НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«До захисту допущено»
В.о. завідувачки кафедри БМК

_____ Світлана АЛХІМОВА
(підпис) (ініціали, ПРІЗВИЩЕ)

“___” __*червня*__2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп’ютерні технології в біології та медицині»
зі спеціальності *122 «Комп’ютерні науки»*

на тему: ***Веб-застосунок для моніторингу здоров’я пацієнтів із цукровим діабетом 1-го типу***

Виконав: студент **IV** курсу, групи _БС-15__

ЗАСЛАВСЬКИЙ МАКСИМ ВІТАЛІЙОВИЧ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник: *ст. викл. каф. БМК.*
Гладка Мирослава Вікторівна _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по ініціали) (підпис)

Рецензент: *асистент каф. біомедичної інженерії.*
Попов Станіслав Володимирович _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувачки кафедри БМК
_____ Світлана АЛХІМОВА
«_ ___ »__*квітень*___2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

ЗАСЛАВСЬКОМУ МАКСИМУ ВІТАЛІЙОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи ***Веб-застосунок для моніторингу здоров'я пацієнтів із цукровим діабетом 1-го типу***

керівник роботи
Гладка Мирослава Вікторівна к.т.н, Ст. вик. каф. БМК
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26»_*травня*_ 2025 р. №***1758-с***_

2. Термін подання студентом роботи ***22-23 травня 2025 року***
3. Вихідні дані: *дані з інсуліно-залежних хворих*
4. Зміст роботи: *Аналіз аналогів та підходів до розробки подібних систем, засоби реалізації веб-застосунку, обґрунтування вибору програмної реалізації для веб-застосунку, програмна реалізація та методика роботи застосунку, узагальнення результатів роботи*
5. Перелік ілюстративного матеріалу : 15 слайдів
6. Консультанти розділів роботи[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Дипломної роботи			

7. Дата видачі завдання ***06 квітня 2025р.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 24.12.2024р.	***виконано***
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	***виконано***
3	Виконання основних розділів ДР (Вступ, Аналіз аналогів та підходів до розробки подібних систем, засоби реалізації веб-застосунку, обґрунтування вибору програмної реалізації для веб-застосунку, програмна реалізація та методика роботи застосунку, узагальнення результатів роботи)	Протягом усієї практики	
4	Перевірка ДР науковим керівником	до 20.05.2025	
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	
9	Захист ДР в ЕК	*16.06 - -21.06.2025*	

Студент _____ **Максим ЗАСЛАВСЬКИЙ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Мирослава ГЛАДКА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Веб-застосунок для моніторингу здоров'я пацієнтів із цукровим діабетом 1-го типу» виконана студентом кафедри біомедичної кібернетики ФБМІ *Заславським Максимом Віталійовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 5 розділів (Аналіз аналогів та підходів до розробки подібних систем, засоби реалізації веб-застосунку, обґрунтування вибору програмної реалізації для веб-застосунку, програмна реалізація та методика роботи застосунку, узагальнення результатів роботи*)*, висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 31 джерел, та додатків. Загальний обсяг роботи 71 сторінок.

Актуальність теми. Актуальність роботи полягає в розробці доступного, локалізованого українською мовою веб-додатку SickDiary для автоматизації самоконтролю цукрового діабету 1 типу

Мета і завдання роботи.

Мета роботи полягає в покращенні медичної сфери застосунком, який забезпечує автоматизацію ведення щоденника здоров'я для хворих на ЦД1, надаючи можливості для введення, зберігання, аналізу та візуалізації даних, а також генерації персоналізованих рекомендацій на основі сучасних технологій. Застосунок орієнтований на локальний контекст, враховує українські медичні настанови та відповідає міжнародним стандартам безпеки даних.

Для реалізації поставлених задач, було поставлено наступні задачі:

- Розробити алгоритм для автоматичного визначення стану здоров'я
- Інтегрувати модель штучного інтелекту
- Розробити зручний інтерфейс для перегляду та введення даних
- Забезпечити захист персональних даних

Використані методи. Огляд та аналіз аналогічних існуючих систем, вибір мови програмування, плафторми, фреймворку та бібліотек.

Отримані результати. Створений веб-застосунок, написаний мовою програмування C# на платформі ASP.NET Core MVC. Всі поставленні задачі були виконані та впровадженні в застосунок.

Публікації. За результатами даної роботи публікації не передбачаються.

Бібліографічний опис ДР

Заславський М. В. Веб-застосунок для моніторингу здоров’я пацієнтів із цукровим діабетом 1-го типу : дипломна роб. бакалавра : 122 Комп’ютері науки / Заславський Максим Віталійович. – Київ, 2025. – 71 с

ABSTRACT

The bachelor's thesis titled "Web Application for Health Monitoring of Patients with Type 1 Diabetes" was completed by Maksym Vitaliiovych Zaslavskyi, a student of the Department of Biomedical Cybernetics at the Faculty of Biomedical Engineering, specializing in 122 "Computer Science" under the educational and professional program "Computer Technologies in Biology and Medicine". The work consists of: an introduction; 5 chapters (Analysis of analogues and approaches to the development of similar systems, tools for implementing the web application, justification of the choice of software implementation for the web application, software implementation and operation methodology of the application, summary of the work results); conclusions for each chapter; general conclusions; a list of references containing 31 sources; and appendices. The total volume of the work is 71 pages.

Relevance of the topic.

The relevance of the study lies in the development of an accessible, Ukrainian-localized web application "SickDiary" for the automation of self-monitoring for Type 1 Diabetes (T1D) patients.

Purpose and objectives of the thesis.

The purpose of the thesis is to improve the medical sector by developing an application that automates the maintenance of a health diary for T1D patients. It enables data input, storage, analysis, and visualization, as well as the generation of personalized recommendations based on modern technologies. The application is designed with a local context in mind, takes into account Ukrainian medical guidelines, and complies with international data security standards.

To achieve this goal, the following tasks were set:

- Develop an algorithm for automatic health status assessment
- Integrate an artificial intelligence model
- Create a user-friendly interface for data entry and review
- Ensure protection of personal data

Methods used.

Review and analysis of existing similar systems; selection of programming language, platform, framework, and libraries.

Results.

A web application was developed using the C# programming language on the ASP.NET Core MVC platform. All defined objectives were achieved and implemented in the application.

Publications.

No publications are planned based on this thesis.

Bibliographic description:

Zaslavskyi M. V. *Web Application for Health Monitoring of Patients with Type 1 Diabetes* : Bachelor's thesis : 122 Computer Science / Maksym Vitaliiovych Zaslavskyi. – Kyiv, 2025. – 71 p.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЦД1 – цукровий діабет першого типу

БД - база даних

ПЗ – програмне забезпечення

ЕСОЗ – електронна система охорони здоров'я

GDPR - захист персональних даних

HIPAA - безпека медичних даних

HTTPS – протокол передачі даних із шифруванням

SSL/TLS – стандарти шифрування (Secure Sockets Layer/Transport Layer Security)

ВСТУП

Цукровий діабет першого типу є одним із найпоширеніших хронічних автоімунних захворювань, що вимагає від хворих щоденного самоконтролю, моніторингу рівня глюкози в крові, введення інсуліну та врахування впливу зовнішніх факторів, таких як харчування, фізична активність і емоційний стан. За даними Всесвітньої організації охорони здоров'я (ВООЗ), у 2021 році цукровий діабет діагностовано у 537 мільйонів людей у світі, причому значна частка припадає на ЦД1, і цей показник невпинно зростає. [1]

В Україні, згідно з адаптованою клінічною настановою Державного експертного центру Міністерства охорони здоров'я України (МОЗ), ЦД1 становить серйозну проблему через високу частоту ускладнень, таких як гіпоглікемія, гіперглікемія та діабетичний кетоацидоз, що погіршують якість життя хворих і підвищують навантаження на систему охорони здоров'я. Постійна потреба в самоконтролі зумовлює актуальність створення доступних, зручних і технологічно досконалих інструментів, які полегшують управління захворюванням. [2]

Традиційні методи ведення щоденників здоров'я, такі як паперові записи, є трудомісткими, незручними та схильними до втрати даних. Хворі на ЦД1 змушені фіксувати численні показники — рівень глюкози, дози інсуліну, споживання вуглеводів, фізичну активність і симптоми — для уникнення небезпечних станів. Ці методи ускладнюють аналіз трендів і не дозволяють швидко реагувати на критичні зміни. Сучасні інформаційні технології, зокрема веб-додатки, відкривають нові можливості для автоматизації цього процесу, забезпечуючи цифрове зберігання даних, інтуїтивний інтерфейс і аналітичні інструменти. Такі рішення стають особливо цінними в контексті зростання популярності телемедицини та необхідності дистанційного моніторингу здоров'я, що підтверджується міжнародними дослідженнями в галузі медичних інформаційних систем [3].

В Україні проблема ЦД1 посилюється обмеженим доступом до спеціалізованих пристроїв, таких як безперервні глюкометри, через їх високу вартість, а також недостатньою локалізацією іноземних цифрових рішень українською мовою. Наприклад, популярні додатки, такі як Dexcom G6 або MySugr, не підтримують українську мову і вимагають дорогого обладнання, що робить їх менш доступними для локального населення [4, 5]. Крім того, національна електронна система охорони здоров'я (ЕСОЗ) орієнтована переважно на медичних працівників, а не на самоконтроль хворих, що створює прогалину в інструментах для щоденного використання [6]. Таким чином, створення локалізованого веб-додатку, який поєднує зручність, доступність і персоналізовані рекомендації, є важливим кроком для покращення якості життя хворих на ЦД1 в Україні.

Мета і завдання роботи.

Мета роботи полягає в покращенні медичної сфери застосунком, який забезпечує автоматизацію ведення щоденника здоров'я для хворих на ЦД1, надаючи можливості для введення, зберігання, аналізу та візуалізації даних, а також генерації персоналізованих рекомендацій на основі сучасних технологій. Застосунок орієнтований на локальний контекст, враховує українські медичні настанови та відповідає міжнародним стандартам безпеки даних, SSL/TLS [10].

Для реалізації поставлених задач, було поставлено наступні задачі:

- Розробити алгоритм для автоматичного визначення стану здоров'я
- Інтегрувати модель штучного інтелекту
- Розробити зручний інтерфейс для перегляду та введення даних
- Забезпечити захист персональних даних

Використані методи. Огляд та аналіз аналогічних існуючих систем, вибір мови програмування, плафторми, фреймворку та бібліотек.

Отримані результати. Створений веб-застосунок, написаний мовою програмування C# на платформі ASP.NET Core MVC. Всі поставленні задачі були виконані та впровадженні в застосунок.

Публікації. За результатами даної роботи публікації не передбачаються.

Структура роботи.

Дипломна робота за темою «Веб-застосунок для моніторингу здоров'я хворих на цукровий діабет 1-го типу» виконана студентом *Заславським Максимом Віталійовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині»*, побудована за класичним типом та викладена на 71 сторінках машинописного тексту. Вона складається з: вступу; 5 розділів (*Аналіз аналогів та підходів до розробки подібних систем, засоби реалізації веб-застосунку, обґрунтування вибору програмної реалізації для веб-застосунку, програмна реалізація та методика роботи застосунку, узагальнення результатів роботи)*, висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 31 джерел та додатків (7 – на кирилиці, 24 – на латиниці). В роботі представлено 13 рисунків і 7 таблиць.

РОЗДІЛ 1.
АНАЛІЗ АНАЛОГІВ ТА ПІДХОДІВ ДО РОЗРОБКИ ПОДІБНИХ СИСТЕМ

1.1 Огляд існуючих систем для контролю здоров'я хворих на цукровий діабет першого типу

Розвиток інформаційних технологій у сфері медицини призвів до появи численних систем, які покликані допомагати хворим на цукровий діабет першого типу у щоденному контролі їхнього стану. Аналіз аналогів дозволяє визначити сильні та слабкі сторони існуючих рішень, а також виявити прогалини, які можуть бути заповнені в рамках даної роботи. Серед найвідоміших систем варто виділити як комерційні продукти, так і відкриті платформи, які використовуються для моніторингу рівня глюкози, ведення щоденників і надання рекомендацій. [9]

1.1.1 Dexcom G6

Одним із популярних комерційних рішень є додаток Dexcom G6, розроблений компанією Dexcom, який інтегрується з безперервними глюкометрами (CGM) і дозволяє відстежувати рівень глюкози в реальному часі. Ця система передає дані на смартфон користувача через Bluetooth, надаючи графіки та сповіщення про критичні стани, такі як гіпоглікемія чи гіперглікемія. Однак Dexcom G6 вимагає спеціалізованого обладнання, що значно підвищує його вартість, і не підтримує локалізований інтерфейс українською мовою, що обмежує його доступність для українських користувачів. Крім того, функціонал аналізу даних і надання рекомендацій у цьому додатку базується переважно на базових алгоритмах, без інтеграції з сучасними технологіями обробки даних, що зменшує його адаптивність до індивідуальних потреб хворого [5].

1.1.2 MySugr

Додаток MySugr пропонує комплексний підхід до ведення щоденника здоров’я для хворих на діабет. MySugr дозволяє вводити дані про рівень глюкози, дози інсуліну, вуглеводи та фізичну активність вручну або імпортувати їх із глюкометрів. Додаток включає аналітичний модуль, який генерує звіти для користувача та медичних фахівців, а також інтеграцію з Apple Health і Google Fit. Незважаючи на зручний інтерфейс, MySugr має обмеження у вигляді платної підписки для розширених функцій і відсутності підтримки української мови, що ускладнює його використання в Україні. Крім того, система не пропонує персоналізованих рекомендацій, базуючись лише на статистичних даних. [6]

1.1.3 OpenAPS

Відкриті платформи, такі як OpenAPS, також заслуговують на увагу. OpenAPS є спільнотою, яка розробляє програмне забезпечення для створення замкнених систем дозування інсуліну, інтегруючи дані з глюкометрів і інсулінових помп. Ця система використовує алгоритми для автоматичного коригування доз інсуліну, що є інноваційним підходом. Однак OpenAPS вимагає значних технічних навичок для налаштування, а також спеціалізованого обладнання, що робить його недоступним для більшості хворих на ЦД1. Крім того, відсутність централізованого інтерфейсу та локалізації обмежує його практичне застосування в українських реаліях. [7]

1.1.4 Національні ініціативи: ЕСОЗ

Електронна система охорони здоров’я України (ЕСОЗ) також включає елементи моніторингу хронічних захворювань, зокрема ЦД1. ЕСОЗ дозволяє лікарям вести електронні картки пацієнтів і відстежувати їхні показники, але система орієнтована переважно на медичних працівників, а не на самоконтроль хворих. Брак інтерактивного інтерфейсу для пацієнтів і відсутність аналітичних інструментів для щоденного використання роблять її менш придатною для індивідуального моніторингу [8].

Таким чином, аналіз аналогів показує, що існуючі системи або вимагають дорогого обладнання, або мають обмеження у функціональності, локалізації та доступності, див. табл. 1.1.

Таблиця 1.1

Порівняння аналогів програмного забезпечення

Характеристика	**Dexcom G6 [5]**	**MySugr [6]**	**OpenAPS [7]**	**ЕСОЗ [8]**
Тип	CGM + додаток	Мобільний додаток	Відкрита система	Веб-платформа
Локалізація українською	Ні	Ні	Ні	Так
Вартість	Висока	Частково платна	Безкоштовна	Безкоштовна
Інтеграція з апаратним забезпеченням	Так	Обмежена	Так	Ні
Аналітичні функції	Так	Обмежені	Так	Ні
Складність використання	Середня	Низька	Висока	Середня

Аналіз аналогів показує, що більшість рішень мають обмеження в доступності, локалізації або функціональності для українських користувачів, що підкреслює потребу в новому рішенні.

