МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку для планування робочого часу «TaskBalance»

Виконав: студент групи 4157ст

_____ Юріков Д.А.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Юрікову Дмитру Андрійовичу _____
(Прізвище, ім'я, по батькові)
1. Тема роботи: Розробка вебзастосунку для планування робочого часу "TaskBalance" __
_____
Керівник роботи Макарова Л.М. _____
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р. _____
3. Вихідні дані по роботі: _____
_____
4. Перелік питань, що належать до розробки (найменування розділів)_____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).
5. Перелік презентаційних матеріалів Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Архітектура ПЗ, Діаграма варіантів використання, Сценарії варіантів використання, Концептуальна модель даних, Діаграма класів, Діаграми взаємодії, Логічна модель бази даних, Технічний проект ПЗ, Результати розробки, Висновки, Заключний слайд. _____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проекту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Юріков Д.А.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота спрямована на розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, яка виникає через потребу ефективного управління робочим часом в умовах сучасного інформаційного середовища.

У кваліфікаційній роботі представлені аналіз практичної задачі інженерії програмного забезпечення до вебзастосунку для планування робочого часу «TaskBalance», аналіз існуючих рішень та постановка задачі на розробку програмного забезпечення. Розроблено ескізний, технічний та робочий проекти вебзастосунку, представлено результати розробки та розділ про охорону праці.

Кваліфікаційна робота викладена на 97 сторінках друкованого тексту, містить 21 рисунок, 32 таблиці, список використаних джерел із 15 найменувань та 5 додатків.

ABSTRACT

The qualification work is aimed at solving a practical problem of software engineering, characterized by the complexity and uncertainty of conditions, which arises due to the need for effective management of working time in the conditions of the modern information environment.

The qualification work presents an analysis of a practical problem of software engineering for the web application for planning working time "TaskBalance", an analysis of existing solutions and a statement of the task for software development. A sketch, technical and working project of the web application has been developed, the development results and a section on labor protection are presented.

The qualification work is presented on 97 pages of printed text, contains 21 figures, 32 tables, a list of references of 15 names and 5 appendices.

ЗМІСТ

ВСТУП

У сучасному бізнес-середовищі рідко можна зустріти компанію чи спеціаліста, які не застосовують інструменти планування та контролю робочого часу. Технології стрімко розвиваються, а разом із ними зростає і потреба в інструментах, що дозволяють організувати робочі процеси максимально ефективно. На ринку представлені різні рішення: від простих онлайн-календарів до спеціалізованих застосунків з розширеним функціоналом, але нерідко їм бракує комплексного підходу або гнучких налаштувань під конкретні потреби користувачів [1].

Діяльність багатьох спеціалістів (офісних працівників, фрілансерів, менеджерів різних рівнів) часто ведеться в умовах багатозадачності, що вимагає чіткого розподілу часу, пріоритизації завдань і можливості швидко реагувати на зміни. Відсутність системного підходу до управління часом може призвести до втрати ефективності, затримок у проєктах, а іноді й до фінансових втрат. Тому особливо важливо впроваджувати програмні рішення, що допомагають планувати робочий день, контролювати прогрес виконання завдань і аналізувати продуктивність.

Сьогодні у сфері тайм-менеджменту існують різноманітні техніки та методи: Pomodoro [2], 52/17, 90/30, 1-3-5 [3] та інші, які довели свою результативність. Однак, аби отримати максимум користі від цих технік, потрібно мати відповідний інструмент, що дозволить їх застосовувати у зручному форматі, інтегрувати з календарними сервісами, збирати аналітику та надавати рекомендації. Саме тому виникає потреба у розробці сучасного вебзастосунку для планування робочого часу «TaskBalance», покликаного об’єднати в собі гнучкий набір інструментів управління часом, зручний інтерфейс і широкий спектр можливостей персоналізації.

Загалом, вебзастосунок «TaskBalance» покликаний розв’язати практичну задачу інженерії програмного забезпечення, що характеризується комплексністю

та невизначеністю умов а саме - надати користувачам інструмент для підвищення особистої продуктивності та раціонального розподілу робочих ресурсів. Вебзастосунок, який планується розробити, призначений для впровадження найсучасніших методів тайм-менеджменту в робочий процес, а також для збору й обробки даних про виконані завдання з метою формування зрозумілої та детальної аналітики.

Метою кваліфікаційної роботи є розробка вебзастосунку для планування робочого часу «TaskBalance».

Для досягнення поставленої мети треба виконати такі кроки:

1. Проаналізувати актуальні методики управління часом та виявити їхні переваги й недоліки в контексті розробки вебзастосунку «TaskBalance».
2. Вивчити існуючі програмні рішення, які пропонують схожий функціонал, та визначити їхні слабкі сторони, що можуть бути покращені у майбутньому продукті.
3. Виконати постановку задачі на розробку вебзастосунку для планування робочого часу «TaskBalance».
4. Створити ескізний проєкт, що включатиме опис архітектури, основних модулів та взаємодії між ними.
5. Розробити технічний проєкт із детальними вимогами до функціоналу, дизайну, безпеки та продуктивності системи.
6. Реалізувати робочий прототип, інтегрувавши календарні сервіси, інструменти тайм-менеджменту та засоби аналітики.
7. Провести тестування вебзастосунку, перевірити його стабільність і відповідність вимогам.

Об’єктом кваліфікаційної роботи є процес розробки вебзастосунку для планування робочого часу «TaskBalance».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ РОБОЧОГО ЧАСУ «TASKBALANCE»

1.1 Аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку для планування робочого часу «TaskBalance»

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку «TaskBalance» для ефективного планування робочого часу.

Зазначена задача може бути вирішена шляхом автоматизації та вдосконалення процесів планування і контролю часу за допомогою спеціалізованих технік та інтеграцій із зовнішніми сервісами (наприклад, календарями Google Calendar, Apple Calendar та іншими популярними рішеннями).

Вирішення цієї задачі дозволить користувачам ефективно управляти своїм робочим часом, а саме:

– використовувати різні методики тайм-менеджменту, такі як Pomodoro (розподіл часу на 25 хвилин роботи та 5 хвилин відпочинку) [2], 52/17 (52 хвилини праці та 17 хвилин відпочинку) [3], 90/30 (90 хвилин роботи та 30 хвилин перерви) [3], метод 1-3-5 (щоденне виконання одного великого завдання, трьох середніх та п’яти малих) та двохвилинне правило (негайне виконання будь-якого завдання, яке займає менше двох хвилин) [3];

– створювати персоналізовані стратегії, адаптуючи робочий процес під власні потреби та умови;

– синхронізувати задачі та події з популярними календарними сервісами (Google Calendar, Apple Calendar), що дозволяє завжди бачити актуальний розклад та отримувати вчасні нагадування;

– аналізувати власну продуктивність завдяки збору та статистичній обробці даних про тривалість, регулярність та якість виконання завдань;

– отримувати детальну аналітику .

Методики тайм-менеджменту забезпечують структуру робочого дня та допомагають уникнути надмірного навантаження або втрати концентрації. Наприклад, техніка Pomodoro допомагає утримувати увагу на завданні завдяки чітко визначеним часовим проміжкам роботи та відпочинку, що знижує втому та підвищує ефективність.

Інтеграція з календарями дозволяє уникнути проблем з дублюванням завдань та забуванням важливих справ. Це забезпечує єдиний центр управління всіма подіями та завданнями, спрощуючи організацію особистого та робочого часу.

Персоналізований підхід надає користувачам гнучкість у налаштуванні свого робочого процесу. Завдяки цьому можна враховувати індивідуальні особливості, пріоритети та стиль роботи, що сприяє ефективнішому досягненню поставлених цілей.

Аналітичні функції додатку дозволяють користувачам регулярно оцінювати свій прогрес, бачити слабкі та сильні сторони своєї діяльності, виявляти «вузькі місця» в організації часу та отримувати рекомендації щодо їх виправлення. Завдяки цьому користувач може покращити свою продуктивність та досягати більш вагомих результатів у роботі.

1.2 Аналіз існуючих засобів для планування робочого часу

Розглянемо та проаналізуємо основні недоліки та переваги вже існуючих на даний час вебзастосунків для планування робочого часу.

Програмний продукт «Pomofocus»

Розробник: Yuya Uzu

Цей вебзастосунок використовує техніку «Помодоро» для підвищення продуктивності, розбиваючи роботу на короткі інтервали з перервами. Він дозволяє легко відстежувати робочий час завдяки простому та зручному інтерфейсу, що сприяє концентрації та організації роботи [4].

Інтерфейс вебзастосунку «Pomofocus» представлений на рисунку 1.1.

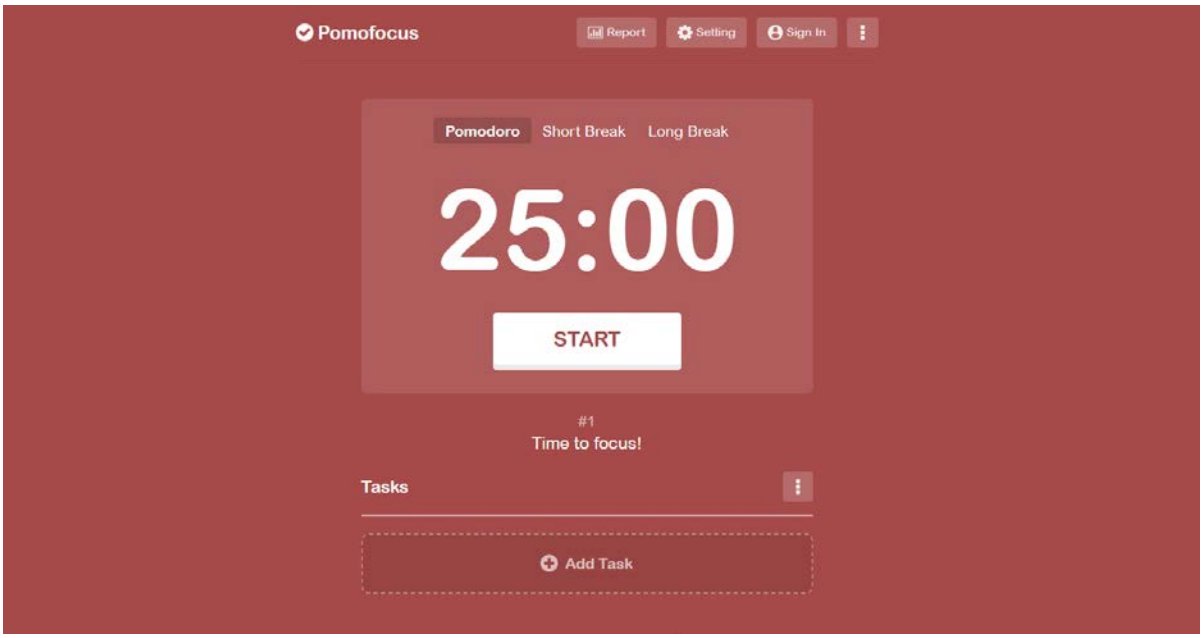

Рисунок 1.1 - Інтерфейс вебзастосунку «Pomofocus»

Основні функціональні можливості:

– реалізація таймера за методикою Pomodoro;
– налаштування інтервалів роботи та коротких/довгих перерв;
– візуальний відлік часу з інтуїтивно зрозумілим інтерфейсом.

Переваги:

– проста та зручна у використанні система;
– повністю безкоштовний доступ;
– адаптивний дизайн, що забезпечує комфортну роботу на різних пристроях.

Недоліки:

– обмежені можливості для управління завданнями (відсутність повноцінного планування);

– відсутність інтеграції з іншими сервісами та розширених аналітичних функцій.

Програмний продукт «Focus To-Do»

Розробник: Focus To-Do LLC

Цей застосунок об’єднує таймер «Помодоро» із менеджером завдань, допомагаючи планувати день та ефективно керувати часом. Focus To-Do дозволяє створювати списки справ, встановлювати пріоритети та відслідковувати прогрес, що сприяє досягненню поставлених цілей [5].

Інтерфейс вебзастосунку «Focus To-Do» представлений на рисунку 1.2.

Основні функціональні можливості:

– інтеграція таймера за методикою Pomodoro із системою управління завданнями;

– створення списків завдань, встановлення пріоритетів і дедлайнів;

– нагадування та відстеження виконання завдань;

– синхронізація даних між різними пристроями.

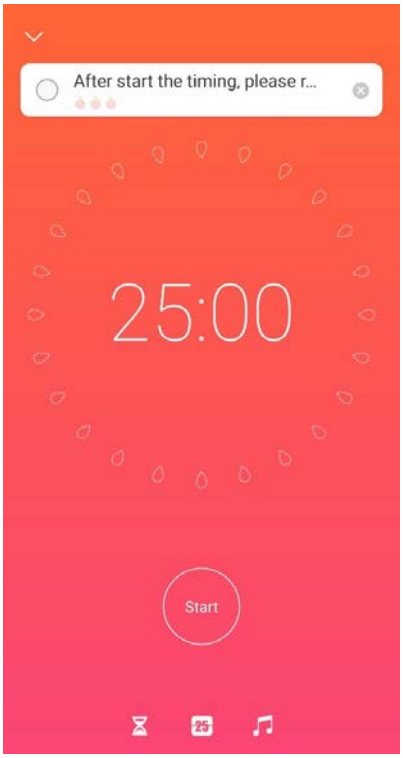

Рисунок 1.2 - Інтерфейс застосунку «Focus To-Do»

Переваги:

– комбіноване рішення, яке дозволяє ефективно поєднувати техніку Pomodoro з плануванням завдань;

– інтуїтивно зрозумілий інтерфейс і можливість персоналізації робочого процесу;

– наявність аналітичних звітів для оцінки продуктивності.

Недоліки:

– обмеження функціоналу у безкоштовній версії (деякі можливості доступні лише за підпискою);

– спостерігаються проблеми з синхронізацією даних між пристроями.

Програмний продукт «Todoist»

Розробник: Doist

Популярний додаток для управління завданнями, який дозволяє створювати проекти, додавати підзадачі та співпрацювати з іншими користувачами. Завдяки інтуїтивно зрозумілому інтерфейсу та широкому функціоналу, Todoist допомагає ефективно організувати як особисті, так і професійні справи [6].

Інтерфейс застосунку «Todoist» представлений на рисунку 1.3.

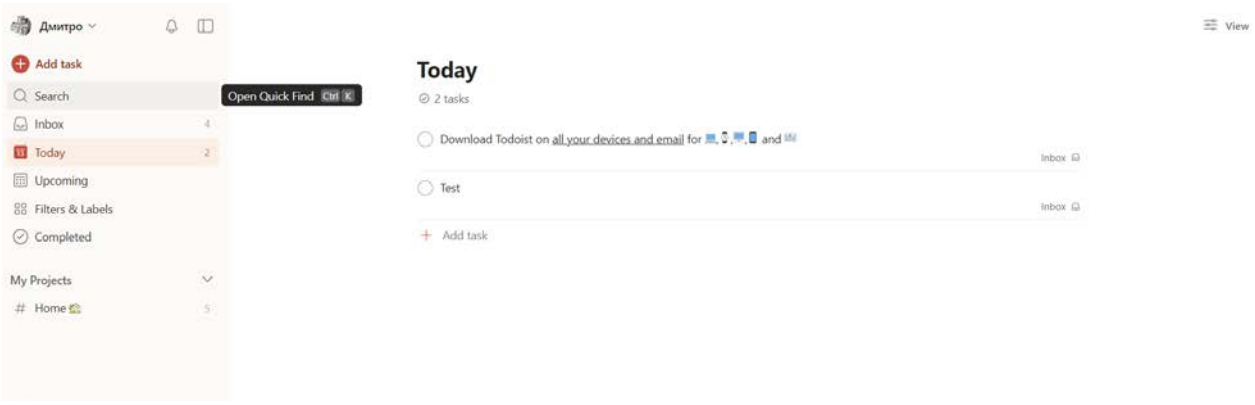

Рисунок 1.3 - Інтерфейс вебзастосунку «Todoist»

Основні функціональні можливості:

– створення та організація завдань, підзадач і проєктів;

– встановлення дедлайнів, пріоритетів та нагадувань;

– використання системи тегів та фільтрів для оптимального управління завданнями;

– інтеграція з різноманітними сервісами (Google Calendar, Zapier та ін.);

– підтримка роботи на різних платформах (веб, мобільні додатки, десктоп).

Переваги:

– чистий і інтуїтивний інтерфейс, що сприяє швидкому засвоєнню;

– потужні інструменти для організації робочого процесу;

– гнучкість і можливість масштабування для користувачів з різними потребами.

Недоліки:

– розширені функції доступні лише у платній (преміум) версії;

– відсутність спеціалізованих функцій для реалізації конкретних технік тайм-менеджменту, таких як Pomodoro.

На основі проведеного аналізу існуючих застосунків для управління часом (Pomofocus, Focus To-Do, Todoist) можна зробити висновок, що сучасний ринок пропонує рішення, які успішно вирішують окремі завдання, проте кожен з них має свої суттєві недоліки. Зокрема:

– «Pomofocus» забезпечує простоту використання та зручний інтерфейс, але обмежений лише методикою Pomodoro та не пропонує розширених функцій планування і аналізу продуктивності;

– «Focus To-Do» поєднує таймер Pomodoro із системою управління завданнями, проте функціонал значною мірою розширюється лише у платній версії, а проблеми з синхронізацією можуть заважати безперервній роботі;

– «Todoist» є потужним інструментом для організації завдань і має інтеграцію з багатьма сервісами, але відсутність спеціалізованих функцій для реалізації конкретних методик тайм-менеджменту робить його менш зручним для користувачів.

Враховуючи зазначені недоліки, розробка нового вебзастосунку«TaskBalance» є обґрунтованою з наступних причин:

– інтегрованість методик: вебзастосунок дозволить поєднати переваги різних стратегій тайм-менеджменту (Pomodoro, 52/17, 90/30, 1-3-5, двохвилинне правило), що забезпечить користувачам можливість обирати оптимальні підходи до організації робочого часу;

– розширений функціонал: реєстрація, створення особистого кабінету та збереження історії завдань дозволять користувачам аналізувати свою продуктивність, що сприятиме постійному вдосконаленню робочих процесів;

– інтеграція з календарями: синхронізація із сервісами Google Calendar, Apple Calendar та іншими допоможе зберігати актуальний розклад завдань, забезпечуючи своєчасні нагадування і уникнення пропуску важливих подій;

– аналітичні можливості: збір і обробка даних про виконання завдань дозволить користувачам отримувати детальну аналітику, що сприятиме виявленню слабких місць у плануванні робочого процесу.

Таким чином, розробка вебзастосунку«TaskBalance» є необхідною для створення комплексного інструменту управління часом, який об’єднає переваги існуючих рішень та усуне їх слабкі сторони, що в кінцевому результаті сприятиме підвищенню ефективності роботи користувачів.

1.3 Постановка задачі на розробку вебзастосунку для планування робочого часу «TaskBalance»

Виходячи з попередньо проведеного аналізу практичної задачі інженерії програмного забезпеченна, для забезпечення ефективного управління робочим часом було прийнято рішення розробити вебзастосунок «TaskBalance». Завдяки цьому інструменту користувачі зможуть створювати індивідуальні графіки, автоматизувати розподіл завдань та отримувати вчасні повідомлення про наближення строків виконання чи інші зміни в розкладі. Такий підхід сприятиме

оперативній адаптації до змін у робочому процесі, дозволяючи коригувати пріоритети та підтримувати високу продуктивність без затримок.

В рамках постановки задачі для вебзастосунку «TaskBalance» були визначені наступні функціональні вимоги:

– реєстрація користувача;
– авторизація користувача;
– можливість виходу з системи;
– додавання та редагування завдань для планування робочого часу;
– налаштування персоналізованого робочого розкладу з використанням різних методик тайм-менеджменту (наприклад, Pomodoro, 52/17, 90/30, 1-3-5, правило двох хвилин);
– інтеграція з популярними календарними сервісами (Google Calendar, Apple Calendar) для синхронізації завдань та подій;
– моніторинг виконання завдань за запитом користувача;
– автоматизований моніторинг виконання завдань за встановленим розкладом;
– оповіщення про наближення дедлайнів, завершення робочих інтервалів або зміни в розкладі;
– збереження історії оповіщень та даних про виконання завдань;
– перегляд детальної аналітики щодо продуктивності та тривалості виконання завдань.

Для забезпечення стабільної та ефективної роботи вебзастосунку для планування робочого часу «TaskBalance» сервер повинен відповідати наступним мінімальним технічним характеристикам:

– процесор: Intel Core i5-11400 з тактовою частотою 4,5 ГГц, 4 ядра;
– оперативна пам’ять: 16 Гб;
– жорсткий диск: 500 Мб вільного простору;
– інтернет-з'єднання: 1 Гбіт/с;
– операційна система: Linux.

