МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

МАДАРАШ Наталя Анатоліївна

Веб-базована система управління особистими завданнями/ Web-based personal task management system

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконала студентка групи КН-42
Н. А. Мадараш
_____

Науковий керівник:
І. Р. Кіт
_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
МАДАРАШ Наталя Анатоліївна
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-базована система управління особистими завданнями / Web-Based Personal Task Management System
керівник роботи Кіт Іван Романович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- аналіз існуючих рішень та літератури у сфері управління завданнями для виявлення недоліків та переваг;
- аналіз вимог користувачів до системи управління завданнями;
- проектування архітектури системи та інтерфейсу користувача;
- реалізація функціоналу для створення, редагування та видалення завдань;
- встановлення зв'язків між завданнями та управління їх статусом і пріоритетністю;
- забезпечення можливості пошуку та фільтрування завдань;
- тестування та валідація системи для забезпечення її надійності та зручності використання.

5. Перелік графічного матеріалу в роботі:
- алгоритм роботи додатку;
- діаграми класів;
- знімки роботи системи.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ Н.А. Мадараш
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І.Р. Кіт
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-базована система управління особистими завданнями» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 57 сторінок і містить 24 ілюстрації, 2 додатки та 31 використане джерело.

Метою даної кваліфікаційної роботи є аналіз та розробка веб-базованої системи для управління особистими завданнями, яка надасть користувачам зручний і ефективний інструмент для організації їхнього робочого та особистого життя.

Методами розроблення обрано метод інкрементної моделі, яка дозволяє поступово вдосконалювати систему через циклічні ітерації, а також використовувались сучасні технології та інструменти, такі як .NET для створення надійного і масштабованого веб-додатка.

Результати дослідження: розроблено веб-базовану систему управління особистими завданнями, яка забезпечує користувачам зручний інструмент для підвищення продуктивності та організації робочих і особистих завдань.

Результати роботи можуть бути використані для організації особистих та робочих завдань, а також у будь-якій сфері, де необхідне ефективне планування та виконання завдань.

Ключові слова: УПРАВЛІННЯ ЗАВДАННЯМИ, МЕНЕДЖМЕНТ ЧАСУ, ВЕБ-БАЗОВАНА СИСТЕМА, .NET.

ANNOTATION

Qualification work on the topic "Web-based Personal Task Management System" for obtaining the Bachelor's degree in the specialty 122 "Computer Science" of the educational program "Computer Science" is written in a volume of 57 pages and contains 24 illustrations, 2 appendices, and 31 used sources.

The aim of this qualification work is to analyze and develop a web-based system for personal task management, which will provide users with a convenient and efficient tool for organizing their work and personal life.

The research methods chosen include the incremental model, which allows for the gradual improvement of the system through cyclical iterations. Modern technologies and tools, such as .NET, were also used to create a reliable and scalable web application.

Research results: a web-based personal task management system has been developed, providing users with a convenient tool for increasing productivity and organizing work and personal tasks.

The results of the work can be used for organizing personal and work tasks, as well as in any area where effective planning and task execution are required.

Keywords: TASK MANAGEMENT, TIME MANAGEMENT, WEB-BASED SYSTEM, .NET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – програмний інтерфейс додатку (Application Programming Interface)

CSS – каскадні таблиці стилів (Cascading Style Sheets)

HTML – мова розмітки гіпертексту (HyperText Markup Language)

HTTP – протокол передачі гіпертекстових документів (HyperText Transfer Protocol)

JSON – текстовий формат обміну даними (JavaScript Object Notation)

JWT – веб-токен JSON (JSON Web Token)

LINQ – запити, інтегровані в мову (Language Integrated Query)

ORM – об’єктно-реляційний маппер (Object-Relational Mapping)

REST – підхід до архітектури представлення стану (Representational State Transfer)

SQL – мова структурованих запитів (Structured Query Language)

UML – уніфікована мова моделювання (Unified Modeling Language)

ВСТУП

Управління особистими завданнями відіграє важливу роль у сучасному житті, допомагаючи людям організовувати свій час та ресурси. Ефективне управління завданнями дозволяє підвищити продуктивність, забезпечити організованість та зменшити рівень стресу. З розвитком технологій з'являється все більше інструментів, які допомагають автоматизувати та спрощувати цей процес.

Проаналізувавши існуючі рішення у сфері менеджменту завдань, було виявлено необхідність створення власної системи, що відповідала би потребам користувачів. Основна мета полягає у розробці веб-базованої платформи, яка була би доступною та зручною для щоденного використання. Під час дослідження було виявлено, що більшість популярних рішень спрямовані переважно на корпоративний сектор, що часто ускладнює їх використання для простих користувачів. Тому було прийнято рішення розробити систему з урахуванням потреб таких осіб, забезпечивши її простим та інтуїтивно зрозумілим інтерфейсом.

Актуальність теми створення веб-базованої системи управління завданнями є надзвичайно важливим у сучасних умовах, коли все більше людей працюють віддалено і потребують доступу до своїх завдань з будь-якого місця. Аналіз існуючих рішень у сфері менеджменту завдань показав, що багато з них спрямовані переважно на корпоративний сектор, що ускладнює їх використання для простих користувачів. Це створює необхідність розробки власної системи, яка би відповідала потребам індивідуальних користувачів, забезпечуючи її простим та інтуїтивно зрозумілим інтерфейсом.

Мета роботи полягає в розробці веб-базованої системи для управління особистими завданнями, яка надасть користувачам зручний і ефективний інструмент для організації їхнього робочого та особистого життя. Основні завдання, які необхідно вирішити для досягнення цієї мети, включають:

- Аналіз вимог користувачів до системи управління завданнями.
- Проектування архітектури системи та інтерфейсу користувача.
- Реалізація функціоналу для створення, редагування та видалення завдань.

– встановлення зв'язків між завданнями та управління їх статусом і пріоритетністю;

– забезпечення можливості пошуку та фільтрування завдань;

– тестування та валідація системи для забезпечення її надійності та зручності використання;

– для досягнення поставленої мети в роботі буде використано наступні методи;

– аналіз існуючих рішень та літератури у сфері управління завданнями для виявлення недоліків та переваг;

– проектування архітектури системи та розробка er-діаграм для бази даних;

– використання сучасних технологій та інструментів для реалізації веб-базованої системи;

– проведення тестування функціоналу системи та її користувацького інтерфейсу;

– збір зворотного зв'язку від потенційних користувачів для вдосконалення системи.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися: «ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-БАЗОВАНОЇ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ЗАВДАННЯМИ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд предметної області дослідження

Тайм-менеджмент – це термін, який широко використовується в повсякденному житті, академічному і професійному світах, але йому важко дати чітке тлумачення. Згідно з Лейкіном [1], тайм-менеджмент включає в себе процес розуміння того, що вам потрібно, поглиблення цілей, яких вам потрібно досягти, а також визначення пріоритетів і планування завдань, необхідних для досягнення цих цілей. Роботи Кауфмана-Скарборо та Ліндквіста [2] пояснює тайм-менеджмент як спосіб оцінити відносну важливість діяльності через розробку плану пріоритетів. Ці два терміни підкреслюють, що визначення пріоритетів діяльності це спосіб ефективного управління часом.

На основі вивчення літератури Брігітта Класенса, Вен-Делієна ван Еерде та Крістеля Рутте [3] пропонується наступне визначення тайм-менеджменту “поведінка, спрямована на досягнення ефективного використання часу при виконанні певної діяльності”.

Вони визначають, що використання часу не може бути самостійною ціллю, досягнутою без урахування інших факторів. У центрі уваги тайм-менеджменту - цілеспрямована діяльність, до прикладу: виконання завдання, яке здійснюється у спосіб з ефективним використанням часу. Згідно огляду літератури про тайм-менеджмент [4], ця поведінка включає такі аспекти:

– поведінка оцінки часу спрямована на усвідомлення тут і зараз або минулого, теперішнього і майбутнього [2] та самоусвідомлення використання свого часу (ставлення, пізнання, наприклад, Wratcher and Jones [5]), що допомагає приймати завдання та обов'язки, які відповідають межам власних здібностей;

– поведінка планування, така як: постановка цілей, планування завдань, визначення пріоритетів, складання списків справ, групування завдань (наприклад, Britton and Tesser [16]; Macan [7, 8]), які спрямовані на ефективне використання часу;

– моніторинг поведінки, метою якого є спостереження за використанням

часу протягом виконання діяльності, що формує зворотний зв'язок, який дозволяє обмежити вплив переривань з боку іншого (наприклад, Fox and Dwyer [9]; Zijlstra та ін. [10]).

Під час вивчення впливу цих моделей поведінки, було виявлено, що планування показало найбільш значущі результати. Бонд і Фезер [11], наприклад, виявили, що фактор відчуття мети був найважливішим. Макан [7] виявив, що підшкала постановки цілей і визначення пріоритетів значною мірою пов'язана з такими результатами, як відчуття контролю над часом і задоволеність роботою. Вони стверджують, що короткострокове планування є більш ефективною технікою управління часом, ніж довгострокове, оскільки плани можуть бути скориговані відповідно до негайних змін або непередбачуваних ситуацій, що забезпечує гнучкість. Три дослідження (Hall and Hursch [12]; King та ін. [13]; Orpen [14]) показали позитивний зв'язок між навчанням технікам тайм-менеджменту та продуктивністю, що означає, що застосування ефективних стратегій управління часом може мати позитивний вплив на те, чого ми хочемо досягти.

Стратегії тайм-менеджменту часто асоціюються з рекомендацією ставити цілі або завдання. Але тоді залишається питання, які завдання потрібно виконати в першу чергу? Нам доведеться визначити пріоритетність цих завдань. Це можна зробити різними способами, ось кілька прикладів:

– ABC-аналіз [1]: це метод, який фокусується на класифікації великих даних за групами. Техніка використовує пріоритетні групи A, B і C. A, B і C використовуються для:

A: завдання, які є терміновими та важливими;

B: завдання, які важливі, але не термінові;

C: завдання, які не є важливими.

За допомогою цього методу ми можемо впорядкувати завдання за групами пріоритетності.

– Аналіз Парето: ідея полягає в тому, що 80 відсотків завдань можна виконати за 20 відсотків наявного часу. Решта 20 відсотків завдань займуть решту 80 відсотків наявного часу. Це можна використати, щоб розділити завдання на дві групи. Перша група повинна отримати найвищий пріоритет [15].

– Метод Ейзенхауера: цей метод походить від цитати, яку приписують Дуайту Д. Ейзенхауеру: "У мене є два типи проблем: термінові та важливі. Термінові не є важливими, а важливі ніколи не бувають терміновими. Це метод, який розміщує завдання в матриці рішень, як показано на рисунку 1.1 (також відома як "скринька Ейзенхауера" або "матриця рішень Ейзенхауера" [16]).

Матриця використовує наступні 4 квадранти (дивитися таблиця 1.1):

- квадрант "Важливо / Терміново";
- квадрант "Важливо / Не терміново";
- квадрант "Не важливо / Терміново";
- квадрант "Не важливо / Не терміново".

Таблиця 1.1 – Матриця Ейзенхауера

	Терміново	Не терміново
Важливо	Квадрант 1	Квадрант 2
	Важливо і терміново	Важливо, але не терміново
Не важливо	Квадрант 3	Квадрант 4
	Не важливо, але терміново	Не важливо і не терміново

Попри те, що існує багато інших методів визначення пріоритетності завдань, можна зробити висновок, що вибір і виконання завдань є важливим фактором, коли хтось хоче продуктивно управляти своїм часом. Оскільки короткострокове планування вважається більш ефективною технікою тайм-менеджменту, ніж довгострокове, буде більш доцільно зосередити дослідження на способах управління незавершеними справами та короткостроковими завданнями.

1.2. Аналіз наявних програмних рішень

У ході роботи було розглянуто та проаналізовано існуючі рішення реалізації застосунків із менеджменту завдань. Нижче наведено приклад декількох популярних із них.

Google Keep [17] – чудовий інструмент для ведення нотаток, відомий своєю універсальністю та зручними функціями (рисунок 1.1). Його широка популярність зумовлена кількома ключовими характеристиками, які задовольняють різноманітні потреби користувачів у різних сферах.

Однією з головних причин, чому користувачі полюбляють Google Keep, є його безперешкодна інтеграція в цифрову екосистему. Завдяки тісній інтеграції з іншими сервісами Google, такими як Gmail, Google Диск і Google Календар, він сприяє злагодженому робочому процесу, дозволяючи користувачам легко отримувати доступ до своїх нотаток і організовувати їх поряд з іншими важливими завданнями та документами. Крім того, його функції для спільної роботи дають можливість командам ефективно працювати разом, сприяючи ефективній комунікації та управлінню проектами.

Оскільки, Google Keep є безкоштовним і доступним сервісом це робить його кращим вибором як для приватних осіб, так і для організацій. Його доступність на різних платформах, включаючи Android, iOS і веб-браузери, забезпечує безперебійну синхронізацію і доступність з будь-якого пристрою, підвищуючи зручність і продуктивність роботи користувачів. Загалом, Google Keep продовжує підтримувати свою репутацію надійного і незамінного інструменту для ведення нотаток і управління завданнями завдяки своїм надійним функціям і дизайну, орієнтованому на користувача.

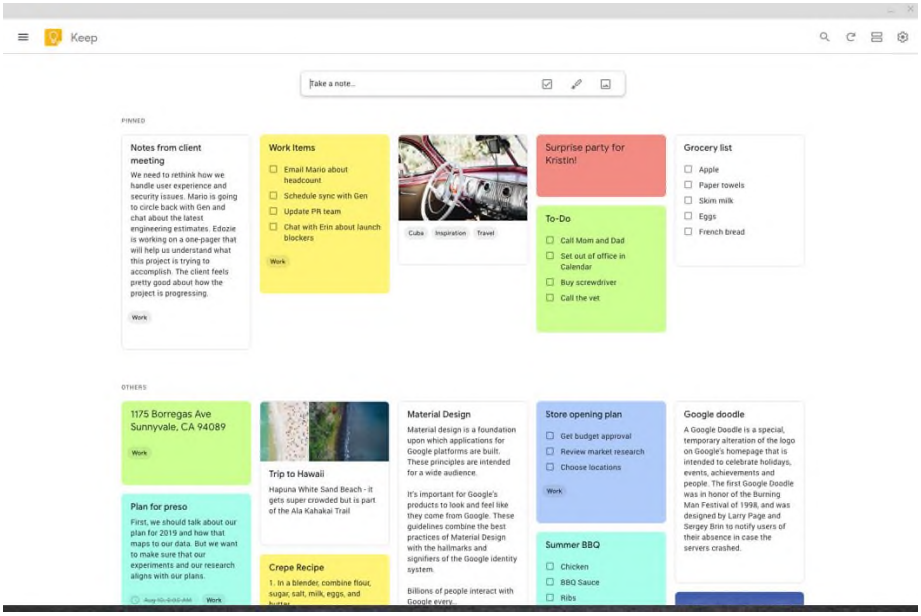


Рисунок 1.1 – Головний екран Google Keep

Microsoft To Do [18] – це універсальна програма для організації завдань, призначена допомогти користувачам ефективно організувати свою повсякденну діяльність. Завдяки чистому та інтуїтивно зрозумілому інтерфейсу (дивитися рисунок 1.2), Microsoft To Do дозволяє користувачам створювати списки, встановлювати нагадування та визначати пріоритети завдань без особливих зусиль [19].

