МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність Комп'ютерні науки

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Овчаренка Кирила Руслановича
(прізвище, ім'я, по батькові здобувача)

на тему Розробка веб-сервісу персонального тайм-менеджменту з використанням фреймворку Vue.js
(повна назва теми)

за матеріалами _____
(повна назва бази дослідження)

науковий керівник _____ _____ Шокотько Л.М.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 09.06.2025р. №12

Завідувач кафедри _____
(підпис)

к.е.н., доцент Радько В.М.
Наук. ступінь, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність _Комп'ютерні науки___

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"07" квітня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА ЗДОБУВАЧУ

Овчаренку Кирилу Руслановичу

1. Тема роботи «Розробка веб-сервісу персонального тайм-менеджменту з використанням фреймворку Vue.js»

науковий керівник роботи Шокотько Людмила Миколаївна
затверджені наказом вищого навчального закладу від «04» квітня 2025 р. № 224-ст

2. Строк подання здобувачем роботи 31.05.2025р.

3. Зміст кваліфікаційної роботи бакалавра, об'єкт, предмет та мета дослідження:

Розділ 1 Дослідження предметної області проєкту. Постановка задачі.

Розділ 2 Моделювання процесу управління задачами у веб-сервісі персонального тайм-менеджменту.

Розділ 3 Програмна реалізація веб-сервісу персонального тайм-менеджменту.

Об'єкт дослідження – інформаційні технології управління процесами персонального тайм-менеджменту.
Предмет дослідження – процеси, методи та інструменти розробки веб-сервісу персонального тайм-менеджменту.
Мета кваліфікаційної роботи бакалавра – проектування архітектури та програмна реалізація веб-сервісу персонального тайм-менеджменту

4. Дата видачі завдання 04.04.2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 28.04.2025р.	25.04.2025
2	Підготовка розділу 2	до 16.05.2025р.	15.05.2025
3	Підготовка розділу 3	до 30.05.2025р.	29.05.2025
4	Реєстрація завершеної дипломної роботи	до 31.05.2025р.	30.05.2025
5	Отримання відгуку від наукового керівника	03-04.06.2025р.	04.06.2025
6	Отримання зовнішньої рецензії	05-06.06.2025р.	06.06.2025
7	Перевірка кваліфікаційної роботи на плагіат	02-09.06.2025р.	04.06.2025
8	Попередній захист кваліфікаційної роботи на кафедрі	03.06.2025р.	03.06.2025
9	Допуск кафедрою кваліфікаційної роботи до захисту	09.06.2025р.	09.06.2025
10	Підготовка студента до захисту в ЕК	до 17.06.2025р.	17.06.2025

Завдання підготував науковий керівник _____ Шокотько Л.М.

Завдання одержав здобувач _____ Овчаренко К.Р.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 60 стор., 1 таблиця, 42 рисунка, 3 додатка, 18 джерел.

Об’єкт дослідження: інформаційні технології управління процесами персонального тайм-менеджменту.

Метою роботи є розробка веб-сервісу персонального тайм-менеджменту.

Веб-сервіс персонального тайм-менеджменту надає великий набір сучасних та ефективних інструментів для управління завданнями, що поєднує простоту інтерфейсу, багатофункціональність та гнучкість у використанні. Сервіс стане вагомим внеском у вдосконалення підходів організації роботи та підвищить ефективність користувачів у їхній повсякденній діяльності.

Було проведено аналіз інструментальних засобів щодо автоматизованого проєктування та розробки програмних продуктів.

Веб-сервіс реалізований з використанням HTML, CSS, JavaScript (Vue.js). Забезпечено адаптивність інтерфейсу та масштабування. В ході проєктування були створенні UML-діаграми: варіантів використання, классів.

Сфери застосування: IT-сфера, освіта, бізнес, персональне використання.

Використання сервісу позитивно впливає на ефективність діяльності за рахунок підвищення продуктивності праці, зменшення часу на планування та координацію завдань.

Ключові слова: автоматизація, адаптація, веб-сервіс, тайм-менеджмент, LocalStorage, UML, Vue.js

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CLI – Command Line Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

ER – Entity-Relationship

HTML – HyperText Markup Language

id – identifier

ISO/IEC 25010 – International Organization for Standardization / International Electrotechnical Commission 25010

SPA – Single Page Application

UML – Unified Modeling Language

W3C – World Wide Web Consortium

WCAG 2.1 – Web Content Accessibility Guidelines 2.1

ВСТУП

У сучасному світі, в якому темп життя постійно зростає, перед людьми постала проблема з ефективним управлінням часом – організацією завдань. Це важливо як задля досягнення особистих цілей успіху, так і для професійної діяльності. У більшості випадків це може стосуватися студентів, працівників великих компаній, фрілансерів, а також людей з когнітивними порушеннями та всіх інших, кому необхідний високий рівень самоорганізації для реалізації своєї діяльності. Коли людина не має контролю над своїм часом, це може призвести до негативних обставин, включаючи:

- зниження продуктивності;
- втрату мотивації та мети;
- проблеми зі сном;
- хронічний стрес.

Цифрові технології допомагають вирішити цю проблему, надаючи корисні інструменти, що дозволяють організовувати повсякденні завдання, забезпечуючи контроль за їх виконанням. Від простих нотаток у телефоні до великих функціональних систем – вони стали невід'ємною частиною сучасного способу життя. Одним з найбільш затребуваних рішень у цьому напрямку є веб-сервіс персонального тайм-менеджменту, або так званий Task Manager – що дозволяє не тільки систематизувати завдання, але й дотримуватись дедлайнів, планувати день і уникати перевантаження, що особливо важливо в умовах багатозадачності та високого темпу життя.

Веб-сервіси персонального тайм-менеджменту стали затребуваними завдяки своїй доступності, зручності та універсальності у використанні. Вони надають користувачам наступне:

- можливість створювати та керувати завданнями через інтуїтивно зрозумілий інтерфейс користувача;
- встановлювати пріоритети та визначати терміни виконання;

– можливість інтегруватися з іншими сервісами.

Наприклад, завдяки інтеграції з календарями або програмами для обміну повідомлень, такі сервіси забезпечують синхронізацію роботи між особистими та командними цілями. Незважаючи на широкий вибір існуючих рішень, серед яких Microsoft To Do, Todoist, TickTick, Google Tasks, Any.Do, залишається ніша для створення нового програмного продукту, який міг би зібрати всі найкращі функції цих сервісів, доповнивши їх своїми унікальними можливостями для легкості користувачів.

Актуальність роботи полягає у розробці веб-сервісу для управління завданнями, що відповідатиме сучасним вимогам до зручності, функціональності та адаптивності. У нинішньому світі, в короткий проміжок часу з'являються нові вимоги та виклики, що сприяють користувача шукати прості, але потужні рішення, що підходять під реалії.

Метою роботи є розробка веб-сервісу персонального тайм-менеджменту − програмного продукту, здатного задовольнити потреби великого оточення користувачів – від студентів та фрілансерів до людей із когнітивними порушеннями здоров'я.

Для досягнення цієї мети вирішені наступні задачі:

– проаналізовано сучасні підходи до розробки веб-сервісів;
– проаналізовано існуючі на ринку програмного забезпечення веб-сервіси персонального тайм-менеджменту;
– визначено ключові функціональні вимоги до веб-сервісу персонального тайм-менеджменту;
– виконано моделювання логічної та фізичної моделі з використанням UML-діаграм;
– реалізовано програмний продукт з можливістю масштабування та подальшого вдосконалення;
– проведено комплексне тестування веб-сервісу, з оглядом адаптації під різні пристрої.

Об’єкт дослідження: інформаційні технології управління процесами персонального тайм-менеджменту.

Предметом дослідження є процеси, методи та інструменти розробки веб-сервісу персонального тайм-менеджменту з акцентом на інтеграцію сучасних веб-технологій (HTML, CSS, JavaScript, Vue.js), адаптивного дизайну та функціональності, спрямованої на оптимізацію особистої продуктивності.

Реалізація програми включає впровадження технологій HTML5, CSS3, JavaScript, що забезпечує високу якість програмного продукту, стабільність його роботи та інтерактивність. Особлива увага приділена створенню адаптивного та інтуїтивного інтерфейсу, що дозволяє користувачам легко взаємодіяти на різних пристроях.

Таким чином, дана кваліфікаційна робота покликана розробити сучасний та ефективний інструмент для управління завданнями, що поєднує простоту інтерфейсу, багатофункціональність та гнучкість у використанні. Сервіс стане вагомим внеском у вдосконалення підходів організації роботи та підвищить ефективність користувачів у їхній повсякденній діяльності.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЄКТУ. ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області дослідження

Стрімкий розвиток світу, у якому цифровізація стала важливою частиною в різних сферах життя, має бути забезпеченим інструментами, які допоможуть людям керувати своїм часом. Веб-сервіс персонального тайм-менеджменту – один з таких інструментів, який є незамінною частиною в упорядкуванні часу людини.

Веб-сервіс персонального тайм-менеджменту зобов'язаний відповідати ключовим запитам користувачів: зручність відстеження завдань та прогресу їх виконання, управління часом. Ці запити передбачають такі функції, як:

- створення завдань із детальним описом;
- додавання дедлайну до завдання (за потребою);
- редагування завдань, при помилці або зміні завдання;
- можливість відмічати виконані завдання;
- відстеження виконання завдань на поточний час.

Також він повинен підтримуватись як на комп'ютері, так і на мобільному пристрої, при цьому бути оптимізованим під різні розміри екранів.

Реалізація веб-сервісу базується на сучасному підході до адаптивності, інтерактивності та простоті використання. Поняття веб-сервісу почало формуватися задовго до свого теперішнього вигляду. Все почалося в епосі Web 1.0, також її охарактеризували як «тільки для читання», вона була з початку до середини 1990-х років, розроблена Тімом Бернерсом-Лі [1]. Це перша епоха Інтернету, яка дала можливість створювати сайти, які мали обмежені можливості взаємодії, завдяки технології HTML.

HTML – це мова програмування, яку використовують для розмітки сторінки під час перегляду у браузері.

Взаємодія з сайтом включала: перегляд веб-сторінок, заповнення простих форм, а також переходи за гіперпосиланнями. Якщо сайт складався зі статичної веб-сторінки, то взаємодія була лише одна – перегляд веб-сторінки.

Наступною епохою стала Web 2.0, її також називають «соціальний веб», що проходила в період з середини 1990-х до середини 2010-х років. У цій епосі люди створювали технології, які використовуються для створення сайтів сучасного формату. Для прикладу можна навести дві технології, що використовуються в Web 2.0: CSS та JavaScript.

CSS – це мова яка описує стилі, вона використовує каскадні таблиці стилів для відображення елементів веб-сторінки написаних мовою HTML. За допомогою цієї мови можна: міняти розташування елементів на веб-сторінці; змінювати текст, змінюючи колір, шрифт, кегель; додавати анімацію та багато чого іншого.

JavaScript – це високорівнева мова програмування, яка з'явилася в 1995 році і дала початок для створення функціональних та «живих» сайтів [2]. З її допомогою сайти стали орієнтованими на взаємодію з користувачами. Розглянемо чотири інструменти, які можна реалізувати за допомогою JavaScript:

- інтерактивність – дозволяє елементам реагувати на відповідні дії користувача;
- валідація форм – дає можливість перевіряти форму на правильність заповнення перед відправкою;
- динамічне оновлення – дозволяє без перезавантаження сайту, оновити необхідну частину;
- сучасна навігація – надає можливість не робити повне перезавантаження веб-сторінки, що дозволяє користувачам плавно пересуватися сайтом.

Останньою або, можна сказати, нинішньою епохою, запровадженою Тімом Бернерсом-Лі, є Web 3.0. Вона відрізняється від епохи Web 2.0 тим, що в ній створили технологію блокчейну, а також прагнуть зробити децентралізацію з метою зменшення залежності від великих технологій.

Також необхідно передбачити нормативну базу, яка необхідна для контролю, як загальних підходів якості програмного продукту, так і специфічних аспектів роботи з веб-сервісом. Загальні підходи до розробки веб-сервісу вимагають передбачення оцінок ISO/IEC 25010 – серії міжнародних стандартів для оцінки програмного продукту [3]. Серія стандартів включає вісім характеристик, чотири з яких це: рівень продуктивності, надійність, зручності використання, сумісність.

До специфічних аспектів роботи з веб-сервісом можна віднести важливість дотримання рекомендацій W3C щодо доступності WCAG 2.1, що дозволяє зробити веб-сервіс зручним для користувачів з обмеженими можливостями, таким як: людям з низьким рівнем зору або з когнітивними порушеннями [4]. WCAG 2.1 включає такі рекомендації, як:

- адаптивність – функціонал веб-сервісу має бути доступний як у горизонтальній, так і вертикальній орієнтації;
- помітність – контент має бути чітко відокремлений від заднього фону, для легкості сприйняття.

1.2 Огляд доступних програмних рішень

Сьогодні є безліч інструментів для тайм-менеджменту, які допомагають людям організовувати свою роботу і керувати своїм часом. Найбільш популярними є Microsoft To Do, TickTick, Google Tasks, Todoist та Any.Do. Давайте розглянемо кожен із них.

Microsoft To Do (рис. 1.1) – це сучасний, інтуїтивно зрозумілий веб-сервіс, розроблений корпорацією Microsoft. Завдяки інтуїтивно зрозумілому інтерфейсу, навіть користувачі, які не мають досвіду використання схожих інструментів, зможуть швидко освоїти всі його функції. Веб-сервіс дозволяє:

- створювати та редагувати завдання;
- додавати нагадування;
- встановлювати дедлайни;

– виділяти важливі завдання;
– групувати та сортувати завдання;
– відстежувати завдання на поточний день.

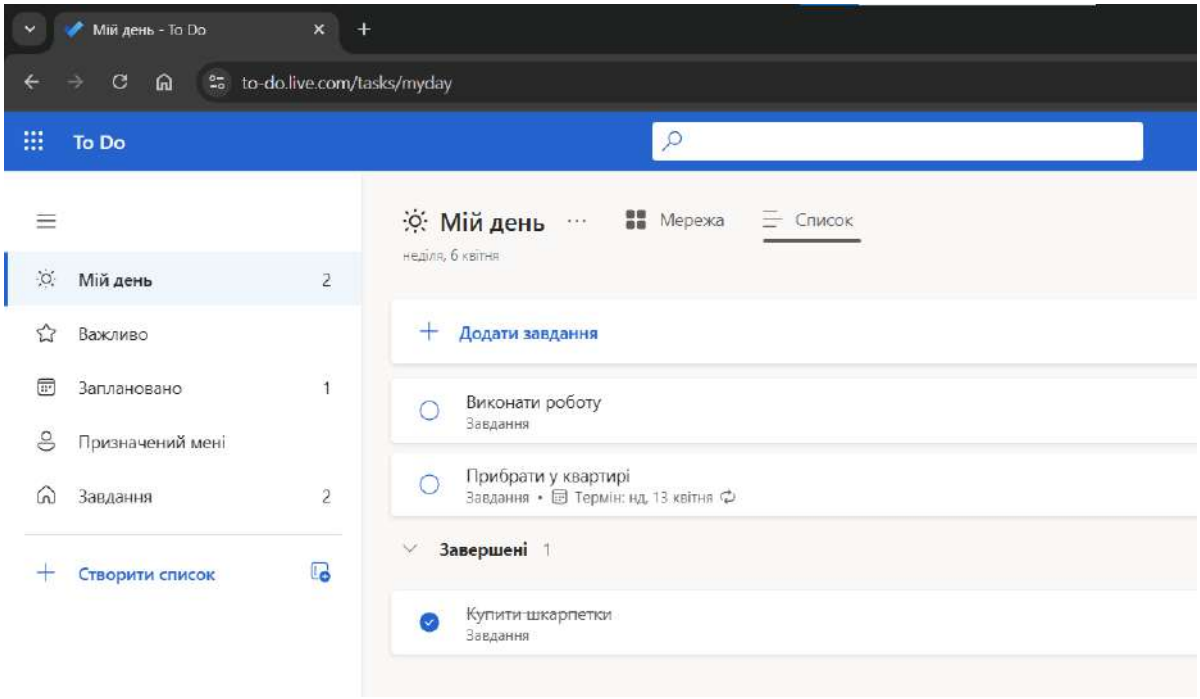

Рис. 1.1. Інтерфейс Microsoft To Do

Примітка. Джерело: розроблено із використанням [5]

До важливої відмінності Microsoft To Do від схожих веб-сервісів можна віднести його інтеграцію з популярними програмними продуктами екосистеми Microsoft, яка включає такі програмні продукти, як Outlook, Teams, Office. Це дозволяє користувачам ефективно організовувати як робочі так і особисті завдання в одному просторі, при цьому давши можливість синхронізації даних між пристроями. Так, наприклад, завдання, створене в програмі Outlook, автоматично з'являється в Microsoft To Do, якщо обидві програми мають однин обліковий запис. Так само повідомлення з дедлайнами з’являються в обох програмах, що дозволяє вчасно реагувати на них.

Проте, якщо не брати до уваги його вагому перевагу з інтеграцією в екосистему Microsoft, додаток має певні обмеження. Одним з таких обмежень є відсутність розширених інструментів для аналізу виконання завдань або

автоматизації процесів – це може стати вагомим обмеженням для людей, яким необхідний потужний інструмент для планування часу.

Щодо Microsoft To Do можна зробити висновок, що цей простий інструмент для планування завдань є гарним рішенням для користувачів, яким потрібен простий інтерфейс, швидкий функціонал та глибока інтеграція з програмними продуктами Microsoft.

TickTick (рис. 1.2) – це багатогранний веб-сервіс, який допомагає організовувати завдання, підвищуючи продуктивність користувачів. Функціонал даного веб-сервісу допомагає не тільки впорядкувати завдання, але й відслідковувати, як ви витрачаєте свій час на виконання завдань. Він буде гарним варіантом для тих, хто має амбіції в покращенні своїх навичок у тайм-менеджменті, зробивши день більш планованим та продуктивним завдяки його функціоналу.