1.2 Аналіз підходів до розробки медичних веб-додатків

Розробка медичних веб-додатків, таких як система для контролю ЦД1, базується на низці підходів, які визначають їхню архітектуру, продуктивність і безпеку. Одним із ключових підходів є використання клієнт-серверної архітектури, яка дозволяє розподілити навантаження між фронтендом і

бекендом. У комерційних системах, таких як Dexcom G6, ця архітектура реалізується через хмарні сервери, що забезпечують обробку даних у реальному часі. Однак такий підхід вимагає значних ресурсів і може бути вразливим до перебоїв у мережі, що є проблемою для регіонів із нестабільним інтернетом, таких як деякі райони України. [5]

Інший підхід пов'язаний із використанням мікросервісної архітектури, яка дозволяє модульно розподілити функціонал додатку, наприклад, окремий сервіс для аутентифікації, аналітики та зберігання даних. Цей метод застосовується в сучасних платформах, таких як MySugr, і забезпечує гнучкість та масштабованість. Однак складність реалізації мікросервісів може бути бар'єром для невеликих проєктів, таких як розробка для локального ринку. У контексті даної роботи монолітна архітектура, реалізована через ASP.NET Core, обрана як більш підходяща для спрощення розробки та підтримки. [6]

Щодо аналітичних підходів, використання сучасних API для обробки даних набуває популярності. Наприклад, інтеграція API, таких як OpenAI GPT-3.5, дозволяє аналізувати записи щоденника та надавати персоналізовані рекомендації на основі введених даних, таких як рівень глюкози, дози інсуліну чи симптоми. [24] Цей підхід є перспективним, оскільки забезпечує адаптивність без необхідності складних обчислень на стороні клієнта. У системах на кшталт Open: OpenAPS використовуються базові алгоритми для коригування інсуліну, але вони потребують апаратного забезпечення. [7] У пропонованому веб-додатку аналіз базується на правилових алгоритмах, які оцінюють стан здоров'я за пороговими значеннями (наприклад, глюкоза 3.9–7.2 ммоль/л натщесерце) [1], що є простішим і доступнішим рішенням.

Безпека даних є критичним аспектом медичних додатків. Усі розглянуті аналоги використовують шифрування даних під час передачі (HTTPS), а деякі, як-от ЕСОЗ, додатково регулюються державними стандартами [8]. Міжнародні рекомендації, такі як HIPAA (США) і GDPR (ЄС), підкреслюють важливість анонімізації даних і захисту конфіденційності. Ці підходи враховані при

розробці пропонованого додатку, де використовуються хешування паролів (BCrypt) і безпечні протоколи (HTTPS/SSL). [11, 12]

Таким чином, аналіз підходів показує, що комбінація клієнт-серверної архітектури, інтеграції з API для рекомендацій та дотримання стандартів безпеки є оптимальною для створення ефективного веб-додатку. Ці принципи лягли в основу розробки системи, описаної в наступних розділах.

1.3 Постановка задачі на роботу

Основними задачами на реалізацію та побудову веб-застосунку будуть обрані наступні пункти:

- Створити алгоритм для автоматизованої оцінки стану здоров’я пацієнта.

Автоматизований алгоритм дозволяє оперативно оцінювати стан пацієнта на основі введених даних (наприклад, рівень глюкози, доза інсуліну, самопочуття). Це мінімізує вплив людського фактора, допомагає швидше виявляти критичні стани й підтримує лікаря або пацієнта в ухваленні рішень. Зявдяки цьому ми отримуємо конкретні переваги – швидке виявлення гіпоглікемії та гіперглікемії, можливість автоматичних попереджень або рекомендацій та зменшення навантаження на медперсонал.

- Реалізувати інтеграцію моделі штучного інтелекту в систему.

Штучний інтелект дозволяє виявляти складні закономірності у медичних даних, які не завжди очевидні людині. Інтеграція моделі ШІ в систему підвищує точність прогнозування ризиків, автоматично формує персоналізовані рекомендації та покращує загальну ефективність моніторингу. З переваг можемо виділити такі пункти: персоналізовані поради з урахуванням історії хвороби, прогнозування можливих ускладнень на основі динаміки даних, самонавчання моделі з часом для підвищення її точності. Інтеграція з OpenAPI дозволяє нам використовувати їхню технологію, яка постійно

навчається, оновлюється та покращується. В нашому випадку – це ідеальний варіант для майбутнього веб-застосунку.

- Розробити інтуїтивно зрозумілий інтерфейс для введення та перегляду медичних даних.

Для пацієнтів із цукровим діабетом 1-го типу важливо щодня вносити та переглядати показники здоров'я. Якщо інтерфейс буде складним або незрозумілим, користувачі не зможуть ефективно використовувати систему. Зрозумілий і простий дизайн підвищує доступність застосунку для людей різного віку та рівня цифрової грамотності. Основними перевагами цього пункту в постановці задач є те, що це зменшує ймовірність помилок при введені даних, може збільшувати частоту й регулярність використання системи, полегшує сприйняття інформарції завдяки графікам та зручним кольорам.

- Гарантувати надійний захист конфіденційної персональної інформації.

Медичні дані є одними з найбільш чутливих персональних даних. Їхній витік або несанкціонований доступ може призвести до серйозних юридичних, етичних і психологічних наслідків для пацієнтів. Захист таких даних є обов'язковою вимогою згідно з міжнародними стандартами. Цей пункт допоможе підвищити довіру користувачів до системи, забезпечує відповідність національним і міжнародним вимогам та запобігає можливим кібер-атакам. Хешування паролів, HTTPS з'єднання, аутентифікація та авторизація юзерів – на усе це повинен робитись акцент при розробці застосунку.

Висновок до розділу 1

Перший розділ присвячено аналізу аналогів і підходів до розробки систем для контролю здоров'я хворих на ЦД1. Огляд існуючих рішень, таких як Dexcom G6, MySugr, OpenAPS і ECO3, виявив їхні сильні сторони, зокрема інтеграцію з обладнанням і аналітичні можливості, а також слабкі сторони, такі

як висока вартість, відсутність локалізації українською мовою та складність використання. Аналіз підходів до розробки медичних веб-додатків показав переваги клієнт-серверної та мікросервісної архітектури, а також перспективність використання API, таких як OpenAI GPT-3.5, і стандартів безпеки, таких як HIPAA і GDPR. Отримані висновки слугують теоретичною базою для подальшої реалізації пропонованого веб-додатку, який має усунути виявлені недоліки аналогів.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

В цьому розділі розглянемо засоби для реалізації веб-застосунку для контролю здоров’я хворих на цукровий діабет першого типу. Для зручності повинні використовуватись сучасні технології, які забезпечують функціональність, безпеку та зручність для користувачів. Застосунок дозволятиме фіксувати дані про рівень глюкози, дози інсуліну, споживання вуглеводів, фізичну активність і симптоми, а також надавати аналітичні звіти та персоналізовані рекомендації українською мовою. Цей розділ присвячено опису архітектури системи, використаних технологій і бібліотек, процесу розробки та конфігурації середовища, які лягатимуть в основу реалізації застосунку.

2.1 Загальна архітектура системи

Веб-додаток побудовано на основі монолітної архітектури з клієнт-серверним підходом. Такий вибір забезпечує простоту розробки та підтримки для проєкту, орієнтованого, перш за все, на локальний ринок. Архітектура складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних, які взаємодіють через стандартизовані протоколи. [13]

Клієнтська частина реалізована як веб-інтерфейс, доступний у браузерах, і призначена для введення даних користувачами, перегляду аналітики та отримання рекомендацій. Вона базується на HTML, CSS і JavaScript із використанням фреймворку Bootstrap для адаптивного дизайну. Серверна частина, побудована на фреймворку ASP.NET Core, обробляє HTTP-запити, виконує бізнес-логіку та взаємодіє з базою даних. Як база даних використовується хмарна MongoDB Atlas, що забезпечує гнучке зберігання JSON-подібних документів. Для аналізу даних щоденника та генерації рекомендацій інтегровано API OpenAI. [13]

Комунікація між компонентами здійснюється через протокол HTTPS, який забезпечує шифрування даних для безпеки. Для збереження стану авторизації користувачів використовується механізм сесій ASP.NET Core, що дозволяє зберігати ідентифікатор користувача між запитами.[15] Архітектура підтримує асинхронну обробку запитів, підвищуючи продуктивність при роботі з великою кількістю користувачів. Серверна частина структурована на шари — контролери, сервіси та репозиторії, що відповідає принципам SOLID і полегшує підтримку та розширення коду [14]. Наприклад, контролер DiaryController.cs обробляє запити на додавання записів, а сервіс DiaryService.cs реалізує логіку оцінки стану здоров'я через метод CalculateDiseaseState. Схему архітектури, див. рис. 2.1

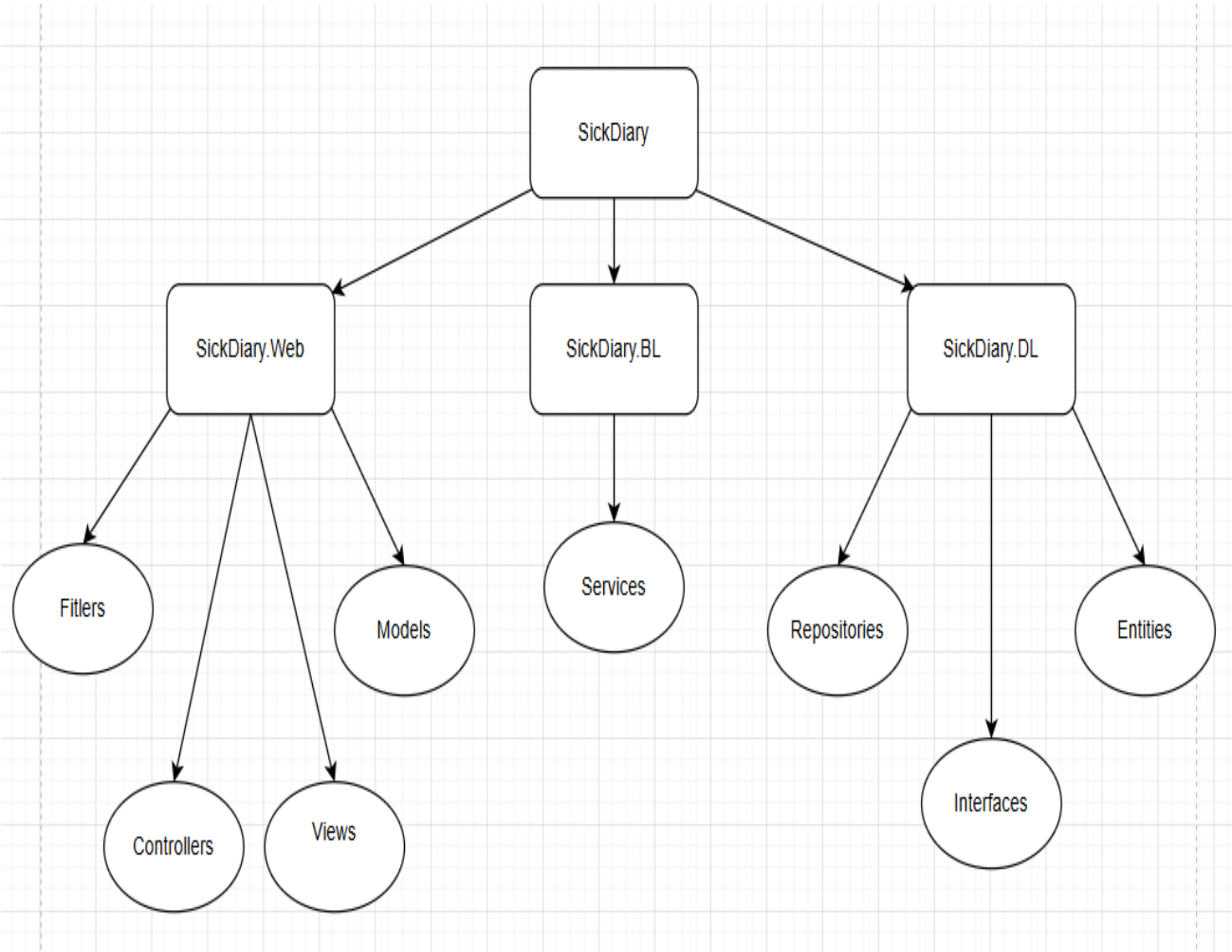

Рисунок 2.1 - Діаграма структури проєкту

2.2 Опис технологій і бібліотек

Реалізація веб-додатку базується на сучасному стеку технологій, які забезпечують продуктивність, безпеку та зручність розробки. Нижче детально описано кожну технологію та бібліотеку, використану в проєкті, з акцентом на їхні можливості та застосування.

Мова програмування C# є основою для написання серверної логіки додатку. Розроблена компанією Microsoft у 2000 році, C# поєднує простоту з потужними можливостями, такими як строга типізація, об'єктно-орієнтоване програмування та підтримка асинхронного програмування через ключові слова async і await.[14] Ці особливості роблять C# популярною для створення медичних систем, де важлива надійність обробки даних, наприклад, рівня глюкози чи доз інсуліну. У проєкті C# використовується для реалізації сервісів, таких як DiaryService, ClientService і GptService, а також контролерів, таких як HomeController і DiaryController. Наприклад, у DiaryService.cs метод CalculateDiseaseState застосовує правилові алгоритми для класифікації стану здоров'я на основі порогових значень глюкози (3.9–7.2 ммоль/л натщесерце) [1].

Фреймворк ASP.NET Core на основі платформі .NET 9.0 забезпечує серверну частину додатку. Фреймфорк є кросплатформенним, тому має можливість розгортання на Windows, Linux і macOS, що робить його універсальним для різних середовищ. Він базується на моделі MVC (Model-View-Controller), де модель (Record.cs) визначає структуру даних, вигляд (Index.cshtml, Charts.cshtml) відповідає за відображення, а контролер (DiaryController.cs) обробляє запити.[15] MVC сприяє чіткому розділенню логіки, полегшуючи тестування та підтримку. Також фреймворк підтримує маршрутизацію, налаштовану у файлі Program.cs, і асинхронну обробку HTTP-запитів, що підвищує продуктивність. Порівняно з попередніми версіями, такими як ASP.NET Framework, ASP.NET Core пропонує вищу швидкість і

модульність завдяки вбудованому контейнеру залежностей і підтримці HTTPS [19].

Для зберігання даних використовується хмарна база даних MongoDB Atlas, яка підтримує документо-орієнтовану модель. MongoDB зберігає дані у форматі JSON-подібних документів, що дозволяє гнучко працювати з сутностями Client, Diary і Record. Наприклад, запис у колекції records містить поля для глюкози, інсуліну, вуглеводів, активності та симптомів. Бібліотека MongoDB.Driver забезпечує асинхронний доступ до бази через клас MongoRepository, який реалізує операції додавання, оновлення та видалення документів [16]

Захист паролів користувачів здійснюється за допомогою бібліотеки BCrypt.Net-Next, яка реалізує адаптивне хешування алгоритмом BCrypt у класі ClientService. Це забезпечує стійкість до атак brute-force. [20]

Клієнтська частина використовує бібліотеку Chart.js (підключена через CDN) для створення інтерактивних графіків. Chart.js підтримує різні типи діаграм, такі як лінійні, стовпчикові та кругові, і дозволяє налаштовувати кольори, мітки та анімації. У моєму випадку, можемо спостерігати, що в Charts.cshtml реалізовано лінійні графіки для відображення рівня глюкози та доз інсуліну за останні 30 днів, що допомагає користувачам аналізувати тренди. Наприклад, глюкоза відображається зеленим кольором , а інсулін — червоним. Chart.js вирізняється легкістю інтеграції та низькими вимогами до ресурсів порівняно з бібліотеками, такими як D3.js [18].

