ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для планування особистого часу»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро КОРЛЯКОВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41
Дмитро КОРЛЯКОВ

Керівник: Наталя АНДРУСЕВИЧ
ст.викладач кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Корлякову Дмитру Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для планування особистого часу»
керівник кваліфікаційної роботи ст. викладач кафедри ІПЗ Наталя АНДРУСЕВИЧ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: Теоретичні відомості про принципи тайм-менеджменту та цифрового планування, огляд сучасних web-застосунків для планування особистого часу, вимоги до функціональності та зручності інтерфейсу користувача, технічна документація з описом фреймворків та бібліотек для web-розробки,
методи зберігання та обробки даних у web-застосунках,
матеріали з тестування та оцінки якості web-систем.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд застосунку для планування особистого часу існуючих аналогів, визначення вимог до програмного забезпечення.
2. Огляд та аналіз засобів реалізації застосунку для планування особистого часу
3. Проєктування архітектури застосунку для планування особистого часу
4. Програмна реалізація та тестування застосунку для планування особистого часу

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Алгоритм роботи застосунку.
6. Діаграма класів.
7. Екранні форми.
8. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Огляд та аналіз застосунку для планування особистого часу	14.03-16.03.2025	
4	Проєктування застосунку для планування особистого часу.	17.03-23.03.2025	
5	Програмна реалізація застосунку	24.03-23.04.2025	
6	Тестування застосунку	24.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Дмитро КОРЛЯКОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Наталя АНДРУСЕВИЧ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 48 стор., 3 табл., 23 рис., 20 джерел.

Мета роботи – створення web-застосунку для ефективного планування особистого часу з використанням сучасних технологій C#, ASP.NET Core та SQL Server.

Об'єкт дослідження – процес організації та планування особистого часу користувачів.

Предмет дослідження – програмне забезпечення для планування часу.

Короткий зміст роботи: У роботі проаналізовано підходи до тайм-менеджменту та особливості програмних засобів для самостійного планування. Проведено аналіз популярних сервісів: Google Calendar, Todoist, Notion. На основі дослідження розроблено web-застосунок TimePlanner, який дозволяє створювати розклади, події, нагадування, списки завдань, а також візуалізувати навантаження користувача.

Застосунок реалізовано на основі фреймворку ASP.NET Core із використанням MVC-архітектури. Для побудови інтерфейсу використано Razor Pages та фреймворк Bootstrap, що забезпечують адаптивність та зручність користування. Як СУБД обрано SQL Server, а для взаємодії з нею застосовано Entity Framework Core. Забезпечено повноцінну систему аутентифікації та авторизації за допомогою ASP.NET Identity [13]. Проведено модульне тестування, реалізовано базову систему ролей та захисту персональних даних.

Сфера використання – індивідуальні користувачі, які бажають ефективно планувати особистий час за допомогою надійного та зручного web-застосунку.

КЛЮЧОВІ СЛОВА: C#, ASP.NET CORE, RAZOR PAGES, SQL SERVER, ENTITY FRAMEWORK CORE, MVC, BOOTSTRAP, ASP.NET IDENTITY, ТАЙМ-МЕНЕДЖМЕНТ, WEB-ЗАСТОСУНОК.

ЗМІСТ

ВСТУП

Питання ефективного планування часу є актуальним у сучасних умовах високої динаміки життя та інформаційного навантаження. З кожним роком обсяг справ, завдань і відповідальностей зростає, і часто виникає відчуття, що часу на все просто не вистачає. Паралельно з навчанням доводиться виконувати ще й особисті обов'язки, займатися саморозвитком, шукати баланс між роботою та відпочинком. І коли на горизонті з'являється дедлайн або важлива подія, часто згадуєш про щось у останній момент. У такі миті особливо відчувається цінність добре організованого власного часу.

Ідея створення web-застосунку для планування особистого часу виникла як відповідь на поширену проблему неефективного управління завданнями. Існуючі сервіси — від мобільних додатків до традиційних інструментів на кшталт Google Calendar [11] або паперових щоденників — часто виявляються або надмірно складними у використанні, або, навпаки, недостатньо функціональними. У більшості випадків бракує гнучкості, що ускладнює адаптацію під індивідуальні потреби користувачів.

Проблема якісного планування часу є надзвичайно актуальною в умовах сучасного ритму життя, високого інформаційного навантаження та потреби в багатозадачності. Без ефективного інструменту для організації особистого часу навіть найпростіші завдання можуть накопичуватися, спричиняючи стрес, зниження продуктивності й втрату мотивації.

Обрана тема кваліфікаційної роботи стала спробою розробити універсальний, зручний і функціональний web-застосунок, здатний не лише фіксувати завдання, а й надавати користувачеві інструменти для візуалізації прогресу, аналізу завантаженості та підвищення ефективності власного планування.

Метою було — створити web-застосунок, який допоможе кожному користувачеві ефективніше управляти власним часом, розставляти пріоритети, бачити прогрес та уникати типових помилок у плануванні.

У даній роботі детально описано всі етапи створення такого застосунку: від аналізу існуючих рішень до розробки архітектури, вибору технологій та тестування результату. Особливу увагу приділю тим моментам, які особисто здавались найцікавішими або найскладнішими, поділюсь власними висновками і ідеями для подальшого розвитку проєкту.

Отримані результати можуть бути корисними для інших студентів у процесі навчання та практичного засвоєння матеріалу розробників, а сам застосунок допоможе організовувати власний час максимально ефективно і зручно.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ ОСОБИСТОГО ЧАСУ

1.1 Актуальність теми

У сучасному суспільстві час — це один із найцінніших ресурсів, і правильне його використання часто визначає успіх у багатьох сферах життя. З розвитком цифрових технологій кількість завдань і інформації, яку доводиться обробляти щодня, лише зростає. Постійні зміни у графіках, нові обов'язки, дедлайни й неочікувані справи можуть легко вибити з колії навіть найорганізованішу людину. Саме тому питання раціонального планування часу є надзвичайно актуальним як для студентів, так і для працюючих людей, підприємців, батьків та всіх, хто прагне досягти балансу між різними сферами свого життя.

Особливо гостро це відчувається серед молоді та студентства. Навчання у вищому навчальному закладі часто супроводжується необхідністю поєднувати навчальні завдання, додаткові курси, підробітки, соціальне життя та особистий розвиток. У цьому вирі легко втратити фокус і пропустити щось важливе, якщо немає надійного інструменту для планування.

