоненти, які взаємодіють з рівнем даних та виконують операції збереження та вибірки даних;

— Object-Relational Mapping (ORM) — допомагає відобразити об'єктну модель на реляційну базу даних та автоматично створює SQL-запити і виконує операції з даними;

— підключення та менеджмент підключень — містить логіку управління підключеннями до бази даних, включаючи конфігурацію та безпеку.

Рівень бізнес логіки (Business Logic Layer) відповідає за обробку даних та визначення бізнес-правил та логіки додатку. Даний рівень має такі основні елементи:

— бізнес-об'єкти — це класи, які представляють сутності та містять логіку для виконання операцій над ними;

— сервіси — надають інтерфейс для виклику функціональності бізнес-логіки, координують дії бізнес-об'єктів та виконують послідовні операції;

— бізнес-правила –рівень, який містить правила, що визначають, які операції дозволені, а які — заборонені.

Останній рівень — це рівень представлення (Presentation Layer), який відповідає за відображення даних та взаємодію з користувачем. У мові C#, цей рівень включає такі складові:

— графічний інтерфейс (GUI) — визначає зовнішній вигляд додатку, з яким буде взаємодіяти користувач;

— контролери — приймають запити від користувача та взаємодіють з рівнем бізнес логіки для отримання та відображення даних;

— валідація та обробка помилок — включає в себе валідацію введених даних та обробку помилок, щоб забезпечити коректну роботу додатку [4].

1.2 Аналіз технологій доступу до даних у ASP.Net

1.2.1 Огляд технології Entity Framework Core

Технологія Entity Framework Core (EF Core), розроблена корпорацією Microsoft, на сьогоднішній день вважається однією з найбільш потужних технологій в області доступу до даних в реляційних базах даних. EF Core є фреймворком для

роботи з інформацією з бази даних як з об'єктами платформи .Net, що дозволяє розробникам спростити взаємодію з даними та базами даних.

Однією з ключових характеристик EF Core є її високий рівень абстракції від фізичної структури баз даних. Організація даних здійснюється на рівні об'єктів, а не таблиць, індексів чи ключів. Ця абстракція дозволяє розробникам працювати на більш високому рівні і незалежно від конкретної бази даних. На фізичному рівні, EF Core використовує Structured Query Language (SQL) для взаємодії з даними, але для розробників, концептуальний рівень бази даних перетворюється в набір об'єктів [5].

EF Core підтримує різні системи управління базами даних завдяки провайдерам. Microsoft надає вбудовані провайдери для популярних систем управління базами даних (СУБД), таких як PostgreSQL, SQLite і MS SQL Server. Це робить EF Core універсальним і забезпечує можливість вибору потрібної системи відповідно до вимог проекту.

Однією з найбільших переваг EF Core є його уніфікований Application Programming Interface (API) для доступу до даних. Необхідність заміни СУБД вимагає лише налаштування підключення до нового провайдера. Решта коду, який відповідає за взаємодію з базою даних, залишається без змін. Це дає розробникам можливість зосередитися на бізнес-логіці та не хвилюватися про деталі доступу до даних [6].

Entity Framework Core може бути інтегрований на різних платформах стеку .NET Core, включаючи десктопні додатки, веб-додатки і кросплатформені рішення. Завдяки цій кросплатформеності, EF Core може бути використаний на операційних системах Windows, Linux та macOS.

Центральною концепцією Entity Framework є сутність, яка являє собою набір даних, пов'язаних з певним об'єктом. Замість роботи з таблицями і SQL-запитами, розробники можуть працювати з об'єктами та їхніми відносинами, що робить код більш читабельним і об'єктно-орієнтованим.

При роботі з EF Core, розробникам необхідно спочатку описати модель даних, яку необхідно відобразити в базі даних. Ця модель включає в себе сутності та їхні відносини. Потім створюється контекст даних, який є точкою входу до бази даних і

містить колекцію DbSet для кожної сутності. Кожен DbSet відповідає таблиці в базі даних і надає можливість виконувати запити до даних.

Для встановлення підключення до бази даних, розробники вказують рядок підключення, який містить інформацію про сервер, назву бази даних, тип СУБД і інші параметри. Ця інформація використовується для налаштування підключення до бази даних.

Якщо виникне необхідність змінити тип СУБД, EF Core робить цей процес досить простим. Розробникам потрібно лише змінити рядок підключення на новий, і EF Core автоматично створить нову базу даних та відобразить модель даних на нову структуру.

1.2.2 Огляд технології Microsoft ActiveX Data Object для .NET

Технологія ActiveX Data Object для .NET (ADO.NET), розроблена Microsoft, є незамінним інструментом для програмістів та розробників, які працюють з базами даних та взаємодіють із різними джерелами даних. Ця технологія, була створена декілька десятиліть тому, продовжує залишатися актуальною та надійною, завдяки набору класів та бібліотек, які вона пропонує.

ADO.NET розроблено з врахуванням різноманітних потреб в розробці програмного забезпечення. Ця технологія надає інструменти для створення клієнтських баз даних, обробки даних, взаємодії з сховищами даних та багатьох інших завдань. Її різноманітність та гнучкість дозволяють програмістам створювати рішення, які відповідають конкретним вимогам їх проектів [7].

Однією з головних переваг ADO.NET є висока швидкість взаємодії з базою даних. Це особливо важливо в сучасному світі, де великі обсяги даних є нормою. ADO.NET надає дуже швидкий доступ до даних, що робить його незамінним інструментом для великих систем з великою кількістю складних даних. При виконанні запитів, що охоплюють декілька сутностей, технологія ADO.NET зазвичай видає результати набагато швидше, ніж інші інструменти, включаючи популярний Entity Framework Core.

Проте варто враховувати два важливі недоліки ADO.NET. По-перше, ця технологія не підтримує кросплатформеність, що означає, що обмеження щодо операційної системи та системних вимог сервера може бути проблематичним для деяких проектів. По-друге, ADO.NET вимагає вручну вставляти SQL-запити у код C# та ініціювати їх виконання. Це може бути надзвичайно складним завданням для розробників, які працюють з об'єктами мови C#. З цього погляду, Entity Framework Core виходить вперед, оскільки надає можливість створювати запити за допомогою мови LINQ, яка існує у мові C# за замовчуванням. Entity Framework автоматично транслює LINQ-запити у SQL-запити, що полегшує роботу з даними та забезпечує більшу читабельність коду [8].

У підсумку, ADO.NET залишається важливим інструментом для багатьох проектів, завдяки своїй швидкості та різноманітності функцій. Проте важливо зважувати переваги та недоліки при виборі ORM технології, що буде використовуватись в проекті.

Отже, EF Core зручніше та гнучкіше ніж ADO.Net, в той час як ADO.Net перемагає у швидкодії, що важливо для великих громіздких проектів. Якщо ж проект невеликий, такий як невелика серверна частина, краще використовувати Entity Framework.

1.3 Аналіз технологій авторизації та автентифікації у веб-додатках

Фундаментальними аспектами будь-якого веб-застосунка є автентифікація та авторизація, які визначають безпеку та управління доступом.

Автентифікація та авторизація відіграють важливу роль в забезпеченні безпеки веб-застосунку. Автентифікація гарантує, що користувач, який намагається отримати доступ до системи, є тим, за кого він себе видає. Авторизація визначає, які дії та ресурси користувач може виконувати після успішної автентифікації [9].

Ці два процеси важливі для досягнення таких ключових цілей:

— конфіденційність — запобігання несанкціонованого доступу до конфіденційних даних та ресурсів;

— цілісність — забезпечення, що дані не будуть змінені або псуватися несанкціонованою особою;

— доступність — відповідні користувачі матимуть доступ до ресурсів, коли це потрібно.

Основна автентифікація на основі пароля — це найпоширеніший метод. Користувач вводить ім'я користувача та пароль. Система перевіряє ці дані відносно збережених облікових записів. Цей метод простий, але вимагає належного управління паролями, такого як шифрування та політики паролів.

Окрім цього існує двофакторна автентифікація (2FA). Двофакторна автентифікація — це вдосконалення основного методу. Користувач, окрім пароля, повинен надати другий фактор ідентифікації, такий як одноразовий код, отриманий через SMS або мобільний додаток, або фізичний пристрій (наприклад, ключ безпеки). Це ускладнює життя потенційним зловмисникам і забезпечує більшу безпеку.

Біометрична автентифікація використовує унікальні біологічні риси користувача для ідентифікації, такі як відбиток пальця, розпізнавання обличчя або сканування відбитка ока. Цей метод забезпечує точність та зручність для користувачів, але вимагає відповідної інфраструктури та обладнання.

Для забезпечення авторизації часто використовують ролі користувачів. Ролі допомагають класифікувати користувачів залежно від їхніх функцій та обов'язків у веб-застосунку. Наприклад, адміністратор має більше прав та можливостей, ніж звичайний користувач. Ролі регулюють, які операції доступні користувачеві.

Дозволи визначають конкретні дії, які користувач може виконувати в межах своєї ролі. Наприклад, адміністратор може мати право на створення та видалення користувачів, тоді як звичайні користувачі не мають таких прав.

Забезпечення належної автентифікації та авторизації важливо для запобігання несанкціонованому доступу до даних та функціональностей веб-додатка. Використання сучасних методів та технологій допомагає забезпечити безпеку та захищеність користувачів та їх даних [10].

1.3.1 Авторизація з використанням ASP.NET Identity

Авторизація — це критично важливий елемент будь-якого веб-застосунку, який регулює доступ користувачів до ресурсів і функцій системи. Технологія ASP.NET Identity є потужним інструментом для реалізації системи авторизації на платформі .NET.

Користувачі спілкуються один з одним в реальному часі через соціальні мережі, такі як Facebook, Twitter та інші соціальні веб-сайти. Розробники бажають, щоб користувачі могли увійти за допомогою своїх ідентифікаторів, щоб мати покращений UX на їхніх веб-сайтах. Сучасна система авторизації повинна дозволяти перенаправляти вхід на постачальників аутентифікації, таких як Facebook, Twitter та інші.

З розвитком веб-розробки змінювалися і патерни веб-розробки. Юніт-тестування коду додатку стало ключовою проблемою для розробників додатків. У 2008 році ASP.NET додав новий фреймворк, заснований на паттерні Модель-Вид-Контролер (MVC), частково для допомоги розробникам у побудові ASP.NET-додатків, які можна тестувати юніт-тестами. Розробники, які хотіли протестувати свою логіку додатку, також хотіли це робити з системою членства.

З урахуванням цих змін у веб-розробці була розроблена ASP.NET Identity.

Система ASP.NET Identity може бути використана з усіма фреймворками ASP.NET, такими як ASP.NET MVC, Web Forms, Web Pages, Web API та SignalR. ASP.NET Identity може бути використана при створенні веб-, мобільних, магазинних або гібридних додатків.

ASP.NET Identity — це фреймворк для керування аутентифікацією та авторизацією в ASP.NET додатках. ASP.NET Identity забезпечує легкість додавання профільних даних про користувача. Програміст має контроль над інформацією про користувача та профілю. Наприклад, легко можна дозволити системі зберігати дати народження, введені користувачами при реєстрації облікового запису в додатку.

За замовчуванням система ASP.NET Identity зберігає всю інформацію про користувача в базі даних. ASP.NET Identity використовує Entity Framework Code First для реалізації процесу зберігання [11].

ASP.NET Identity надає готові інструменти для створення, оновлення, видалення та перевірки облікових записів користувачів. Фреймворк забезпечує безпеку за допомогою хешування паролів, валідації та обробки сесій.

ASP.NET Identity може працювати з різними джерелами даних, такими як SQL Server, Entity Framework, або інша довільна база даних. Він також підтримує зв'язок багато-до-багатьох між ролями користувачів та дозволами.

1.3.2 Авторизація на HTTP-куках

HTTP-куки (Cookies, веб-куки, браузерні куки) — це невеликий фрагмент даних, який сервер відправляє веб-браузеру користувача. Браузер може зберегти куки і відправити їх назад на той самий сервер із подальшими запитами. Зазвичай HTTP-куки використовуються для визначення того, чи дані запити походять від того ж браузера, що дозволяє, наприклад, утримувати користувача в системі під час авторизації. Вони зберігають інформацію про стан для протоколу HTTP.

Куки в основному використовуються для трьох цілей:

— управління сеансом, тобто авторизація користувачів, кошики покупок, рахунки в іграх або будь-яка інша інформація, яку сервер повинен пам'ятати;

— персоналізація, зокрема налаштування користувача, теми та інші налаштування;

— відстеження, тобто фіксація та аналіз поведінки користувачів.

Раніше куки також використовувалися для загального зберігання даних на клієнтському боці. Це було логічним, коли куки були єдиною можливістю зберігання даних на клієнтському боці. Але зараз рекомендується використовувати сучасні інтерфейси зберігання даних на клієнтському боці. Куки відправляються з кожним запитом, що може погіршити продуктивність, особливо на мобільних підключеннях

до Інтернету. Сучасні API для зберігання даних на клієнтському боці включають Web Storage API (localStorage і sessionStorage) і IndexedDB [12].

Авторизація на куках є традиційним методом для зберігання ідентифікаційної інформації на стороні клієнта. Коли користувач успішно автентифікується, сервер генерує і підписує куку, яка зберігає інформацію про користувача, таку як ідентифікатор та роль. Ця кука надсилається на клієнтську сторону і зберігається в браузері.

Після отримання HTTP-запиту сервер може відправити один або декілька заголовків Set-Cookie разом із відповіддю. Зазвичай браузер зберігає ці куки і включає їх до запитів, які він відправляє на той самий сервер, включаючи їх в заголовок HTTP Cookie. Можна вказати дату закінчення чи період часу, після якого куки не повинні відправлятися. Також можна встановити додаткові обмеження для конкретного домену і шляху, щоб обмежити, де куки відправляються.

HTTP-заголовок відповіді Set-Cookie відправляє куки від сервера до користувача (user agent). Приклад встановлення куки представлено в лістингу 1.1.

Лістинг 1.1 — Приклад заголовка set-cookie

```
Set-Cookie: cookie_name=cookie_value
```

У цьому прикладі "cookie_name" — це ім'я куки, а "cookie_value" — значення. Коли браузер отримує цей заголовок у відповіді сервера, він зберігає цю куку і включає її до запитів, які він відправляє на той самий сервер [13].

Переваги авторизації на куках включають простоту використання та підтримку сесій. Однак вона має певні обмеження. Куки зазвичай зберігаються на стороні клієнта, що робить їх вразливими до атак на перехоплення. Також вони не підходять для розподілених систем або мікросервісних архітектур.

1.3.3 Авторизація з використанням JWT-Токенів

JSON Web Token (JWT-токен) — це відкритий стандарт, описаний в RFC 7519, який визначає компактний і самодостатній спосіб безпечної передачі інформації між сторонами у формі об'єкта JSON. Ця інформація може бути перевіреною та

довіреною, оскільки вона містить цифровий підпис. JWT може бути підписаним з використанням секретного ключа (за алгоритмом HMAC) або пари публічного/приватного ключа з використанням RSA або ECDSA алгоритмів.

JWT-токени є сучасним методом авторизації, який використовується для безпеки та передачі інформації між сторонами. Вони являють собою структурований набір даних, закодований у вигляді токену, який підписується і може містити інформацію про користувача та його права.

Інформацію, що міститься в об'єкті JSON, можна перевірити і впевнитися в її автентичності завдяки цифровому підпису. JWT можуть бути зашифрованими для забезпечення конфіденційності між сторонами. Також JWT можуть бути підписаними, а не зашифрованими. Вони видаються провайдером ідентифікації і називаються JSON Web Signatures (JWS). [14].

Коли токени підписані за допомогою пари публічного/приватного ключа, підпис також підтверджує, що тільки та сторона, яка має приватний ключ, могла підписати токен.

Перед використанням отриманого JWT, слід належним чином перевірити його підпис. Успішно перевірений токен означає лише те, що інформація, яка міститься в токені, не була змінена іншими особами. Це не означає, що інші не могли бачити вміст, який зберігається у відкритому вигляді. Тому не варто зберігати важливу інформацію у JWT. Краще вживати інших заходів для того, щоб запобігти перехопленню JWT, таких як відправка JWT тільки через HTTPS, дотримання найкращих практик та використання лише надійних та оновлених бібліотек.

JWT-токени дозволяють сторонам підтверджувати автентичність та авторизацію без необхідності зберігати стан на сервері або на стороні клієнта. Вони є ідеальним вибором для розподілених систем і мікросервісів, оскільки можуть бути передані між різними сервісами.

JWT можуть бути використані у різних сценаріях, наприклад при авторизації. Після успішного входу користувача за його обліковими даними повертається ідентифікаційний токен (ID token). Згідно специфікацій OpenID Connect (OIDC), ID token завжди є JWT.

Після успішного входу користувача додаток може запитувати доступ до маршрутів, служб або ресурсів (наприклад, API) від імені цього користувача. Для цього в кожному запиті він повинен передавати токен доступу, який також може бути у формі JWT. Системи одноразового входу (Single Sign-on, SSO) широко використовують JWT через невеликий обсяг даних у цьому форматі та можливість легкого використання на різних доменах.

JWT є відмінним способом безпечної передачі інформації між сторонами, оскільки їх можна підписувати, що означає, що відправники — це ті, за кого вони видають себе. Крім того, структура JWT дозволяє перевірити, чи не було змінено вміст токену.

Використання JWT токенів має деякі переваги, у порівнянні з іншими, наприклад:

— компактність, тобто JSON менше розміркованій в порівнянні з XML, тому після кодування JWT виявляється меншим, ніж SAML-токен;

— безпечність — JWT може бути симетрично підписаним загальним секретним ключем з використанням алгоритму HMAC;

— поширеність — JSON є поширеними у більшості мов програмування, оскільки він відображаються безпосередньо в об'єкти;

— легкість обробки — JWT легше обробляти на пристроях користувачів, особливо на мобільних [15].

1.3.4 Обґрунтування вибору технології для авторизації в веб-застосунку

Вибір між авторизацією на куках та JWT-токенами залежить від конкретних потреб та вимог веб-додатка. Куки підходять для простих додатків з внутрішніми сесіями та більшим контролем зі стороні сервера. JWT-токени підходять для розподілених систем, мобільних додатків та мікросервісів, де важлива масштабованість та безпека.

Важливо враховувати, що обидва методи можуть використовуватися в комплексі для реалізації різних аспектів авторизації в одному веб-додатку. Ваш вибір повинен базуватися на конкретних вимогах та цілях проекту.

ASP.NET Identity — потужний інструмент, який спрощує реалізацію авторизації у веб-додатках. Розуміння роботи цієї технології та вибір оптимального методу авторизації є важливими завданнями при розробці безпечних та функціональних додатків.

Сесійні куки — це станові елементи. Вони містять дані, які сервер відправляє веб-браузеру для тимчасового використання. Дані аутентифікації всередині куки зберігаються як на клієнтському, так і на серверному боці. Сервер веде облік активних сесій в базі даних, тоді як браузер зберігає ідентифікатор активної сесії. Коли запит надсилається на сервер, ідентифікатор сесії використовується для пошуку інформації, такої як ролі користувачів чи дозволи для аутентифікації, щоб перевірити, чи сесія все ще дійсна.

Куки використовують одну сесію на всіх піддоменах: для цього вони приймають аргумент "Domain", в якому вказується ім'я домену, для якого кука є дійсною. Встановлення імені домену, наприклад, yoursite.com, дозволяє використовувати одну й ту саму сесію на домені та всіх його піддоменах.

Куки зменшують можливість маніпуляцій клієнтським JavaScript — можна обмежити доступ з боку клієнта, встановивши прапорець HttpOnly. Це зменшує ймовірність атак типу між-сайтового сценарію (XSS) на вебдодаток, оскільки більшість атак XSS включають використання зловмисного коду JavaScript.

Куки потребують невеликої кількості пам'яті. Для зберігання простого ідентифікатора користувача куки використовують лише кілька кілобайт. Залежно від того, яка інформація зберігається в куці, кожний запит передає мінімальну кількість байтів.

Куки керуються браузером. Це відбувається автоматично, тому програмісту не потрібно про це турбуватися.

Проте, в свою чергу куки трохи гірші з точки зору безпеки, адже атаки Cross-Site Request Forgery (CSRF) можливі лише при обробці сесій на основі куків. Атрибут

SameSite дозволяє вам визначити, чи слід відправляти куки до додатків третіх сторін зі строгими або легкими налаштуваннями.

Також існують певні проблеми з масштабуванням. Оскільки сесії пов'язані з певним сервером, виникають проблеми з масштабуванням додатку. У випадку балансування навантаження, якщо зареєстрований користувач перенаправляється на новий сервер, дані існуючої сесії втрачаються. Для запобігання цьому сесії повинні зберігатися в спільній базі даних або кеші. Це збільшує складність кожної взаємодії.