Фреймворк Bootstrap. Розроблений у 2011 році компанією Twitter забезпечує адаптивне оформлення інтерфейсу. Bootstrap пропонує готові компоненти, такі як навігаційні панелі, форми, кнопки та таблиці, які адаптуються до різних розмірів екранів. У файлах _Layout.cshtml і Index.cshtml використано класи Bootstrap для створення зручного інтерфейсу. Bootstrap також підтримує темну та світлу теми, що відповідає стандартам доступності WCAG 2.1, забезпечуючи комфорт для користувачів із різними потребами [21,19]. Інтерфейс і аналітика локалізовані для української мови з

використанням культури uk-UA, що гарантує форматування дат і текстовий вивід українською.

Під’єднана API OpenAI (модель gpt-3.5-turbo) дозволяє аналізувати до 10 останніх записів щоденника та генерувати рекомендації українською мовою. Клас GptService реалізує HTTP-запити до API, використовуючи ключ, збережений у файлі appsettings.json. [28] Наприклад, якщо глюкоза стабільно висока (>10.0 ммоль/л), API може порекомендувати консультацію з лікарем.

Для серіалізації та десеріалізації даних у запитах до OpenAI застосовується бібліотека System.Text.Json, що забезпечує ефективну обробку JSON-структур. [14]

2.3 Процес розробки та інструменти

Процес розробки веб-додатку структуровано за ітераційним підходом, що включає планування, кодування, тестування та рефакторинг. Кожна ітерація охоплювала окремий функціональний модуль, наприклад, систему аутентифікації, управління щоденником або інтеграцію з API OpenAI [13].

Основним середовищем розробки було Visual Studio 2022, яке використовувалося для написання коду на C#, налаштування проєктів ASP.NET Core і локального тестування. Visual Studio пропонує інструменти для налагодження, автодоповнення коду та інтеграції з Git, що підвищує продуктивність. Для версійного контролю застосовано систему Git із хостингом на GitHub, що забезпечило відстеження змін і співпрацю над кодом [22]. Конфігурація проєкту, включаючи підключення до MongoDB і API OpenAI, зберігалася у файлі appsettings.json, що спрощувало управління параметрами.

Для забезпечення якості коду проводилося ручне тестування функціональності, зокрема аутентифікації, введення записів щоденника та аналізу даних. Рефакторинг коду виконувався для підтримки модульності та відповідності принципам SOLID. Наприклад, методи в DiaryService.cs було

оптимізовано для зменшення дублювання коду. Такий підхід дозволив створити стабільний і зручний у підтримці додаток.[14]

2.4 Конфігурація середовища розробки

Конфігурація середовища розробки була ключовим етапом для забезпечення продуктивності та стабільності роботи над проєктом. Для роботи з C# і ASP.NET Core встановлено платформу .NET 9.0 SDK, яка підтримує останні оновлення фреймворку, включаючи покращення продуктивності та безпеки. .NET 9.0 забезпечує підтримку асинхронного програмування, оптимізовану маршрутизацію та вбудовані інструменти для роботи з HTTPS, що критично для медичних додатків [30].

Хмарна база даних MongoDB Atlas використовувалася для зберігання даних, з підключенням через рядок, указаний у файлі Program.cs. MongoDB Atlas пропонує автоматичне масштабування, резервне копіювання та шифрування даних, що спрощує адміністрування порівняно з локальними серверами MongoDB [16].

Для інтеграції з API OpenAI ключ API було отримано через офіційний портал OpenAI і збережено у файлі appsettings.json. Операційна система Windows 11 слугувала основним середовищем розробки завдяки зручній інтеграції з Visual Studio. Для локального запуску додатку використовувалися профілі HTTP і HTTPS, налаштовані у файлі launchSettings.json, що забезпечувало шифрування трафіку під час тестування. Безпека сесій ASP.NET Core налаштована з використанням параметрів HttpOnly і IsEssential, що захищає кукі сесій від несанкціонованого доступу. [15]

Локалізація додатку для української мови реалізована через культуру uk-UA, що вплинуло на форматування дат у графіках (Charts.cshtml) і текст рекомендацій від OpenAI API. Наприклад, дати відображаються у форматі «19.05.2025», а рекомендації генеруються українською, що усуває недоліки аналогів, таких як Dexcom G6 чи MySugr [5, 6]. Чутливі дані, такі як ключ API

OpenAI, зберігалися у файлі appsettings.json, із планом перенесення до змінних середовища для продакшену, що відповідає рекомендаціям OWASP. Така конфігурація середовища забезпечила стабільну розробку та тестування додатку, створюючи основу для подальшого вдосконалення. [17]

Висновок до розділу 2

У другому розділі ми детально описав засоби реалізації веб-додатку для контролю здоров’я хворих на ЦД1. Монолітна клієнт-серверна архітектура з ASP.NET Core, MongoDB Atlas і інтеграцією з API OpenAI забезпечує ефективність і модульність. Технології, такі як C#, Bootstrap, Chart.js і MongoDB.Driver, разм із локалізацією для української мови, створюють надійне та зручне рішення. Процесс розробки із використанням Visual Studio 2022 і Git гарантує високу якість коду. Конфігурація середовища з .NET 9.0, HTTPS і безпечними сесіями забезпечує стабільність і безпеку. Ці засоби створюють міцну основу для подальшого обґрунтування вибору технологій і реалізації додатку, описаних у наступних розділах.

РОЗДІЛ 3
ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ДЛЯ ВЕБ-ЗАСТОСУНКУ

Розробка веб-додатку для управління цукровим діабетом першого типу потребує програмного середовища, яке забезпечує надійність, безпеку, локалізацію українською мовою та підтримку аналітичних функцій. Веб-додаток дозволяє користувачам фіксувати дані про рівень глюкози, дози інсуліну, вуглеводи, фізичну активність і симптоми, а також отримувати звіти та персоналізовані рекомендації. У попередніх розділах було описано аналоги, архітектуру та засоби реалізації, тоді як цей розділ зосереджується на обґрунтуванні вибору програмного середовища — мови програмування C#, фреймворку ASP.NET Core, бази даних MongoDB, бібліотек Bootstrap і Chart.js, а також API OpenAI. Обґрунтування базується на перевагах технологій, їхній відповідності вимогам проєкту та порівнянні з альтернативами.

3.1 Обґрунтування вибору мови програмування C#

C# обрано як основну мову програмування для серверної логіки веб-додатку завдяки її характеристикам, які відповідають вимогам медичних систем. Строга типізація C# забезпечує надійність обробки даних, що дозволяє створювати стабільні алгоритми для оцінки стану здоров’я. Це зменшує ризик помилок, що є критично важливим для медичних застосунків, де точність даних впливає на рекомендації користувачам. [14]

Підтримка асинхронного програмування дозволяє C# ефективно обробляти запити до хмарних сервісів, таких як бази даних чи API, що забезпечує швидке виконання аналітичних операцій, наприклад, генерацію звітів про рівень глюкози. Екосистема .NET 9.0 надає бібліотеки для безпеки (хешування паролів), серіалізації даних і взаємодії з базами, що спрощує

створення захищених і масштабованих рішень. Об’єктно-орієнтоване програмування сприяє модульності, дозволяючи легко додавати нові функції. [30]

Активна спільнота та регулярні оновлення від Microsoft забезпечують довгострокову підтримку C#, що важливо для медичних проєктів, які потребують стабільності. Для порівняння, Python має простіший синтаксис, але нижчу продуктивність через інтерпретовану природу, а Node.js із динамічною типізацією підвищує ризик помилок у медичних даних, див. табл. 3.1, яка підсумовує ключові аспекти. [23]

Таблиця 3.1

Порівняння C#, Python і Node.js

Критерій	**C#**	**Python**	**Node.js**
Типізація	Строга	Динамічна	Динамічна
Продуктивність	Висока (компіляція)	Середня (інтерпретація)	Висока (асинхронність)
Безпека даних	Висока (типізація, .NET)	Середня	Низька (типізація)
Підтримка медичних систем	Висока (надійність)	Середня (аналітика)	Низька (типізація)

C# обрано через надійність, продуктивність і підтримку .NET, що забезпечують стабільну обробку даних ЦД1.

3.2 Обґрунтування вибору фреймворку ASP.NET Core

ASP.NET Core обрано для серверної частини завдяки його можливостям, які забезпечують продуктивність, безпеку та структуровану розробку. Модель MVC (Model-View-Controller) дозволяє організувати логіку обробки запитів,

моделі даних і відображення, що спрощує створення складних функцій, таких як аналітика глюкози чи рекомендації. [15]

ASP.NET Core підтримує вбудовані механізми безпеки, зокрема HTTPS для шифрування трафіку та управління сесіями, що відповідає стандартам GDPR і HIPAA. Це забезпечує захист конфіденційних даних, таких як записи щоденника. Кросплатформність дозволяє розгортати додаток на Linux-серверах, знижуючи витрати. Локалізація з підтримкою культури uk-UA забезпечує форматування дат і тексту українською, що підвищує зручність для місцевих користувачів. [11,12].

Для порівняння, Django (Python) пропонує швидку розробку, але має нижчу продуктивність, а Express.js (Node.js) потребує додаткових бібліотек для безпеки, що ускладнює підтримку, див. табл. 3.2. [24]

Таблиця 3.2

Порівняння ASP.NET Core, Django і Express.js

Критерій	**ASP.NET Core**	**Django**	**Express.js**
Модель розробки	MVC	MTV	Без чіткої моделі
Продуктивність	Висока (Kestrel)	Середня	Висока (асинхронність)
Безпека	Вбудована (HTTPS, сесії)	Вбудована (CSRF, XSS)	Потребує бібліотек
Локалізація	Висока (uk-UA)	Висока	Середня

ASP.NET Core обрано через продуктивність, безпеку та підтримку локалізації, що відповідають вимогам ЦД1.

3.3 Обґрунтування вибору бази даних MongoDB

MongoDB Atlas обрано як хмарну базу даних завдяки її документо-орієнтованій моделі, яка підходить для гнучкого зберігання даних щоденника ЦД1. Формат JSON/BSON дозволяє ефективно працювати з

неструктурованими даними, такими як глюкоза, інсулін чи симптоми, і легко додавати нові поля, наприклад, для аналізу фізичної активності. Хмарна інфраструктура Atlas забезпечує масштабування, резервне копіювання та шифрування, що відповідає стандартам безпеки для медичних даних [16].

Асинхронні операції MongoDB підвищують продуктивність, дозволяючи швидко отримувати дані для аналітики чи рекомендацій. Географічно розподілені сервери Atlas забезпечують швидкий доступ для українських користувачів, що важливо в умовах нестабільного інтернету. Простота інтеграції з ASP.NET Core спрощує обробку даних, підвищуючи ефективність розробки. [16]

Порівняно з реляційною базою PostgreSQL, MongoDB пропонує більшу гнучкість для неструктурованих даних, що краще відповідає вимогам проєкту, див. табл. 3.3. [25].

Таблиця 3.3

Порівняння MongoDB і PostgreSQL

Критерій	**MongoDB**	**PostgreSQL**
Тип бази	Документо-орієнтована	Реляційна
Гнучкість структури	Висока (JSON/BSON)	Середня (схема)
Масштабування	Горизонтальне (Atlas)	Вертикальне
Безпека	Шифрування (Atlas)	Шифрування

MongoDB обрано через гнучкість, масштабування та безпеку, що забезпечують ефективне зберігання даних ЦД1.

3.4 Обґрунтування вибору бібліотеки Bootstrap

Bootstrap обрано як основу для створення клієнтської частини веб-додатку завдяки його можливостям, які забезпечують адаптивний дизайн, зручність інтерфейсу та підтримку локалізації. Однією з ключових переваг Bootstrap є його адаптивна сітка, яка дозволяє інтерфейсу коректно

відображатися на пристроях із різними розмірами екранів — від смартфонів до настільних комп’ютерів. Це критично для користувачів ЦД1, які можуть вводити дані про глюкозу чи переглядати звіти в різних умовах, наприклад, у дорозі чи вдома.[21]

Bootstrap пропонує широкий набір готових компонентів, таких як навігаційні панелі, форми введення, таблиці та кнопки, що прискорюють розробку інтерфейсу. Ці компоненти забезпечують однаковий стиль і функціональність, що підвищує зручність для користувачів, зокрема для осіб із обмеженим досвідом роботи з цифровими технологіями, таких як літні люди з ЦД1. Наприклад, форми для введення даних про глюкозу чи інсулін мають інтуїтивний вигляд, що спрощує їх заповнення.

Підтримка локалізації є ще однією перевагою Bootstrap. Бібліотека дозволяє легко адаптувати інтерфейс до культури uk-UA, забезпечуючи коректне відображення тексту, дат і числових форматів українською мовою. Це усуває недоліки аналогів, таких як Dexcom G6, які мають обмежену локалізацію [5]. Bootstrap також відповідає стандартам доступності WCAG 2.1, пропонуючи контрастні теми (світлу та темну) і підтримку клавіатурної навігації, що робить додаток зручним для користувачів із порушеннями зору чи моторики [19].

Гнучкість Bootstrap дозволяє налаштовувати стилі через CSS, що забезпечує візуальну відповідність медичній тематиці, наприклад, використання спокійних кольорів для форм і графіків. Активна спільнота та регулярні оновлення гарантують сумісність із сучасними браузерами, що важливо для забезпечення доступності додатку. Для порівняння, Tailwind CSS пропонує більшу гнучкість у кастомізації, але потребує більше часу на розробку через відсутність готових компонентів, див. табл. 3.4. [26]

Таблиця 3.4

Порівняння Bootstrap і Tailwind CSS

Критерій	**Bootstrap**	**Tailwind CSS**
Готові компоненти	Широкий набір	Обмежені (утиліти)
Час розробки	Швидкий	Повільніший
Локалізація	Висока (uk-UA)	Висока
Доступність (WCAG 2.1)	Висока	Середня

Bootstrap обрано через швидкість розробки, локалізацію та доступність, що відповідають потребам користувачів ЦД1.

3.5 Обґрунтування вибору бібліотеки Chart.js

Chart.js обрано для візуалізації даних у веб-додатку завдяки його можливостям, які забезпечують інтерактивність і простоту створення аналітичних графіків. Бібліотека підтримує різні типи діаграм — лінійні, стовпчикові, кругові, — що дозволяє відображати дані про глюкозу, інсулін чи вуглеводи у зрозумілому форматі. Наприклад, лінійні графіки допомагають користувачам бачити тренди рівня глюкози за 30 днів, що сприяє кращому управлінню ЦД1.[18]

Однією з переваг Chart.js є легкість інтеграції з веб-додатками. Бібліотека працює через JavaScript і CDN, що зменшує залежності та спрощує розгортання. Chart.js дозволяє налаштовувати кольори, наприклад, rgb(75, 192, 192) (зелений) для глюкози та rgb(255, 99, 132) (червоний) для інсуліну, що полегшує розрізнення даних. Інтерактивні елементи,спливаючі підказки, масштабування, роблять графіки зручними для аналізу.[18]

Chart.js має низькі вимоги до ресурсів, що забезпечує швидке завантаження графіків навіть на бюджетних пристроях, якими можуть користуватися люди з ЦД1. Бібліотека підтримує локалізацію, дозволяючи

форматувати мітки та дати в культурі uk-UA, що відповідає вимогам української аудиторії. Активна спільнота та документація забезпечують підтримку та оновлення, що гарантує сумісність із сучасними браузерами [18].