Сучасні технології дають змогу автоматизувати і спростити процес планування, зробити його більш наочним, інтерактивним і персоналізованим. Однак багато існуючих рішень не завжди відповідають реальним потребам користувачів — або складні в освоєнні, або обмежені у функціоналі, або занадто «загальні», без врахування індивідуальних підходів до організації часу. Часто трапляється так, що людині доводиться одночасно користуватися кількома додатками, вести записи у різних місцях, що ще більше ускладнює планування та знижує його ефективність.

Саме тому розробка простого, інтуїтивно зрозумілого й водночас потужного web-застосунку для планування власного часу є дуже актуальною задачею. Такий інструмент допоможе людям не тільки контролювати виконання своїх справ, а й

аналізувати свою продуктивність, визначати «слабкі місця» у власному розкладі, формувати здорові звички та ефективніше досягати поставлених цілей.

Крім того, саме web-формат застосунку дозволяє бути незалежним від пристрою та операційної системи — планувати свій час можна з будь-якого місця, де є доступ до інтернету. Це робить рішення максимально доступним і зручним для широкого кола користувачів.

Актуальність теми кваліфікаційної роботи полягає у вирішенні реальної проблеми, з якою щоденно стикається велика кількість людей, і у створенні інструменту, який допоможе підвищити якість життя завдяки ефективному використанню часу.

1.2. Мета та завдання дослідження

Мета дослідження полягає у створенні сучасного web-застосунку для планування власного часу, який би був зручним у користуванні, максимально адаптивним до потреб користувача та дозволяв ефективно керувати особистими справами, завданнями й пріоритетами. Метою розробки стало створення інструменту, який дозволяє не лише формувати списки справ, але й здійснювати аналіз витраченого часу, відстежувати прогрес і, як наслідок, підвищувати рівень контролю над особистим життям. Ідея проєкту сформувалась як відповідь на реальну потребу в зручному сервісі для організації часу, який би не обмежувався базовими функціями типових "to do"-застосунків. Передбачалося, що такий застосунок допомагатиме користувачеві ставити цілі, відстежувати їх виконання, виявляти проблемні зони та за потреби змінювати підходи до планування.

Для досягнення цієї мети були поставлені наступні завдання дослідження:

1. Проаналізувати існуючі інструменти та сервіси для планування часу, визначити їхні сильні та слабкі сторони, а також виявити потреби користувачів, які залишаються не задоволеними.
2. Сформулювати основні вимоги до функціоналу майбутнього застосунку на основі особистого досвіду та результатів аналізу аналогів.

3. Розробити архітектуру web-застосунку, визначити оптимальну структуру бази даних та логіку взаємодії між компонентами системи.
4. Вибрати сучасні технології для реалізації (C#, ASP.NET, Entity Framework, SQL Management Studio), які дозволять забезпечити надійність, гнучкість та масштабованість рішення.
5. Реалізувати основний функціонал: створення, редагування та видалення завдань, встановлення пріоритетів, відстеження статусу виконання, можливість переглядати календар та статистику.
6. Забезпечити простий та зрозумілий інтерфейс для користувача, щоб робота із застосунком не вимагала спеціальних знань або тривалого навчання.
7. Провести тестування застосунку на різних етапах розробки для виявлення помилок і недоліків, а також оцінити зручність та ефективність роботи з ним.
8. Оцінити результати і сформулювати висновки щодо подальшого розвитку проекту, можливих напрямків удосконалення та розширення функціоналу.

Виконання поставлених завдань сприяло поглибленню знань у сфері web-розробки та надало практичний досвід створення програмного продукту, орієнтованого на вирішення реальних задач і потенційно корисного для широкого кола користувачів.

1.3. Об’єкт і предмет дослідження

Об’єктом дослідження у даній роботі виступають інформаційні системи, які призначені для організації, управління та планування особистого часу користувачів. Мова йде саме про ті цифрові інструменти, що допомагають людині ефективно контролювати власний розклад, фіксувати завдання, встановлювати пріоритети та стежити за виконанням планів. До таких систем належать як настільні, так і web-додатки, мобільні сервіси, які з кожним роком стають дедалі популярнішими у повсякденному житті.

Предметом дослідження є принципи, методи та технології створення саме web-застосунку для планування власного часу, а також особливості розробки його

функціоналу й користувацького інтерфейсу. У фокусі уваги — вибір оптимальних технічних рішень, моделювання структури даних, організація зручної та інтуїтивної взаємодії з користувачем, забезпечення швидкодії та стабільності системи.

На практиці це означає, що основна увага приділяється питанням:

- як саме організувати процес додавання, редагування та контролю завдань;
- яким чином відображати календарні події й пріоритети;
- як забезпечити простоту використання та гнучкість налаштувань для різних типів користувачів.

Крім того, предмет дослідження охоплює питання застосування сучасних технологій розробки web-додатків (зокрема, C#, ASP.NET, Entity Framework, SQL Management Studio), а також аналізує досвід використання таких систем у реальних умовах.

Загалом, у роботі детально розглядається процес створення ефективного інструменту для планування часу — від ідеї до готового рішення, що може принести реальну користь у повсякденному житті.

1.4. Методика дослідження

У роботі поєднано теоретичний аналіз існуючих рішень із практичним досвідом розробки власного web-застосунку. Вся методика дослідження будувалася на кількох основних етапах, які органічно поєднувалися між собою.

Першим етапом стала детальна оцінка наявних на ринку сервісів для планування часу шляхом аналізу їх функціональних можливостей, зручності використання, наявних обмежень, а також вивчення досвіду реальних користувачів. Окрім офіційних описів, важливими джерелами інформації стали відкриті відгуки в інтернеті, думки користувачів із близького оточення, а також практичний досвід експлуатації подібних рішень. Проведений аналіз дав змогу чітко визначити перелік актуальних проблем, які не охоплюються більшістю

існуючих застосунків, і сформувати пріоритетний перелік функцій для реалізації в першочерговому порядку.