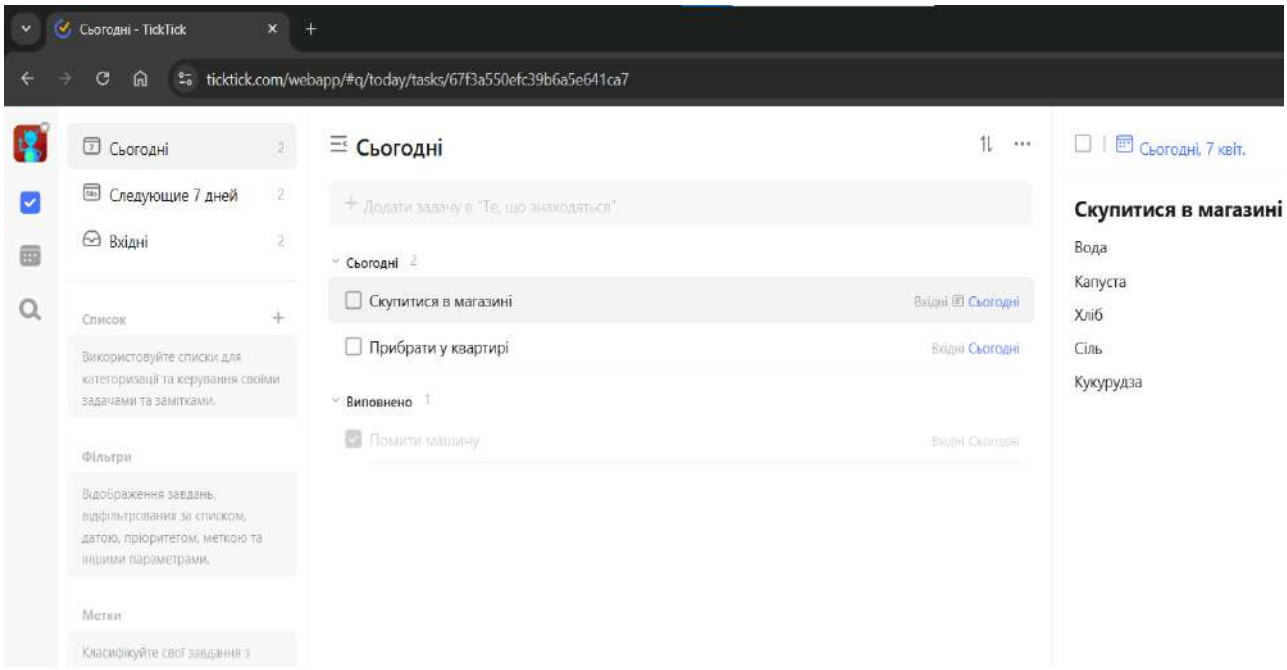

Рис. 1.2. Інтерфейс TickTick

Примітка. Джерело: розроблено із використанням [6]

Однією з функцій, що його виділяє від інших, є вбудований таймер «Pomodoro» – це корисний інструмент, який мотивує користувачів працювати за

чітким графіком. Він поділяє час на короткі проміжки в яких є час для перерви, що сприяє концентрації і допомагає відійти від можливості вигоряння. Також TickTick дає можливість відстежувати витрачений час на виконання завдання, що дозволяє аналізувати з якою продуктивністю було виконано завдання, а також шукати шляхи для покращення.

Проаналізувавши веб-сервіс TickTick, можна сказати, що він підходить як студентам, викладачам, так і офісним працівникам, фрілансерам і тим людям, яким необхідний інструмент для тайм-менеджменту. У цьому веб-сервісі багато інструментів, серед яких є такі, як персоналізовані нагадування, інтеграція з календарем та можливість роботи в команді.

Google Tasks (рис. 1.3) – це веб-сервіс з низьким рівнем інтерфейсу користувача, який має хороший функціонал для особистого користування, але для професійного використання він може надати лише можливість інтеграції з Google сервісами.

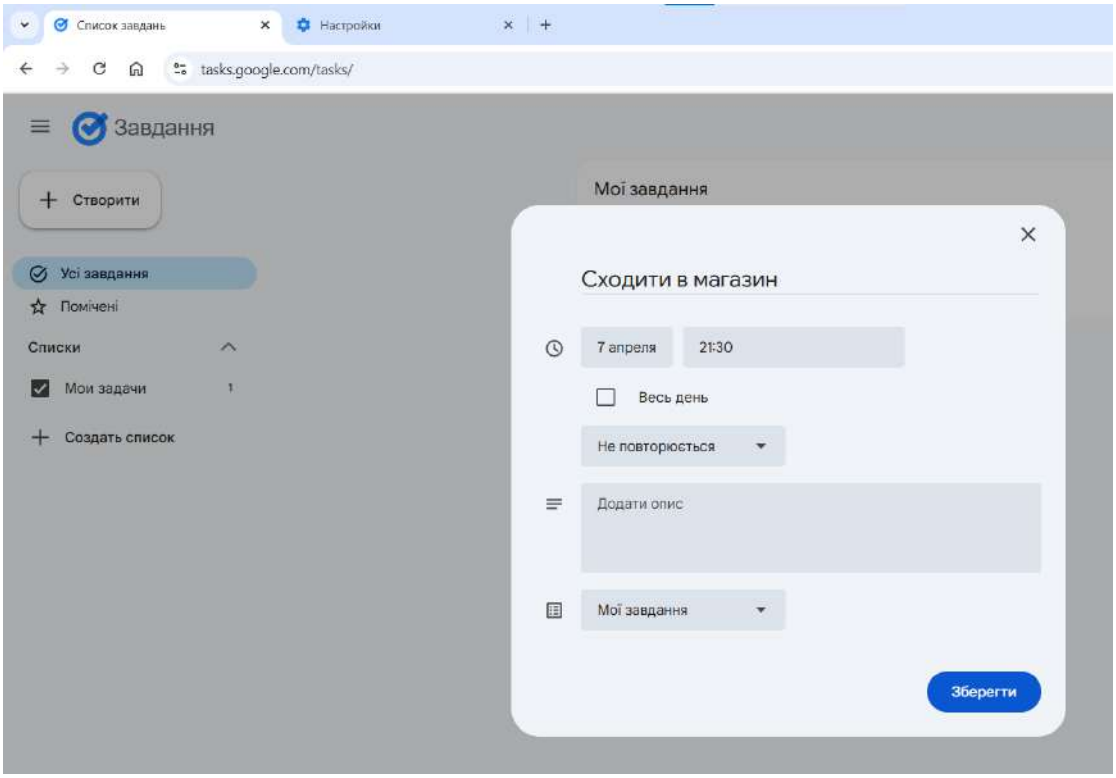

Рис. 1.3. Інтерфейс Google Tasks

Примітка. Джерело: розроблено із використанням [7]

Глибока інтеграція Google Task з екосистемою Google – це головна його перевага. Вона дозволяє швидко створювати й управляти завданням без необхідності перемикатися між сервісами. Простий, лаконічний, інтуїтивно зрозумілий інтерфейс який позбавлений зайвих функцій, робить його вдалим вибором для тих, кому потрібен мінімалістичний дизайн та швидкість у роботі. Крім того, Google Tasks безкоштовний для користувачів з обліковим записом Google.

Втім, треба зазначити, що в цьому веб-сервісі є й обмеження. Він має маленький набір функцій, яких буде недостатньо для складних проєктів чи розширеного планування. Також, відсутня в додатку можливість співпраці в команді, робить неможливою роботу в групі або управління великим проєктом.

Todoist (рис. 1.4) – це багатофункціональний веб-сервіс, що дозволяє легко та зручно забезпечити користувача плануванням завдань, зробивши особисті та робочі процеси більш оптимізованими, структурувавши їх. Для робочих процесів він зручний тим, що організовує процеси, забезпечуючи завданням, проєктам та термінам для їх виконання гнучке налаштування. Підтримка тегів, глибокої аналітики та пріоритетів робить веб-сервіс хорошим вибором для відстеження ефективності та продуктивності під час виконання завдань.

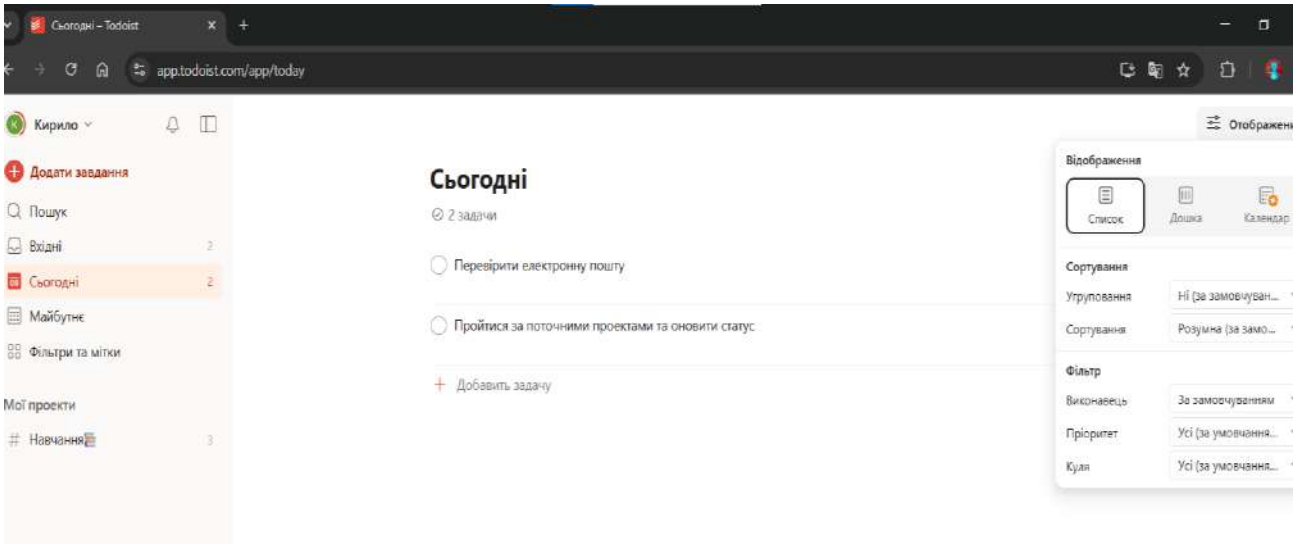

Рис. 1.4. Інтерфейс Todoist

Примітка. Джерело: розроблено із використанням [8]

Переваги Todoist включають:

- можливість створення проєктів із завданнями;
- фільтрацію завдань за певними тегами, що дозволяє швидко знаходити необхідні завдання;
- детальну аналітику та статистику виконання завдань, дозволяючи користувачам відстежувати свою продуктивність та моніторити довгострокові проєкти;
- інтеграцію з Google Календарем, для автоматичної синхронізації завдань між веб-сервісами [9].

До недоліків Todoist відноситься те, що безкоштовна версія занадто обрізана у функціоналі, до таких функцій відноситься нагадування та відображення списку завдань у вигляді календаря, що може бути обмеженням для тих користувачів, яким необхідний більший функціонал. Плюс до всього ця преміум-підписка, що дає доступ до всіх функцій, має вартість вищу, ніж в альтернативних веб-сервісах, що може відвернути користувача з обмеженими засобами від придбання цього варіанту.

Any.Do (рис. 1.5) – це яскравий та універсальний інструмент для тайм-менеджменту, що використовується при виконанні повсякденних завдань як у особистому житті, так і під час роботи. Цей веб-сервіс може надати потужний інструментарій, серед якого є інструмент – щоденний планувальник для розподілу завдань по днях за допомогою пріоритетів. Наприклад, з його допомогою можна вивести список завдань на день, в якому перші завдання матимуть найвищий рівень пріоритету в порівнянні з рештою. Також є можливість інтегруватися з популярними календарями та голосовими помічниками, що дозволяє користувачеві додавати завдання та отримувати нагадування для спрощення роботи з веб-сервісом.

Яскравий і простий інтерфейс Any.Do дозволяє з легкістю розібратися з усіма його інструментами. Всі основні функції веб-сервісу зібрані в інтуїтивно зрозумілих місцях, що робить його простим для нових користувачів. Це робить

Any.Do хорошим інструментом для людей, які хочуть збільшити ефективність свого часу.

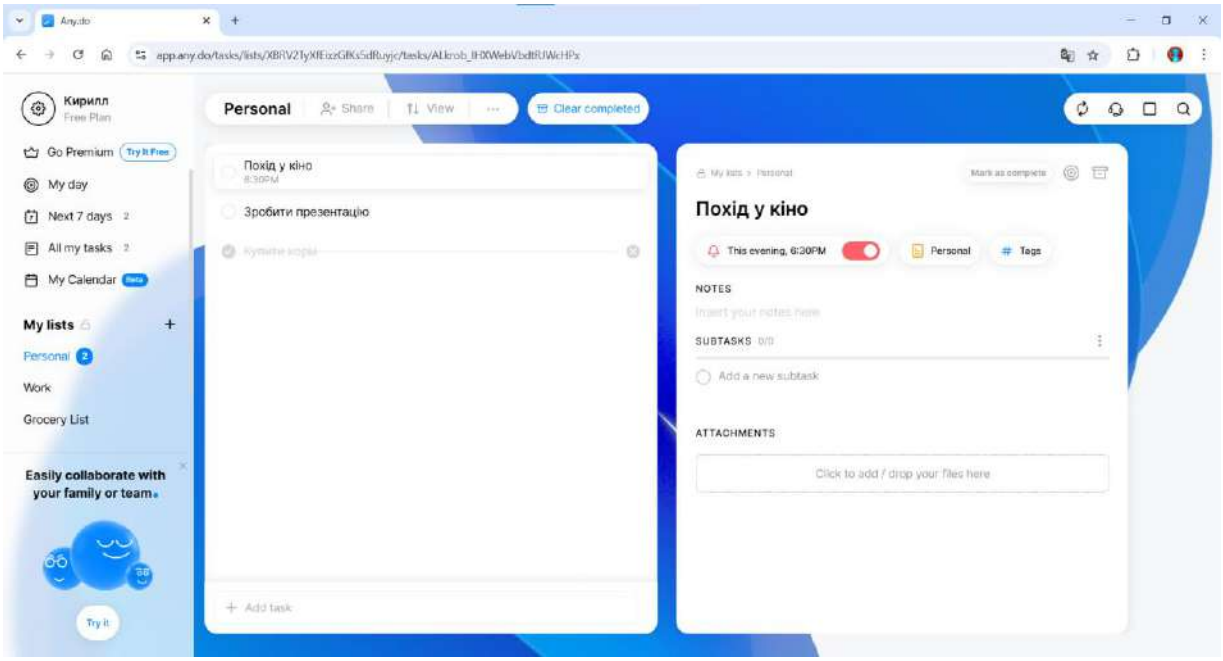

Рис. 1.5. Інтерфейс Any.Do

Примітка. Джерело: розроблено із використанням [10]

Варто зазначити, що для отримання основних функцій необхідна преміум-підписка, тому що безкоштовна версія має обмежений функціонал, якого може не вистачити для продуктивного використання. Це може обмежити користувачів при виборі, які не мають фінансової стабільності.

Аналіз існуючих програмних продуктів показав широкий вибір інструментів для тайм-менеджменту, показавши, що кожен з них має свої особливості, які можна врахувати при розробці власного веб-сервісу. Наприклад, Microsoft To Do відрізняється інтеграцією з програмними продуктами Microsoft, TickTick пропонує функцію «Pomodoro» та можливість відстеження часу, Google Tasks є простим та зручним для базового планування, Todoist забезпечує розширені можливості для аналізу продуктивності, а Any.Do пропонує функцію щоденного планувальника.

Цей огляд дав зрозуміти, що існує велика кількість веб-сервісів для персонального тайм-менеджменту, але є ніша спектра потреб, яка залишається нереалізованою для деяких користувачів. Наприклад, для людей з вадами зору або людей, які переживають за конфіденційність своїх даних.

1.3 Призначення і ключові сценарії застосування програмного продукту

Одним із важливих аспектів ефективного формування та планування роботи є управління часом. Будь то особисті цілі, академічні проєкти чи професійні завдання, практичний та ефективний інструмент персонального тайм-менеджменту може значно підвищити продуктивність та рівень якості життя. У зв'язку з цим розробка веб-сервісу для персонального тайм-менеджменту є актуальним рішенням, що допомагає ефективно використовувати такі ресурси, як час, зусилля та увага.

Основна ідея такого веб-додатка полягає у створенні універсального середовища, яке дозволяє користувачам ефективно організовувати свою діяльність. Користувач програмного продукту повинен мати можливість створювати, редагувати, відстежувати та аналізувати виконання завдання, а також визначати за потреби дедлайни.

Важливо, щоб веб-сервіс персонального тайм-менеджменту відповідав потребам широкого кола користувачів:

- студентів, яким потрібно організовувати навчання;
- працівникам компаній для управління робочими процесами;
- фрілансерів, що працюють з клієнтськими замовленнями;
- іншим особам, які прагнуть структурувати свій час.

Крім того, важливо створити простий і інтуїтивно зрозумілий інтерфейс, щоб зробить веб-сервіс доступним широкому колу користувачів. Завдяки інтуїтивній структурі, користувачі можуть швидко зрозуміти функціональність веб-сервісу та адаптувати його під свої індивідуальні потреби.

Розробка такого веб-додатку включає забезпечення вибору для користувача гаджету при взаємодії з веб-сервісом. Це сприяє підвищенню гнучкості в організації життєвого процесу.

Застосування подібного інструменту відкриває широкі можливості для підвищення продуктивності. По-перше, він дозволяє мінімізувати хаос у плануванні завдяки чіткому структуруванню завдань. По-друге, допомагає уникати пропуски важливих дедлайнів через нагадування. По-третє, забезпечує зручність в аналізі виконаної роботи, що сприяє кращому розумінню прогресу та плануванню майбутніх дій.

Такий інструмент допомагає уникати перевантаження та пропущених дедлайнів і забезпечує баланс між роботою та відпочинком. Сприяє підвищенню продуктивності, дисципліни та організованості, що є ключовими факторами успіху у сучасному світі. В умовах зростаючої конкуренції та вимог до якості виконання завдань, наявність ефективного інструмента для їх управління може стати вирішальним фактором у досягненні поставлених цілей.

Висновки до розділу 1

У першому розділі було оглянуто основи розробки веб-сервісу персонального тайм-менеджменту, що допомагає при організації та реалізації часу. Було оглянуто такі речі, як:

- історія Web, з технологіями створеними в кожній епосі;
- що таке рекомендації WCAG 2.1;
- що таке серія стандартів ISO/IEC 25010;
- п'ять веб-сервісів персонального тайм-менеджменту – Microsoft To Do, TickTick, Google Tasks, Todoist та Any.Do.

Метою веб-сервісу персонального тайм-менеджменту є надання користувачеві інструментів для планування, аналізу та відстеження часу. Завдяки аналізу стало зрозуміло, що такий веб-сервіс повинен обов'язково надавати такий функціонал, як:

- створення та редагування завдань з урахуванням дедлайнів;
- відстеження виконаних завдань за допомогою діаграм;
- відстеження поточного плану на день;
- легкий пошук потрібних завдань.