Вимоги до організаційних вхідних та вихідних даних

- Реєстрація користувача:
  - вхідні дані: ім’я, пароль, електронна адреса;
  - вихідні дані: створення нового запису в базі даних користувачів.
- Авторизація користувача:
  - вхідні дані: електронна адреса, пароль;
  - вихідні дані: генерація унікального ідентифікатора сесії.
- Додавання нового завдання:
  - вхідні дані: назва завдання, опис, дедлайн, категорія, пріоритет;
  - вихідні дані: збереження інформації про завдання у базі даних.
- Моніторинг стану завдань:
  - вхідні дані: дані про завдання, збережені в базі (статус виконання, залишковий час до дедлайну);
  - вихідні дані: оновлення статусу завдання, відображення статистики виконання та збереження актуальних даних у системі.
- Оповіщення про зміни в завданнях:
  - вхідні дані: електронна адреса користувача, ідентифікатор завдання, інформація про зміни (зміна дедлайну чи пріоритету);
  - вихідні дані: надсилання сповіщення (електронного або внутрішнього) з актуальною інформацією.
- Профіль користувача:
  - вхідні дані: додаткові дані, такі як номер телефону, фотографія профілю;
  - вихідні дані: розширений запис у профілі користувача для подальшої персоналізації.
- Аналітика продуктивності:
  - вхідні дані: дані про виконання завдань (час виконання, затримки, відхилення від плану);

– вихідні дані: зведені звіти та діаграми, що допомагають оцінити ефективність розкладу.

– Налаштування сповіщень:

– вхідні дані: вибір користувача щодо типу сповіщень (SMS, електронна пошта) та їх частоти;

– вихідні дані: збереження індивідуальних налаштувань сповіщень у системі.

Детальна постановка задачі представлена у технічному завдані, яке наведено у додатку А.

2 РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ РОБОЧОГО ЧАСУ «TASKBALANCE»

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Побудова моделі варіантів використання

Діаграма варіантів використання (Use Case Diagram) - це один із типів UML-діаграм, що дає змогу у наочній формі представити вимоги до розроблюваного програмного забезпечення. Вона є початковою концептуальною моделлю системи, яку проєктують, та містить три основні складові: варіанти використання (прецеденти), акторів (дійових осіб) і відносини між ними.

Варіант використання відображає функціональні можливості або сервіси, які система пропонує актору. Відносини слугують семантичними зв’язками між елементами діаграми. Один актор може звертатися одразу до кількох варіантів використання, якщо йому потрібні різні сервіси. У свою чергу, один і той самий варіант використання може обслуговувати різних акторів, надаючи кожному з них необхідний функціонал [7].

До базових типів відносин у такій діаграмі належать асоціації, узагальнення та залежності, серед яких виділяють включення (include) та розширення (extend).

На рисунку 2.1 наведено діаграму варіантів використання для вебзастосунку «TaskBalance», що демонструє взаємодію акторів (користувачів) із різними прецедентами (функціями системи).

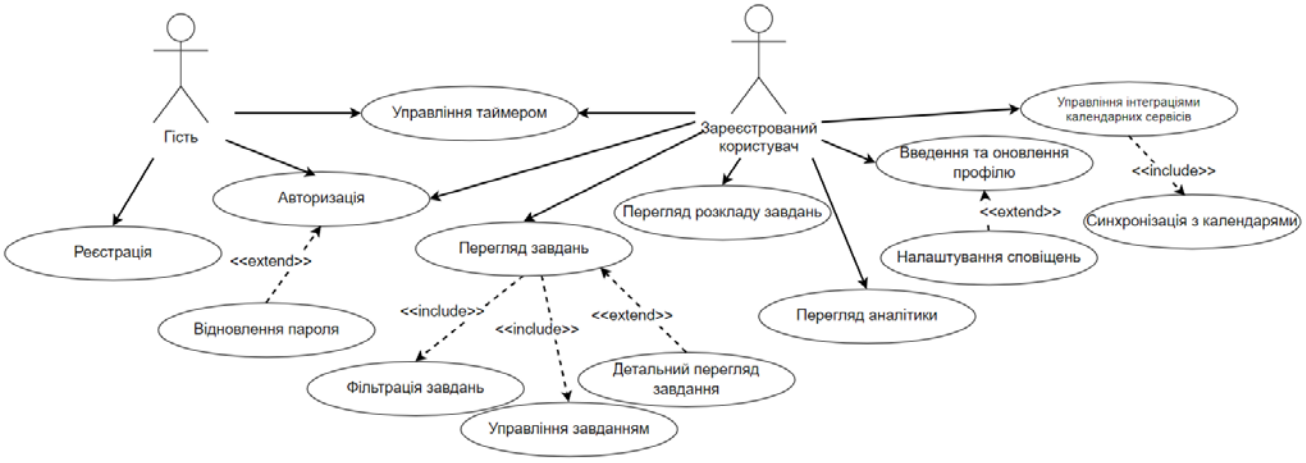

Рисунок 2.1 - Діаграма варіантів використання вебзастосунку «TaskBalance»

Після детального вивчення діаграми варіантів використання розглянемо кожен із них окремо, надавши розгорнутий опис. У таблиці 2.1 наведено повний перелік варіантів використання.

Таблиця 2.1 - Варіанти використання та їх формулювання

Назва	Опис
1	2
Реєстрація	Дозволяє гостю створити новий обліковий запис, вказавши необхідні дані (ім'я, email, пароль).
Авторизація	Дозволяє користувачу увійти до системи, надавши правильну електронну адресу та пароль.
Відновлення пароля	Сценарій розширює процес авторизації, дозволяючи відновити пароль, якщо користувач його забув.
Управління таймером	Дозволяє запускати та зупиняти таймер для відстеження робочого часу як гостю, так і зареєстрованому користувачу.
Перегляд завдань	Дозволяє зареєстрованому користувачу переглядати список наявних завдань.
Фільтрація завдань	Є складовою (включенням) «Перегляду завдань»; надає можливість відбирати завдання за певними критеріями.
Детальний перегляд завдання	Розширює «Перегляд завдань», даючи змогу побачити розширену інформацію про конкретне завдання.
Перегляд розкладу завдань	Відображає завдання у вигляді календаря чи часової шкали, дозволяючи планувати робочий час.
Введення та оновлення профілю	Дозволяє зареєстрованому користувачу змінювати особисті дані та налаштування у своєму профілі.
Управління інтеграціями календарних сервісів	Дозволяє підключати сторонні календарі (наприклад, Google Calendar), щоб автоматизувати робочі процеси.

Продовження таблиці 2.1

1	2
Синхронізація з календарями	Є складовою (включенням) «Управління інтеграціями», забезпечуючи обмін даними між застосунком і зовнішніми календарями.
Перегляд аналітики	Дозволяє зареєстрованому користувачу отримувати статистичні дані про виконані завдання та продуктивність (загальний відсоток виконання, графіки тощо).
Налаштування сповіщень	Дає можливість керувати параметрами оповіщень (email), які повідомляють про наближення дедлайнів, зміни в розкладі або завершення робочих інтервалів.
Управління завданнями	Дозволяє зареєстрованому користувачу додавати нові завдання, редагувати існуючі та видаляти непотрібні завдання.

Як видно з таблиці 2.1, реалізовані варіанти використання охоплюють увесь життєвий цикл взаємодії користувача з системою.

2.1.2 Специфікації варіантів використання

Нижче наведено приклади таблиць специфікацій для кожного варіанта використання (прецеденту) згідно з діаграмою варіантів використання вебзастосунку «TaskBalance». Специфікації прецедентів наведено у таблицях 2.2 - 2.15.

Таблиця 2.2 - Прецедент використання «Реєстрація»

Складова	Опис
Ідентифікатор	1
Короткий опис	Дозволяє гостю створити новий обліковий запис, указавши необхідні дані (ім'я, email, пароль).
Головний актор	Гість
Другорядні актори	Немає
Базовий потік	1. Користувач натискає кнопку «Реєстрація».
	2. Користувач вводить дані для реєстрації.
	3. Система зберігає дані та створює нового користувача.
	4. Система автоматично авторизує новоствореного користувача.
Передумови	Користувач не повинен бути авторизованим.
Постумови	Новий користувач стає авторизованим і отримує доступ до функцій системи.
Альтернативний потік	1. Користувач з введеною електронною адресою вже існує.
	2. Система виводить повідомлення про помилку й не створює дубльованого користувача.

Таблиця 2.3 - Прецедент використання «Авторизація»

Складова	Опис
Ідентифікатор	2
Короткий опис	Дозволяє користувачу увійти до системи, вказавши коректну електронну адресу та пароль.
Головний актор	Гість (або будь-який користувач, що ще не авторизувався)
Другорядні актори	Немає
Базовий потік	1. Користувач натискає кнопку «Увійти».
	2. Користувач вводить email та пароль.
	3. Система перевіряє достовірність даних.
	4. У разі успіху користувач авторизується й отримує доступ до системи.
Передумови	Користувач не повинен бути авторизованим.
Постумови	Користувач отримує статус «Зареєстрований» і доступ до функцій системи.
Альтернативний потік	1. Email або пароль введені некоректно.
	2. Система виводить повідомлення про помилку та пропонує повторити спробу.

Таблиця 2.4 - Прецедент використання «Відновлення пароля»

Складова	Опис
Ідентифікатор	3
Короткий опис	Дозволяє користувачу відновити пароль, якщо він його забув.
Головний актор	Гість (або користувач, що не може увійти)
Другорядні актори	Немає
Базовий потік	1. Користувач переходить до функції відновлення пароля.
	2. Вводить email, з яким було зареєстровано акаунт.
	3. Система надсилає код для відновлення пароля на вказану адресу.
	4. Користувач вводить код для відновлення і встановлює новий пароль.
Передумови	Користувач має дійсний обліковий запис, але не пам’ятає пароль.
Постумови	Пароль успішно змінено, користувач може авторизуватися з новим паролем.
Альтернативний потік	1. Введено email, що не зареєстрований у системі.
	2. Система виводить повідомлення про відсутність такого користувача.

Таблиця 2.5 - Прецедент використання «Управління таймером»

Складова	Опис
1	2
Ідентифікатор	4
Короткий опис	Дозволяє запускати, зупиняти або ставити на паузу таймер для відстеження робочого часу.
Головний актор	Гість / Зареєстрований користувач
Другорядні актори	Немає

Продовження таблиці 2.5.

1	2
Базовий потік	1. Користувач натискає кнопку «Запустити таймер».
	2. Система починає відлік часу.
	3. Користувач може зупинити або поставити на паузу таймер у будь-який момент.
	4. Система фіксує загальну тривалість відслідкованого часу (для зареєстрованого користувача зберігає в базі).
Передумови	Таймер не запущений або перебуває у стані очікування.
Постумови	Відстежений період часу збережено (для зареєстрованого користувача) або зафіксовано у тимчасовому режимі (для гостя).
Альтернативний потік	1. Користувач намагається запустити таймер, що вже активний.
	2. Система відображає повідомлення про те, що таймер вже працює.

Таблиця 2.6 - Прецедент використання «Перегляд завдань»

Складова	Опис
Ідентифікатор	5
Короткий опис	Дозволяє зареєстрованому користувачу переглядати список своїх завдань.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач обирає розділ «Завдання».
	2. Система завантажує список завдань із бази даних, відображаючи їх статус, дедлайн та інші дані.
	3. Користувач переглядає список.
Передумови	Користувач повинен бути авторизованим.
Постумови	Система відображає актуальний перелік завдань.
Альтернативний потік	1. У користувача немає завдань у системі.
	2. Система відображає повідомлення про відсутність завдань.

Таблиця 2.7 - Прецедент використання «Фільтрація завдань»

Складова	Опис
1	2
Ідентифікатор	6
Короткий опис	Дозволяє зареєстрованому користувачу відбирати завдання за певними критеріями (статус, пріоритет, дата).
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач відкриває список завдань.
	2. Користувач задає критерії фільтрації.
	3. Система відображає перелік завдань, що відповідають вибраним фільтрам.
Передумови	Користувач має доступ до списку завдань (тобто авторизований).
Постумови	Користувач бачить лише завдання, що відповідають встановленим критеріям.

Продовження таблиці 2.7.

1	2
Альтернативний потік	1. Задані критерії фільтрації, під які не підпадає жодне завдання.
	2. Система відображає повідомлення про відсутність результатів.

Таблиця 2.8 - Прецедент використання «Детальний перегляд завдання»

Складова	Опис
Ідентифікатор	7
Короткий опис	Дозволяє переглянути повну інформацію про вибране завдання: опис, дедлайн, історію змін.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач обирає конкретне завдання зі списку.
	2. Система відображає повну інформацію про завдання, включно з коментарями, файлами або історією змін (якщо передбачено).
Передумови	Користувач авторизований і має принаймні одне завдання.
Постумови	Користувач отримує детальну інформацію про вибране завдання.
Альтернативний потік	1. Завдання не знайдене або видалене.
	2. Система виводить повідомлення про неможливість відобразити деталі.

Таблиця 2.9 - Прецедент використання «Перегляд розкладу завдань»

Складова	Опис
Ідентифікатор	8
Короткий опис	Дозволяє зареєстрованому користувачу переглянути свої завдання у форматі календаря.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач переходить до розділу «Розклад завдань».
	2. Система відображає завдання з урахуванням їх дат та дедлайнів у вигляді календаря.
	3. Користувач може переглянути деталі кожного завдання.
Передумови	Користувач авторизований і має заплановані завдання.
Постумови	Користувач бачить повну картину своїх завдань у хронологічному порядку.
Альтернативний потік	1. У користувача немає запланованих завдань.
	2. Система відображає порожній календар і повідомлення про відсутність завдань.

Таблиця 2.10 - Прецедент використання «Введення та оновлення профілю»

Складова	Опис
1	2
Ідентифікатор	9
Короткий опис	Дозволяє зареєстрованому користувачу змінювати свої особисті дані (ім’я, контактні дані, налаштування тайм-менеджменту).

Продовження таблиці 2.10.

1	2
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач переходить до налаштувань профілю.
	2. Користувач змінює потрібні дані (ім’я, контакт, параметри тайм-менеджменту).
	3. Система зберігає зміни в базі даних.
Передумови	Користувач авторизований.
Постумови	Дані профілю оновлені, збережені та відображаються при наступному вході.
Альтернативний потік	1. Користувач вводить некоректні дані (неправильний формат email).
	2. Система виводить повідомлення про помилку й не зберігає зміни.

Таблиця 2.11 - Прецедент використання «Управління інтеграціями календарних сервісів»

Складова	Опис
Ідентифікатор	10
Короткий опис	Дозволяє зареєстрованому користувачу підключити зовнішні календарні сервіси (Google Calendar, Apple Calendar) і налаштувати синхронізацію.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач відкриває розділ «Інтеграції».
	2. Користувач обирає потрібний сервіс і вводить дані (ключ API).
	3. Система зберігає налаштування інтеграції та виконує синхронізацію (див. Таблиця 2.12).
Передумови	Користувач авторизований і має відповідні дані для підключення сервісу.
Постумови	Зовнішній календар синхронізується із системою «TaskBalance».
Альтернативний потік	1. Введено некоректні налаштування або недійсний ключ API.
	2. Система виводить повідомлення про помилку та пропонує повторити спробу.

Таблиця 2.12 - Прецедент використання «Синхронізація з календарями»

Складова	Опис
1	2
Ідентифікатор	11
Короткий опис	Дозволяє імпортувати завдання з підключених календарних сервісів і експортувати заплановані завдання в зовнішній календар.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає

Продовження таблиці 2.12

1	2
Базовий потік	1. Користувач ініціює синхронізацію або вона виконується автоматично за розкладом.
	2. Система отримує дані з зовнішнього календаря та оновлює власний розклад завдань.
Передумови	Успішне налаштування інтеграції (див. Таблиця 2.11).
Постумови	Дані у «TaskBalance» і зовнішньому календарі синхронізовані.
Альтернативний потік	1. Недійсний токен або проблеми з підключенням до сервісу.
	2. Система відображає помилку й припиняє синхронізацію.

Таблиця 2.13 - Прецедент використання «Перегляд аналітики»

Складова	Опис
Ідентифікатор	12
Короткий опис	Дозволяє зареєстрованому користувачу отримувати статистичні дані про виконані завдання та продуктивність (загальний відсоток виконання, графіки).
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач переходить до розділу «Аналітика» в меню або на панелі навігації.
	2. Система завантажує збережені дані про виконані завдання та робочий час із бази даних.
	3. Система обробляє ці дані та відображає загальний відсоток виконання завдань, статистику за обраний період (день, тиждень, місяць) та графік продуктивності.
	4. Користувач переглядає результати та може змінювати період аналізу.
Передумови	Користувач має успішно авторизуватися в системі та мати хоча б одне виконане завдання.
Постумови	Користувач отримує огляд своєї продуктивності за вибраний період.
Альтернативний потік	1. Якщо користувач не має жодних виконаних завдань, система виводить повідомлення про відсутність даних для аналізу.
	2. Якщо виникла помилка при завантаженні даних (відсутній зв’язок із базою даних), система показує повідомлення про неможливість відобразити аналітику.

Таблиця 2.14 - Прецедент використання «Налаштування сповіщень»

Складова	Опис
1	2
Ідентифікатор	13
Короткий опис	Дозволяє зареєстрованому користувачу керувати параметрами оповіщень, які надсилаються про наближення дедлайнів, зміни в розкладі або завершення робочих інтервалів.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає

Продовження таблиці 2.14.

1	2
Базовий потік	1. Користувач переходить до розділу «Налаштування сповіщень» у профілі.
	2. Система відображає поточні параметри сповіщень (вид сповіщень, частота).
	3. Користувач змінює налаштування (вмикає/вимикає сповіщення, обирає час надсилання email).
	4. Система зберігає нові налаштування в базі даних та застосовує їх у майбутніх оповіщеннях.
Передумови	Користувач авторизований та має доступ до налаштувань свого акаунту.
Постумови	Нові налаштування сповіщень збережені й активовані для користувача.
Альтернативний потік	1. Якщо користувач вказав некоректні дані (неправильний формат email), система показує повідомлення про помилку й не зберігає зміни.
	2. Якщо виникла помилка під час збереження (збій з'єднання з базою даних), система повідомляє про це й пропонує повторити спробу.

Таблиця 2.15 - Прецедент використання «Управління завданнями»

Складова	Опис
Ідентифікатор	14
Короткий опис	Дозволяє зареєстрованому користувачу створювати нові завдання, редагувати існуючі та видаляти непотрібні завдання.
Головний актор	Зареєстрований користувач
Другорядні актори	Немає
Базовий потік	1. Користувач переходить до розділу «Завдання».
	2. Користувач обирає дію: створити нове завдання, редагувати або видалити існуюче.
	3. Система обробляє дію (створює, оновлює або видаляє запис) та зберігає зміни у базі даних.
	4. Система відображає оновлений список завдань.
Передумови	Користувач повинен бути авторизованим.
Постумови	Список завдань у «TaskBalance» змінено відповідно до виконаних дій (нові завдання створено, редагування внесено, непотрібні завдання видалено).
Альтернативний потік	Немає

Таким чином, маючи на руках детальні специфікації варіантів використання вебзастосунку «TaskBalance», можна переходити до наступних етапів проєктування та реалізації цього програмного забезпечення.

2.2 Архітектура вебзастосунку

Архітектура програмного забезпечення - це структурний план системи, який визначає, як компоненти взаємодіють між собою та з зовнішнім середовищем. Вона базується на архітектурних шаблонах, що допомагають застосовувати перевірені рішення для типових завдань. Архітектуру можна уявити як план будинку: лише замість цегли та бетону тут використовуються код, бази даних і сервери - усе чітко організовано для забезпечення надійної та ефективної роботи системи [8].

2.2.1 Розробка діаграми компонентів вебзастосунку для планування часу «TaskBalance»

У процесі розробки вебзастосунку TaskBalance було обрано трьохрівневу клієнт-серверну архітектуру, яка передбачає розділення системи на:

– рівень представлення - інтерфейс користувача, що працює у веббраузері;

– рівень логіки застосунку - серверна частина, яка обробляє запити та бізнес-процеси;

– рівень зберігання - база даних, що відповідає за збереження користувацької інформації та задач.

Такий підхід забезпечує зручну розробку, просте тестування, надійний доступ до даних та гнучке масштабування.

Переваги обраної архітектури:

– чітке розділення функцій - кожен рівень працює незалежно, що полегшує розробку, підтримку та внесення змін;

– масштабованість - усі частини системи можуть масштабуватись окремо, у тому числі з використанням хмарних технологій;

– гнучкість - нові функції додаються без необхідності втручання в усю систему;

– безпека - централізована обробка даних дозволяє контролювати доступ і зменшує ризик витоку інформації;

– мобільність - інтерфейс доступний з будь-якого пристрою через браузер, що робить TaskBalance зручним для користувачів у різних умовах.

Діаграма архітектури вебзастосунку для планування часу TaskBalance представлено на рисунку 2.2.