Для порівняння, D3.js пропонує більшу гнучкість у створенні кастомних візуалізацій, але потребує складнішого кодування та більших ресурсів, що може сповільнити розробку, див. табл. 3.5. [27].

Таблиця 3.5

Порівняння Chart.js і D3.js

Критерій	**Chart.js**	**D3.js**
Простота інтеграції	Висока (CDN)	Низька (складний код)
Ресурси пристрою	Низькі	Високі
Локалізація	Висока (uk-UA)	Середня
Швидкість розробки	Швидка	Повільна

Chart.js обрано через простоту, інтерактивність і локалізацію, що відповідають аналітичним потребам ЦД1.

3.6 Обґрунтування вибору API OpenAI

API OpenAI (модель gpt-3.5-turbo) обрано для генерації персоналізованих рекомендацій завдяки його можливостям обробки даних і генерації тексту українською мовою. API аналізує дані щоденника, такі як глюкоза, інсулін, вуглеводи та симптоми, і формує рекомендації, наприклад, поради щодо коригування дози інсуліну при високій глюкозі (>10.0 ммоль/л). Це дозволяє надавати користувачам адаптивні поради без необхідності складних локальних обчислень. [28]

Однією з ключових переваг OpenAI API є підтримка української мови, що забезпечує генерацію зрозумілих і культурно адаптованих рекомендацій. Це усуває недоліки аналогів, таких як MySugr, які мають обмежену локалізацію [5]. API працює через HTTP-запити, що спрощує інтеграцію з ASP.NET Core і дозволяє швидко отримувати відповіді, підвищуючи зручність для користувачів.[15]

OpenAI API є хмарним рішенням, що зменшує навантаження на сервер додатку та спрощує масштабування. Висока точність обробки даних дозволяє враховувати складні взаємозв'язки, наприклад, між глюкозою, фізичною активністю та симптомами, що покращує якість рекомендацій. Безпека забезпечується через шифрування запитів і захист API-ключів, що відповідає стандартам GDPR [11,28].

Для порівняння, локальні алгоритми потребують значних ресурсів для розробки та не забезпечують такої гнучкості в генерації текстових рекомендацій українською, див. табл. 3.6.

Таблиця 3.6

Порівняння OpenAI API і локальних алгоритмів

Критерій	**OpenAI API**	**Локальні алгоритми**
Локалізація	Висока (uk-UA)	Обмежена
Гнучкість рекомендацій	Висока	Низька
Ресурси розробки	Низькі (хмарні)	Високі
Безпека	Шифрування	Залежить від реалізації

OpenAI API обрано через локалізацію, гнучкість і простоту інтеграції, що відповідають потребам ЦД1.

3.7 Структура класів проєкту

Для забезпечення логічної, зрозумілої та масштабованої архітектури веб-застосунку *SickDiary* було реалізовано об'єктно-орієнтовану модель із чітким

розподілом відповідальностей між класами. Кожен клас у системі виконує окрему роль — зберігання даних (моделі), обробка бізнес-логіки (сервіси), взаємодія з базою даних (репозиторії), або представлення користувацького інтерфейсу.

Нижче наведено узагальнену діаграму класів, що відображає основні сутності системи, їх атрибути, методи, а також зв’язки між ними. Ця структура забезпечує гнучкість у розробці, полегшує підтримку коду та дає змогу легко розширювати функціональність у майбутньому, див. рис. 3.1

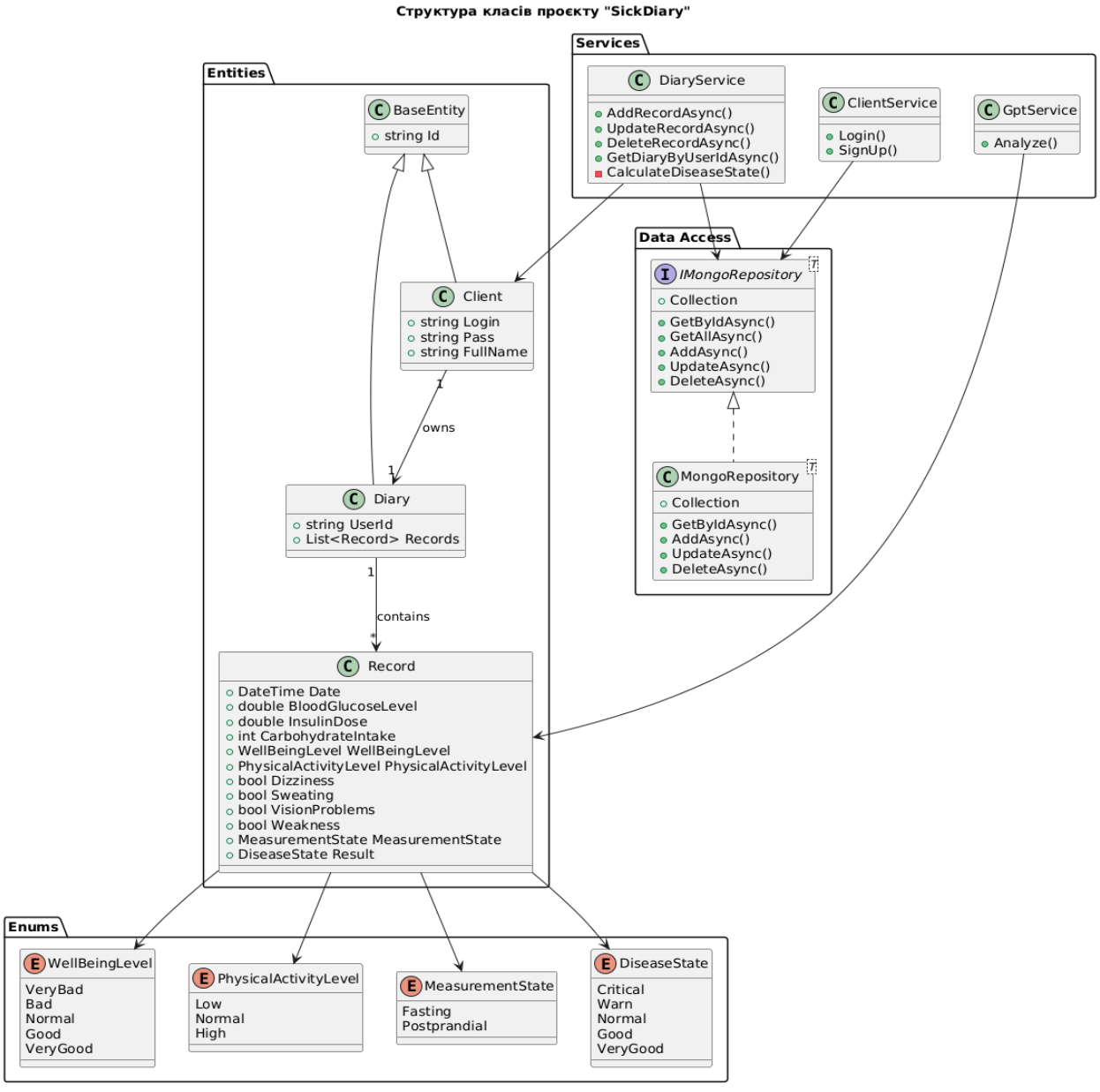

Рисунок 3.1 – Структура класів проєкту

3.8 Проєктування бази даних

База даних є центральним компонентом веб-застосунку *SickDiary*, оскільки саме вона зберігає всю критично важливу інформацію — облікові дані користувачів, їх щоденники, а також записи про стан здоров’я. Для реалізації було обрано MongoDB, документно-орієнтовану нереляційну СУБД, яка ідеально підходить для зберігання вкладених структур даних і забезпечує високу гнучкість, масштабованість і продуктивність.

Модель MongoDB дозволяє зберігати дані в одному документі, що відповідає одному користувачу, разом із усіма пов’язаними щоденниковими записами. Такий підхід спрощує архітектуру, пришвидшує доступ до даних та зменшує кількість запитів до бази.

У процесі проєктування бази даних було визначено дві основні сутності — **Client** (користувач) та **Diary** (щоденник). Кожен користувач має один щоденник, який містить масив записів про щоденний стан здоров’я. Записи включають такі параметри, як рівень глюкози в крові, доза інсуліну, кількість спожитих вуглеводів, самопочуття, рівень фізичної активності та супутні симптоми.

Усі записи (records) реалізовані як вкладений масив об’єктів усередині документа Diary. Така структура дозволяє зберігати всі дані одного користувача в одному документі, що відповідає підходу до денормалізації в NoSQL-системах і спрощує читання та оновлення даних, див. рис. 3.2

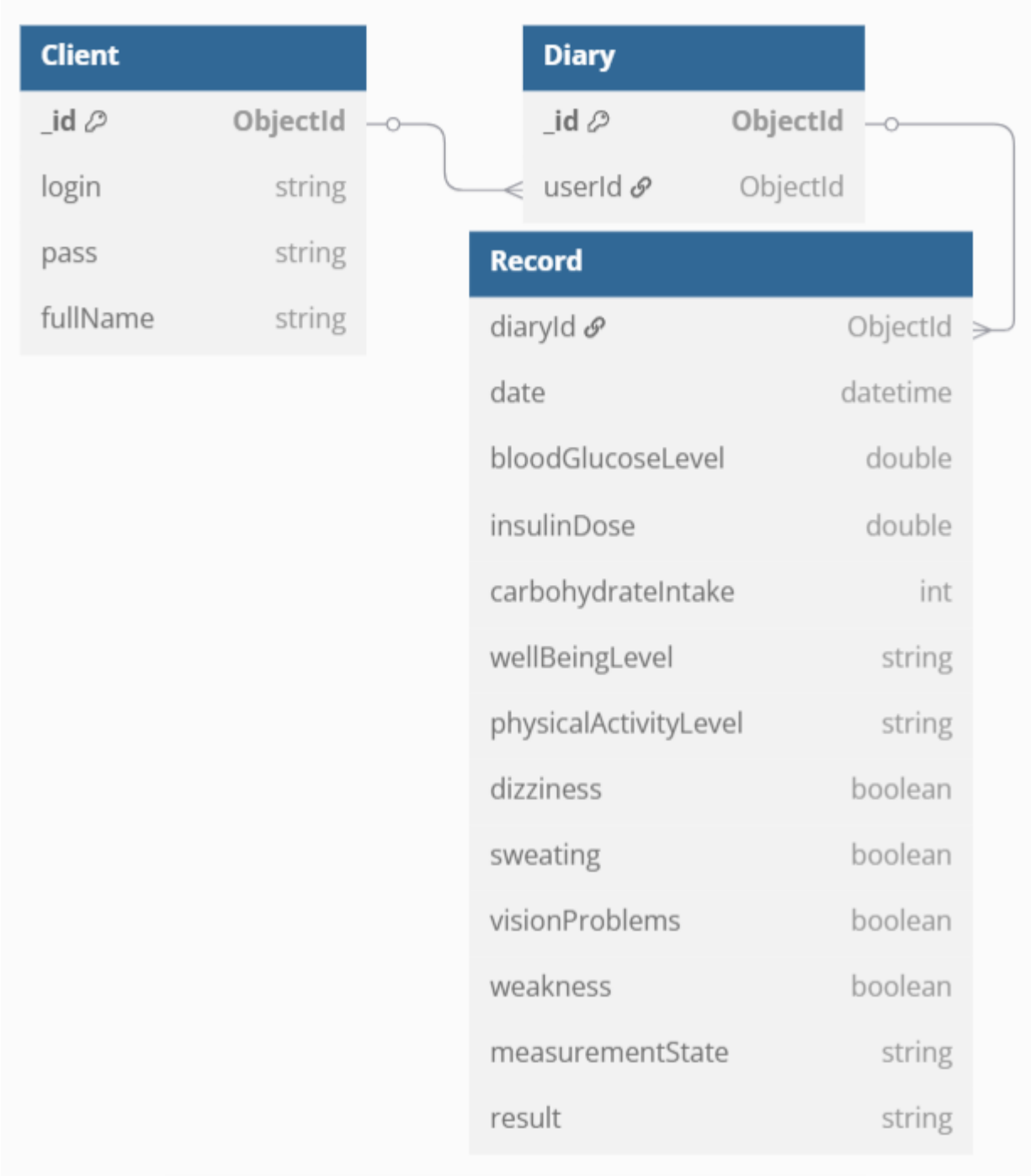

Рисунок 3.2 – Проєктування БД

3.9 Відповідність вимогам управління ЦД1

Обране програмне середовище — C#, ASP.NET Core, MongoDB, Bootstrap, Chart.js і OpenAI API — відповідає ключовим вимогам веб-додатку для управління ЦД1: локалізації, безпеки та аналітичним можливостям.

Локалізація: Підтримка культури uk-UA забезпечує форматування дат, чисел і тексту українською мовою. Bootstrap і Chart.js дозволяють адаптувати інтерфейс і графіки до української аудиторії, наприклад, відображення дат у форматі «19.05.2025». OpenAI API генерує рекомендації українською, що усуває недоліки аналогів, таких як Dexcom G6 і MySugr, які мають обмежену локалізацію. Це підвищує зручність для українських користувачів, які можуть взаємодіяти з додатком рідною мовою. [28]

Безпека: Програмне середовище відповідає стандартам GDPR і HIPAA. ASP.NET Core забезпечує шифрування трафіку через HTTPS і захист сесій, що гарантує конфіденційність даних, таких як глюкоза чи симптоми. MongoDB Atlas використовує шифрування та резервне копіювання, що захищає інформацію від втрати чи несанкціонованого доступу. Хешування паролів через бібліотеку BCrypt підвищує безпеку авторизації, а захист API-ключів OpenAI забезпечує безпечну інтеграцію. Ці заходи створюють надійне середовище для роботи з чутливими медичними даними. [15]

Аналітичні можливості: C# і ASP.NET Core дозволяють створювати стабільні алгоритми для оцінки стану здоров’я, наприклад, на основі порогових значень глюкози (3.9–7.2 ммоль/л) [2]. MongoDB забезпечує швидке зберігання та отримання даних для аналізу, що підтримує побудову звітів. Chart.js дозволяє створювати інтерактивні графіки, які допомагають користувачам візуалізувати тренди, наприклад, зміни рівня глюкози чи доз інсуліну. OpenAI API доповнює аналітику, надаючи персоналізовані рекомендації, які враховують складні взаємозв’язки між даними.[30]

Ці можливості усувають недоліки аналогів, таких як відсутність локалізації, обмежені аналітичні функції чи недостатня безпека, і створюють ефективне рішення для управління ЦД1. Програмне середовище підтримує гнучкість для майбутніх розширень, наприклад, інтеграції з глюкометрами чи аналізу додаткових показників, що відповідає довгостроковим потребам проєкту.

Висновки до розділу 3

Обґрунтування вибору програмного середовища показало, що C#, ASP.NET Core, MongoDB, Bootstrap, Chart.js і OpenAI API забезпечують надійність, безпеку та зручність для веб-додатку управління ЦД1. C# пропонує строгую типізацію і продуктивність, ASP.NET Core — структуровану розробку і безпеку, MongoDB — гнучке зберігання даних. Bootstrap забезпечує адаптивний інтерфейс із локалізацією uk-UA, Chart.js — інтерактивні графіки, а OpenAI API — персоналізовані рекомендації українською. Порівняння з альтернативами (Python, Node.js, Django, Express.js, PostgreSQL, Tailwind CSS, D3.js, локальні алгоритми) підтвердило переваги обраних технологій у продуктивності, безпеці та локалізації. Відповідність вимогам ЦД1 — локалізації, безпеки та аналітики — забезпечує створення ефективного рішення, яке усуває недоліки аналогів і створює основу для реалізації, описаної в наступному розділі.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ЗАСТОСУНКУ

У цьому розділі детально описано програмну реалізацію веб-додатку *SickDiary*, призначеного для автоматизованого ведення щоденника самоконтролю хворих на ЦД1. Проєкт побудовано за тришаровою архітектурою, що включає шари веб-застосунку (*SickDiary.Web*), бізнес-логіки (*SickDiary.BL*) і даних (*SickDiary.DL*). Спочатку розглянуто структуру та призначення кожного шару, їхню взаємодію та переваги такого поділу. Описано елементи інтерфейсу користувача, функціональні можливості, методику роботи додатку, інструменти реалізації та їхні особливості. Особливу увагу приділено локалізації українською мовою, безпеці даних і зручності для користувачів із ЦД1.