На основі проведеного аналізу аналогів і зібраних спостережень було сформовано перелік функціональних та нефункціональних вимог до майбутнього застосунку. Особливу увагу на цьому етапі приділяли забезпеченню простоти та інтуїтивної зрозумілості інтерфейсу, мінімалістичному дизайну, а також гнучкості та масштабованості системи з урахуванням можливого розширення функціональності в майбутньому. Було створено попередні ескізи інтерфейсу та визначено базову структуру бази даних.

Для реалізації проєкту було обрано мову програмування C#, фреймворк ASP.NET Core та ORM-технологію Entity Framework Core. Цей стек забезпечує високу продуктивність, надійність та можливість масштабування web-застосунку. Як систему керування базами даних було обрано SQL Server, що добре інтегрується з обраним технологічним стеком і забезпечує ефективне зберігання та організацію даних. Додатковою перевагою стала наявність попереднього досвіду роботи з цією СУБД.

Безпосередня розробка web-застосунку. Основний етап роботи полягав у поетапній реалізації всіх запланованих функцій: від створення користувацького інтерфейсу до налаштування взаємодії з базою даних. У процесі розробки було використовував сучасні підходи — розділення логіки на окремі модулі, повторне використання коду, принципи MVC (Model-View-Controller).

Після реалізації основного функціоналу було проведено комплексне тестування застосунку на різних етапах розробки. Перевірялися коректність роботи окремих функцій, зручність інтерфейсу для кінцевого користувача, а також стійкість системи до нетипових або помилкових дій з боку користувача. У процесі тестування виявлялися окремі недоліки, які оперативно усувалися. Частина рішень коригувалася безпосередньо в ході перевірки, з урахуванням нововиявлених нюансів у поведінці системи.

Заключним етапом стала оцінка ефективності розробленого застосунку щодо вирішення поставлених завдань. Було проаналізовано, наскільки зручною та функціонально повною є система в умовах реального використання. На основі отриманих результатів сформовано попередні рекомендації та пропозиції щодо можливих напрямів подальшого розвитку і вдосконалення застосунку.

Ці етапи взаємопов’язані, і багато рішень приймалися на основі реального досвіду й власних спостережень за тим, як працюють існуючі сервіси й як краще організувати власний час. Завдяки такому підходу вдалося не лише теоретично опрацювати тему, а й створити реально діючий інструмент для вирішення щоденних завдань.

2 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

2.1. Аналіз існуючих рішень

На підготовчому етапі основною метою стало дослідження існуючих інструментів для планування часу з акцентом на їхні функціональні особливості, переваги, недоліки та відмінності у підходах до організації особистого простору користувача. Це допомогло не тільки краще зрозуміти потреби користувачів, а й уникнути повторення чужих помилок.

Серед найбільш відомих інструментів для планування часу можна виділити такі: Google Calendar [11], Microsoft To Do, Trello, Todoist[9] та кілька інших менш відомих додатків. Усі вони мають спільну мету — допомогти користувачам керувати своїми справами, але підходи до цього у кожного сервісу свої.

Розглянемо деякі з них:

1. Google Calendar [11] — зручний для роботи з календарними подіями, дозволяє створювати нагадування, працювати з повторюваними подіями, але не завжди підходить для управління складними завданнями з підзадачами.
2. Microsoft To Do — простий додаток для списків справ, інтегрується з іншими сервісами Microsoft, має базовий функціонал, але не надає багато можливостей для аналітики чи детальної кастомізації.
3. Trello — більше орієнтований на командну роботу, організовує завдання у вигляді дошок і карток, підходить для візуального планування, але для особистого використання може бути занадто «масштабним».
4. Todoist[9] — популярний інструмент для особистого та командного користування, має зручну систему фільтрів, але більшість розширених функцій доступні тільки у платній версії.

Таблиця 2.1

Порівняльний аналіз популярних сервісів для планування часу

Сервіс	**Основні можливості**	**Недоліки**	**Для кого підходить**
Google Calendar	Календар, нагадування, повторювані події	Мало функцій для роботи із завданнями	Студенти, офісні працівники
Microsoft To Do	Списки справ, інтеграція з Office	Обмежений функціонал	Ті, хто користується Office
Trello	Дошки, картки, командна робота	Надмірна складність для особистого плану	Команди, проект-менеджери
Todoist	Теги, фільтри, статистика	Деякі функції тільки у Pro-версії	Широкий спектр користувачів
Notion	База знань, нотатки, завдання	Потребує часу на налаштування	Творчі люди, організації

Проведене дослідження засвідчило, що жоден із проаналізованих сервісів не є універсальним рішенням для особистого планування часу у форматі, який би повністю відповідав визначеним вимогам. Найбільше бракує простоти, гнучкості налаштувань і можливості швидко додавати та змінювати завдання без зайвих дій, наведено в таблиці 2.1.

Крім того, деякі сервіси орієнтовані здебільшого на командну роботу, інші — потребують платної підписки для доступу до розширеного функціоналу, а частина з них перевантажена непотрібними модулями, які відволікають від основної мети.

Саме ці висновки стали основою для формування вимог до власного проєкту. Метою стало створення web-застосунку, який поєднує простоту використання, інтуїтивно зрозумілий інтерфейс і гнучкість функціоналу, а також відповідає практичним потребам користувачів у сфері організації особистого часу.

2.2. Основні вимоги до системи

На основі аналізу існуючих рішень та практичного досвіду використання різних сервісів було сформульовано основні вимоги до майбутнього web-застосунку. Мета була проста: створити інструмент, який дійсно полегшує планування часу, не перевантажений зайвими функціями, але й не обмежений базовим мінімумом. Усі вимоги можна поділити на функціональні та нефункціональні, наведено в таблиці 2.2.

Функціональні вимоги визначають, що саме користувач повинен мати змогу робити в системі:

- додавати, редагувати, видаляти завдання;
- встановлювати пріоритет для кожного завдання;
- групувати завдання за категоріями або тегами;
- переглядати свої завдання у вигляді календаря;
- відзначати виконані завдання;
- переглядати статистику виконання (наприклад, кількість виконаних/прострочених завдань);
- можливість створювати повторювані завдання;
- шукати завдання за ключовими словами;
- легкий та зрозумілий інтерфейс для взаємодії з системою.