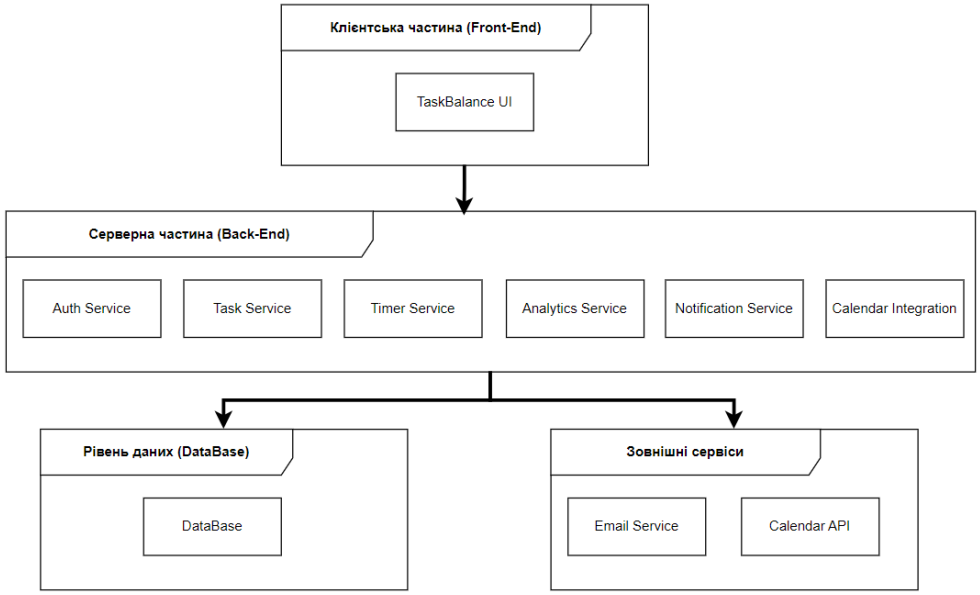

Рисунок 2.2 - Діаграма архітектури вебзастосунку для планування часу «TaskBalance»

У вебзастосунку TaskBalance графічний інтерфейс користувача (View) реалізовано за допомогою сучасних вебтехнологій (HTML, CSS, JavaScript), які обробляються у браузері користувача. За взаємодію з цим інтерфейсом відповідає Controller, який реагує на дії користувача (наприклад, натискання кнопок, введення даних) та передає запити до відповідних сервісів (Service) на сервері.

Сервіси реалізують бізнес-логіку застосунку, обробляють дані, виконують валідацію та координують роботу з внутрішніми об’єктами - моделями (Model). Моделі описують основні сутності (користувачі, завдання, таймери тощо) та забезпечують логіку їхньої обробки.

Моделі взаємодіють з репозиторіями (Repository), які надають стандартні

методи доступу до бази даних. Репозиторії відповідають за зчитування, запис, оновлення та видалення інформації, що зберігається у відповідних таблицях.

Таким чином, взаємодія між компонентами вебзастосунку TaskBalance організована за принципами архітектурного шаблону MVC (Model-View-Controller), що дозволяє досягти чіткого розділення обов'язків, покращує підтримку та розширюваність системи. Графічне представлення діаграми взаємодії компонентів вебзастосунку для планування часу TaskBalance зображено на рисунку 2.3.

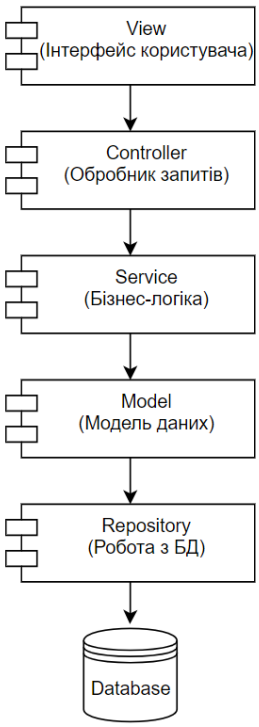

Рисунок 2.3 - Діаграма взаємодії компонентів вебзастосунку «TaskBalance»

Дана діаграма демонструє логічні зв'язки між ключовими елементами системи, а також відображає, яким чином забезпечується обробка даних і виконання функціональних запитів. Це дозволяє глибше зрозуміти організацію внутрішньої архітектури TaskBalance.

2.2.2 Розробка діаграми розміщення вебзастосунку для планування часу «TaskBalance»

Для наочного представлення фізичної структури програмного забезпечення доцільно використати діаграму розміщення (рисунок 2.4). Це один із типів UML-діаграм, який дозволяє відобразити фізичну архітектуру системи - програмні й апаратні компоненти, їхнє розташування та взаємодію.

Діаграма розміщення показує, як у системі пов’язані між собою сервери, клієнтські пристрої, бази даних, додатки й мережі, а також які залежності існують між цими елементами. Такий підхід дає змогу зрозуміти, як компоненти розгорнуті фізично, і краще уявити реальну інфраструктуру програмного забезпечення.

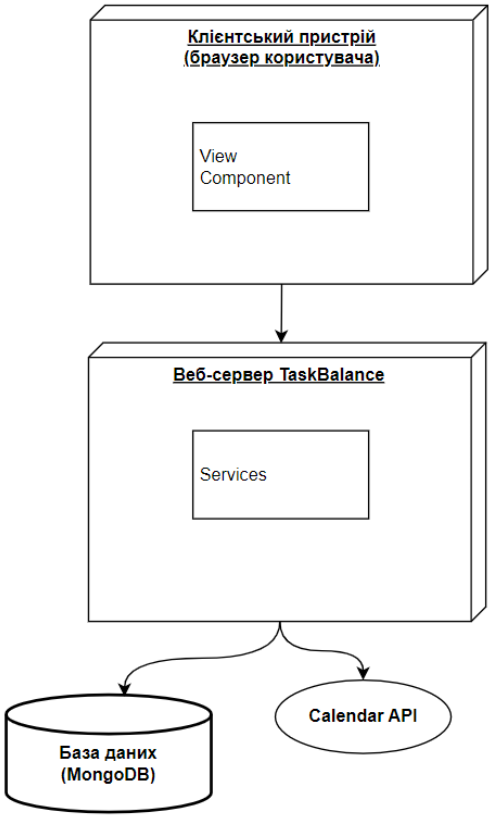

Рисунок 2.4 - Діаграма розміщення вебзастосунку для планування часу «TaskBalance»

На діаграмі розміщення видно, що клієнтський інтерфейс TaskBalance розміщується у веббраузері користувача на персональному комп’ютері або мобільному пристрої, які взаємодіють із вебсервером через інтернет. Вебсервер приймає запити, обробляє їх і звертається до бази даних, що розташована на окремому сервері з відповідною системою управління даними MongoDB.

2.3 Проект вебзастосунку для планування робочого часу «TaskBalance»

2.3.1 Ескізний проект вебзастосунку

Сценарії основних варіантів використання демонструють, що кожен об'єкт вебзастосунку «TaskBalance» має власну поведінку, перебуває в певних станах і може переходити з одного стану в інший, виконуючи конкретні дії під час цих переходів. Така поведінка об'єкта відображається через діаграми станів та діаграми діяльності, що дозволяють графічно представити логіку роботи системи.

Сценарії основних варіантів використання зображені на рисунках 2.5-2.7.

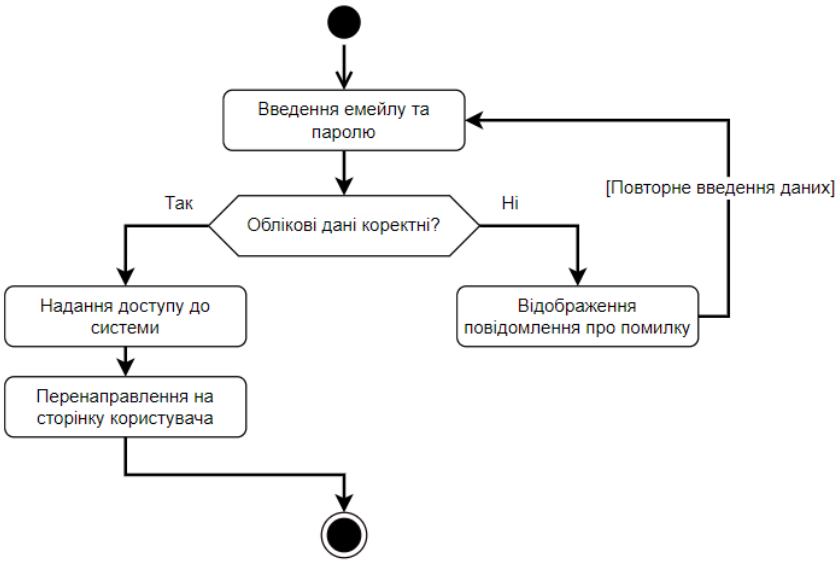

Рисунок 2.5 - Діаграма діяльності для прецеденту «Авторизація»

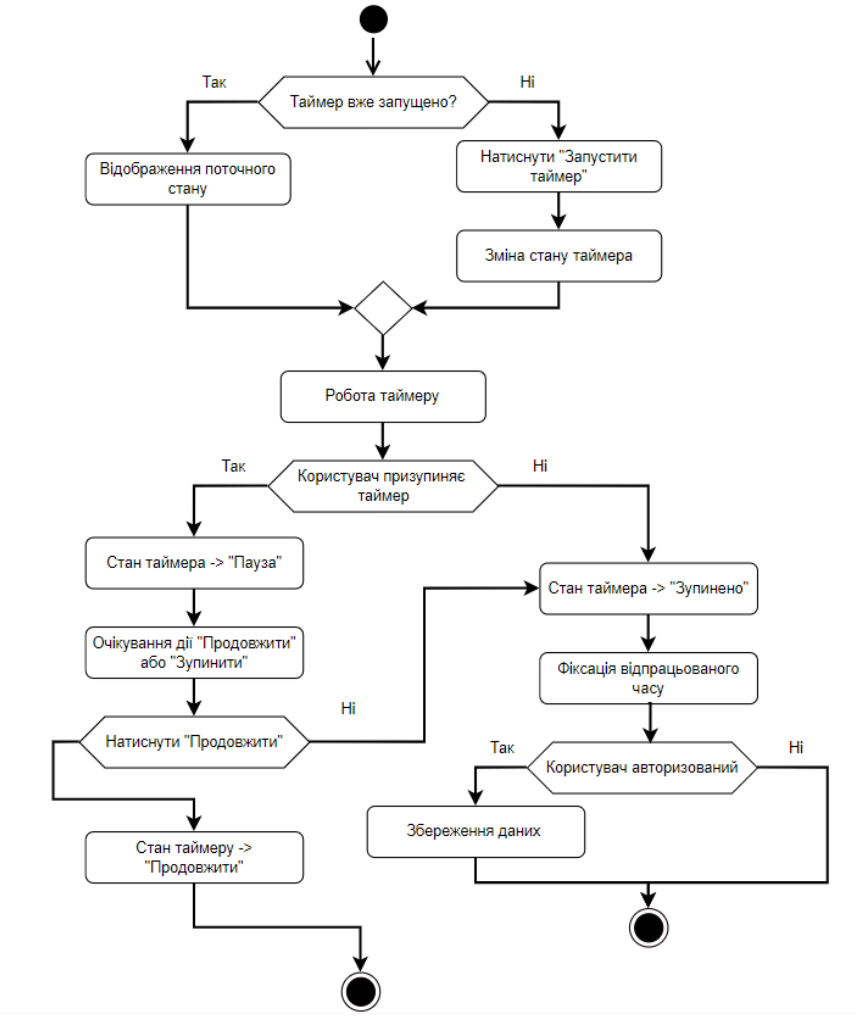

Рисунок 2.6 - Діаграма діяльності для прецеденту «Управління таймеру»

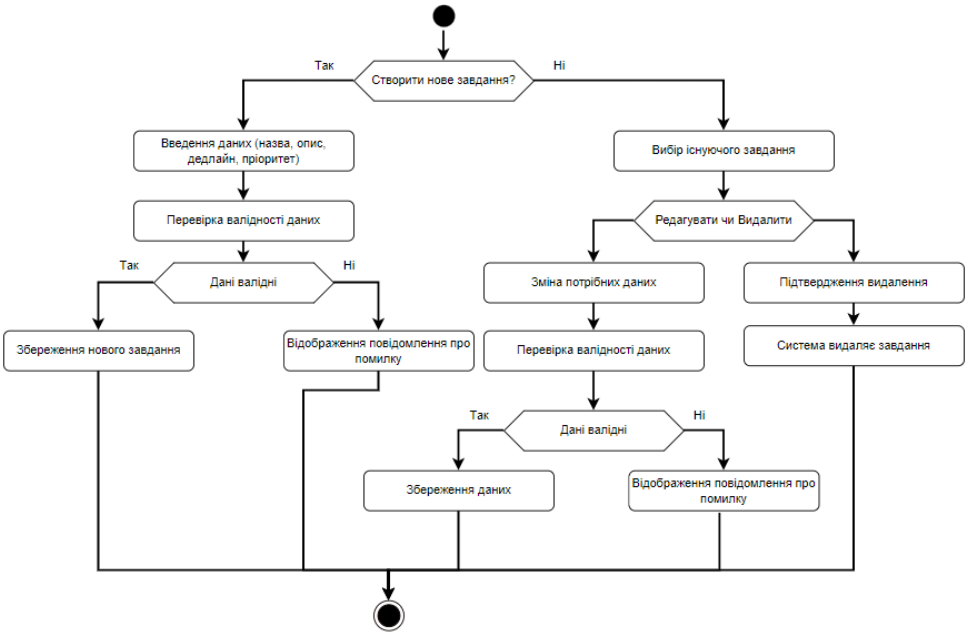

Рисунок 2.7 - Діаграма діяльності для прецеденту «Управління завданням»

Розробка концептуальної моделі

На основі проведеного аналізу предметної галузі та сформульованої задачі було створено концептуальну модель програмного забезпечення. Ця модель показує структуру системи, ключові властивості її складових і зв’язки між ними, необхідні для чіткого розуміння основної мети розробки.

Модель «сутність - зв'язок» є концептуальною моделлю, що фокусується на визначенні основних об'єктів та їх взаємозв'язків, не враховуючи специфічних особливостей конкретної системи управління базами даних. Найбільш поширеною стала ER-модель (Entity-Relationship model), запропонована Пітером Ченом у 1976 році.

Концептуальна модель вебзастосунку для планування робочого часу «TaskBalance», що відображає основні сутності, їх властивості та взаємозв'язки, наведена на рисунку 2.8.

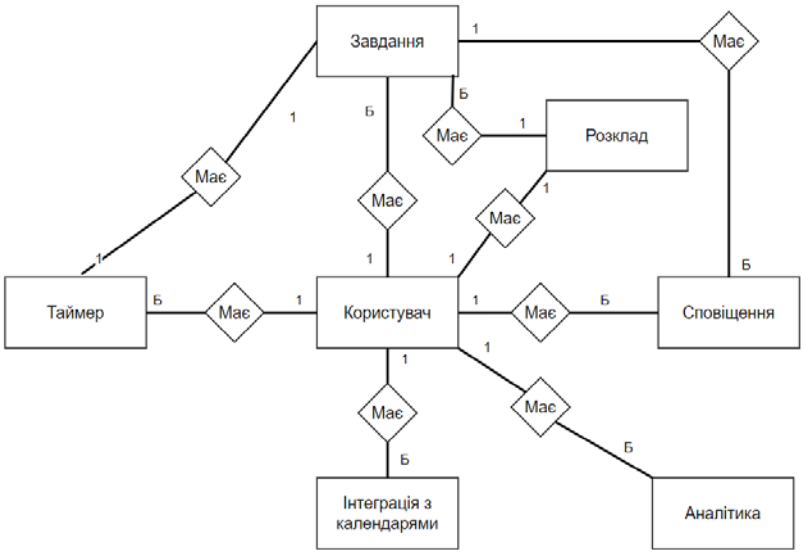

Рисунок 2.8 - Концептуальна модель вебзастосунку для планування робочого часу «TaskBalance»

1. Сутність «Користувач» (User)

Призначення: Відповідає користувачеві системи, який може реєструватися, авторизуватися, створювати та редагувати завдання, запускати таймер,

отримувати сповіщення, переглядати розклад, керувати інтеграціями з календарями та аналізувати власну продуктивність.

Основні атрибути:

– ID користувача (первинний ключ): унікальний ідентифікатор.

– Ім’я : ім’я або псевдонім користувача.

– Email: електронна адреса (повинна бути унікальною).

– Пароль: зберігається в захищеному вигляді (хеш).

– Дата реєстрації: дата, коли користувач створив обліковий запис.

2. Сутність «Завдання» (Task)

Призначення: Зберігає інформацію про робочі завдання, створені користувачем.

Основні атрибути:

– ID завдання (первинний ключ): унікальний ідентифікатор завдання.

– ID користувача (зовнішній ключ): вказує, кому належить завдання.

– Назва: короткий заголовок завдання.

– Опис: детальний опис завдання.

– Стан: статус виконання (нове, в процесі, виконано).

– Пріоритет: визначає важливість (низький, середній, високий).

– Дедлайн: дата/час, до якого бажано завершити завдання.

3. Сутність «Розклад» (Schedule)

Призначення: Зберігає дані про календар чи часову шкалу користувача, де відображаються заплановані завдання або події.

Основні атрибути:

– ID розкладу (первинний ключ): унікальний ідентифікатор розкладу.

– ID користувача (зовнішній ключ): визначає, якому користувачеві належить цей розклад.

– Назва: назва розкладу (наприклад, «Робочий графік», «Особисті завдання»).

– Опис: додаткова інформація про розклад.

– Періодичність: вказує, чи повторюється розклад (наприклад, щотижня, щодня).

4. Сутність «Сповіщення» (Notification)

Призначення: Містить інформацію про оповіщення, які надсилаються користувачеві (про наближення дедлайнів, зміни в розкладі, завершення таймера тощо).

Основні атрибути:

– ID сповіщення (первинний ключ): унікальний ідентифікатор.

– ID користувача (зовнішній ключ): визначає, кому надсилається сповіщення.

– Стан: відображає, чи включене або виключене сповіщення.

– Тип: які сповіщення повинні відправлятися.

5. Сутність «Інтеграція з календарями» (CalendarIntegration)

Призначення: Зберігає інформацію про підключені зовнішні календарі (Google, Apple) та параметри синхронізації.

Основні атрибути:

– ID інтеграції (первинний ключ): унікальний ідентифікатор інтеграції.

– ID користувача (зовнішній ключ): визначає власника інтеграції.

– Сервіс: назва сервісу (Google Calendar, Apple Calendar).

– Токен: ключ доступу або налаштування для API.

6. Сутність «Аналітика» (Analytics)

Призначення: Зберігає дані про продуктивність користувача: кількість завершених завдань, відпрацьований час тощо.

Основні атрибути:

– ID аналітики (первинний ключ): унікальний ідентифікатор запису.

– ID користувача (зовнішній ключ): визначає, кому належить аналітика.

– Завершені завдання: кількість виконаних завдань.

– Загальний час: сумарний час роботи (за таймером).

– Період: часовий проміжок, за який сформована статистика (день, тиждень, місяць).

7. Сутність «Таймер» (Timer)

Призначення: Зберігає інформацію про сесії відстеження робочого часу (запуск, зупинка, пауза).

Основні атрибути:

– ID таймера (первинний ключ): унікальний ідентифікатор сесії.

– ID користувача (зовнішній ключ): визначає, хто запустив таймер.

– ID завдання (зовнішній ключ, опційно): вказує, для якого завдання відстежується час.

– Початок (start_time): дата/час запуску.

– Завершення (end_time): дата/час зупинки або завершення.

Розробка прототипу інтерфейсу користувача

Виходячи з аналізу діаграми варіантів використання вебзастосунку для планування робочого часу «TaskBalance», було розроблено концепцію інтерфейсу користувача цього програмного забезпечення. Інтерфейс користувача охоплює ключові елементи та компоненти вхідних форм, що визначають ефективність взаємодії між користувачем і системою. Ескізи графічного інтерфейсу наведено на рисунках 2.9-2.12.