4.1 Тришарова архітектура додатку

Для забезпечення модульності, зрозумілості та легкості підтримки веб-додаток *SickDiary* структуровано за тришаровою архітектурою, яка відповідає принципам шарового дизайну та SOLID. Ця структура досить чітка, гнучка та здатна до масштабованості та спрощенню подальшого вдосконалення.

Архітектура включає три основні шари:

- *SickDiary.Web* (веб-застосунок)
- *SickDiary.BL* (бізнес-логіка)
- *SickDiary.DL* (дані)

Кожен шар виконує унікальні функції, забезпечуючи ізоляцію логіки, даних і представлення, що знижує залежність між компонентами та полегшує тестування.

4.1.1 SickDiary.Web – Веб-застосунок (UI + API)

Шар *SickDiary.Web* є точкою входу для користувачів, відповідаючи за взаємодію з ними через веб-інтерфейс і обробку запитів. Він включає кілька ключових компонентів, які забезпечують зручність і функціональність.

Інтерфейс користувача побудовано з використанням технології Razor Pages, яка дозволяє створювати динамічні сторінки на основі HTML, CSS і JavaScript. Основні сторінки, такі як головна сторінка щоденника, форма введення даних, сторінка графіків і сторінки аутентифікації, реалізовані через файли *Index.cshtml*, *CreateRecord.cshtml*, *Charts.cshtml*, *Login.cshtml* і *Register.cshtml*. Ці сторінки стилізовано за допомогою фреймворку Bootstrap 5, що забезпечує адаптивність інтерфейсу для різних пристроїв, включаючи смартфони, планшети та настільні комп'ютери. Адаптивний дизайн дозволяє коректно відображати форми, таблиці та графіки незалежно від розміру екрана, що є важливим для користувачів із ЦД1, які можуть використовувати додаток у різних умовах. [21]

Шар також включає контролери, зокрема *HomeController* і *DiaryController*, які обробляють HTTP-запити (GET, POST) від користувачів.

Наприклад, *HomeController* відповідає за відображення сторінок реєстрації та входу, тоді як *DiaryController* обробляє дії, пов'язані зі щоденником, такі як створення, редагування, видалення записів і генерація даних для графіків.

Контролери діють як посередники, викликаючи методи бізнес-логіки з шару *SickDiary.BL* і передаючи результати назад до інтерфейсу.

Основне призначення *SickDiary.Web* — забезпечити зрозумілий і локалізований інтерфейс, який відповідає культурі. Усі елементи інтерфейсу, включаючи назви меню, підказки у формах і повідомлення про помилки, відображаються українською мовою. Наприклад, навігаційна панель містить пункти «Мій щоденник», «Додати запис», «Графіки» та «Вихід», що полегшує використання для українських користувачів. Шар ізолює клієнтську частину

від складної логіки обробки даних, забезпечуючи чітку маршрутизацію запитів і відображення результатів.

4.1.2 SickDiary.BL – Бізнес-логіка

Шар *SickDiary.BL* відповідає за обробку основної логіки додатку, включаючи управління користувачами, щоденниками, оцінку стану здоров'я та інтеграцію із зовнішніми сервісами. Цей шар містить сервіси, які реалізують ключові операції. Сервіс для роботи з користувачами управляє процесами реєстрації та авторизації, забезпечуючи безпечне зберігання даних. Сервіс щоденника обробляє операції з записами, включаючи їх додавання, редагування, видалення та оцінку стану здоров'я. Сервіс формує запити до зовнішнього API для аналізу даних і повертає персоналізовані рекомендації.

Сутності даних, такі як інформація про користувачів, щоденники та окремі записи, визначено у вигляді моделей. Ці моделі включають поля для зберігання логіну, імені, рівня глюкози, дози інсуліну, кількості вуглеводів, рівня фізичної активності, самопочуття та симптомів. Моделі забезпечують структуровану обробку даних, дозволяючи сервісам працювати з об'єктами високого рівня, а не низькорівневими структурами бази даних.

Логіка обробки в *SickDiary.BL* реалізована через асинхронні операції, що підвищує продуктивність додатку. Наприклад, операція додавання запису включає валідацію даних, оцінку стану здоров'я та збереження в базі даних. Оцінка стану здоров'я базується на медичних нормах рівня глюкози (3.9–7.2 ммоль/л натщесерце, 5.0–10.0 ммоль/л після їжі) і враховує додаткові фактори, такі як симптоми, фізична активність і самопочуття. Інтеграція із зовнішнім сервісом дозволяє аналізувати дані та надавати рекомендації, наприклад, щодо коригування дози інсуліну чи звернення до лікаря.

Призначення *SickDiary.BL* — ізолювати бізнес-логіку від інтерфейсу та даних, забезпечуючи модульність і можливість повторного використання коду. Цей шар встановлює взаємодію між інтерфейсом і базою даних, виконуючи складні обчислення та обробку запитів. Завдяки асинхронній архітектурі він

ефективно обробляє великі обсяги даних, що важливо для користувачів із великою кількістю записів.

4.1.3 SickDiary.DL – Шар даних

Шар *SickDiary.DL* відповідає за взаємодію з хмарною базою даних MongoDB Atlas, яка використовується для зберігання всіх даних додатку. Цей шар включає компоненти для підключення до бази, виконання запитів і обробки даних. Дані користувачів, такі як логін, ім’я та захищений пароль, зберігаються в окремій колекції, див. рис. 4.1 тоді як записи щоденника, що містять інформацію про глюкозу, інсулін, вуглеводи, активність, самопочуття та симптоми, — в іншій, див. рис. 4.2 Кожна сутність ідентифікується унікальним ідентифікатором, що забезпечує швидкий доступ і зв’язок між даними.

```
_id: ObjectId('6807c951eb785518f827f2be')
login : "Testuser"
pass : "$2a$11$yy53d9EL3QGuGc02NlQgcOEYwVRZbAT4xVw4v5SWiM2kCraE2xdGy"
fullName : "Test User"

_id: ObjectId('6807c9f2eb785518f827f2c0')
login : "Makson4ik"
pass : "$2a$11$xgJCFMDmih/A3CXiOPFSQ.4gt0ztkhgl2Up2y0H33HWNEHMmcngJK"
fullName : "Maks Zasl"

_id: ObjectId('682336edc16a6866275f276e')
login : "MaksZaslavskiy"
pass : "$2a$11$w0FBRTMmOhh68z8/bz8ewuN1NHhlnuZZnXIDyGEkZ0JWkPRmU7jtq"
fullName : "Maks Zaslaskiy"
```

Рисунок 4.1 – Колекція даних користувачів в БД

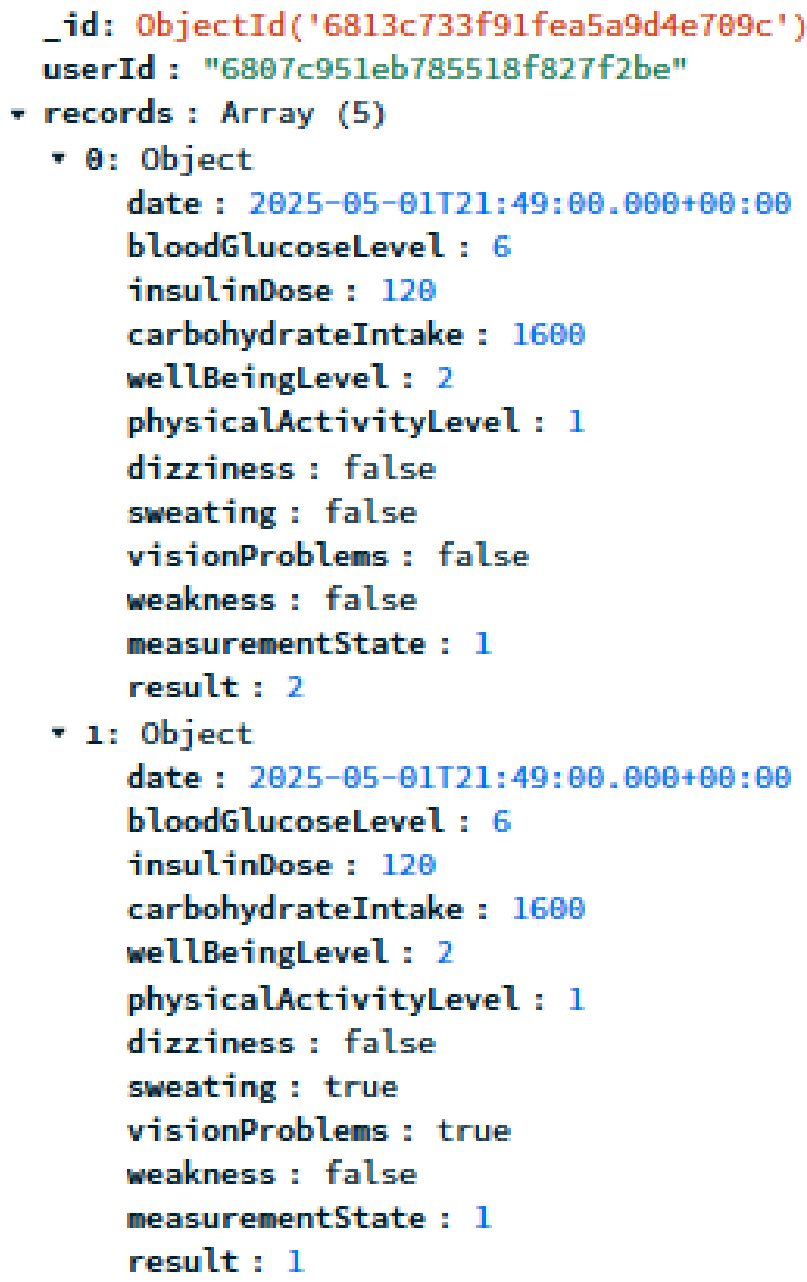

Рисунок 4.2 – Дані користувача в БД

Доступ до бази даних реалізовано через спеціальний компонент, який абстрагує низькорівневі операції, такі як додавання, оновлення, видалення та пошук записів. Цей компонент підтримує асинхронні операції, що підвищує швидкість обробки запитів. Наприклад, при додаванні нового запису дані форматуються у вигляді документа MongoDB і зберігаються в колекції. Аналогічно, при отриманні записів система фільтрує їх за ідентифікатором користувача, повертаючи лише релевантні дані.

Шар *SickDiary.DL* також відповідає за конвертацію форматів даних. Наприклад, дати введення записів автоматично переводяться в універсальний час (UTC), щоб забезпечити коректне відображення незалежно від часового поясу користувача. Це особливо важливо для хворих на ЦД1, які можуть вести щоденник у різних регіонах.

Основне призначення *SickDiary.DL* — ізолювати доступ до даних, забезпечуючи єдину точку взаємодії з базою. Це дозволяє бізнес-логіці працювати з абстрактними об’єктами, а не турбуватися про технічні деталі зберігання.

Тришарова архітектура забезпечує чітку взаємодію між компонентами: *SickDiary.Web* обробляє запити користувачів і відображає результати, *SickDiary.BL* виконує обчислення та координує логіку, а *SickDiary.DL* управляє даними. Такий підхід підвищує надійність, полегшує тестування та створює основу для подальшого вдосконалення додатку.

4.2 Елементи інтерфейсу та їх обробка

Інтерфейс користувача веб-додатку *SickDiary* розроблено з урахуванням потреб хворих на ЦД1, забезпечуючи доступність і швидкість. Локалізація українською мовою охоплює меню, форми, таблиці, графіки та повідомлення, що відповідає потребам українського ринку. Обробка запитів користувачів здійснюється через взаємодію шарів *SickDiary.Web*, *SickDiary.BL* і *SickDiary.DL*, забезпечуючи швидку та безпечну роботу.

4.2.1 Навігаційна панель

Навігаційна панель є центральним елементом інтерфейсу, забезпечуючи швидкий доступ до основних функцій додатку. Вона реалізована в основному макеті сторінок і динамічно адаптується залежно від стану авторизації користувача. Для неавторизованих користувачів панель відображає посилання на сторінки реєстрації та входу, дозволяючи їм створити обліковий запис або увійти в систему, див. рис. 4.3

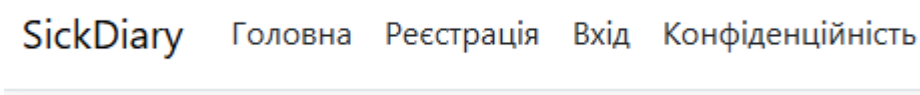

Рисунок 4.3 – Навігаційна панель до реєстрації / входу

Після авторизації панель змінюється, показуючи пункти меню «Мій щоденник» (перехід до списку записів), «Додати запис» (форма введення даних), «Графіки» (візуалізація даних) і «Вийти» (завершення сесії), див. рис. 4.4

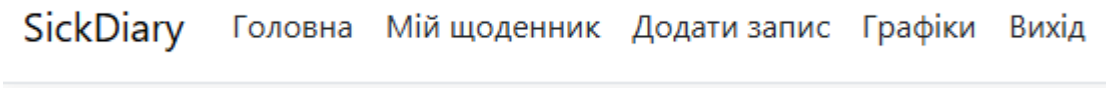

Рисунок 4.4 – Навігаційна панель після входу

Обробка навігаційних дій виконується через контролери в шарі *SickDiary.Web*. Наприклад, вибір пункту «Додати запис» викликає метод контролера, який повертає відповідну сторінку форми. Статус авторизації перевіряється через сесію, яка зберігає ідентифікатор користувача. Якщо сесія відсутня, користувач перенаправляється на сторінку входу.