Крім того, стало зрозуміло, що веб-сервіс повинен давати легкий та інтуїтивно зрозумілий інтерфейс для того, щоб залучити широку аудиторію користувачів: від студентів, фрілансерів до людей з когнітивними порушеннями та користувачів, котрим потрібна конфіденційність. При цьому дається можливість вибору більш зручного для користувача пристрою.

Як висновок можна сказати, що перший розділ закладає фундамент для створення функціонального, простого та зручного веб-сервісу для персонального тайм-менеджменту, який піднімає рівень продуктивності та організованості часу користувача. Він базуватиметься на сучасних технологіях, таких як HTML5, CSS3, JavaScript, з використанням фреймворку Vue.js; а також на сучасному підході до реалізації сервісу, кращої адаптації та стабільності роботи, що дасть користувачеві високий рівень якості використання.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ПРОЦЕСУ УПРАВЛІННЯ ЗАДАЧАМИ У ВЕБ-СЕРВІСІ ПЕРСОНАЛЬНОГО ТАЙМ-МЕНЕДЖМЕНТУ

2.1 Аналіз і обґрунтування вибору фреймворку для створення веб-сервісу персонального тайм-менеджменту

В даний час існує велика кількість фреймворків, які з HTML5, CSS3 та JavaScript дозволяють створювати веб-сервіси, перевагою яких є швидкість роботи, стабільність функціонування та естетичний вигляд інтерфейсу. До них належать такі технології як React, Angular, Vue.js, у таблиці 2.1 надано порівняльний аналіз даних фреймворків [11].

Таблиця 2.1

Аналіз фреймворків React, Vue.js, Angular

Фреймворк	Швидкість роботи	Простота освоєння	Популярність	Інтеграція
React	Висока	Легко	Висока	Висока
Angular	Середня	Складно	Середня	Висока
Vue.js	Висока	Легко	Висока	Висока

Примітка. Джерело: розроблено автором

Розглянемо кожен фреймворк докладніше.

React (рис. 2.1) – це фреймворк, що передбачається як бібліотека JavaScript [12]. Перша його версія була опублікована у 2013 році, командою Facebook. До позитивних якостей React можна віднести:

- швидке оновлення віртуального DOM. Він дає можливість оновлювати лише необхідні частини, при зміні даних, замість оновлення всього DOM;
- компонентність. Веб-сервіс створюється з окремих модулів, які можна використовувати повторно за потреби;

– популярність та інтеграція. Він має як велику аудиторію розробників, так і велику кількість бібліотек. Це дозволяє користувачу створювати та дороблювати необхідний функціонал.

Рис. 2.1. Логотип React

Angular (рис. 2.2) – розроблена Google технологія, що дозволяє створювати SPA веб-сервіси, без необхідності перезавантаження веб-сторінки [13]. Фреймворк використовує TypeScript – мову програмування, яка є інструментом для розширення можливостей JavaScript.

Рис. 2.2. Логотип Angular

До плюсів Angular відносяться компонентність та популярність серед розробників, але при цьому він має суттєві мінуси, такі як:

– складність для новачків у галузі розробки веб-сервісів;
– застосування. Цей фреймворк занадто потужний, тому буде недоречним для малих і непримітних веб-сервісів;
– швидкість роботи. Веб-сервіс на Angular, здатний володіти великим обсягом bundle – це безліч пов'язаних файлів, які необхідні для роботи програми.

Vue.js (рис. 2.3) – це фреймворк, створений для розробки інтерфейсу користувача, з інтуїтивно зрозумілим синтаксисом, навіть для новачків у галузі веб-розробки [14]. Архітектура фреймворку – компонентно-орієнтована, що дає можливість створювати веб-сервіси компонентами, серед них є Vue Instance. Він є одним із головних, оскільки містить у собі всі відомості та методи застосовувані у всьому веб-сервісі.

Рис. 2.3. Логотип Vue.js

Фреймворк Vue.js добре підходить для багатьох сценаріїв, але краще застосовувати при:

- створенні зразка веб-сервісу, оскільки він дозволяє легко та швидко реалізувати проєкт;
- створення маленьких проєктів, якщо ви не знайомі з цим фреймворком, часу піде небагато, оскільки він не потребує великого вивчення синтаксису.

Розглянувши таблицю 2.1 з аналізом фреймворків, а також з їх детальним оглядом, можна зробити висновок, що фреймворк Vue.js підходить для реалізації веб-сервісу персонального тайм-менеджменту, оскільки він компонентно-орієнтований, що дозволяє легко створювати та редагувати частини веб-сервісу завдяки своїй зручності та швидкості. Як основа проєкту стане Vue CLI – це зручний інструмент для налаштування, ініціалізації та управління проєктом на основі фреймворку Vue.js.

2.2 Розробка логічної моделі процесу управління задачами у веб-сервісі персонального тайм-менеджменту

Одним із важливих етапів при створенні веб-сервісу персонального тайм-менеджменту є розробка логічної моделі, яка допомагає створити ефективний, лаконічний та зрозумілий веб-сервіс. Логічна модель є набором UML-діаграм, які наочно показують систематизацію сутностей, демонструючи зв'язки між ними. На цьому етапі формуються функціональні та нефункціональні вимоги до системи, які визначають її майбутню архітектуру. Логічна модель дозволяє описати логіку роботи системи, механізми взаємодії між її частинами та формує основу для розробки користувацьких сценаріїв. Також дає можливість виявляти потенційні проблеми на ранніх етапах, що сприяє зменшенню витрат на тестування та усунення помилок у подальших етапах.

Створення логічної моделі забезпечує:

- зрозумілість та прозорість коду для розробників, дозволяючи їм краще зрозуміти будову проєкту;
- легкість у внесенні змін та доповнень через наочну демонстрацію логіки, що дозволяє передбачити можливі помилки в майбутньому;
- формування бази для інтеграції нових функцій та супроводу існуючого функціоналу.

Першим кроком при розробці логічної моделі процесу управління задачами у веб-сервісі персонального тайм-менеджменту є створення UML-діаграми прецедентів. Вона дозволяє проілюструвати взаємодії користувача з функціональними модулями програмного продукту [15]. У веб-сервісі персонального тайм-менеджменту основними прецедентами, які були виявленні при аналізі ключових сценаріїв застосування та огляді доступних програмних рішень, є:

- управління задачами – створення задач, редагування задач, видалення та перегляд задач;
- пошук задач – знаходження необхідної задачі;

– зміна дизайну – редагування дизайну веб-сервісу;
– відстеження виконання задач – зміна статусу задач та перегляд статистики виконання.

Діаграма прецедентів розроблена та продемонстрована на рисунку 2.4.

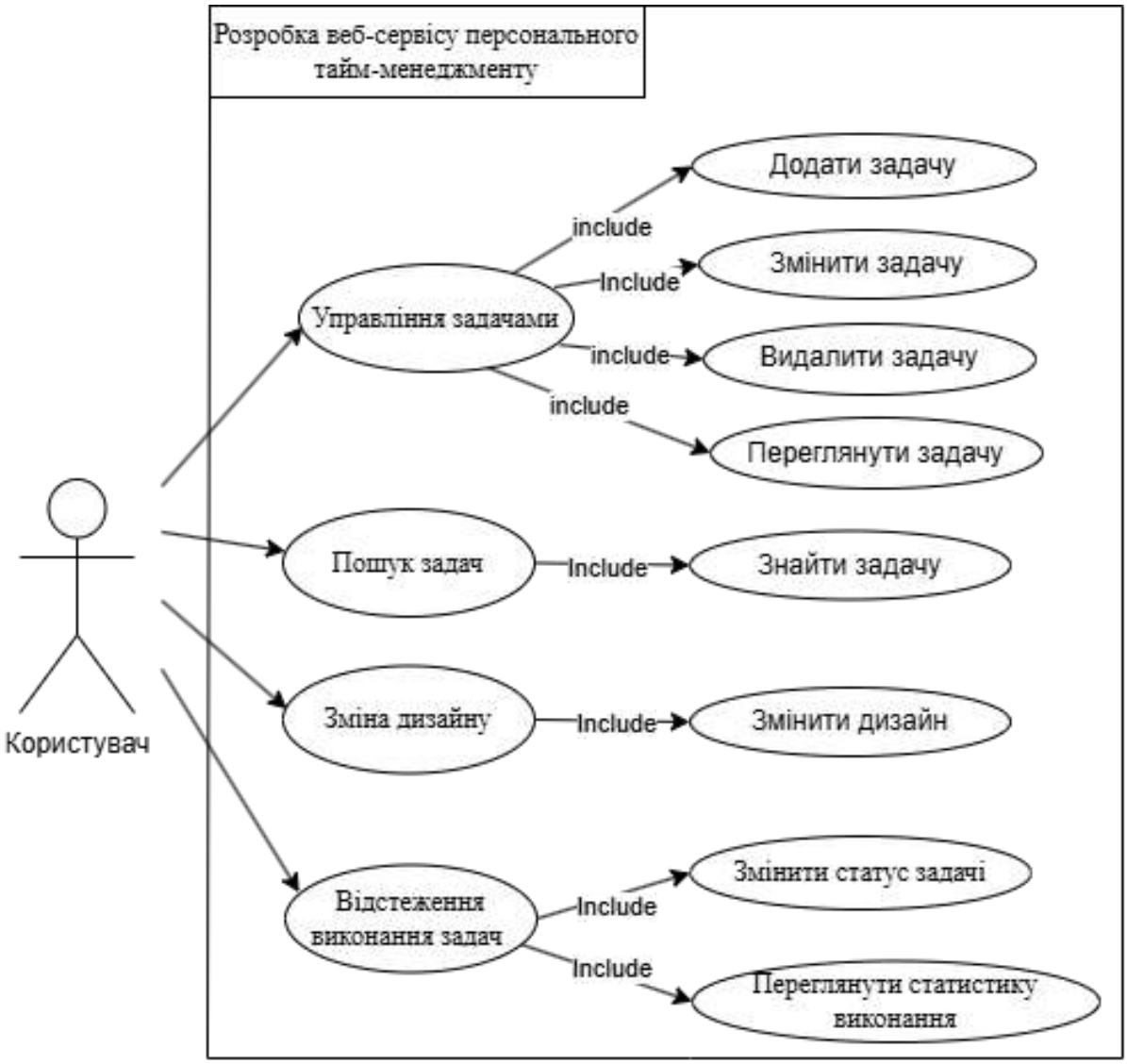

Рис. 2.4. UML-діаграма прецедентів веб-сервісу персонального тайм-менеджменту

Примітка. Джерело: розроблено автором

Наступним кроком створимо UML-діаграму класів – вона потрібна для візуалізації архітектури веб-сервісу, дозволяючи показати класи, включаючи їх

властивості, методи та взаємодії між собою. Клас у цій діаграмі складається з наступним блоків:

- верхній блок, містить назву класу, наприклад Task, Car, User;
- середній блок, показує атрибути цього класу, наприклад name: String, brand: String, number: Integer;
- кінцевий блок, демонструє методи, які цей клас може виконувати, наприклад completedTask: Boolean, deleteNumber: void.

Для веб-сервісу персонального тайм-менеджменту можна виділити такі класи:

- TaskManager – основний клас управління списками завдань. Він містить методи для створення відповідних списків, таких як список завдань на сьогодні чи список завдань з простроченими дедлайнам, а також метод для пошуку завдання;
- TaskItem – представляє окреме завдання. Містить властивості завдання, такі як: id, назва, опис, дата з часом виконання, дата з часом дедлайну, статус виконання завдання. Методи представляють собою відповідні функції для редагування та видалення завдання;
- TaskList – клас який містить список з відповідними завданнями. Має методи, що дозволяють додавати, редагувати та видаляти елементи списку;
- UIManager – відповідає за зміну теми та відображення необхідних компонентів веб-сервісу в інтерфейсі користувача, наприклад: список завдань на сьогодні, календар, форма заповнення нового завдання або редагування існуючого;
- Storage – клас роботи з LocalStorage (локальне сховище). Забезпечує методи для збереження та завантаження даних завдань та теми.

На рисунку 2.5 представлено діаграму класів, розроблену для сервісу персонального тайм-менеджменту.

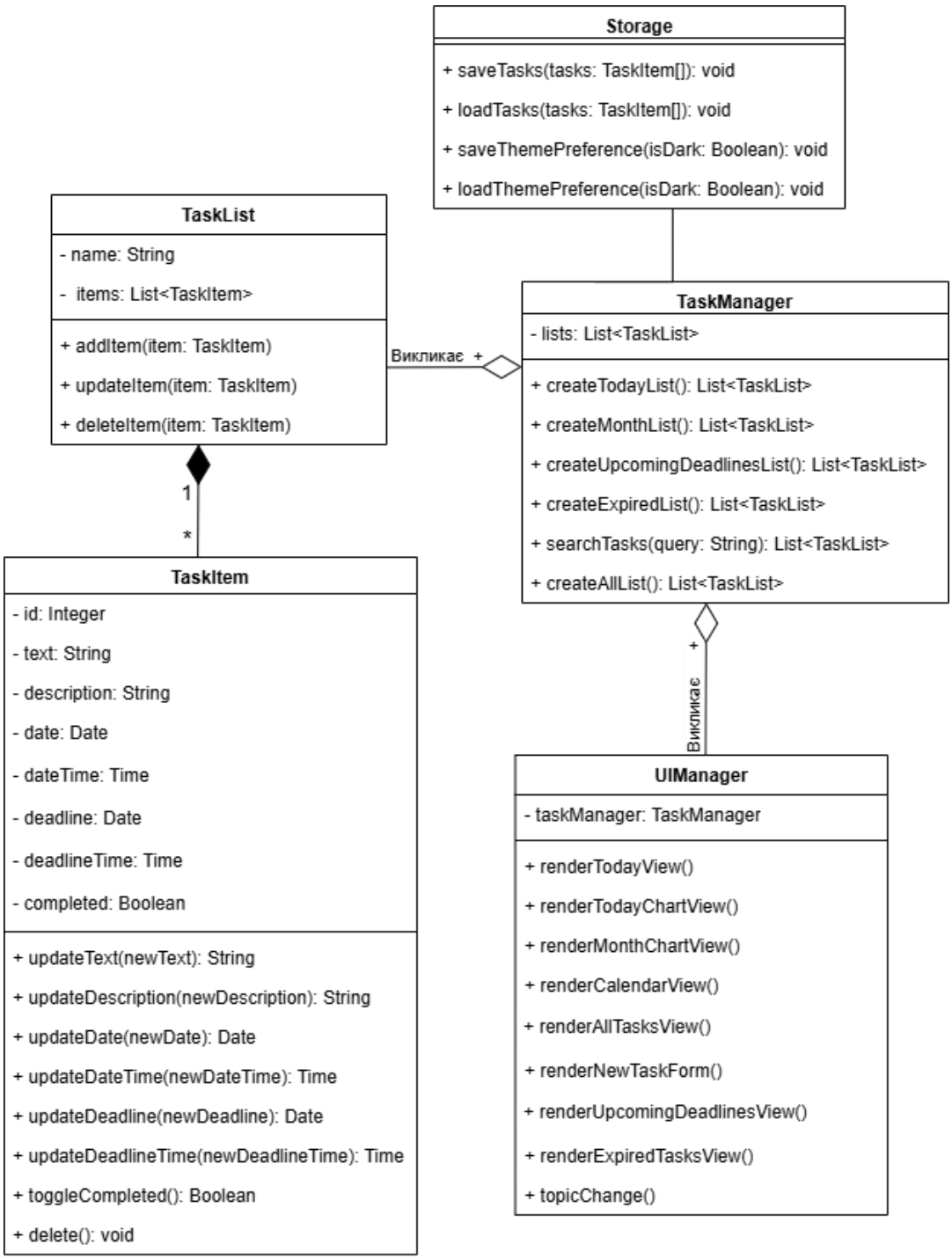

Рис. 2.5. UML-діаграма класів веб-сервісу персонального тайм-менеджменту

Примітка. Джерело: розроблено автором

На останньому кроці розробки логічної моделі приділимо увагу UML-діаграмі послідовності. З її допомогою можна подивитися, як покроково в системі взаємодіють компоненти при виконанні певного прецеденту [16].

Оскільки дані діаграми будуть схожими один з одним, оглянемо два не схожі прецеденти:

- прецедент додавання нового завдання (рис. 2.6);
- прецедент пошуку необхідного завдання, (рис. 2.7).

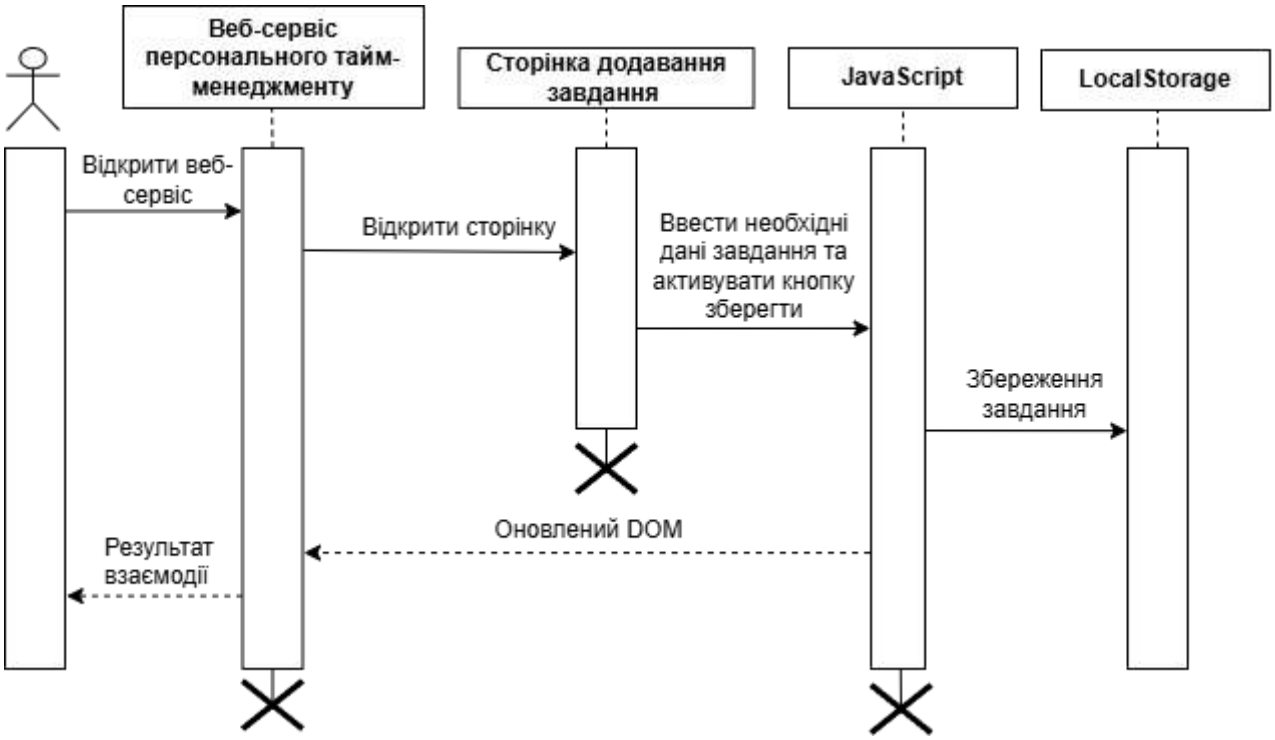

Рис. 2.6. UML-діаграма послідовності веб-сервісу персонального тайм-менеджменту прецеденту додавання нового завдання

Примітка. Джерело: розроблено автором

На рисунку 2.6 можна побачити UML-діаграму послідовності прецеденту додавання нового завдання, вона показує як взаємодіють такі компоненти, як користувач, веб-сервіс персонального тайм-менеджменту, сторінка додавання завдання, JavaScript та LocalStorage. Все розпочинається з відкриття веб-сервісу персонального тайм-менеджменту користувачем, далі йому необхідно відкрити потрібну сторінку, в даному випадку сторінку додавання нового завдання, після чого йому необхідно заповнити необхідні поля у формі та активувати кнопку збереження, внаслідок чого відбудеться збереження завдання в LocalStorage,

оновлення DOM з новим завданням і повернення користувача на головну сторінку.