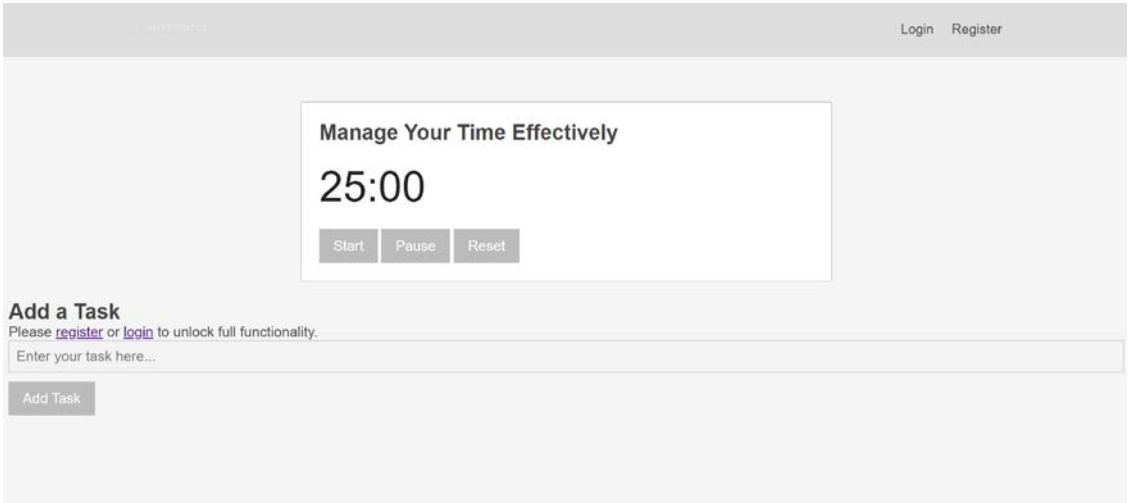

Рисунок 2.9 - Головна сторінка вебзастосунку «TaskBalance»

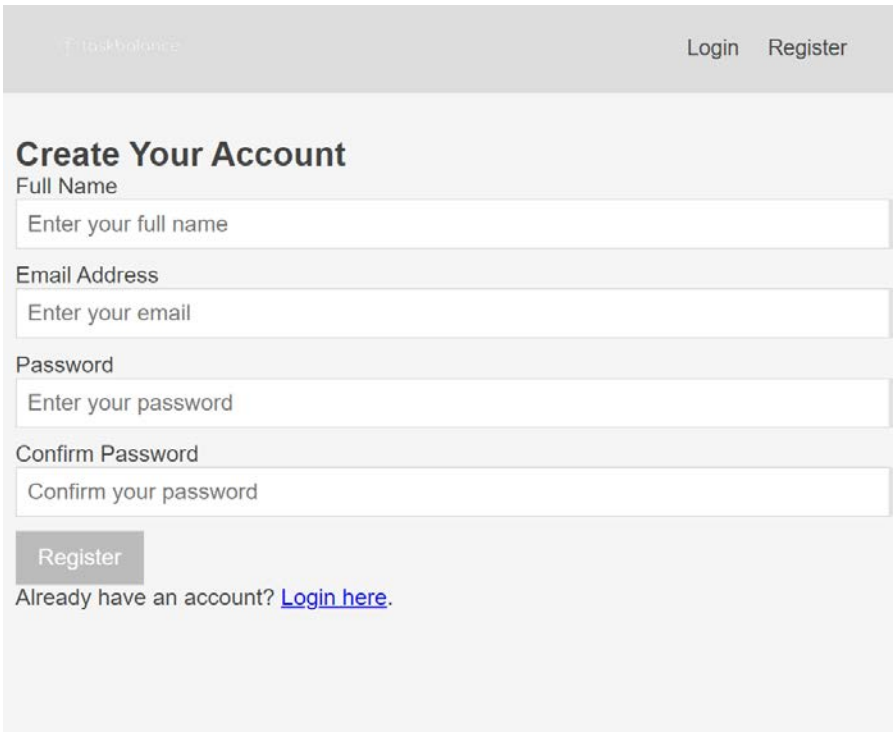

Рисунок 2.10 - Форма реєстрації користувача

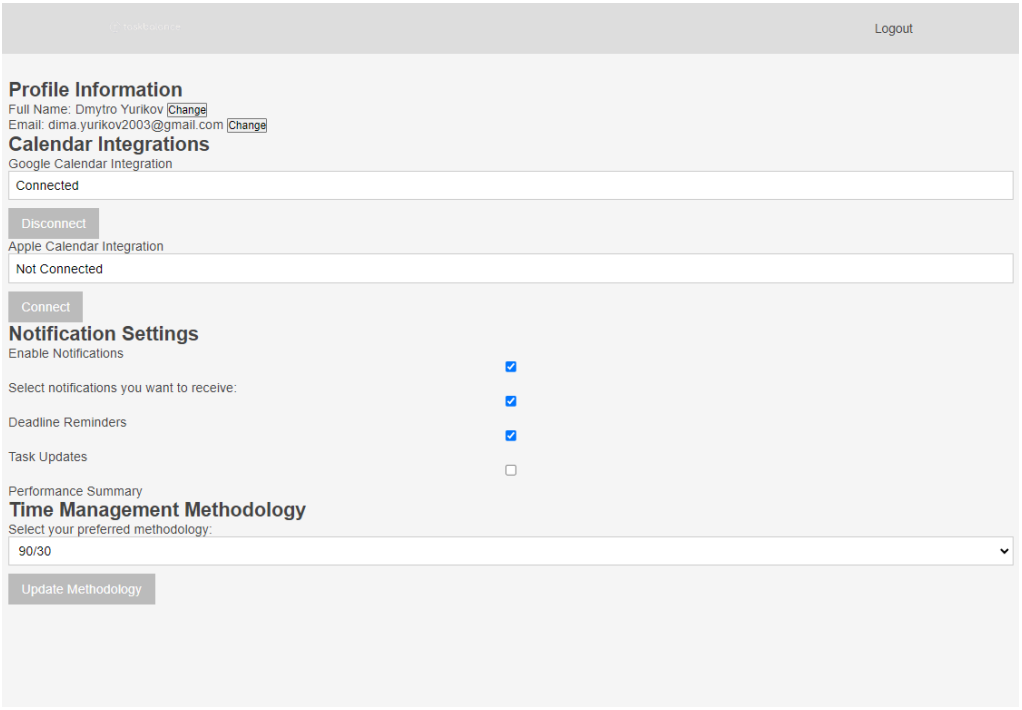

Рисунок 2.11 - Сторінка профілю користувача

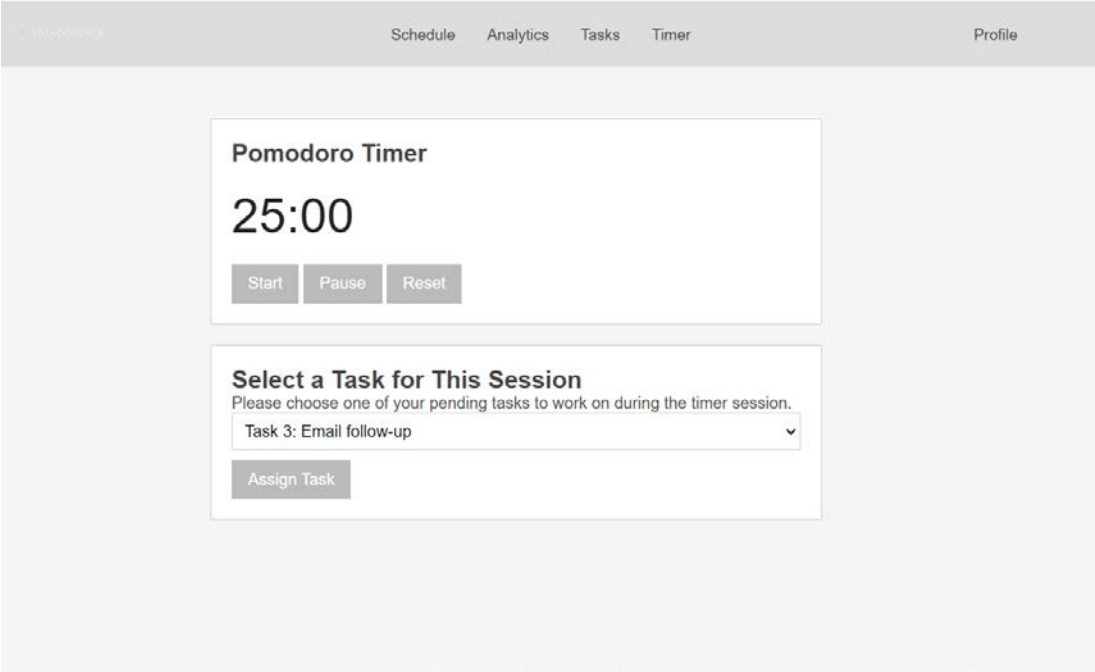

Рисунок 2.12 - Сторінка таймеру зареєстрованого користувача

Маючи готовий ескізний проєкт вебзастосунку для планування робочого часу «TaskBalance», можна переходити до наступного етапу розробки - створення технічного проєкту цього програмного забезпечення.

2.3.2 Технічний проект програмного забезпечення

Діаграма класів UML це візуальна нотація, яка використовується для побудови та візуалізації об'єктно-орієнтованих систем. Діаграма класів в уніфікованій мові моделювання - це статична структурна діаграма, що демонструє властивості системи, класи, операції та зв’язки між об’єктами для опису структури системи. Діаграма класів для вебзастосунку для планування робочого часу «TaskBalance» наведено на рисунку 2.13.

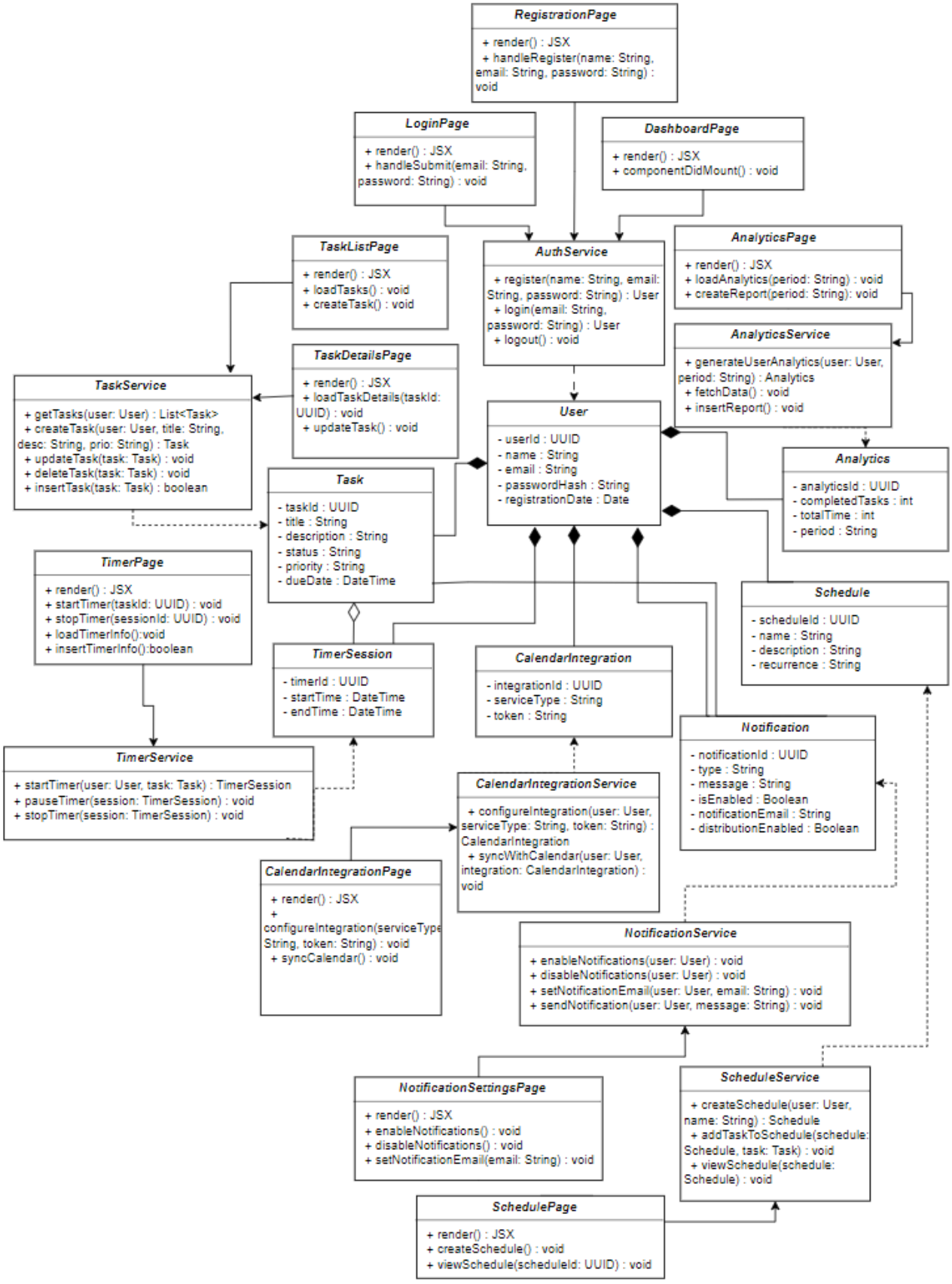

Рисунок 2.13 - Діаграма класів для вебзастосунку для планування робочого часу «TaskBalance»

Розробимо розгорнуту специфікацію класів вебзастосунку для планування робочого часу «TaskBalance», представлено у таблицях 2.16-2.22.

Таблиця 2.16 - Специфікація класу «User»

Складова	Опис
Відповідальність	Клас відповідає за зберігання та управління інформацією про користувача (авторизація, профіль).
Атрибути	- userId : UUID - унікальний ідентифікатор користувача
	- name : String - ім’я користувача
	- email : String - електронна адреса (унікальна)
	- passwordHash : String - хешований пароль
	- registrationDate : Date - дата реєстрації
Операції зі специфікацією нетривіальних операцій	- register() : void - виконує процедуру реєстрації нового користувача
	- login(email, password) : boolean - перевіряє вказані облікові дані та виконує авторизацію
	- updateProfile(name, email, password) : void - оновлює дані профілю (ім’я, пошту, пароль)

Таблиця 2.17 - Специфікація класу «Task»

Складова	Опис
Відповідальність	Зберігає дані про завдання, забезпечує операції створення, редагування та видалення завдань.
Атрибути	- taskId : UUID - унікальний ідентифікатор завдання
	- title : String - назва завдання
	- description : String - детальний опис
	- status : String - стан завдання (нове, в процесі, виконано)
	- priority : String - пріоритет (низький, середній, високий)
	- dueDate : DateTime - крайній термін виконання
Операції зі специфікацією нетривіальних операцій	- createTask() : void - створює нове завдання
	- updateTask() : void - оновлює атрибути завдання (назва, опис, статус, пріоритет тощо)
	- deleteTask() : void - видаляє завдання із системи
	- loadTask(): void - завантажує завданя з бази даних
	- insertTask() : boolean - записує дані у базу даних

Таблиця 2.18 - Специфікація класу «TimerSession»

Складова	Опис
1	2
Відповідальність	Відстежує робочий час користувача, дозволяючи запускати, ставити на паузу та зупиняти відлік.
Атрибути	- timerId : UUID - унікальний ідентифікатор сесії таймера
	- startTime : DateTime - час запуску
	- endTime : DateTime - час зупинки/паузи

Продовження таблиці 2.18.

1	2
Операції зі специфікацією нетривіальних операцій	- start() : void - запускає таймер
	- pause() : void - ставить таймер на паузу
	- stop() : void - зупиняє таймер і фіксує загальний відпрацьований час
	- loadTimerInfo() : void - завантажує інформацію про таймер з бази даних
	- insertTimerInfo() : boolean - записує дані до бази даних

Таблиця 2.19 - Специфікація класу «CalendarIntegration»

Складова	Опис
Відповідальність	Забезпечує зв’язок із зовнішніми календарними сервісами (Google Calendar, Apple Calendar), виконує синхронізацію.
Атрибути	- integrationId : UUID - унікальний ідентифікатор інтеграції
	- serviceType : String - тип сервісу
	- token : String - ключ доступу (API-токен)
Операції зі специфікацією нетривіальних операцій	- configure() : void - налаштовує інтеграцію, зберігає параметри доступу
	- sync() : void - синхронізує завдання/події із зовнішнім календарем

Таблиця 2.20 - Специфікація класу «Analytics»

Складова	Опис
Відповідальність	Збирає та обробляє статистичні дані щодо продуктивності користувача (завершені завдання, загальний відпрацьований час).
Атрибути	- analyticsId : UUID - унікальний ідентифікатор запису аналітики
	- completedTasks : int - кількість виконаних завдань
	- totalTime : int - сумарний відпрацьований час (хвилини/секунди)
	- period : String - період (день, тиждень, місяць)
Операції зі специфікацією нетривіальних операцій	- generateReport() : void - формує звіт або статистику за обраний період (відсоток виконання, динаміку часу)
	- createReport() : void - обробка запита на створення
	- loadAnalytics(): void : завантажує інформацію про аналітичний звіт
	- insertReport() : boolean - записує дані про аналітичний звіт в базу даних
	- fetchData() : String - створює запит на отримання даних про завданя з бази даних

Таблиця 2.21 - Специфікація класу «Schedule»

Складова	Опис
1	2
Відповідальність	Зберігає дані про розклад користувача (календар), де відображаються заплановані завдання та події.

Продовження таблиці 2.21.

1	2
Атрибути	- scheduleId : UUID - унікальний ідентифікатор розкладу
	- name : String - назва розкладу
	- description : String - опис
	- recurrence : String - періодичність (щодня, щотижня)
Операції зі специфікацією нетривіальних операцій	- addTask(task : Task) : void - додає завдання до розкладу
	- removeTask(task : Task) : void - видаляє завдання з розкладу
	- viewSchedule() : void - відображає поточний розклад (у вигляді календаря)

Таблиця 2.22 - Специфікація класу «Notification»

Складова	Опис
Відповідальність	Відповідає за надсилання сповіщень користувачу про важливі події: наближення дедлайнів, зміни в розкладі, завершення робочих інтервалів тощо.
Атрибути	- notificationId : UUID - унікальний ідентифікатор сповіщення
	- type : String - тип сповіщення (email)
	- isEnabled : Boolean - стан (увімкнено/вимкнено)
	- notificationEmail : String - адреса для оповіщень
	- distributionEnabled : Boolean - прапорець розсилки (чи надсилати регулярні нагадування)
Операції зі специфікацією нетривіальних операцій	- enable() : void - увімкнути сповіщення
	- disable() : void - вимкнути сповіщення
	- setNotificationEmail(email : String) : void - встановити адресу для розсилки
	- toggleDistribution(enable : Boolean) : void - керувати розсилкою (увімкнути/вимкнути)

Після розробки специфікації класів вебзастосунку для планування часу «TaskBalance» можна розробити динамічну модель вебзастосунку.

Динамічна модель вебзастосунку

Динамічну модель вебзастосунку для планування робочого часу «TaskBalance» зручно представити у форматі основних діаграм послідовностей. Ці діаграми належать до групи діаграм взаємодії та відображають поведінкові аспекти системи, фокусуючись на взаємодії об’єктів у часовому вимірі. Завдяки цьому можна показати, як об’єкти обмінюються повідомленнями та виконують дії в певному порядку. Діаграми послідовностей доповнюють діаграми варіантів

використання, уточнюючи й деталізуючи сценарії взаємодії. На рисунках 2.14-2.16 наведено динамічну модель вебзастосунку у вигляді основних діаграм послідовностей.

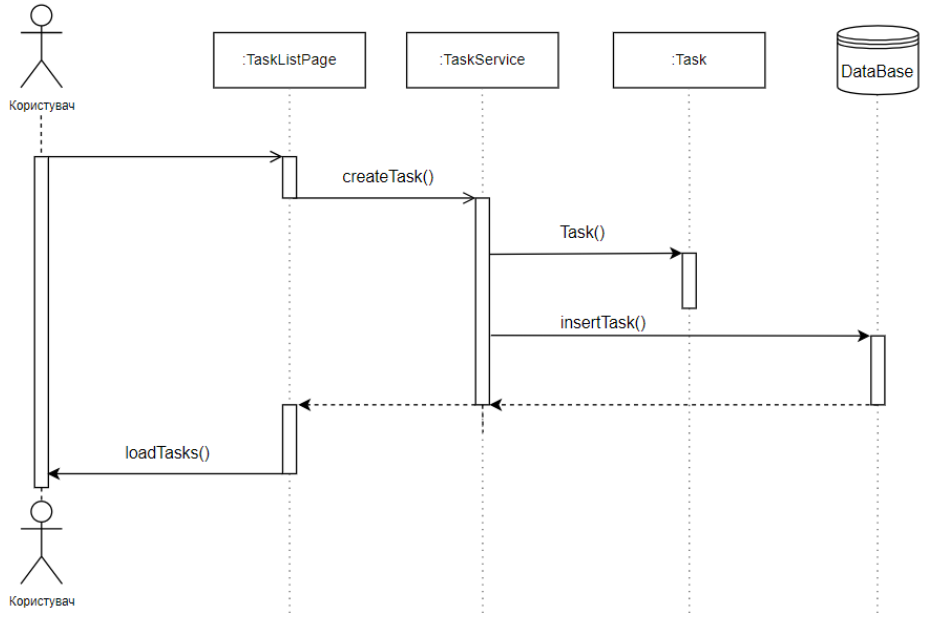

Рисунок 2.14 - Діаграма послідовностей для прецеденту «Додання завдання»

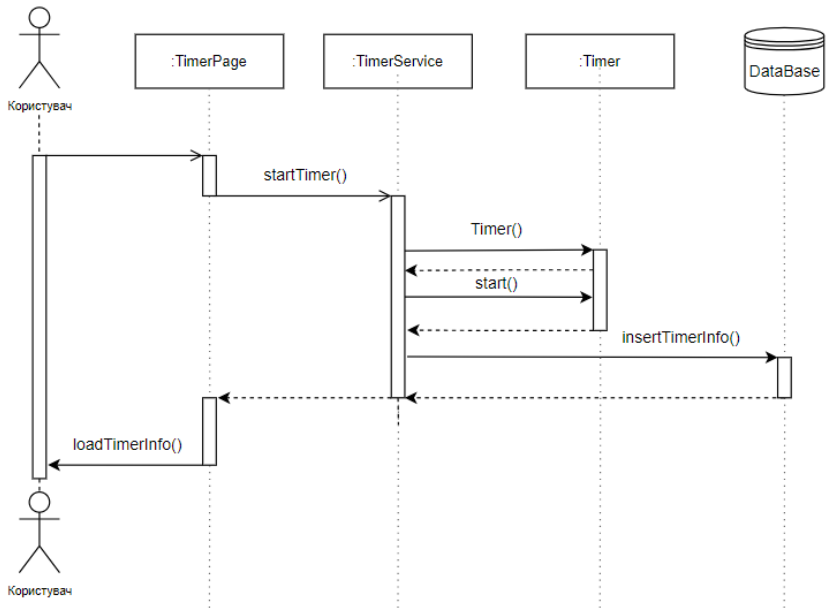

Рисунок 2.15 - Діаграма послідовностей для прецеденту «Запуск таймеру»

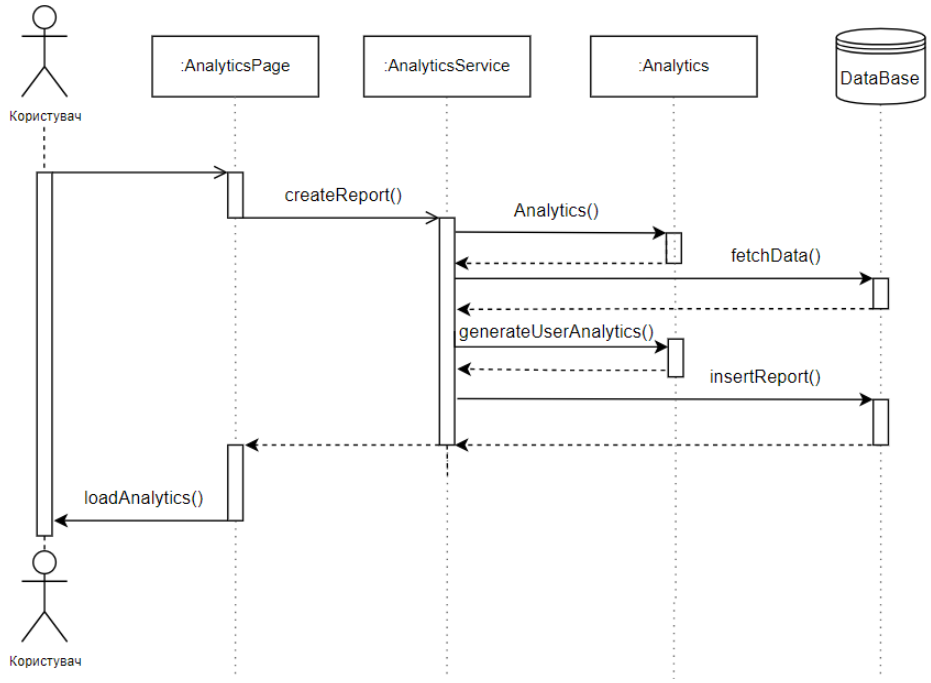

Рисунок 2.16 - Діаграма послідовностей для прецеденту «Створення аналітичного звіту»

Така структура взаємодії забезпечує зрозумілий і надійний сценарій виконання дій, що є ключовим фактором для розробки масштабованої та підтримуваної системи.