Також, вони не підходять для аутентифікації API, оскільки вони надають одноразові ресурси для аутентифікованих кінцевих користувачів і не потребують відстеження сеансів користувача. Куки не працюють ідеально в цьому випадку, оскільки вони відстежують і перевіряють активні сесії. Токени, з іншого боку, надають аутентифікацію за допомогою унікального ідентифікатора при кожному запиті до кінцевих точок API.

Токени, не мають стану за своєю природою, що означає, що серверу не потрібно вести облік токена. Кожен токен самостійний, містить інформацію, необхідну для перевірки та ідентифікації на сервері.

Їх перевагами є гнучкість та легкість використання. Їх самостійність допомагає отримати необхідну інформацію для верифікації без обліку в базі даних. Це робить JWT більш придатними для використання в API, оскільки сервер API не повинен вести облік сеансів користувача.

Також є можливості для крос-платформеності, завдяки тому, що токени не мають стану, вони можуть бути безперешкодно впроваджені на мобільних платформах і застосунках Інтернету речей (IoT), особливо порівняно з куками.

Ще однією перевагою є те, що токени можна зберігати різними способами в браузерах або фронтенд-додатках.

Проте є і мінуси використання, наприклад процес скасування, тому що JWT не можна скасувати. Навіть якщо JWT потрапляє до рук зловмисників, він залишається дійсним до закінчення строку дії, що може призвести до серйозної уразливості в безпеці.

Також він потребує більше простору, може потребувати більше 300 байтів для збереження простого ідентифікатора користувача, оскільки вони також зберігають інші дані для аутентифікації [16].

Тому у веб-застосунку для організації персональних тренувань та дієти з інтеграцією штучного інтелекту було обрано використання авторизації на основі JWT токенів, адже важливо забезпечити надійність та швидкість обробки запитів на серверній частині.

Таким чином, було проаналізовано технології та обрано найефективніші для використання їх при розробці серверної частини веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту. Для авторизації буде використовуватись авторизація на токенах доступу. Для зв’язку з базою даних, буде використана технологія EntityFramework, оскільки вона зручна у використанні та надає обширні можливості для оптимізації роботи бази даних.

2 МЕТОД ІНТЕГРАЦІЇ ВЕБ-ЗАСТОСУНКУ ЗІ ШТУЧНИМ ІНТЕЛЕКТОМ

Штучний інтелект (ШІ) являє собою галузь комп'ютерних наук, яка вивчає створення програм, агентів і систем, здатних виконувати завдання, що зазвичай вимагають людського інтелекту. Ця галузь науки розвивається на швидкому темпі і здатна застосовуватися в різних сферах, включаючи медицину, фінанси, автомобільну промисловість, та, зокрема, сферу здоров'я та фізичної активності.

ШІ ґрунтується на кількох основних принципах, які формують його здатність аналізувати дані та приймати рішення:

— машинне навчання — це підгалузь ШІ, в якій системи навчаються вирішувати завдання на основі даних, тобто моделі машинного навчання визначають закономірності в наборах даних і використовують цю інформацію для прийняття рішень;

— глибинне навчання — це підгалузь машинного навчання, яка використовує нейронні мережі з багатьма шарами для аналізу складних зв'язків у даних;

— аналітика даних — ШІ включає інструменти для аналізу великих обсягів даних і виділення важливих патернів та взаємозв'язків, що дозволяє виявляти приховану інформацію та робити прогнози на основі історичних даних;

— обробка природних мов — ШІ може аналізувати та розуміти мовну інформацію, включаючи текстові повідомлення, голосовий ввід і навіть мовлення.

Штучний інтелект як персональний дієтолог та тренер базується на здатності аналізувати обширні дані та надавати індивідуальні рекомендації для кожного користувача.

Аналіз починається зі збору різноманітних даних про користувача. Це включає в себе інформацію про його харчування (історія їжі, уподобання, алергії), рівень фізичної активності (типи тренувань, інтенсивність, тривалість), медичну історію (хвороби, діагнози, лікування), а також особисті цілі і обмеження. Для збору цих

даних можуть використовуватися мобільні додатки, спеціалізовані прилади (наприклад, трекери фізичної активності) та ручний ввід користувача.

Після збору дані піддаються обробці та структуруванню. Використовуються методи обробки природної мови для аналізу текстової інформації, інші алгоритми для обробки числових даних, та глибинне навчання для виявлення складних патернів в даних.

На основі оброблених даних створюються індивідуалізовані плани харчування та тренувань. ШІ враховує попередні звички користувача, його медичну історію і цілі, і розробляє плани, які оптимально відповідають його потребам. Наприклад, якщо користувач має алергію на певні продукти, ШІ рекомендує альтернативи. Якщо метою є зниження ваги, план харчування буде спрямований на створення дефіциту калорій.

Після впровадження індивідуалізованого плану, ШІ постійно моніторить реакцію користувача. Він може враховувати зміни в показниках здоров'я, рівні фізичної активності та вагу. Якщо з'являються нові обмеження або користувач змінює свої цілі, ШІ адаптує плани харчування та тренувань для відповіді на ці зміни, щоб забезпечити найкращі результати.

Аналіз даних та індивідуалізація є ключовими компонентами використання ШІ як персонального дієтолога та тренера. Цей процес дозволяє створювати індивідуалізовані плани, які максимально враховують потреби та обмеження кожного користувача, сприяючи досягненню оптимальних результатів в здоровому харчуванні та фізичній активності.

Наразі існує багато різних сайтів, які розповідають про важливість правильної дієти та системи тренувань. На таких сайтах зазвичай публікуються статті про харчування, вправи, фізичні навантаження, тощо. Деякі сайти навіть пропонують розрахувати денну норму калорій, використовуючи відомі формули, та спробувати скласти свій раціон виходячи із результатів формули. Але усі ці сайти надають користувачам лише загальну інформацію за спортивною тематикою. Оскільки неможливо, при написанні статті, враховувати особливості організму кожної людини, що займається спортом. Деякі веб-платформи пропонують приватну консультацію із

тренером, що може розписати усі аспекти харчування та тренувань, але, часто, такі послуги є дуже дорогими і далеко не всі користувачі можуть ними скористатись.

Останнім часом, відбувається стрімкий розвиток штучного інтелекту, який знаходиться у відкритому доступі. Він вже навчився писати твори різних жанрів, малювати картини, змінювати існуючі зображення, допомагати знаходити помилки в коді, а також, його можна використовувати як персонального тренера.

Саме тому, при розробці веб-додатку, було вирішено використовувати чат-бот із штучним інтелектом ChatGpt. Оскільки штучний інтелект, має змогу, проаналізувати інформацію про користувача і, відштовхуючись від цієї інформації, побудувати найкращий денний раціон, розрахувати калорії, вказати які вправи повинен робити користувач для досягнення результату. Завдяки цьому, веб-застосунок, зможе знайти підхід до кожного користувача особисто, і побудувати йому дієту опираючись на його цілі, можливості, харчові звички, та образ життя

2.1 Аналіз можливостей штучного інтелекту як персонального дієтолога та тренера

Перед реалізацією інтеграції зі штучним інтелектом ChatGpt необхідно оцінити його можливості як персонального тренеру та дієтологу. Для цього було використано безкоштовну версію штучного інтелекту, для того щоб подивитись яка інформація про користувача йому потрібна для побудови денного раціону.

Першим запитанням до штучного було «Чи зможеш ти розписати дієту для користувача, у якого ціль втратити зайву вагу?». У відповідь було отримано загальні рекомендації для зменшення ваги, такі як пити багато води та контролювати калорії. Це не відповідало вимогам відповіді, які потрібні були для реалізації веб-застосунку із реалізацією персонального дієтологу та тренеру у вигляді штучного інтелекту. Саме тому потрібно спочатку провести налаштування штучного інтелекту, щоб його відповідь була чіткою та конкретною.

У новому чаті першим повідомленням була настанова про те, що штучний інтелект тепер повинен поводити себе як персональний дієтолог, мета якого розписати клієнту дієту (рисунок 2.1).

У відповідь на це штучний інтелект попросив надати інформацію про користувача, таку як мета, харчові звички, а також алергії, будь які обмеження у їжі та будь-які інші обставини важливі для дієтичного плану.

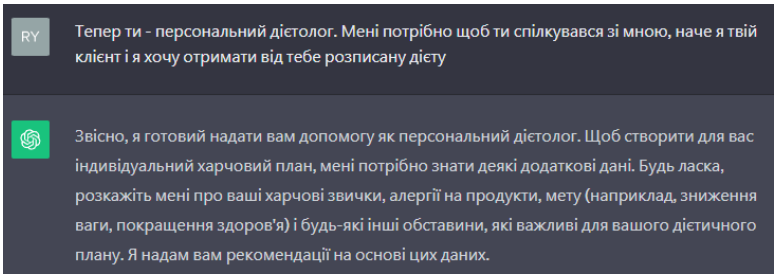


Рисунок 2.1 — Запит-налаштування штучного інтелекту

Після надання необхідної інформації про антропометрію користувача, штучний інтелект згенерував загальні правила, яких потрібно дотримуватись, щоб втратити зайву вагу. Після цього послідкував запит на побудову конкретної дієти на 1 день, і штучний інтелект у відповіді, розписав усі прийому їжі за день (рисунок 2.2).

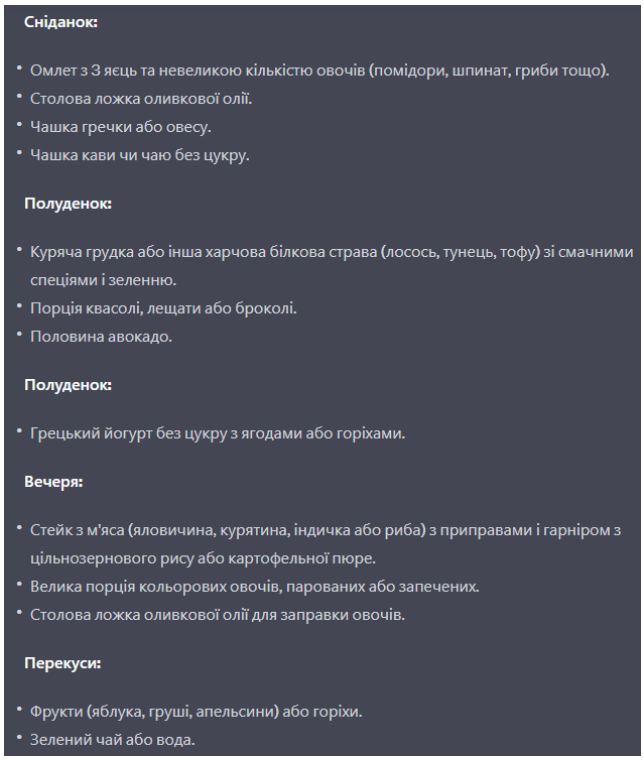


Рисунок 2.2 — Опис дієти на 1 день штучним інтелектом

У результаті штучний інтелект згенерував дієту на 1 день, згідно деяких параметрів користувача. Але треба скласти запит, який буде формувати дієту на декілька днів, враховувати усю необхідну інформацію про користувача, а також містити корисну інформацію про порції та калорії кожної страви.

2.2 Формування різних типів запитів до штучного інтелекту

Необхідно поставити завдання штучному інтелекту, щоб він зрозумів, що саме користувач від нього потребує.

Запит повинен містити усю необхідну інформацію про користувача, від якої будуть залежати рекомендації в харчуванні. Тому у запит буде додано інформацію про вік, стать, зріст, висоту та вагу користувача. Також штучному інтелекту важливо розуміти рівень фізичної активності користувача, оскільки це напряму впливає на кількість калорій, які користувач може споживати за добу. Важливо, щоб штучний інтелект розумів ціль користувача. Ціллю користувача може бути:

— втратити зайву вагу, зберігши м’язи;

— зберегти свою форму;

— набрати м’язову масу.

Інформація, описана вище, є обов’язковою при формуванні запиту до штучного інтелекту. Це потрібно враховувати при проектуванні серверної частини веб-платформи. Але також користувач за бажанням може вказати свої вподобання у їжі та будь які алергії чи непереносимість продуктів. Це також важливо вказати у запиті, який буде оброблятись штучним інтелектом, перед побудовою дієти.

Після чого, важливо вказати в якому форматі користувач очікує отримати відповідь від штучного інтелекту, а також яку інформацію обов’язково потрібно вказати у дієті. Приблизний вигляд фінального запиту до штучного інтелекту представлено у лістинг 2.1.

У лістингу, представлений запит до штучного інтелекту, в якому вказана уся необхідна інформація про користувача. Окрім цього розписано про алергію користувача, його обмеження в їжі, неулюблена та улюблена їжа. Також вказано, що

дієта повинна мати вигляд документу, в якому буде розписано денну допустиму кількість калорій для користувача, а також приблизний калораж кожної страви.

Лістинг 2.1 — Приклад запиту до штучного інтелекту

```
Ти персональний дієтолог. Я чоловік мені 22 роки, мій зріст 172 сантиметри, а моя вага складає 72 кілограми. Мою денну фізичну активність можна описати так: я програміст, тому рідко виходжу з дому, а моя ціль: зкинути зайву вагу. Мені потрібно щоб ти розписав мені дієту згідно моїх параметрів. Але, будь ласка, зверни увагу на деякі деталі, щодо майбутньої дієти: * У мене є алергія на наступне: горіхи. * У мене є обмеження у споживанні: лактоза. * Мені не подобається наступна їжа: гречана каша, тому при побудові дієти використовуй її по мінімуму. * Я люблю наступну їжу: морепродукти. Дієту розпиши наче ти персональний дієтолог, і розписуєш дієту своєму клієнту у вигляді документу. Потрібно розписати дієту на тиждень, з різними стравами, але збереженням денного калоражу. Документ напиши строгою, технічною мовою. Потрібно, щоб він включав в себе денний калораж, а також біля кожної страви було розписано хоча б приблизна кількість її калорій.
```

У відповідь на такий запит штучний інтелект представить текст, оформлений як документ, в якому буде вказано раціон харчування на тиждень, кількість калорій в кожній страві, а також описано скільки калорій всього потрібно отримувати користувачу в день, щоб досягти своєї мети.

По аналогії було сформовано запит для побудови тренування для користувача, відповідно до його цілей. У цьому випадку штучний інтелект налаштовувався на те, щоб бути персональним тренером, замість персонального дієтолога. Уся ж інша інформація подібна до тієї, яка використовувалась при побудові запиту на дієту.

У документації OpenAI API описано вказівки для формування найкращого запиту до штучного інтелекту, щоб отримати обширну відповідь згідно вимог. Розглянемо поради з документації.

Перша вказівка — надати довідковий текст. Штучний інтелект може вигадувати фальшиві відповіді, особливо коли їх запитують про езотеричні теми або про цитати та URL-адреси. Подібно до того, як аркуш нотаток може допомогти студенту краще виконати тест, надання довідкового тексту до цих моделей може допомогти відповісти з меншою кількістю вигадок.

Наступна вказівка — розділяти складні завдання на простіші та менші задачі. Подібно до того, як у розробці програмного забезпечення прийнято розбивати складну систему на набір модульних компонентів, те ж саме стосується завдань, переданих мовній моделі. Складні завдання, як правило, мають більший рівень

помилок, ніж простіші завдання. Крім того, складні завдання часто можна перевизначати як робочий процес із простіших завдань, у якому результати попередніх завдань використовуються для створення вхідних даних для наступних завдань. Таким чином, під час побудови запитів серверна частина розбиває побудову системи тренувань та дієти на окремі задачі, щоб штучний інтелект не плутав інформацію та виконував лише одну задачу за один запит.

Ще однією вказівкою є необхідність надання штучному інтелекту часу на обробку інформації. Подібно того як людині необхідний час щоб прийти до певного рішення, штучному інтелекту потрібен час, щоб обробити надану інформацію та описати відповідь. Окрім цього, запит про «ланцюг думок» перед відповіддю може допомогти моделі більш надійно обґрунтувати свій шлях до правильних відповідей.

Також потрібно тестувати зміни до запиту систематично. Поліпшити продуктивність легше, якщо її можна виміряти. У деяких випадках модифікація підказки дозволить досягти кращої продуктивності на кількох окремих прикладах, але призведе до погіршення загальної продуктивності на більш репрезентативному наборі прикладів. Тому, щоб переконатися, що зміна позитивно впливає на продуктивність, може знадобитися визначити комплексний набір тестів.

Останньою вказівкою є написання чітких інструкцій. Щоб отримати релевантну відповідь, необхідно бути впевненим, що запити містять будь-які важливі деталі чи контекст. Інакше штучний інтелект вимушений здогадуватися, що саме малось на увазі [17].

2.3 Вдосконалення методу інтеграції штучного інтелекту ChatGpt з веб-застосунком

Існує декілька можливостей реалізувати використання штучного інтелекту у веб-застосунку. Деякі компанії, що розробляють штучний інтелект, надають API, на яке можна надсилати запити, які буде обробляти штучний інтелект. Іноді розробляються окремі бібліотеки та фреймворки, під конкретні платформи розробки. Іноді штучний інтелект використовується окремо від проекту, та просто аналізує додаток на можливість його оптимізації.

Компанія OpenAI, яка займається розробкою ChatGpt надає платний доступ до API, який містить велику кількість методів та можливостей для використання їх штучного інтелекту.

2.3.1 Використання методів OpenAI API

OpenAI виділяє широкий спектр використання їх штучного інтелекту для розробників, що використовують API. ChatGpt можна налаштувати таким чином, щоб він шукав помилки у коді, пояснював код з будь-якої мови програмування, перевіряв текст на наявність граматичних помилок, виділяв ключові слова з великого тексту та багато іншого. Для цього представлені різні ендпоінти, деякі з яких більше не підтримуються. Розглянемо ці методи більш детально.

Метод Speech-to-text — переводить аудіо у текст. У цей метод можна передати файл із аудіо, і як результат він поверне стрічку із текстом, що було озвучено в звуковій доріжці. Окрім цього штучний інтелект може перекласти цей текст різними мовами.

Метод Text-to-speech — може озвучити будь-який текст, який приймає як параметр. Відповіддю буде аудіо-файл з озвученим текстом.

Метод Text generating — генерація текстової відповіді на запит про спілкування та опис інформації. Серверна частина веб-додатку використовує цей метод для передачі інформації про користувача штучному інтелекту, після чого у відповідь, ChatGpt опише користувачу дієту та тренування.

Метод Image generation — створення зображення або зміна існуючого згідно підказок, що надав користувач. За допомогою цього методу можна згенерувати зображення для дієти, що запросив користувач.

Методи Moderation — методи, що використовуються для модерації. Штучний інтелект аналізує повідомлення, яке надсилається йому у якості параметру і може визначити такі показники:

— ненависть;

— погрози;

— харасмент;

— завдавання собі шкоди;

— сексуалізація;

— жорстокість.

Штучний інтелект надає числові значення кожного із показників, що дозволяє визначити чи можна публікувати дане повідомлення у веб-додатку.

Вбудовані методи. Вбудовані тексти OpenAI вимірюють пов’язаність текстових рядків. Вбудовані методи зазвичай використовуються для:

— пошуку, коли результати впорядковуються за релевантністю рядку запиту;

— кластеризації, де текстові рядки групуються за подібністю;

— рекомендації, де рекомендовано елементи з пов’язаними текстовими рядками;

— виявлення аномалії, де ідентифікуються випадки з невеликою пов’язаністю;

— вимірювання різноманітності, де аналізуються розподіли подібності;

— класифікація, де текстові рядки класифікуються за їх найбільш подібними мітками [18].

На даний момент у веб-застосунку для організації тренувань та дієти використовується лише метод для створення чату і спілкування зі штучним інтелектом, але із розвитком проекту можна налаштувати штучний інтелект таким чином, щоб після генерації дієти, тексту визначались ключові слова, після чого за допомогою методу із генерацією зображень, ключові слова передавались у якості підказки, щоб ChatGpt згенерував зображення до дієти. Таким чином інтерфейс користувача буде більш наглядним. Окрім цього, в майбутньому, можна додати можливість озвучування дієти штучним інтелектом, для людей із обмеженим зором.

2.3.2 Розробка модуля для інтеграції зі штучним інтелектом

Тригером до підготовки та надсилання запиту до штучного інтелекту є ендпоінт «ask-for-diet» (лістинг 2.2), який приймає ідентифікатор користувача, для якого буде складатись дієта.