UML-діаграма послідовності прецеденту пошуку необхідного завдання (рис. 2.7) має такі компоненти: користувач, веб-сервіс персонального тайм-менеджменту, сторінка пошуку, JavaScript, LocalStorage.

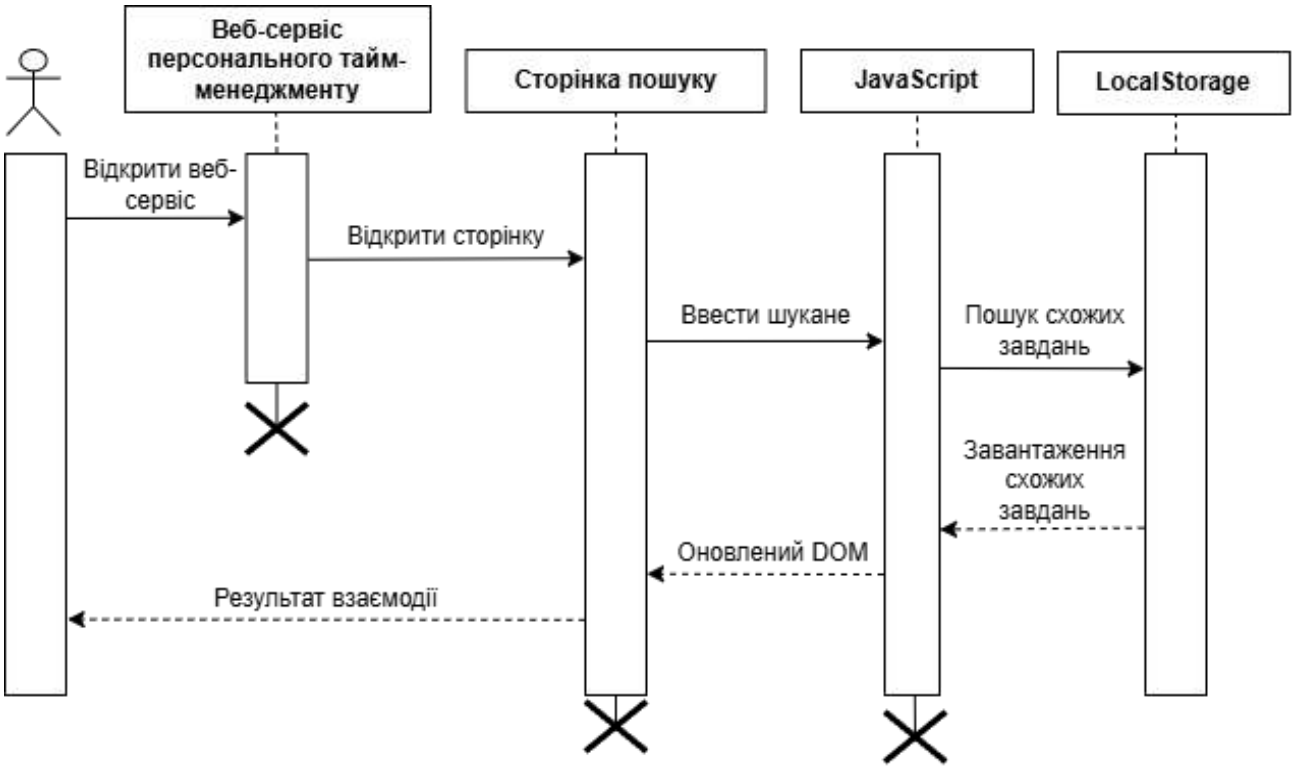

Рис. 2.7. UML-діаграма послідовності веб-сервісу персонального тайм-менеджменту прецеденту пошуку завдання

Примітка. Джерело: розроблено автором

Як і в попередньому прецеденті, все починається з відкриття веб-сервісу персонального тайм-менеджменту користувачем і його переходу на сторінку пошуку, далі йому необхідно ввести текст за яким буде здійснюватися пошук завдання. Список сторінки пошуку буде автоматично оновлюватися в міру друку тексту, так як пошук завдань в LocalStorage, завантаження їх з LocalStorage, оновлення DOM відбувається в реальному часі. Внаслідок чого користувач отримує результат взаємодії миттєво.

2.3 Розробка фізичної моделі процесу управління задачами у веб-сервісі персонального тайм-менеджменту

Після розробки логічної моделі можна розпочати створення фізичної моделі. Це ключовий етап, який трансформує логічну модель у фізичну структуру, що забезпечує зберігання та керування даними користувачів. На цьому етапі детально визначаються компоненти системи, їх зв'язки та розміщення в обчислювальному середовищі, що забезпечує ефективну роботу веб-сервісу та надає можливість масштабувати його функціональність відповідно до вимог користувачів. Також фізична модель дозволяє оптимізувати ресурси та забезпечити правильну організацію роботи всіх елементів програмного продукту, щоб ідентифікувати потенційні проблеми та своєчасно їх усувати.

Починати розробку фізичної моделі слід з визначення вимог до програмного продукту. Ці вимоги включають не лише функціональність, але й масштабованість та продуктивність програмного продукту. Надалі вони стануть специфікаціями, на яких будуть створюватись компоненти та їх взаємодія. Для прикладу можна навести механізм зберігання даних у LocalStorage, сторінки веб-інтерфейсу для взаємодії з користувачами, вони мають бути спроєктовані з урахуванням цих специфікацій для забезпечення стабільної та ефективної роботи веб-сервісу персонального тайм-менеджменту.

Потрібно зазначити, що розробка фізичної моделі є ітеративним процесом, що дозволяє адаптувати систему, видаляючи, змінюючи, додаючи необхідні можливості у реальному часі. Це відбувається за допомогою виявлення нових вимог чи результатів тестування. Важливо забезпечити ефективний моніторинг та керування програмним продуктом після його розгортання, щоб дати користувачеві високий рівень досвіду використання, своєчасним реагуванням на можливі проблеми з продуктивністю.

Слід стверджувати, що розробка фізичної моделі є ключовим етапом на шляху створення стабільної та ефективної системи, що забезпечує високий

рівень продуктивності та зручність користування. Вона дозволяє не тільки реалізувати вимоги до функціональності, але й забезпечити правильну інтеграцію в обчислювальне середовище, що гарантує стабільну роботу веб-сервісу протягом усього циклу створення.

Після створення та опису діаграм логічної моделі, розбору цілей створення фізичної моделі, ми можемо перейти до розробки UML-діаграм фізичної моделі.

UML-діаграма компонентів – це діаграма, що дозволяє відобразити взаємодію основних компонентів та їх інтеграцію із попередньо розробленими логічними моделями, сприяючи створенню цілісної та ефективної архітектури системи [17].

UML-діаграма компонентів веб-сервісу персонального тайм-менеджменту відображає взаємозв'язки між такими основними компонентами системи, як користувач, веб-сервіс, веб-інтерфейс, завдання та дизайн, діаграма представлено на рисунку 2.8, вона є важливою у визначенні взаємодії між компонентами при розробці програмного продукту для ефективного управлінням завданнями.

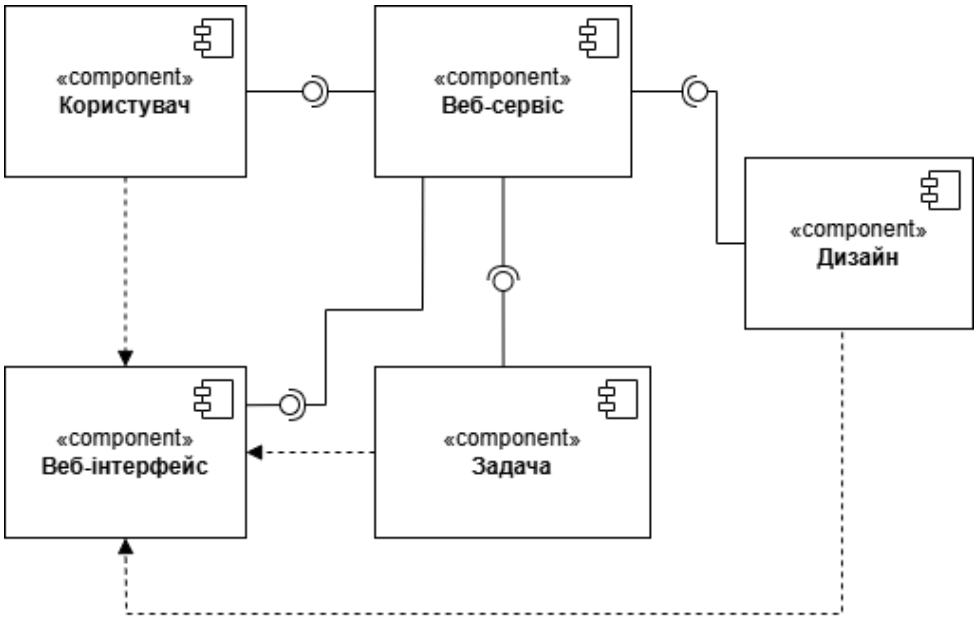

Рис. 2.8. UML-діаграма компонентів веб-сервісу персонального тайм-менеджменту

Примітка. Джерело: розроблено автором

Другою діаграмою фізичної моделі веб-сервісу персонального тайм-менеджменту стане ER-діаграма UML – це діаграма, яка необхідна для відображення об'єктів та їх взаємодії, що дозволить точніше розробити базу даних [18]. В ER-діаграмі UML веб-сервісу персонального тайм-менеджменту (рис. 2.9) було виділено три об'єкти: користувач, завдання та дизайн. На ній наведено приклад, як один користувач змінює дизайн та створює безліч завдань. У роботі буде використовуватися LocalStorage для зберігання даних.

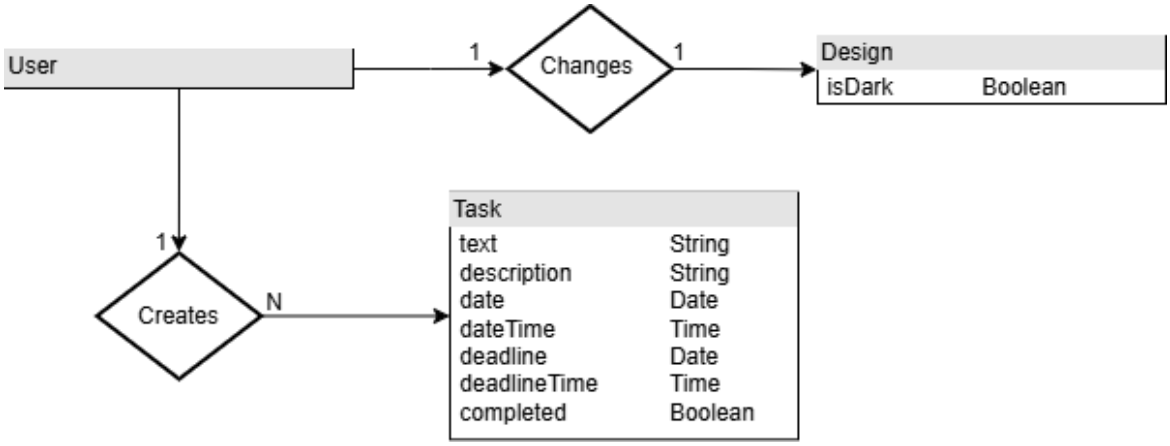

Рис. 2.9. ER-діаграма UML веб-сервісу персонального тайм-менеджменту

Примітка. Джерело: розроблено автором

LocalStorage – це гарне та ефективне рішення для зберігання даних на стороні клієнта, що дозволяє зберігати та завантажувати інформацію про завдання користувачів та інші налаштування без необхідності звертатися до зовнішнього серверу. Це дозволяє створити надійну структуру для зберігання та взаємодії з даними, забезпечуючи користувача стабільною роботою навіть за високих навантажень системи.

Висновки до розділу 2

У другому розділі було проведено аналіз фреймворків та комплексне моделювання процесу управління задачами у веб-сервісі персонального тайм-менеджменту. Цей етап є основою для створення фундаменту майбутнього

програмного продукту, оскільки забезпечує чітке розуміння архітектури, основних компонентів та взаємозв'язків між ними. Завдяки використання логічного підходу вдалося визначити ключові елементи, необхідні для ефективної реалізації функціональності веб-сервісу.

Розробка логічної моделі дозволило деталізувати сутності, які забезпечують управління задачами. Зокрема, було створено UML-діаграму прецедентів, що описує взаємодію користувачів із програмним продуктом через її основні функціональні модулі, такі як створення, редагування, видалення, пошук необхідного, зміна статусу виконання та перегляд статистики виконання завдання, та зміна дизайну. Крім того, UML-діаграма класів визначила структуру об’єктів програмного продукту, їх властивості та методи, що забезпечують взаємодію між компонентами. На основі цих даних було розроблено UML-діаграми послідовності, які демонструють покрокову взаємодію в веб-сервісі, між відповідною сторінкою, логікою JavaScript та LocalStorage. Ці діаграми описують процеси виконання користувацьких сценаріїв: від відкриття користувачем веб-сервісу до обробки дій у JavaScript і подальшого збереження або оновлення даних у LocalStorage. Завдяки такому підходу вдалося проілюструвати механізми роботи системи та забезпечити їхню прозорість для подальшої реалізації.

Фізична модель, у свою чергу, деталізувала логічну структуру, трансформуючи її у практичну реалізацію, орієнтовану на обчислювальне середовище. У межах цього етапу було розглянуто використання LocalStorage для зберігання даних про задачі користувачів та інші налаштування. Цей підхід забезпечує швидкий доступ до інформації та незалежність від серверного сховища. Була створена UML-діаграма компонентів, яка ілюструє взаємозв’язки між основними модулями програмного продукту. Ця діаграма не лише демонструє, як реалізуються функціональні вимоги користувачів, але й допомагає у подальшому вдосконаленні програмного продукту. Розробка ER-діаграми логічної моделі даних дозволила точніше опрацювати структуру

інформації та визначити залежності між елементами, що є основою для ефективної роботи сховища даних.

Таким чином, моделювання процесу управління задачами стало ключовим етапом у створенні веб-сервісу персонального тайм-менеджменту. Логічна та фізична модель, забезпечили структурований підхід до проєктування системи, зменшили ризики виникнення помилок у подальшій реалізації та дозволили підготувати програмний продукт до майбутнього масштабування. Завдяки цьому створений веб-сервіс відповідатиме сучасним вимогам користувачів, поєднуючи зручність, стабільність і високу продуктивність, що є запорукою успіху в реальних умовах використання.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ ПЕРСОНАЛЬНОГО ТАЙМ-МЕНЕДЖМЕНТУ

3.1 Програмна реалізація проєкту

Веб-сервіс персонального тайм-менеджменту MyTaskManager був створений для ефективного планування та управління часом користувачів. Він дозволяє як створювати, редагувати, видаляти завдання, так й переглядати і аналізувати їх виконання. При цьому, веб-сервіс використовує LocalStorage, що дозволило відмовитись від серверного сховища для збереження даних, підвищуючи конфіденційність користувачів.

Алгоритм веб-сервісу персонального тайм-менеджменту складається з багатьох компонентів, у цьому підрозділі буде розглянуто три з них:

- AppHeader.vue, який створює шапку веб-сервісу, що містить панель для навігації та кнопку-іконку зміни теми оформлення веб-сервісу;
- TodayList.vue, що містить список завдань на поточний день;
- TaskForm.vue, компонент, який використовується для створення або редагування завдання.

Компонент ділиться на три частини, перша частина містить шаблон, друга частина містить скрипт, третя частина містить стиль. Оскільки частина стилю має великий обсяг, буде оглянуто тільки частини шаблону та скрипту.