Однією з особливостей програми є її інтеграція з іншими додатками Microsoft 365, що забезпечує уніфікований досвід роботи в екосистемі. Користувачі можуть синхронізувати завдання на різних пристроях, включаючи смартфони, планшети та комп'ютери, забезпечуючи їхню актуальність незалежно від платформи, яку вони використовують. Крім того, Microsoft To Do включає в себе можливості, керовані штучним інтелектом, наприклад, розумні списки, які автоматично класифікують завдання на основі їхніх термінів виконання та важливості.

Загалом, Microsoft To Do є комплексним рішенням для індивідуальних користувачів і команд, які шукають ефективний спосіб керувати своїми завданнями та підвищити продуктивність. Зручний інтерфейс і легка інтеграція з екосистемою Microsoft роблять його цінним інструментом для тих, хто прагне оптимізувати свій робочий процес, незалежно від того, чи йдеться про організацію особистих справ, чи про спільну роботу над робочими проектами.

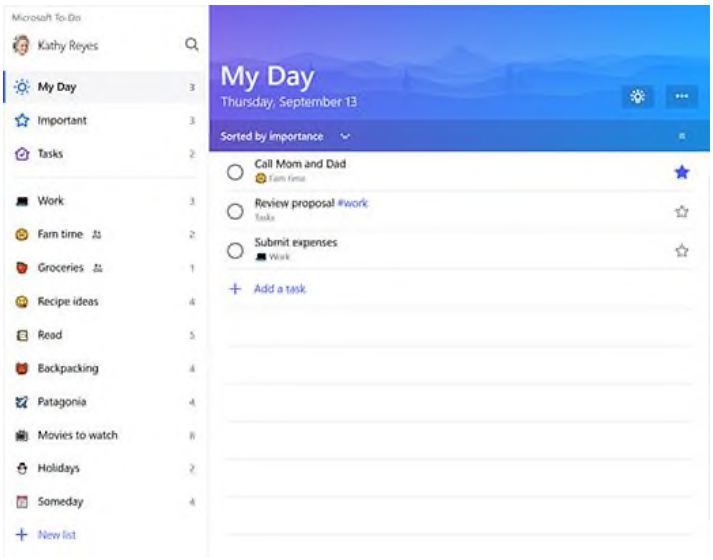


Рисунок 1.2 – Інтерфейс застосунку Microsoft To Do

Trello [20] – популярний інструмент для управління проектами, відомий своїм зручним інтерфейсом та універсальною функціональністю. Він використовує систему дошок, списків і карток, щоб допомогти користувачам організувати завдання і ефективно співпрацювати (дивитися рисунок 1.3). Користувачі можуть створювати дошки для різних проектів або робочих процесів, а потім заповнювати їх списками, що представляють різні етапи або категорії. У межах кожного списку користувачі можуть додавати картки для позначення окремих завдань, ідей або елементів. Ці картки можна налаштовувати за допомогою описів, вкладень, термінів виконання, контрольних списків тощо, що дозволяє детально керувати завданнями.

Однією з ключових переваг Trello є його гнучкість, оскільки він може бути адаптований до різних випадків використання та галузей. Незалежно від того, чи ви керуєте особистими списками справ, чи організовуєте командні проекти, чи координуєте складні робочі процеси, Trello пропонує інструменти, необхідні для того, щоб залишатися організованими та продуктивними. Крім того, Trello підтримує інтеграцію з іншими популярними інструментами та платформами для підвищення продуктивності, що ще більше розширює його можливості та дозволяє користувачам оптимізувати свої робочі процеси. Загалом, інтуїтивно зрозумілий інтерфейс Trello, функції, що налаштовуються, та безперешкодна співпраця

роблять його найкращим вибором для окремих осіб та команд, які прагнуть ефективно керувати завданнями та проектами.

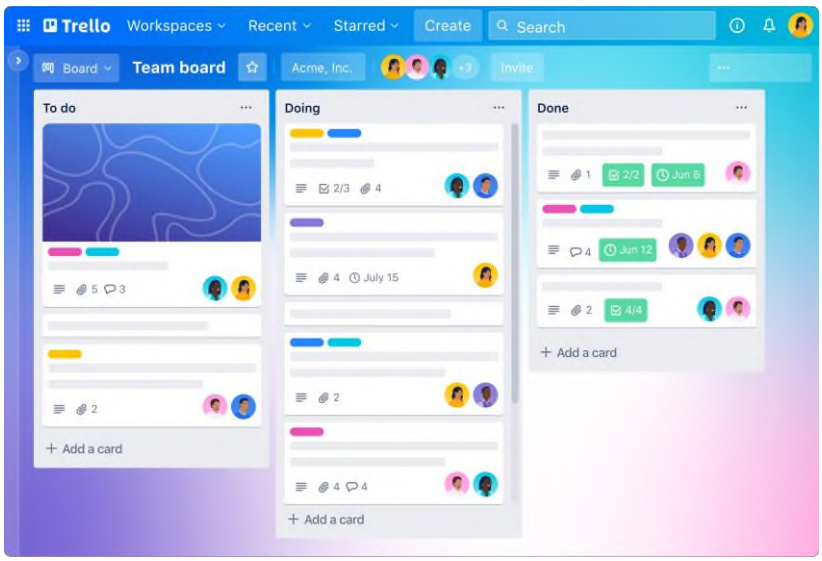


Рисунок 1.3 – Вигляд робочого місця у Trello

У результаті аналізу існуючих рішень для управління завданнями видно, що кожен з розглянутих інструментів має свої переваги та особливості. Google Keep відзначається своєю універсальністю та інтеграцією з екосистемою Google, що робить його зручним використанням для користувачів, які вже користуються іншими сервісами цієї компанії. Microsoft To Do пропонує зручний інтерфейс та інтеграцію з Microsoft 365, що робить його відмінним вибором для користувачів цієї екосистеми. Trello відзначається своєю гнучкістю та адаптованістю до різних сценаріїв використання, що робить його популярним серед різних типів користувачів.

Результати аналізу показують, що кожен з розглянутих інструментів має свої переваги, проте існує потреба в новій платформі, яка комбінує би кращі аспекти цих рішень, не перевантажуючи користувача занадто великою кількістю функцій. Зрозумілий інтерфейс та доступність нового інструменту можуть значно полегшити процес управління завданнями для користувачів різних категорій.

1.3. Постановка задачі

Проаналізувавши існуючі рішення у сфері менеджменту завдань, було виявлено необхідність створення власної системи, що відповідала би потребам користувачів. Основна мета полягає у розробці веб-базованої платформи, яка була би доступною та зручною для щоденного використання. Під час дослідження було виявлено, що більшість популярних рішень спрямовані переважно на корпоративний сектор, що часто ускладнює їх використання для простих користувачів. Тому було прийнято рішення розробити систему з урахуванням потреб таких осіб, забезпечивши її простим та інтуїтивно зрозумілим інтерфейсом.

Цей процес вимагає не лише технічних знань, а й ретельного вивчення потреб цільової аудиторії. Відсутність зайвих функцій та складності у використанні є ключовими аспектами, на які звертається увага під час розробки системи. Це дозволяє забезпечити ефективне використання платформи, щоб задовольнити потреби користувачів у вирішенні їхніх завдань на повсякденній основі.

Такий підхід в розробці дозволяє створити веб-систему, що відповідає сучасним стандартам та вимогам користувачів. Простота та зручність у використанні є основними перевагами, які забезпечують популярність та успішне впровадження застосунку серед користувачів.

Мета проєкту полягає в створенні веб-базованої системи управління особистими завданнями, яка надасть користувачам зручний і ефективний інструмент для організації їхнього робочого та особистого життя. Головні функції системи міститимуть:

– створення особистого кабінету. кожен користувач зможе створити власний обліковий запис для управління своїми завданнями;

– додавання, редагування та видалення завдань. користувачі зможуть легко додавати нові завдання, редагувати їхні параметри та видаляти непотрібні;

– обрання та редагування статусу і пріоритетності завдань. користувачі зможуть встановлювати статус (наприклад, "виконано", "в процесі", "не розпочато") та пріоритетність (наприклад, високий, середній, низький) для кожного завдання;

– встановлення зв'язків між завданнями. користувачі зможуть вказувати зв'язки між різними завданнями, наприклад, вказувати, що одне завдання залежить від іншого або є пов'язане з ним;

– пошук завдань за атрибутами. користувачі можуть шукати завдання за різними атрибутами, такими як назва, дата дедлайну тощо;

– фільтрування завдань. користувачі можуть фільтрувати свої завдання за різними параметрами, такими як статус, категорії, мітки тощо;

– залишання та редагування коментарів: користувачі можуть залишати коментарі до завдань, щоб вказати додаткові відомості або обговорити їх з колегами;

– Перегляд статистики. Користувачі можуть переглядати статистику про виконані та поточні завдання, щоб зрозуміти свій прогрес та ефективність управління часом та завданнями.

Ці функції в сукупності допоможуть користувачам ефективно організовувати свої завдання, прискорюючи процес прийняття рішень та підвищуючи продуктивність.

У завершенні, цей проєкт має потенціал стати корисним інструментом для кожного, хто прагне ефективно керувати своїми завданнями та часом. Забезпечуючи широкий спектр функцій, які включають у себе створення особистого кабінету, редагування та видалення завдань, фільтрування, пошук і аналіз статистики, система надає зручність та контроль користувачам у їхньому щоденному управлінні завданнями. Цей проєкт може стати не тільки практичним інструментом, але і платформою, яка сприяє розвитку особистої організації та продуктивності.

2. АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1. Загальна структура проєктованої системи

Для розробки даного проєкту було використано принцип чистої архітектури (Clean Architecture). Чиста архітектура [21] – це підхід до розробки програмного забезпечення, запропонований Робертом Мартіном, який наголошує на розділенні проблем у додатку для створення гнучкої кодової бази, яку можна легко підтримувати, тестувати та перевіряти. Вона сприяє розподілу проблем шляхом відокремлення програми на окремі шари, кожен з яких відповідає за певну сферу відповідальності. Ці шари розроблені таким чином, щоб бути незалежними від зовнішніх проблем, таких як інтерфейс користувача або база даних, і можуть бути протестовані ізольовано. Основна ідея полягає в тому, щоб підтримувати чітку межу між внутрішнім ядром програми і зовнішніми залежностями. Візуально чисту архітектуру можна представити у вигляді серії концентричних кіл, де внутрішнє коло є ядром програми як можна побачити на рисунку 2.1. Кожне коло представляє окремий рівень з певними обов'язками.

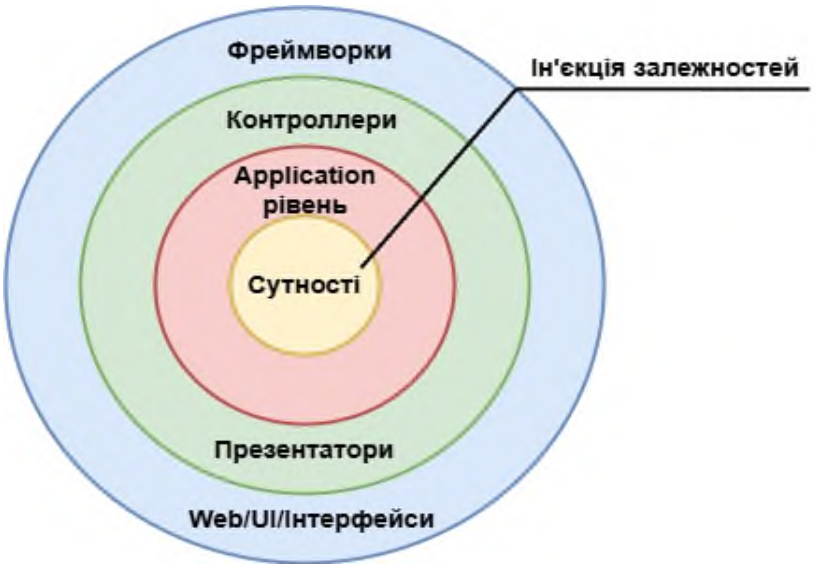
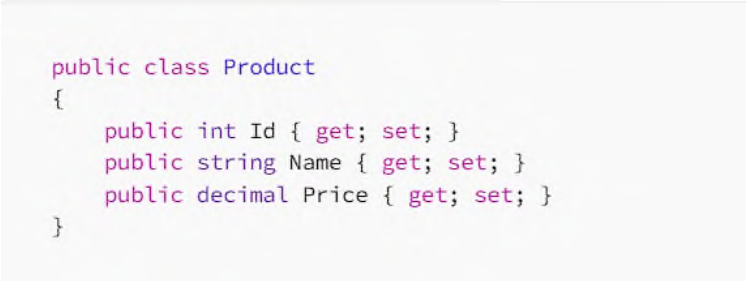


Рисунок 2.1 – Структура чистої архітектури

Для більш детального ознайомлення розберемо ці компоненти. Сутності представляють основні об'єкти програми. Зазвичай це звичайні класи C# [22], які

інкапсулюють основні дані та поведінку вашого домену. Сутності часто є постійними, тобто вони мають унікальний ідентифікатор і можуть бути збережені в базі даних. У програмі C# сутності можуть бути реалізовані як класи або структури. На рисунку 2.2 можна побачити приклад простої сутності на C#.



```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
}
```