Лістинг 2.2 — Приклад ендпоінту ask-for-diet

```
[HttpGet("ask-for-diet")]
        public IActionResult AskForDiet(string userId)
        {
            try
            {
                var dietData = _applicationContext.DietData.FirstOrDefault(c =>
c.UserId == userId);
                var foodDetails = _applicationContext.FoodDetails.FirstOrDefault(d
=> d.UserId == userId);

                if (dietData == null)
                {
                    return NotFound($"No information about diet for user {userId}");
                }

                var dietRequest = _requestBuilder.BuildDietRequest(new
DietRequestModel(dietData, foodDetails));
                var chatResponse = _chatGptService.Ask(dietRequest);

                return Ok(chatResponse);

            }
            catch (Exception e)
            {
                return StatusCode(500, e.Message);
            }}
```

Метод, що представлений вище, отримує із бази даних інформацію для побудови дієти для користувача та харчові звички. Якщо ж інформація, необхідна для генерації дієти, відсутня, ендпоінт поверне помилку, з описом що користувач не надав усієї потрібної інформації. Якщо ж інформація була знайдена, то сервіс для побудови запитів почне побудову повідомлення (лістинг 2.3), що буде надіслане штучному інтелекту.

З прикладу видно, що з самого початку буде побудоване повідомлення, в якому відбувається налаштування ролі штучного інтелекту. У повідомленні сказано, що штучний інтелект повинен вести себе як персональний дієтолог. Після цього, у методі BuildBaseMessage, буде вказано, усю необхідну інформацію про користувача, таку як вік, стать, зріст, вага та інше.

Лістинг 2.3 — Приклад методу для побудови повідомлення штучному інтелекту

```
public string BuildDietRequest(DietRequestModel dietRequestModel)
        {
            var foodDetails = dietRequestModel.FoodDetails;
            var dietRequest = new StringBuilder(_messageBegining);

            dietRequest.AppendLine("\"role\": \"system\", \"content\": \"Ти
персональний дієтолог\"},");
            BuildBaseMessage(dietRequestModel.DietData, dietRequest);
            dietRequest.Append("Мені потрібно щоб ти розписав мені дієту згідно моїх
параметрів.");

            if (IsAnyFoodDetails(foodDetails))
            {
                AppendFoodDetails(foodDetails, dietRequest);
            }
            dietRequest.Append("Дієту розпиши наче ти персональний дієтолог, і
розписуєш дієту своєму клієнту у вигляді документу. " +
                "Потрібно розписати план харчування на тиждень, з різними стравами,
але збереженням денного калоражу. " +
                "Документ напиши строгою, технічною мовою. Потрібно, щоб він включав
в себе денний калораж клієнту, а також біля кожної " +
                "страви було розписано хоча б приблизна кількість її калорій. Ім'я в
документі не вказуй\"}]}");

            var result = dietRequest.ToString();
            return result;
        }
```

Далі в повідомленні буде визначено ціль, яку користувач очікує від штучного інтелекту, тобто описання дієти згідно його даних. Якщо ж користувач вказав додаткову інформацію про харчові звички, це також буде додано до запиту (лістинг 2.4).

В кінці запиту описано формат, в якому серверна частина очікує отримати відповідь від штучного інтелекту, а саме щоб ChatGpt розписав дієту як документ, суто технічною мовою. Дієту потрібно розписати на тиждень та обов’язково вказувати калорії, щоб користувач мав загальне розуміння дієти.

Окрім цього в запиті вказується яка саме модель чату GPT буде використовуватись. З описаного методу видно, що до запиту окремо буде додана уся інформації про харчування, яку вказав користувач. Якщо ж певна інформація не була додана користувачем, вона буде ігноруватись при надсиланні запиту до штучного інтелекту.

Лістинг 2.4 — Приклад методу додавання харчових звичок користувача

```
private void AppendFoodDetails(FoodDetails dietData, StringBuilder dietRequest)
        {
            dietRequest.Append("Але, будь ласка, зверни увагу на деякі деталі, щодо
майбутньої дієти:");

            if (!string.IsNullOrEmpty(dietData.Allergies))
            {
                dietRequest.Append($"* У мене є алергія на наступне:
{dietData.Allergies}.");
            }
            if (!string.IsNullOrEmpty(dietData.FoodRestrictions))
            {
                dietRequest.Append($"* У мене є обмеження у споживанні:
{dietData.FoodRestrictions}.");
            }
            if (!string.IsNullOrEmpty(dietData.DislikeFood))
            {
                dietRequest.Append($"* Мені не подобається наступна їжа:
{dietData.DislikeFood}, " +
                    $"тому при побудові дієти використовуй її по мінімуму.");
            }
            if (!string.IsNullOrEmpty(dietData.FoodPreferences))
            {
                dietRequest.Append($"* Я люблю наступну їжу:
{dietData.FoodPreferences}.");
            }}
```

Після того як запит з повідомленнями сформований, його необхідно надіслати на ендпоінт OpenAI API. Метод для надсилання запиту штучному інтелекту описано у лістингу 2.5.

Лістинг 2.5 — Метод для надсилання запиту на штучний інтелект

```
public string Ask(string chatRequest)
        {
            var content = new StringContent(chatRequest, Encoding.UTF8,
"application/json");

            HttpResponseMessage response =
_chatGptClient.PostAsync(CHAT_POST_ADDRESS, content).Result;

            var responseString = response.Content.ReadAsStringAsync().Result;
            return responseString;}
```

Для надсилання запиту до штучного інтелекту потрібно ініціалізувати HttpClient (лістинг 2.6), і викликати метод PostAsync, який приймає URL-адресу ендпоінту та змінну типу StringContent, яка містить закодований запит, до штучного інтелекту. Даний метод повертає відповідь від ендпоінту, яка зберігається в змінній responseString, для того щоб її можна було повернути на рівень представлення.

Лістинг 2.6 — Метод для ініціалізації HttpClient

```
private void InitHttpClient()
        {
            _chatGptClient = new HttpClient();
            var token = _configuration.GetValue<string>(CHAT_GPT_TOKEN);

            _chatGptClient.DefaultRequestHeaders.Add("authorization", $"Bearer
{token}");}
```

У коді вище описується ініціалізація змінної _chatGptClient. Після створення об’єкту, до заголовків запиту додається заголовок авторизації, який повинен містити токен доступу від чату Gpt. Цей токен зберігається у user-secrets (місце збереження секретної інформації користувача), для того щоб зловмисники не могли його отримати із конфігурації. Якщо ж токену немає, усі запити будуть повертати код статусу 401 Unauthorized, що означає що запит у заголовку не містив токену доступу, через що клієнт не має права на виконання цього методу.

Після надсилання запиту повинно пройти декілька хвилин, щоб штучний інтелект згенерував відповідь та створив дієту для користувача, після чого ендпоінт відповість «200 OK» (що означає, що все виконано успішно) на запит, та відправить відповідь штучного інтелекту на рівень представлення. Увесь код модулю інтеграції з штучним інтелектом описано у додатку Д.

2.3.3 Розробка архітектури модулю інтеграції з штучним інтелектом

Зберігати усю логіку для отримання запиту від клієнту, побудові запиту для штучного інтелекту, відправки цього запиту та очікування відповіді в одному методі було б некоректно з точки зору архітектури та перевикористання коду. Саме тому було прийнято рішення розбити усі необхідні методи по різним класам-сервісам, кожен з яких буде мати лише одну відповідальність, відповідно до першого принципу програмування SOLID. Діаграму класів компоненту інтеграції зі штучним інтелектом для побудови і відправки запитів представлено на рисунку 2.3.

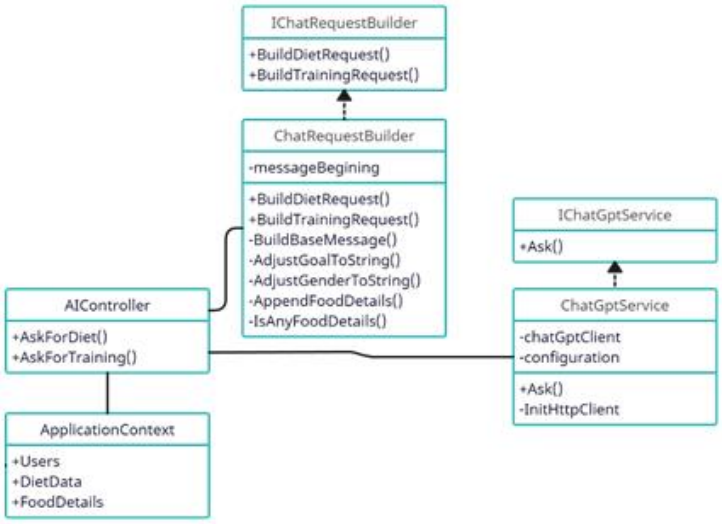


Рисунок 2.3 — Діаграма класів компоненту інтеграції зі штучним інтелектом

Із рисунку 2.3 видно, що окремі класи-сервіси відповідають за різні функціональні можливості. Так, наприклад, після того як контролер отримав запит від клієнту, на побудову дієти, перш за все він звернеться до класу ApplicationContext, щоб отримати інформацію про дієту конкретного користувача.

Інформацію із бази даних він передасть у метод BuildDietRequest класу ChatRequestBuilder, який відповідає за побудову запиту на створення дієти, враховуючи усю інформацію про користувача.

Після того, як запит буде побудовано, контролер викличе метод Ask класу ChatGptService, у який передасть тільки що побудований запит. В свою чергу метод Ask відправить запит на OpenAI API, де запит буде оброблятись штучним інтелектом.

Алгоритм діяльності модулю інтеграції веб-застосунку зі штучним інтелектом представлено у вигляді UML-діаграми діяльності і наведено на рисунку 2.4.

З рисунку 2.4 видно, що після того як приходить запит на контролер від клієнту, перш за все відбувається перевірка чи користувач існує. Якщо користувач, ідентифікатор якого було отримано як параметр запиту, не існує, ендпоінт поверне відповідь 404 з описом помилки, що користувача не знайдено у базі даних. Після чого відбудеться перевірка чи користувач заповнив необхідну інформацію для побудови запиту. Якщо ж ні — то ендпоінт також поверне помилку 404, але в описі вже буде вказано що інформація про користувача не знайдена.

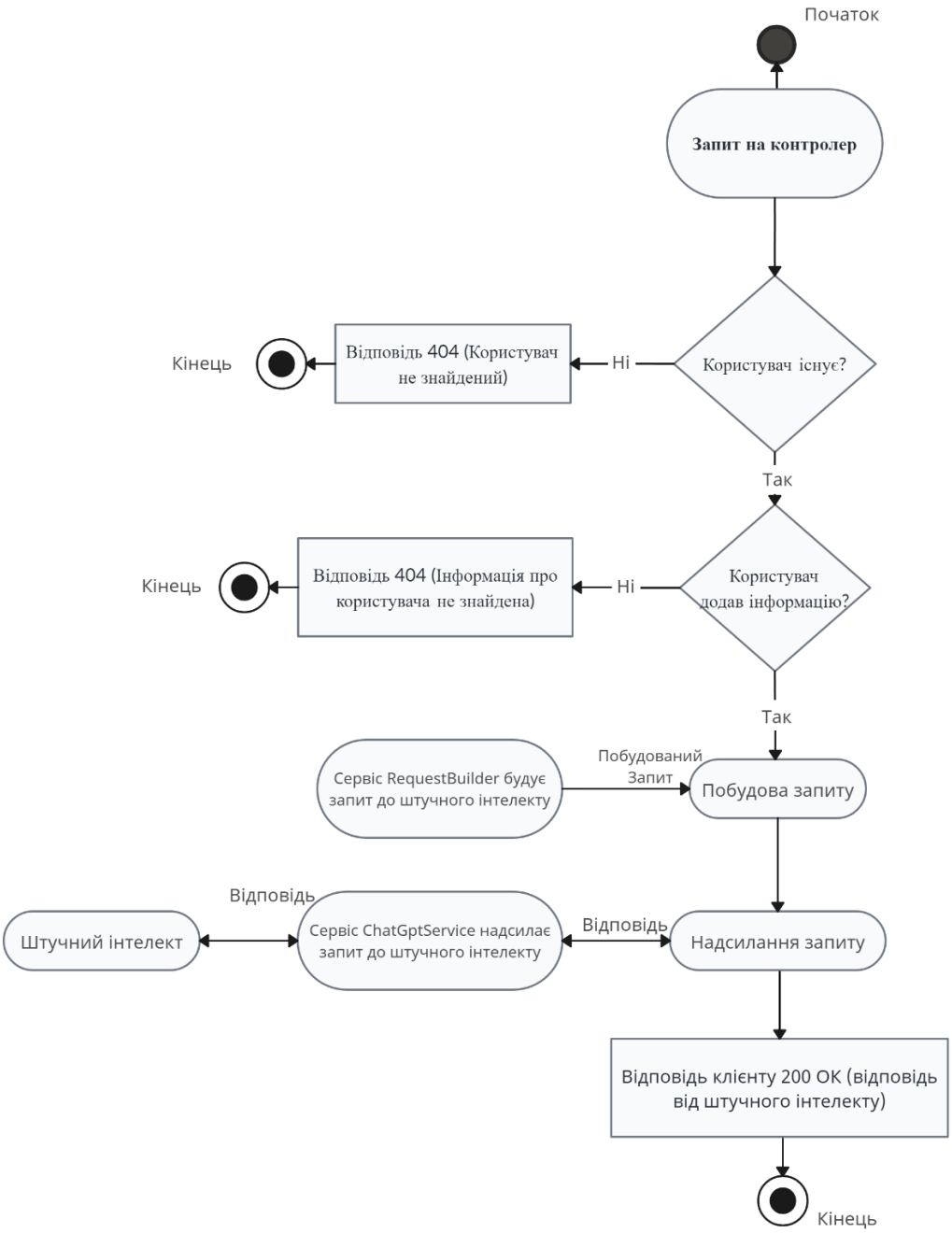


Рисунок 2.4 — UML-діаграма діяльності модулю інтеграції веб-застосунку зі штучним інтелектом

Після того як верифікація пройшла, відбудеться побудова запиту до штучного інтелекту, що описана в класі ChatGptRequestBuilder. Після того як запит побудований, за допомогою класу ChatGptService запит буде надіслано до штучного інтелекту. Після отриманої відповіді, вона буде надіслана клієнту із кодом статусу 200 OK.

3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ

При розробці серверної частини веб-застосунку потрібно провести аналіз вимог до проекту та визначити інформацію, що повинна зберігатись у базі даних.

3.1 Визначення вимог до запиту, що надсилається на серверну частину

Веб-застосунок повинен містити інформацію про користувача, та давати змогу реєстрації та авторизації. Лише авторизовані користувачі матимуть змогу його використовувати. Для реєстрації користувача було визначено такі поля:

— ім’я (буде містити в собі ім’я та прізвище користувача);

— електронна пошта користувача, яка буде використовуватись для авторизації у веб-додатку;

— пароль.

Зберігати пароль у базі даних було б некоректно, з точки зору безпеки даних. Саме тому у базі даних зберігається лише хеш паролю, для того, щоб була можливість перевірити правильність введеного паролю під час авторизації.

Після реєстрації користувача, потрібно отримати інформацію про його харчові звички, а також його ціль, для можливості побудови запиту штучному інтелекту, який в майбутньому опише дієту. Проаналізувавши які дані потрібні штучному інтелекту для побудови правил харчування, було визначено такі поля:

— вік;

— зріст;

— поточна вага;

— стать;

— ціль (схуднення, збереження ваги, набір м’язової маси);

— фізична активність користувача.

Базуючись на цій інформації, алгоритми серверної частини зможуть збудувати запит для штучного інтелекту на побудову персональної дієти та тренувань користувача. Окрім цього, при побудові денного раціону користувача важливо

враховувати його харчові звички, а також обмеження. Тому додатково користувач може вказати такі поля:

— алергії на будь які продукти;

— обмеження в їжі (наприклад непереносимість лактози);

— вподобання в їжі (таким продуктам буде надаватись перевага при побудові раціону);

— їжа, що не подобається (такі продукти будуть або виключені, або мінімізовані у раціоні користувача).

Маючи усю цю інформацію можна побудувати найкращу дієту для користувача, враховуючи його ціль, фізичну активність, та вподобання у їжі. Ці дані будуть зберігатись у базі даних, прив’язані за ідентифікатором конкретного користувача. Інформацію про антропометрію та ціль користувача потрібно вказати, перед побудовою запиту до штучного інтелекту, адже без неї запит буде не повним. В свою чергу інформація про харчові звички є необов’язковою.

3.2 Розробка структурної схеми серверної частини веб-застосунку

Перед програмною реалізацією веб-застосунку необхідно визначити основні програмні модулі, що покривають функціональні можливості. У веб-застосунку необхідно описати можливості авторизації користувача, для забезпечення безпеки веб-застосунку, та можливість користувача згенерувати дієту та систему тренувань за допомогою штучного інтелекту. Окрім цього необхідно дати користувачам можливість збереження згенерованих відповідей. Для цього необхідно підключити базу даних до серверної частини. Вигляд загальної структурної схеми серверної частини веб-застосунку представлено на рисунку 3.1.

З рисунку 3.1 видно, що клієнтська частина відправляє запити на контролери. При розробці API, клієнтською частиною може виступати будь-що, зокрема мобільні додатки, сторінки сайтів, програми, тощо. В даному випадку клієнтська частина — це рівень представлення веб-застосунку.

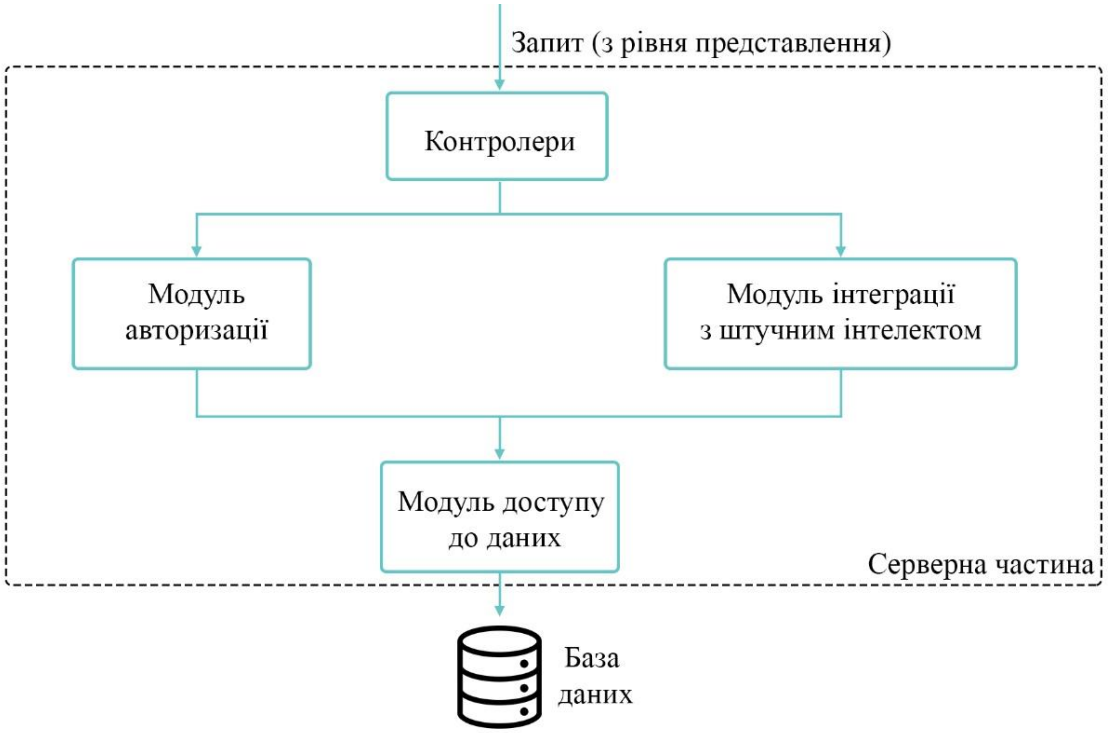


Рисунок 3.1 — Структурна схема серверної частини веб-застосунку

В свою чергу контролери мають доступ до сервісу авторизації. З його допомогою можна зареєструвати користувача, авторизувати користувача за електронною поштою та паролем, згенерувати користувачу токен доступу та токен оновлення, видалити аккаунт користувача та вийти з облікового запису. Спочатку клієнт, для того щоб отримати доступ до методів інтеграції зі штучним інтелектом, повинен отримати токен доступу. Для цього необхідно авторизувати користувача, після чого буде згенеровано токен доступу, який необхідно додати до заголовків запитів.

Після авторизації користувача та отримання токену доступу клієнт матиме доступ до методів інтеграції з штучним інтелектом. Цей модуль відповідає за створення модифікованого запиту до штучного інтелекту для створення дієт та системи тренувань. Після генерації запиту, його буде надіслано до штучного інтелекту OpenAI, а його відповідь буде надіслано клієнту.