Перший компонент – це AppHeader.vue, який відображає шапку веб-сервісу. Його шаблон (рис. 3.1), містить такі блоки коду:

- 3-6 рядок – створює логотип та назву веб-сервісу персонального тайм-менеджменту MyTaskManager;
- 7-20 рядок – потрібний для відображення навігаційних посилань у вигляді кнопок-іконок, чотирьох сторінок: «Сьогодні», «Календар», «Пошук», «Додавання нового завдання»;

– 21-23 рядок – відображає кнопку-іконку зміни оформлення теми веб-сервісу, що при активації змінюється на протилежну до теми.

```
<template>
    <header class="app-header">
        <div class="logo-container">
            <img src="@/assets/Favicon.png" alt="Logo" class="logo" />
            <h1 class="title">MyTaskManager</h1>
        </div>
        <nav class="nav-links">
            <router-link to="/" class="nav-link">
                <img src="@/assets/icon-only-today.png" class="image-button" alt="Today" />
            </router-link>
            <router-link to="/calendar" class="nav-link">
                <img src="@/assets/icon-calendar.png" class="image-button" alt="Calendar" />
            </router-link>
            <router-link to="/tasks-list" class="nav-link">
                <img src="@/assets/icon-font-search.png" class="image-button" alt="Tasks List" />
            </router-link>
            <router-link to="/new-task" class="nav-link">
                <img src="@/assets/icon-post.png" class="image-button" alt="New Task" />
            </router-link>
        </nav>
        <button id="image-button" @click="toggleTheme" class="nav-link theme-toggle">
            <img :src="themeIcon" alt="Toggle Theme" class="image-button" />
        </button>
    </header>
</template>
```

Рис. 3.1. Шаблон компонента AppHeader.vue

Примітка. Джерело: розроблено автором

Скрипт компонента AppHeader.vue (рис. 3.2), містить наступні блоки коду:

– 28-29 рядок – компонент експортується з назвою AppHeader для використання його в інших частинах веб-сервісу;

– 30-35 рядок – містить початкові реактивні дані, такі як тема та іконка теми;

– 36-38 рядок – під час монтування активується метод, що завантажує налаштування теми користувача;

– 40-48 рядок – при активації методу змінюється оформлення теми веб-сервісу, оновлюється DOM, змінюється кнопка-іконка та зберігається вибір у LocalStorage;

– 49-56 рядок – якщо темна тема збережена у LocalStorage, при відкритті веб-сервісу активується даний метод, який змінює оформлення теми та змінює кнопку-іконку.

```
<script>
export default {
    name: 'AppHeader',
    data() {
        return {
            isDarkTheme: false,
            themeIcon: require('@/assets/icon-dark-theme.png'),
        };
    },
    mounted() {
        this.loadThemePreference();
    },
    methods: {
        toggleTheme() {
            this.isDarkTheme = !this.isDarkTheme;
            document.body.classList.toggle('dark-theme', this.isDarkTheme);
            localStorage.setItem('theme', this.isDarkTheme ? 'dark' : 'light');

            this.themeIcon = this.isDarkTheme
                ? require('@/assets/icon-light-theme.png')
                : require('@/assets/icon-dark-theme.png');
        },
        loadThemePreference() {
            const savedTheme = localStorage.getItem('theme');
            if (savedTheme === 'dark') {
                this.isDarkTheme = true;
                document.body.classList.add('dark-theme');
                this.themeIcon = require('@/assets/icon-light-theme.png');
            }
        },
    },
};
</script>
```

Рис. 3.2. Скрипт компонента AppHeader.vue

Примітка. Джерело: розроблено автором

Наступним компонентом буде розглянуто TodayList.vue, що відображає завдання на поточний день. Шаблон компонента TodayList.vue було розбито на такі блоки коду (рис. 3.3):

– 2-3 рядок – створює контейнер із заголовком «Завдання на сьогодні», в якому будуть утримуватися наступні блоки зі списку;

– 5-10 рядок – даний код здійснює перевірку на наявність завдань на сьогодні, якщо завдання присутні він групує їх за часом початку виконання і відображає користувачу наступним чином: час початку та список завдань на цей час;

– 11-25 рядок – є макетом кожного завдання, він містить кнопки-іконки для виконання, редагування та видалення завдання, відображає текст завдання і, за наявності, дедлайн. Також, завдяки CSS-класам, візуально виділяє виконане та прострочене завдання;
– 26-31 рядок – закриття всіх блоків та, якщо немає завдань, виведення тексту: «На сьогодні завдань немає».

```
1  <template>
2      <div class="container">
3          <h2>Завдання на сьогодні</h2>
4          <hr>
5          <div v-if="todayTasks.length" class="task-container">
6              <div v-for="(tasks, time) in groupedTasks" :key="time" class="task-group">
7                  <div class="task-time">{{ time }}</div>
8                  <ul class="my-list">
9                      <li v-for="task in tasks" :key="task.id"
10                         :class="{ 'completed': task.completed, 'overdue': isOverdue(task) }">
11                         <button @click="toggleComplete(task)" class="image-button-calendar-item">
12                             <img src="@/assets/icon-done.png" id="image-button-calendar-item">
13                         </button>
14                         <router-link :to="'/edit-task/' + task.id" class="image-button-calendar-item">
15                             <img src="@/assets/icon-edit.png" id="image-button-calendar-item">
16                         </router-link>
17                         <button @click="deleteTask(task.id)" class="image-button-calendar-item">
18                             <img src="@/assets/icon-delete.png" id="image-button-calendar-item" />
19                         </button>
20                         <div class="task-content">
21                             <span class="task-title" :title="task.title">{{ task.title }}</span>
22                             <span v-if="task.deadlineDate && task.deadlineTime" class="task-deadline">
23                                 до {{ getDeadlineDisplay(task) }}
24                             </span>
25                         </div>
26                     </li>
27                 </ul>
28             </div>
29         </div>
30         <p v-if="!todayTasks.length">На сьогодні завдань немає.</p>
31     </div>
32 </template>
```

Рис. 3.3. Шаблон компонента TodayList.vue

Примітка. Джерело: розроблено автором

Перша частина скрипту компонента TodayList.vue (рис. 3.4) містить два блоки коду:

– 35-36 рядок – необхідний для імпортування функцій з центрального сховища стану та експортування об'єкта компонента Vue;
– 37-53 рядок – даний блок коду поділено на дві частини: перша частина необхідна для фільтрації завдань з сьогоднішньою датою, а друга частина

використовується для групування за часом відфільтрованих завдань, сортуючи їх за зростанням.

```
34  <script>
35  import store from '@/store/state.js';
36  export default {
37      computed: {
38          todayTasks() {
39              const today = new Date().toISOString().split('T')[0];
40              return store.state.tasks.filter(task => task.date === today);
41          },
42          groupedTasks() {
43              const grouped = {};
44              this.todayTasks.forEach(task => {
45                  let time = task.time || '00:00';
46                  if (!grouped[time]) {
47                      grouped[time] = [];
48                  }
49                  grouped[time].push(task);
50              });
51              return Object.fromEntries(Object.entries(grouped).sort());
52          }
53      },
```

Рис. 3.4. Перша частина скрипту компонента TodayList.vue

Примітка. Джерело: розроблено автором

Друга частина демонструє третій блок коду скрипту, методи компонента TodayList.vue (рис. 3.5):

- 55-57 рядок – метод, який перемикає положення стану завдання між виконаним і не виконаним через функцію у сховищі;
- 58-61 рядок – код, необхідний для визначення простроченого завдання шляхом порівняння дати та часу дедлайну з поточним положенням дати, часу та перевіркою на виконання;
- 62-68 рядок – форматується відображення дедлайну, якщо дата дедлайну припадає на сьогодні, то відображатиметься лише час, в іншому випадку дедлайн відображатиметься як дата та час;
- 69-72 рядок – метод, що виконує видалення завдання, видаляючи зі сховища та оновлюючи DOM.

```
54      methods: {
55          toggleComplete(task) {
56              store.toggleTaskCompletion(task);
57          },
58          isOverdue(task) {
59              const deadlineDate = new Date(`${task.deadlineDate}T${task.deadlineTime}`);
60              return task.deadlineDate && deadlineDate < new Date() && !task.completed;
61          },
62          getDeadlineDisplay(task) {
63              const [year, month, day] = task.deadlineDate.split('-');
64              const formattedDate = `${day}.${month}.${year}`;
65              return task.deadlineDate === new Date().toISOString().split('T')[0]
66                  ? task.deadlineTime
67                  : `${formattedDate} ${task.deadlineTime}`;
68          },
69          deleteTask(id) {
70              store.removeTask(id);
71              this.$emit('taskDeleted', id);
72          },
73      }
74  };
75  </script>
```

Рис. 3.5. Друга частина скрипту компонента TodayList.vue

Примітка. Джерело: розроблено автором

Останнім буде розглянуто компонент TaskForm.vue, який потрібен для створення форми додавання або редагування завдання. Його шаблон (рис. 3.6) містить наступні блоки коду:

- 2-3 рядок – створення контейнера, у якому буде розміщено форму для додавання та редагування завдання;
- 4 рядок – заголовок, який змінюватиметься залежно від дії у формі, створення нового або редагування існуючого завдання;
- 6 рядок – відкриття тегу форми, щоб не перезавантажувалася сторінка під час відправлення;
- 7-8 рядок – блок коду, який створює поля в які необхідно вводити завдання та його опис, при цьому опис не є обов'язковим для заповнення;
- 9-15 рядок – код, що створює місця для введення дати і часу початку виконання завдання, вони є обов'язковими для заповнення;
- 16-23 рядок – блок коду, який відображає поля для введення дати та часу дедлайну, при цьому вони не є обов'язковими для заповнення;

– 24-31 рядок – створення кнопок, які дозволяють швидко додати дедлайн на відповідну кількість днів після дати початку виконання;
– 33-40 рядок – блок створює кнопки збереження та видалення завдання, при цьому кнопка видалення буде доступна тільки в режимі редагування завдання.

```
<template>
    <div class="main">
        <div class="container">
            <h2>{{ isEditMode ? 'Редагування' : 'Нове завдання' }}</h2>
            <hr />
            <form @submit.prevent="submitForm" id="new-task-form">
                <input type="text" v-model="taskData.title" placeholder="Введіть заголовок..." required />
                <textarea id="description" v-model="taskData.description" placeholder="Введіть опис..."></textarea>
                <div class="form-row">
                    <label for="date">Дата</label>
                    <div class="deadline-container">
                        <input type="date" id="date" v-model="taskData.date" required />
                        <input type="time" id="time" v-model="taskData.time" required />
                    </div>
                </div>
                <div class="form-row">
                    <label>Дедлайн</label>
                    <div class="deadline-container">
                        <input type="date" id="deadline-date" v-model="taskData.deadlineDate" />
                        <input type="time" id="deadline-time" v-model="taskData.deadlineTime"
                            :required="isDeadlineTimeRequired" />
                    </div>
                </div>
                <div class="form-row">
                    <div class="deadline-buttons">
                        <button type="button" @click="addDaysToDeadline(1)">1 день</button>
                        <button type="button" @click="addDaysToDeadline(3)">3 дні</button>
                        <button type="button" @click="addDaysToDeadline(5)">5 днів</button>
                        <button type="button" @click="addDaysToDeadline(7)">7 днів</button>
                    </div>
                </div>
                <hr />
                <div class="button-container">
                    <button type="submit" id="image-button">
                        <img src="@/assets/icon-post.png" alt="Сохранить" class="image-button" />
                    </button>
                    <button type="button" id="image-button" v-if="isEditMode" @click="deleteTask">
                        <img src="@/assets/icon-delete.png" alt="Удалить" class="image-button" />
                    </button>
                </div>
            </form>
        </div>
    </div>
</template>
```

Рис. 3.6. Шаблон компонента TaskForm.vue

Примітка. Джерело: розроблено автором

Скрипт компонента TaskForm.vue, має великий обсяг, тому був розбитий на дві частини. У першій частині скрипту компонента TaskForm.vue (рис. 3.7) виділяються такі блоки коду:

– 47-48 рядок – використовується для імпорту даних зі сховища та експорту компонента;
– 49 рядок – цей рядок коду необхідний для отримання даних ззовні компонента під час редагування завдання;
– 50-63 рядок – якщо через props були отримані дані, у блоці відбувається їхнє розбиття на частини, дата і час виконання завдання буде розбито на окремі поля, так само і з датою та часом дедлайну. За відсутності даних будуть ініціалізовані порожні поля;
– 64-68 рядок – цей блок коду визначає, чи перебуває компонент у положенні редагування, через перевірку завдання на наявність id;
– 69-73 рядок – блок стежить, чи вказана дата дедлайну і ставить прапор за її наявності.

```
46  <script>
47  import store from '@/store/state.js';
48  export default {
49      props: ['task'],
50      data() {
51          return {
52              taskData: this.task
53                  ? {
54                      ...this.task,
55                      date: this.task.dateTime?.split('T')[0] || '',
56                      time: this.task.dateTime?.split('T')[1] || '',
57                      deadlineDate: this.task.deadline ? this.task.deadline.split('T')[0] : '',
58                      deadlineTime: this.task.deadline ? this.task.deadline.split('T')[1] : ''
59                  }
60                  : { title: '', description: '', date: '', time: '', deadlineDate: '', deadlineTime: '' },
61              isDeadlineTimeRequired: false
62          };
63      },
64      computed: {
65          isEditMode() {
66              return !!this.task?.id;
67          }
68      },
69      watch: {
70          'taskData.deadlineDate'(newVal) {
71              this.isDeadlineTimeRequired = !!newVal;
72          }
73      },
```

Рис. 3.7. Перша частина скрипту компонента TaskForm.vue

Примітка. Джерело: розроблено автором

Друга частина скрипту компонента TaskForm.vue (рис. 3.8), містить наступні методи:

– 75-88 рядок – використовується під час активації кнопки збереження, щоб зберегти нове або відредаговане завдання. Після збереження даних відбувається перенаправлення користувача на головну сторінку;
– 89-94 рядок – дозволяє видалити завдання у режимі редагування. При цьому в результаті видалення перенаправляє користувача на головну сторінку;
– 95-107 рядок – метод який виконується для швидкого введення дедлайну через кнопки, додаючи відповідну кількість днів до дати виконання, залишаючи час не змінним.

```
74      methods: {
75          submitForm() {
76              this.taskData.dateTime = `${this.taskData.date}T${this.taskData.time}`;
77              this.taskData.deadline = this.taskData.deadlineDate && this.taskData.deadlineTime
78                  ? `${this.taskData.deadlineDate}T${this.taskData.deadlineTime}`
79                  : null;
80
81              if (this.isEditMode) {
82                  store.updateTask(this.taskData);
83              } else {
84                  this.taskData.id = Date.now();
85                  store.addTask(this.taskData);
86              }
87              this.$router.push('/');
88          },
89          deleteTask() {
90              if (this.isEditMode) {
91                  store.removeTask(this.taskData.id);
92              }
93              this.$router.push('/');
94          },
95          addDaysToDeadline(days) {
96              if (this.taskData.date && this.taskData.time) {
97                  const [year, month, day] = this.taskData.date.split('-');
98                  const [hour, minute] = this.taskData.time.split(':');
99
100                 const taskDate = new Date(Date.UTC(year, month - 1, day, hour, minute));
101
102                 taskDate.setUTCDate(taskDate.getUTCDate() + days);
103
104                 this.taskData.deadlineDate = taskDate.toISOString().split('T')[0];
105                 this.taskData.deadlineTime = taskDate.toISOString().split('T')[1].slice(0, 5);
106             }
107         }
108     }
109 };
110 </script>
```

Рис. 3.8. Друга частина скрипту компонента TaskForm.vue

Примітка. Джерело: розроблено автором

Після опису частини компонентів можна перейти до огляду сторінок веб-сервісу, що дозволить зрозуміти структуру, функціональність і логіку при взаємодії користувача з інтерфейсом.

3.2 Інтерфейс веб-сервісу персонального тайм-менеджменту

Веб-сервіс персонального тайм-менеджменту MyTaskManager складається з чотирьох сторінок, на кожній з них є шапка, яка складається з трьох частин: перша частина демонструє логотип і назву веб-сервісу, друга показує кнопки-іконки навігаційної панелі, третя відображає кнопку зміни теми оформлення. Сторінка «Сьогодні» (рис. 3.9) відображається під час відкриття веб-сервісу, дозволяючи користувачеві без зайвих рухів переглянути завдання, які були заплановані на відповідний час цього дня.

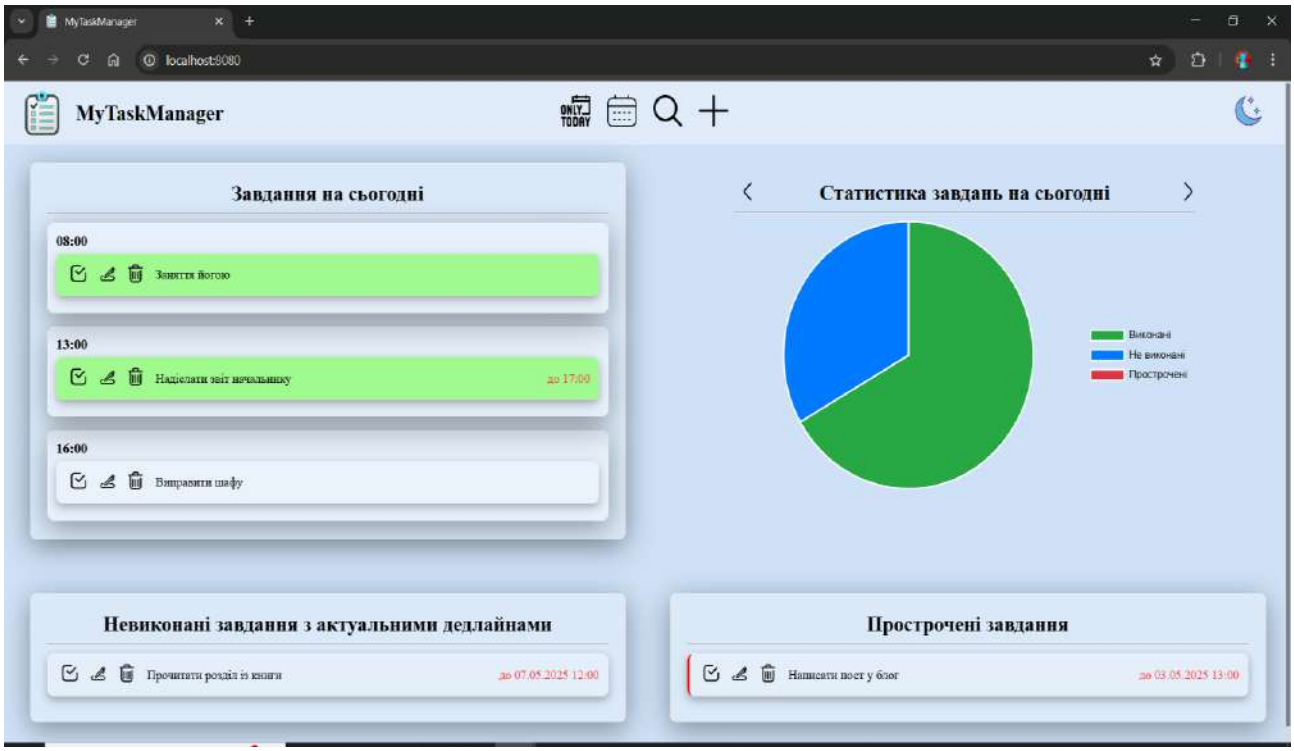

Рис. 3.9. Сторінка «Сьогодні» веб-сервісу персонального тайм-менеджменту MyTaskManager

Примітка. Джерело: розроблено автором

Також дана сторінка дозволяє користувачеві аналізувати виконання завдань за день або місяць, використовуючи кругову діаграму. Вона відображає статистику завдань за такими критеріями як: виконані завдання, не виконані завдання, завдання з простроченим дедлайном. У діаграмі статистики завдань за місяць додано додаткову категорію – завдання з актуальними дедлайнами. Для аналізу виконання завдань також допоможуть два списки: перший демонструє завдання з простроченими дедлайнами, а другий завдання, у яких дедлайн скоро закінчиться.