Рисунок 2.2 –Приклад простої сутності на C#

Application рівень або Use Cases – це серце бізнес-логіки додатку. Вони інкапсулюють основну функціональність та бізнес-правила додатку. Варіанти використання не залежать від зовнішніх компонентів, таких як база даних або інтерфейс користувача. Натомість вони визначають взаємодію між сутностями та реалізують специфічну поведінку додатку. У C# варіанти використання можуть бути реалізовані у вигляді класів або методів. Спрощений приклад Use Case для створення нового продукту можна побачити на рисунку 2.3.

```
public class CreateProductUseCase
{
    private readonly IProductRepository _productRepository;

    public CreateProductUseCase(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    public void CreateNewProduct(Product product)
    {
        // Business logic for creating a new product
        if (product.Price <= 0)
        {
            throw new ArgumentException("Product price must be greater than 0.");
        }

        _productRepository.Add(product);
    }
}
```

Рисунок 2.3 – Спрощений приклад Use Case

Адаптери інтерфейсу – цей рівень слугує мостом між внутрішнім ядром програми та зовнішнім середовищем, таким як користувацький інтерфейс, бази даних та веб-сервіси. Він містить компоненти, які адаптуються та взаємодіють із зовнішніми фреймворками та інструментами. У C# це можуть бути контролери, презентатори та код доступу до даних. Приклад HTTP-контролера, який використовує CreateProductUseCase для обробки запитів на створення продукту на рисунку 2.4.

```
public class ProductController : ApiController
{
    private readonly CreateProductUseCase _createProductUseCase;

    public ProductController(CreateProductUseCase createProductUseCase)
    {
        _createProductUseCase = createProductUseCase;
    }

    [HttpPost]
    public IActionResult CreateProduct(ProductDto productDto)
    {
        // Map the DTO to a Product entity
        var product = new Product
        {
            Name = productDto.Name,
            Price = productDto.Price
        };

        // Use the CreateProductUseCase
        try
        {
            _createProductUseCase.CreateNewProduct(product);
            return Ok("Product created successfully.");
        }
        catch (ArgumentException ex)
        {
            return BadRequest(ex.Message);
        }
    }
}
```

Рисунок 2.4 – Приклад HTTP-контролера

Фреймворки та драйвери – зовнішній рівень чистої архітектури. Місце, де відбувається інтеграція із зовнішніми фреймворками та інструментами, такими як бази даних, веб-фреймворки та сторонні бібліотеки. Цей рівень містить код,

специфічний для обраного технологічного стеку. У C# це можуть бути контекстні класи бази даних, контролери API та компоненти інтерфейсу користувача.

Для спрощеної роботи із базою даних було використано Entity Framework. Entity Framework (EF) [23] — це об'єктно-реляційний маппер (ORM), який є частиною платформи .NET, розробленої Microsoft. Він дозволяє розробникам працювати з базами даних за допомогою .NET об'єктів, усуваючи необхідність писати більшу частину низькорівневого коду доступу до даних. EF значно полегшує процес взаємодії з базами даних і забезпечує більш чистий та ефективний спосіб управління даними в додатках.

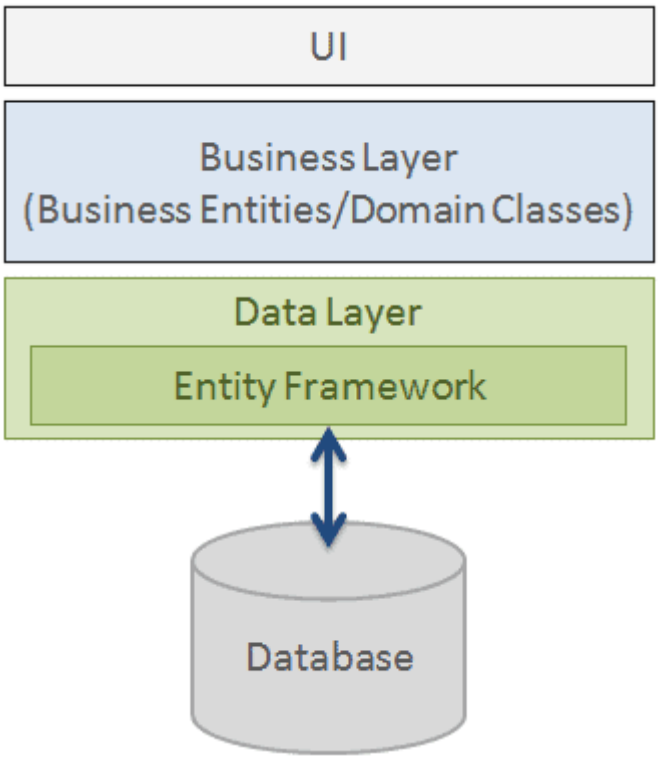


Рисунок 2.5 – Принцип інтеграції Entity Framework із застосунком [24].

Entity Framework містить наступні функції та можливості:

– дозволяє розробникам працювати з базою даних у вигляді об'єктів .NET, а не з реляційними таблицями. Це означає, що таблиці, рядки і стовпці перетворюються у класи, об'єкти та властивості відповідно;

– підтримує LINQ (Language Integrated Query), що дозволяє писати запити до бази даних безпосередньо на C#. Це забезпечує інтуїтивний і зручний спосіб доступу до даних без необхідності писати SQL-запити;

– автоматично відстежує зміни, внесені до об'єктів, і дозволяє легко зберігати ці зміни в базу даних. Це спрощує процес оновлення даних і зменшує кількість коду;

– підтримує механізм міграцій, який дозволяє керувати змінами структури бази даних у часі. Це включає додавання нових таблиць, зміну існуючих або видалення таблиць, зберігаючи при цьому дані;

– дозволяє контролювати, коли і як завантажуються пов'язані дані. Це може бути здійснено за допомогою трьох стратегій: eager loading (завантаження пов'язаних даних разом з основними даними), lazy loading (завантаження пов'язаних даних по мірі потреби) і explicit loading (явне завантаження пов'язаних даних).

2.2. Функціональна модель програмного забезпечення

Для кращого розуміння роботи системи було вирішено побудувати use case діаграму. Use case діаграма — це тип діаграми в мовах моделювання UML, що використовується для відображення функціональних вимог до системи шляхом представлення взаємодії між акторами та системою, що розробляється. Вона показує різні сценарії використання, які описують, як користувачі взаємодіють із системою для досягнення певної мети, допомагаючи виявити функціональність, необхідну для задоволення цих вимог.

Побудована діаграма відображає сценарії використання для користувача в системі управління завданнями. Вона показує основні дії, які користувач може виконувати, а також взаємозв’язки між цими діями. Розглянути діаграму можна на рисунку 2.6 .

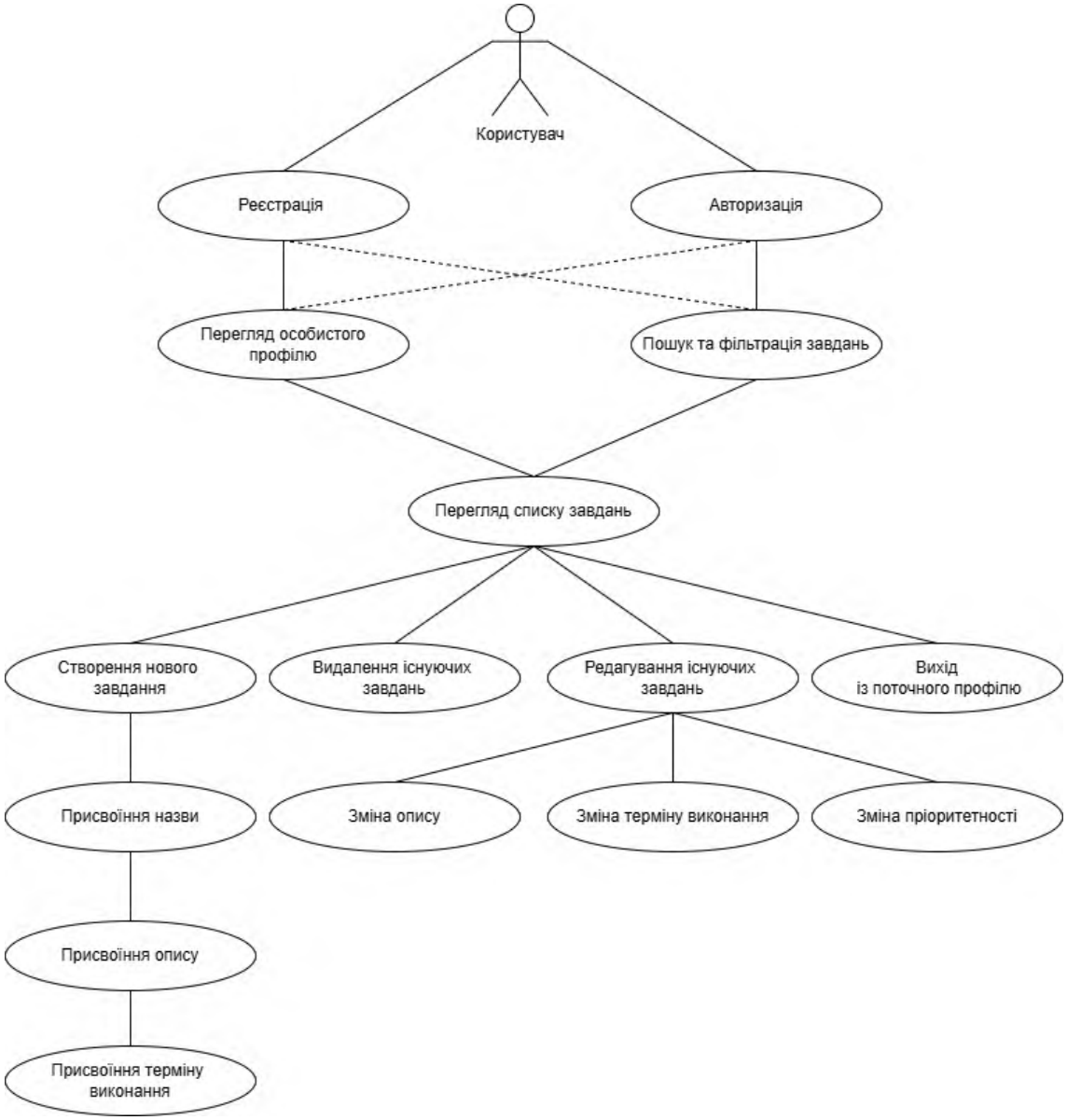


Рисунок 2.6. – Діаграма сценаріїв використання

Для кращого розуміння даної діаграми було розглянуто кожен її пункт детальніше.

Користувач є головною фігурою в цій системі. Всі дії, які відображені на діаграмі, починаються або закінчуються взаємодією користувача із системою. Користувач може бути новим (який ще не зареєстрований) або існуючим (який вже має обліковий запис).

Реєстрація – це перший крок для нових користувачів, які хочуть скористатися системою. Під час реєстрації користувач створює обліковий запис, вводячи необхідну інформацію, таку як ім’я, електронна пошта та пароль. Це дозволяє

користувачу отримати доступ до персоналізованих функцій та зберігати свої завдання.

Авторизація доступна для існуючих користувачів, які можуть увійти в систему за допомогою своїх облікових даних (електронна пошта і пароль). За допомогою неї у користувача є доступ до його особистого профілю та усіх функцій системи. Вона забезпечує безпеку даних користувача, тобто, доступ до завдань може отримати лише їх власник.

Перегляд особистого профілю означає, що після входу в систему користувач може переглядати та редагувати свій особистий профіль. Це включає зміну персональної інформації, налаштування безпеки, перегляд активності, а також інші налаштування, що дозволяють користувачу персоналізувати свій досвід використання системи.

Пошук та фільтрація завдань, за допомогою цієї функції користувач має змогу ефективно знаходити потрібні завдання серед великої кількості. Пошук може здійснюватися за допомогою вводу ключового слова або застосування різних фільтрів (наприклад, за датою, пріоритетом, статусом виконання), щоб швидко знаходити потрібні завдання та керувати ними.

Перегляд списку завдань – це функція, що надає користувачу можливість бачити всі свої завдання в одному місці. Він може бачити назву завдання, опис, термін виконання та пріоритет. Звідси можна виконувати подальші дії, такі як: створення нового завдання, редагування або видалення існуючих.

Створення нового завдання – одна із ключових функцій даної системи. Користувач може створити нове завдання, вводячи необхідну інформацію, а саме: назва, опис та термін виконання. Це дозволяє організувати свої справи та встановити пріоритети для нових задач. Створені завдання автоматично додаються до списку завдань.

Видалення існуючих завдань дозволяє користувачу видаляти завдання, які більше не актуальні або були виконані. Це допомагає підтримувати актуальність та порядок у списку завдань, забезпечуючи зручність використання системи.

Редагування існуючих завдань – необхідна функція, якщо завдання вимагає корекції. Вона включає зміну назви, опису, терміну виконання або пріоритету

завдання. Редагування допомагає підтримувати завдання актуальними та відповідними до поточних потреб користувача.

Присвоєння назви допомагає швидко орієнтуватися в списку завдань. При створенні або редагуванні завдання користувач може присвоїти йому назву, яка коротко відображатиме суть завдання.

Присвоєння опису допомагає уникнути непорозумінь та забезпечує чіткість завдань. Додатково до назви, користувач може додати детальний опис завдання, що пояснює, що саме треба зробити.

Присвоєння терміну виконання – це функція, що допомагає користувачу планувати свій час і забезпечує своєчасне виконання завдань, оскільки кожне завдання може мати кінцевий термін виконання.

Вихід із поточного профілю необхідний, коли користувач завершує роботу в системі. Також вихід із облікового запису важливий для безпеки даних. Оскільки завдяки цій можливості ніхто інший не зможе отримати доступ до персональної інформації та завдань без відповідної авторизації.

2.3. Інформаційне забезпечення

Проектування інформаційного забезпечення є критично важливим етапом у розробці будь-якої системи, що передбачає створення ефективної структури для збору, зберігання, обробки та передачі даних. Воно спрямоване на забезпечення надійності, доступності та безпеки інформації, що сприяє ефективному функціонуванню організації або проєкту.

Microsoft SQL Server [25] є розгорнутою системою управління реляційними базами даних (RDBMS), розробленою та підтримуваною компанією Microsoft. Цей продукт є однією з основних платформ для зберігання, маніпуляції, управління та аналізу даних, що використовуються як у малих, так і у великих підприємствах. SQL Server забезпечує надійну, масштабовану та безпечну платформу для роботи з базами даних, що робить його популярним вибором серед різних організацій та розробників.