Для інтеграції штучного інтелекту з веб-застосунком було розроблено програмний модуль, який включає в себе два класи та два інтерфейси:

— інтерфейс IChatRequestBuilder, в якому визначені усі необхідні публічні функції, необхідні для отримання запиту для побудови дієти та тренувань;

— інтерфейс IChatGptService, в якому визначені усі необхідні публічні методи для інтеграції зі штучним інтелектом;

— клас ChatRequestBuilder, в якому реалізовано методи для побудови запиту для отримання дієти та тренувань, а також приватні методи, в яких винесено базові елементи повідомлення штучному інтелекту та додавання харчових звичок відповідно до їх наявності;

— клас ChatGptService, в якому реалізовано метод, що ініціалізує клієнта та метод, що надсилає запит штучному інтелекту та отримує відповідь.

Окрім цього на структурній схемі представлено рівень доступу до даних, який відповідає за зв’язок серверної частини з базою даних. У ньому описані методи для отримання інформації із бази даних за певними параметрами, а також збереження об’єктів у базу даних.

Розроблено UML-діаграму діяльності серверної частини, яка представлена в графічному ДОДАТКУ Г.

З діаграми видно, що перш за все користувачу необхідно авторизуватись, для того щоб отримати токен доступу, оскільки більшість методів захищені авторизацією. Якщо користувача не існує, необхідно його зареєструвати, використовуючи електронну пошту та пароль. Після авторизації користувача для нього згенерується унікальний токен доступу, який необхідно додати до заголовків всіх запитів.

Після цього користувач, повинен додати інформацію про свої антропометричні дані та харчові звички, адже ця інформація використовується при побудові запиту до штучного інтелекту. Ці дані будуть зберігатись у базі даних, прив’язані за ідентифікатором конкретного користувача.

Після додавання інформації, у модулі інтеграції з штучним інтелектом буде побудовано запит до штучного інтелекту, використовуючи усю інформацію про користувача. Побудований модифікований запит буде надісланий до штучного

інтелекту та серверна частина перейде у режим очікування відповіді від штучного інтелекту, яка після її отримання буде надіслана клієнту.

3.3 Реалізація серверної частини на платформі ASP.Net

Для імплементації серверної частини було обрано платформу ASP.Net, що використовує мову програмування C#, оскільки ця платформа пропонує гнучкий підхід у розробці, а також можливість використання різних фреймворків для роботи із базою даних, реалізації авторизації, надсилання запитів на стороннє API тощо. В свою чергу платформа ASP.Net пропонує декілька підходів, до створення серверної частини, а саме:

— MVC;

— Web-Api;

— Web-Forms.

Технологія Web-Forms застаріла, тому вона не розглядалась для використання при створенні веб-додатку.

Технології MVC та Web-Api досі підтримуються Microsoft, тому було проведено аналіз обох технологій для використання однієї із них під час розробки.

MVC (Models Views Controllers) — технологія що базується на підході створення моделей, представлень та контролерів. Кожен із шарів відповідає за різні частини роботи.

Моделі відповідають за набір даних, що передаються між рівнем логіки та представлення. Вони містять в собі необхідну інформацію для обробки зі сторони сервісів, та відображення для користувачів зі сторони представлень.

Представлення відповідають за інтерфейс, що бачить користувач. У представленнях визначається як саме буде виглядати веб-сторінка, яку побачить користувач а також де буде представлена інформація, яку рівень представлень отримав від серверної частини.

Контролери відповідають за рівень бізнес-логіки. На рівні контролерів описано алгоритми для отримання інформації з бази даних, обробка цієї інформації та надсилання цієї інформації на рівень представлення.

Використовуючи технологію MVC можна чітко розділити усі рівні додатку за шарами. Але основним мінусом цієї технології є потреба у використанні Razor Pages при розробці рівня представлення. Razor Pages використовує файли із розширенням .cshtml (CSharpHtml). Тобто в одному файлі міститься розмітка сторінки, що описується за допомогою мови html, та логіка, що описується мовою C#. При роботі у команді використання файлів .cshtml не є доцільним, оскільки команда, яка займається рівнем представлення буде обмежена при виборі технологій для розробки.

В свою чергу Web-Api дозволяє писати серверну частину, яка у відповідь на запити буде відправляти не готове представлення Razor Page, а код статусу про успішність операції. Окрім цього, відповідь буде містити усі необхідні данні, що потрібно відобразити на рівні представлення.

Тому при розробці серверної частини, за допомогою Web-Api описується контролер із набором методів. Кожен із методів повертає певний код статусу (лістинг 3.1).

Лістинг 3.1 — Приклад методу контролеру

```
        [Authorize]
        [HttpPost("add-diet-data")]
        public async Task<ActionResult> AddDietData([FromBody] AddDietDataRequest
addDietDataRequest)
        {
            var result = _userService.AddDietData(new
Models.UserModels.DietData(addDietDataRequest));

            return result ? Ok() : BadRequest();
        }
```

У лістингу 3.1 описано метод для додавання даних про дієту до конкретного користувача. Всередині методу використовується сервіс користувачів _userService, який намагається записати нову інформацію у базу даних і прив’язати її за ідентифікатором певного користувача. У випадку, якщо дані були додані успішно, у відповідь прийде кодом статусу 200 OK. Якщо ж щось пішло не так, наприклад даного користувача не існує, відповідно немає за ким прив’язувати дані про дієту, у відповідь на виклик даного методу повернеться код статусу 400 BadRequest, який також називається Bad Request.

Завдяки можливостям, що надає Web-Api було створено метод, який додає інформацію про дієту користувача, і буде виконуватись, коли клієнт відправить запит на адресу «http://localhost:5087/api/user/add-diet-data». У відповідь клієнт отримає код статусу, який відображає успішність додання даних. Завдяки такому підходу клієнт вільний у виборі технології для розробки рівню представлення.

Клієнтом не обов'язково повинен бути сайт. Будь який додаток може надсилати запити за даною адресою. Це є перевагою використання підходу Web-Api, адже у майбутньому можна розробити веб-застосунок, який буде використовувати дану серверну частину для отримання дієти для користувача.

3.3.1 Коди статусу та методи серверної частини веб-застосунку

API працює за таким принципом — приходить запит на ендпоінт, після чого відбувається обробка інформації, та метод відповідає на запит кодом статусу. Наприклад, на запит про отримання дієти користувачем, серверна частина дістане інформацію про користувача із бази даних, та відправить запит до штучного інтелекту. Якщо штучний інтелект відповів і запит пройшов успішно, метод відправить код статусу 200 OK, всередині якого буде міститись відповідь штучного інтелекту.

Якщо ж користувача з таким ідентифікатором не було у базі даних, або він не додав усю необхідну інформацію для побудови запиту, серверна частина поверне код статусу 404, з описом помилки. Код статусу 404 також можна назвати NotFound або «не знайдено».

Загалом усі коди статусу можна поділити на умовні категорії:

— від 100 до 199 — інформаційні;

— від 200 до 299 — успішні;

— від 300 до 399 — перенаправлення;

— від 400 до 499 — клієнтські помилки;

— від 500 до 599 — серверні помилки.

За рахунок даної інформації, клієнт може розуміти яку саме відповідь вони отримали, і що з цим робити. Так, наприклад, якщо клієнт отримав 400-ий код

статусу, це означає що він припустився помилки під час створення запиту. Наприклад відповідь 400 — BadRequest або «поганий запит», означає якусь помилку в запиті, 401 — Unauthorized або «неавторизований», означає що користувач не авторизувався, і не додав токен доступу до заголовків. В свою чергу 500-ті коди статусу навпаки означають що проблема сталась на сервері і клієнт до неї не причасний, наприклад 500 InternalServerError або ж «внутрішня помилка серверу» означає що на сервері сталась непередбачувана помилка. 200-ті статус коди завжди означають успішно виконану операцію.

Окрім цього існують запити різних типів, такі як:

— Post — використовуються для надсилання інформації на сервер, щоб створити чи оновити ресурс;

— Get — використовується для отримання інформації від серверу;

— Put — зазвичай використовуються для оновлення інформації на сервері;

— Delete — використовується для видалення інформації із серверної частини.

У веб-застосунку реалізовано методи, що представлені на рисунку 3.2

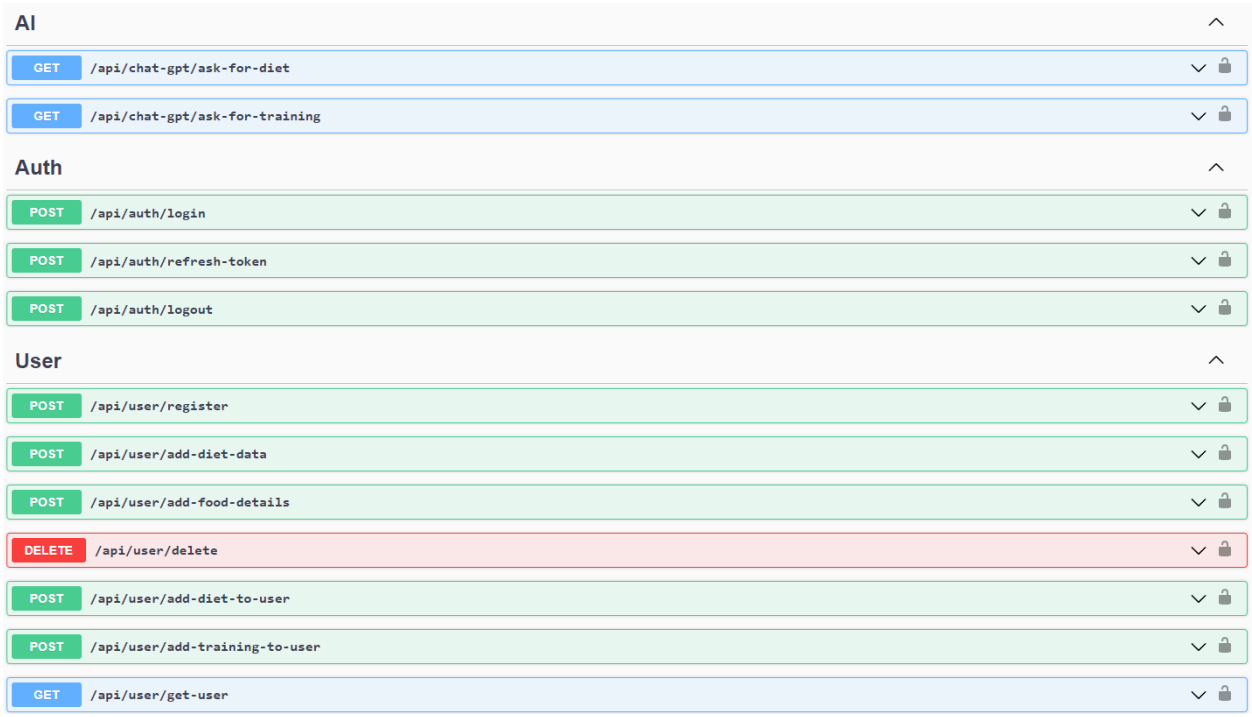


Рисунок 3.2 — Реалізовані ендпоінти у веб-застосунку

На рисунку 3.2 представлено методи, що розділені за різними контролерами і розташовані у різних розділах. Це зроблено для налаштування Routing (роутингу) між ендпоінтами. Так, наприклад, всі методи, що пов’язані з штучним інтелектом знаходяться під розділом AI, методи що пов’язані з авторизацією знаходиться у розділі Auth, а усе що пов’язане з користувачем знаходиться у розділі User.

Тип методу відображений біля його назви. Так, наприклад, метод login у контролері User є методом Post, що означає, що він використовується для відправлення інформації на сервер для подальшої обробки серверною частиною. Тобто серверна частина обробить інформацію про електронну пошту, та у випадку, якщо такий користувач існує, залогінить його та видасть йому токен доступу. Цей ендпоінт не використовується для отримання інформації про користувача. В свою чергу ендпоінт «ask-for-diet» є методом типу Get, що означає, що він використовується для отримання інформації. Тобто при виклику цього методу, клієнтська частина отримає дієту для конкретного користувача, що буде згенерована штучним інтелектом. Також у контролері User є метод типу Delete, який використовується для видалення аккаунту користувача.

3.3.2 Розробка додатку Web-Api на платформі Asp.Net

Для розробки додатку Web-Api потрібно провести налаштування проекту, щоб дозволити відправку запитів із інших ресурсів, а також специфічні параметри для авторизації, часу життя запиту, підключення до бази даних та додавання технології Swagger до проекту.

Після створення проекту з’являється файл Program.cs, у якому і проводяться налаштування.

Так, наприклад, для налаштування авторизації використовується метод AddAuthentication, який приймає певні параметри (лістинг 3.2)

У лістингу 3.2 додається схема аутентифікації, а також проводяться налаштування JWT токену, за допомогою методу AddJwtBearer(). Всередині цього методу проводиться специфічні налаштування токену, такі як збереження токену, параметри валідації, потреба у валідації життєвого циклу токену, підпису та ключа.

Лістинг 3.2 — Приклад налаштування авторизації у Program.cs

```
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
})
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.SaveToken = true;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,

            ValidIssuer = authOptions.Issuer,
            ValidAudience = authOptions.Audience,
            IssuerSigningKey = authOptions.GetSymmetricSecurityAccessKey(),

            ClockSkew = TimeSpan.Zero
        };
    });
```

Окрім цього, відбуваються налаштування символів, які повинен містити пароль користувача, щоб він вважався надійним (лістинг 3.3). З нього видно, що при створенні паролю, користувач повинен вказати хоча б вісім символів, а також повинна бути хоча б одна велика літера, одна маленька та хоча б одне число. Якщо ж пароль, який користувач створює, не буде відповідати даним вимогам, користувач отримає помилку під час реєстрації.

Лістинг 3.3 — Приклад налаштування символів у паролі користувача

```
builder.Services.AddIdentity<User, IdentityRole>(opts =>
{
    opts.Password.RequiredLength = 8;
    opts.Password.RequireNonAlphanumeric = false;
    opts.Password.RequireLowercase = true;
    opts.Password.RequireUppercase = true;
    opts.Password.RequireDigit = true;
})
```

Налаштування підключення для бази даних приведено у лістингу 3.4. На ньому показано реєстрацію контексту бази даних. Контекст — це клас, який містить в собі абстрактні таблиці, що будуть створені у базі даних. Для підключення до бази даних потрібно використовувати стрічку підключення. Вона зберігається у

конфігурації, за кодовою назвою VntuDbConnection, тому код, що представлений у лістингу, отримає стрічку підключення із конфігурації.

Лістинг 3.4 — Приклад налаштування підключення до БД

```
builder.Services.AddDbContext<ApplicationContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("VntuDbConnection")),
                    ServiceLifetime.Singleton);
```

Також потрібно налаштувати CORS (Cross-Origin Resource Sharing), для того, щоб забезпечити доступ до методів серверної частини із іншого додатку. У нашому випадку іншим додатком виступає рівень представлення. Налаштування CORS представлено у лістингу 3.5.

Лістинг 3.5 — Приклад налаштування CORS

```
builder.Services.AddCors(policy => policy.AddPolicy("AllowAnyOrigin",
                builder =>
                {
                    builder
                        .AllowAnyMethod()
                        .AllowAnyHeader()
                        .SetIsOriginAllowed(origin => true)
                        .AllowCredentials();
                }));
```

У лістингу 3.5 дозволяється використання будь якого методу та заголовку під час надсилання запитів до серверної частини.

Для зручності тестування та розробки рівня представлення у проект було додано технологію Swagger. Вона дозволяє відобразити усі методи, що описані у серверній частині, а також вказати які параметри потрібні для виконання методу. Окрім цього на сторінці Swagger буде задокументовано статуси коду, які метод може повернути на запит, а також що саме вони означають. Додавання Swagger у проект описано у лістингу 3.6.

Лістинг 3.6 — Приклад додавання Swagger у проект

```
app.UseSwagger();
app.UseSwaggerUI();
```

Окрім цього, при використанні авторизації у проекті важливо налаштувати Swagger таким чином, щоб після генерації токену доступу, можна було додати його

до заголовку усіх майбутніх запитів. Завдяки цьому можна буде надсилати запити також на методи, що захищенні авторизацією. Приклад такої конфігурації приведено у лістингу 3.7.

Лістинг 3.7 — Приклад конфігурації Swagger на додання токену

```
public void Configure(SwaggerGenOptions options)
        {
            options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
            {
                In = ParameterLocation.Header,
                Description = "Please provide a valid token",
                Name = "Authorization",
                Type = SecuritySchemeType.Http,
                BearerFormat = "JWT",
                Scheme = "Bearer"
            });
            options.AddSecurityRequirement(new OpenApiSecurityRequirement
            {
                {
                    new OpenApiSecurityScheme
                    {
                        Reference = new OpenApiReference
                        {
                            Type = ReferenceType.SecurityScheme,
                            Id = "Bearer"
                        }
                    },
                    Array.Empty<string>()});}
```

У лістингу 3.7 описано метод Configure, у якому вказується поведінка сторінки Swagger при доданні токену. Перш за все, на сторінці з’явиться кнопка Authorization (рисунок 3.3), у яку потрібно вставити токен доступу. Після цього до усіх запитів у заголовки буде додаватись токен доступу, що забезпечить виконання навіть тих методів, що захищені авторизацією.

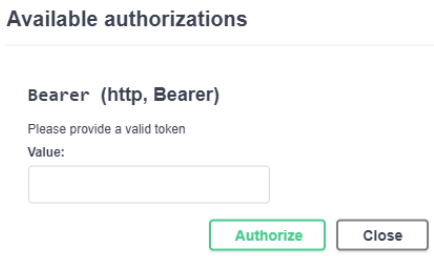


Рисунок 3.3 — Вигляд кнопки Authorize на сторінці Swagger

Після того, як конфігурація проекту завершена, можна приступати до розробки контролерів, моделей, сервісів бізнес-логіки та рівню доступу до даних.

Платформа ASP.Net Core відрізняється від ASP.Net Framework тим, що вона кросплатформена, а отже серверна частина може бути запущена на будь якому серверному середовищі. Тобто веб-застосунок можна захостити на сервері з операційними системами Windows, Mac OS або Linux, і він буде працювати однаково на всіх середовищах.

3.4 Реалізація авторизації у серверній частині веб-застосунку для організації тренувань та дієти

Для забезпечення безпеки зі сторони серверної частини веб додатку важливо додати обмеження на використання методів, які надсилають запити до штучного інтелекту. Оскільки кожен такий запит витрачає токени, які конвертуються в гроші необхідно забезпечити доступ до даних методів лише зареєстрованим користувачам. Таким чином, у випадку, якщо зловмисники заволодіють URL-адресою, за якою виконується метод, який надсилає запит до штучного інтелекту, він не зможе ним скористатись, якщо не додасть токен доступу до заголовків запиту.

3.4.1 Розробка сервісу реєстрації

Користувачу потрібно надати можливість зареєструватись. Лише після створення аккаунту у користувача буде можливість зайти у свій обліковий запис, для отримання токену доступу, який в майбутньому дасть доступ до використання функцій, що захищенні авторизацією. Для цього було описано метод RegisterUserAsync (лістинг 3.8).

Для реєстрації користувачу потрібно ввести базову інформацію про себе, таку як ім’я, адреса електронної пошти, пароль та поле «повторити пароль», щоб бути впевненим що користувач не припустився помилок при складанні свого паролю.

У лістингу 3.8 описано асинхронний метод для реєстрації користувача. Цей метод є методом типу Post, з шляхом «register». Він приймає модель RegisterUserRequest, яка містить усі необхідні для реєстрації поля.

Лістинг 3.8 — Описання методу RegisterUserAsync

```
[HttpPost("register")]
        public async Task<ActionResult>
RegisterUserAsync([FromBody]RegisterUserRequest registerUserRequestModel)
        {
            if (registerUserRequestModel.Password !=
registerUserRequestModel.RepeatPassword)
            {
                return BadRequest("Passwords are not matching");
            }

            var registerResponse = await
_authService.RegisterAsync(registerUserRequestModel);

            if (registerResponse is RegisterUserResponseErrors
registerResponseErrors)
            {
                return BadRequest(registerResponseErrors.RegisterErrors);
            }

            return Ok(registerResponse);
        }
```

З самого початку у методі перевіряється чи поля «Пароль» та «Повторити пароль» співпадають. Якщо ці поля не однакові, метод поверне код статусу 400 BadRequest, з описанням помилки, що поля не співпадають.

Якщо ж перевірка пройде успішно, наступним буде крок виклик методу RegisterAsync, у сервісу авторизації (лістинг 3.9). У даному методі використовується клас UserManager, який є класом з простору імен «Microsoft.AspNetCore.Identity». Тобто цей клас є вбудованим класом для забезпечення авторизації від Microsoft.