Нефункціональні вимоги стосуються якості роботи системи та її експлуатації:

- швидка робота інтерфейсу;
- безпека зберігання даних користувача;
- можливість масштабування при зростанні кількості користувачів;
- адаптивність під різні пристрої (комп’ютер, планшет, смартфон);
- надійність (мінімізація ризику втрати даних);
- простота впровадження та супроводу.

Таблиця 2.2

Основні вимоги до web-застосунку для планування часу

Категорія	**Вимога**	**Опис / Пояснення**
Функціональна	Додавання завдань	Створення нової задачі із зазначенням дати та часу
Функціональна	Редагування/видалення завдань	Можливість змінити або видалити будь-яку задачу
Функціональна	Пріоритети та категорії	Вибір рівня важливості, групування за типом справ
Функціональна	Перегляд календаря	Відображення задач у вигляді календаря/списку
Функціональна	Повторювані завдання	Автоматичне створення типових задач (наприклад, щотижня)
Функціональна	Статистика виконання	Перегляд прогресу, кількості виконаних завдань
Нефункціональна	Адаптивний інтерфейс	Коректна робота на різних пристроях

Продовження таблиці 2.2

Основні вимоги до web-застосунку для планування часу

Категорія	**Вимога**	**Опис / Пояснення**
Нефункціональна	Безпека даних	Авторизація, захист персональної інформації
Нефункціональна	Надійність роботи	Автоматичне збереження та резервування даних
Нефункціональна	Легкість використання	Мінімум дій для створення і керування завданнями

Головна ідея полягала у тому, щоб користувач міг максимально швидко і просто організувати свій день або тиждень, не замислюючись над технічними деталями (див. рисунок 2.1). Особлива увага була приділена адаптивності інтерфейсу, що забезпечує зручність використання web-застосунку як на стаціонарних комп’ютерах, так і на мобільних пристроях. Важливим також стало питання безпеки — усі дані повинні зберігатися у захищеному вигляді, з можливістю відновлення у разі збоїв.

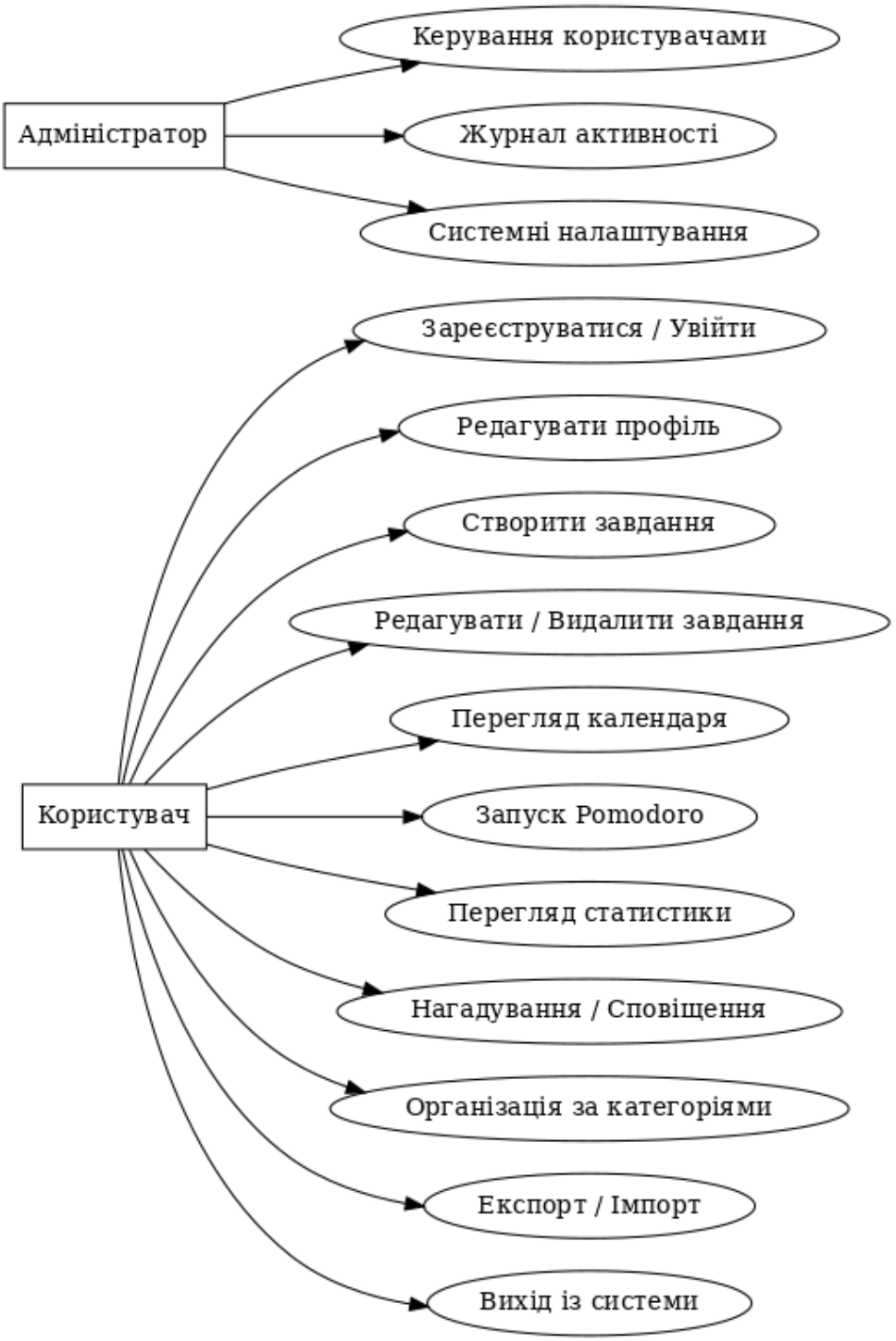


Рис. 2.1 Діаграма варіантів використання для ролей користувача та адміністратора в системі TimePlanner

2.3. Огляд цільової аудиторії та її потреб

В процесі аналізу предметної області важливо не тільки вивчити існуючі сервіси, а й зрозуміти, хто саме є їх основними користувачами та які реальні потреби вони мають. Адже від розуміння цільової аудиторії напряму залежить, які функції повинні бути втілені у майбутньому застосунку, як виглядатиме інтерфейс і навіть наскільки глибоко варто занурюватись у питання безпеки чи адаптивності.