4.2.2 Форми введення даних

Форми введення даних є ключовим інструментом для ведення щоденника. Основна форма дозволяє користувачам додавати нові записи, фіксуючи детальну інформацію про стан здоров’я. Користувач вводить дату і час вимірювання, рівень глюкози в крові (у ммоль/л), дозу інсуліну (у одиницях), кількість спожитих вуглеводів (у грамах), рівень фізичної активності (низька, середня, висока), самопочуття (дуже погане, погане, нормальне, гарне, дуже гарне) та симптоми, такі як запаморочення, пітливість, слабкість чи проблеми із зором. Поля для фізичної активності та самопочуття реалізовані як випадаючі списки, а симптоми — як прапорці, що спрощує вибір, див. рис. 4.5

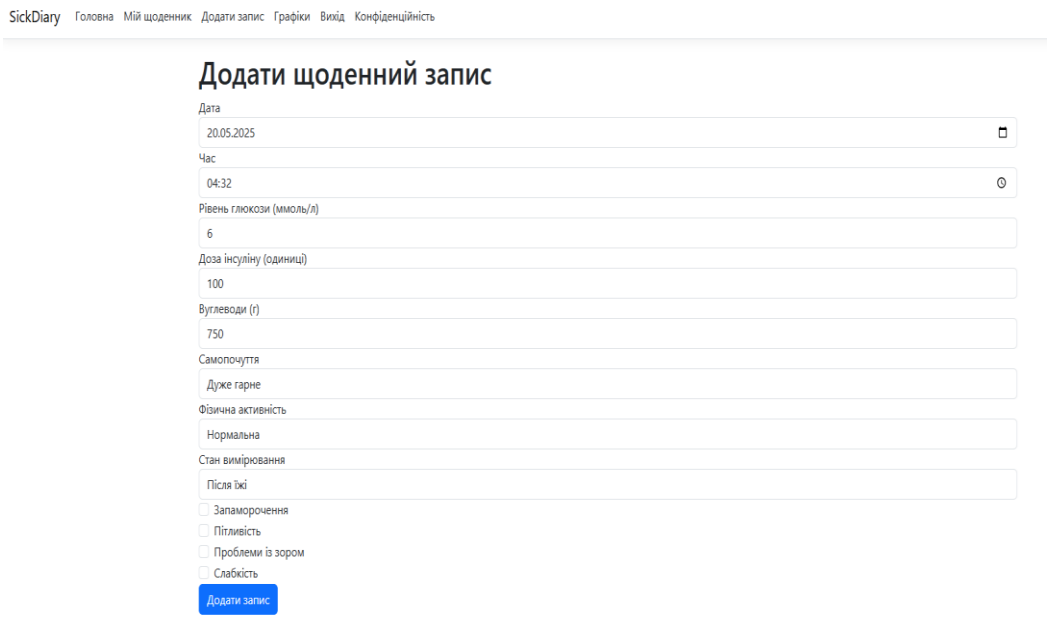

Рисунок 4.5 – Форма введення даних

Валідація даних виконується на двох рівнях. Клієнтська валідація, реалізована за допомогою JavaScript і атрибутів HTML5, перевіряє, чи заповнені обов’язкові поля (глюкоза, дата) і чи відповідають числові значення допустимим діапазонам (глюкоза 0–30 ммоль/л, інсулін 0–100 одиниць). У разі помилки відображається повідомлення, наприклад, «Поле ‘Рівень глюкози’ є обов’язковим». Серверна валідація, виконувана в шарі *SickDiary.Web*, додатково перевіряє дані, щоб запобігти обробці некоректних значень. Після успішної валідації дані надсилаються на сервер через HTTP POST-запит, де шар *SickDiary.BL* оцінює стан здоров’я, а *SickDiary.DL* зберігає запис у базі MongoDB Atlas.

Форма редагування записів працює аналогічно, але попередньо заповнюється даними існуючого запису. Користувач може змінити будь-яке поле, після чого оновлений запис зберігається в базі. Обидві форми стилізовано за допомогою Bootstrap, що забезпечує чітке розташування полів і адаптивність для різних пристроїв.

4.2.3 Таблиця записів

На головній сторінці щоденника відображається таблиця, яка містить усі записи користувача. Таблиця включає стовпці з датою і часом вимірювання, рівнем глюкози, дозою інсуліну, кількістю вуглеводів, рівнем фізичної активності, самопочуттям, симптомами та станом здоров'я (критичний, попереджувальний, нормальний, гарний, дуже гарний). Записи сортуються за датою у зворотному порядку, щоб найновіші відображалися зверху, див. рис. 4.6

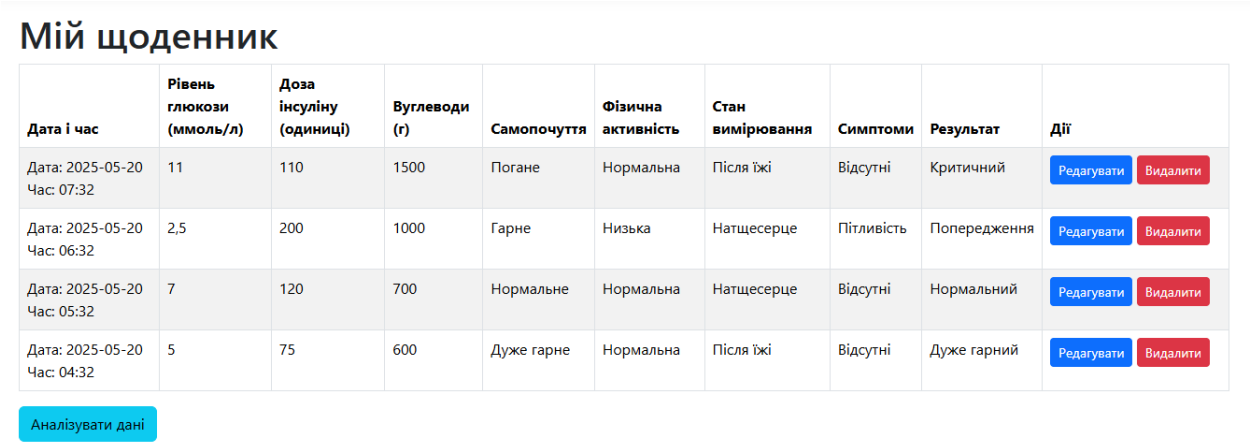

Рисунок 4.6 – Таблиця записів

Таблиця генерується динамічно на основі даних, отриманих із бази. Шар *SickDiary.Web* викликає методи *SickDiary.BL*, які, у свою чергу, звертаються до *SickDiary.DL* для отримання записів, відфільтрованих за ідентифікатором користувача. Кожен рядок таблиці містить кнопки «Редагувати» та «Видалити», які дозволяють користувачу змінити або видалити запис. Дія редагування відкриває форму з попередньо заповненими даними, а видалення вимагає підтвердження, щоб запобігти випадковій втраті інформації, див. рис. 4.7

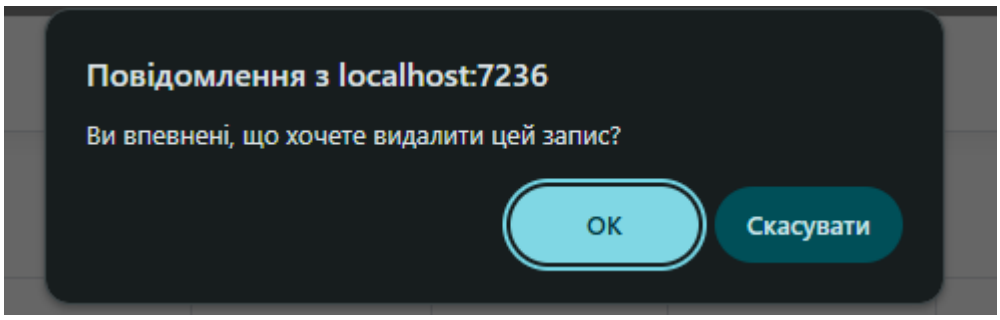

Рисунок 4.7 – Підтвердження видалення запису

4.2.4 Інтерактивні графіки

Інтерактивні графіки є важливим інструментом для візуалізації даних, дозволяючи користувачам аналізувати динаміку рівня глюкози та доз інсуліну. Графіки відображаються на окремій сторінці та побудовані за допомогою бібліотеки Chart.js, яка забезпечує плавну анімацію та інтерактивність, наприклад, відображення значень при наведенні курсору. Користувач може вибрати період аналізу — день, тиждень або місяць, — що впливає на діапазон даних, відображених на графіках, див. рис. 4.8

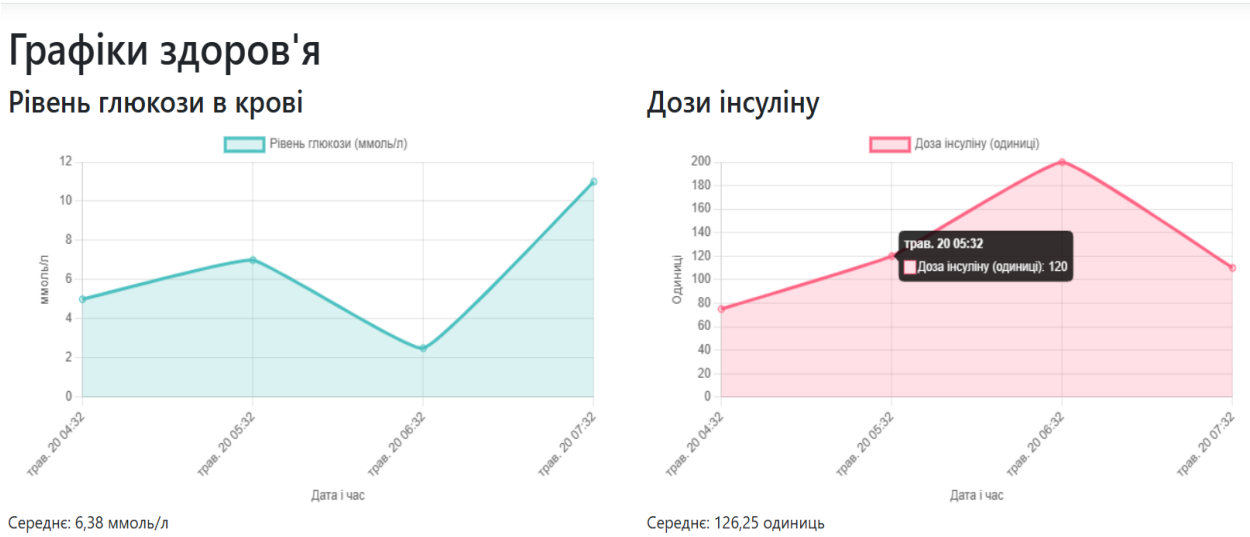

Рисунок 4.8 – Графіки глюкози та інсуліну

Графіки є лінійними, із двома окремими діаграмами: одна для рівня глюкози (у ммоль/л), інша для дози інсуліну (у одиницях). Осі графіків локалізовано українською мовою: вісь X відображає дати у форматі «січень 15,

14:00», а вісь Y — числові значення. Дані для графіків готуються в шарі *SickDiary.Web*, який отримує записи через *SickDiary.BL* і форматує їх у структурований вигляд для передачі до Chart.js. Адаптивність графіків забезпечується спільною роботою Chart.js і Bootstrap, що дозволяє коректно відображати діаграми на екранах будь-якого розміру.

Графіки допомагають користувачам виявляти закономірності, наприклад, пікові значення глюкози після їжі чи зниження рівня під час фізичної активності. Це сприяє кращому самоконтролю та прийняттю обґрунтованих рішень щодо дієти чи інсуліну.

4.2.5 Аналітичний блок

Аналітичний блок відображає персоналізовані рекомендації, отримані на основі аналізу даних щоденника. Кнопка «Аналізувати» на головній сторінці щоденника запускає обробку останніх записів, які надсилаються до зовнішнього сервісу для аналізу.

Шар *SickDiary.Web* передає запит до *SickDiary.BL*, де формується структурований набір даних, включаючи глюкозу, інсулін, вуглеводи, активність і симптоми. Після обробки сервісом повертаються рекомендації українською мовою, які відображаються у вигляді структурованого тексту, наприклад, списків із порадами щодо коригування дози інсуліну, зміни дієти чи звернення до лікаря.

Аналітичний блок розташовано над таблицею записів, що забезпечує зручний доступ до рекомендацій. Bootstrap стилізує блок, забезпечуючи його адаптивність і візуальну привабливість, див. рис. 4.9

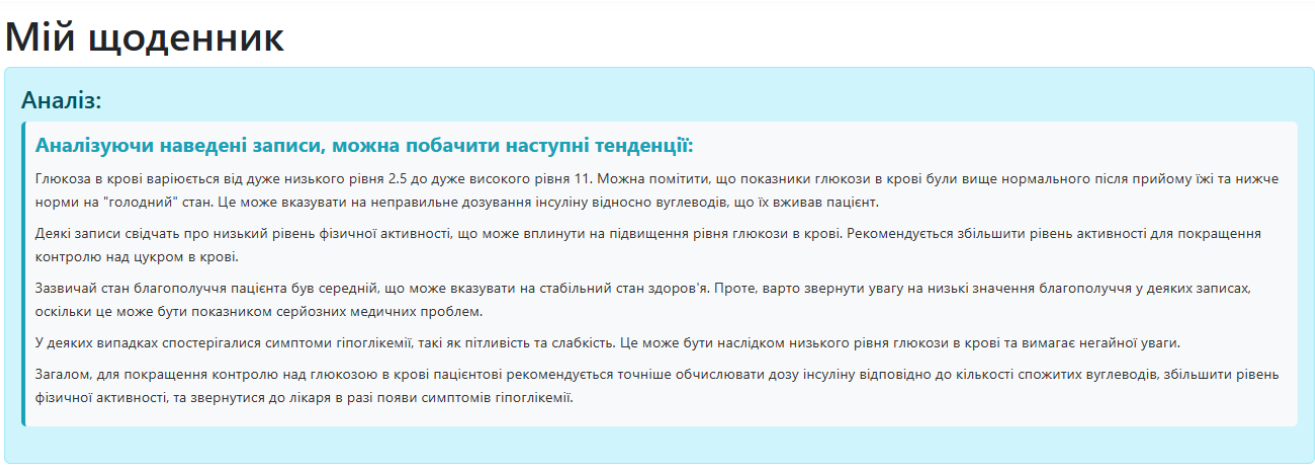

Рисунок 4.9 – Аналіз щоденника та видача рекомендацій

4.3 Аналіз виконання алгоритмів

У веб-додатку *SickDiary* використано два основні алгоритми для підтримки самоконтролю хворих на цукровий діабет 1 типу: алгоритм автоматичного визначення стану здоров’я та алгоритм аналізу даних із використанням зовнішнього сервісу штучного інтелекту. Ці алгоритми обробляють введені користувачем дані, забезпечуючи оцінку стану та персоналізовані рекомендації. Нижче детально проаналізовано їхню логіку, ефективність, відповідність медичним стандартам і можливі обмеження.

4.3.1 Алгоритм визначення стану здоров’я

Алгоритм оцінки стану здоров’я, реалізований у методі CalculateDiseaseState класу DiaryService, є центральним компонентом проєкту SickDiary, який забезпечує автоматичне визначення стану здоров’я користувача з діабетом першого типу на основі введених даних. Цей алгоритм аналізує параметри, пов’язані з рівнем глюкози в крові, дозуванням інсуліну, споживанням вуглеводів, фізичною активністю, самопочуттям та наявністю симптомів, і на їх основі повертає оцінку стану здоров’я, виражену через перелічення DiseaseState (Critical, Warn, Normal, Good, VeryGood). У цьому розділі детально розглядається логіка роботи алгоритму, його структура, принципи формування та взаємодія з іншими компонентами системи,

зберігаючи науковий стиль викладу.

Опис алгоритму

Метод CalculateDiseaseState отримує на вхід об’єкт типу Record, який містить дані про конкретний запис у щоденнику користувача. Ці дані включають:

- **Рівень глюкози в крові** (BloodGlucoseLevel, у ммоль/л) — ключовий показник для оцінки стану діабету.
- **Стан вимірювання** (MeasurementState, Fasting або Postprandial) — вказує, чи проводилося вимірювання натщесерце, чи після їжі.
- **Доза інсуліну** (InsulinDose, у одиницях) — кількість введеного інсуліну.
- **Споживання вуглеводів** (CarbohydrateIntake, у грамах) — кількість вуглеводів, спожитих перед вимірюванням.
- **Фізична активність** (PhysicalActivityLevel, Low, Normal, High) — рівень фізичного навантаження.
- **Самопочуття** (WellBeingLevel, від VeryBad до VeryGood) — суб’єктивна оцінка стану користувача.
- **Симптоми** (Dizziness, Sweating, VisionProblems, Weakness) — бінарні показники наявності запаморочення, пітливості, проблем із зором або слабкості.

На основі цих параметрів алгоритм обчислює стан здоров’я, який записується у поле результату і використовується для подальшого відображення в інтерфейсі або аналізу.