Лістинг 3.9 — Приклад методу реєстрації користувача

```
public async Task<RegisterUserResponse> RegisterAsync(RegisterUserRequest
registerUserRequestModel)
        {
            var userToRegister = new User(registerUserRequestModel);
            var result = await _userManager.CreateAsync(userToRegister,
registerUserRequestModel.Password);
            if (!result.Succeeded)
            {
                return new RegisterUserResponseErrors() { RegisterErrors =
result.Errors?.Select(x => x.Description).ToList() };
            }
            return CreateRegisterReponse(userToRegister);}
```

Метод, представлений лістингу 3.9 є асинхронним та приймає модель RegisterUserRequest, яка містить інформацію про користувача. Із моделі запиту створюється об’єкт типу User, щоб можна було зберегти його у базі даних, та виконати реєстрацію.

Після цього, викликається вбудований асинхронний метод CreateAsync, який створює користувача, та зберігає його інформацію у базі даних. Збереження паролю було б не доречним з точки зору безпеки, тому цей метод переведе пароль у хеш, тобто зашифровану послідовність символів, і вже у такому вигляді збереже її у базі даних. Таким чином, коли користувач буде вводити свій пароль при авторизації, хеш паролю буде дешифруватись із набору символів, у ту стрічку, що користувач ввів при реєстрації, та відбудеться порівняння.

Метод CreateAsync повертає результат операції, який зберігається у змінну result, після чого відбувається перевірка чи операція була виконана успішно. Якщо ж під час створення користувача сталась якась помилка, то замість RegisterUserResponse, метод поверне об’єкт нащадку помилку RepisterUserResponseError, яка буде містити в собі помилки, що відбулись при реєстрації. Таким чином, користувач на рівні представлення, побачить що було не вірно введено при реєстрації (рисунок 3.4).

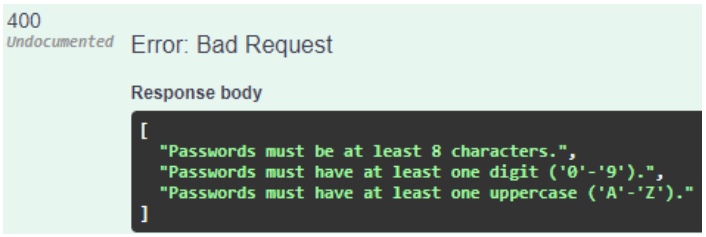


Рисунок 3.4 — Приклад помилок при реєстрації

Якщо ж користувач ввів усю інформацію про себе вірно то метод поверне RegisterUserResponseModel, яка буде містити в собі інформацію про зареєстрованого користувача (рисунок 3.5). Детальніше модуль авторизації описано у додатку Ж.

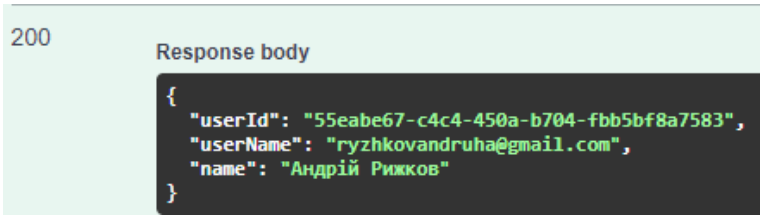


Рисунок 3.5 — Приклад успішної реєстрації користувача

Після успішної реєстрації користувач зможе залогінитись, використовуючи свій логін та пароль.

3.4.2 Розробка сервісу авторизації

Для отримання токену доступу користувачу потрібно авторизуватись, використовуючи логін та пароль, які він вказав під час реєстрації. Для цього потрібно використовувати метод LoginAsync (лістинг 3.10), який можна знайти за URL-адресою «login».

Лістинг 3.10 — Приклад методу LoginAsync у контролері

```
[HttpPost("login")]
        public async Task<IActionResult> LoginAsync([FromBody] LoginRequest
loginRequest)
        {
            var loginResponse = await _authService.LoginAsync(loginRequest.Email,
loginRequest.Password);

            if (loginResponse == null)
            {
                return BadRequest("Email or Password is invalid");
            }

            Response.Cookies.Append("Refresh-Token", loginResponse.RefreshToken, new
CookieOptions() { HttpOnly = true });

            return Ok(loginResponse);
        }
```

Цей метод є асинхронним та приймає як параметр електронну пошту та пароль. Ці поля передаються у метод LoginAsync (лістинг 3.11), який використовує вбудований клас SignInManager із простору імен «Microsoft.AspNetCore.Identity». Якщо метод LoginAsync об’єкту _authService повернув null, це означає що авторизація пройшла з помилками, тобто користувач ввів неправильні данні. В такому випадку метод поверне код статусу 400 Bad Request із описом помилки, що користувач не правильно ввів логін або пароль. Якщо ж авторизація пройшла успішно, у куки браузера буде додано новий запис у вигляді Refresh-Token, тобто токену оновлення. Він використовується для продовження роботи, якщо токен доступу прострочився, але користувач продовжує роботу із додатком у рамках однієї сесії.

Лістинг 3.11 — Приклад методу LoginAsync у сервісі авторизації

```
public async Task<LoginResponse> LoginAsync(string email, string password)
        {
            var user = await _userManager.FindByEmailAsync(email);
            var result = await _signInManager.PasswordSignInAsync(user, password,
false, false);

            if (!result.Succeeded)
            {
                return default;
            }
            var loginResponse = CreateLoginResponse(user);
            var loggedInUser = _dbContext.Users.FirstOrDefault(user => user.Id ==
loginResponse.UserId);

            loggedInUser.RefreshToken = loginResponse.RefreshToken;
            _dbContext.SaveChanges();

            return loginResponse;}
```

У лістингу 3.11 для авторизації використовується метод PasswordSignInAsync об’єкта _signInManager. Цей метод як параметри приймає користувача, його пароль, для порівняння цього паролю із тим, що користувач вказував на етапі реєстрації, та два параметри логічного типу даних. Перший параметр вказує чи повинні куки залишатись після того як браузер закривається, другий — чи повинен акаунт блокуватись у випадку, якщо користувач декілька раз не правильно ввів пароль.

Цей метод повертає результат, який зберігається у змінній result. Якщо результат був неуспішним, метод поверне null.

У випадку ж, якщо логін пройшов успішно, буде створено loginResponse, який містить інформацію про користувача, що увійшов в свій акаунт. Окрім цього буде створено токен оновлення, який буде додано у базу даних і прив’язано за ідентифікатором користувача, що увійшов до свого акаунту. Після успішно пройденої авторизації користувач отримає токен доступу, який буде додано до усіх його майбутніх запитів на серверну частину.

3.4.3 Розробка механізму оновлення токену доступу

У проекті також реалізовано можливість оновлення токену доступу. З точки зору безпеки, час життя токену доступу повинен бути від однієї до кількох годин, щоб використання одного і того самого токену протягом довгого часу не призвело до втручання зловмисників. Але вимагати у користувача вводити облікові данні кожну

годину було б некоректно з точки зору UX дизайну. Саме тому було використано механізм оновлення токену доступу.

Під час авторизації, для користувача створюється токен оновлення (Refresh-Token), і зберігається у базі даних за конкретним користувачем. Коли токен доступу прострочується, клієнт повинен надіслати запит на метод оновлення токену доступу (лістинг 3.12).

Цей метод є Post методом, який можна знайти за URL-адресою «refresh-token» та приймає як параметр токен оновлення. Для валідації токену використовується метод ValidateRefreshToketRequestAsync (лістинг 3.13) об’єкта _tokenService.

Лістинг 3.12 — Приклад методу RefreshTokenAsync

```
[HttpPost("refresh-token")]
        public async Task<IActionResult> RefreshAccessTokenAsync(RefreshTokenRequest
refreshTokenRequest)
        {
            try
            {
                await
_tokenService.ValidateRefreshTokenRequestAsync(refreshTokenRequest);
            }
            catch (KeyNotFoundException knfe)
            {
                return NotFound(knfe.Message);
            }
            catch (UnauthorizedAccessException uae)
            {
                return Unauthorized(uae.Message);
            }
            catch (Exception e)
            {
                return BadRequest(e.Message);
            }

            return
Ok(_tokenService.CreateRefreshTokenResponse(refreshTokenRequest));
        }
```

Якщо цей метод викинув помилку, це означає що токен не валідний, і в залежності від типу помилки різні коди статусу будуть відправлятись на рівень представлення. Так, наприклад, якщо прийшла помилка KeyNotFound, то на рівень представлення буде відправлено код статусу 404 Not Found, що означає що користувач з таким ідентифікатором не був знайдений.

Якщо токен оновлення, що надіслали з клієнта, відповідає токену оновлення, що зберігається у базі даних за користувачем, і валідація пройшла успішно, то метод поверне новий токен доступу для конкретного користувача.

У цьому методі із бази даних отримується користувач за його унікальним ключем. Якщо такого користувача не існує, буде згенеровано виняток KeyNotFoundException з описом помилки, що користувача з таким ідентифікатором не існує. Якщо ж токен оновлення, що надійшов з рівня представлення не відповідає токену доступу, що було отримано із бази даних, буде згенеровано виняток UnauthorizedAccessException з описом, що токен оновлення не правильний.

Лістинг 3.13 — Приклад методу ValidateRefreshTokenRequestAsync

```
public async Task<bool> ValidateRefreshTokenRequestAsync(RefreshTokenRequest
refreshTokenRequest)
        {
            JwtSecurityToken jwtSecurityToken;
            try
            {
                jwtSecurityToken = new
JwtSecurityToken(refreshTokenRequest.RefreshToken);
            }
            catch (Exception)
            {
                throw;
            }
            var user = await _dbContext.Users.FirstOrDefaultAsync(user => user.Id ==
refreshTokenRequest.UserId);
            if (user == null)
            {
                throw new KeyNotFoundException("User with such id is not found");
            }
            if (user.RefreshToken != refreshTokenRequest.RefreshToken)
            {
                throw new UnauthorizedAccessException("Refresh token is wrong");
            }
            if (jwtSecurityToken.ValidTo < DateTime.Now)
            {
                throw new UnauthorizedAccessException("Refresh token is expired");
            }

            return true;
        }
```

У випадку коли токени співпадають, але час життя цього токену пройшов, буде згенеровано виняток з аналогічною помилкою з попереднім прикладом, але опис помилки буде, що токен прострочився, і його більше не можна використовувати. Якщо ж валідація пройшла успішно, метод поверне логічне значення true, а на рівні

контролера буде викликано метод CreateRefreshTokenResponse (лістинг 3.14), який оновить токен доступу.

Лістинг 3.14 — Опис методу CreateRefreshTokenResponse

```
public RefreshTokenResponse CreateRefreshTokenResponse(RefreshTokenRequest
refreshTokenRequest)
        {
            return new RefreshTokenResponse()
            {
                UserId = refreshTokenRequest.UserId,
                AccessToken = CreateToken(false)};}
```

У цьому методі будується RefreshTokenResponse, який оновлює токен, та призначає його конкретному користувачу за його ідентифікатором. Метод CreateToken приймає як параметр логічне значення, яке ідентифікує про те, чи потрібно створити токен оновлення, чи токен доступу. Після чого проініціалізуються усі потрібні поля, такі як видавець токену, користувач, для якого цей токен призначений, час його життя, а також із конфігурації буде отримано таємну фразу, яка буде зашифрована.

Після цього користувач отримає новий токен доступу без необхідності знову вводити свої облікові данні. При цьому токен оновлення не повинен мати нескінченний час життя. Раз у 60 днів токен оновлення прострочується, і користувачу все ж таки доведеться ввести свої облікові данні.

Таким чином, у серверній частині веб-застосунку було реалізовано захист від зовнішнього втручання, не погіршуючи при цьому UX. За рахунок створення окремого токену доступу для кожного користувача, зловмисники не зможуть надсилати велику кількість платних запитів до штучного інтелекту. Тому, для економічної ефективності проекту, користувач повинен буде продивитись рекламу, поки його дієта та система тренувань генерується штучним інтелектом. Таким чином, можна не втрачати кошти на запитах до штучного інтелекту.

4 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ

У веб-застосунку усі зареєстровані користувачі повинні зберігатись у базі даних, для того щоб у будь який момент рівень представлення міг отримати інформацію про користувача. Окрім цього, збереження користувачів у базі даних потрібно для авторизації, оскільки коли при авторизації користувач вводить свої облікові данні, потрібно перевірити чи існує така електронна пошта та пароль у веб-застосунку.

4.1 Конфігурація Entity Framework

Для створення та маніпуляцій над базою даних в проекті ASP.Net Web-Api було обрано технологію Entity Framework. Ця технологія дозволяє працювати із таблицями бази даних як із об’єктами мови програмування C#.

Потрібно налаштувати додаток на використання контексту, який містить колекції об’єктів, що являють собою таблиці із бази даних. Приклад налаштування було приведено у лістингу 2.4. Окрім цього, потрібно налаштувати зберігання користувачів із технології ASP.Net Identity у цій базі даних. Приклад налаштування збереження користувачів наведено у лістингу 4.1.

Лістинг 4.1 — Приклад налаштування збереження користувачів

```
builder.Services.AddIdentity<User, IdentityRole>
    .AddEntityFrameworkStores<ApplicationContext>()
    .AddDefaultTokenProviders()
    .AddSignInManager<SignInManager<User>>()
    .AddUserManager<UserManager<User>>();
```

У лістингу 4.1 використовується метод AddIdentity, для використання технології ASP.Net Identity, що використовується для впровадження авторизації. Після цього викликається метод AddEntityFrameworkStores, в який передається контекст де зберігаються інформація, що будуть перенесені у базу даних. Окрім цього, додаються SignInManager та UserManager, які будуть маніпулювати користувачами, що зберігаються у базі даних.

4.2 Створення моделі даних

Для збереження інформації про користувача необхідно визначити модель даних, яка буде містити усю необхідну інформацію. При використанні технології ASP.Net Identity з'являється можливість використовувати базовий клас IdentityUser, яка містить базові поля, такі як ім'я користувача, електронна пошта, номер телефону та інше. Тому при описанні моделі даних, потрібно наслідувати базовий клас IdentityUser, і у моделі даних User (лістинг 4.2) додати усі необхідні поля, що не передбачені у батьківському класі.

Лістинг 4.2 — Визначення моделі користувача

```
public class User : IdentityUser
    {
        public string Name { get; set; }
        public string RefreshToken { get; set; }

        public DietData DietData { get; set; }
        public FoodDetails FoodDetails { get; set; }
    }
```

Окрім полів, що визначені в базовому класі, у веб-застосунку важливо мати інформацію про ім'я та прізвище користувача, зберігати у базі даних токен оновлення, для можливості оновити токен доступу, а також необхідно зберегти дані для побудови дієти користувача (лістинг 4.3) та його харчові звички (лістинг 4.4).

Лістинг 4.3 — Модель даних з інформацією для побудови дієти

```
public class DietData
    {
        public int Id { get; set; }
        public int Age { get; set; }
        public int Height { get; set; }
        public int Weight { get; set; }
        public Gender Gender { get; set; }
        public DietGoal Goal { get; set; }
        public string PhysicalActivity { get; set; }

        public string UserId { get; set; }
        public User User { get; set; }
    }
```

Для того, щоб штучний інтелект зміг успішно побудувати дієту для користувача, йому важливо прийняти до уваги багато факторів. Уся необхідна інформація описана у моделі даних, що представлена у лістингу 4.3. Для побудови

дієти важливо вказати вік, зріст, стать, ціль дієти та фізичну активність користувача. Окрім цього у моделі описано поле Id, яке автоматично згенерується при записі об’єкту до бази даних.

Також у моделі описані поля UserId та User. Вони будуть використовуватись як вторинний ключ, для зв’язку із таблицею користувачів, оскільки кожні такі данні повинні бути прив’язані за конкретним користувачем. Технологія EntityFramework автоматично співвіднесе вторинний ключ із таблицею користувачів, використовуючи посилальну властивість User.

Лістинг 4.4 — Модель даних харчових звичок користувача

```
public class FoodDetails
    {
        public int Id { get; set; }

        public string Allergies { get; set; }
        public string FoodRestrictions { get; set; }
        public string FoodPreferences { get; set; }
        public string DislikeFood { get; set; }

        public string UserId { get; set; }
        public User User { get; set; }
}
```

У лістингу 4.4 представлена уся необхідна інформація, про харчові звички користувача, яку потрібно враховувати при побудові дієти. Так наприклад, потрібно попередити штучний інтелект про наявність алергії у користувача, його обмеження у їжі, страви яким користувач віддає перевагу, та які не любить. Окрім цього цю таблицю також потрібно зв’язати із таблицею користувачів за допомогою вторинного ключа.

4.3 Опис контексту даних у технології Entity Framework

Клас контексту містить колекції об’єктів типу DbSet (лістинг 4.5), що сигналізує технології EntityFramework, що у базі даних, що буде підключена до проекту, необхідно створити таблицю з таким самим іменем і полями, що визначені у мові програмування C#.

Лістинг 4.5 — Приклад створення колекцій DbSet

```
public class ApplicationContext: IdentityDbContext<User>
    {
        public DbSet<DietData> DietData { get; set; }
        public DbSet<FoodDetails> FoodDetails { get; set; }}
```

З лістингу вище видно, що у контексті даних визначено 2 колекції DbSet. Одна із них типізована класом DietData, інша класом FoodDetails. Це моделі даних для збереження інформації для побудови запиту про дієту та харчових звичках користувача відповідно.

Але у контексті не визначено окремої колекції DbSet для зберігання користувача. Все тому що, клас контексту наслідується від базового класу IdentityDbContext, типізований моделлю даних User. Це означає, що колекція для збереження користувачів буде знаходитись на рівень вище, а саме у батьківському класі, але клас нащадок може працювати із нею. Тому не потрібно явно визначати ще одну колекцію, для зберігання моделі користувачів User.

Окрім цього, якщо потрібно явно вказати тип зв’язку між пов’язаними таблицями, необхідно перевизначити метод OnModelCreating (лістинг 4.6), всередині якого необхідно прописати усі зв’язки між таблицями, та вказати вторинний ключ.

Лістинг 4.6 — Перевизначений метод OnModelCreating

```
protected override void OnModelCreating(ModelBuilder modelBuilder){
            modelBuilder.Entity<User>()
                .HasOne(u => u.DietData)
                .WithOne(cd => cd.User)
                .HasForeignKey<DietData>(cd => cd.UserId);
            modelBuilder.Entity<User>()
                .HasOne(u => u.FoodDetails)
                .WithOne(dd => dd.User)
                .HasForeignKey<FoodDetails>(dd => dd.UserId);}
```

У коді, описаному вище, вказується, що між сутністю користувача та сутностями, які зберігають інформацію про дієту та харчові звички користувача, необхідно визначити зв’язок один до багатьох із вторинним ключем UserId.

4.4 Оптимізація запитів до бази даних

Оптимізація запитів до бази даних за допомогою технології Entity Framework Core містить ряд стратегій та технік для зменшення кількості та швидкості виконання запитів, що виконуються в контексті Entity Framework.

Запити Eager Loading — це техніка завантаження пов’язаних об’єктів одночасно з основним запитом до бази даних, щоб уникнути проблеми N+1 запитів і зменшити кількість звернень до бази даних. Проблема N+1 запитів (N+1 Query Problem) — це ситуація, коли для отримання N записів головної сутності потрібно виконати один запит, а потім ще N окремих запитів для отримання пов'язаних даних для кожного з цих N записів. Використання методу «Include», що використовується для для завантаження пов’язаних об’єктів разом з основним запитом представлено на лістингу 4.7.

Лістинг 4.7 — Приклад використання методу Include

```
var dietDataWithUser = _applicationContext.DietData
                    .Include(d => d.User)
                    .Where(d => d.UserId == userId);
```

У прикладі вище завдяки методу Include, дані про користувачів (User) будуть завантажені разом з даними про дієту (DietData) у єдиному запиті до бази даних, замість використання окремих запитів для кожного запису дієти.

Окрім цього, щоб зменшити обсяг даних, що передаються із бази даних, можна використовувати метод Select (лістинг 4.8), завдяки якому можна визначити набір даних, які будуть вибрані із таблиці бази даних, замість вибору усього об’єкту.

Лістинг 4.8 — Приклад використання методу Select

```
var negativeFoodDetails = _applicationContext.FoodDetails
                    .Where(fd => fd.UserId == userId)
                    .Select(fd => new { fd.FoodRestrictions, fd.DislikeFood,
fd.Allergies})
                    .ToList();
```

У прикладі, описаному вище, відбувається вибірка лише негативних деталей про харчові звички користувача. Замість того, щоб отримувати із бази даних весь об’єкт, в якому міститься комплексна інформація, яку вказав користувач, серверна

частина вибере із бази даних лише інформацію про обмеження в їжі, їжу, що не подобається, та алергії для конкретного користувача.