2.3.3 Робочий проект вебзастосунку

Вибір мови програмування, системи керування базами даних і середовища розробки

Для розробки вебзастосунку TaskBalance, який призначений для планування робочого часу, було обрано сучасний технологічний стек, що поєднує ефективність, гнучкість і широкі можливості масштабування. Основними критеріями вибору стали: популярність технологій у професійному середовищі, доступність документації, підтримка спільноти, а також зручність у створенні масштабованих інтерфейсів користувача.

Мова програмування - JavaScript

JavaScript - це інтерпретована мова програмування, яка використовується переважно для створення динамічного вмісту на вебсторінках. Вона є основною технологією фронтенд-розробки і підтримується всіма сучасними веббраузерами без потреби встановлення додаткових компонентів [9].

У вебзастосунку TaskBalance JavaScript використовується разом із бібліотекою React для реалізації клієнтської частини [10]. React дозволяє будувати інтерфейс за компонентною архітектурою, що підвищує повторне використання коду, спрощує тестування та пришвидшує розробку. Використання React також дозволяє легко реалізовувати односторінкові застосунки (SPA), що є актуальним для інструментів тайм-менеджменту.

Система керування базами даних - MongoDB

Для зберігання даних у застосунку TaskBalance була обрана MongoDB - документоорієнтована система керування базами даних з відкритим кодом [11]. Вона працює за принципом NoSQL і зберігає дані у форматі BSON (бінарний аналог JSON), що ідеально підходить для структурованих, але гнучких даних.

MongoDB забезпечує:

– гнучкість структури: можна зберігати завдання з довільною кількістю полів, що важливо для розширення функціоналу;

– масштабованість: система підтримує горизонтальне масштабування для обробки великих обсягів інформації;

– просту інтеграцію з JavaScript/React: через офіційний Node.js-драйвер та бібліотеки типу Mongoose;

– хмарне розгортання (MongoDB Atlas) - спрощує налаштування, резервне копіювання та моніторинг даних.

Дані про завдання, користувачів, таймери та статистику продуктивності зберігаються у вигляді документів у відповідних колекціях MongoDB, що дозволяє гнучко налаштовувати структуру відповідно до логіки застосунку.

Середовище розробки - Visual Studio Code

Для написання, тестування та налагодження коду використовується Visual Studio Code (VSCode) [12] - сучасне кросплатформне інтегроване середовище розробки. Серед його ключових переваг:

– підтримка розширень: можливість підключення плагінів для React, MongoDB, ESLint, Prettier тощо;

– вбудований термінал: дозволяє запускати сервери, менеджери пакетів (npm/yarn) без виходу з редактора;

– інтелектуальне автодоповнення: покращує швидкість написання коду;

– інтеграція з Git: забезпечує контроль версій і командну роботу.

Таким чином, обраний стек технологій - JavaScript + React для фронтенду, MongoDB для бекенду та VSCode як середовище розробки - дозволяє ефективно реалізувати функціонал TaskBalance з урахуванням сучасних вимог до вебзастосунків.

Розробка фізичної модель бази даних

Фізична модель бази даних описує структуру даних за допомогою засобів конкретної СУБД та демонструє спосіб їх зберігання. На рисунку 2.17 представлена фізична модель бази даних вебзастосунку для планування часу TaskBalance. Вона відображає структуру колекцій, їхні взаємозв’язки та визначає основні поля документів, що забезпечують зберігання даних про користувачів, завдання, розклади, нагадування та аналітику продуктивності.

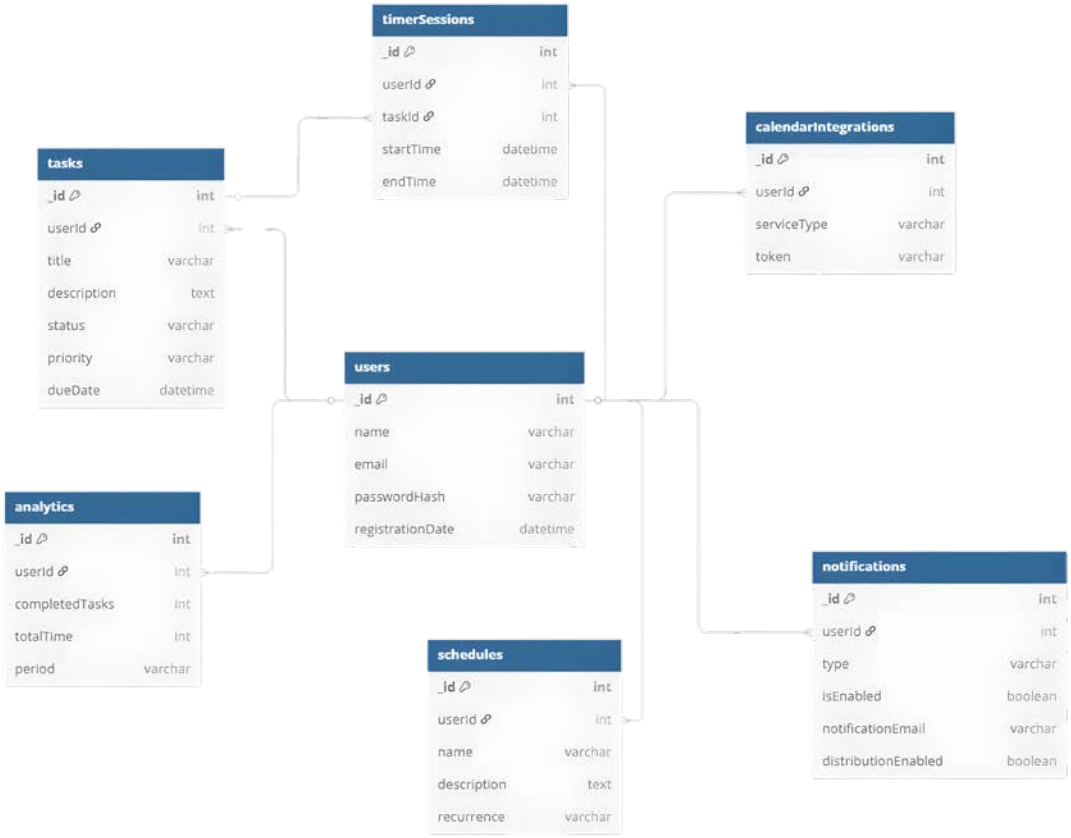

Рисунок 2.17 - Фізична модель бази даних вебзастосунку для планування часу «TaskBalance»

Нижче у таблицях 2.23 - 2.29 наведено опис таблиць фізичної моделі бази даних вебзастосунку TaskBalance та їх відповідність логічній моделі даних.

Таблиця 2.23 - Опис атрибутів сутності «users»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор користувача
-	name	String	Ім'я користувача
-	email	String	Електронна адреса (унікальна)
-	passwordHash	String	Хешований пароль
-	registrationDate	Date	Дата реєстрації користувача

Таблиця 2.24 - Опис атрибутів сутності «tasks»

Ключ	Поле	Тип даних	Опис
1	2	3	4
Primary Key	_id	ObjectId	Унікальний ідентифікатор завдання
Foreign Key	userId	ObjectId	Ідентифікатор користувача

Продовження таблиці 2.24.

1	2	3	4
-	title	String	Назва завдання
-	description	String	Опис завдання
-	status	String	Стан завдання (new, in_progress, done)
-	priority	String	Пріоритет (low, medium, high)
-	dueDate	DateTime	Крайній термін виконання

Таблиця 2.25 - Опис атрибутів сутності «timerSessions»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор сесії таймера
Foreign Key	userId	ObjectId	Ідентифікатор користувача
Foreign Key	taskId	ObjectId	Ідентифікатор завдання
-	startTime	DateTime	Час запуску таймера
-	endTime	DateTime	Час завершення або паузи таймера

Таблиця 2.26 - Опис атрибутів сутності «calendarIntegrations»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор інтеграції
Foreign Key	userId	ObjectId	Ідентифікатор користувача
-	serviceType	String	Тип сервісу (Google, Apple)
-	token	String	API-токен для доступу

Таблиця 2.27 - Опис атрибутів сутності «analytics»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор аналітичного запису
Foreign Key	userId	ObjectId	Ідентифікатор користувача
-	completedTasks	Number	Кількість виконаних завдань
-	totalTime	Number	Загальний відпрацьований час (в хвилинах)
-	period	String	Період (day, week, month)

Таблиця 2.28 - Опис атрибутів сутності «schedules»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор розкладу
Foreign Key	userId	ObjectId	Ідентифікатор користувача
-	name	String	Назва розкладу
-	description	String	Опис події
-	recurrence	String	Періодичність (daily, weekly, monthly)

Таблиця 2.29 - Опис атрибутів сутності «notifications»

Ключ	Поле	Тип даних	Опис
Primary Key	_id	ObjectId	Унікальний ідентифікатор сповіщення
Foreign Key	userId	ObjectId	Ідентифікатор користувача
-	type	String	Тип сповіщення (email, push)
-	isEnabled	Boolean	Стан сповіщення (true - активне, false - вимкнене)
-	notificationEmail	String	Адреса для надсилання електронних листів
-	distributionEnabled	Boolean	Прапорець розсилки регулярних нагадувань

Кінцевим результатом є відображення таблиць та зв'язків між ними в середовищі обраної системи керування базами даних (СКБД).

Тестування програмного забезпечення

Виконано аналіз правильних та неправильних класів еквівалентності та здійснимо тестування на коректність роботи вебзастосунку TaskBalance у різних типових і нетипових ситуаціях. Тестування базується на класах, реалізованих згідно з діаграмою класів, що описує архітектуру програмного забезпечення для планування робочого часу. Як приклад розглянуто тестові сценарії для класу «User».

Представлено код класу, який буде тестуватися:

```
class User {
constructor(id, name, email, passwordHash, registrationDate) {
this.id = id;
this.name = name;
this.email = email;
this.passwordHash = passwordHash;
this.registrationDate = registrationDate || new Date().toISOString();
}

register(name, email, passwordHash) {
this.name = name;
this.email = email;
this.passwordHash = passwordHash;
this.registrationDate = new Date().toISOString();
}
login(email, passwordHash) {
return this.email === email && this.passwordHash === passwordHash;
}
updateProfile(name, email, passwordHash) {
if (name) this.name = name;
if (email) this.email = email;
if (passwordHash) this.passwordHash = passwordHash;
}
```

```
}
export default User;
```

Текст вебзастосунку для планування робочого часу TaskBalance наведено в Додатку Б.

В таблиці 2.30 представлені класи еквівалентності для класу «User»

Таблиця 2.30 - Класи еквівалентності для класу «User»

№	Поле	Правильний клас	Неправильний клас	Очікувана поведінка
1	name	Рядок з літер, 2-30 символів	Порожній рядок, числа, спецсимволи	Приймається / Виводиться помилка
2	email	Валідна email-адреса (user@mail.com)	Без @ або домену, з пробілами	Приймається / Виводиться помилка
3	passwordHash	Рядок довжиною 8-64 символи	Занадто короткий (<8), порожній	Приймається / Виводиться помилка
4	registrationDate	Коректна ISO-дата (2025-05-01)	Невалідний формат дати або null	Зберігається / Встановлюється за замовчуванням
5	login()	Email і пароль збігаються	Email або пароль неправильні	Повертає true / false
6	updateProfile()	Вхідні дані валідні	Одне або кілька полів - невалідні	Дані оновлюються / Виводиться помилка

Почнемо з тестування правильних класів еквівалентності для класу «User», результати якого наведено у табл. 2.31. Згідно з результатами тестування, вебзастосунок «TaskBalance» коректно обробляє вхідні дані в межах допустимих значень та забезпечує стабільну роботу згідно з вимогами функціональності. Таким чином, на основі аналізу правильних класів еквівалентності можна зробити висновок, що реалізація логіки класу «User» є працездатною і відповідає очікуваній поведінці системи.

Таблиця 2.31 - Тестування для правильних класів еквівалентності класу «User»

Вхідні дані	Очікуваний результат
1	2
name: "Олександр", email: "olex@gmail.com", password: "Secure123"	Користувача зареєстровано, дата реєстрації встановлена
name: "Марія", email: "maria.smith@example.com", password: "MyPass456"	Успішне створення облікового запису

Продовження таблиці 2.31.

1	2
email: "olex@gmail.com", password: "Secure123"	Вхід успішний: повертається true
updateProfile(name: "Олександр К.", email: "olex.k@site.ua", password: "QwErTy789")	Профіль оновлено згідно з новими даними
passwordHash довжиною 12 символів (наприклад, "Abcd1234Efgh")	Приймається як валідне значення паролю
email: "test.user@ukr.net", password: "Pass9876"	Користувача авторизовано
name: "Юлія", password: "StrongOne321", email: "yulia@domain.org"	Дані оброблено коректно, створено користувача

Протестуємо неправильні класи еквівалентності для класу «User», результати чого наведено в таблиці 2.32.

Таблиця 2.32 - Тестування для неправильних класів еквівалентності класу «User»

Вхідні дані	Очікуваний результат
name: "", email: "user@gmail.com", password: "Qwerty123"	Помилка: ім’я не може бути порожнім
name: "!", email: "user@gmail.com", password: "Qwerty123"	Помилка: ім’я містить недопустимі символи
name: "Катерина", email: "kateryna", password: "Secure123"	Помилка: email не відповідає формату
name: "Ігор", email: "igor@domain", password: "123"	Помилка: пароль занадто короткий
name: "Анна", email: "anna@gmail.com", password: ""	Помилка: порожнє поле пароля
email: "", password: "Qwerty123"	Помилка: відсутній email
name: "Дмитро", email: "user@mail.com", password: "short"	Помилка: пароль не відповідає мінімальним вимогам
updateProfile(name: "", email: "bademail@", password: "pass")	Помилка: невалідні дані профілю

За результатами тестування неправильних класів еквівалентності для класу «Користувач», вебзастосунок TaskBalance функціонує коректно, не допускаючи введення недопустимих значень, окрім передбачених допустимих символів та форматів. Усі перевірки проводилися з використанням підходу "чорного ящика", тобто без урахування внутрішньої структури коду. Тестування полягало у послідовному введенні заздалегідь підготовлених тестових даних та спостереженні за реакцією системи з подальшим аналізом результатів.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ РОБОЧОГО ЧАСУ «TASKBALANCE»

У цій кваліфікаційній роботі було вирішено практичну задачу інженерії програмного забезпечення, що відзначається комплексністю та невизначеністю умов - розроблено вебзастосунок для планування робочого часу «TaskBalance». Застосунок призначений для підвищення особистої продуктивності, систематизації завдань і ефективного управління часом. Розробка програмного забезпечення здійснювалась відповідно до встановлених стандартів та вимог до сучасних вебсистем. Зовнішній вигляд головного інтерфейсу вебзастосунку представлено на рисунку 3.1.

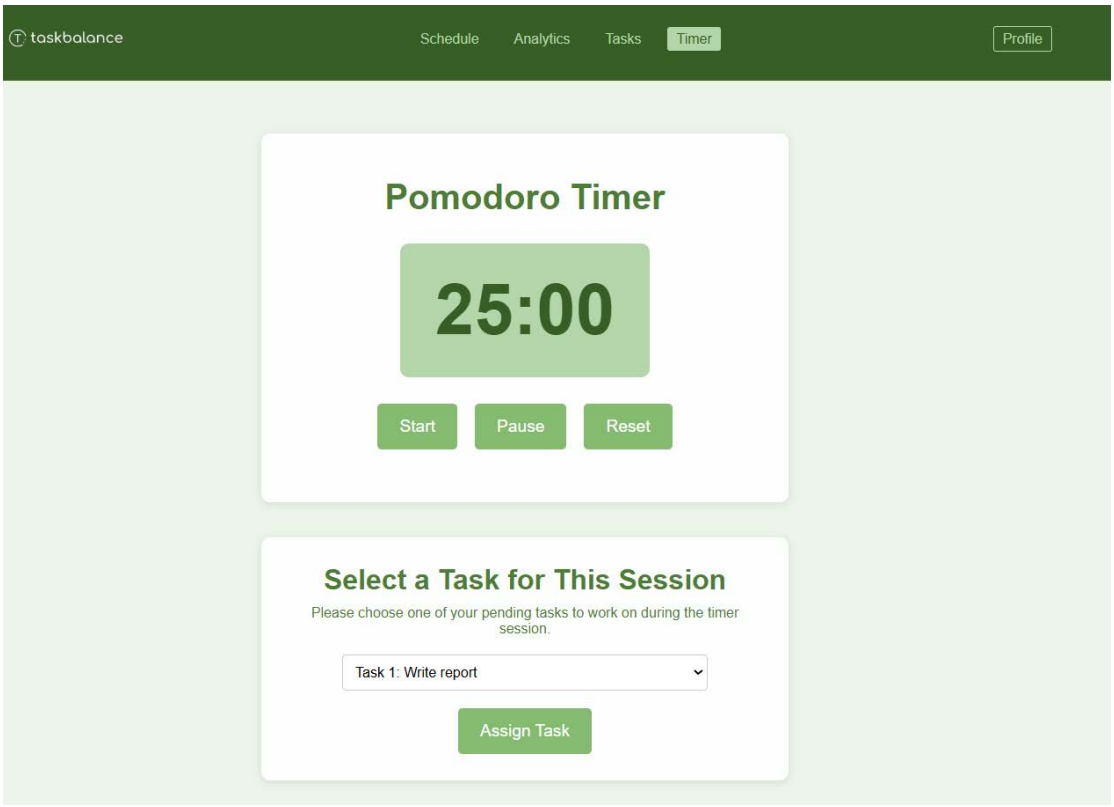

Рисунок 3.1 - Головна сторінка вебзастосунку «TaskBalance»

Проведено детальний аналіз проблематики планування робочого часу в сучасних умовах, а також вивчено тенденції автоматизації цього процесу за

допомогою цифрових рішень. Для формування архітектурної основи майбутнього вебзастосунку було опрацьовано літературу, нормативні документи та інформаційні ресурси з відкритого доступу, що висвітлюють сучасні підходи до організації персонального тайм-менеджменту. Здійснено огляд існуючих програмних засобів із подібним функціональним призначенням, визначено їхні переваги й недоліки.

На основі результатів аналізу сформульовано вимоги до функціоналу вебзастосунку TaskBalance. Розроблено проект програмного забезпечення, реалізовано його основні компоненти та проведено тестування. У рамках ескізного проекту засобами мови моделювання UML побудовано концептуальну модель, створено діаграми діяльності та розроблено макети інтерфейсу користувача. На етапі технічного проекту розроблено логічну модель бази даних, діаграму класів і динамічну модель програмного забезпечення у вигляді діаграм послідовностей.

В межах робочого проєкту було обґрунтовано вибір інструментів розробки: як мову програмування обрано JavaScript, для реалізації інтерфейсу використано бібліотеку React, а для зберігання даних - документоорієнтовану систему MongoDB. Середовищем розробки обрано Visual Studio Code. Також створено фізичну модель бази даних, реалізовано програмний код та виконано всебічне тестування системи.

Результати випробувань підтвердили працездатність та відповідність розробленого програмного забезпечення вимогам технічного завдання. У межах проекту підготовлено повний комплект супровідної документації: технічне завдання, опис програми, інструкція користувача, а також програма і методика тестування. У додатку Б подано фрагменти коду застосунку.