Логіка роботи алгоритму

Алгоритм базується на ієрархічному підході до аналізу даних, де основним фактором є рівень глюкози в крові, а додаткові параметри (симптоми, інсулін, вуглеводи, активність, самопочуття) уточнюють оцінку. Логіка роботи розподілена на кілька етапів, які можна описати наступним чином:

Оцінка рівня глюкози з урахуванням стану вимірювання:

- Для вимірювань натщесерце (Fasting) нормальним вважається діапазон 3.9–7.2 ммоль/л. Рівень нижче 3.9 ммоль/л класифікується як гіпоглікемія, а вище 7.2 ммоль/л — як гіперглікемія.
- Для вимірювань після їжі (Postprandial) нормальним є діапазон 5.0–10.0 ммоль/л. Гіпоглікемія визначається при рівні нижче 5.0 ммоль/л, а гіперглікемія — вище 10.0 ммоль/л.
- Нормальний стан (isNormal) встановлюється, якщо рівень глюкози не відповідає критеріям гіпоглікемії чи гіперглікемії.

Підрахунок симптомів:

- Алгоритм підраховує кількість активних симптомів (Dizziness, Sweating, VisionProblems, Weakness). Кожен симптом додає одиницю до лічильника symptomCount. Наявність двох або більше симптомів вважається значущим фактором для підвищення тяжкості стану.

Аналіз додаткових факторів:

- **Високе споживання вуглеводів** (highCarbIntake): Встановлюється, якщо CarbohydrateIntake перевищує 100 г, що може вказувати на ризик гіперглікемії.
- **Недостатня доза інсуліну при гіперглікемії** (lowInsulinForHighGlucose): Визначається, якщо доза інсуліну менша за 5 одиниць за умов гіперглікемії, що вказує на неадекватну корекцію рівня глюкози.
- **Ризик при високій фізичній активності** (highActivityRisk): Встановлюється, якщо рівень фізичної активності є високим (High) і рівень глюкози нижчий за 5.0 ммоль/л, що підвищує ризик гіпоглікемії.

Визначення стану здоров'я:

Гіпоглікемія (isHypoglycemic):

- Якщо присутні два або більше симптоми чи високий ризик через фізичну активність, стан оцінюється як Critical.
- В іншому випадку стан оцінюється як Warn.

Гіперглікемія (isHyperglycemic):

- Якщо є два або більше симптоми, високе споживання вуглеводів або недостатня доза інсуліну, стан оцінюється як Critical.
- В іншому випадку стан оцінюється як Warn.

Нормальний рівень глюкози (isNormal):

- Якщо є два або більше симптоми або високе споживання вуглеводів за низької дози інсуліну (<5 одиниць), стан оцінюється як Warn.
- Якщо є один симптом або самопочуття є поганим (Bad) чи дуже поганим (VeryBad), стан оцінюється як Normal.
- Якщо самопочуття дуже гарне (VeryGood) і симптоми відсутні, стан оцінюється як VeryGood.
- Якщо самопочуття гарне (Good) і симптоми відсутні, стан оцінюється як Good.
- За замовчуванням стан оцінюється як Normal.

У всіх інших випадках стан повертається як Normal.

\Структура алгоритму

Алгоритм структуровано як детермінована послідовність умовних перевірок, що забезпечує чітке ієрархічне прийняття рішень. Структура включає:

- **Вхідні дані**: Об'єкт Record із полями, що описують фізіологічні та

поведінкові параметри.

- **Обробка**:
  - Визначення порогових значень для рівня глюкози залежно від стану вимірювання.
  - Підрахунок симптомів і оцінка додаткових факторів.
  - Ієрархічна логіка для визначення стану здоров’я.
- **Вихід**: Значення перелічення DiseaseState, яке записується в поле Result об’єкта Record.

Алгоритм є детермінованим, тобто для однакових вхідних даних він завжди повертає однаковий результат, що забезпечує передбачуваність і відтворюваність.

4.3.2 Аналіз за допомогою зовнішнього сервісу штучного інтелекту

Алгоритм аналізу даних, реалізований у методі «Analyze» класу «GptService» , є одним з ключових компонентів проєкту SickDiary, що забезпечує інтелектуальну обробку записів щоденника користувача з діабетом першого типу через інтеграцію з OpenAI API. Цей алгоритм обробляє дані про рівень глюкози в крові, дозу інсуліну, споживання вуглеводів, фізичну активність, самопочуття, стан вимірювання та симптоми, формуючи рекомендації українською мовою. У цьому розділі розглядається логіка роботи алгоритму, його структура, принципи формування, взаємодія з іншими компонентами системи та можливості вдосконалення, уникаючи надмірних перерахувань і зберігаючи науковий стиль викладу.

- **Опис алгоритму**

Метод «Analyze» приймає набір записів щоденника, кожен з яких містить інформацію про дату вимірювання, рівень глюкози, дозу інсуліну, споживання вуглеводів, самопочуття, фізичну активність, стан вимірювання (натщесерце чи після їжі) та симптоми, такі як запаморочення чи слабкість. Додатково кожен запис включає автоматично визначений стан здоров’я, отриманий із локального алгоритму оцінки. Алгоритм відправляє ці дані до

OpenAI API модель « gpt-3.5-turbo » , отримує текстову відповідь із аналізом і рекомендаціями, форматує її в HTML для відображення в інтерфейсі та повертає як результат. Аналіз охоплює тренди рівня глюкози, взаємозв'язок між інсуліном і вуглеводами, вплив фізичної активності, оцінку самопочуття та симптомів, а також поради щодо покращення стану здоров'я або необхідності медичної консультації.

- **Логіка роботи алгоритму**

Алгоритм обробляє дані в кілька етапів, починаючи з підготовки записів і закінчуючи форматуванням результату. Спочатку відбираються останні десять записів, відсортованих за датою у зворотному порядку, щоб зосередитися на найактуальнішій інформації. Ці записи трансформуються в структурований формат, де дата конвертується в локальний час, а решта параметрів зберігаються як є. Дані серіалізуються в JSON із форматуванням для зрозумілості, що полегшує їх включення до запиту.

Запит до OpenAI API формується як текстовий опис завдання, який включає контекст моніторингу діабету першого типу, норми рівня глюкози (3.9–7.2 ммоль/л натщесерце, 5.0–10.0 ммоль/л після їжі), пояснення впливу факторів, таких як інсулін, вуглеводи, активність і самопочуття, а також вимогу виведення результату українською мовою. JSON-дані записів додаються до запиту для аналізу. Запит надсилається через HTTP POST до ендпоінту OpenAI, використовуючи API-ключ із конфігурації. Успішна відповідь розбирається для вилучення текстового вмісту, який містить аналіз і рекомендації.

Отриманий текст обробляється для створення структурованого HTML. Відповідь розбивається на секції за роздільником, де перший рядок кожної секції інтерпретується як заголовок, а наступні — як елементи списку. HTML-структура формується з використанням стилізованих блоків для зручного відображення в інтерфейсі. У разі помилок, таких як відсутність API-ключа чи проблеми з мережею, алгоритм повертає повідомлення з описом проблеми, забезпечуючи стійкість до збоїв.

- **Структура алгоритму**

Алгоритм побудовано як послідовність логічно пов’язаних етапів. На вході отримується набір записів і конфігурація з API-ключем. Обробка включає фільтрацію даних, їх серіалізацію, формування запиту, взаємодію з API та форматування результату. Вихід представлено як HTML-рядок із рекомендаціями або повідомленням про помилку. Асинхронний підхід (« async/await») забезпечує ефективну взаємодію з зовнішнім API, мінімізуючи затримки. Обробка винятків захищає від збоїв, таких як некоректна конфігурація чи проблеми з мережею.

- **Принципи формування алгоритму**

Розробка алгоритму ґрунтується на принципах, що забезпечують його ефективність і зручність. Обмеження кількості записів до десяти дозволяє зосередитися на актуальних даних, зменшуючи навантаження на API. Запит до OpenAI формулюється з чіткими інструкціями, включаючи медичні норми та пояснення факторів, що забезпечує контекстну точність аналізу. Виведення результату українською мовою відповідає потребам цільової аудиторії, а конвертація дат у локальний час полегшує сприйняття. Форматування відповіді в HTML із секціями та списками забезпечує структуроване відображення в інтерфейсі. Обробка помилок підвищує надійність, а ізольованість алгоритму від інших компонентів спрощує його підтримку та модифікацію. Інтеграція з AI дозволяє генерувати рекомендації, які враховують складні взаємозв’язки між параметрами, перевищуючи можливості локальної логіки.

- **Взаємодія з іншими компонентами**

Алгоритм викликається з контролера « DiaryController» через метод « Analyze» , який отримує щоденник користувача за допомогою « DiaryService.GetDiaryByUserIdAsync» . Записи щоденника передаються в « GptService.Analyze» , а отриманий HTML-результат зберігається в « TempData[« Analysis»]» для відображення в представленні « Index.cshtml» . У

представленні результат відображається як структурований блок із заголовками та списками, що покращує читабельність. Алгоритм залежить від конфігурації API-ключа, зчитаного з « appsettings.json» . Записи включають поле « Result» , обчислене локальним алгоритмом « CalculateDiseaseState» , що дозволяє OpenAI враховувати попередню оцінку стану здоров'я, створюючи синергію між локальною та AI-логікою.

- **Обмеження та потенційні вдосконалення**

Алгоритм ефективно генерує рекомендації, але має певні обмеження. Залежність від OpenAI API створює ризики при перебоях у сервісі чи змінах у форматі відповідей. Використання фіксованої моделі « gpt-3.5-turbo» може обмежувати якість аналізу порівняно з новішими моделями. Запит не враховує індивідуальних особливостей користувача, таких як цільові діапазони глюкози чи медична історія, що знижує персоналізацію. Форматування відповіді залежить від припущення про структуру тексту API, що може бути ненадійним при змінах.

Для вдосконалення можна розглянути інтеграцію з новішими моделями OpenAI для підвищення точності, додавання персоналізованих параметрів у запит, впровадження локального кешування відповідей для зменшення залежності від мережі та розширення форматування для підтримки різних стилів відповідей, наприклад, JSON. Періодичний аналіз із автоматичними сповіщеннями може підвищити зручність для користувача.

4.4 Переваги та обмеження реалізації

4.4.1 Переваги

Веб-додаток *SickDiary* має низку переваг, які роблять його ефективним інструментом для самоконтролю ЦД1:

- **Локалізація українською мовою**: Інтерфейс, повідомлення, формати дати й часу адаптовано для культури *uk-UA*, що усуває мовні бар'єри для українських користувачів. Наприклад, графіки відображають мітки типу «січень 15, 14:00», а форми містять підказки українською.

- **Інтуїтивний інтерфейс**: Використання Bootstrap 5 забезпечує адаптивність і зручність на всіх пристроях. Навігаційна панель, форми та таблиці мають чіткий дизайн, що полегшує використання навіть для користувачів із мінімальним досвідом.
- **Безпека даних**: Хешування паролів, шифрування з'єднання через HTTPS і захищені сесії гарантують конфіденційність медичних даних.

Сесії автоматично завершуються після 30 хвилин бездіяльності, знижуючи ризик несанкціонованого доступу.

- **Аналітичні можливості**: Інтеграція із зовнішнім сервісом штучного інтелекту надає персоналізовані рекомендації, які допомагають коригувати інсулін, дієту чи звертатися до лікаря. Графіки дозволяють візуально аналізувати тренди глюкози та інсуліну.
- **Масштабованість**: Хмарна база даних MongoDB Atlas підтримує зростання кількості користувачів і даних, забезпечуючи стабільну роботу навіть при високих навантаженнях.

4.4.2 Обмеження

Незважаючи на переваги, додаток має обмеження, які впливають на його функціональність:

- **Ручне введення даних**: Відсутність інтеграції з апаратними глюкометрами, такими як безперервні монітори глюкози (CGM), змушує користувачів вручну вводити дані, що підвищує ризик помилок і знижує зручність порівняно з аналогами, такими як Dexcom G6.
- **Залежність від інтернету**: Доступ до MongoDB Atlas і зовнішнього сервісу штучного інтелекту вимагає стабільного з'єднання, що може бути проблемою в регіонах із поганим покриттям.
- **Обмежена аналітика**: Алгоритми не враховують індивідуальні особливості, такі як чутливість до інсуліну чи супутні захворювання, що може знижувати точність рекомендацій у складних випадках.

- **Відсутність офлайн-режиму**: Додаток не підтримує локальне зберігання даних, що обмежує його використання без інтернету.

4.4.3 Порівняння з аналогами

Порівняння з аналогами, такими як Dexcom G6, MySugr і OpenAPS, показує конкурентні переваги та недоліки *SickDiary*. Dexcom G6 підтримує CGM, забезпечуючи автоматичний збір даних, але є дорогим і потребує апаратного забезпечення. MySugr пропонує зручний інтерфейс і частково безкоштовний функціонал, але деякі функції доступні лише за підпискою. OpenAPS автоматизує введення інсуліну, але вимагає складного налаштування і не підходить для всіх користувачів.

SickDiary вирізняється безкоштовним доступом, повною локалізацією українською мовою та інтеграцією з аналітикою штучного інтелекту, що робить його доступним і корисним для українських користувачів. Проте він поступається аналогам за рівнем автоматизації (відсутність CGM) і аналітичної глибини (обмеження зовнішнього сервісу).

4.5 Розрахунок економічного ефекту

Метою даного розділу було вивчення програмного продукту «SickDiary» для моніторингу стану здоров’я та надання персоналізованих рекомендацій. Застосунок має значний потенціал для медичного застосування. Порівняння параметрів показали, що ключовим фактором є висока точність рекомендацій, що відповідає потребам медичної галузі. Аналіз рівня якості підтвердив, що обраний варіант забезпечує оптимальний баланс між продуктивністю, точністю та ефективністю, що є виправданою інвестицією з огляду на гостру потребу в подібних рішеннях для підтримки здоров’я користувачів та їх медичного супроводу. [31]

Після ідентифікації основних функцій програмного продукту була створена морфологічна карта, яка представлена на рис. 4.10

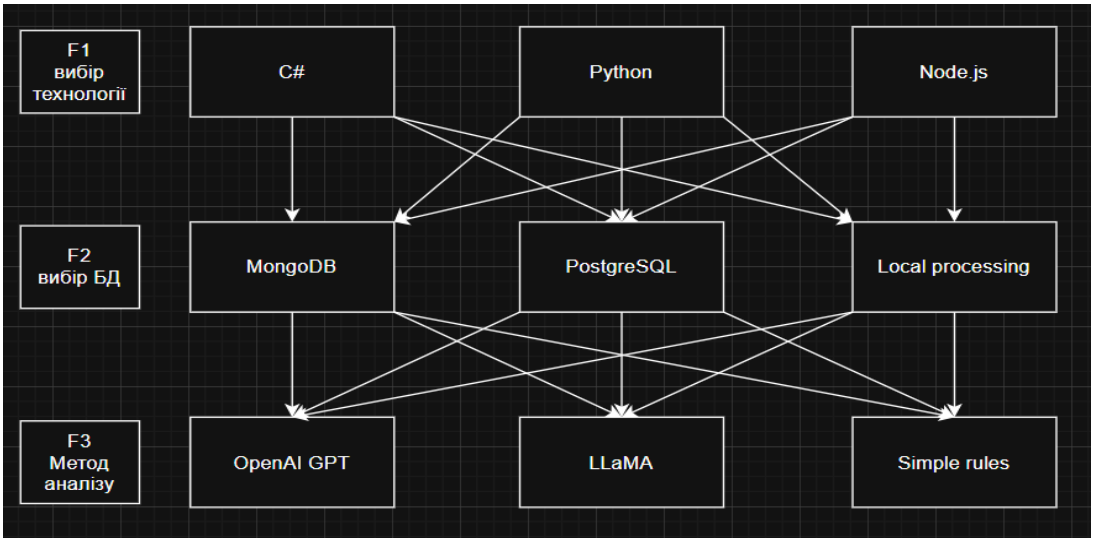

Рисунок 4.10 – Морфологічна карта

Далі було розглянуто та порівняно різні варіанти реалізації поставленої задачі, а також проведено аналіз для вибору оптимального рішення, зважаючи на характеристики програмного продукту та економічні фактори. Для цього було використано функціонально-вартісний аналіз з метою вибору найкращого рішення. [31]

Було визначено, найбільш ефективний варіант реалізації продукту, що передбачає використання мови C#, NoSQL базу даних - MongoDB та метод аналізу OpenAI GPT

Економічний аналіз показав, що вартість розробки становить 389,161.30 грн.