Ще одним способом оптимізації запитів до бази даних є запити «Deferred Execution» — це особливість Entity Framework і інших LINQ-подібних запитів в .NET, яка полягає в тому, що запит не виконується негайно після його створення, а відкладається до того моменту, коли фактично потрібні дані. Це дозволяє оптимізувати та уникнути надмірного витрати ресурсів. Приклад таких запитів наведено в лістингу 4.9.

Лістинг 4.9 — Приклад запиту Deferred Execution

```
var dietData = _applicationContext.DietData.FirstOrDefault(c => c.UserId == userId);
```

Запит, що описаний вище буде виконаний лише у той момент, коли дані будуть потрібні для використання. Таким чином, лише коли у коді відбудеться звернення до будь якої змінної об’єкту dietData виконається запит до бази даних для отримання інформації, а серверна частина продовжить роботу із записом бази даних як з об’єктом типу DietData. Усі методи доступу до даних описані у додатку Е.

Таким чином, за допомогою методів, наведених вище, у серверній частині веб-застосунку для організації тренувань та дієти, було проведено оптимізацію запитів до бази даних.

4.5 Тестування модулю інтеграції зі штучним інтелектом ChatGpt

Проведемо тестування сервісу інтеграції зі штучним інтелектом, щоб перевірити працездатність розробленого компоненту.

Для того, щоб відправити запит до штучного інтелекту необхідно створити користувача (рисунок 4.1), щоб отримати токен доступу за яким буде виконаний метод.

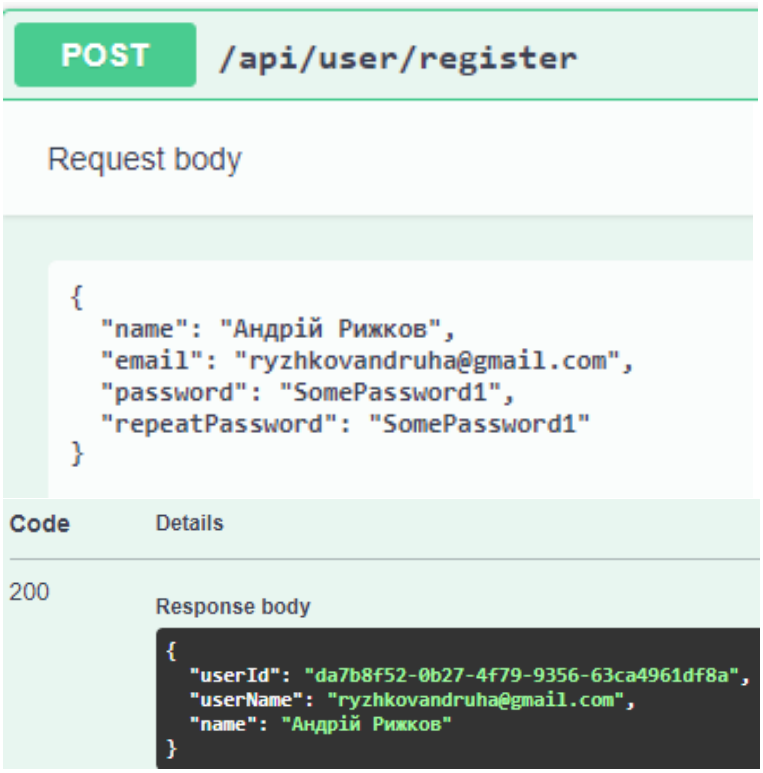


Рисунок 4.1 — Приклад реєстрації користувача

Після того як користувач зареєстрований, можна увійти до облікового запису за логіном та паролем, що були вказані під час реєстрації. Коли користувач увійде в обліковий запис, він отримає унікальний токен доступу, за допомогою якого можна буде вказати додаткову інформацію. Приклад входу в обліковий запис представлено на рисунку 4.2.

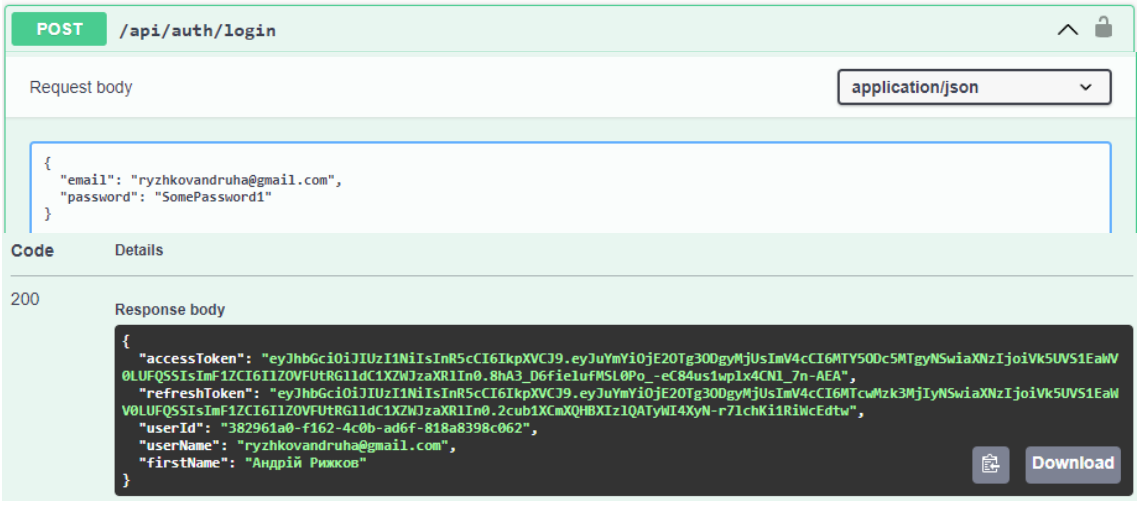


Рисунок 4.2 — Приклад логіну користувача

Після отриманого токену доступу, його потрібно додати до заголовків наступних запитів, щоб можна було використовувати ендпоінти, що захищені авторизацією.

Наступним кроком буде додавання інформації, яка необхідна для побудови дієти (рисунок 4.3). Для збереження інформації для конкретного користувача, у тіло запиту також потрібно передати ідентифікатор користувача, який клієнт отримав при реєстрації.

Окрім цього, за бажанням користувача, він може додати детальну інформацію про харчові звички, яка буде використовуватись при побудові запиту до штучного інтелекту.

Коли усі необхідні данні про користувача зберігаються у базі даних, можна використовувати ендпоінт для побудови дієти (лістинг 4.10). Він отримає інформацію із бази даних, і на її основі побудує запит, що буде відправлений до OpenAI API.

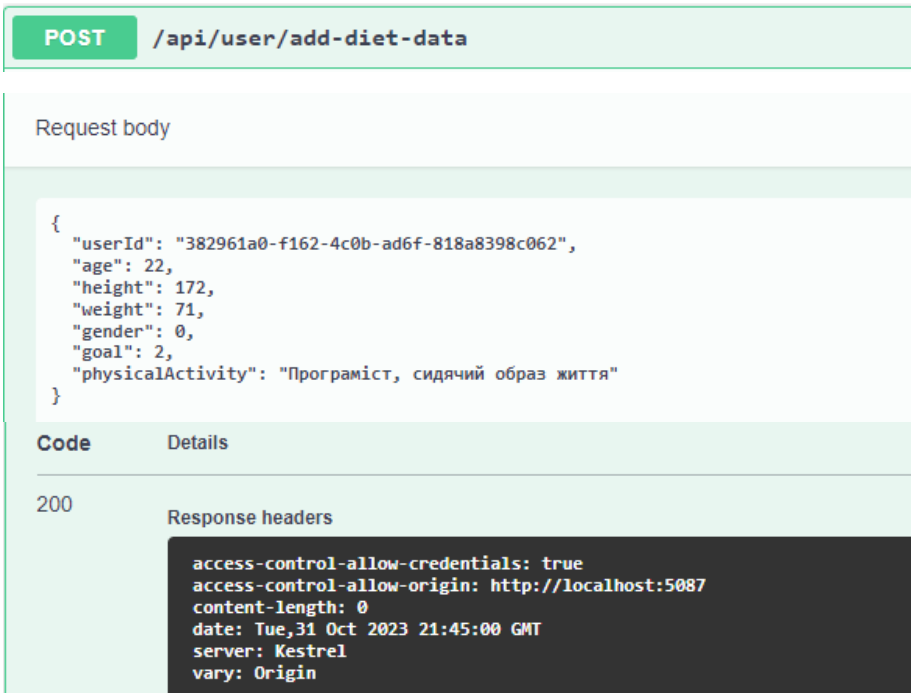


Рисунок 4.3 — Додавання інформації про користувача

Штучний інтелект обробить запит і у відповіді відправить дієту у вигляді тексту, а також деяку службову інформацію про виконання запиту, таку як:

— index — індекс повідомлення від штучного інтелекту;

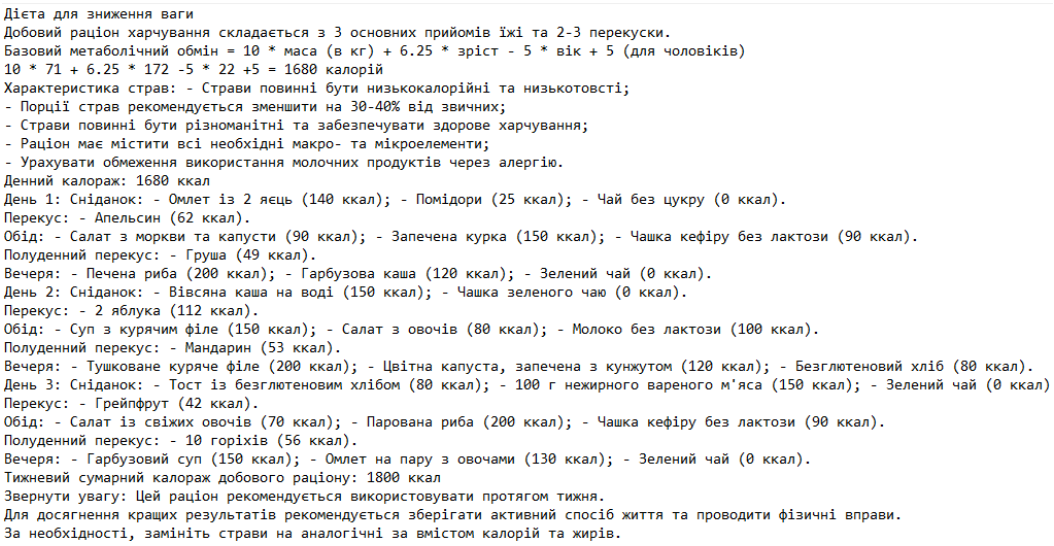
— message — відповідь чату GPT у вигляді тексту;

— finish_reason — причина закінчення генерації відповіді;

— usage — об’єкт, який містить в собі інформацію про кількість використаних токенів на генерацію відповіді.

З лістингу 4.10 видно, що штучний інтелект надіслав відповідь на запит, в якій розписана дієта користувача. Були проведені обрахунки за формулами метаболічного обміну людини, щоб врахувати кількість калорій, які повинен вживати користувач в день, для досягнення мети. Завдяки цьому користувач не лише бачить денний раціон, який він може споживати, але й розуміє принцип дієти і денну норму калорій.

Лістинг 4.10 — Скріншот відповіді штучного інтелекту на запит генерації дієти



Штучний інтелект врахував вподобання морепродуктів, алергію на горіхи та непереносимість лактози користувача. Згідно з цими харчовими звичками ChatGpt побудував дієту на тиждень, з розписаною денною нормою калорій.

Окрім цього, штучний інтелект повертає текст із символами форматування, тому на рівні представлення розробники зможуть відобразити дієту згідно UI/UX дизайну. Аналогічно з цим штучний інтелект може розписати програму тренувань для користувача.

Також важливо порівняти швидкість обробки запиту безкоштовною версією штучного інтелекту та платною версією, яка використовується при запиті на API. Для

веб-застосунку використовується версія штучного інтелекту 3.5-turbo. Приставка «turbo» означає, що він повинен надавати відповідь швидше, ніж безкоштовна версія штучного інтелекту. Окрім цього, версія 3.5-turbo містить оновлену інформацію, та кращу обробку запитів, за рахунок чого, штучний інтелект зможе краще описати відповідь на користувацький запит.

Після проведення тесту із запущеним таймером, серверна частина отримала відповідь за 1 хвилину, 34 секунди на 10 мілісекунд. В свою чергу безкоштовна версія, на аналогічний запит сформувала відповідь 1 хвилину 41 секунду та 8 мілісекунд. Окрім цього, відповідь, що отримала серверна частина була більш обширна, та містила в собі розписану дієту у вигляді документу, з детальним планом харчування користувача та основними принципами денних калорій. Відповідь безкоштовної версії, почалась зі слів «На жаль, я не можу надати індивідуально розроблену дієту з точними кількостями калорій та стравами, оскільки я не маю можливості виміряти конкретні кількості інгредієнтів чи провести детальну консультацію для врахування усіх особистих харчових уподобань та індивідуальних особливостей. Однак, я можу запропонувати загальні рекомендації з харчування, що можуть бути корисними», що означає, що відповідь версії 3.5-turbo буде інформативніша та корисніша для користувачів.

Також слід враховувати, що час на побудову запиту користувачем, з надаванням усієї необхідної інформації буде тривати довше, адже дослідження усієї необхідної інформації для побудови дієт та системи тренувань займе значно більше часу, ніж заповнення простої форми, що представлена на рівні представлення веб-застосунку. Слід зазначити, що час генерації відповіді безкоштовною та платною версією може відрізнятись, залежно від навантаження на сервери OpenAI.

Серверна частина протестована та працює успішно. Під час тестування не було виявлено критичних помилок, які заважають роботі веб-застосунку. Модуль інтеграції зі штучним інтелектом описує дієти для користувачів, враховуючи їх антропометричні данні та харчові звички. Обробка запиту платною версією штучного інтелекту відбувається швидше, ніж безкоштовною версією, а відповіді більш інформативні.

5 ЕКОНОМІЧНА ЧАСТИНА

Ефективне впровадження науково-технічної розробки стає можливим, якщо вона відповідає поточним вимогам науково-технічного прогресу та враховує економічні аспекти. Надання оцінки економічної ефективності отриманих результатів науково-дослідної роботи є важливою частиною цього процесу.

Магістерська робота, що присвячена розробці та дослідженню «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина», віднесена до науково-технічних робіт, спрямованих на введення на ринок. Рішення про комерціалізацію розробки може бути прийняте протягом самої роботи, дозволяючи реалізувати можливість виведення її на ринок. Цей напрямок розглядається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Однак для успішного втілення цього процесу важливо знайти зацікавленого інвестора, який був би зацікавлений у реалізації цього проекту, і переконати його в обґрунтованості таких інвестицій.

Для цього визначені наступні етапи виконання робіт:

— проведено комерційний аудит науково-технічної розробки, що включає в себе визначення науково-технічного рівня та комерційного потенціалу;

— розраховані витрати на реалізацію науково-технічної розробки;

— проведено розрахунок економічної ефективності науково-технічної розробки в разі її впровадження та комерціалізації потенційним інвестором, а також обґрунтовано економічну доцільність комерціалізації для інвестора.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» є оцінювання науково-технічного рівня та

рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1

Таблиця 5.1 — Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведені до таблиці 5.2. Для опитування було залучені експерти: к.пед.н. доц. кафедри Войцеховська О.В., к.пед.н., доц. Добровольська Н.В., к.т.н, доц. Тарновський М.Г.

Таблиця 5.2 — Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Войцеховська О.В.	Добровольська Н.В.	Тарновський М.Г.
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	4	3	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	$СБ_1$=43	$СБ_2$=42	$СБ_3$=44
Середньоарифметична сума балів $СБ_c$	(43+42+44)/3 = 43		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки

Продовження таблиці 5.3

41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» становить 43 бали, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

Результатом магістерської кваліфікаційної роботи «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» — веб-застосунок для надання користувачам дієт та програм тренувань під їх індивідуальні потреби.

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою:

$$q_i = \frac{P_i}{P_{базі}}. \qquad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за i-м параметром;

P_i — значення i-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в

таблиці 5.4.

Таблиця 5.4 — Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Швидкодія, %	90	94	1,02	20
SEO-оптимізація, %	92	95	1,03	30
SSL-шифрування, %	96	93	1,03	30
Адаптивність, %	90	95	1,06	20

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам; 0 — не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою**:**

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \tag{5.2}$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за i-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \tag{5.3}$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник i-го параметра;

α_i — вагомість i-го параметричного показника, $\sum_{i=1}^{n}\alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{тп} = 1{,}02\cdot 0{,}2+1{,}03\cdot 0{,}3+1{,}03\cdot 0{,}3 +1{,}06\cdot 0{,}2 = 0{,}93.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i , \qquad (5.4)$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр i-го виду;

β_i — частка i-го економічного параметра, $\sum_{i=1}^{m}\beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.
Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0{,}76 \cdot 0{,}5 + 0{,}84 \cdot 0{,}5 = 0{,}80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$К_{ІНТ} = 1 \cdot 0{,}93 / 0{,}80 = 1{,}16.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці [19].

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \tag{5.6}$$

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 42000 \cdot 10 / 21 = 19091 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.5.

Таблиця 5.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	42000	1909,1	10	19091
Інженер-програміст	39000	1772,7	55	97500
Всього				116591

Витрати на основну заробітну плату робітників [20] ($З_р$) за відповідними найменуваннями робіт НДР на тему «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б);

K_c — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p — середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ — тривалість зміни, год.

$$C_1 = 6500{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8 \text{ грн.}$$

$$З_{р1} = 65{,}8 \cdot 2 = 131{,}6 \text{ грн.}$$

Результати приведено в таблиці 5.6

Таблиця 5.6 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	2	1	65,8	131,6
2.Налагоджувальні	10	2	72,4	723,8
3.Випробувальні	2	4	98,7	197,4
Всього				1052,9

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (5.9)$$

де $Н_{дод}$ — норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_{дод} = (116591 + 1052{,}9) \cdot 11 / 100\% = 12940{,}81 \text{ грн.}$$

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ — норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (116591+ 1052,9+12940,81) · 22 / 100% = 28728,61 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина».

Витрати на матеріали ($М$) (таблиця 5.7), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \quad (5.11)$$

де $Н_j$ — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу j-го найменування, грн/кг;

$К_j$ — коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів *j*-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А 4	146	1	146
Ручка	14	1	14
Диск оптичний OPTIMA CD	15	1	15
Flesh-пам'ять GOODRAM 64 C10A	410	1	410
Всього			585
З врахуванням коефіцієнта транспортування			643,5

5.3.4 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації [21] за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.12)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = (45000 \cdot 1) / (2 \cdot 12) = 1875 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.8.

Таблиця 5.8– Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	45000	2	1	1875,00
Приміщення лабораторії	190000	20	1	791,67
Всього				2666,67

5.3.5 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.13)$$

де W_{yi} — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,5 грн;

$K_{вni}$ — коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$;

$$B_e = 0,25 \cdot 250,0 \cdot 7,5 \cdot 0,5 / 0,8 = 292,97 \text{ грн.}$$

5.3.6 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру,

аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.14)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$$В_{св} = (116591 + 1052{,}9) \cdot 20 / 100\% = 23528{,}75 \text{ грн.}$$

5.3.7 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (116591 + 1052{,}9) \cdot 50 / 100\% = 58821{,}88 \text{ грн.}$$

5.3.8 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.16)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = (116591 + 1052{,}9) \cdot 100 / 100\% = 117643{,}77 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \qquad (5.17)$$

$$В_{заг} = 116591+1052{,}9+12940{,}81+28728{,}61+643{,}5+2666{,}67+292{,}97+23528{,}75+58821{,}88+117643{,}77 = 363004{,}48 \text{ грн.}$$

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 363004,48 / 0,9 = 403338,31 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN — збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа;

$Ц_б$ — вартість послуги у році до впровадження інформаційної системи, приймемо 2000,00 грн;

$\pm\Delta Ц_о$ — зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від

можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.20)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ=40%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ=*18%*;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1 \cdot 500 + 2000 \cdot 1500) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 640684{,}79 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1 \cdot 500 + 2000 \cdot (1500 + 1200)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 1153578{,}9 \text{грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1 \cdot 500 + 2000 \cdot (1500 + 1200 + 850)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 1516585{,}2 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.21)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 640684,79/(1+0,18)^1+1153578,9 /(1+0,18)^2+1516585,2 /(1+0,18)^3=$$
$$= 2216736,01 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.22)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 403338,31 грн.