2.3.1 Цільова аудиторія

На основі особистого досвіду, спілкування з друзями, а також огляду користувацьких відгуків у популярних додатках для планування часу, можна виділити декілька основних груп потенційних користувачів:

1. Студенти та учні: для цієї категорії особливо важливо організовувати навчальний процес, готуватися до іспитів, здавати роботи у встановлені строки, не забувати про додаткові курси чи гуртки. Важливими для них є простота використання та наочність відображення завдань.
2. Молоді фахівці та офісні працівники: для людей, які тільки починають кар'єру, питання організації часу нерідко стоїть дуже гостро: багато нових завдань, необхідність контролювати дедлайни, планувати зустрічі, брати участь у командних проектах. Вони цінують можливість групувати завдання, встановлювати пріоритети, отримувати нагадування.
3. Люди, що поєднують кілька видів діяльності: це можуть бути фрілансери, підприємці або ті, хто встигає і працювати, і навчатися, і займатися особистими справами. Їм потрібен максимально гнучкий інструмент, що дозволяє легко перемикатися між різними напрямками.
4. Особи, які прагнуть підвищити особисту продуктивність**:** часто такі люди вже мають досвід використання різних сервісів, проте шукають рішення, яке допоможе їм не лише планувати, а й аналізувати власну ефективність, формувати корисні звички, боротися з прокрастинацією.

2.3.2 Основні потреби та очікування користувачів

Кожна з цих груп має свої унікальні потреби, але є й спільні очікування від системи планування часу:

1. Швидкий і зручний інтерфейс: ніхто не хоче витрачати багато часу на вивчення складних інструментів або заповнення зайвих полів — додаток повинен працювати інтуїтивно.
2. Можливість персоналізації: багато користувачів хочуть налаштовувати систему «під себе»: створювати власні категорії, теми оформлення, нагадування.
3. Простота синхронізації між пристроями: важливо, щоб дані були доступні і з комп'ютера, і з телефона, без складних налаштувань чи втрат інформації.
4. Гнучка система нагадувань і пріоритетів:. для багатьох принципово важливо мати змогу відзначати найважливіші завдання, отримувати нагадування у потрібний момент, а не тоді, коли це вже неактуально.
5. Безпека особистої інформації: особливо для тих, хто планує не лише робочі, а й особисті чи навіть конфіденційні справи.

Саме ці спостереження стали основою для формування функціональних і нефункціональних вимог до web-застосунку, щоб він не був ще одним “загальним” рішенням, а реально відповідав живим запитам і викликам сучасного користувача.

3 ПРОЕКТУВАННЯ WEB-ЗАСТОСУНКУ

3.1 Загальна архітектура системи

Під час розробки TimePlanner основна увага приділялася створенню системи, яка залишалася б простою у використанні для користувача, водночас зберігаючи внутрішню гнучкість і масштабованість. Це означало, що архітектура повинна підтримувати різноманітні сценарії роботи, дозволяти легко додавати нові функції й забезпечувати надійне збереження всіх даних.

В основі TimePlanner лежить класична багаторівнева архітектура: клієнт–сервер, де клієнтська частина відповідає за інтерфейс та взаємодію з користувачем, а серверна — за бізнес-логіку та роботу з базою даних. Такий підхід дозволяє чітко розділити відповідальність між різними компонентами системи та спростити підтримку коду в майбутньому.

Серверна частина побудована з використанням C# [18] та ASP.NET [4],, що забезпечує високу продуктивність, стабільність і безпеку. Для взаємодії з базою даних застосовано Entity Framework, який суттєво полегшує роботу з моделями та міграціями даних.

У структурі TimePlanner можна виділити кілька ключових компонентів:

1. Інтерфейс користувача (web-клієнт): усі основні функції для планування, перегляду статистики, використання Pomodoro [8] та роботи з особистим кабінетом.
2. Модуль автентифікації та авторизації: відповідає за реєстрацію, вхід користувачів, а також розмежування доступу до окремих функцій (наприклад, адміністратор може керувати всіма користувачами).
3. Модуль роботи із завданнями: дає можливість створювати, редагувати, видаляти й фільтрувати задачі, визначати пріоритети, встановлювати дедлайни та позначати виконання.

4. Модуль Pomodoro[8]: реалізує функціонал таймера для роботи у техніці Pomodoro з можливістю зберігати історію сесій.
5. Модуль аналітики та статистики: відображає статистику виконаних задач, динаміку, прогрес, щоденні огляди та лідерборд користувачів.
6. База даних: зберігає всю інформацію про користувачів, задачі, сесії Pomodoro[8], огляди та ролі.

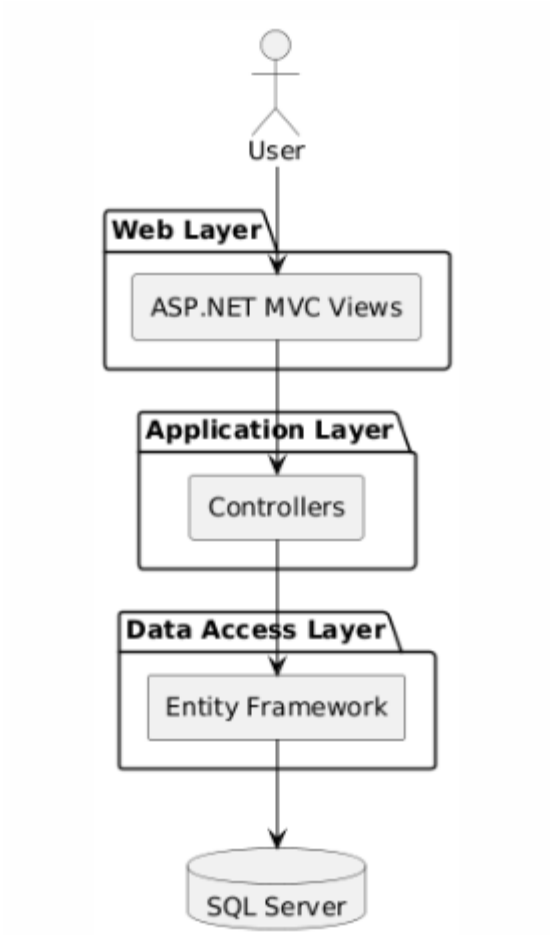


Рис.3.1 Структура TimePlanner

Вся робота системи побудована так, що користувачі взаємодіють із застосунком через зручний web-інтерфейс. Кожна дія (додавання задачі, запуск Pomodoro, аналіз статистики тощо) — це окремий запит до серверної частини, яка обробляє інформацію, зберігає або оновлює дані у базі, і повертає результат у вигляді оновленої сторінки чи сповіщення, зображено на рисунку 3.1.