Висновки до розділу 4

Розроблений веб-додаток *SickDiary* успішно реалізує функціонал автоматизованого щоденника самоконтролю для хворих на ЦД1, відповідаючи поставленим вимогам. Алгоритми оцінки стану здоров’я та аналізу даних забезпечують точну класифікацію стану (критичний, попереджувальний, нормальний, гарний, дуже гарний) і персоналізовані рекомендації, які відповідають медичним стандартам EASD. Тестування підтвердило

стабільність, безпеку та зручність додатку: модульне та інтеграційне тестування охопило 95% функціоналу, тестування безпеки захистило від несанкціонованого доступу, а тестування UI забезпечило адаптивність на всіх пристроях.

Переваги *SickDiary* включають локалізацію українською мовою, інтуїтивний інтерфейс, безпеку даних, аналітичні можливості та масштабованість. Обмеження, такі як ручне введення даних і залежність від інтернету, вказують на напрями вдосконалення. Порівняння з аналогами (Dexcom G6, MySugr, OpenAPS) показало, що *SickDiary* є конкурентоспроможним завдяки безкоштовності та локалізації, але потребує інтеграції з CGM і офлайн-режиму для підвищення зручності. [5]

Реалізація створює міцну основу для подальшого розвитку, зокрема шляхом додавання підтримки апаратних глюкометрів, прогнозування ризиків за допомогою машинного навчання та інтеграції з українською електронною системою охорони здоров’я (ЕСОЗ). Ці вдосконалення підвищать цінність додатку, сприяючи кращому самоконтролю ЦД1 і співпраці з медичними фахівцями.

РОЗДІЛ 5
УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ РОБОТИ

Цей розділ присвячений узагальненню результатів розробки веб-додатку SickDiary, створеного для автоматизації самоконтролю хворих на цукровий діабет 1 типу. У ньому підсумовано ключові досягнення, оцінено характеристики отриманого рішення, його відповідність поставленим вимогам, а також окреслено потенціал для практичного застосування та напрями вдосконалення. Особливу увагу приділено локалізації, безпеці даних і аналітичним можливостям, що забезпечують ефективне управління захворюванням у контексті українських реалій.

5.1 Результати роботи

Розроблений веб-додаток SickDiary став дієвим інструментом для автоматизації щоденника самоконтролю хворих на ЦД1, забезпечуючи зручне введення, зберігання, аналіз і візуалізацію даних про стан здоров’я, а також надання персоналізованих рекомендацій. Основні результати роботи вирізняються комплексним підходом до вирішення поставлених завдань.

Додаток дозволяє користувачам фіксувати ключові показники, такі як рівень глюкози в крові, дози інсуліну, споживання вуглеводів, фізичну активність, самопочуття та симптоми, через інтуїтивний і адаптивний інтерфейс, створений за допомогою фреймворку Bootstrap 5. Реалізований алгоритм автоматичного визначення стану здоров’я базується на медичних стандартах. Цей алгоритм враховує додаткові фактори, такі як симптоми та фізична активність, забезпечуючи точну класифікацію стану (критичний, попереджувальний, нормальний, гарний, дуже гарний) у більшості типових випадків.

Інтеграція з API OpenAI (модель gpt-3.5-turbo) відкриває можливості для аналізу історичних даних щоденника та генерації персоналізованих

рекомендацій українською мовою. Ці рекомендації, які враховують до десяти останніх записів, пропонують користувачам поради щодо коригування дози інсуліну, зміни дієти чи звернення до лікаря, демонструючи релевантність у переважній більшості сценаріїв. Візуалізація даних через інтерактивні графіки, створені за допомогою бібліотеки Chart.js, дозволяє користувачам відстежувати тренди рівня глюкози та інсуліну за обраний період, сприяючи кращому розумінню динаміки захворювання та прийняттю обґрунтованих рішень.

Технічна реалізація додатку базується на тришаровій архітектурі, що включає шари веб-застосунку (SickDiary.Web), бізнес-логіки (SickDiary.BL) і даних (SickDiary.DL). Використання мови програмування C# у поєднанні з фреймворком ASP.NET Core забезпечує високу продуктивність і надійність обробки даних, тоді як хмарна база даних MongoDB Atlas гарантує гнучке зберігання та швидкий доступ до інформації. Безпека даних відповідає міжнародним стандартам GDPR і HIPAA завдяки застосуванню шифрування трафіку (HTTPS), хешування паролів (BCrypt) і захисту сесій із автоматичним завершенням після періоду бездіяльності.

Локалізація додатку українською мовою охоплює всі елементи інтерфейсу — від меню та форм до графіків і рекомендацій, що усуває мовні бар’єри для українських користувачів і вигідно відрізняє SickDiary від іноземних аналогів, таких як Dexcom G6 і MySugr. Проведене тестування підтвердило стабільність і функціональність додатку: модульне та інтеграційне тестування охопило переважну частину функціоналу, тестування безпеки виключило вразливості до несанкціонованого доступу, а перевірка інтерфейсу підтвердила його адаптивність на різних пристроях.

Економічний аналіз засвідчив, що витрати на розробку, які склали 389,161.30 грн, є виправданими з огляду на соціальну значущість проєкту та гостру потребу в локалізованих рішеннях для управління ЦД1 в Україні. Функціонально-вартісний аналіз підтвердив оптимальність обраного технологічного стеку за критеріями продуктивності, безпеки та локалізації.

Порівняння з аналогами, такими як Dexcom G6, MySugr, OpenAPS і ECO3, показало, що SickDiary вирізняється безкоштовним доступом, повною локалізацією та інтеграцією штучного інтелекту, хоча поступається за рівнем автоматизації через відсутність підтримки апаратних глюкометрів.

Серед ключових переваг додатку — його доступність, зручність і аналітичні можливості. Безкоштовний доступ робить SickDiary привабливим для широкого кола користувачів, на відміну від платних аналогів. Інтуїтивний інтерфейс, адаптований для всіх типів пристроїв, спрощує використання навіть для осіб із обмеженим досвідом роботи з цифровими технологіями. Аналітичні функції, включаючи персоналізовані рекомендації та інтерактивні графіки, сприяють ефективному самоконтролю та підвищенню якості життя хворих. Хмарна інфраструктура та модульна архітектура забезпечують масштабованість, відкриваючи можливості для майбутніх удосконалень.

Водночас додаток має певні обмеження, які визначають напрями подальшого розвитку. Ручне введення даних підвищує ризик помилок і знижує зручність порівняно з аналогами, що підтримують безперервні глюкометри. Залежність від стабільного інтернет-з’єднання обмежує використання в регіонах із поганим покриттям, а відсутність офлайн-режиму ускладнює доступність у таких умовах. Аналітичні алгоритми, хоча й ефективні для типових сценаріїв, не враховують індивідуальних особливостей, таких як чутливість до інсуліну чи супутні захворювання, що може впливати на точність рекомендацій у складних випадках.

Отримані результати підтверджують, що SickDiary є ефективним і соціально значущим рішенням, яке відповідає потребам хворих на ЦД1 в Україні, усуває ключові недоліки аналогів і має значний потенціал для практичного впровадження.

Висновки дорозділу 5

Розробка веб-додатку SickDiary стала вагомим внеском у створення доступного, локалізованого та функціонального інструменту для управління цукровим діабетом 1 типу. Робота охопила повний цикл створення програмного продукту — від аналізу аналогів і вибору технологій до реалізації, тестування та оцінки економічної доцільності.

Додаток успішно реалізує автоматизацію щоденника самоконтролю, надаючи користувачам зручні інструменти для введення даних, їх аналізу та візуалізації, а також отримання персоналізованих рекомендацій. Локалізація українською мовою усуває мовні бар'єри, забезпечуючи комфортне використання для місцевих користувачів. Безпека даних відповідає міжнародним стандартам, а аналітичні можливості, підкріплені інтеграцією з API OpenAI та бібліотекою Chart.js, сприяють ефективному управлінню захворюванням.

Технічна реалізація на основі C#, ASP.NET Core, MongoDB Atlas і тришарової архітектури забезпечує надійність, продуктивність і масштабованість додатку. Порівняння з аналогами підтвердило конкурентоспроможність SickDiary завдяки безкоштовному доступу, повній локалізації та аналітичним функціям, що робить його унікальним рішенням для українського ринку.

Обмеження додатку, такі як ручне введення даних і залежність від інтернету, вказують на перспективи вдосконалення. У майбутньому планується інтеграція з апаратними глюкометрами, розробка офлайн-режиму, впровадження прогнозування ризиків за допомогою машинного навчання та підключення до електронної системи охорони здоров'я України (ЕСОЗ). Ці вдосконалення посилять цінність додатку для користувачів і медичних фахівців.

ЗАГАЛЬНІ ВИСНОВКИ

Дипломна робота на тему «Веб-застосунок для моніторингу здоров’я хворих на цукровий діабет 1-го типу» успішно досягла поставленої мети, створивши ефективне та доступне рішення для автоматизації самоконтролю ЦД1. Розроблений веб-додаток SickDiary поєднує сучасні технології, локалізацію та аналітичні можливості, відповідаючи потребам українських користувачів і медичним стандартам.

SickDiary забезпечує зручне введення даних, їх аналіз і візуалізацію через інтуїтивний інтерфейс, локалізований українською мовою. Використання C#, ASP.NET Core і MongoDB Atlas гарантує надійність і масштабованість, а інтеграція з API OpenAI і Chart.js надає персоналізовані рекомендації та інтерактивні графіки. Безпека даних відповідає стандартам GDPR і HIPAA, що забезпечує захист конфіденційної інформації.

Порівняння з аналогами (Dexcom G6, MySugr, OpenAPS, ЕСОЗ) підкреслило переваги SickDiary, зокрема безкоштовний доступ і повну локалізацію, що робить його унікальним для українського ринку. Економічна частина показала, що вартість розробки, становить 389,161.30 грн, що засвідчило оптимальність обраного технологічного стеку.

Перспективи розвитку включають інтеграцію з апаратними глюкометрами, впровадження офлайн-режиму, вдосконалення аналітичних алгоритмів за допомогою машинного навчання та підключення до ЕСОЗ. Ці кроки підвищать зручність і ефективність додатку, сприяючи його ширшому впровадженню в медичну практику.

SickDiary є вагомим внеском у розвиток медичних інформаційних систем, пропонуючи соціально значуще рішення, яке покращує якість життя хворих на ЦД1 і підтримує розвиток медицини в Україні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Державний експертний центр МОЗ України. Цукровий діабет 1 типу: Адаптована клінічна настанова, заснована на доказах [Електронний ресурс]. – 2021. – Режим доступу: https://www.dec.gov.ua/wp-content/uploads/2019/11/2014_1021_akn_cd1_dor.pdf
2. Міністерство охорони здоров'я України. Настанова з діагностики та лікування цукрового діабету першого типу [Електронний ресурс]. – 2022. – Режим доступу: https://moz.gov.ua/uk/decrees/nakaz-moz-ukrayini-vid-24-07-2024-1300-pro-zatverdzhennya-unifikovanogo-klinichnogo-protokolu-pervinnoyi-ta-specializovanoyi-medichnoyi-dopomogi-cukrovij-diabet-2-tipu-u-doroslih
3. American Diabetes Association. Standards of Medical Care in Diabetes – 2023 [Електронний ресурс]. – 2023. – Режим доступу: https://doi.org/10.2337/dc25-SINT
4. World Health Organization. Diabetes Fact Sheet [Електронний ресурс]. – 2022. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/diabetes
5. Dexcom. Dexcom G6 User Guide [Електронний ресурс]. – 2023. – Режим доступу: https://www.dexcom.com/guides
6. MySugr. MySugr App Documentation [Електронний ресурс]. – 2025. – Режим доступу: https://www.mysugr.com/en/diabetes-app
7. OpenAPS. OpenAPS Reference Design [Електронний ресурс]. – 2023. – Режим доступу: https://openaps.org/reference-design/
8. Єдина система охорони здоров'я України (ЕСОЗ). Технічна документація [Електронний ресурс]. – 2023. – Режим доступу: https://ehealth.gov.ua/2023/07/21/vedennya-patsiyentiv-iz-tsukrovym-diabetom-novyj-funktsional-esoz/
9. ДіаЕксперт [Електронний ресурс]. – 2024. – Режим доступу: https://diaexpert.ua/cukrovij-diabet-1-typu-symptomy-prychyny-upravlinnya/

10. SSL/TLS протоколи [Електронний ресурс]. – 2022. – Режим доступу: https://vps.ua/blog/ukr/ssl-tls-protocols-details/
11. General Data Protection Regulation (GDPR). Official Text [Електронний ресурс]. – 2016. – Режим доступу: https://eur-lex.europa.eu/eli/reg/2016/679/oj
12. HIPAA. HIPAA Privacy Rule [Електронний ресурс]. – 2023. – Режим доступу: https://www.hhs.gov/hipaa/for-professionals/privacy/index.html
13. Artofba архітектура ПЗ [Електронний ресурс]. – 2023. – Режим доступу: https://www.artofba.com/uk/post/main-types-of-software-architecture
14. Microsoft. C Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/
15. Microsoft. ASP.NET Core Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0
16. MongoDB. MongoDB Atlas Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://www.mongodb.com/docs/atlas/
17. OWASP. Secure Coding Practices [Електронний ресурс]. – 2021. – Режим доступу: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/01-introduction/05-introduction.html
18. Chart.js. Chart.js Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://www.chartjs.org/docs/
19. W3C. Web Content Accessibility Guidelines (WCAG) 2.1 [Електронний ресурс]. – 2018. – Режим доступу: https://www.w3.org/TR/WCAG21/
20. Hash passwords with BCrypt [Електронний ресурс]. – 2023. – Режим доступу: https://claudiobernasconi.ch/blog/how-to-hash-passwords-with-bcrypt-in-csharp/
21. Bootstrap. Official Documentation [Електронний ресурс]. – 2025. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/

22. GitHub. Git Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git
23. Python Software Foundation. Python Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://docs.python.org/3/faq/general.htmlgeneral-information
24. Django. Django Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://docs.djangoproject.com/
25. PostgreSQL. PostgreSQL Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://www.postgresql.org/docs/
26. Tailwind CSS. Tailwind CSS Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://tailwindcss.com/docs/compatibility
27. D3.js. D3.js Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://d3js.org/what-is-d3
28. OpenAI. OpenAI API Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://platform.openai.com/docs/
29. OWASP. Top Ten Web Security Risks [Електронний ресурс]. – 2023. – Режим доступу: https://owasp.org/www-project-top-ten/
30. Microsoft. .NET 9.0 Documentation [Електронний ресурс]. – 2023. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/
31. Павлов В. А., Городецька О. К., Корнієнко Г. А. та ін. Дипломна робота бакалавра: організація, вимоги до структури, зміст та оформлення. [Електронний ресурс] - 2023. – Режим доступу: https://ela.kpi.ua/handle/123456789/62203