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 403338,31 = 806676,61 \text{ грн.}$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.23}$$

де $ПП$ — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2216736,01 грн;

PV — теперішня вартість початкових інвестицій, 806676,61 грн;

$E_{абс} = ПП - PV$ = 2216736,01 - 806676,61 = 1410059,39 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.24}$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1410059,39 /806676,61)^{1/3} — 1 = 0,65.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f\,, \tag{5.25}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25;

$\tau_{мін}$= 0,1+0,25 = 0,35 < 0,65 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\,, \tag{5.26}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 0{,}65 = 1{,}5 \text{ р.}$$

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина» становить 43 бали, що, свідчить про

комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,16 рази.

Також термін окупності становить 1,5 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина».

ВИСНОВКИ

У комплексній магістерській кваліфікаційній роботі було спроектовано та реалізовано серверну частину веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту мовою C#.

Проаналізовано сучасні технології для зв’язку серверної частини веб-додатку із базою даних, реалізації авторизації для впровадження захисту серверної частини від зловмисників та методи інтеграції веб-застосунку з штучним інтелектом. За результатами проведеного аналізу було обрано стек технологій. Для розробки серверної частини використано Web-API. Для зв’язку серверної частини веб-застосунку із базою даних було використано Entity Framework. Авторизацію забезпечено за допомогою JWT Tokens.

Вдосконалено метод інтеграції веб-застосунку з штучним інтелектом ChatGPT 3.5-turbo за рахунок побудови та надсилання запиту до OpenAI API, сформованого на основі визначених в роботі вимог, що дало можливість покращити User Experience при використанні серверної частини веб-застосунку, а також зменшити час обробки запиту штучним інтелектом та пришвидшити роботу користувача зі штучним інтелектом.

В роботі спроектовано багаторівневу архітектуру серверної частини, що дозволило розділити різні функціональні можливості у різних модулях. Програмна реалізація серверної частини веб-застосунку для організації тренувань та дієти виконана на платформі ASP.Net Web-Api, з дотриманням принципів об’єктно-орієнтованого програмування та принципів SOLID.

Також в проекті реалізовано механізм авторизації на JWT-Токенах, що дозволило захистити методи для інтеграції з штучним інтелектом від зловмисників та дало можливість тільки авторизованим користувачам використовувати штучний інтелект для побудови тренувань та дієти.

Спроектовано базу даних MSSQL, в якій зберігаються персональні дані користувача, які використовуються для побудови дієти та системи тренувань штучним інтелектом. База даних спроектована таким чином, щоб користувач мав

можливість зберігати дієти та системи тренувань, які йому сподобались. Запити до бази даних оптимізовані за рахунок використання техніки Eager Loading та запитів Deferred Execution.

Проведене тестування серверної частини показало відсутність критичних помилок, стабільність роботи серверної частини та правильність виконання заданих функцій. База даних зберігає необхідну інформацію, а серверна частина може швидко отримати необхідну інформацію про користувача. Передбачена можливість в подальшому перенесення веб-застосунку в хмарне середовище, а також розширення функціональних можливостей при взаємодії зі штучним інтелектом.

В роботі проведений економічний аналіз доцільності розробки. Економічний ефект складає 0.65 грн. Термін окупності складає 1,5 роки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аналіз архітектури розподіленого слабкозв’язного серверного додатку інформаційної системи / Войцеховська О. В., Рижков А. К. //Матеріали LI наукової-технічної конференції підрозділів ВНТУ, 2022 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/allfitki/all-fitki-2022/paper/view/15103/12731

2. Аналіз методів авторизації при проектуванні серверної частини веб-додатку / Войцеховська О. В., Городецька О.С., Рижков А. К.// Матеріали міжнародної науково-практичної інтернет-конференції “Електронні інформаційні ресурси: створення, використання, доступ”, 2023 р. Режим доступу: https://drive.google.com/file/d/1oVmxS3W_sEQPjes9S9AzWDaJxDxi6I0X/view

3. Understanding Multilayer Architecture [електронний ресурс]: https://www.c-sharpcorner.com/UploadFile/1492b1/understanding-multilayered-architecture-in-net/

4. A Multi-Layer Back-End Application Architecture in .NET Core [електронний ресурс]: https://hamzaak.medium.com/a-multi-layer-back-end-application-architecture-in-net-core-c08898f2427e

5. What is Entity Framework [електронний ресурс]: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

6. Introduction to Entity Framework [електронний ресурс]: https://www.partech.nl/nl/publicaties/2020/11/introduction-to-entity-framework

7. ADO.NET Overview [електронний ресурс]: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview

8. Overview of the advantages of ADO.Net [електронний ресурс]: https://conf.ztu.edu.ua/wp-content/uploads/2021/06/211.pdf

9. Authentication and Authorization in Web API [електронний ресурс]: https://dotnettutorials.net/lesson/authentication-and-authorization-in-web-api/

10. Authentication vs Authorization [електронний ресурс]: https://frontegg.com/blog/authentication-vs-authorization#:~:text=Types%20of%20authorization%20include%20discretionary,being%20used%20to%20implement%20them

11. Introduction to authorization in ASP.NET Core [електронний ресурс]: https://learn.microsoft.com/en-us/aspnet/core/security/authorization/introduction?view=aspnetcore-7.0

12. Cookie-based vs Cookieless authorization [електронний ресурс]: https://www.loginradius.com/blog/engineering/cookie-based-vs-cookieless-authentication/

13. Using HTTP cookies [електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

14. Introduction to JSON Web Tokens [електронний ресурс]: https://jwt.io/introduction

15. JSON Web Tokens [електронний ресурс]: https://auth0.com/docs/secure/tokens/json-web-tokens

16. JWT vs Cookie: Why Comparing the Two Is Misleading [електронний ресурс]: https://jerrynsh.com/all-to-know-about-auth-and-cookies/#:~:text=Neither%20JWT%20nor%20Cookie%20are,within%20your%20browser's%20Cookies%20storage.

17. Prompt engineering [електронний ресурс]: https://platform.openai.com/docs/guides/prompt-engineering

18. Embeddings — OpenAI API [електронний ресурс]: https://platform.openai.com/docs/guides/embeddings

19. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.

20. Витрати на оплату праці як складова собівартості продукту [електронний ресурс] https://osvita.ua/vnz/reports/econom_pidpr/21918/

21. Терміни амортизації основних засобів у податковому обліку [електронний ресурс] https://if.tax.gov.ua/media-ark/news-ark/437911.html

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О. Д. Азаров
“ ” 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання комплексної магістерської кваліфікаційної роботи
«Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина»
08-54.КМКР.051.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ
Войцеховська О. В.
Студент групи 1КІ-22м
Рижков А. К.

Вінниця 2023

1 Підстава для використання КМКР

1.1 Актуальність розробки полягає у вдосконаленні методу інтеграції серверної частини веб-застосунку з штучним інтелектом, шляхом побудови модифікованого запиту за користувача, та надсилання запиту на обробку штучним інтелектом.

1.2 наказ про затвердження теми кваліфікаційної роботи.

2 Мета і призначення КМКР

— метою комплексної магістерської кваліфікаційної роботи є розширення функціональних можливостей серверного додатку, завдяки інтеграції зі штучним інтелектом, що дасть можливість покращити User Experience (UX) та збільшити швидкодію обробки запитів штучним інтелектом;

— призначення розробки — програмна реалізація покращеного методу інтеграції з штучним інтелектом, що включає побудову модифікованого запиту враховуючи інформацію про користувача а також надсилання та обробку відповіді від штучного інтелекту.

3 Вихідні данні для виконання КМКР

3.1 Проведення аналізу технологій для реалізації серверної частини.

3.2 Аналіз та вдосконалення методів інтеграції штучного інтелекту з веб-застосунком.

3.3 Розробка серверної частини веб-застосунку.

3.4 Проетування бази даних веб-застосунку.

3.5 Виконання розрахунків для доведення доцільності нової розробки.

Середовище розробки Visual Studio для платформи ASP.NET Core, платформа Web-API.

4 Технічні вимоги до виконання КМКР

Основними вимогами до виконання КМКР є:

— наявність серверної частини, яка буде інтегруватись з штучним інтелектом;

— наявність бази даних для збереження інформації про користувача.

5 Етапи КМКР та очікувані результати

Робота виконується за п’ять етапів, що наведені в таблиці А.1.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного стану інтеграцій веб-застосунків з штучним інтелектом	04.10.2022	15.10.2022	Аналітичний огляд літературних джерел
2	Побудова модифікованих запитів до штучного інтелекту для опису системи тренувань та дієти	16.10.2022	04.11.2022	Структурні моделі, 2 розділ
3	Розробка серверної частини, яка буде зв'язана з базою даних та містити механізм авторизації	05.11.2022	30.11.2022	3 і 4 розділи
4	Підготовка економічної частини	30.11.2022	03.12.2022	5 розділ
5	Оформлення пояснювальної записки, графічного матеріалу і/або презентації	04.12.2022	18.12.2022	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту КМКР

До захисту КМКР подаються: пояснювальна записка КМКР, графічні і ілюстративні матеріали, протокол попереднього захисту КМКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи проходження перевірки на плагіат, анотації до КМКР українською та іноземною мовами, нормоконтроль про відповідність оформлення КМКР діючим вимогам.

7 Порядок контролю виконання та захисту КМКР

Виконання етапів графічної та розрахункової документації КМКР контролюється науковим керівником згідно зі встановленими термінами. Захист КМКР відбувається на засіданні екзаменаційної комісії, затверджено наказом ректора.

8 Вимоги до оформлення та проядок виконання КМКР

При оформлюванні КМКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

ДОДАТОК Б

Структурна схема серверної частини веб-застосунку

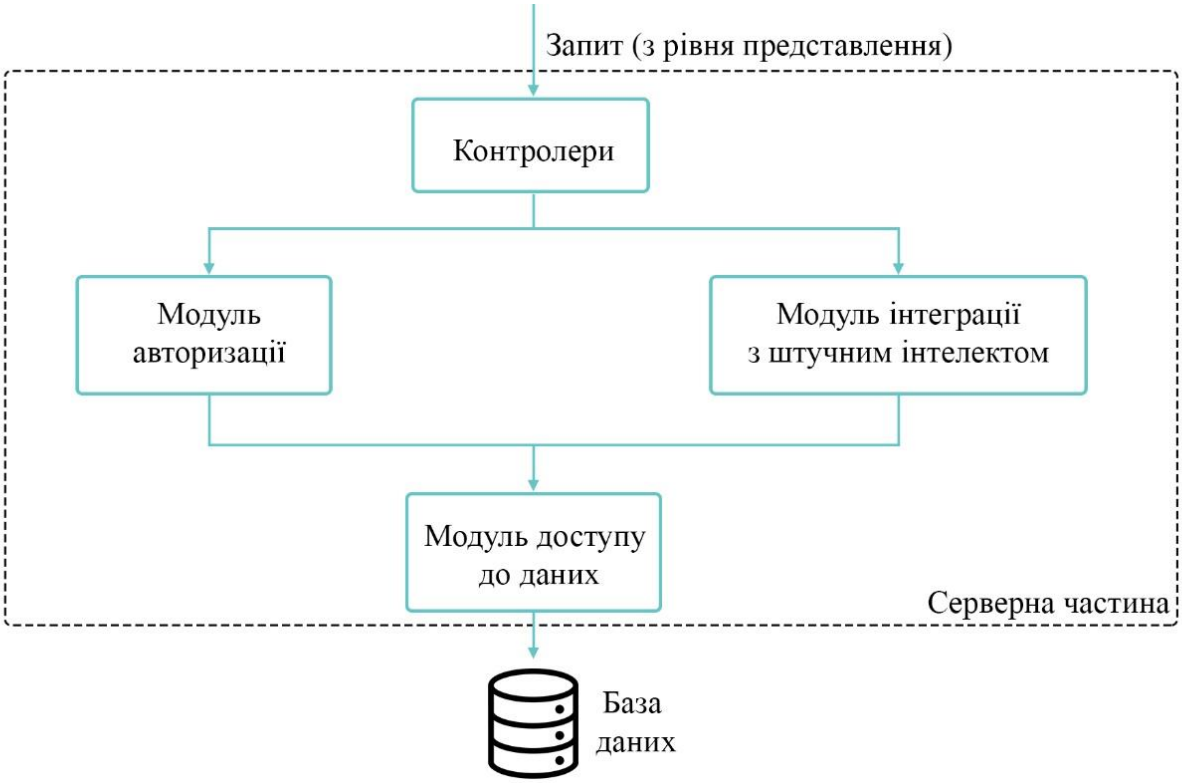


Рисунок Б.1 — Структурна схема серверної частини веб-застосунку

ДОДАТОК В

Діаграма класів компоненту інтеграції зі штучним інтелектом

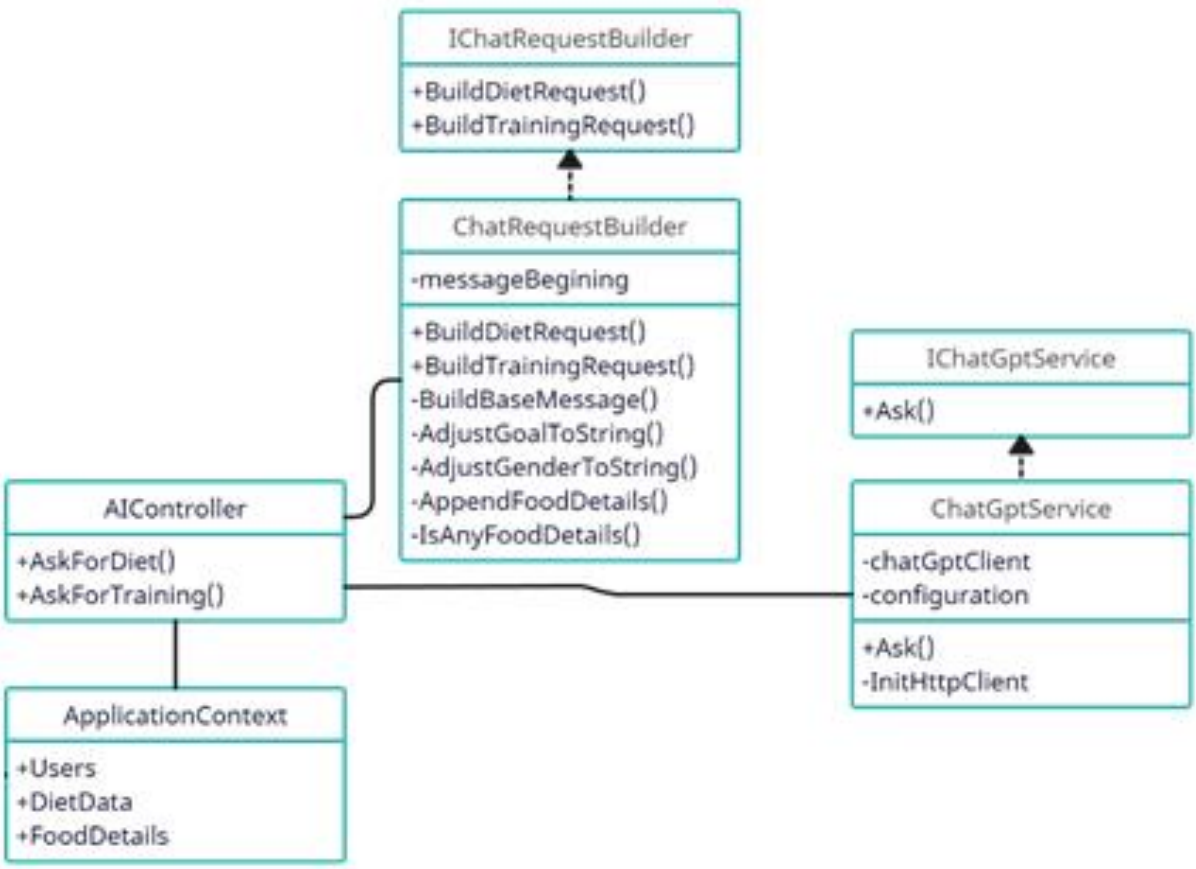


Рисунок В.1 — Діаграма класів компоненту інтеграції зі штучним інтелектом

ДОДАТОК Г

UML-діаграма діяльності модулю інтеграції веб-застосунку зі штучним інтелектом

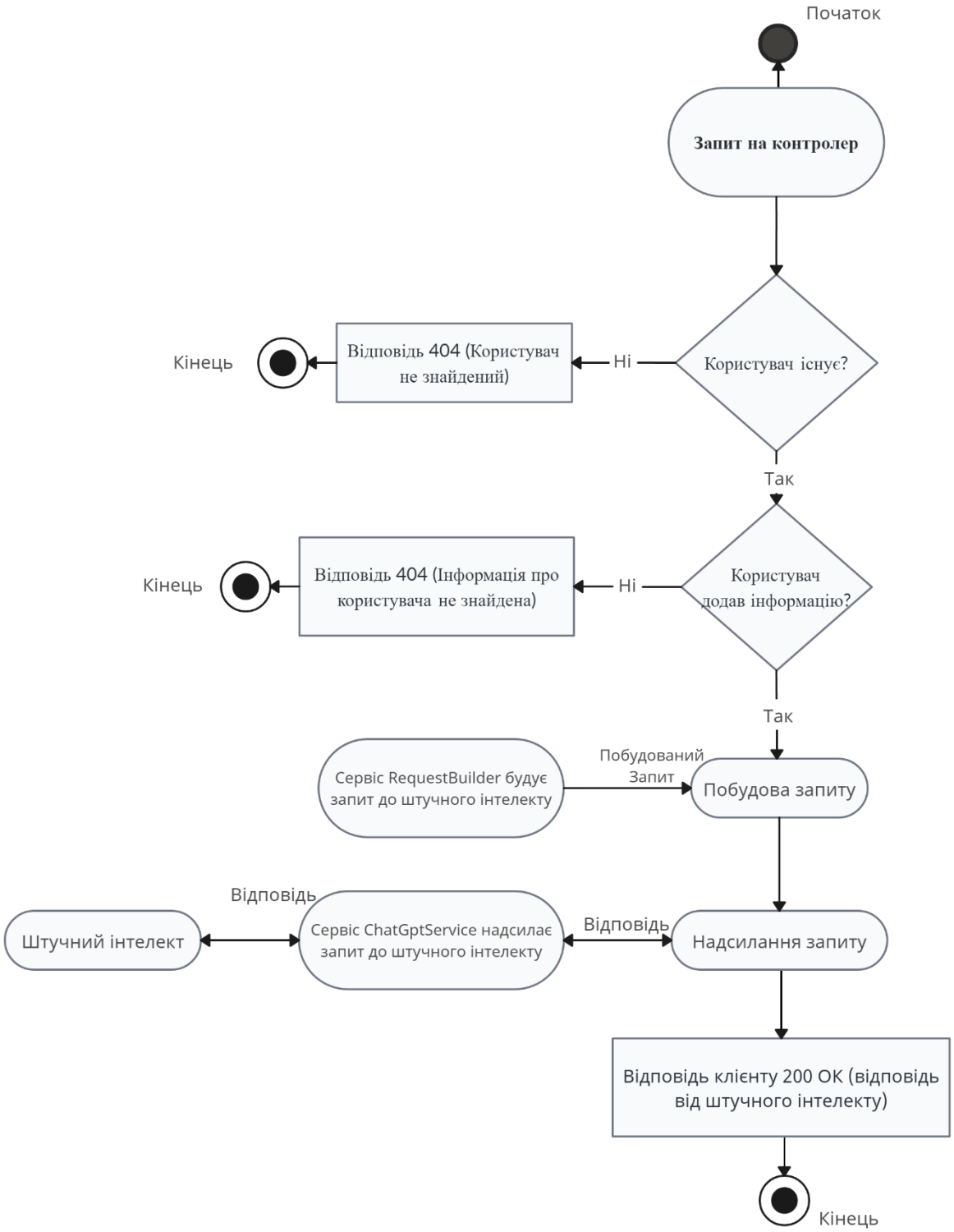


Рисунок Г.1 — UML-діаграма діяльності модулю інтеграції веб-застосунку зі штучним інтелектом