Такий підхід дозволяє гнучко управляти всіма процесами, не перевантажувати інтерфейс зайвими деталями та гарантувати безпеку даних.

3.2 Функціональні можливості

Головною ідеєю при розробці TimePlanner було створення такого інструменту, який би закривав усі базові потреби користувача у плануванні часу та був інтуїтивно зрозумілим з першого використання. Ось які функціональні можливості були реалізовані у системі:

1. Управління завданнями:

- створення, редагування та видалення задач: користувач може додати нову задачу, вказавши її назву, опис, категорію, рівень пріоритету, дедлайн та теги. У будь-який момент задачу можна змінити або видалити, якщо вона стала неактуальною.
- фільтрація та сортування: зручно фільтрувати завдання за категорією, статусом, терміном виконання, а також шукати по тегах або ключовим словам. Це допомагає швидко знаходити потрібні справи навіть при великій кількості задач (див. скріншот “Мої завдання”).
- встановлення статусу та пріоритету: для кожної задачі можна обрати статус ("нове", "в роботі", "завершено") та пріоритет (низький, середній, високий). Це допомагає структурувати завантаження і зосереджуватися на найважливішому.
- 2. Робота з календарем:
- відображення задач у вигляді календаря: користувачі можуть переглядати свої завдання на календарі, що дозволяє планувати час на день, тиждень або місяць, та швидко оцінювати завантаженість.

3. Аналітика та статистика:

- статистика виконання: система відображає кількість усіх задач, скільки з них завершено, які завдання заплановані на сьогодні, тиждень чи місяць. Є

можливість переглядати графік виконаних задач за різні періоди (див. скріншот “Графік завершених задач”).

- щоденні огляди: окремий функціонал дозволяє додавати короткі нотатки про підсумки дня: що вдалося, які були труднощі, що хочеться поліпшити. Це допомагає аналізувати свою продуктивність та зростати над собою.
- лідерборд: для додаткової мотивації користувачі можуть бачити, хто виконав найбільше завдань за певний період. Лідерборд підкреслює ігровий елемент і дозволяє “конкурувати” за досягненнями.

4. Pomodoro таймер

- Реалізація методики Pomodoro:

Користувач може запускати таймер на 25 хвилин (або інший інтервал), працювати в фокусі, після чого система пропонує зробити перерву.

Вся історія Pomodoro-сесій зберігається, що дозволяє аналізувати свою концентрацію та ефективність протягом дня чи тижня.

5. Реєстрація, автентифікація та адміністрування

- Система облікових записів:

Кожен користувач має свій профіль, усі дані зберігаються безпечно та недоступні іншим.

- Адміністративна панель:

Передбачено інтерфейс для адміністрування — адміністратор може переглядати користувачів, видаляти їх, керувати доступом. Це дозволяє контролювати систему та реагувати на порушення, якщо такі виникають.

6. Гнучкість та простота

- Адаптивний дизайн:

Інтерфейс зручно виглядає на різних пристроях, що дозволяє працювати з планувальником і на комп’ютері, і на смартфоні.

- Інтуїтивний інтерфейс:

Усі основні функції завжди “під рукою” — створення задач, перемикання між вкладками (Головна, Завдання, Календар, Статистика, Pomodoro), фільтрація, статистика, профіль.

3.3. Моделювання даних (ER-діаграма, структура бази даних)

Нижче наведена ER-діаграма рисунку 3.2, створена у SQL Management Studio, що ілюструє основні зв'язки між таблицями у базі даних TimePlanner:

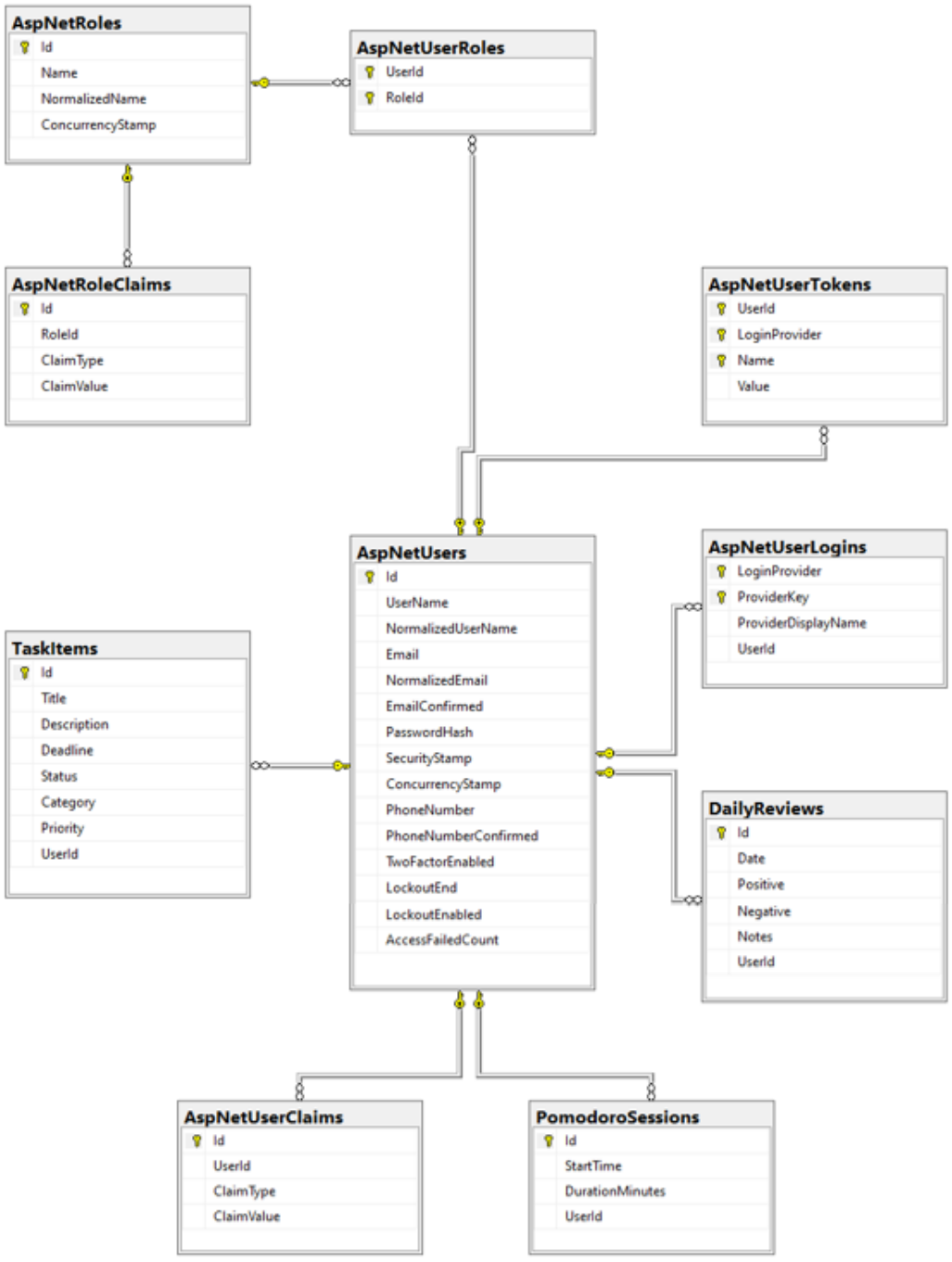


Рис.3.2 ER-діаграма зв’язків

Взаємозв'язки між таблицями

- Кожен користувач може мати багато задач, оглядів та сесій Pomodoro[8];
- Ролі, claims та logins — дозволяють налаштувати гнучку систему доступу й автентифікації;
- Зв'язки реалізовані через зовнішні ключі (UserId), що забезпечує цілісність даних;

Опис основних таблиць бази даних

1. AspNetUsers

Це основна таблиця користувачів. У ній зберігається вся ключова інформація про кожного, хто зареєструвався у системі:

- UserName, Email — основні ідентифікатори користувача;
- PasswordHash — захищене зберігання пароля;
- PhoneNumber, EmailConfirmed — додаткові контакти та статус підтвердження;
- TwoFactorEnabled — можливість двофакторної автентифікації;
- LockoutEnabled, AccessFailedCount — дані для безпеки та захисту облікового запису;

Кожен користувач може мати кілька ролей, задач, сесій Pomodoro, оглядів і claims.

2. AspNetRoles, AspNetUserRoles, AspNetRoleClaims

Ці таблиці відповідають за управління ролями користувачів і їхніми правами. Дозволяють:

- Призначати ролі (“користувач”, “адміністратор” тощо);
- Вказувати окремі права доступу через claims.

3. TaskItems

Одна з ключових таблиць. Тут зберігається вся інформація про завдання користувача:

- Id — унікальний ідентифікатор задачі;

- Title, Description — коротка назва та детальний опис задачі;
- Deadline — кінцевий термін виконання;
- Status — стан задачі (“нове”, “в роботі”, “виконано”);
- Category — категорія (наприклад, “робота”, “навчання”, “особисте”);
- Priority — рівень пріоритету (1 — низький, 2 — середній, 3 — високий);
- UserId — посилання на користувача, якому належить ця задача.

Всі задачі прив’язані до конкретного користувача. Це дозволяє формувати індивідуальні списки завдань для кожного.

4. PomodoroSessions

Ця таблиця відповідає за зберігання інформації про сесії Pomodoro:

- Id — унікальний ідентифікатор сесії;
- StartTime — час початку сесії;
- DurationMinutes — тривалість в хвилинах;
- UserId — посилання на користувача.

5. DailyReviews

Тут фіксуються щоденні підсумки користувача:

- Id — ідентифікатор запису;
- Date — дата створення огляду;
- Positive, Negative — короткі нотатки, що вдалося та що викликало труднощі;
- Notes — додаткові коментарі;
- UserId — посилання на користувача.

Така таблиця допомагає формувати звички саморефлексії та аналізувати свій прогрес.

6. Додаткові таблиці безпеки

- AspNetUserClaims, AspNetUserLogins, AspNetUserTokens — використовуються стандартною системою ASP.NET Identity для керування claims, логінами через зовнішні сервіси, токенами тощо.

Структура бази даних побудована таким чином, щоб гнучко та надійно зберігати всю інформацію, потрібну для роботи з завданнями, аналізу статистики й забезпечення високого рівня безпеки. Таке моделювання дозволяє легко розширювати функціонал у майбутньому та інтегрувати нові модулі.

3.4. Опис основних сценаріїв використання

Проєктування TimePlanner здійснювалося з орієнтацією на практичну користь для кінцевого користувача, з акцентом на вирішення реальних щоденних потреб, а не лише реалізацію функціональних можливостей. Ось декілька основних сценаріїв використання, які найбільш типові для цільової аудиторії.

Сценарій 1: Щоденне планування справ

Приклад ситуації:

Студент розпочинає новий день і хоче чітко розпланувати свої завдання: підготовка до занять, виконання домашніх завдань, підготовка до іспиту, а також особисті справи.

Як це відбувається в системі (зображено на рисунку 3.4):

1. Користувач відкриває сторінку “Мої завдання” та натискає “Створити”.
2. Вказує назву, опис, категорію (“Навчання”, “Робота”, “Особисте”) встановлює дедлайн та обирає пріоритет.
3. За потреби додає теги, наприклад, #реферат, #читання.
4. Зберігає завдання — і бачить їх у загальному списку або календарі.
5. Протягом дня відмічає виконані задачі, відслідковує прогрес у статистиці.