Сторінка «Календар» (рис. 3.10) дає можливість перегляду завдань на певний день шляхом знаходження дня через зручний календар. Він містить два селектори, перший перемикає місяці, а другий роки. Перед селекторами стоїть кнопка-іконка для швидкого переходу на поточний день календаря, який виділяється синім кольором. У кожному контейнері дня може відображатися до двох завдань, якщо їх більше, то зверху праворуч з'являється мітка у вигляді кола з кількістю прихованих завдань. Мітка може бути двома кольорами: червоною, якщо є не виконане завдання і зеленою, якщо всі завдання виконані.

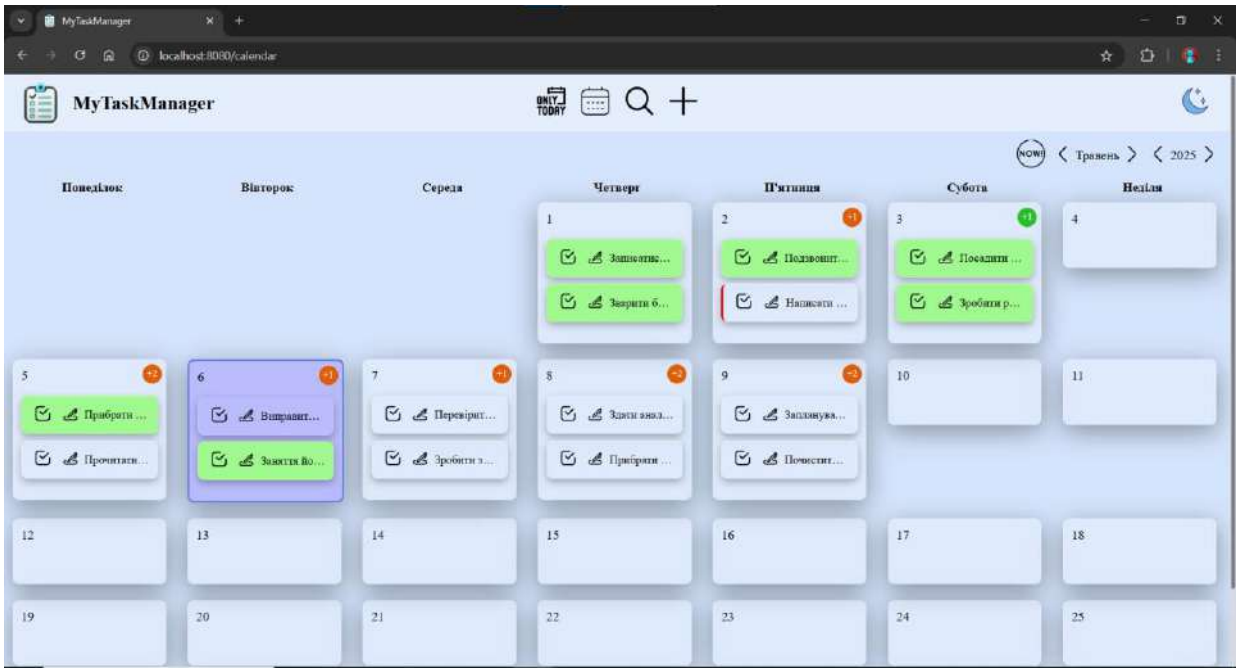

Рис. 3.10. Сторінка «Календар» веб-сервісу персонального тайм-менеджменту MyTaskManager

Примітка. Джерело: розроблено автором

Сторінка «Пошук» (рис. 3.11) представляє з себе функціонал, що дозволяє знайти необхідне завдання, шляхом введення частини тексту завдання або його опису в поле, призначене для пошуку тексту. Під полем введення тексту знаходиться список із завданнями, який, за замовчуванням, заповнений усіма завданнями. Коли користувач вводить текст, список оновлюється в режимі реального часу, що дозволяє швидше знайти потрібне завдання.

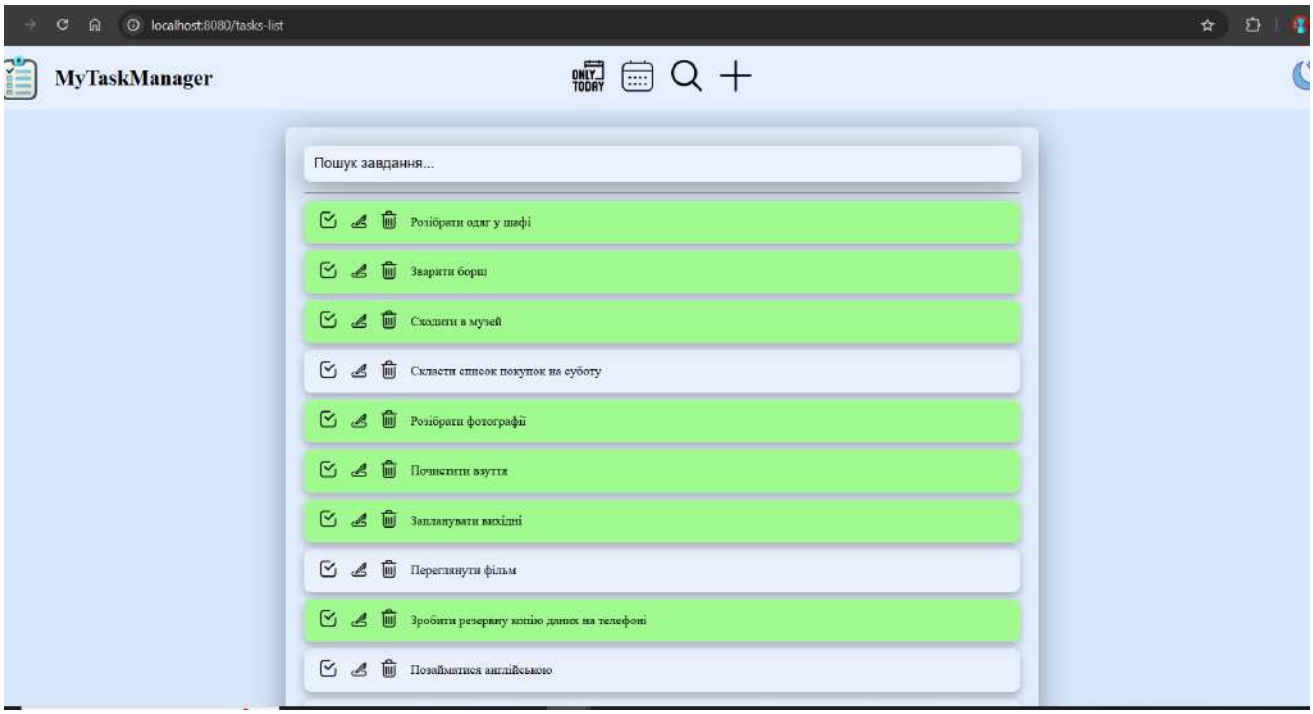

Рис. 3.11. Сторінка «Пошук» веб-сервісу персонального тайм-менеджменту MyTaskManager

Примітка. Джерело: розроблено автором

Сторінка «Додавання нового завдання» (рис. 3.12) являє собою форму, яка використовується для створення нового або редагування існуючого завдання. Вона включає поле для введення завдання, поле для введення опису, поля для введення дати і часу виконання, а також дати і часу дедлайну, кнопки для швидкого додавання дедлайну і кнопку збереження завдання. Під час редагування завдання заголовок форми змінюється, а поруч із кнопкою збереження з'являється додаткова кнопка видалення.

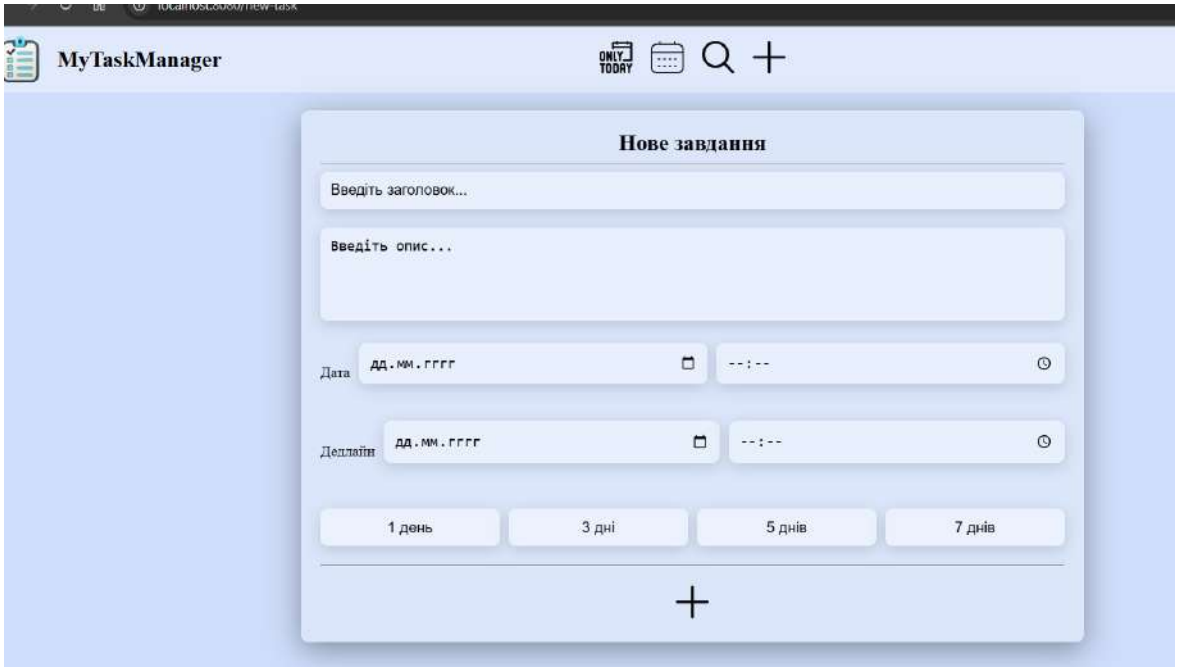

Рис. 3.12. Сторінка «Додавання нового завдання» веб-сервісу персонального тайм-менеджменту MyTaskManager

Примітка. Джерело: розроблено автором

Огляд сторінок веб-сервісу показав наявність лаконічності та інтуїтивної простоти в інтерфейсі. На рисунках відображено чітку структуру сторінок з логічним розміщенням елементів, що забезпечує зручність використання. При цьому був відстежений простий і зрозумілий дизайн, що в цей же час виділяє сервіс серед конкурентів.

3.3 Тестування веб-сервісу персонального тайм-менеджменту

Процес тестування веб-сервісу персонального тайм-менеджменту складається з двох частин: аналізу шляху та перегляду результатів п'яти сценаріїв, таких як:

- створення нового завдання;
- редагування існуючого завдання;
- пошук завдання за описом через сторінку «Пошук»;
- пошук завдань на певний день;

– зміна теми оформлення веб-сервісу.

Усі сценарії розглядатимуться на комп’ютері зі сторінки «Сьогодні», оскільки це перша сторінка після відкриття веб-сервісу персонального тайм-менеджменту.

Першим сценарієм є створення нового завдання. Для цього необхідно перейти на сторінку «Додавання нового завдання» та заповнити необхідні поля. Заповнивши відповідні поля для завдання «Перевірити пошту», яке необхідно виконати 08.05.2025 о 12:00, дедлайн у цьому завданні до 15:00 відповідного дня, а опис містить: «Має написати Віктор Володимирович за нарахування», можна натиснути кнопку Зберегти, після чого завдання зберігається, а користувач з сторінки із формою перенаправляється на сторінку «Сьогодні». Дату можна ввести за допомогою зручного маленького календаря (рис. 3.13), який викликається відповідною кнопкою у полі введення дати.

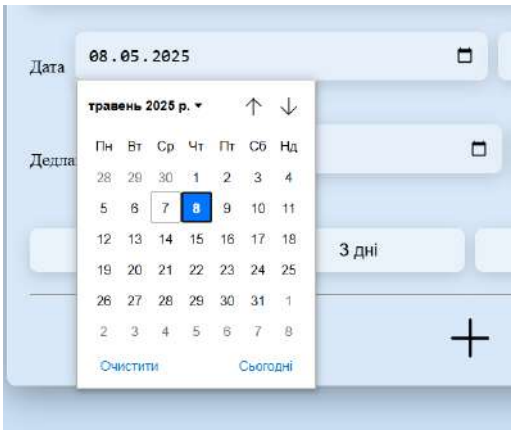

Рисунок 3.13. Календар для введення дати

Примітка. Джерело: розроблено автором

Для перевірки можна використати як сторінку «Календар» так і сторінку «Пошук», в даному сценарії використовуємо сторінку «Пошук» і переконуємося, що завдання було додано на початок списку (рис. 3.14), у вигляді стандартного макета завдання, який відображає три кнопки: помітки як виконане, редагування та видалення, а також містить саме завдання та його дедлайн.

Для другого сценарію стверджуватиметься, що в завданні з першого сценарію була допущена помилка. Щоб її вирішити, необхідно активувати кнопку-іконку редагування, після чого користувача буде направлено у форму для редагування завдання, поля якої заповнені відповідними даними.

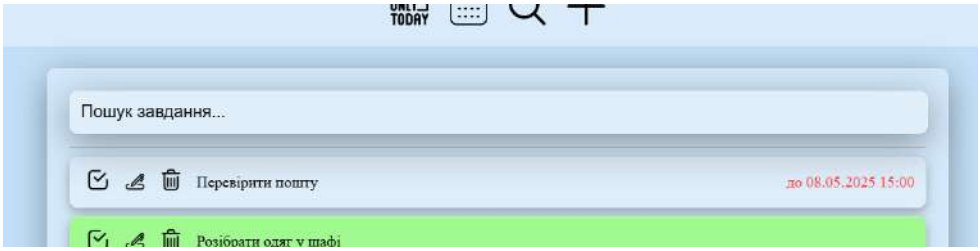

Рис. 3.14. Результат додавання нового завдання

Примітка. Джерело: розроблено автором

Редагуємо завдання з «Перевірити пошту» на «Перевірити робочу пошту» та активуємо кнопку-іконку збереження, яка перенаправляє на сторінку «Сьогодні», а завдання зберігається відредагованим. Після чого знову можна перейти на сторінку «Пошук» та переконатися, що завдання було оновлено (рис. 3.15).

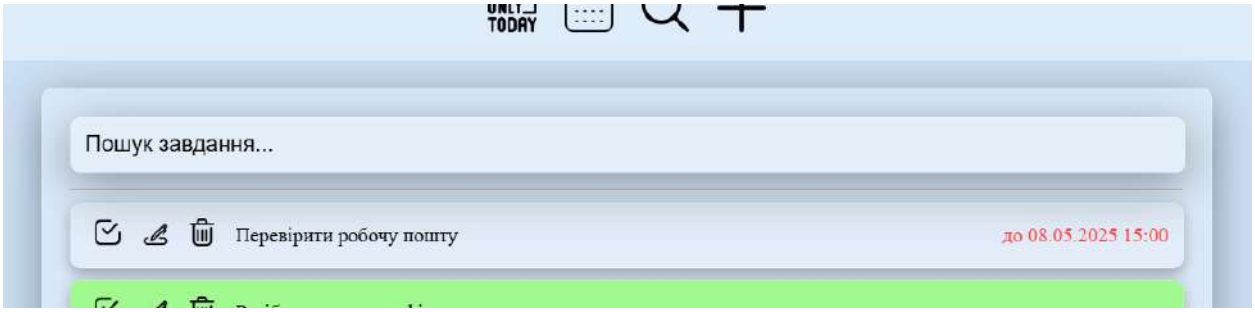

Рис. 3.15. Результат редагування завдання

Примітка. Джерело: розроблено автором

Для наступного сценарію − пошуку необхідного завдання за описом, було створено завдання: «Сходити за покупками в супермаркет перед святом», з описом, який включає список продуктів: «Молоко, хліб, цукор, торт, ковбаса». Це завдання можна знайти через рядок пошуку завдання. Для цього потрібно

відкрити сторінку «Пошук» і вводити опис завдання у відповідне поле, при цьому список буде автоматично оновлюватись фільтруючи завдання за текстом, який був введений. Це можна побачити, проаналізувавши моменти, коли було введено два символи слова з опису (рис. 3.16) і коли було введено чотири символи слова з опису (рис. 3.17).

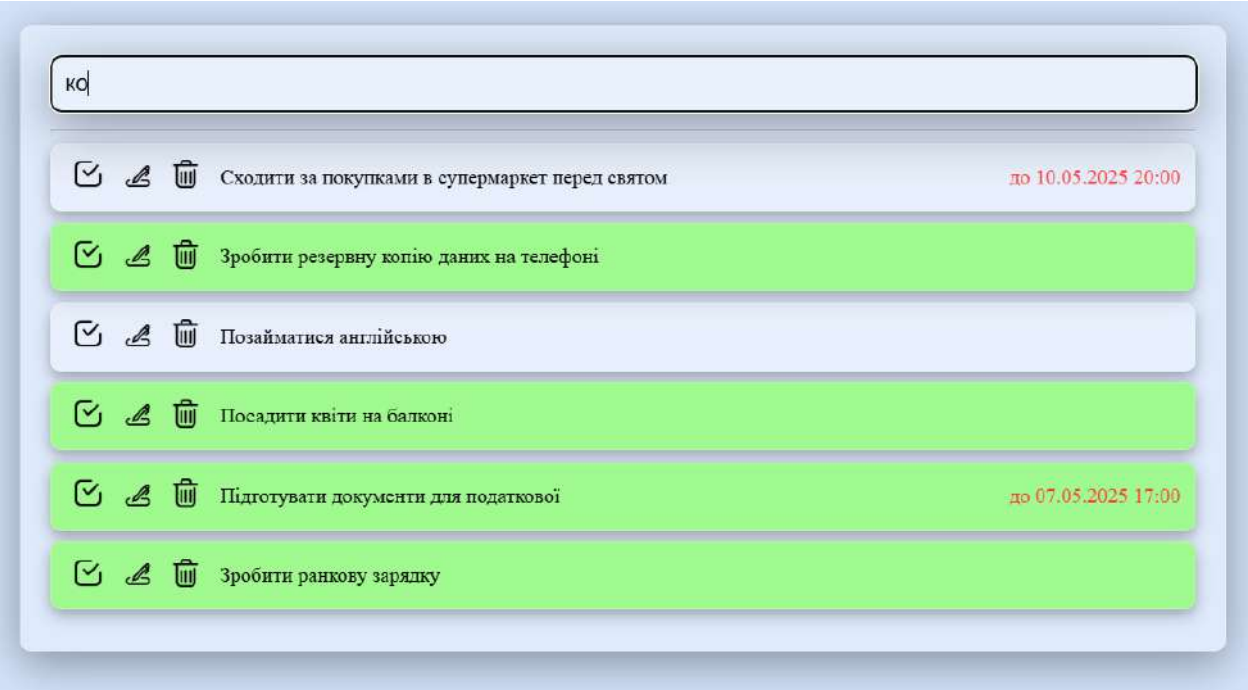

Рис. 3.16. Фільтрування списку за двома символами з опису завдання

Примітка. Джерело: розроблено автором

Наступним буде протестований сценарій пошуку завдань на певний день, у нашому сценарії ми шукатимемо завдання на 8 травня 2025 року.