MS SQL Server підтримує широкий спектр функцій, які забезпечують ефективне управління даними:

– Транзакційна обробка: підтримка ACID (Atomicity, Consistency, Isolation, Durability) гарантує, що транзакції виконуються надійно та без втрати даних.

– Індексація та оптимізація запитів: різноманітні типи індексів дозволяють оптимізувати запити та підвищити продуктивність системи.

– Безпека: MS SQL Server забезпечує високий рівень безпеки даних за допомогою механізмів аутентифікації, авторизації, шифрування та управління доступом.

– Реплікація та резервування: інструменти для реплікації даних та резервного копіювання забезпечують високу доступність та відновлюваність даних у разі збоїв.

– Повнотекстовий пошук: підтримка повнотекстового індексування та пошуку дозволяє виконувати складні текстові запити з високою швидкістю.

– Підтримка різних типів даних: MS SQL Server підтримує не лише традиційні реляційні дані, але й JSON, XML, просторові дані та інші типи даних.

Microsoft SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Azure, Power BI, Dynamics 365, що дозволяє створювати комплексні рішення для аналізу та управління даними. Крім того, SQL Server підтримує інтеграцію з різними мовами програмування та платформами, такими як .NET, Java, Python, що забезпечує гнучкість у розробці додатків.

Таким чином, Microsoft SQL Server є надійним та гнучким інструментом для управління реляційними базами даних. Його багатофункціональність, висока продуктивність, інтеграція з іншими продуктами Microsoft та підтримка різних типів даних роблять його незамінним для широкого спектру застосувань, від малих додатків до великих корпоративних систем.

На рисунку 2.7 представлена схема бази даних, яка містить дев’ять таблиць, використовуваних для управління користувачами та їх ролями. Розглянемо детальніше кожну із них та її сутності.

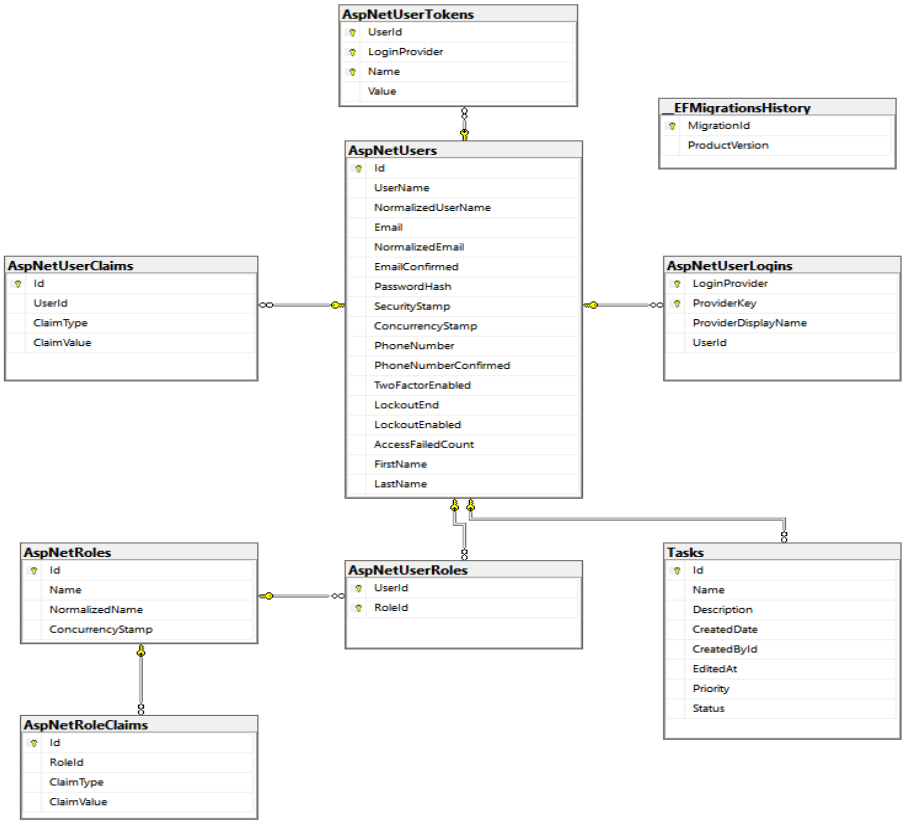


Рисунок 2.7 – Діаграма бази даних.

AspNetUsers зберігає інформацію про користувачів. Вона містить сутності, які показано в таблиці 2.1.

Таблиця 2.1 – Таблиця AspNetUsers із її сутностями.

Найменування сутності	Значення
Id	Унікальний ідентифікатор користувача
UserName	Ім'я користувача
NormalizedUserName	Нормалізоване ім'я користувача для порівняння
Email	Електронна пошта користувача
NormalizedEmail	Нормалізована електронна пошта для порівняння
EmailConfirmed	Підтвердження електронної пошти
PasswordHash	Хеш паролю користувача
SecurityStamp	Мітка безпеки для перевірки автентичності користувача
ConcurrencyStamp	Мітка для контролю конкурентності

Продовження таблиці 2.1

PhoneNumber	Номер телефону користувача
PhoneNumberConfirmed	Підтвердження номера телефону
TwoFactorEnabled	Прапорець, чи включено двофакторну аутентифікацію
LockoutEnd	Дата та час закінчення блокування користувача
LockoutEnabled	Прапорець, чи включено блокування користувача
AccessFailedCount	Кількість невдалих спроб входу
FirstName	Ім'я користувача
LastName	Прізвище користувача

AspNetRoles зберігає інформацію про ролі. Вона містить такі сутності, які показано в таблиці 2.2.

Таблиця 2.2 – Таблиця AspNetRoles із її сутностями.

Найменування сутності	Значення
Id	Унікальний ідентифікатор ролі
Name	Назва ролі
NormalizedName	Нормалізована назва ролі для порівняння
ConcurrencyStamp	Мітка для контролю конкурентності

AspNetUserRoles використовується для зв'язку користувачів з ролями. Вона містить дві основні сутності, які показано в таблиці 2.3.

Таблиця 2.3 – Таблиця AspNetUserRoles із її сутностями.

Найменування сутності	Значення
UserId	Ідентифікатор користувача, зовнішній ключ з таблиці aspnetusers
RoleId	Ідентифікатор ролі, зовнішній ключ з таблиці aspnetroles

AspNetUserClaims зберігає клейми користувачів. Вона містить такі сутності, які показано в таблиці 2.4.

Таблиця 2.4 – Таблиця AspNetUserClaims із її сутностями.

Найменування сутності	Значення
Id	Унікальний ідентифікатор
UserId	Ідентифікатор користувача, зовнішній ключ з таблиці aspnetusers
ClaimType	Тип клейму
ClaimValue	Значення клейму

AspNetUserLogins зберігає інформацію про логіни користувачів через різних провайдерів. Вона містить такі сутності, які показано в таблиці 2.5.

Таблиця 2.5 – Таблиця AspNetUserLogins із її сутностями.

Найменування сутності	Значення
LoginProvider	Провайдер логіну
ProviderKey	Ключ провайдера
ProviderDisplayName	Відображуване ім'я провайдера
UserId	Ідентифікатор користувача, зовнішній ключ з таблиці aspnetusers

AspNetUserTokens зберігає токени користувачів. Вона містить такі сутності, які показано в таблиці 2.6.

Таблиця 2.6 – Таблиця AspNetUserTokens із її сутностями.

Найменування сутності	Значення
UserId	Ідентифікатор кор., зовн. ключ з таб. aspnetusers
LoginProvider	Провайдер логіну
Name	Ім'я токена
Value	Значення токена

AspNetRoleClaims зберігає клейми ролей. Вона містить такі сутності, які показано в таблиці 2.7.

Таблиця 2.7 – Таблиця AspNetRoleClaims та її сутності.

Найменування сутності	Значення
Id	Унікальний ідентифікатор
RoleId	Ідентифікатор ролі, зовнішній ключ з таб. aspnetroles
ClaimType	Тип клейму
ClaimValue	Значення клейму

Tasks зберігає інформацію про задачі. Вона містить такі сутності, які показано в таблиці 2.8.

Таблиця 2.8 – Таблиця Tasks із її сутностями.

Найменування сутності	Значення
Id	Унікальний ідентифікатор завдання
Name	Назва завдання
Description	Опис завдання
CreatedDate	Дата створення завдання
CreatedById	Ідентифікатор користувача, який створив завдання
EditedAt	Дата останнього редагування завдання
Priority	Пріоритет завдання
Status	Статус завдання

EFMigrationsHistory зберігає інформацію про міграції бази даних. Вона містить такі сутності, які показано в таблиці 2.9.

Таблиця 2.9 – Таблиця EFMigrationsHistory та її сутності.

Найменування сутності	Значення
MigrationId	Ідентифікатор міграції
ProductVersion	Версія продукту

3. ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Реалізація програмного забезпечення

У даному розділі розглядається представлено детальний аналіз діаграм класів, що використовуються для візуалізації структури системи. Оскільки для розробки було використано принцип чистої архітектури, проєкт розділено на чотири окремі проєкти. По кожному із них було побудовано діаграму класів, окрім application рівня, адже в ньому немає жодних класів.

На рисунку 3.1 наведено domain діаграму класів, що ілюструє основні сутності та їх взаємозв'язки в системі управління завданнями. Центральною сутністю є TaskEntity, яка містить властивості, що характеризують завдання: CreatedBy, CreatedById, CreatedDate, Description, EditedAt, Id, Name, Priority та Status. Ці властивості дозволяють відстежувати авторство, дати створення та редагування, а також статус і пріоритет завдання. Сутність UserEntity успадковується від класу IdentityUser і включає додаткові властивості FirstName та LastName, що дозволяють зберігати ім'я та прізвище користувача, а також колекцію завдань, створених цим користувачем. Класи Status та TaskPriority представлені як перерахування, що визначають можливі значення статусу завдання (NotStarted, InProgress, Completed) та його пріоритету (Low, Medium, High). Це забезпечує структуроване та типізоване зберігання даних про завдання, підвищуючи ефективність їх обробки та інтеграції в системі.

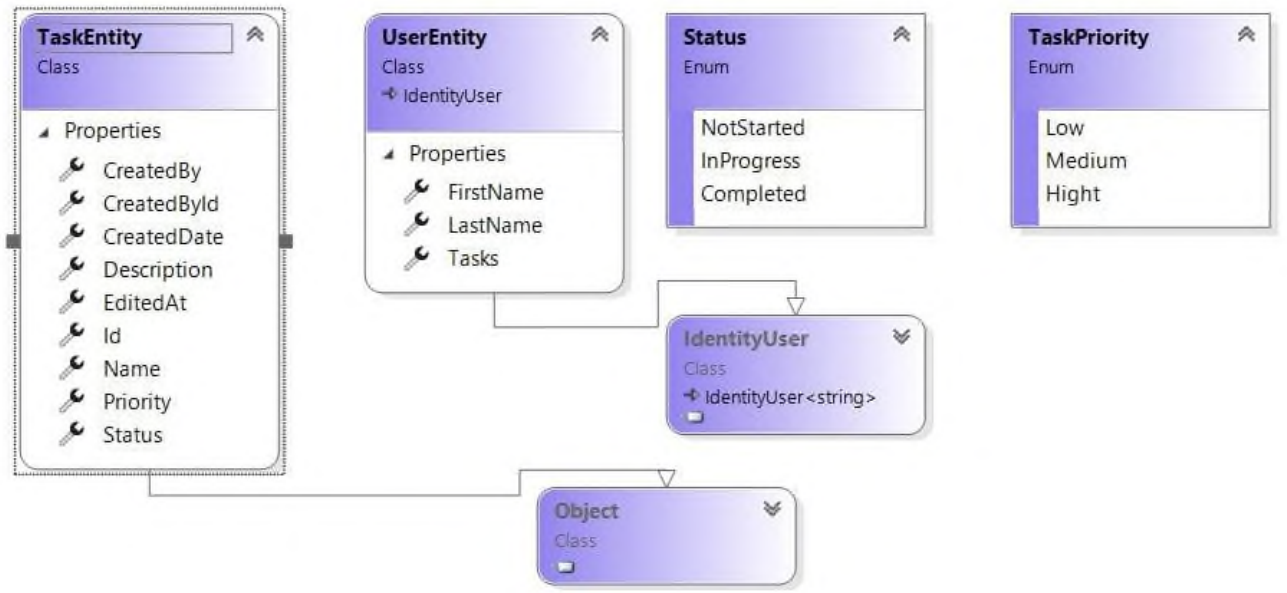


Рисунок 3.1 – Діаграма класів domain рівня

Діаграма класів persistance рівня (рисунок 3.2) представляє архітектуру та взаємозв'язки між різними класами, залученими в управління даними в додатку. Діаграма ілюструє ієрархію та зв'язки між класами, відповідальними за міграції бази даних та управління контекстом у додатку. В основі знаходиться клас TaskManagementDbContext, який успадковується від IdentityDbContext і управляє сутностями, такими як Task та UserEntity. Він включає властивості та методи, необхідні для взаємодії з базою даних, наприклад, OnModelCreating. Міграції представлені такими класами, як Initial, SeedData, UpdateUserEntity та UpdatetaskEntity, кожен з яких містить методи для побудови цільових моделей (BuildTargetModel) та виконання операцій (Up, Down). Класи TaskEntityConfiguration та TaskManagementSnapshot визначають конфігурації схеми та знімки стану моделі. Абстрактні класи Migration та ModelSnapshot слугують базовими класами, які інкапсулюють загальні функції, що використовуються похідними класами для підтримання цілісності та версіонування схеми бази даних. Ця структура забезпечує надійне та масштабоване управління персистенцією даних у системі.