Вебзастосунок для планування робочого часу TaskBalance реалізує ключовий набір функцій, серед яких:

- реєстрація користувача;
- авторизація користувача;
- можливість виходу з системи;

– додавання та редагування завдань для планування робочого часу;

– налаштування персоналізованого робочого розкладу з використанням різних методик тайм-менеджменту (наприклад, Pomodoro, 52/17, 90/30, 1-3-5, правило двох хвилин);

– інтеграція з популярними календарними сервісами (Google Calendar, Apple Calendar) для синхронізації завдань та подій;

– моніторинг виконання завдань за запитом користувача;

– автоматизований моніторинг виконання завдань за встановленим розкладом;

– оповіщення про наближення дедлайнів, завершення робочих інтервалів або зміни в розкладі;

– збереження історії оповіщень та даних про виконання завдань;

– перегляд детальної аналітики щодо продуктивності та тривалості виконання завдань.

Розроблений вебзастосунок призначений для використання з метою планування робочого процесу та підвищення особистої ефективності.

Проведено випробування вебзастосунку. Програму та методику випробування наведено у Додатку Д.

Протокол випробування

Під час проведення випробувань вебзастосунку TaskBalance було здійснено повну перевірку функціональності, а також проведено навантажувальне тестування та тестування на відмовостійкість у рамках цілісного середовища. Усі виявлені недоліки було задокументовано в окремих протоколах і усунуто до моменту запуску системи у тестову експлуатацію.

Методи випробувань

Випробування виконано за стратегією «чорного ящика», без урахування внутрішньої реалізації логіки. Через значну кількість функцій, результати тестування наведено для декількох ключових сценаріїв.

1. Перевірка відповідності функціоналу технічному завданню

Перевірка функції "Додавання завдання"

У головному вікні обрано розділ «Завдання», натиснуто кнопку «Додати». У діалоговому вікні заповнено всі необхідні поля (назва, опис, дедлайн тощо) та натиснуто «Зберегти». Система додала завдання до списку без помилок.

Перевірка функції "Пошук завдань"

Вибрано розділ із завданнями, у полі пошуку введено ключове слово, після чого натиснуто кнопку «Пошук». Система повернула релевантні результати згідно з критеріями фільтрації.

Перевірка функції "Формування аналітичного звіту"

У розділі «Аналітика» обрано відповідний період, зазначено тип звіту та натиснуто кнопку «Згенерувати». Після обробки даних система сформувала звіт, який можна зберегти або експортувати.

2. Перевірка програмно-апаратної сумісності

Проведено тестування на різних пристроях з веббраузерами останніх версій (Chrome, Firefox, Edge), а також на різних операційних системах (Windows, macOS, Linux).

Усі конфігурації відповідали або перевищували мінімальні технічні вимоги, зазначені в технічному завданні.

Було перевірено повторне виконання тестів після усунення виявлених на попередньому етапі помилок.

3. Візуальна перевірка інтерфейсу

Проведено остаточне налагодження інтерфейсу для виявлення дрібних недоліків у відображенні елементів UI.

Перевірено адаптивність застосунку на різних розмірах екрану, зокрема ноутбуках, планшетах і мобільних пристроях.

Виправлено незначні відхилення у верстці та поведінці кнопок/форм при взаємодії.

Результат випробувань

Система TaskBalance успішно пройшла перевірку за всіма визначеними критеріями. Функціональність відповідає вимогам технічного завдання, а якість реалізації забезпечує її подальше розгортання в реальних умовах експлуатації.

4 ОХОРОНА ПРАЦІ

Охорона праці являє собою комплекс соціально-економічних, правових, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на захист життя, здоров’я та працездатності людини під час виконання трудових обов’язків [13].

Ця проблема є надзвичайно актуальною, оскільки сьогодні Україна знаходиться у вразливій позиції щодо забезпечення безпеки праці, збереження життя та здоров’я населення. Багато законів лишаються лише формальними документами, не отримуючи належної реалізації, а значна кількість працівників працює без офіційного оформлення, що позбавляє їх можливості захистити свої права.

Основні складові охорони праці включають правові норми трудового законодавства, вимоги гігієни та техніки безпеки, організацію виробничої санітарії, а також заходи протипожежного захисту та електробезпеки.

Головна мета - створення умов, що є безпечними, не становлять загрози для життя та здоров’я і забезпечують ефективне виконання трудових завдань.

4.1 Аналіз небезпечних і шкідливих факторів у відділі розробки програмного забезпечення

У відділі розробки програмного забезпечення працівники переважно виконують інтелектуальну працю за допомогою комп’ютерної техніки. Незважаючи на відсутність важкої фізичної роботи, умови праці в офісному середовищі можуть мати низку небезпечних і шкідливих чинників, що впливають на здоров’я співробітників. Серед них: мікрокліматичні відхилення, недостатнє освітлення, вплив електромагнітних полів, зорове навантаження, статичне положення тіла, психологічне перевантаження, а також ризики електротравм і

пожеж.

Для комфортної та безпечної роботи необхідно дотримуватись оптимальних параметрів мікроклімату. За санітарними нормами, температура повітря в офісному приміщенні повинна становити 22-24°C у теплу пору року та 18-21°C у холодну. Відносна вологість - 40-60%, швидкість руху повітря - не більше 0,2 м/с. Відхилення від цих показників можуть спричиняти дискомфорт, зниження працездатності, дратівливість, головний біль або ризик простудних захворювань.

Недостатнє або неправильно організоване освітлення є поширеним негативним фактором у роботі з комп'ютером. Низький рівень освітленості знижує ефективність праці, викликає втому очей і головний біль. Згідно з ДСанПіН 3.3.2.007-98 [14], рівень освітленості робочого місця за монітором має становити 400-750 лк. Світловий потік повинен бути рівномірним, без мерехтіння та тіней. Також важливо уникати відблисків на екрані, використовуючи штори, жалюзі або антивідблискові фільтри.

Сучасні працівники відділу розробки програмного забезпечення здебільшого використовують монітори з рідкокристалічними (LCD), світлодіодними (LED) або OLED-екранами. Попри те, що ці дисплеї значно безпечніші за застарілі електронно-променеві трубки, тривала робота перед монітором усе ж спричиняє підвищене навантаження на органи зору. До основних негативних впливів належать зорове перевантаження, сухість очей, зниження гостроти зору, головний біль і загальна втома. Причинами цього є тривала концентрація погляду на екрані, надмірна яскравість, мерехтіння або надмірне синє світло, яке випромінюється більшістю дисплеїв.

Комп'ютери, маршрутизатори, принтери та інша офісна техніка є джерелами неіонізуючого електромагнітного випромінювання (НІВ), яке, хоч і не має прямої загрози для здоров'я при дотриманні норм, все ж потребує контролю. Особливо це актуально у випадках великої кількості техніки в одному приміщенні. Регулярна перевірка рівня електромагнітного поля та правильне заземлення обладнання мінімізують ці ризики.

Всі пристрої у відділі працюють від електромережі, тому існує потенційна

загроза електротравм. Згідно з вимогами охорони праці, струми, які не загрожують життю людини, становлять: для змінного струму - до 1,5 мА, для постійного - до 7 мА.

Офісні приміщення, де розташоване комп’ютерне обладнання, віднесені до категорії пожежної небезпеки "В", адже тут знаходиться велика кількість легкозаймистих матеріалів - меблі, папір, пластикові елементи обладнання, кабелі тощо. Основні джерела загрози - коротке замикання, перегрів техніки, неправильне використання електроприладів.

Робота в ІТ-сфері вимагає постійної концентрації, вирішення складних задач, швидкої адаптації до змін, дедлайнів, що може спричинити емоційне вигорання, підвищений рівень стресу, тривожність. До негативних факторів також належать однотипність діяльності, обмежений рух та соціальна ізоляція. У таких умовах зростає важливість психологічної підтримки, мікропауз для розвантаження, створення сприятливого психологічного клімату у колективі.

4.2 Розробка заходів по зменшенню дії небезпечних і шкідливих факторів у відділі розробки програмного забезпечення

Зниження впливу небезпечних та шкідливих виробничих факторів у відділі розробки програмного забезпечення є ключовим етапом у забезпеченні ефективної роботи та охорони здоров’я працівників. Хоча ІТ-сфера вважається менш ризикованою порівняно з промисловим виробництвом, тривале перебування в офісному середовищі передбачає постійний вплив низки фізичних, психофізіологічних і навіть хімічних факторів. Тому впровадження системних, організаційних та інженерно-технічних заходів є необхідним.

Особливу увагу слід приділяти умовам мікроклімату, організації освітлення, ергономіці робочого простору, електробезпеці, профілактиці зорового і психологічного перенавантаження, а також протипожежним заходам. Кожен з цих напрямків потребує конкретних рішень і регулярного контролю.

З метою забезпечення безпеки та здорових умов праці у відділі розробки програмного забезпечення необхідно впроваджувати низку організаційних, технічних, санітарно-гігієнічних та профілактичних заходів. Усі дії мають відповідати чинним державним нормам, зокрема вимогам, викладеним у Наказі МОЗ України №400 від 30.07.2012 р. [15], що регулює роботу з джерелами електромагнітного випромінювання.

Забезпечення сприятливого мікроклімату

Комфортні мікрокліматичні умови сприяють підтриманню працездатності та знижують ризик розвитку простудних захворювань, перевтоми та зниження концентрації уваги. Для їх підтримання рекомендується використовувати системи автоматичного клімат-контролю, а також щоденно провітрювати приміщення.

Основні заходи з покращення мікроклімату:

– підтримка температури на рівні 22-24°C в теплий період року та 18-21°C - у холодний;

– забезпечення вологості повітря в межах 40-60%;

– використання кондиціонерів, зволожувачів і очисників повітря;

– регулярне вологе прибирання для зменшення запиленості.

Раціональна організація освітлення

Освітлення має значний вплив на продуктивність та емоційний стан працівників. Низький рівень освітленості або наявність блисків на екрані монітора може спричинити втому очей, головний біль і зниження працездатності.

Доцільно впроваджувати змішане освітлення - поєднання природного і штучного. Водночас важливо обирати джерела світла з комфортною колірною температурою та без мерехтіння.

Рекомендовані заходи:

– використання світлодіодних або люмінесцентних ламп з індексом кольоропередачі $Ra \geq 80$;

– рівень освітленості робочої зони - від 400 до 750 лк;

– встановлення штор або жалюзі для контролю природного освітлення;

– використання моніторів з матовим покриттям або антивідблискових

екранів.

Ергономіка та організація робочого місця

Неправильно організоване робоче місце призводить до розвитку захворювань опорно-рухового апарату, зниження концентрації та загального незадоволення умовами праці. Сучасна ергономіка передбачає створення умов, за яких людина може працювати максимально ефективно, не перенапружуючи тіло та психіку.

Основні рекомендації з ергономіки:

– використання регульованих крісел з підлокітниками та підтримкою попереку;

– розміщення монітора на відстані 50-70 см від очей;

– висота столу - близько 70-75 см, а екрана - на рівні очей або трохи нижче;

– наявність підставки для ніг за потреби.

Зниження зорового навантаження

Зорове перенавантаження - один із найпоширеніших факторів ризику у працівників, які працюють з комп’ютерами. Для його профілактики варто застосовувати сучасні технології захисту зору та організовувати режим роботи з регулярними перервами.

Ефективні заходи:

– використання моніторів із частотою оновлення 75-144 Гц;

– використання функцій фільтрації синього світла або захисних окулярів;

– впровадження правила "20-20-20";

– організація коротких перерв - 10-15 хв щогодини з легкими вправами для очей.

Психоемоційне розвантаження та профілактика вигорання

Психологічний комфорт - основа стабільної продуктивності в інтелектуальній праці. Стрес, перевтома та емоційне вигорання можуть виникнути навіть у добре обладнаному офісі, якщо не враховуються потреби

працівників у відпочинку та підтримці.

Рекомендовані заходи:

– впровадження гнучкого графіка та можливості дистанційної роботи;

– створення зони відпочинку для неформального спілкування;

– організація тренінгів із тайм-менеджменту, антистресу та емоційного інтелекту;

– забезпечення доступу до психологічної підтримки за потреби.

Підвищення рівня електробезпеки та пожежної безпеки

Оскільки у відділі працює велика кількість електронного обладнання, необхідно суворо дотримуватись норм електробезпеки та протипожежного захисту.

Комплекс необхідних заходів:

– встановлення заземлення для всіх електроприладів;

– перевірка справності кабелів, подовжувачів, розеток;

– наявність чотирьох вуглекислотних вогнегасників типу ВВ-5;

– розміщення плану евакуації у доступному місці;

– встановлення димових датчиків та пожежної сигналізації;

– проведення інструктажів із пожежної безпеки щонайменше раз на півроку.

Ефективне поєднання організаційних, інженерно-технічних та психологічних заходів дозволяє не лише знизити рівень небезпеки в офісному середовищі, а й створити умови, за яких працівники почуватимуться комфортно, захищено та будуть максимально продуктивними.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов - розроблено вебзастосунок для планування робочого часу «TaskBalance».

У ході виконання кваліфікаційної роботи було проаналізовано практичну задачу інженерії програмного забезпечення, зокрема сучасні тенденції в автоматизації планування робочого часу та організації особистої продуктивності. Проведено огляд існуючих цифрових рішень у сфері тайм-менеджменту, на основі якого зроблено висновок про доцільність створення власного вебзастосунку. Здійснено постановку задачі для розробки вебзастосунку TaskBalance, а також проведено детальний аналіз вимог до майбутнього програмного забезпечення. У рамках роботи виконано ескізне, технічне та робоче проектування системи, реалізовано ключову функціональність і представлено результати розробки. Окрему увагу також приділено загальним аспектам охорони праці відповідно до чинного законодавства України.

Розроблений вебзастосунок «TaskBalance» спрямований на спрощення організації робочого часу та підвищення ефективності управління завданнями за рахунок реалізації наступних функцій:

- реєстрація користувача;
- авторизація користувача;
- можливість виходу з системи;
- додавання та редагування завдань для планування робочого часу;
- налаштування персоналізованого робочого розкладу з використанням різних методик тайм-менеджменту (наприклад, Pomodoro, 52/17, 90/30, 1-3-5, правило двох хвилин);
- інтеграція з популярними календарними сервісами (Google Calendar, Apple Calendar) для синхронізації завдань та подій;

– моніторинг виконання завдань за запитом користувача;

– автоматизований моніторинг виконання завдань за встановленим розкладом;

– оповіщення про наближення дедлайнів, завершення робочих інтервалів або зміни в розкладі;

– збереження історії оповіщень та даних про виконання завдань;

– перегляд детальної аналітики щодо продуктивності та тривалості виконання завдань.

Таким чином, реалізація вебзастосунку для планування робочого часу «TaskBalance» дозволяє забезпечити ефективне планування та контроль робочого часу, що є важливим фактором підвищення продуктивності в сучасних умовах діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методи тайм-менеджменту для ефективного розподілу часового ресурсу. URL: https://gudhub.com.ua/blog/marketing/metody-taym-menedzhmentu-porady-upravlinnya-chasom/ (дата звернення: 20.02.2025).

2. Що таке метод Pomodoro? URL: https://experience.dropbox.com/uk-ua/resources/pomodoro-technique (дата звернення: 21.02.2025).

3. Тайм-менеджмент. Методи управління часовим ресурсом. URL: https://lb.ua/society/2022/02/22/506318_taymmenedzhment_metodi_upravlinnya.html (дата звернення: 21.02.2025).

4. Вебзастосунок «Pomofocus». URL: https://pomofocus.io/ (дата звернення: 25.02.2025).

5. Вебзастосунок «Focus To-Do». URL: https://www.focustodo.cn/ (дата звернення: 25.02.2025).

6. Вебзастосунок «Todoist». URL: https://www.todoist.com/ (дата звернення: 25.02.2025).

7. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 10.03.2025).

8. Архітектура програмного забезпечення: все, що треба знати. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 20.03.2025).

9. Що таке JavaScript і для чого він потрібен. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення: 19.04.2025).

10. Що таке React JS і для чого він потрібен? URL: https://dan-it.com.ua/uk/blog/chto-takoe-react-js-i-dlja-chego-on-nuzhen/ (дата звернення: 19.04.2025).

11. Що таке MongoDB? URL: https://www.guru99.com/uk/what-is-mongodb.html (дата звернення: 19.04.2025).

12. Visual Studio Code. URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 19.04.2025).

13. Про охорону праці: Закон України № 2695-XII від 14.10.1992. Дата оновлення: 01.01.2025. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text (дата звернення: 05.03.2025).

14. Санітарні норми та правила при роботі з відеодисплейними терміналами. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=6007 (дата звернення: 21.03.2025).

15. Про затвердження Державних санітарних норм і правил роботи з джерелами електромагнітного випромінювання: Наказ МОЗ України № 400 від 30.07.2012. URL: https://zakon.rada.gov.ua/laws/show/z1430-12#Text (дата звернення: 21.03.2025).

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Найменування програми - вебзастосунок для планування робочого часу «TaskBalance».

Область застосування - автоматизація процесу управління часом для окремих користувачів та організацій, що дозволяє ефективно організовувати робочий день, інтегрувати сучасні методики тайм-менеджменту та синхронізувати завдання з популярними календарними сервісами.

1 Підстави для розробки

Підставою для розробки вебзастосунку «TaskBalance» є наказ на затвердження тем кваліфікаційних робіт бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням вебзастосунку «TaskBalance» є автоматизація процесів управління часом та планування завдань, що включає:

– реєстрацію та авторизацію користувачів;
– створення, редагування та видалення завдань;
– інтеграцію з календарними сервісами (Google Calendar, Apple Calendar тощо) для синхронізації розкладу;
– збір та аналіз даних про виконання завдань та продуктивність;
– створення розпису у вигляді календаря.

2.2 Експлуатаційне призначення

Експлуатаційним призначенням вебзастосунку «TaskBalance» є планування робочого часу користувачів, що дозволяє ефективно організовувати робочі процеси, контролювати виконання завдань та здійснювати їх аналіз. Завдяки інтеграції з популярними календарними сервісами, вебзастосунок забезпечує синхронізацію розкладу та своєчасні нагадування, що сприяє підвищенню продуктивності та зниженню рівня стресу під час роботи.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Вебзастосунок «TaskBalance» повинен виконувати наступні функції

Авторизація користувача

Забезпечення авторизації користувача в програмі для надання доступу до всіх функціональних можливостей. Обов’язковими полями є: логін та пароль.

Перегляд даних

Для перегляду даних користувач обирає відповідну сторінку інтерфейсу (наприклад, «Завдання», «Користувачі», «Інтеграції», «Аналітика», «Історія змін»). Кожна сторінка містить структуроване відображення інформації, що дозволяє швидко орієнтуватися в системі.

Додавання нового запису

Для додавання нового запису (наприклад, створення нового завдання чи інтеграції) користувач переходить на відповідну сторінку і натискає кнопку «Додати». Відкривається форма для заповнення необхідних полів, після чого дані зберігаються в базі даних.

Редагування запису

Редагування здійснюється шляхом вибору конкретного запису на відповідній сторінці (наприклад, завдання або профілю користувача). Після цього користувач отримує можливість змінити значення атрибутів через форму редагування, а внесені зміни зберігаються в системі.

Видалення запису

Для видалення запису користувач обирає потрібну сторінку та конкретний запис, після чого виконується підтвердження операції видалення. Це дозволяє видаляти як окремі завдання, так і записи про інтеграції чи інші дані.

Формування нагадувань і управління завданнями

Функціонал управління завданнями включає можливість встановлення нагадувань про наближення дедлайнів та зміну статусів завдань (наприклад, перехід від «нове» до «в процесі» чи «виконано»). Це сприяє організації робочого процесу та планування робочого часу.

Формування звітів

Автоматичне формування звітів відбувається після вибору користувачем необхідних параметрів. Вебзастосунок аналізує дані про виконання завдань, обчислює відсоток виконання, а також генерує аналітичні звіти, що дозволяють оцінити ефективність управління часом і робочими процесами.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Дані зберігаються в базі даних MongoDB - нереляційній системі управління базами даних з довільним доступом.

- Реєстрація користувача:
  - вхідні дані: ім’я, пароль, електронна адреса;
  - вихідні дані: створення нового запису в базі даних користувачів.
- Авторизація користувача:
  - вхідні дані: електронна адреса, пароль;
  - вихідні дані: генерація унікального ідентифікатора сесії.
- Додавання нового завдання:
  - вхідні дані: назва завдання, опис, дедлайн, категорія, пріоритет;
  - вихідні дані: збереження інформації про завдання у базі даних.
- Моніторинг стану завдань:
  - вхідні дані: дані про завдання, збережені в базі (статус виконання, залишковий час до дедлайну);

– вихідні дані: оновлення статусу завдання, відображення статистики виконання та збереження актуальних даних у системі.

– Оповіщення про зміни в завданнях:

– вхідні дані: електронна адреса користувача, ідентифікатор завдання, інформація про зміни (зміна дедлайну чи пріоритету);

– вихідні дані: надсилання сповіщення (електронного або внутрішнього) з актуальною інформацією.

– Профіль користувача:

– вхідні дані: додаткові дані, такі як номер телефону, фотографія профілю;

– вихідні дані: розширений запис у профілі користувача для подальшої персоналізації.

– Аналітика продуктивності:

– вхідні дані: дані про виконання завдань (час виконання, затримки, відхилення від плану);

– вихідні дані: зведені звіти та діаграми, що допомагають оцінити ефективність розкладу.

– Налаштування сповіщень:

– вхідні дані: вибір користувача щодо типу сповіщень (SMS, електронна пошта) та їх частоти;

– вихідні дані: збереження індивідуальних налаштувань сповіщень у системі.