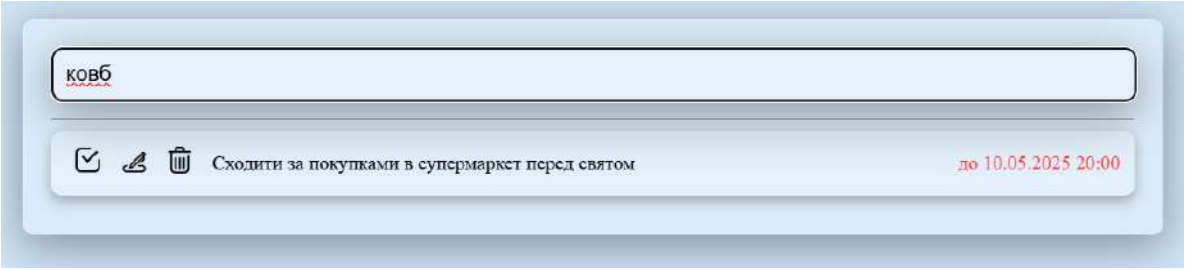

Рис. 3.17. Результат фільтрування списку за чотирма символами з опису завдання

Примітка. Джерело: розроблено автором

Для цього потрібно перейти на сторінку «Календар» і відшукати необхідний день. Якщо шуканий день знаходиться в іншому місяці, то цей місяць можна відкрити за допомогою зручних селекторів; при необхідності швидкого повернення на поточний місяць, можна використовувати відповідну кнопку-іконку (рис. 3.18).

Рис. 3.18. Кнопка та селектори для перемикання місяців та років

Примітка. Джерело: розроблено автором

Після того як знайшли необхідний день (рис. 3.19), можна побачити тільки два завдання. Щоб побачити всі завдання необхідно натиснути на мітку, що знаходиться зверху праворуч, вона дасть можливість переглянути всі завдання на цей день в зручному форматі, розбивши їх на блоки за часом (рис. 3.20).

Рис. 3.19. Кнопка календаря на певний день

Примітка. Джерело: розроблено автором

Для останнього сценарію − зміни теми оформлення веб-сервісу, необхідно лише активувати відповідну кнопку. Оскільки вона знаходиться у правій частині шапки веб-сервісу, можна сказати, що вона доступна в будь-який момент. При її активації вона змінює тему оформлення на протилежну, так само змінюється і малюнок на кнопці зміни теми. У нашому випадку можна побачити на рисунку 3.21, як веб-сервіс змінив свою тему оформлення на темну.

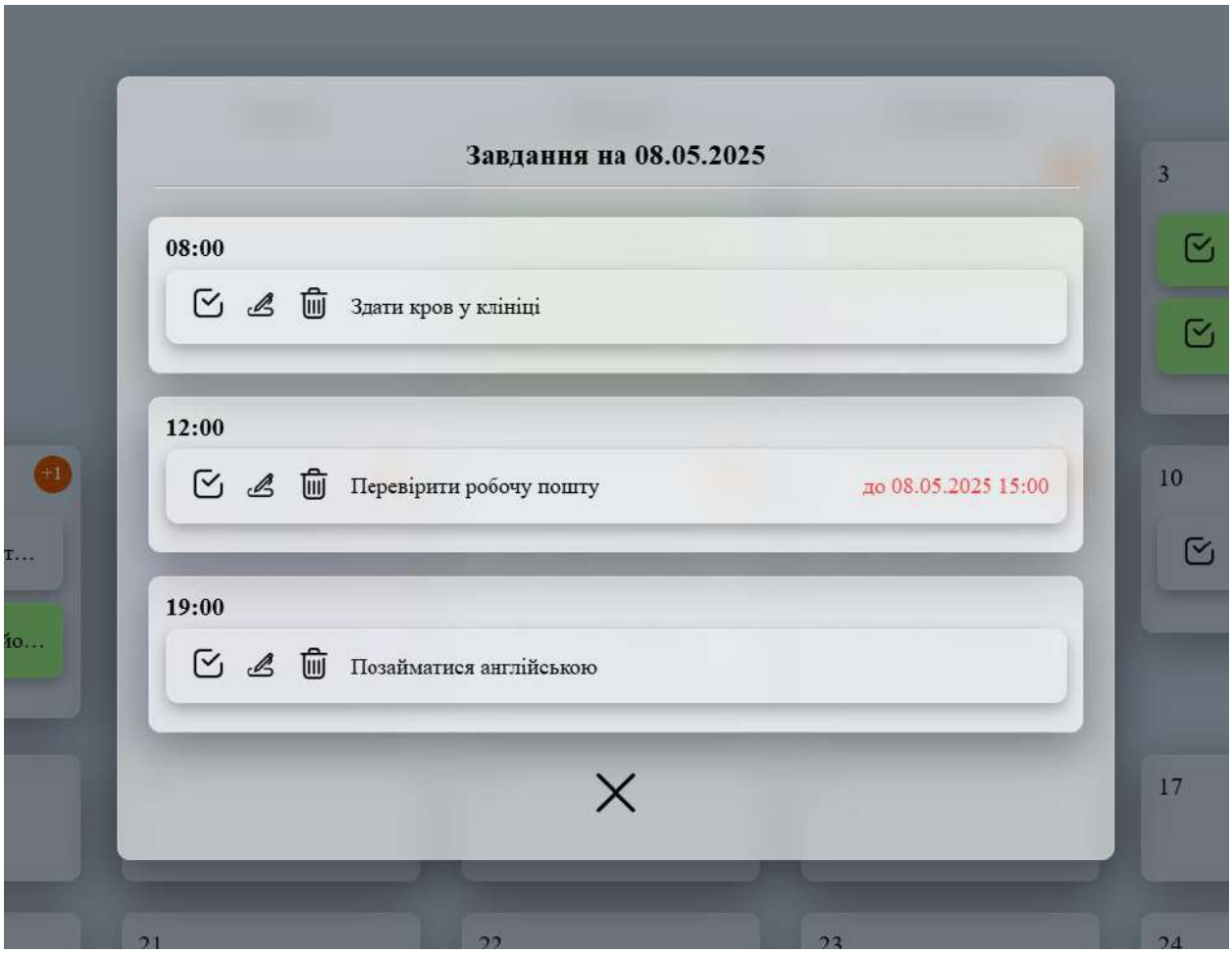

Рис. 3.20. Результат пошуку завдань на 8 травня 2025 року

Примітка. Джерело: розроблено автором

Сторінка має такі зміни:

- задній фон став переливатися темними кольорами, що змінило кольори контейнерів, оскільки вони напівпрозорі;
- текст було змінено на білий для читання завдань;

– кнопки-іконки та кнопка зміни теми стали підсвічуватись, для відокремлення їх від заднього фону.

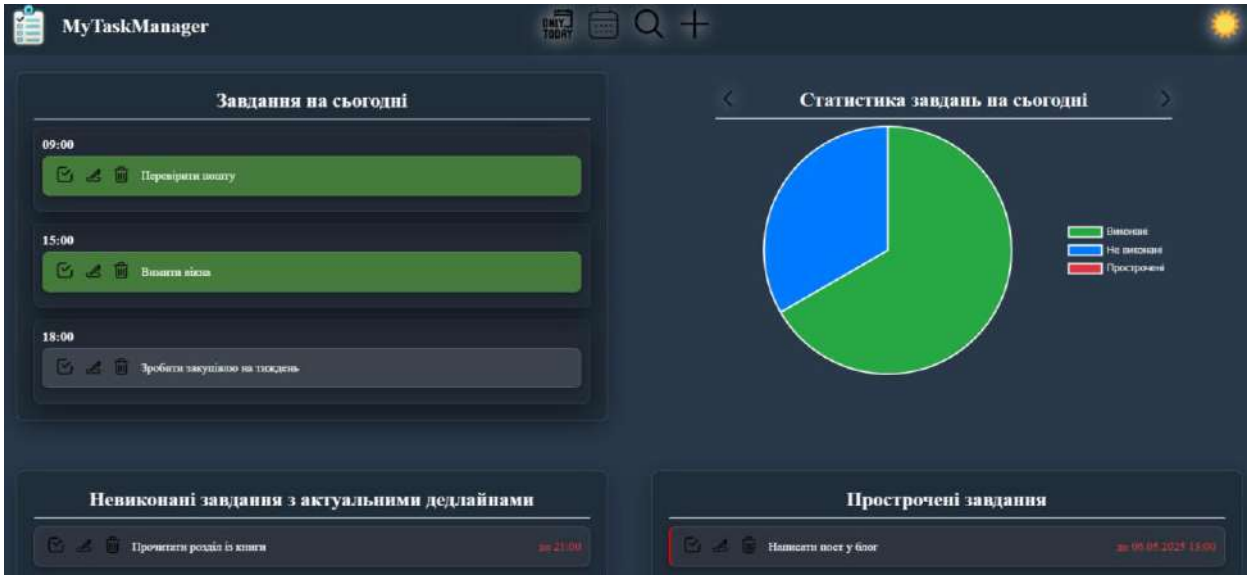

Рис. 3.21. Результат зміни теми оформлення на темну

Примітка. Джерело: розроблено автором

Можна впевнено стверджувати, що тестування веб-сервісу показало очікуваний результат, зумовлений правильною роботою сервісу при виконанні відповідних сценаріїв.

3.4 Адаптивність веб-сервісу

З метою забезпечення користувачів високим рівнем зручності та універсальністю використання, веб-сервіс персонального тайм-менеджменту MyTaskManager був створений з використанням сучасних вимог адаптивного дизайну. Це дозволяє веб-сервісу автоматично підлаштовувати інтерфейс під різні пристрої та їх роздільну здатність екранів. Наприклад, веб-сервіс MyTaskManager може підлаштовувати свій інтерфейс як для всієї ширини екрану комп'ютера, так і для частини екрана, коли його необхідно розділити. Також були не забуті користувачі на мобільних пристроях – смартфонах. Вони мають невеликі екрани в порівнянні з комп'ютерами, що робить їх особливо складними

для реалізації комфортного веб-сервісу. Для них було перероблено відображення деякого функціоналу, щоб вони були не урізані у функціоналі та комфорті використання порівняно з користувачами на комп'ютерах.

Першими будуть оглянуті зміни у відображенні веб-сервісу персонального тайм-менеджменту MyTaskManager на смартфоні. Після відкриття веб-сервісу ми потрапляємо на сторінку «Сьогодні», але так як екрани мають різні співвідношення сторін, то макет сторінки відрізняється від веб-сервісу відкритого на всю ширину екрана комп'ютера. Тепер усі частини йдуть в один ряд. Все починається з контейнера «Завдання на сьогодні» (рис. 3.22), після чого йде частина з діаграмами «Статистика завдань на сьогодні» та «Статистика завдань за місяць» (рис. 3.23) і закінчується двома контейнерами зі списками «Невиконані завдання з актуальними дедлайнами» та «Прострочені завдання».

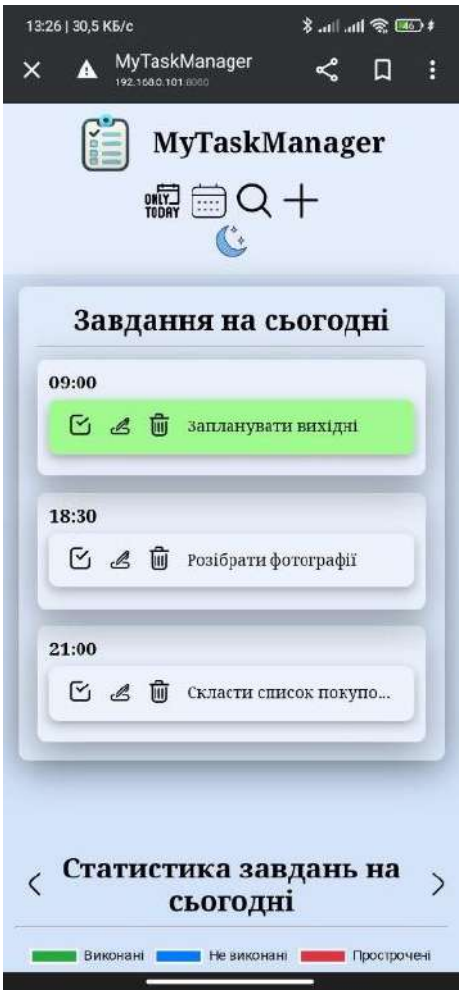

Рис. 3.22. Перша частина сторінки «Сьогодні» через смартфон

Примітка. Джерело: розроблено автором

Шапка також зазнала змін: тепер вона відображається у вигляді перевернутого трикутника. Це пов'язано з централізацією елементів у рядках, які тепер відображаються у такому порядку:

– перший рядок – логотип та назва веб-сервісу персонального тайм-менеджменту MyTaskManager;
– другий рядок – кнопки-іконки навігаційної панелі;
– третій рядок – кнопка для зміни теми оформлення.

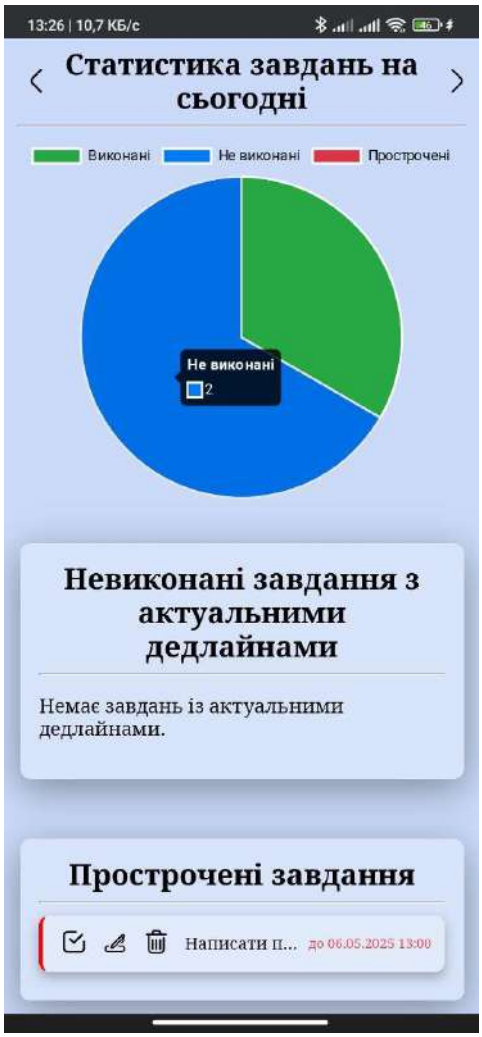

Рис. 3.23. Друга частина сторінки «Сьогодні» на смартфоні

Примітка. Джерело: розроблено автором

Наступною оглянемо сторінку «Календар», вона зазнала великих змін. Оскільки контейнери із завданнями були масивними для екранів смартфонів,

вони відображаються як кнопки (рис. 3.24). За цими кнопками можна зрозуміти наступне: якщо на ній присутня точка червоного або зеленого кольору під цифрою, яка є числом відповідного дня, то в дні присутні завдання. Якщо точка червоного кольору – це говорить про те, що у дні присутні не виконані завдання; якщо точка зеленого кольору – всі завдання виконані. Так само, як і при відкритті веб-сервісу на всю ширину екрана комп'ютера, поточний день виділяється синім кольором.

Рис. 3.24. Сторінка «Календар» на смартфоні

Примітка. Джерело: розроблено автором

При натисканні користувачем на необхідний день буде відкрито список із завданнями, що належать до відповідної дати (рис. 3.25), вони представлені як при активації мітки веб-сервісу відкритого на всій ширині екрана комп'ютера – згруповані по блокам часу виконання. У списку так само відображається

заголовок, що складається з «Завдання на» та дня, для якого були відкриті завдання у форматі «дд.мм.рррр» – це дозволяє користувачеві зрозуміти на який день були відкриті завдання, задля уникнення непередбачених ситуацій при використанні веб-сервісу зі смартфона. Наприкінці списку знаходиться кнопка-іконка, яка при натисканні здійснює закриття списку із завданнями та перенаправляє користувача назад до календаря, що дає комфортну навігацію та інтуїтивну взаємодію з інтерфейсом календаря.

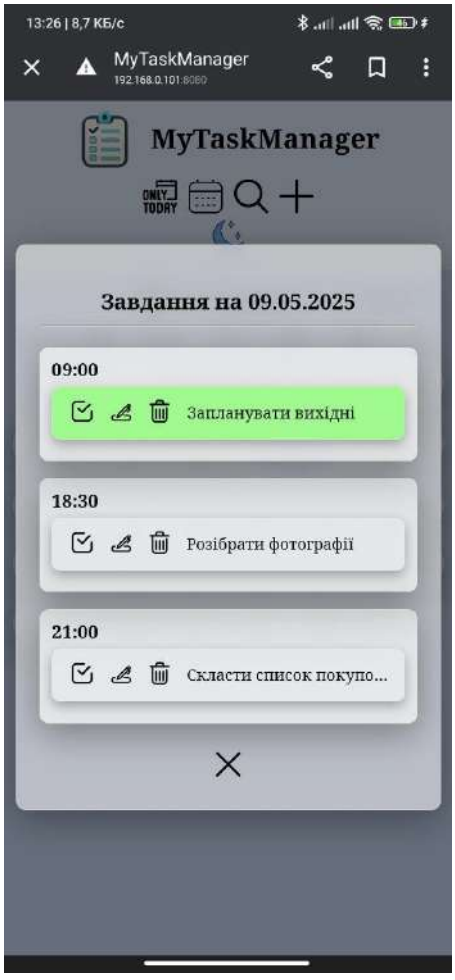

Рис. 3.25. Список завдань на відповідний день на смартфоні

Примітка. Джерело: розроблено автором

Сторінка «Пошук» була несуттєво змінена, в ній змінилося тільки одне – це контейнер, що включає поле для пошуку і список завдань, тепер він став відображатися на всю ширину екрана з мінімальними відступами.