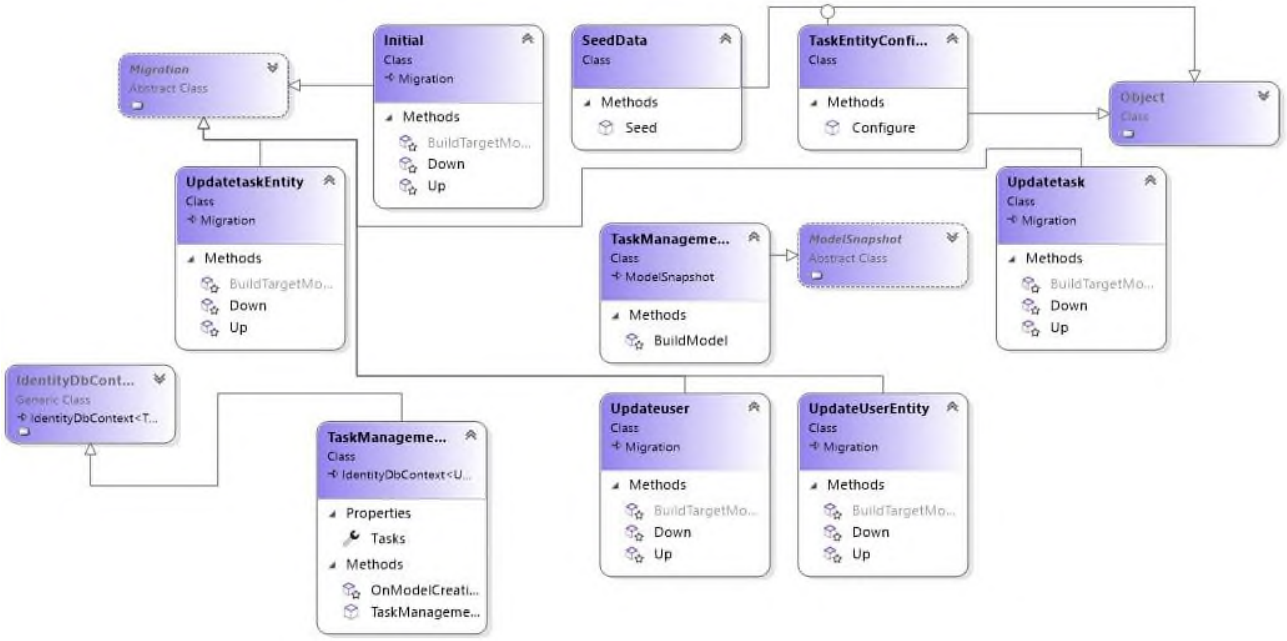


Рисунок 3.2 – Діаграма класів persistence рівня

Наведена діаграма класів presentation рівня (рисунок 3.3) ілюструє структуру та взаємозв'язки між класами, що відповідають за обробку представлення даних у додатку. Центральним абстрактним класом є PageModel, який слугує базою для інших моделей сторінок, таких як LoginModel, LogoutModel та RegisterModel. Ці класи наслідують загальні властивості та методи, що дозволяє забезпечити узгодженість у роботі з різними сторінками. Класи CreateTaskCommand, DeleteTaskCommand, EditTaskCommand та відповідні їм обробники (CreateTaskHandler, DeleteTaskHandler, EditTaskHandler) відповідають за виконання CRUD-операцій над завданнями. Вони використовують поля для зберігання контексту бази даних та інших необхідних сервісів, забезпечуючи таким чином належну обробку даних. Додаткові класи, такі як CreateTaskModel, EditTaskModel, SearchTaskModel, TaskModel та ErrorViewModel, визначають структуру даних, які будуть використовуватися у відповідних операціях та представленнях. Кожен з цих класів містить властивості, що відповідають за опис, ідентифікацію та статус завдань, що дозволяє підтримувати чітку організацію та доступ до даних у межах системи. Ця структура забезпечує ефективне управління операціями на рівні представлення та підвищує зручність роботи з додатком для кінцевого користувача.

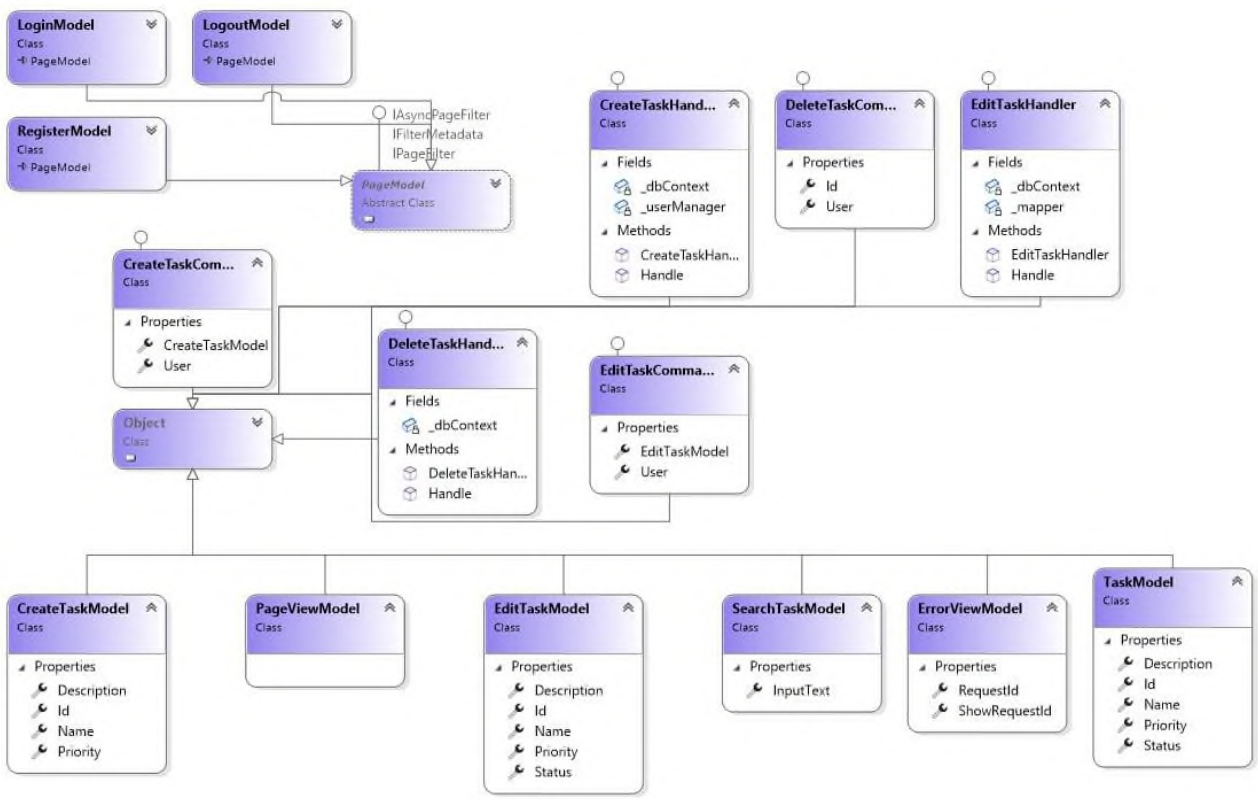


Рисунок 3.3 – Діаграма класів presentation рівня

3.2. Користувацький інтерфейс

Під час роботи над даним проєктом було розроблено веб-базовану систему управління особистими завданнями. Для більш детального ознайомлення нижче представлено зображення користувацького інтерфейсу.

На рисунку 3.4 зображено веб-сторінку входу до системи управління завданнями. Інтерфейс містить форму для аутентифікації користувачів, яка включає поля для введення електронної пошти та пароля, а також опцію "Remember me?" для збереження сеансу. Над формою розміщене привітання "Welcome back!", яке вказує на те, що сторінка призначена для повторного входу користувачів. Вгорі сторінки розташоване навігаційне меню з посиланнями для реєстрації ("Register") та входу ("Login"), а внизу знаходиться інформація про авторські права та політику конфіденційності. Ця сторінка забезпечує базову функціональність для контролю доступу до системи шляхом перевірки облікових даних користувачів.

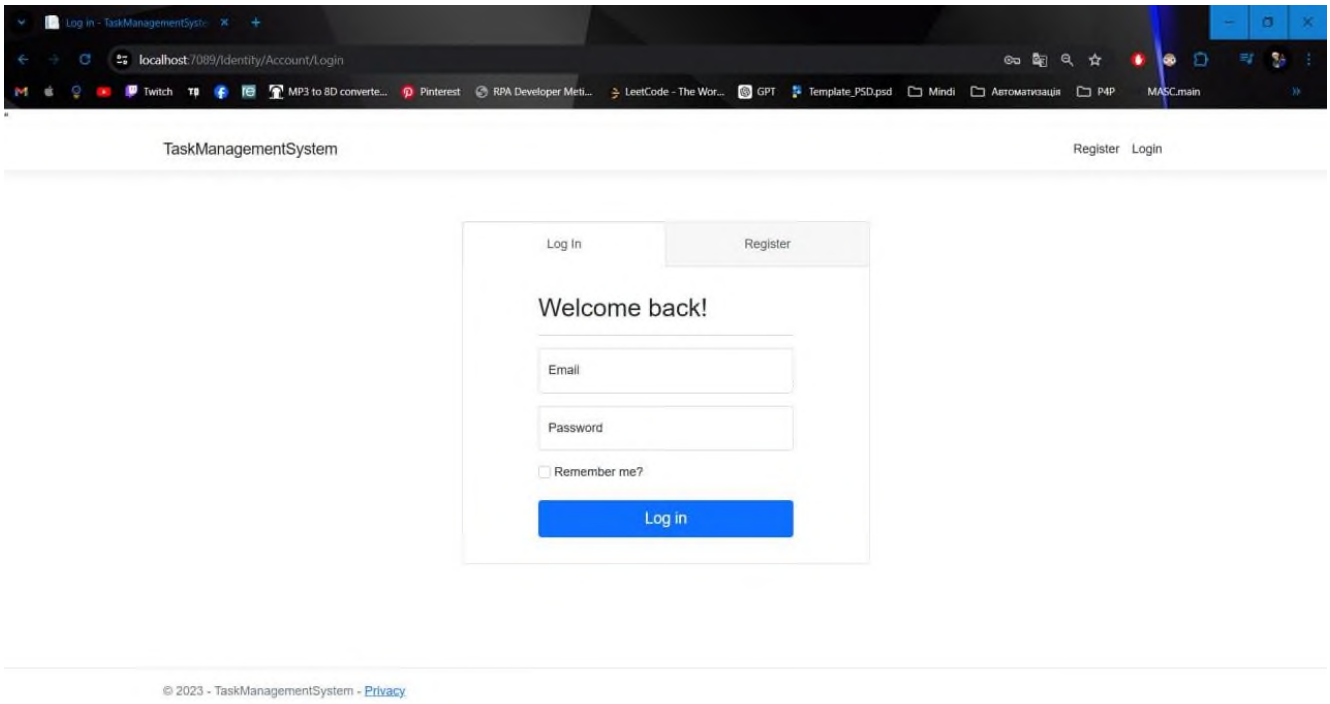


Рисунок 3.4 – Головна сторінка входу у систему

Рисунок 3.5 показує реєстрацію у системі. Інтерфейс містить форму для створення нового облікового запису, яка включає поля для введення імені, прізвища, електронної пошти, пароля та підтвердження пароля. Над формою розміщене заголовок "Create a new account", який вказує на призначення сторінки. Вгорі сторінки знаходиться навігаційне меню з вкладками для входу ("Log In") та реєстрації ("Register"), що дозволяє користувачам перемикатися між цими функціями. Ця сторінка надає користувачам можливість створення нових облікових записів шляхом введення необхідної інформації, що є важливим етапом у процесі забезпечення безпеки та управління доступом до системи.

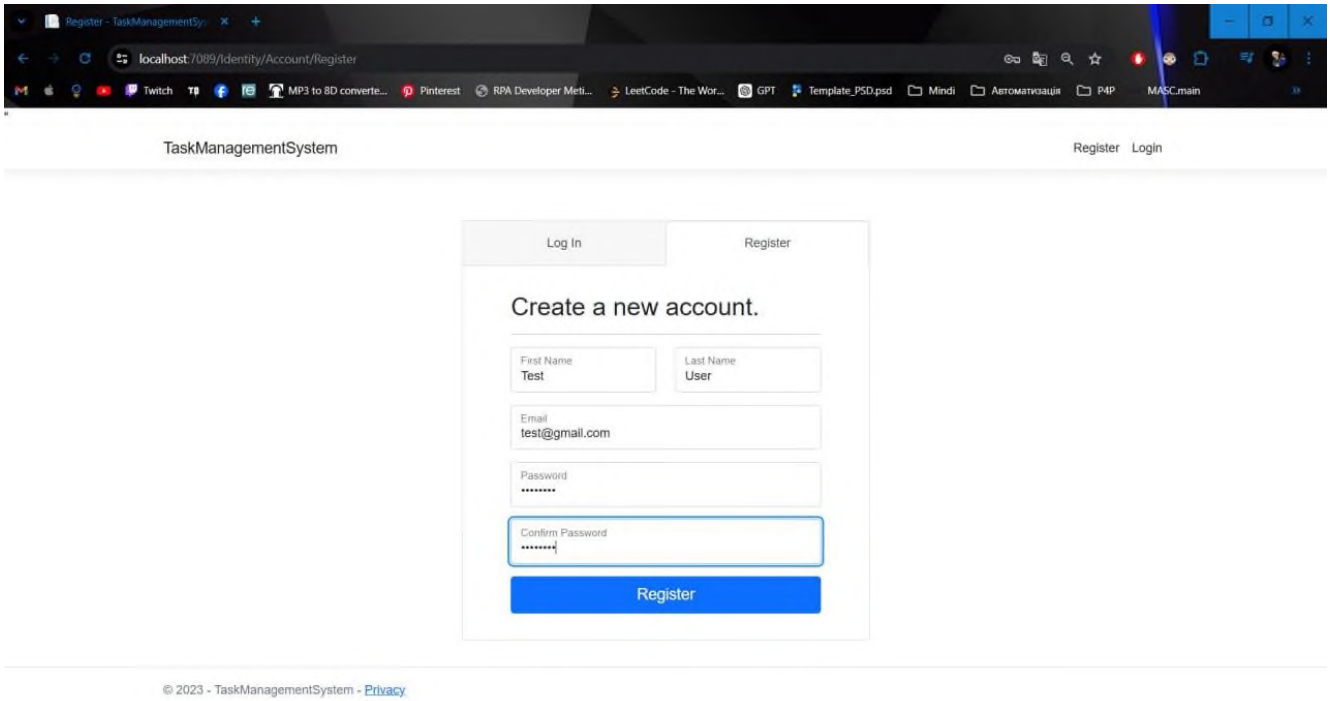


Рисунок 3.5 – Сторінка з формою для реєстрації

Наступне вікно рисунок 3.6 представляє веб-сторінку аутентифікації з попередньо заповненою формою для входу. У ній користувач вже відмітив опцію "Remember me?", яка дозволяє зберігати дані для автоматичного входу в майбутньому. Над формою розташоване привітання "Welcome back!", яке вказує на те, що користувач повертається до системи. Навігаційне меню у верхній частині сторінки містить вкладки для переходу до реєстрації ("Register") та входу ("Login"). Ця сторінка є ключовою для контролю доступу до системи, надаючи безпечний вхід користувачів за допомогою перевірки їх облікових даних.