3.2 Надійність вебзастосунку

Програмний продукт має забезпечувати стабільну роботу за умови справної роботи апаратного забезпечення. Якщо в процесі експлуатації виникають збої, система повинна автоматично відновлювати свою роботу після перезапуску вебзастосунку. Для збереження даних використовується MongoDB, що гарантує базовий рівень захисту інформації.

При випадковій втраті окремих даних система продовжує працювати коректно, або, якщо це неможливо, відразу повідомляє користувача про виниклу проблему. Також, у разі введення неправильних даних, вебзастосунок виводить повідомлення про помилку та надає можливість оперативного її виправлення.

3.3 Вимоги до умов експлуатації

Користувачі повинні мати базові знання роботи з комп’ютером. Система експлуатується на комп’ютерах, які працюють у контрольованому середовищі з оптимальними умовами опалення та вентиляції, що сприяє стабільній роботі апаратного забезпечення.

3.4 Вимоги до складу та параметрів технічних засобів

Оскільки вебзастосунок «TaskBalance» працює у браузері, його експлуатація не потребує спеціального апаратного забезпечення. Рекомендується використовувати сучасний браузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari тощо) з актуальною версією для забезпечення коректної роботи.

Для оптимальної роботи вебзастосунку потрібен стабільний доступ до Інтернету. На стаціонарних комп’ютерах достатньо ОС Windows 10 (або новішої), macOS або Linux з мінімум 4 ГБ оперативної пам’яті. Для мобільних пристроїв рекомендується використовувати iOS 12 або Android 8 і вище.

4 Вимоги до програмної документації

Програмна документація вебзастосунку «TaskBalance» повинна включати наступні компоненти:

– технічне завдання, що містить опис мети, функціональних вимог та специфікації вебзастосунку;

– текст програми, з вихідним кодом та коментарями, що пояснюють його роботу;

– опис програми, який містить детальний опис архітектури, логіки роботи та використовуваних технологій;

– інструкцію користувача, що забезпечує детальні рекомендації з встановлення, налаштування та використання вебзастосунку;

– програму і методику випробувань програмного забезпечення, що включає план тестування, результати тестових сценаріїв та рекомендації щодо усунення виявлених помилок.

5 Техніко-економічні показники

Техніко-економічні показники не розраховуються, оскільки розробка вебзастосунку «TaskBalance» здійснюється в рамках кваліфікаційної роботи.

6 Стадії та етапи розробки

Стадії та етапи розробки вебзастосунку «TaskBalance» представлені в таблиці А.1.

Таблиця А.1 - Стадії та етапи розробки вебзастосунку «TaskBalance»

№ п/п	Назва етапів роботи	Терміни виконання
1	Аналіз практичної задачі інженерії ПЗ	31.03.2025
2	Аналіз вимог до вебзастосунку «TaskBalance»	07.04.2025
3	Розробка архітектури вебзастосунку	21.04.2025
4	Розробка проекту вебзастосунку	05.05.2025
5	Реалізація та тестування вебзастосунку	12.05.2025

7 Порядок контролю та приймання

Контроль за розробкою здійснюється на кожному етапі - від аналізу до тестування - з урахуванням вимог, визначених у технічному завданні. Кожна стадія розробки повинна бути виконана в обумовлені строки та погоджена із замовником.

Приймання продукту відбувається згідно з програмою та методикою випробувань. Результати тестування документуються за допомогою протоколів, що містять інформацію про проведені випробування та виявлені недоліки. У разі виявлення помилок складається акт з описом виявлених недоліків, який підписується представниками як замовника, так і розробника, та затверджується

керівниками відповідних організацій. Розробник зобов’язаний усунути зазначені недоліки протягом не більше ніж двох тижнів та повідомити замовника про готовність до повторної перевірки.

ДОДАТОК Б - КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лістинг Б.1 - Сервіс керування завданнями TaskService.js

```
import { v4 as uuidv4 } from 'uuid';

export const TaskService = {
getTasks(user) {
const tasks = JSON.parse(localStorage.getItem(`tasks_${user.userId}`)) || [];
return tasks;
},

createTask(user, title, desc, prio, status, dueDate) {
const newTask = {
taskId: uuidv4(),
title,
description: desc,
priority: prio,
status,
dueDate,
};
const tasks = this.getTasks(user);
tasks.push(newTask);
localStorage.setItem(`tasks_${user.userId}`, JSON.stringify(tasks));
return newTask;
},

updateTask(user, updatedTask) {
let tasks = this.getTasks(user);
tasks = tasks.map(task => task.taskId === updatedTask.taskId ? updatedTask :
task);
localStorage.setItem(`tasks_${user.userId}`, JSON.stringify(tasks));
},

deleteTask(user, taskId) {
const tasks = this.getTasks(user).filter(task => task.taskId !== taskId);
localStorage.setItem(`tasks_${user.userId}`, JSON.stringify(tasks));
}
};
```

Лістинг Б.2 - Сервіс таймера TimerService.js

```
export const TimerService = {
timer: null,
startTime: null,
remainingTime: 0,
isRunning: false,
defaultDuration: 25 * 60,

start(durationInSeconds, onTick, onComplete) {
if (this.isRunning) return;

this.remainingTime = durationInSeconds || this.defaultDuration;
this.startTime = Date.now();
this.isRunning = true;

this.timer = setInterval(() => {
const now = Date.now();
const elapsed = Math.floor((now - this.startTime) / 1000);
const timeLeft = this.remainingTime - elapsed;
```

```
if (timeLeft <= 0) {
clearInterval(this.timer);
this.isRunning = false;
onComplete && onComplete();
} else {
onTick && onTick(timeLeft);
}
}, 1000);
},

pause() {
if (!this.isRunning) return;
clearInterval(this.timer);
this.remainingTime -= Math.floor((Date.now() - this.startTime) / 1000);
this.isRunning = false;
},

resume(onTick, onComplete) {
if (this.isRunning || this.remainingTime <= 0) return;
this.start(this.remainingTime, onTick, onComplete);
},

reset() {
clearInterval(this.timer);
this.timer = null;
this.remainingTime = this.defaultDuration;
this.isRunning = false;
}
};
```

Лістинг Б.3 - Сервіс розкладу ScheduleService.js

```
export const ScheduleService = {
getSchedule(user) {
const  schedule  =  JSON.parse(localStorage.getItem(`schedule_${user.userId}`))  ||
[];
return schedule;
},

addToSchedule(user, task) {
const schedule = this.getSchedule(user);
schedule.push({
taskId: task.taskId,
title: task.title,
dueDate: task.dueDate,
priority: task.priority,
});
localStorage.setItem(`schedule_${user.userId}`, JSON.stringify(schedule));
},

removeFromSchedule(user, taskId) {
const schedule = this.getSchedule(user).filter(entry => entry.taskId !== taskId);
localStorage.setItem(`schedule_${user.userId}`, JSON.stringify(schedule));
},

updateSchedule(user, updatedTask) {
let schedule = this.getSchedule(user);
schedule = schedule.map(entry => {
if (entry.taskId === updatedTask.taskId) {
return {
...entry,
```

```
title: updatedTask.title,
dueDate: updatedTask.dueDate,
priority: updatedTask.priority,
};
}
return entry;
});
localStorage.setItem(`schedule_${user.userId}`, JSON.stringify(schedule));
}
```

Лістинг Б.4 - Сервіс сповіщень NotificationService.js

```
export const NotificationService = {
scheduleNotification(task) {
const now = new Date();
const due = new Date(task.dueDate);
const diff = due.getTime() - now.getTime();

if (diff <= 0) return;

setTimeout(() => {
alert(`Нагадування: завдання "${task.title}" має бути виконане до
${task.dueDate}`);
}, diff);
},

scheduleAll(user, tasks) {
tasks.forEach(task => {
if (task.dueDate && task.status !== 'completed') {
this.scheduleNotification(task);
}
});
}
};
```

Лістинг Б.5 - Компонент TaskDetailsPage.jsx

```
import React, { useEffect, useState } from 'react';
import { TaskService } from '../services/TaskService';

const TaskDetailsPage = ({ user, taskId, onBack }) => {
const [task, setTask] = useState(null);

useEffect(() => {
const tasks = TaskService.getTasks(user);
const selected = tasks.find(t => t.taskId === taskId);
setTask(selected);
}, [taskId, user]);

const handleChange = (e) => {
const { name, value } = e.target;
setTask(prev => ({ ...prev, [name]: value }));
};

const handleSave = () => {
TaskService.updateTask(user, task);
onBack && onBack();
};

if (!task) return <p>Loading...</p>;
```

```
return (
<div>
<h2>Edit Task</h2>
<label>Title:</label>
<input name="title" value={task.title} onChange={handleChange} /><br />
<label>Description:</label>
<textarea   name="description"   value={task.description}   onChange={handleChange}
/><br />
<label>Priority:</label>
<select name="priority" value={task.priority} onChange={handleChange}>
<option value="low">Low</option>
<option value="medium">Medium</option>
<option value="high">High</option>
</select><br />
<label>Status:</label>
<select name="status" value={task.status} onChange={handleChange}>
<option value="todo">To Do</option>
<option value="in_progress">In Progress</option>
<option value="completed">Completed</option>
</select><br />
<label>Due Date:</label>
<input  type="date"  name="dueDate"  value={task.dueDate}  onChange={handleChange}
/><br />
<button onClick={handleSave}>Save</button>
<button onClick={onBack}>Back</button>
</div>
);
};

export default TaskDetailsPage;
```

Лістинг Б.6 - Сервіс аналітики AnalyticsService.js

```
export const AnalyticsService = {
getSummary(user) {
const tasks = JSON.parse(localStorage.getItem(`tasks_${user.userId}`)) || [];
const total = tasks.length;
const completed = tasks.filter(t => t.status === 'completed').length;
const overdue = tasks.filter(t => new Date(t.dueDate) < new Date() && t.status !==
'completed').length;
const inProgress = tasks.filter(t => t.status === 'in_progress').length;

return {
total,
completed,
overdue,
inProgress,
completedPercent: total > 0 ? Math.round((completed / total) * 100) : 0
};
},

getChartData(user) {
```

```
const tasks = JSON.parse(localStorage.getItem(`tasks_${user.userId}`)) || [];
const dailyStats = {};

tasks.forEach(task => {
const day = new Date(task.dueDate).toLocaleDateString();
if (!dailyStats[day]) dailyStats[day] = 0;
if (task.status === 'completed') dailyStats[day]++;
});

return Object.entries(dailyStats).map(([date, count]) => ({ date, count }));
      }
```

ДОДАТОК В - ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Загальні відомості

Вебзастосунок для планування часу «TaskBalance» призначений для ефективного планування часу, управління завданнями та підвищення продуктивності. Програмне забезпечення надає можливість створювати розклади на основі популярних методик тайм-менеджменту, відстежувати виконання завдань, отримувати сповіщення про дедлайни та аналізувати ефективність роботи.

Для розробки програмного забезпечення були обрані наступні засоби:

- мова програмування: JavaScript;
- фреймворк для фронтенду: React;
- база даних: MongoDB;
- середовище розробки: Visual Studio Code.

Застосунок функціонує у веббраузері, що робить його доступним для будь-якого пристрою з підключенням до Інтернету. Взаємодія з серверною частиною відбувається за допомогою REST API, що забезпечує швидкий обмін даними з MongoDB.Для функціонування програмного забезпечення достатньо будь-якого пристрою з доступом до Інтернету та сучасного браузера (Google Chrome, Firefox, Microsoft Edge, Safari).

2 Функціональне призначення

Вебзастосунок для планування часу TaskBalance виконує наступні основні функції:

- реєстрація та авторизація користувачів;
- додавання та редагування завдань для планування робочого часу;
- налаштування персоналізованого робочого розкладу з використанням різних методик тайм-менеджменту (Pomodoro, 52/17, 90/30, 1-3-5, правило двох хвилин);

– інтеграція з популярними календарними сервісами (Google Calendar, Apple Calendar) для синхронізації завдань та подій;

– моніторинг виконання завдань за запитом користувача або автоматизовано;

– оповіщення про наближення дедлайнів, завершення робочих інтервалів або зміни в розкладі;

– збереження історії оповіщень та даних про виконання завдань;

– перегляд детальної аналітики щодо продуктивності та тривалості виконання завдань.

3 Опис логічної структури

Вебзастосунок TaskBalance побудований на основі трирівневої клієнт-серверної архітектури, яка включає наступні рівні:

Рівень представлення (Presentation Layer)

Рівень представлення відповідає за взаємодію з користувачем та відображення інтерфейсу. Він реалізований на основі сучасних вебтехнологій, таких як HTML, CSS, JavaScript, з використанням бібліотеки React. Саме на цьому рівні відбувається введення даних користувачем, їх візуалізація та відправка запитів на сервер.

Рівень логіки застосунку (Application Logic Layer)

Цей рівень реалізований у вигляді серверної частини, що обробляє бізнес-логіку застосунку. На цьому рівні відбувається:

– управління задачами користувача (додавання, редагування, видалення);

– формування розкладу на основі методик тайм-менеджменту;

– інтеграція з календарними сервісами (Google Calendar, Apple Calendar);

– моніторинг виконання завдань;

– управління сповіщеннями про дедлайни та завершення інтервалів роботи.

Вся логіка реалізована у вигляді окремих сервісів (Service), що працюють з моделями (Model) та репозиторіями (Repository), організованими за принципом MVC (Model-View-Controller).

Рівень зберігання (Data Access Layer)

Для зберігання інформації використовується база даних MongoDB, яка забезпечує надійне та швидке збереження даних про користувачів, задачі, розклад та аналітику. Доступ до даних здійснюється через абстрактний рівень доступу (Repository Pattern), що дозволяє ізолювати бізнес-логіку від конкретних запитів до бази даних.

Така структура забезпечує:

– гнучкість та масштабованість - кожен рівень може змінюватися та масштабуватися незалежно;

– безпечність - доступ до бази даних відбувається лише через захищені серверні методи;

– зручність у розробці та підтримці - чітке розділення обов’язків між рівнями дозволяє легко вносити зміни в один з компонентів, не впливаючи на інші.

4 Технічні засоби

Для коректної роботи вебзастосунку для планування часу TaskBalance необхідний пристрій наступної мінімальної комплекції:

– Процесор: Intel i3 або аналогічний від AMD;

– Оперативна пам’ять: 4 ГБ;

– Дисковий простір: 500 МБ вільного місця для кешу браузера;

– Операційна система: Windows 10+, macOS 10.15+, Linux Ubuntu 18.04+;

– Інтернет-з'єднання: для роботи з базою даних MongoDB та синхронізації даних.

5 Виклик та завантаження

Вебзастосунок для планування часу TaskBalance доступний за веб-адресою (URL) та не вимагає встановлення на пристрій. Для початку роботи достатньо перейти за посиланням та пройти авторизацію або реєстрацію.

Після авторизації користувачу відкривається Головна сторінка з доступом до основних функцій:

– Панель завдань - відображення поточних та запланованих завдань;
– Розклад - інтерактивний календар з відмітками завдань;
– Аналітика - графіки та статистика виконання завдань;

6 Вхідні та вихідні дані

Вхідні дані:

Реєстрація користувача:

– Ім'я, електронна пошта, пароль.

Авторизація користувача:

– Електронна пошта, пароль.

Додавання нового завдання:

– Назва завдання, опис, дедлайн, категорія, пріоритет.

Налаштування розкладу:

– Вибір методики (Pomodoro, 52/17, 90/30, тощо), дати та часу виконання завдань.

Інтеграція з календарем:

– Вхідні дані з Google Calendar або Apple Calendar для синхронізації подій.

Налаштування сповіщень:

– Тип сповіщення (електронна пошта, пуш-повідомлення), періодичність.

Профіль користувача:

– Ім'я, електронна пошта, фото, номер телефону.

Вихідні дані:

Списки завдань:

– Відображення активних, завершених та прострочених завдань.

Розклад завдань:

– Відображення запланованих подій у календарі.

Аналітика продуктивності:

– Графіки, статистика виконаних завдань, порівняння за періодами.

Сповіщення:

– Попередження про дедлайни, завершення інтервалів роботи, зміни в розкладі.

Профіль користувача:

– Персональні дані та налаштування.

ДОДАТОК Г - КЕРІВНИЦТВО КОРИСТУВАЧА

Виконання програми

Запуск вебзастосунку TaskBalance здійснюється через будь-який сучасний веббраузер шляхом введення відповідної адреси у рядок пошуку. На екрані з’являється головна сторінка застосунку, яка містить вступну інформацію та кнопки для переходу до форм реєстрації або авторизації. Вигляд головної сторінки наведено на рисунку Г.1.

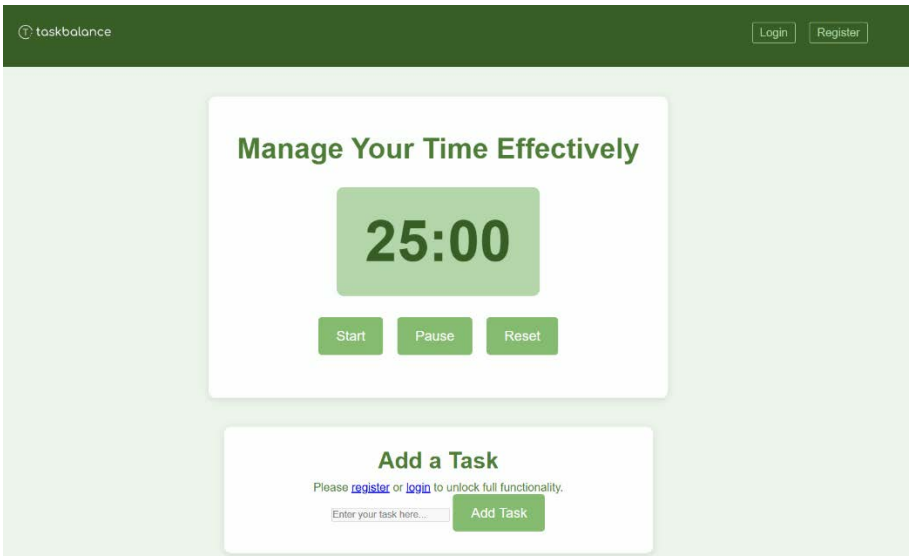

Рисунок Г.1 - Головна сторінка вебзастосунку «TaskBalance»

На рисунку Г.2 представлено форму авторизації користувача. Для входу в систему потрібно ввести зареєстровану електронну адресу та пароль. Після успішної авторизації користувач отримує доступ до особистого кабінету.

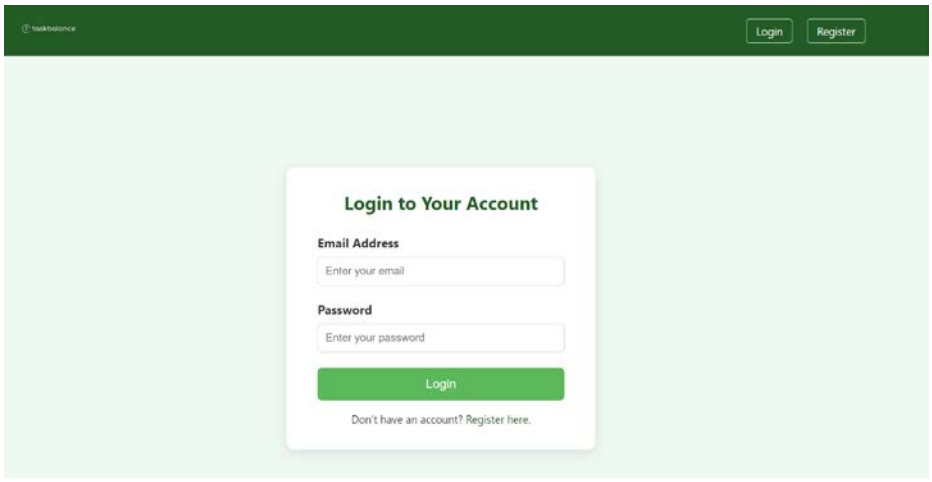

Рисунок Г.2 - Форма авторизації користувача

Якщо обліковий запис відсутній, користувач може створити його за допомогою форми реєстрації, де потрібно вказати ім’я, адресу електронної пошти та пароль. Зовнішній вигляд форми реєстрації зображено на рисунку Г.3.

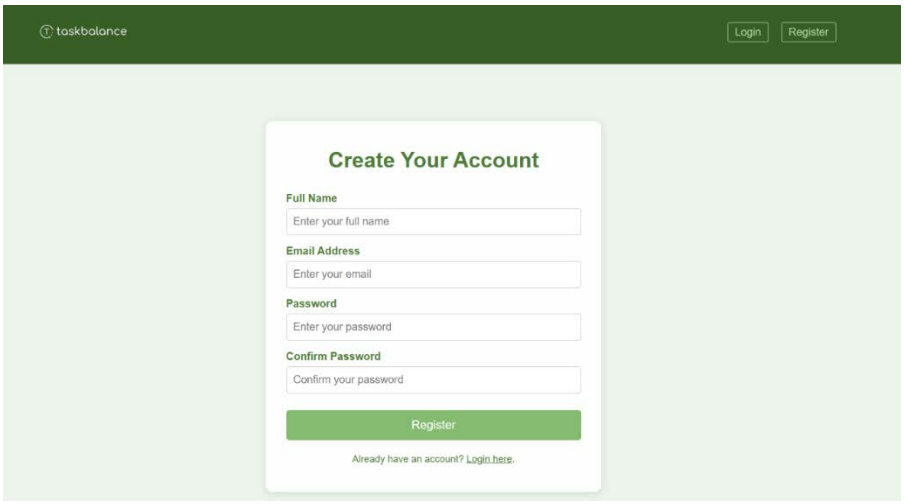

Рисунок Г.3 - Форма реєстрації нового користувача

Після входу до системи користувач потрапляє на сторінку профілю, що містить персональні дані, налаштування інтеграції з календарними сервісами та можливість змінювати параметри сповіщень. Інтерфейс цієї сторінки наведено на рисунку Г.4.