На сторінці «Додавання нового завдання» (рис. 3.26), контейнер також став відображатися на всю ширину екрана з мінімальними відступами, а також було змінено введення даних у поля:

– дата та час виконання завдання;
– дата та час дедлайну.

Тепер введення даних у ці поля відбувається тільки через спливаючі віджети – маленький календар для дати і стовпи для часу, що прокручуються.

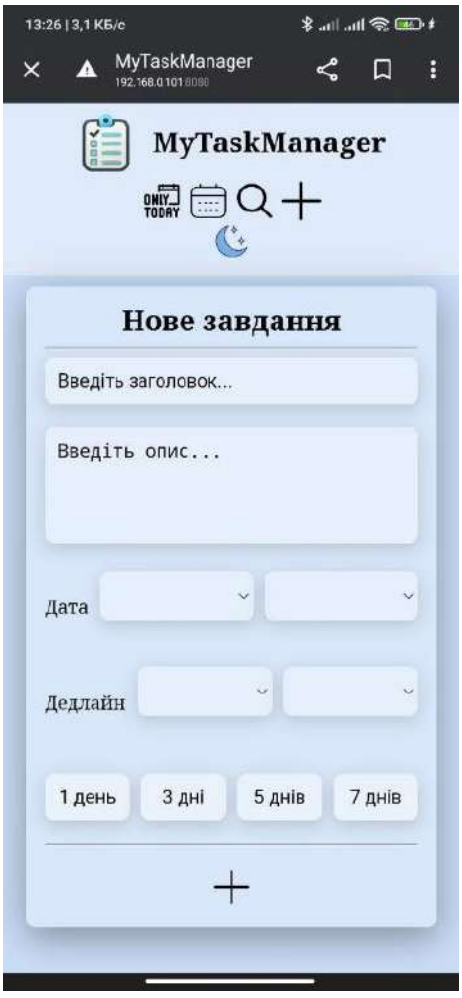

Рис. 3.26. Сторінка «Додавання нового завдання» через смартфон

Примітка. Джерело: розроблено автором

Далі буде розглянута адаптація, коли користувачеві необхідно зменшити ширину веб-сервісу, наприклад розмістити його паралельно з якоюсь додатковою програмою, або ж можна сказати простіше – на одній половині

екрана. Сторінка «Сьогодні» в такому випадку буде плавно зменшуватися до критичного моменту, коли частини сторінки будуть змушені стати в один ряд, як при використанні веб-сервісу через смартфон. Мінімальною шириною екрану можна зробити до моменту коли логотип, назва, кнопки-іконки панелі навігації та кнопка зміни теми оформлення стануть впритул один до одного.

Сторінка «Пошук» та сторінка «Додавання нового завдання» будуть зменшуватися таким чином: спочатку зменшуватиметься ширина від відповідного краю контейнера до краю екрана, при перетині мінімальної ширини починає зменшуватися ширина контейнера.

Сторінка «Календар» також зменшується до певного моменту, він настає швидше за всі сторінки, оскільки ширина кожного завдання в блоці дня швидко стає маленькою, що призводить до непорозуміння завдання. При настанні цього моменту календар набуває вигляду як у смартфоні (рис. 3.27). Це дозволяє зменшити календар до максимального мінімуму ширини веб-сервісу.

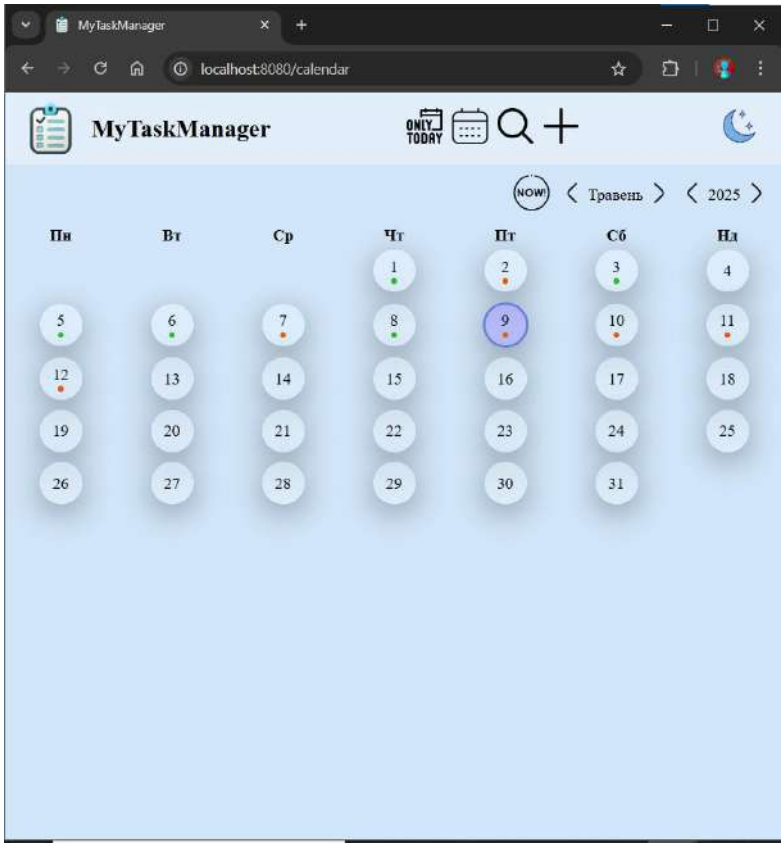

Рис. 3.27. Масштабування сторінки «Календар»

Примітка. Джерело: розроблено автором

При цьому, при натисканні на необхідний день відкриється список із завданнями на цей день (рис. 3.28), він відображатиме завдання як при активації мітки у веб-сервісі, відкритого на всю ширину екрана чи відкритого на смартфоні.

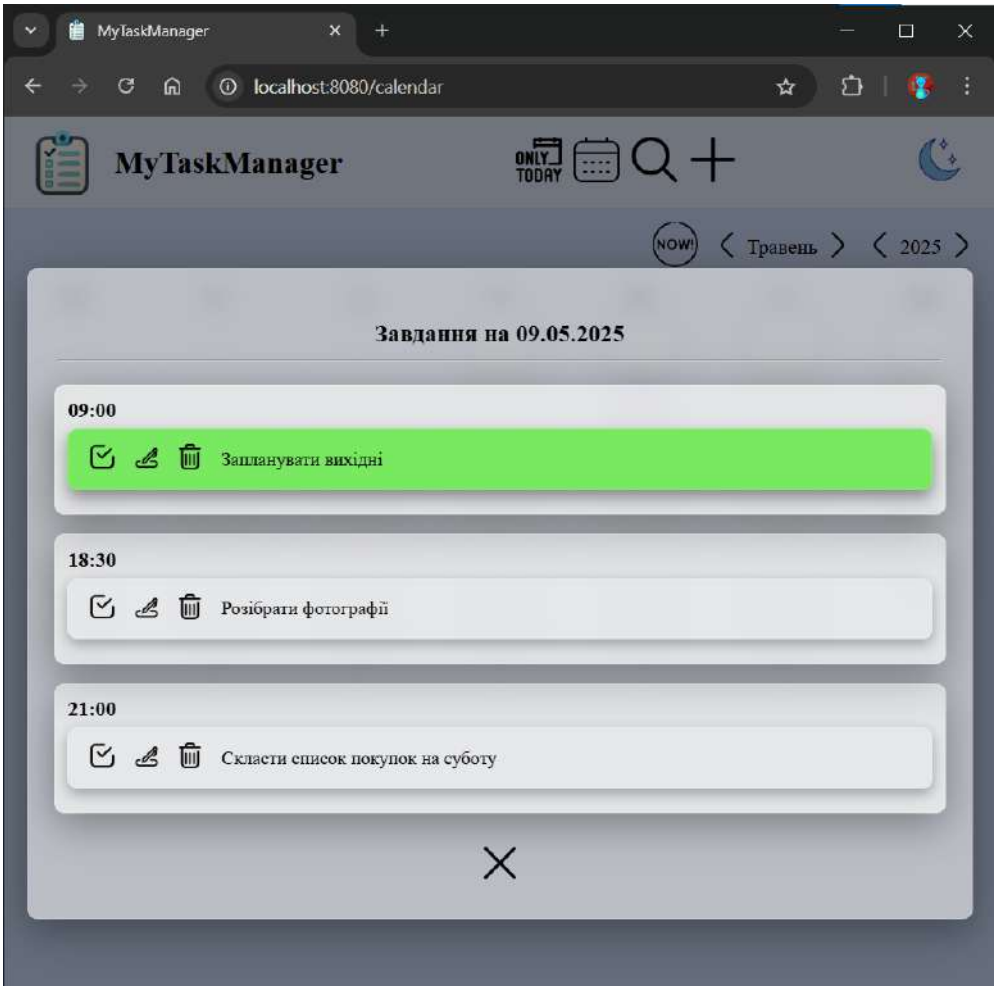

Рис. 3.28. Список завдань на відповідний день

Примітка. Джерело: розроблено автором

В результаті огляду веб-сервісу відкритого на двох пристроях: смартфоні та комп'ютері на половину екрана, адаптація показала гарний результат. Великі елементи сторінки почали відображатися в іншому вигляді, при цьому логіка роботи зазнала незначних змін. Тому можна зробити висновок, що веб-сервіс має адаптивний дизайн під різні типи пристроїв та їх розміри.

Висновки до розділу 3

У третьому розділі був описаний та протестований веб-сервіс персонального тайм-менеджменту MyTaskManager, що дозволяє керувати своїм часом – шляхом управління та аналізом своїх завдань через інтуїтивно зрозумілий інтерфейс. Було оглянуто наступні чотири сторінки:

- «Сьогодні» – дозволяє відслідковувати та аналізувати завдання на поточний день через список завдань на сьогодні, діаграму з двома режимами роботи та двома списками: завдання з актуальними дедлайнами та завдання з простроченими дедлайнами;
- «Календар» – відображає календар для простого знаходження всіх завдань на певний день;
- «Пошук» – дає можливість знайти необхідне завдання шляхом введення тексту завдання або частини його опису;
- «Додавання нового завдання» – демонструє форму для додавання нового завдання, або редагування існуючого.

Всі сторінки мають адаптивний дизайн, що дозволяє комфортно взаємодіяти з веб-сервісом як на комп'ютері, так і на мобільному пристрої. При цьому за допомогою кнопки зміни теми оформлення у шапці, можна змінити колірну схему веб-сервісу. Світла тема добре підходить для використання у світлий час доби, а темна – у темний час, що дозволяє уникнути втоми очей користувача.

Деяку кількість розроблених компонентів було проаналізовано, розбивши кожен із них на три частини:

- перша частина – це шаблон, що показує як відображатиметься компонент на сторінці;
- друга частина – це скрипт, що забезпечує функціональність компонента, реалізуючи логіку поведінки;

– третя частина – це стилі оформлення, вони не були проаналізовані, оскільки мають великі обсяги.

Проведене тестування веб-сервісу MyTaskManager показало ефективність та стабільність роботи в умовах реального використання. Всі протестовані функції сервісу показали коректну роботу, без помилок та збоїв, що свідчить про стійку працездатність веб-сервісу з необхідним рівнем функціональності для персонального тайм-менеджменту.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи був поетапно створений багатофункціональний веб-сервіс персонального тайм-менеджменту – MyTaskManager. Він охоплює теоретичне обґрунтування, моделювання процесів та практичну реалізацію.

У першому розділі був проведений глибокий аналіз предметної області, що дозволив визначити ключові вимоги до сучасних веб-сервісів для управління завданнями. На основі цього аналізу сформульовано принципи функціональності, адаптивності та інтуїтивності інтерфейсу користувача, що служать основою для розробки програмного продукту. Проведений огляд існуючих рішень, таких як Microsoft To Do, TickTick, Google Tasks, Todoist та Any.Do продемонстрував, як використання найкращих практик сучасних сервісів у поєднанні із сучасними технологіями, такими як HTML5, CSS3 та JavaScript, дозволяє створити програмний продукт, максимально адаптований до потреб користувачів. Підсумовували, що веб-сервіс персонального тайм-менеджменту повинен об'єднувати багатофункціональність, інтеграцію з іншими сервісами, адаптивність, доступність і простоту використання з різних пристроїв.

У другому розділ виконано аналіз фреймворків та детальне моделювання процесу управління завданнями. Розроблено логічні та фізичні моделі системи, що включають UML-діаграми:

- UML-діаграму прецедентів;
- UML-діаграму класів;
- дві UML-діаграми послідовностей;
- UML-діаграму компонентів;
- ER-діаграму.

Це дозволило детально описати структуру даних, механізми взаємодії елементів системи та ключові етапи обробки інформації. Використання LocalStorage як база даних забезпечило швидкість доступу до даних та незалежність від серверного сховища. Ретельне моделювання дозволило знизити

ризики помилок у процесі реалізації, підвищити ефективність інтеграції функціональних модулів та забезпечити підготовленість програмного продукту до майбутнього масштабування.

У третьому розділі представлено реалізований веб-сервіс MyTaskManager, що складається з чотирьох сторінок: «Сьогодні», «Календар», «Пошук», «Додавання нового завдання». Відповідні сторінки мають наступний функціонал:

- створення завдання з описом та можливістю додавання дедлайну;
- редагування завдань під час відстеження помилки;
- видалення неактуальних завдань;
- зміна статусу виконання завдання;
- зміна теми оформлення веб-сервісу;
- відстеження статистики виконання завдань за поточний день чи місяць;
- відстеження дедлайнів на даний момент;
- пошук усіх завдань на певний день через календар;
- пошук завдань за назвою чи описом.

Тестування веб-сервісу підтвердило його стабільність, надійність та ефективність. Реалізовані алгоритми забезпечують безперебійне виконання всіх основних сценаріїв використання. Завдяки адаптивному дизайну забезпечено зручність як на повному та частковому екрані комп'ютера, так і на мобільному пристрої.

Для подальшого покращення веб-сервісу можна виділити такі рекомендації:

- впровадити захищену базу даних для можливості користувача не прив'язуватися до певного пристрою;
- давати завданням пріоритети виконання, для глибшої організації часу;
- додати кнопку створення завдання до інтерфейсу календаря, давши користувачеві можливість відстежити план на певний день і додати завдання безпосередньо через список;

- реалізувати функцію нагадування, щоб користувачі могли своєчасно отримувати нагадування про важливі завдання або про завдання з дедлайнами, що закінчуються, збільшивши рівень організованості користувачів;
- реалізувати респонсивний дизайн.

Таким чином, під час виконання кваліфікаційної роботи було створено сучасний, багатофункціональний веб-сервіс MyTaskManager, що відповідає ключовим вимогам користувачів щодо функціональності, зручності та адаптивності. Знайомство з теоретичними знаннями, ретельне моделювання та практична реалізація – дозволило розробити інструмент, який не тільки полегшує організацію часу, а й сприяє підвищенню продуктивності, що робить його ефективним рішенням для широкої аудиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Романенко К. WEB 1.0, WEB 2.0 І WEB 3.0 – В ЧОМУ ВІДМІННОСТІ ТА ІСТОРІЯ СТВОРЕННЯ. 2024. URL: https://drukarnia.com.ua/articles/web-1-0-web-2-0-i-web-3-0-v-chomu-vidminnosti-ta-istoriya-stvorennya-MgWuT#heading-2-39 (дата звернення: 05.04.2025).
2. Ivashchuk I. Чому JavaScript – перспективна мова програмування? Поради початківцям. 2021. URL: https://dou.ua/forums/topic/35184/ (дата звернення: 05.04.2025).
3. Якість програмного забезпечення (ISO/IEC 25010). URL: https://qalight.ua/ru/baza-znaniy/kachestvo-programmnogo-obespecheniya/ (дата звернення: 05.04.2025).
4. Іськов Р. WCAG 2.1: що нового? 2018. URL: https://inclusive-it.medium.com/wcag-2-1-%D1%89%D0%BE-%D0%BD%D0%BE%D0%B2%D0%BE%D0%B3%D0%BE-d78056ff659f (дата звернення: 06.04.2025).
5. Microsoft To Do. URL: https://to-do.live.com/tasks/myday (дата звернення: 06.04.2025).
6. TickTick. URL: https://ticktick.com/webapp/#p/inbox/tasks (дата звернення: 07.04.2025).
7. Google Tasks. URL: https://tasks.google.com/tasks/ (дата звернення: 07.04.2025).
8. Todoist. URL: https://app.todoist.com/app/today (дата звернення: 08.04.2025).
9. Волошин М. Що таке Todoist: огляд одного з найпопулярніших менеджерів проектів. 2022. URL: https://apix-drive.com/ru/blog/reviews/chto-takoe-todoist-obzor (дата звернення: 08.04.2025).
10. Any.do. URL: https://app.any.do/tasks/all?referral=FP_897e631e-91fe-4374-83ce-e9e9676661fe (дата звернення: 12.04.2025).

11. Порівняння між фронтенд фреймворками: Angular, React та Vue.js. 2024. URL: https://it-rating.ua/porivnyannya-mij-frontend-freymvorkami-angular-react-ta-vuejs (дата звернення: 16.04.2025).
12. Як стати розробником React? Усе що потрібно знати. 2023. URL: https://foxminded.ua/react-developer/ (дата звернення: 17.04.2025).
13. Перші кроки в Angular: Все, що потрібно знати розробнику-початківцю. 2023. URL: https://foxminded.ua/ru/angular/ (дата звернення: 17.04.2025).
14. Vue.js як JavaScript-фреймворк. 2023. URL: https://foxminded.ua/vue-js/ (дата звернення: 18.04.2025).
15. Каграманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. 2022. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 26.04.2025).
16. Zosym M. Діаграма послідовності (Sequence Diagrams). 2022. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 30.04.2025).
17. Моралес Д. Повне розуміння діаграми компонентів UML за допомогою легкого методу. 2023. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/ (дата звернення: 05.05.2025).
18. Браун Ф. Модель діаграми зв’язків сутностей (ER) із прикладом СУБД. 2024. URL: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html (дата звернення: 10.05.2025).

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату

та розміщення в Репозитарії Університету

Я, _Овчаренко Кирило Русланович_(ПІП),

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

Розробка веб-сервісу персонального тайм-менеджменту з використанням фреймворку Vue.js ліотеки

(назва роботи повністю) виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

01.06.2025 р.		К.Р. Овчаренко
Дата	підпис	ініціали, прізвище