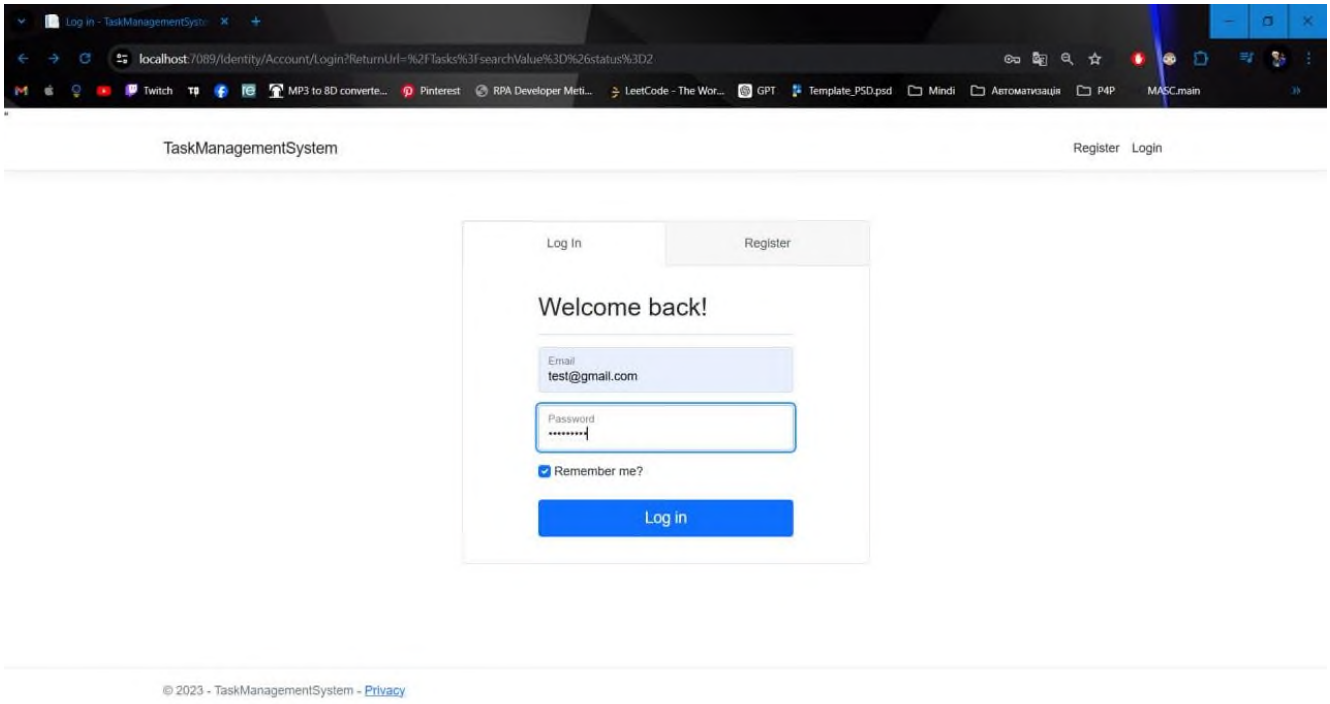


Рисунок 3.6 – Сторінка із логінізацією в систему.

Рисунку 3.7 зображує головну сторінку після входу у систему, яка відображає перелік завдань користувача. В центральній частині сторінки знаходиться заголовок "Your Tasks" та кнопка "Create" для створення нового завдання. Під заголовком розміщено панель пошуку та фільтрації, що дозволяє користувачеві знаходити та сортувати завдання за певними критеріями. Нижче наведено три картки завдань з відповідними назвами, описами, статусами та пріоритетами. Кожне завдання має опції для редагування ("Edit") та видалення (позначено хрестиком). У верхньому правому куті сторінки знаходиться привітання користувача "Hello - Test!" та кнопка "Logout" для виходу з системи. Ця сторінка є центральною у взаємодії користувача із системою, надаючи можливості управління, організації та контролю виконання завдань.

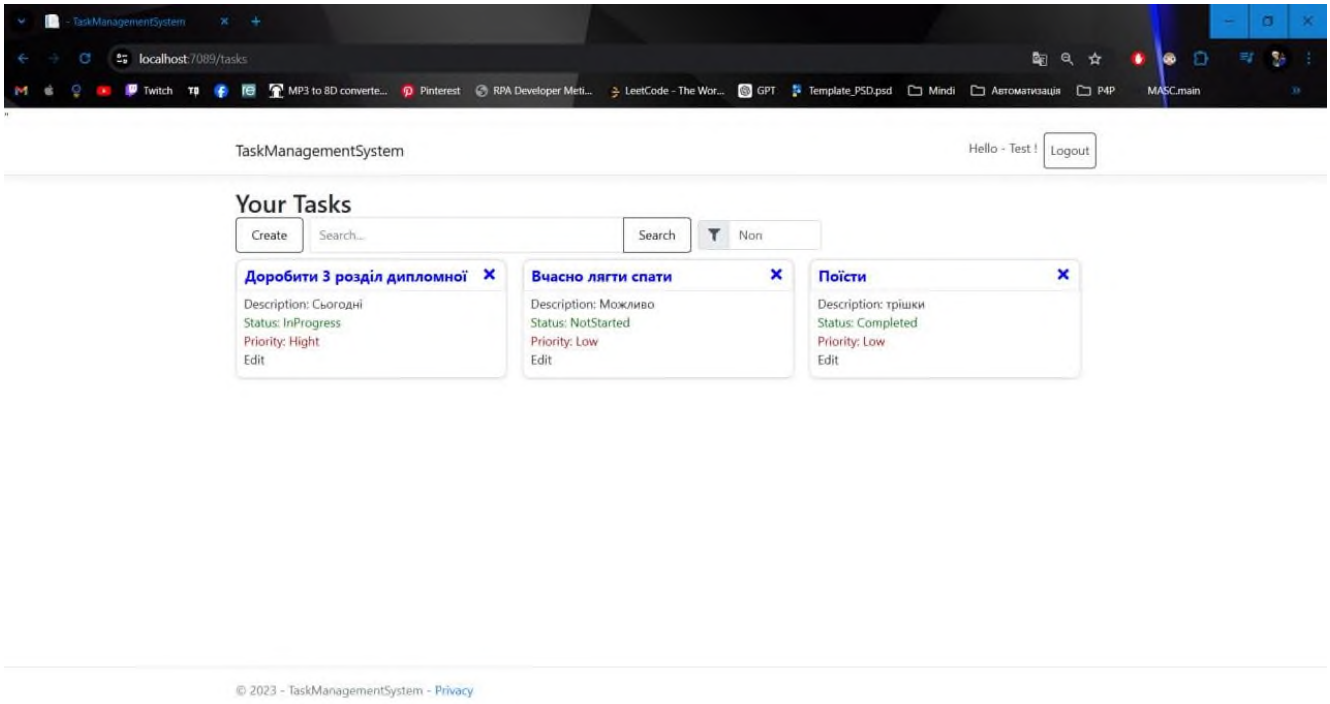


Рисунок 3.7 – Інтерфейс сторінки авторизованого користувача.

У вікні на рисунку 3.8 можна побачити процес створення нового завдання, що є ключовим функціоналом застосунка. У центрі зображення знаходиться модальне вікно під назвою "New task", яке містить форму для введення деталей завдання. Форма включає поля для назви завдання ("Task Name"), опису завдання ("Task Description"), та рівня пріоритету ("Priority"). Поле для назви завдання є обов'язковим для заповнення, про що свідчить червоне повідомлення "The Name field is required". Користувач може натиснути кнопку "Create" для створення завдання або "Cancel" для скасування операції. Ця функціональність є важливою для ефективного управління завданнями, дозволяючи користувачам додавати нові завдання з відповідними параметрами.

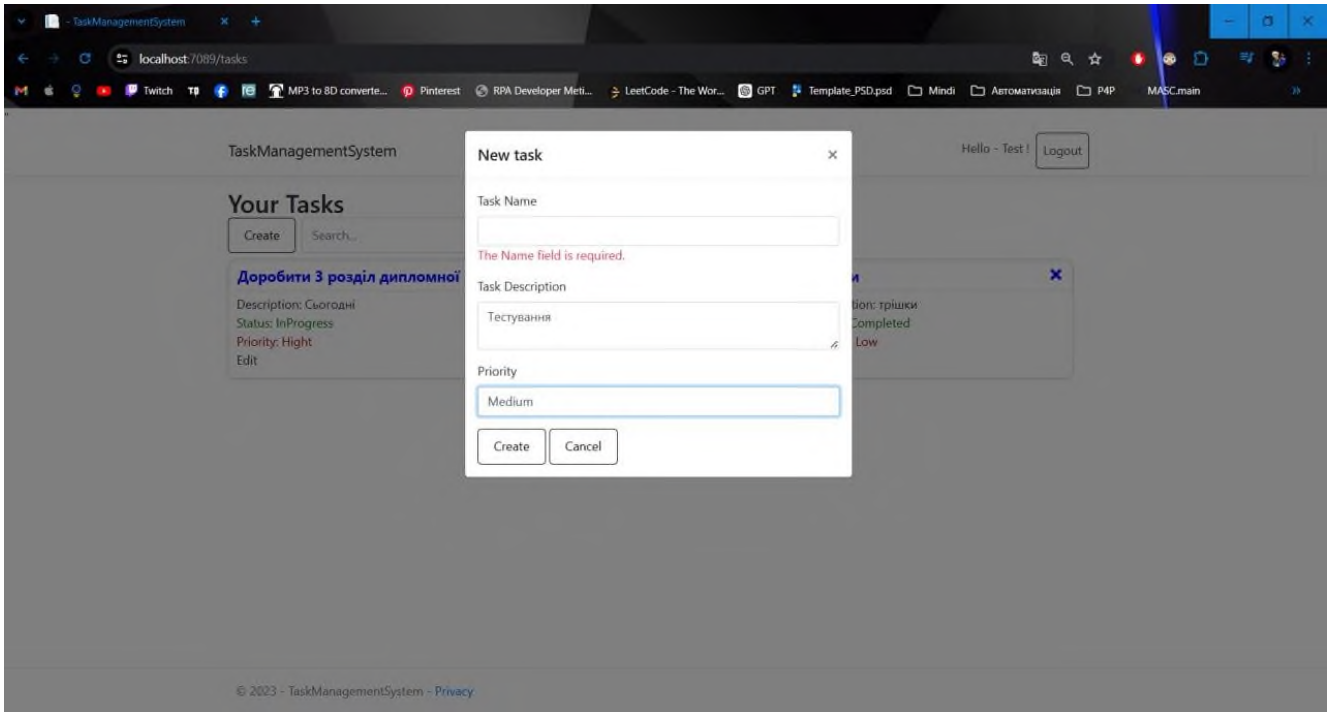


Рисунок 3.8 – Модальне вікно із створенням нового завдання.

У центрі рисунка 3.9 знаходиться модальне вікно, яке використовується для редагування завдання. У ньому є кілька полів для введення та редагування даних: "Task Name", "Task Description", "Priority" та "Status". Аналогічно, як і під час створення завдання, знизу вікна розташовані дві кнопки: "Save" та "Cancel". Загалом, зображення демонструє процес редагування завдання у системі управління завданнями, що включає можливість змінювати назву, опис, пріоритет та статус завдання, з можливістю збереження або скасування змін.

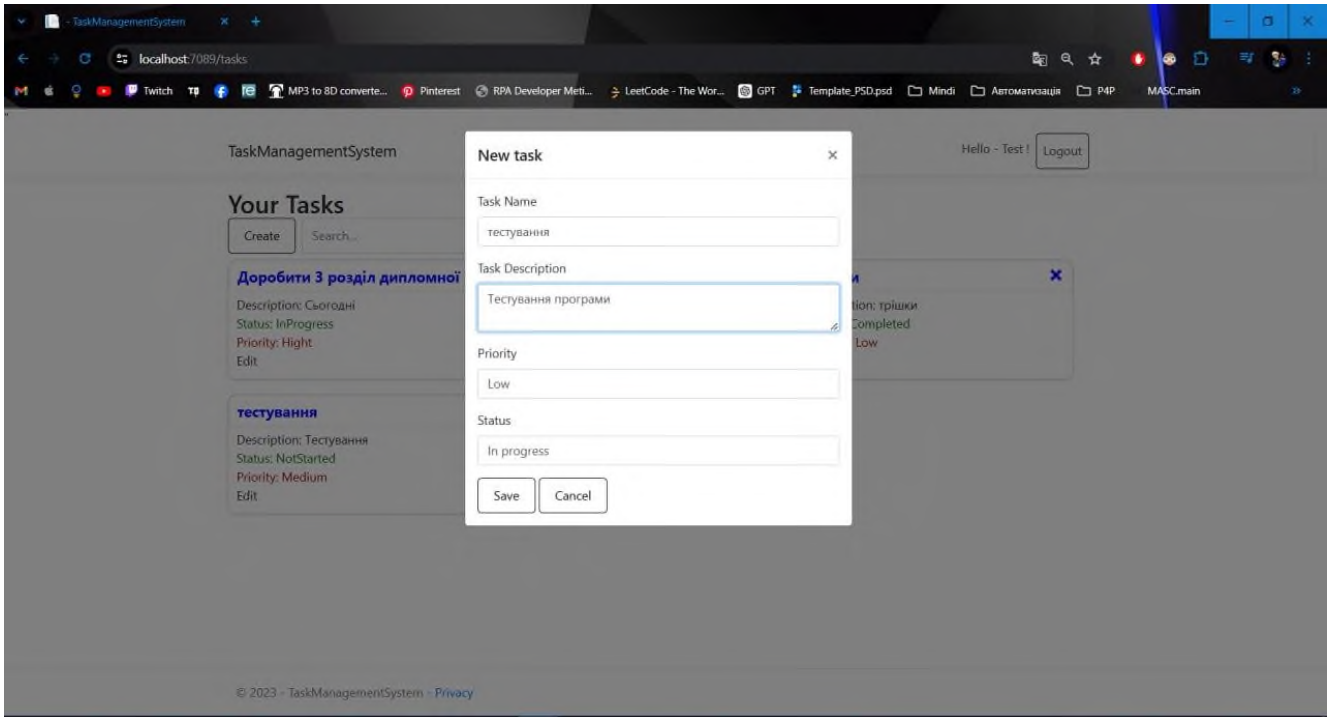


Рисунок 3.9 – Модальне вікно із редагуванням завдання.