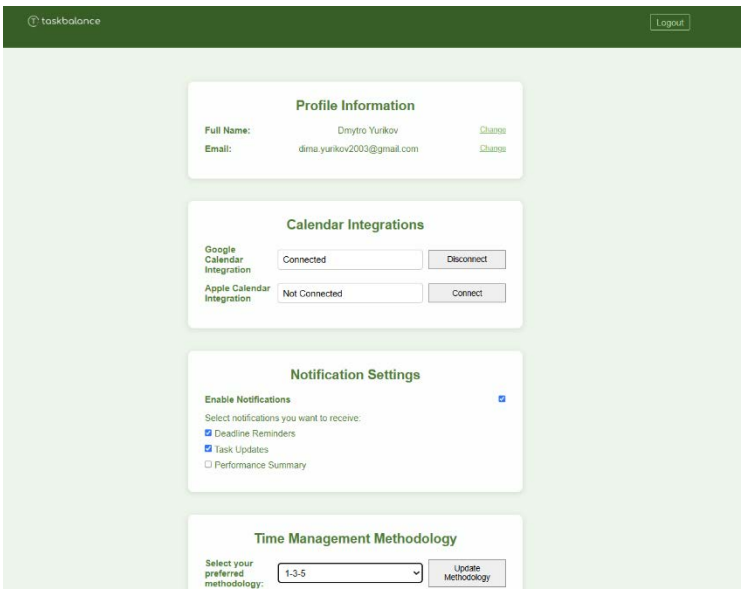

Рисунок Г.4 - Сторінка профілю користувача

Один із ключових функціональних модулів вебзастосунку - це таймер, який дозволяє працювати за методиками тайм-менеджменту (наприклад, Pomodoro) та відстежувати тривалість виконання завдань. Таймер може бути прив’язаний до конкретного завдання. Вигляд модуля таймера показано на рисунку Г.5.

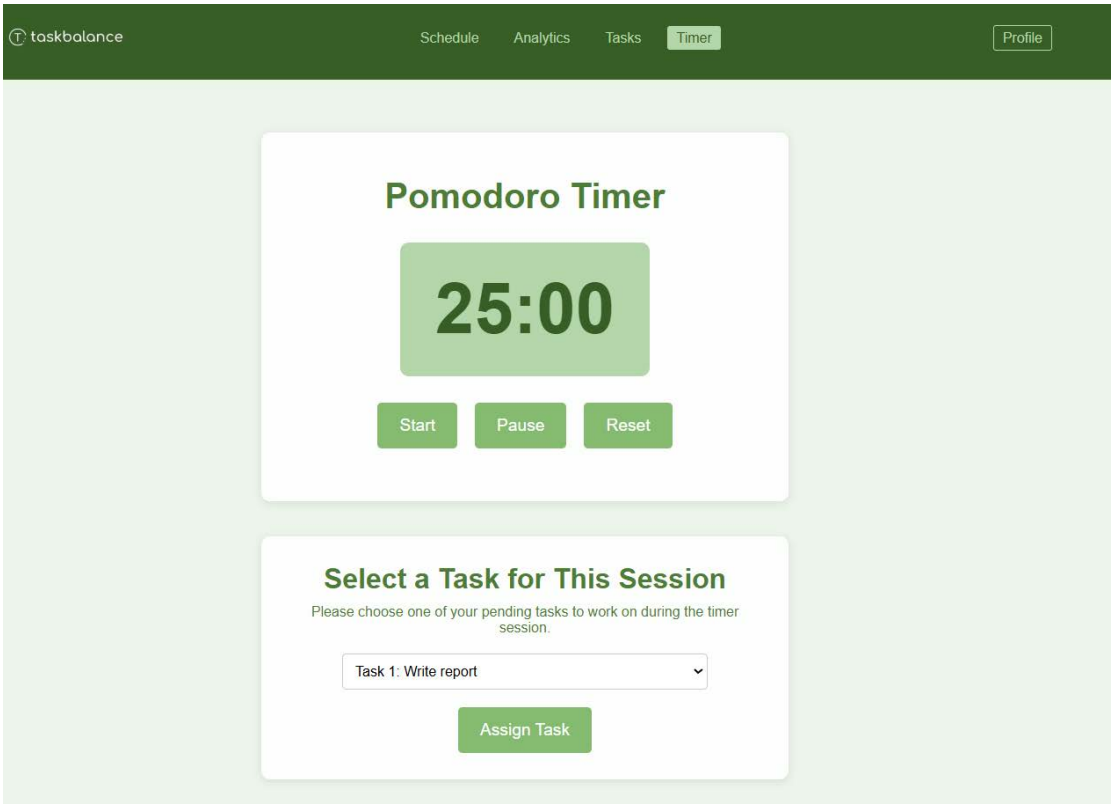

Рисунок Г.5 - Сторінка таймера зареєстрованого користувача

Ще один важливий елемент системи - календар розкладу завдань. На цій сторінці користувач бачить свої завдання у хронологічному порядку. Якщо завдань на обраний день не заплановано, система повідомляє, що вільний час можна використати для відпочинку або перепланування. Інтерфейс модуля розкладу зображено на рисунку Г.6.

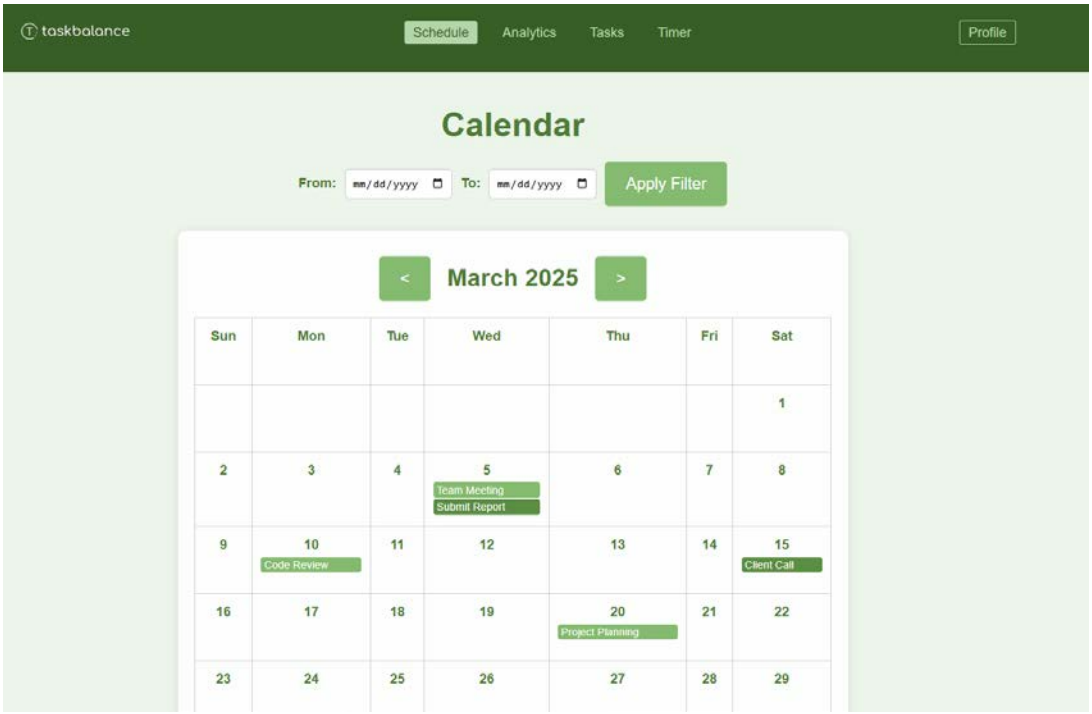

Рисунок Г.6 - Сторінка розкладу завдань користувача

На рисунку Г.7 наведено сторінку управління завданнями, де реалізовано функції створення, редагування, сортування та фільтрації задач за статусом, пріоритетом і дедлайном. Усі завдання автоматично зберігаються в базі даних та доступні для перегляду.

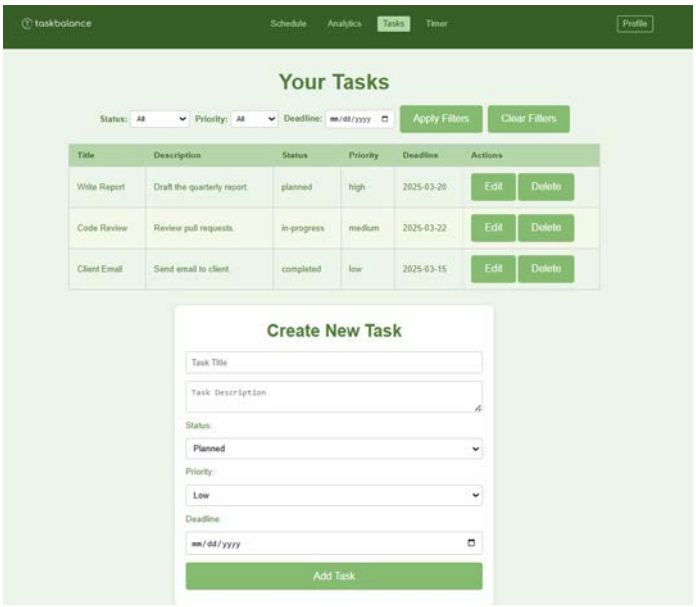

Рисунок Г.7 - Сторінка управління завданнями

Функціональність аналітики продуктивності представлено на рисунку Г.8. Користувач може обрати період аналізу (день, тиждень або місяць) і переглянути кількісні та візуальні дані щодо своєї ефективності - відсоток виконаних завдань, тривалість роботи, регулярність тощо.

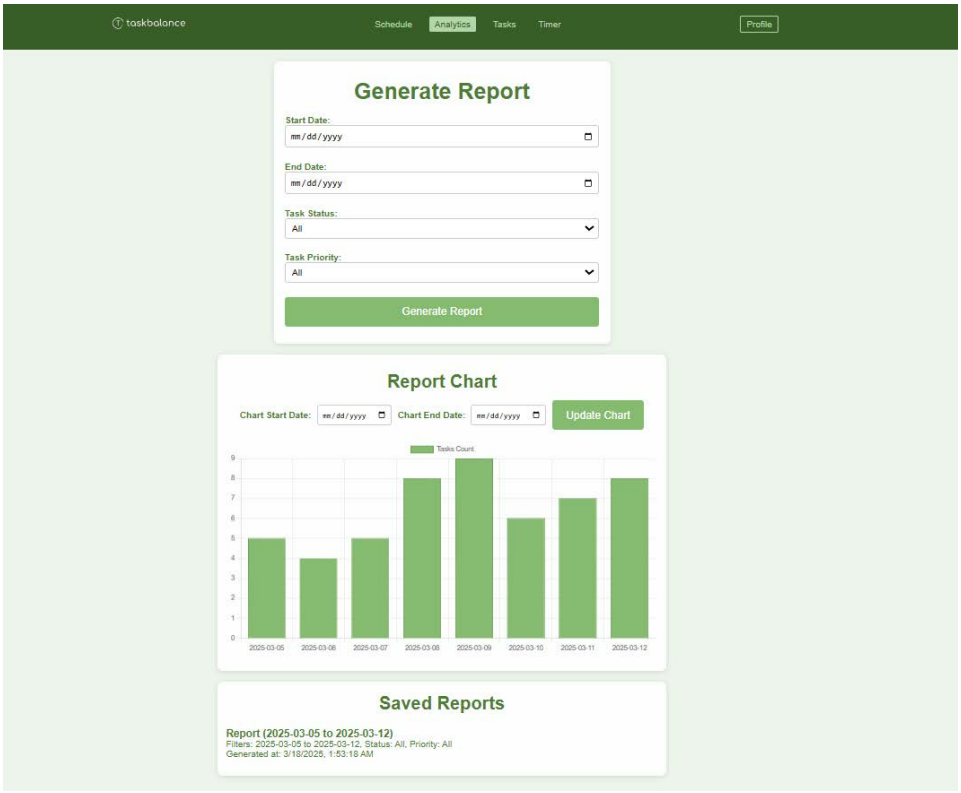

Рисунок Г.8 - Сторінка аналітики продуктивності користувача

Завершення сеансу роботи із системою здійснюється через натискання кнопки «Вийти», яка доступна у головному меню. Після цього користувач повертається на форму авторизації.

ДОДАТОК Д - ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об'єкт випробувань

1.1Найменування випробуваного вебзастосунку

Повна назва об’єкта випробувань: вебзастосунок для планування робочого часу «TaskBalance».

1.2 Область застосування випробовуємого вебзастосунку

Призначення вебзастосунку для планування робочего часу TaskBalance полягає у забезпеченні користувачів зручним та ефективним інструментом для планування робочого часу, управління завданнями, налаштування персоналізованого розкладу та підвищення продуктивності.

Вебзастосунок забезпечує:

- реєстрацію та авторизацію користувачів;
- додавання, редагування та видалення завдань;
- налаштування персоналізованого розкладу з використанням методик тайм-менеджменту (Pomodoro, 52/17, 90/30, 1-3-5, правило двох хвилин);
- інтеграцію з популярними календарними сервісами (Google Calendar, Apple Calendar);
- моніторинг виконання завдань та автоматизований моніторинг за розкладом;
- оповіщення про дедлайни, завершення робочих інтервалів та зміни у розкладі;
- збереження історії оповіщень та даних про виконання завдань;
- перегляд детальної аналітики щодо продуктивності та тривалості виконання завдань.

1.3 Позначення випробуваного вебзастосунку

Вебзастосунок «TaskBalance».

2 Мета випробувань

Метою проведення випробувань є перевірка відповідності функціоналу вебзастосунку TaskBalance вимогам, які викладені у технічному завданні (додаток А), а також забезпечення надійності, зручності використання та відповідності очікуваній продуктивності.

3 Вимоги до вебзастосунку

Вебзастосунок повинен реалізовувати функції, описані в технічному завданні, яке наведено в додатку А, включаючи управління завданнями, налаштування розкладу, моніторинг продуктивності та інтеграцію з зовнішніми календарями.

4 Вимоги до програмної документації

Для проведення випробувань повинна бути надана наступна документація:

– технічне завдання на розробку вебзастосунку;
– текст вебзастосунку (вихідний код);
– опис архітектури та функціональних можливостей;
– інструкція користувача;
– програма і методика випробувань вебзастосунку.

5 Засоби і порядок випробувань

5.1 Програмні засоби випробувань

До програмних засобів, необхідних для випробувань, належать:

– операційна система Windows, MacOS або Linux;
– браузер Google Chrome або Mozilla Firefox (остання версія);
– встановлене серверне середовище Node.js (версія 18 і вище);
– база даних MongoDB (версія 6.0 і вище);
– підключення до Інтернету для інтеграції з календарними сервісами.

5.2 Апаратні засоби випробувань

Система повинна задовольняти наступним мінімальним вимогам:

– процесор: не нижче Intel Core i3 або аналог;
– оперативна пам’ять: не менше 4 ГБ;
– дисковий простір: до 500 МБ вільної пам'яті;
– доступ до мережі Інтернет.

5.3 Порядок проведення випробувань

Проведення випробування вебзастосунку повинно відбуватися у наступному порядку:

– виконання інструкцій щодо кожної функції;
– аналіз отриманих результатів і перевірка відповідності програмного продукту вимогам технічного завдання;
– у випадку виявлення помилок, складається перелік недоліків та визначаються терміни їх усунення. Після виправлення проводиться повторне тестування.

6 Методи випробувань

Тестування проводиться в умовах типового використання, включаючи запуск вебзастосунку, введення даних, обробку подій, а також перевірку повідомлень та сповіщень. Результати тестів фіксуються у вигляді тест-кейсів з урахуванням тестового оточення, залежностей та дій користувача.

Для оцінювання заявлених характеристик вебзастосунку застосовуються методи «чорної скриньки»:

1. Перевірка доступності функціоналу перегляду завдань

Ідентифікатор: TC-TASKS-01

Предмети тестування: Функціонал перегляду завдань.

Специфікація вхідних даних: Натискання на кнопку перегляду завдань.

Специфікація результатів: Список завдань відкривається без помилок.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;
– натиснути на кнопку «Мої завдання»;
– перевірити, що система відображає список завдань для вибору.

2. Перевірка відображення запланованих завдань у розділі розкладу

Ідентифікатор: TC-SCHEDULE-01

Предмети тестування: Відображення запланованих завдань у розкладі.

Специфікація вхідних даних: Відкриття розділу розкладу.

Специфікація результатів: Відображаються лише заплановані завдання.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: У базі мають бути заплановані, виконані та пропущені завдання.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;
– перейти в розділ «Розклад»;
– перевірити, що у списку відображаються лише заплановані завдання.

3. Перевірка доступу до аналітики продуктивності

Ідентифікатор: TC-ANALYTICS-01

Предмети тестування: Аналітика продуктивності.

Специфікація вхідних даних: Натискання на розділ аналітики.

Специфікація результатів: Користувач отримує доступ до звітів і діаграм.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– перейти в розділ «Аналітика»;

– перевірити, що доступні звіти та діаграми.

4. Перевірка доступу до функціоналу управління розкладом

Ідентифікатор: TC-SCHEDULE-MANAGE-01

Предмети тестування: Управління розкладом завдань.

Специфікація вхідних даних: Відкриття розділу "Розклад" та вибір опції "Редагувати розклад".

Специфікація результатів: Користувач отримує можливість додавати, редагувати або видаляти завдання у розкладі.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– перейти в розділ «Розклад»;

– обрати опцію «Редагувати розклад»;

– додати нове завдання, змінити існуюче або видалити виконане;

– перевірити, що зміни відображаються коректно.

5. Перевірка функціоналу створення нової задачі

Ідентифікатор: TC-TASK-CREATE-01

Предмети тестування: Додавання нової задачі.

Специфікація вхідних даних: Натискання кнопки «Додати задачу» та заповнення форми.

Специфікація результатів: Нова задача зберігається у списку завдань та відображається в розкладі.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– натиснути на кнопку «Додати задачу»;

– заповнити форму з інформацією про задачу (назва, опис, дедлайн, пріоритет);

– натиснути «Зберегти»;

– перевірити, що нова задача відображається у списку завдань та в розкладі.

6. Перевірка функціоналу редагування задачі

Ідентифікатор: TC-TASK-EDIT-01

Предмети тестування: Редагування існуючої задачі.

Специфікація вхідних даних: Вибір задачі зі списку та натискання кнопки «Редагувати».

Специфікація результатів: Зміни зберігаються, інформація оновлюється в інтерфейсі.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– обрати задачу зі списку завдань;

– натиснути «Редагувати»;

– внести зміни в поля (наприклад, змінити дедлайн або опис);

– зберегти зміни;

– перевірити, що оновлена інформація відображається коректно.

7. Перевірка функціоналу видалення задачі

Ідентифікатор: TC-TASK-DELETE-01

Предмети тестування: Видалення існуючої задачі.

Специфікація вхідних даних: Вибір задачі та натискання кнопки «Видалити».

Специфікація результатів: Задача видаляється зі списку та не відображається у розкладі.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– обрати задачу зі списку завдань;

– натиснути «Видалити»;

– підтвердити дію;

– перевірити, що задача зникла зі списку та не відображається у розкладі.

8. Перевірка функціоналу оповіщень

Ідентифікатор: TC-NOTIFICATION-01

Предмети тестування: Надсилання сповіщень про заплановані завдання.

Специфікація вхідних даних: Додавання задачі з визначеним дедлайном.

Специфікація результатів: Система надсилає повідомлення за певний час до дедлайну.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– додати нову задачу з дедлайном, наприклад, через годину;

– перевірити, що за 15 хвилин до дедлайну з’являється оповіщення;

– переконатися, що після дедлайну з'являється повідомлення про прострочену задачу (якщо не виконана).

9. Перевірка функціоналу інтеграції з календарями

Ідентифікатор: TC-CALENDAR-01

Предмети тестування: Синхронізація завдань з Google Calendar та Apple Calendar.

Специфікація вхідних даних: Додавання задачі з опцією синхронізації.

Специфікація результатів: Завдання з'являється у підключеному календарі.

Тестове оточення: Вебзастосунок TaskBalance + обраний календар.

Спеціальні процедурні вимоги: Потрібен активний акаунт у Google або Apple.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– додати нову задачу та обрати опцію синхронізації з Google або Apple Calendar;

– перейти до підключеного календаря та перевірити, що задача відображається правильно.

10. Перевірка функціоналу аналітики продуктивності

Ідентифікатор: TC-PERFORMANCE-01

Предмети тестування: Відображення аналітичних звітів.

Специфікація вхідних даних: Відкриття розділу «Аналітика».

Специфікація результатів: Відображаються діаграми та звіти щодо виконання задач.

Тестове оточення: Вебзастосунок TaskBalance.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

– відкрити вебзастосунок;

– перейти в розділ «Аналітика»;

– перевірити, що система відображає статистику виконання завдань та ефективність роботи.