ДОДАТОК Д

UML-діаграма діяльності серверної частини веб-застосунку зі штучним інтелектом

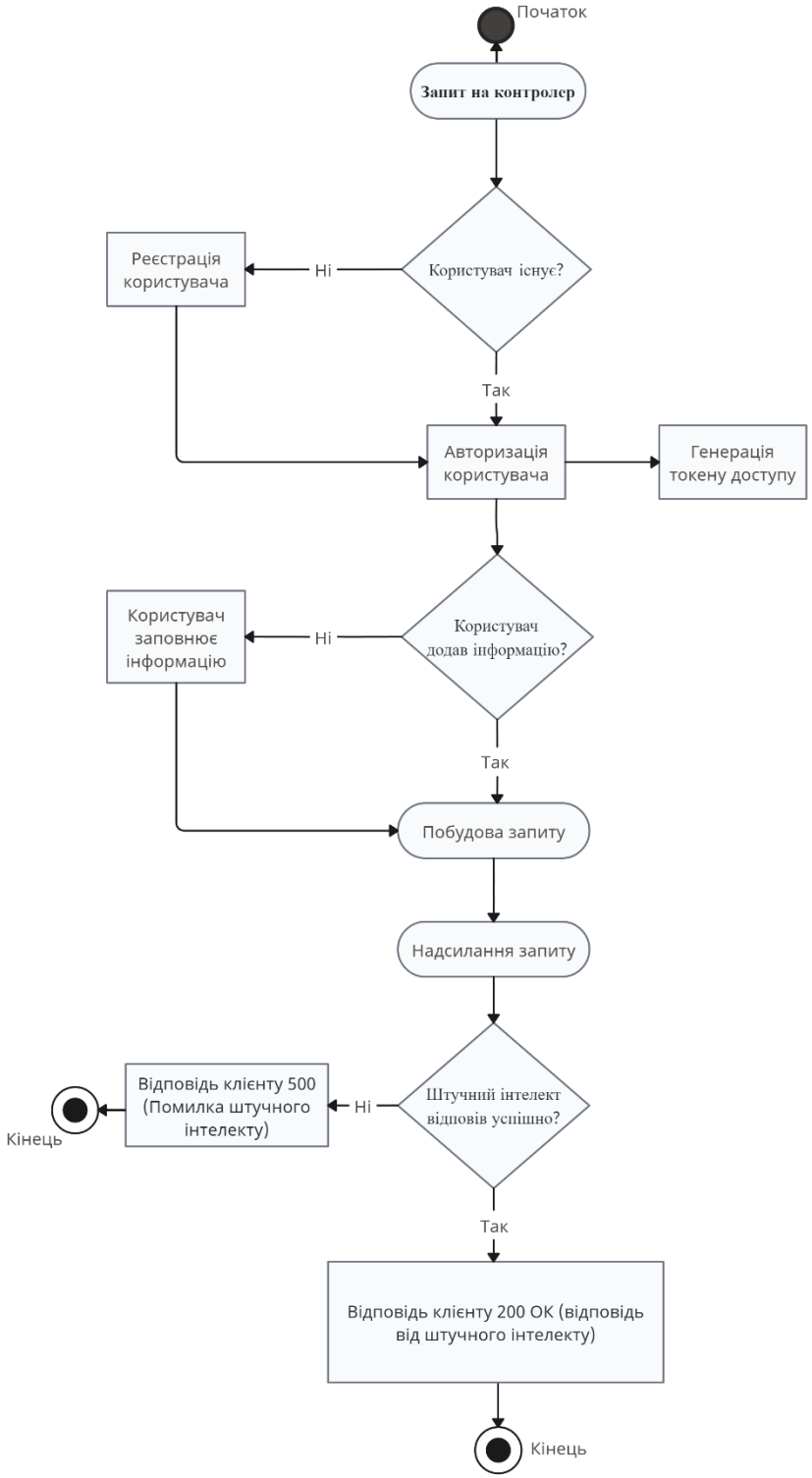


Рисунок Д.1 — UML-діаграма діяльності серверної частини веб-застосунку зі штучним інтелектом

ДОДАТОК Е

Лістинг модулю інтеграції з штучним інтелектом

Лістинг Е.1 — Програмний код класу AIController

```
using AI_Diet.Context;
using AI_Diet.Models.UserModels;
using AI_Diet.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Net.Http;

namespace AI_Diet.Controllers
{
    [Authorize]
    [Route("api/chat-gpt")]
    [ApiController]
    public class AIController : ControllerBase
    {
        private IChatGptService _chatGptService;
        private IChatRequestBuilder _requestBuilder;

        private ApplicationContext _applicationContext;

        public AIController(IChatGptService chatGptService, IChatRequestBuilder
requestBuilder, ApplicationContext applicationContext)
        {
            _chatGptService = chatGptService;
            _requestBuilder = requestBuilder;

            _applicationContext = applicationContext;
        }

        [HttpGet("ask-for-diet")]
        public IActionResult AskForDiet(string userId)
        {
            try
            {
                var dietData = _applicationContext.DietData.FirstOrDefault(c => c.UserId ==
userId);
                var foodDetails = _applicationContext.FoodDetails.FirstOrDefault(d =>
d.UserId == userId);

                if (dietData == null)
                {
                    return NotFound("User not found");
                }

                var dietRequest = _requestBuilder.BuildDietRequest(new
DietRequestModel(dietData, foodDetails));
                var chatResponse = _chatGptService.Ask(dietRequest);

                return Ok(chatResponse);

            }
            catch (Exception e)
            {
                return StatusCode(500, e.Message);
            }
        }

        [HttpGet("ask-for-training")]
        public IActionResult AskForTraining(string userId)
```

```
        {
            try
            {
                var dietData = _applicationContext.DietData.FirstOrDefault(c => c.UserId ==
userId);

                if (dietData == null)
                {
                    return NotFound("User not found");
                }

                var dietRequest = _requestBuilder.BuildTrainingRequest(dietData);
                var chatResponse = _chatGptService.Ask(dietRequest);

                return Ok(chatResponse);

            }
            catch (Exception e)
            {
                return StatusCode(500, e.Message);
            }
        }
    }
}
```

Лістинг Е.2 — Програмний код класу ChatGptService

```
using System.Text;

namespace AI_Diet.Services
{
    public class ChatGptService : IChatGptService
    {
        private HttpClient _chatGptClient;
        private IConfiguration _configuration;

        private const string CHAT_POST_ADDRESS =
"https://api.openai.com/v1/chat/completions";
        private const string CHAT_GPT_TOKEN = "ChatGptToken";

        public ChatGptService(IConfiguration configuration)
        {
            _configuration = configuration;

            InitHttpClient();
        }

        public string Ask(string chatRequest)
        {
            var content = new StringContent(chatRequest, Encoding.UTF8,
"application/json");

            HttpResponseMessage response = _chatGptClient.PostAsync(CHAT_POST_ADDRESS,
content).Result;

            var responseString = response.Content.ReadAsStringAsync().Result;
            return responseString;
        }

        private void InitHttpClient()
        {
            _chatGptClient = new HttpClient();
            var token = _configuration.GetValue<string>(CHAT_GPT_TOKEN);
```

```
            _chatGptClient.DefaultRequestHeaders.Add("authorization", $"Bearer {token}");
            _chatGptClient.Timeout = TimeSpan.FromMinutes(5);
        }
    }
}
```

Лістинг Е.2 — Програмний код класу ChatRequestBuilder

```
using AI_Diet.Models;
using AI_Diet.Models.UserModels;
using System.Text;

namespace AI_Diet.Services
{
    public class ChatRequestBuilder : IChatRequestBuilder
    {
        private string _messageBegining = "{\"model\": \"gpt-3.5-turbo\",\"messages\":[{ ";

        public string BuildDietRequest(DietRequestModel dietRequestModel)
        {
            var foodDetails = dietRequestModel.FoodDetails;
            var dietRequest = new StringBuilder(_messageBegining);

            dietRequest.AppendLine("\"role\": \"system\", \"content\": \"Ти персональний
дієтолог\"},");
            BuildBaseMessage(dietRequestModel.DietData, dietRequest);
            dietRequest.Append("Мені потрібно щоб ти розписав мені дієту згідно моїх
параметрів.");

            if (IsAnyFoodDetails(foodDetails))
            {
                AppendFoodDetails(foodDetails, dietRequest);
            }
            dietRequest.Append("Дієту розпиши наче ти персональний дієтолог, і розписуєш
дієту своєму клієнту у вигляді документу. " +
                "Потрібно розписати дієту на тиждень, з різними стравами, але збереженням
денного калоражу. " +
                "Документ напиши строгою, технічною мовою. Потрібно, щоб він включав в себе
денний калораж, а також біля кожної " +
                "страви було розписано хоча б приблизна кількість калорій. Ім'я в документі
не вказуй\"}]}");
            var result = dietRequest.ToString();
            return result;
        }

        public string BuildTrainingRequest(DietData dietData)
        {
            var trainingRequest = new StringBuilder(_messageBegining);

            trainingRequest.Append("\"role\": \"system\", \"content\": \"Ти персональний
тренер\"},");
            BuildBaseMessage(dietData, trainingRequest);
            trainingRequest.Append("Мені потрібно, щоб ти розписав мені персональні
тренування у спортивному залі, " +
                "по дням, для досягнення моєї цілі. Тренування розпиши наче ти персональний
тренер, " +
                "і розписуєш тренування своєму клієнту у вигляді документу\"}]}");

            return trainingRequest.ToString();
        }

        private void BuildBaseMessage(DietData dietData, StringBuilder request)
        {
```

```
            var goal = AdjustGoalToString(dietData.Goal);
            var gender = AdjustGenderToString(dietData.Gender);

            request.Append($"{{ \"role\": \"user\", \"content\": \"" +
                $"Я {gender} мені {dietData.Age} років, мій зріст {dietData.Height}
сантиметрів, " +
                $"а моя вага складає {dietData.Weight} кілограм. Мою денну фізичну
активність можна описати так: {dietData.PhysicalActivity}," +
                $"а моя ціль: {goal}.");
        }

        private string AdjustGoalToString(Enums.DietGoal goal)
        {
            switch (goal)
            {
                case Enums.DietGoal.LooseWeight:
                    return "Похудати";
                case Enums.DietGoal.KeepWeight:
                    return "Зберегти масу";
                case Enums.DietGoal.GainWeight:
                    return "Набрати м'язову масу";
                default:
                    return "Невідома";
            }
        }

        private string AdjustGenderToString(Enums.Gender gender)
        {
            switch (gender)
            {
                case Enums.Gender.Male:
                    return "чоловік";
                case Enums.Gender.Female:
                    return "жінка";
                default:
                    return "неідентифікований";
            }
        }
        private void AppendFoodDetails(FoodDetails dietData, StringBuilder dietRequest)
        {
            dietRequest.Append("Але, будь ласка, зверни увагу на деякі деталі, щодо
майбутньої дієти:");
            if (!string.IsNullOrEmpty(dietData.Allergies))
            {
                dietRequest.Append($"* У мене є алергія на наступне:
{dietData.Allergies}.");
            }
            if (!string.IsNullOrEmpty(dietData.FoodRestrictions))
            {
                dietRequest.Append($"* У мене є обмеження у споживанні:
{dietData.FoodRestrictions}.");
            }
            if (!string.IsNullOrEmpty(dietData.DislikeFood))
            {
                dietRequest.Append($"* Мені не подобається наступна їжа:
{dietData.DislikeFood}, " +
                    $"тому при побудові дієти використовуй її по мінімум.");
            }
            if (!string.IsNullOrEmpty(dietData.FoodPreferences))
            {
                dietRequest.Append($"* Я люблю наступну їжу: {dietData.FoodPreferences}.");
            }
        }
        private bool IsAnyFoodDetails(FoodDetails dietData)
```

```
{
    if (dietData != null &&
        (!string.IsNullOrEmpty(dietData.DislikeFood) ||
        !string.IsNullOrEmpty(dietData.FoodRestrictions) ||
        !string.IsNullOrEmpty(dietData.FoodPreferences) ||
        !string.IsNullOrEmpty(dietData.Allergies)))
    {
        return true;
    }
    return false;
}}}
```

ДОДАТОК Ж

Лістинг модулю доступу до даних

Лістинг Ж.1 — Програмний код класу ApplicationContext

```
using AI_Diet.Models.UserModels;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace AI_Diet.Context
{
    public class ApplicationContext: IdentityDbContext<User>
    {
        public DbSet<DietData> DietData { get; set; }
        public DbSet<FoodDetails> FoodDetails { get; set; }
        public ApplicationContext(DbContextOptions options): base(options)
        {
            Database.EnsureDeleted();
            Database.EnsureCreated();
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>()
                .HasOne(u => u.DietData)
                .WithOne(cd => cd.User)
                .HasForeignKey<DietData>(cd => cd.UserId);

            modelBuilder.Entity<User>()
                .HasOne(u => u.FoodDetails)
                .WithOne(dd => dd.User)
                .HasForeignKey<FoodDetails>(dd => dd.UserId);

            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Лістинг Ж.2 — Програмний код класу User

```
using AI_Diet.Models.RequestModels;
using Microsoft.AspNetCore.Identity;

namespace AI_Diet.Models.UserModels
{
    public class User : IdentityUser
    {
        public string Name { get; set; }
        public string RefreshToken { get; set; }

        public DietData DietData { get; set; }
        public FoodDetails FoodDetails { get; set; }
        public string Diet { get; set; }
        public string Training { get; set; }

        public User()
        {}

        public User(RegisterUserRequest registerUserRequestModel)
        {
            Name = registerUserRequestModel.Name;
            Email = registerUserRequestModel.Email;
            UserName = registerUserRequestModel.Email;
```

```
        }}}
```

Лістинг Ж3 — Програмний код класу UserService

```
using AI_Diet.Authorization.Services.Interfaces;
using AI_Diet.Context;
using AI_Diet.Models.RequestModels;
using AI_Diet.Models.ResponseModels;
using AI_Diet.Models.UserModels;

namespace AI_Diet.Authorization.Services
{
    public class UserService : IUserService
    {
        ApplicationContext _dbContext;

        public UserService(ApplicationContext dbContext)
        {
            _dbContext = dbContext;
        }

        public bool AddDietData(DietData dietData)
        {
            try
            {
                _dbContext.DietData.Add(dietData);
                _dbContext.SaveChanges();

                return true;
            }
            catch
            {
                return false;
            }
        }

        public bool AddDietToUser(AddDietToUserRequestModel dietModel)
        {
            try
            {
                var user = _dbContext.Users.FirstOrDefault(user => user.Id ==
dietModel.UserId);
                user.Diet = dietModel.Diet;
                _dbContext.SaveChanges();

                return true;
            }
            catch
            {
                return false;
            }
        }

        public bool AddTrainingToUser(AddTrainingToUserRequestModel trainingModel)
        {
            try
            {
                var user = _dbContext.Users.FirstOrDefault(user => user.Id ==
trainingModel.UserId);
                user.Training = trainingModel.TrainingProgramm;
                _dbContext.SaveChanges();

                return true;
            }
            catch
```

```
    {
        return false;
    }
}

public GetUserResponseModel GetUser(string userId)
{
    var user = _dbContext.Users.FirstOrDefault(user => user.Id == userId);
    if (user == null)
    {
        return null;
    }
    return new GetUserResponseModel(user);
}

public bool AddFoodDetails(FoodDetails foodDetails)
{
    try
    {
        _dbContext.FoodDetails.Add(foodDetails);
        _dbContext.SaveChanges();

        return true;
    }
    catch
    {
        return false;
    }        }    }}
```

ДОДАТОК И

Лістинг модулю авторизації

Лістинг И.1 — Програмний код класу AuthController

```
using AI_Diet.Authorization.Services.Interfaces;
using AI_Diet.Models.RequestModels;
using Microsoft.AspNetCore.Mvc;

namespace AI_Diet.Controllers
{
    [Route("api/auth")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private IAuthService _authService;
        private ITokenService _tokenService;

        public AuthController(IAuthService authService, ITokenService tokenService)
        {
            _authService = authService;
            _tokenService = tokenService;
        }

        [HttpPost("login")]
        public async Task<IActionResult> LoginAsync([FromBody] LoginRequest loginRequest)
        {
            var loginResponse = await _authService.LoginAsync(loginRequest.Email,
loginRequest.Password);

            if (loginResponse == null)
            {
                return BadRequest("Email or Password is invalid");
            }

            Response.Cookies.Append("Refresh-Token", loginResponse.RefreshToken, new
CookieOptions() { HttpOnly = true });
            return Ok(loginResponse);
        }

        [HttpPost("refresh-token")]
        public async Task<IActionResult> RefreshAccessTokenAsync(RefreshTokenRequest
refreshTokenRequest)
        {
            try
            {
                await _tokenService.ValidateRefreshTokenRequestAsync(refreshTokenRequest);
            }
            catch (KeyNotFoundException knfe)
            {
                return NotFound(knfe.Message);
            }
            catch (UnauthorizedAccessException uae)
            {
                return Unauthorized(uae.Message);
            }
            catch (Exception e)
            {
                return BadRequest(e.Message);
            }

            return Ok(_tokenService.CreateRefreshTokenResponse(refreshTokenRequest));
        }
```

```
        [HttpPost("logout")]
        public async Task<IActionResult> LogoutAsync()
        {
            await _authService.LogoutAsync();

            Response.Cookies.Delete("Refresh-Token", new CookieOptions() { HttpOnly = true
});
            return Ok();
        }
    }
}
```

Лістинг И.2 — Програмний код класу AuthService

```
using AI_Diet.Authorization.Services.Interfaces;
using AI_Diet.Context;
using AI_Diet.Models.RequestModels;
using AI_Diet.Models.ResponseModels;
using AI_Diet.Models.UserModels;
using Microsoft.AspNetCore.Identity;

namespace AI_Diet.Authorization.Services
{
    public class AuthService : IAuthService
    {
        private SignInManager<User> _signInManager;
        private UserManager<User> _userManager;
        private ApplicationContext _dbContext;
        private ITokenService _tokenService;

        public AuthService(SignInManager<User> signInManager, UserManager<User>
userManager, ITokenService tokenService, ApplicationContext dbContext)
        {
            _signInManager = signInManager;
            _userManager = userManager;
            _tokenService = tokenService;
            _dbContext = dbContext;
        }

        public async Task<LoginResponse> LoginAsync(string email, string password)
        {
            var user = await _userManager.FindByEmailAsync(email);
            var result = await _signInManager.PasswordSignInAsync(user, password, false,
false);

            if (!result.Succeeded)
            {
                return default;
            }

            var loginResponse = CreateLoginResponse(user);
            var loggedInUser = _dbContext.Users.FirstOrDefault(user => user.Id ==
loginResponse.UserId);

            loggedInUser.RefreshToken = loginResponse.RefreshToken;
            _dbContext.SaveChanges();

            return loginResponse;
        }

        public async Task LogoutAsync()
        {
            await _signInManager.SignOutAsync();
        }
```

```
        public async Task<RegisterUserResponse> RegisterAsync(RegisterUserRequest
registerUserRequestModel)
        {
            var userToRegister = new User(registerUserRequestModel);
            var result = await _userManager.CreateAsync(userToRegister,
registerUserRequestModel.Password);

            if (!result.Succeeded)
            {
                return new RegisterUserResponseErrors() { RegisterErrors =
result.Errors?.Select(x => x.Description).ToList() };
            }

            return CreateRegisterReponse(userToRegister);
        }

        public async Task<DeleteUserResponse> DeleteUserAsync(string userId)
        {
            var user = await _userManager.FindByIdAsync(userId);

            if (user == null)
            {
                return new DeleteUserResponse(false) { Errors = new List<string>() { "User
not found" } };
            }

            var result = await _userManager.DeleteAsync(user);
            return new DeleteUserResponse(result.Succeeded) { Errors =
result.Errors?.Select(x => x.Description).ToList() };
        }

        private LoginResponse CreateLoginResponse(User user)
        {
            return new LoginResponse
            {
                AccessToken = _tokenService.CreateToken(false),
                RefreshToken = _tokenService.CreateToken(true),
                UserId = user.Id,
                UserName = user.Email,
                FirstName = user.Name,
            };
        }

        private RegisterUserResponse CreateRegisterReponse(User user)
        {
            return new RegisterUserResponse
            {
                UserId = user.Id,
                UserName = user.Email,
                FirstName = user.Name,
            };
        }
    }
}
```

Лістинг И.3 — Програмний код налаштування авторизації

```
builder.Services.AddIdentity<User, IdentityRole>(opts =>
{opts.Password.RequiredLength = 8;
    opts.Password.RequireNonAlphanumeric = false;
    opts.Password.RequireLowercase = true;
    opts.Password.RequireUppercase = true;
    opts.Password.RequireDigit = true;
})
```

```
    .AddEntityFrameworkStores<ApplicationContext>()
    .AddDefaultTokenProviders()
    .AddSignInManager<SignInManager<User>>()
    .AddUserManager<UserManager<User>>();
AuthOptions authOptions =
builder.Configuration.GetSection(nameof(AuthOptions)).Get<AuthOptions>();
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
})
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.SaveToken = true;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = authOptions.Issuer,
            ValidAudience = authOptions.Audience,
            IssuerSigningKey = authOptions.GetSymmetricSecurityAccessKey(),
            ClockSkew = TimeSpan.Zero
        };});
builder.Services.AddSingleton(c => authOptions);
builder.Services.AddAuthorization();
```

ДОДАТОК К

Протокол перевірки кваліфікаційної робити

Назва роботи: Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 3. Серверна частина

Тип роботи: комплексна магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97,4% Схожість 2,6%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Рижков А. К.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Войцеховська О. В.
(підпис) (прізвище, ініціали)