На рисунку 3.10 показано модальне вікно "Delete тестування task?", яке запитує користувача про підтвердження видалення завдання з назвою "тестування". Це модальне вікно має дві кнопки: "Cancel" та "Confirm". Кнопка "Cancel" призначена для скасування операції видалення та закриття модального вікна без жодних змін. Кнопка "Confirm" призначена для підтвердження операції видалення завдання. Такий підхід підвищує безпеку та запобігає випадковому видаленню важливих даних.

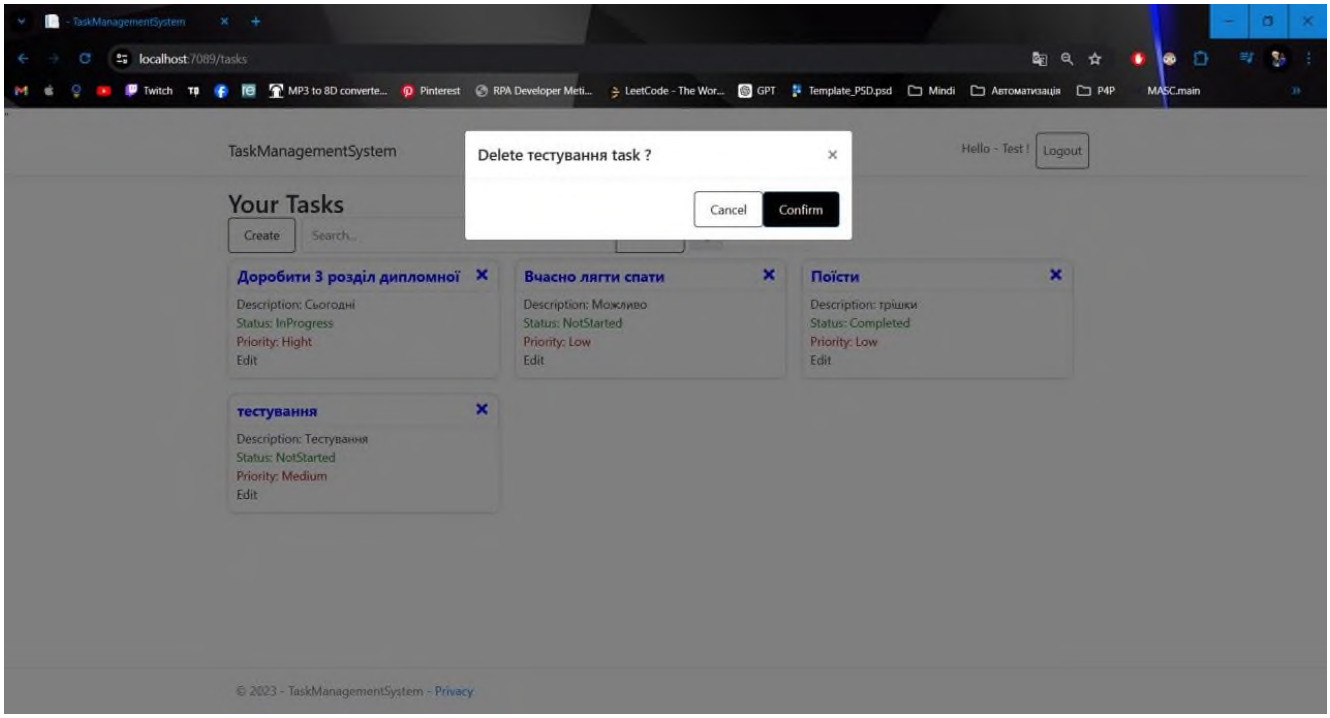


Рисунок 3.10 – Модальне вікно із видаленням завдання.

На рисунку 3.11 представлено функцію пошуку за ключовими словами. У полі пошуку введено текст "Допо", що відповідає частині назви завдання "Доробити 3 розділ дипломної". В результаті на сторінці відображається одне завдання, що відповідає введеному ключовому слову.

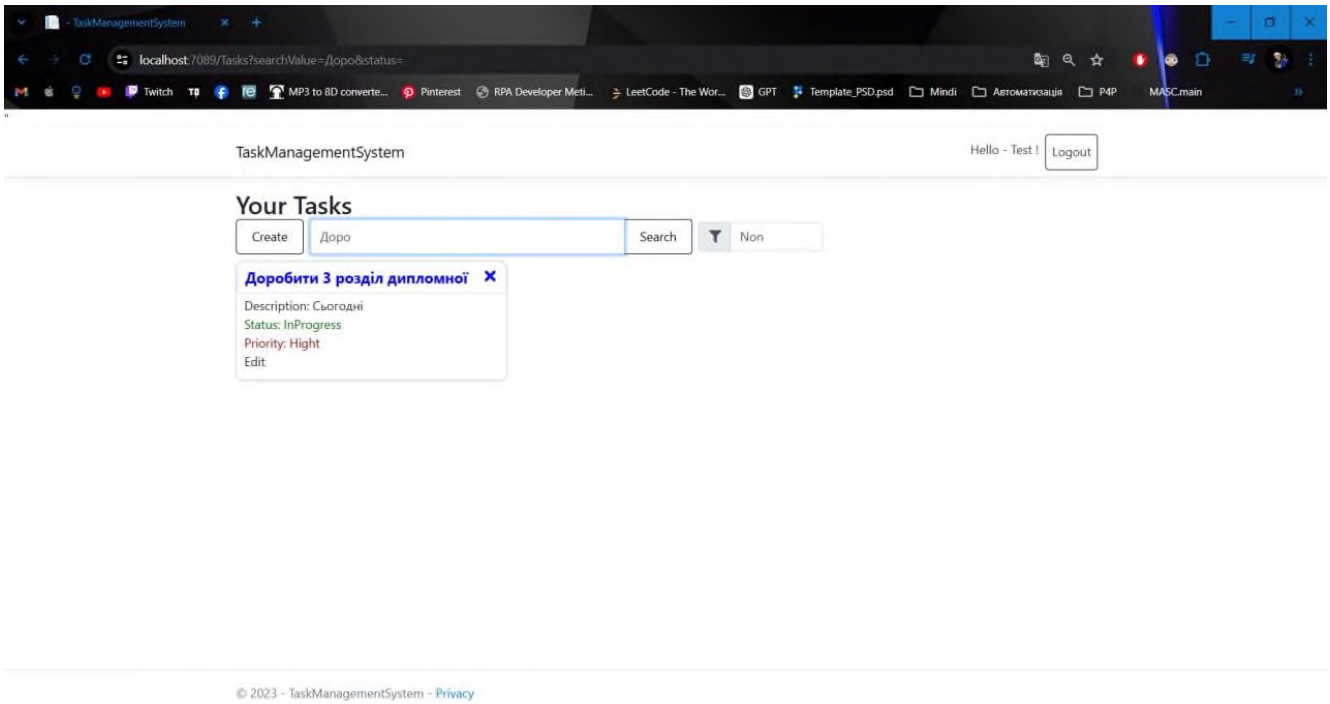


Рисунок 3.11 – Показ функції пошуку за ключовими словами.

На рисунку 3.12 було показано роботу ще однієї функції – фільтрації за статусом завдання. Праворуч від кнопки "Search" розташована кнопка з іконкою фільтра для додаткових налаштувань фільтрації. Відкрите меню фільтрації показує можливість вибору статусу завдань: "Non" (жодне), "Not Started" (не почате), "In progress" (виконується) та "Completed" (виконане). У цьому випадку вибрано статус "Completed". У результаті фільтрації на сторінці відображається лише одне завдання, яке відповідає обраному статусу, – "Поїсти".

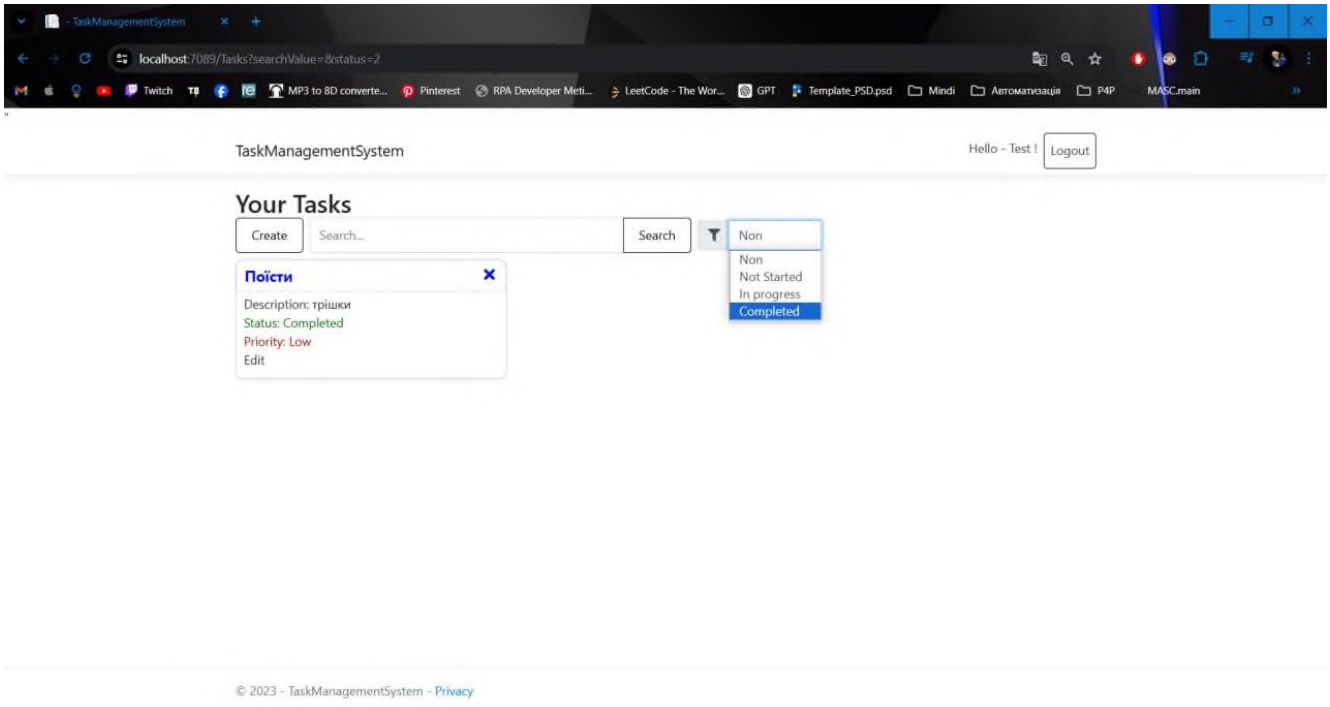


Рисунок 3.12 – Зображення функції фільтрації за статусом завдання.

3.3. Тестування розробленої системи

Задля покращення роботи системи і виявлення помилок було використано Swagger. Swagger [26] — це набір відкритих інструментів і специфікацій для проектування, створення, документування та споживання RESTful веб-сервісів. Основною компонентою Swagger є специфікація OpenAPI [27], яка забезпечує стандартизований формат для опису API, включаючи кінцеві точки, методи запитів, параметри та типи даних.

Swagger використовується для автоматизації та спрощення процесу тестування RESTful API [28] , забезпечуючи зручні інструменти для генерації тестових сценаріїв та інтерактивного тестування. Завдяки специфікації OpenAPI, Swagger дозволяє автоматично створювати точні та актуальні тестові кейси, що базуються на специфікаціях API, забезпечуючи верифікацію відповідності між документованими та фактичними поведінками сервісу. Використання Swagger у тестуванні допомагає виявляти помилки на ранніх етапах розробки, забезпечуючи високу якість та надійність веб-сервісів.

На початку роботи із Swagger було згенеровано JSON Web Token (дивитися рисунок 3.13). JWT [29] – це компактний, автономний спосіб безпечного обміну інформацією між сторонами у вигляді JSON-об'єкта. Він використовується для підтвердження автентичності та авторизації користувачів у веб-додатках. Ключова особливість JWT полягає в тому, що він складається з трьох частин: заголовка, корисного навантаження і криптографічного підпису. Заголовок містить інформацію про тип токена та алгоритм шифрування, корисне навантаження включає заявлені твердження про суб'єкта токена, а підпис забезпечує цілісність та автентичність токена. JWT не вимагає зберігання сесійної інформації на сервері, що робить його зручним для розподілених систем і масштабованих додатків. Інтеграція JWT із Swagger дозволяє документувати та тестувати захищені API. Користувачі можуть вводити JWT токени для доступу до захищених ендпоїнтів безпосередньо через Swagger-інтерфейс, що забезпечує зручне та безпечне тестування API.

Рисунок 3.13 – Повідомлення про згенерований JWT токен.

У ході роботи було проведено тестування усіх функцій розроблюваної системи, результати якого, можна побачити в Додаток А.

Приклади успішного тестування функціоналу можна побачити на рисунку 3.14 та 3.15, де показано інтерфейс документації API у вигляді Swagger UI, що використовується для тестування різних ендпоінтів веб-сервісу. У даному випадку було проведено тестування ендпоінтів авторизації та профілю. Як результат, було відображено відповідь сервера з кодом стану 200, який означає успішне виконання запиту.

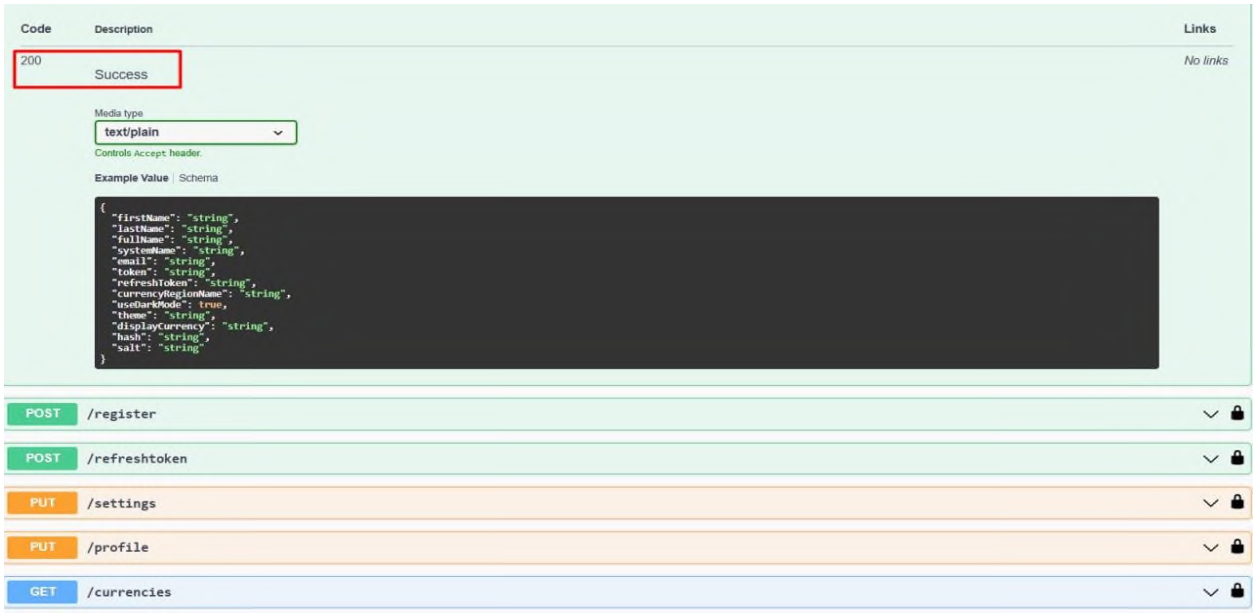


Рисунок 3.14 – Приклад успішного тестування ендпоїнту авторизації.

Рисунок 3.15 – Приклад успішного тестування ендпоїнту профілю.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто та реалізовано веб-базовану систему управління особистими завданнями. Метою цього проєкту було створення зручного та ефективного інструменту для організації робочого та особистого життя користувачів. На основі проведеного дослідження, проектування та реалізації було досягнуто наступних результатів:

В ході дослідження було проаналізовано наявні програмні рішення для управління завданнями, такі як Google Keep, Microsoft To Do і Trello. Виявлено, що кожен з цих інструментів має свої переваги і недоліки, які задовольняють різні потреби користувачів. Зважаючи на це, було прийнято рішення розробити власну веб-базовану платформу, яка поєднує кращі аспекти цих рішень, але зосереджена на простоті та зручності використання для індивідуальних користувачів. Проект передбачає створення системи, яка включає в себе функції створення особистого кабінету, додавання, редагування та видалення завдань, визначення їхньої пріоритетності, фільтрування, пошуку і перегляду статистики.

Також в роботі описано загальну структуру проєктованої системи на основі принципу чистої архітектури (Clean Architecture), яка забезпечує розподіл обов'язків між різними шарами системи для досягнення гнучкості та простоти підтримки. Було детально розглянуто компоненти цієї архітектури, такі як сутності, варіанти використання, адаптери інтерфейсу та зовнішні фреймворки. Також було обговорено використання Entity Framework для спрощення роботи з базою даних, що забезпечує ефективне управління даними в додатку.

На останок в роботі розглянуто реалізацію програмного забезпечення системи управління завданнями, зокрема, аналіз діаграм класів для domain, persistence і presentation рівнів. Було детально описано структуру та взаємозв'язки між основними класами, що забезпечують функціональність системи відповідно до принципу чистої архітектури. Окремо було розглянуто користувацький інтерфейс і проведено тестування розробленої системи, результати якого підтвердили її коректну роботу та відповідність очікуванням.

У завершенні, цей проєкт має потенціал стати корисним інструментом для кожного, хто прагне ефективно керувати своїми завданнями та часом. Забезпечуючи широкий спектр функцій, які включають у себе створення особистого кабінету, редагування та видалення завдань, фільтрування, пошук і аналіз статистики, система надає зручність та контроль користувачам у їхньому щоденному управлінні завданнями. Цей проєкт може стати не тільки практичним інструментом, але і платформою, яка сприяє розвитку особистої організації та продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lakein, Alan and Leake, Pip. How To Get Control of Your Time and Your Life. PH Wyden New York, 1973.
2. Kaufman, Carol Felker and Lane, Paul M and Lindquist, Jay D. Time Congruity in the Organization: A Proposed Quality-of-life Framework. Journal of Business and Psychology, 6(1):79106, September 1991.
3. Claessens, Brigitte JC and Van Eerde, Wendelien and Rutte, Christel G. and Roe, Robert A. Planning Behavior and Perceived Control of Time at Work. Journal of Organizational Behavior, 25(8):93750, November 2004.
4. Claessens, Brigitte JC and Van Eerde, Wendelien and Rutte, Christel G and Roe, Robert A. A Review of the Time Management Literature. Personnel review, 36(2):255276, 2007.
5. Wratcher, MA and Jones, RO. A Time Management Workshop for Adult Learners. In Journal of College Student Personnel, volume 27, pages 566567, Missouri, USA, 1986.
6. Britton, Bruce K and Tesser, Abraham. Effects of Time-management Practices on College Grades. Journal of educational psychology, 83(3):405, September 1991.
7. Macan, Therese Ho. Time Management: Test of a Process Model. Journal of applied psychology, 79(3):381391, 1994.
8. Macan, Therese Ho. Time-management Training: Effects on Time Behaviors, Attitudes, and Job Performance. The Journal of psychology, 130(3):229236, 1996.
9. Fox, Marilyn L and Dwyer, Deborah J. Stressful Job Demands and Worker Health: An Investigation of the Effects of Self-Monitoring1. Journal of Applied Social Psychology, 25(22):19731995, November 1995.
10. Zijlstra, Fred RH and Roe, Robert A and Leonora, Anna B and Krediet, Irene. Temporal Factors in Mental Work: Effects of Interrupted Activities. Journal of Occupational and Organizational Psychology, 72(2):163-185, June 1999.

11. Bond, Michael J and Feather, NT. Some Correlates of Structure and Purpose in the Use of Time. Journal of Personality and Social Psychology, 55(2):321, August 1988.

12. Hall, Brandon L and Hursch, Daniel E. An Evaluation of the Effects of a Time Management Training Program on Work Efficiency. Journal of Organizational Behavior Management, 3(4):7396, October 1982.

13. King, Abby C and Winett, Richard A and Lovett, Steven B. Enhancing Coping Behaviors in At-risk Populations: The Effects of Time-management Instruction and Social Support in Women from Dual-earner Families. Behavior Therapy, 17(1):5766, January 1986.

14. Orpen, Christopher. The Effect of Time-management Training on Employee Attitudes and Behavior: A Field Experiment. The Journal of psychology, 128(4):393396, 1994.

15. Ferriss, Tim. The 4-Hour Workweek: Escape 9-5, Live Anywhere, and Join the New Rich. Crown Publishing Group, 2007.

16. McKay, B and McKay, K. The Eisenhower Decision Matrix: How to Distinguish Between Urgent and Important Tasks and Make Real Progress in Your Life. A Man's Life, Personal Development, 2013.

17. Google Keep. URL: https://keep.google.com/ (дата звернення: 21.05.2024)

18. Microsoft To Do. URL: https://to-do.office.com/tasks/ (дата звернення: 21.05.2024)

19. Microsoft To Do: A Basic Overview & Review. URL: https://www.elegantthemes.com/blog/business/microsoft-to-do (дата звернення: 21.05.2024)

20. Trello. URL: https://trello.com/uk (дата звернення: 21.05.2024)

21. Clean Architecture: A Comprehensive Guide with C#. URL: https://medium.com/@pooja0403keshri/clean-architecture-a-comprehensive-guide-with-c-676beed5bdbb (дата звернення: 21.05.2024)

22. C# language documentation. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 21.05.2024)

23. Entity Framework documentation. URL: https://learn.microsoft.com/uk-ua/ef/ (дата звернення: 21.05.2024)

24. Діаграма принципу інтеграції Entity Framework із застосунком. URL: https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx (дата звернення: 21.05.2024)

25. SQL Server technical documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16 (дата звернення: 21.05.2024)

26. Swagger documentation. URL: https://swagger.io/docs/ (дата звернення: 21.05.2024)

27. OpenAPI Specification. URL: https://swagger.io/specification/ (дата звернення: 21.05.2024)

28. What is a RESTful API?. URL: https://aws.amazon.com/what-is/restful-api/ (дата звернення: 21.05.2024)

29. JSON Web Tokens. URL: https://jwt.io/ (дата звернення: 21.05.2024)

30. Мадараш Н. А. «ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-БАЗОВАНОЇ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ЗАВДАННЯМИ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

31. Методичні рекомендації до виконання кваліфікаційної роботи / [Комар М. П., Саченко А. О., Васильків Н. М. та ін.]. – Тернопіль: ЗУНУ, 2024. – 52 с.

Додаток А

Таблиця тестування функціоналу системи

Функція	Сценарій тестування	Очікуваний результат	Фактичний результат
Реєстрація	Введення дійсних даних для реєстрації нового користувача та натискання кнопки "Register"	Користувач успішно зареєстрований і перенаправлений на головну сторінку	Успішно (повернуто код 200)
Логінізація	Введення дійсних даних для входу в систему та натискання кнопки "Log in"	Користувач успішно входить в систему і бачить свої завдання	Успішно (повернуто код 200)
Створення нового завдання	Натискання кнопки "Create", введення інформації завдання та підтвердження дії	Нове завдання з'являється в списку користувача	Успішно (повернуто код 200)
Редагування завдання	Натискання кнопки "Edit" біля існуючого завдання, внесення змін та збереження	Зміни зберігаються і відображаються в завданні	Успішно (повернуто код 200)
Видалення	Натискання	Завдання	Успішно

завдання	кнопки "X" біля існуючого завдання та підтвердження видалення	видаляється зі списку	(повернуто код 200)
Пошук завдання за ключовими словами	Введення ключових слів у поле пошуку та натискання кнопки "Search"	Відображаються завдання, що відповідають пошуковому запиту	Успішно (повернуто код 200)
Фільтрація за статусом завдання	Вибір статусу завдання з фільтра	Відображаються завдання з вибраним статусом	Успішно (повернуто код 200)
Відображення створених завдань	Після входу в систему можна переглядати список створених завдань	Відображаються всі завдання, створені користувачем	Успішно (повернуто код 200)
Вихід із системи	Натискання кнопки "Logout"	Користувач виходить із системи і перенаправляєтьс я на сторінку входу	Успішно (повернуто код 200)

Додаток Б

Апробація отриманих результатів

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Єдиному державному реєстрі юридичних осіб, фізичних осіб-підприємців та громадських формувань: 1509433
Адреса: вул. Зодчих, буд. 40, офіс 103, м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «УКРЛОГОС Груп».
ЄДРПОУ: 44574526
IBAN: UA783052990000026003016111951
Банк: ФІЛІЯ АТ КБ «ПРИВАТБАНК» МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7172 від 31.10.2020.

ДОВІДКА

ПРО ПРИЙНЯТТЯ ТЕЗ ДО ПУБЛІКАЦІЇ

11.06.2024

Шановний(і) авторе(и):
Мадараш Наталя Анатоліївна,

Організаційний комітет з радістю повідомляє, що тези «ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-БАЗОВАНОЇ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ЗАВДАННЯМИ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

Опубліковані тези будуть доступні з 21.06.2024 за посиланням:
https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-21.06.2024

. . . . . .

Електронні сертифікати учасників конференції та подяки науковим керівникам будуть доступні з 21 червня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться до 5 липня.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 34 від 05.01.2024).

З повагою,

Директор Молодіжної наукової ліги
Голова оргкомітету конференції
ІГОР КОРЕНЮК

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-БАЗОВАНОЇ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ЗАВДАННЯМИ

Мадараш Наталя Анатоліївна
здобувач вищої освіти факультету
комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна
Науковий керівник: Кіт Іван Романович
викладач кафедри інформаційно обчислювальних систем і управління
Західноукраїнський національний університет, Україна

Управління особистими завданнями є важливим аспектом організації часу та ресурсів у сучасному житті. З появою нових технологій збільшується кількість інструментів, що допомагають автоматизувати цей процес. Основною метою роботи є розробка веб-базованої платформи для управління особистими завданнями, яка буде доступною та зручною для щоденного використання. Серед існуючих рішень більшість спрямовані на корпоративний сектор, що ускладнює їх використання для звичайних користувачів. Тому була створена система, що враховує потреби індивідуальних користувачів, забезпечуючи її простим та інтуїтивно зрозумілим інтерфейсом.

Тайм-менеджмент включає в себе процес розуміння того, що потрібно досягти, визначення пріоритетів і планування завдань. Відомі методи тайм-менеджменту включають АВС-аналіз, метод Ейзенхауера, аналіз Парето та інші. Ці методи допомагають ефективно використовувати час і досягати цілей.

Проведений аналіз показав, що існуючі інструменти для управління завданнями не завжди задовольняють потреби звичайних користувачів. Тому було прийнято рішення розробити власну систему, яка буде мати простий та інтуїтивний інтерфейс. Основні функції системи включають створення, редагування, видалення завдань, встановлення їх статусу та пріоритетності, а також можливість пошуку та фільтрації завдань.

Система розроблена на основі принципу чистої архітектури (Clean Architecture), що забезпечує розподіл обов'язків між різними шарами системи. Використано Entity Framework для спрощення роботи з базою даних, що забезпечує ефективне управління даними.

Розробка веб-базованої системи управління особистими завданнями включала створення діаграм класів для різних рівнів системи. Користувацький інтерфейс включає основні функції для створення та управління завданнями. Проведене тестування показало, що система працює коректно та відповідає очікуванням користувачів.

Проект має потенціал стати корисним інструментом для ефективного управління завданнями та часом, забезпечуючи зручність та контроль користувачам у їхньому щоденному житті.

Список використаних джерел

1. Lakein Alan and Leake Pip. How To Get Control of Your Time and Your Life. PH Wyden, New York, 1973.
2. Kaufman Carol Felker, Lane Paul M., Lindquist Jay D. Time Congruity in the Organization: A Proposed Quality-of-life Framework. Journal of Business and Psychology, 1991.
3. Claessens Brigitte JC, Van Eerde Wendelien, Rutte Christel G., Roe Robert A. Planning Behavior and Perceived Control of Time at Work. Journal of Organizational Behavior, 2004.
4. Fox Marilyn L., Dwyer Deborah J. Stressful Job Demands and Worker Health: An Investigation of the Effects of Self-Monitoring1. Journal of Applied Social Psychology, 1995.
5. McKay B., McKay K. The Eisenhower Decision Matrix: How to Distinguish Between Urgent and Important Tasks and Make Real Progress in Your Life. A Man's Life, Personal Development, 2013.