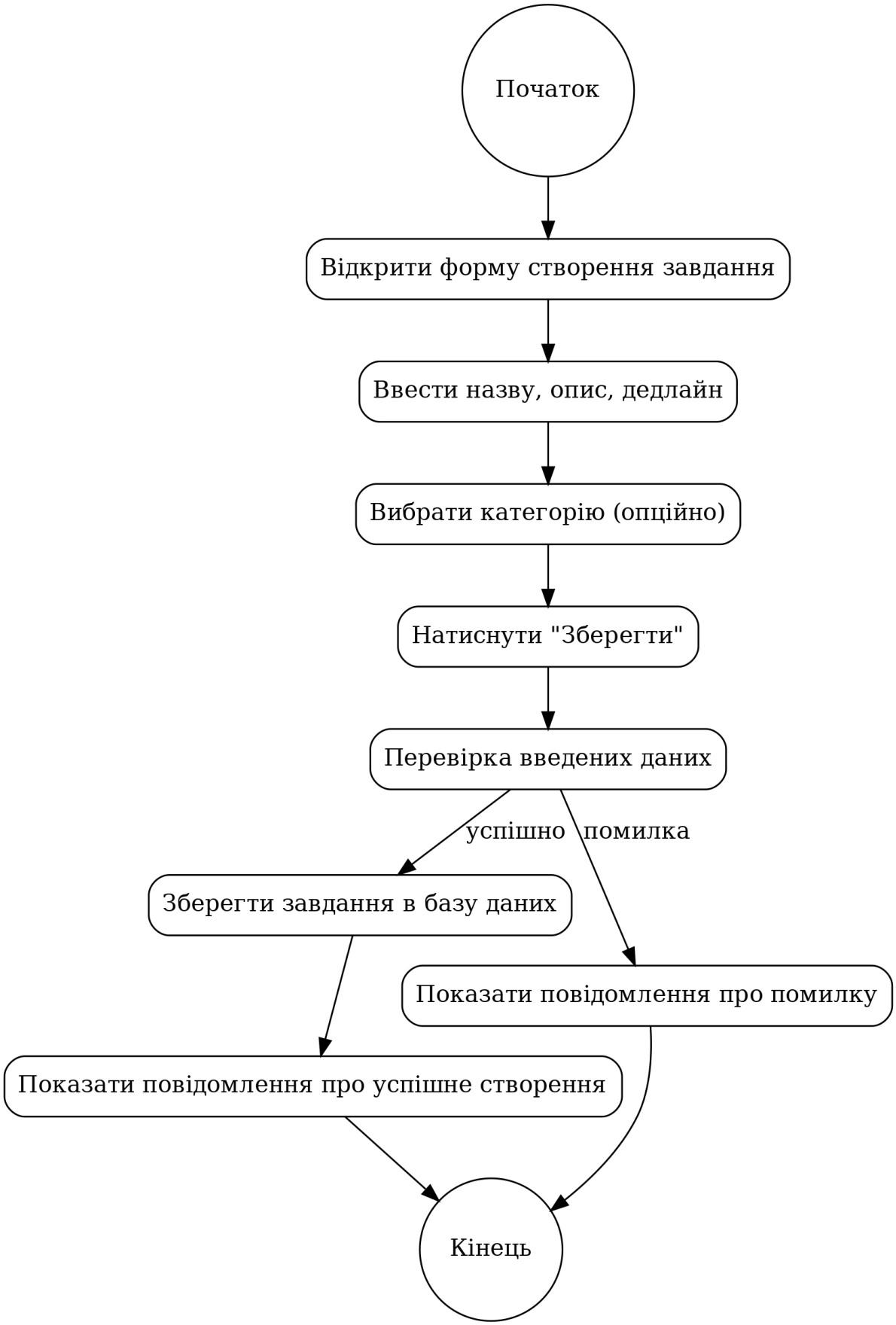


Рис. 3.4 Діаграма діяльності для процесу додавання завдання в TimePlanner

Сценарій 2: Фокусована робота за методикою Pomodoro

Приклад ситуації:

Користувач хоче сконцентруватися на складному завданні без відволікань.

Як це реалізовано:

1. Вибирає завдання, над яким працюватиме, відкриває вкладку “Pomodoro”.
2. Запускає таймер на 25 хвилин, працює над задачею, не відволікаючись.
3. Після сигналу робить 5-хвилинну перерву, відзначає сесію як виконану.
4. Історія всіх сесій Pomodoro зберігається — користувач бачить статистику своєї концентрації і може аналізувати, коли працював найефективніше.

Сценарій 3: Аналіз виконаних справ та саморефлексія

Приклад ситуації:

Після напруженого тижня користувач хоче підбити підсумки: скільки задач було виконано, що вдалося найкраще, які завдання залишилися невиконаними.

Як це працює в TimePlanner:

1. Користувач переходить у розділ “Статистика” і переглядає графік виконаних задач за день, тиждень чи місяць.
2. Дивиться, які категорії були найпродуктивнішими, скільки задач виконано вчасно.
3. Вносить щоденний/тижневий огляд — коротко записує, що вийшло добре (поле “Positive”), що варто покращити (“Negative”) та залишає власні нотатки (“Notes”).

Завдяки цій рефлексії користувач формує здорові звички, вчиться краще розподіляти час у майбутньому.

Сценарій 4: Мотивація через лідерборд

Приклад ситуації:

Група друзів або колег вирішила змагатися, хто більше задач виконає за місяць.

Як TimePlanner допомагає:

1. Усі користувачі бачать свій рейтинг у розділі “Лідерборд”.
2. Той, хто закриває найбільше задач, піднімається вгору у списку.
3. Такий підхід створює додатковий ігровий інтерес, стимулює не відкладати справи на потім і досягати цілей разом із друзями чи командою.

Сценарій 5: Адміністрування користувачів (для адміністратора)

Приклад ситуації:

Адміністратор системи хоче проконтролювати активність користувачів або видалити акаунт порушника.

Як це реалізовано:

1. Адміністратор входить у “Панель адміністратора”.
2. Може переглядати список усіх користувачів, кількість виконаних задач кожного, дату останньої активності.
3. За потреби може видалити користувача або переглянути його завдання.

Кожен з цих сценаріїв — не вигадка, а результат реального досвіду користування TimePlanner. Всі вони показують, як система допомагає не лише вести список справ, а й аналізувати свою продуктивність, мотивувати себе і залишатися організованим у сучасному ритмі життя.

4 РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ

4.1. Вибір інструментальних засобів

Під час розробки TimePlanner основним завданням було досягнення оптимального балансу між надійністю, простотою підтримки та актуальністю використаних технологій. Вибір стеку був зумовлений не лише вимогами до функціоналу, а й особистим досвідом використання окремих мов та фреймворків [4].

Основні інструменти та технології наведено в таблиці 4.1

Мова програмування:

Для реалізації серверної частини було обрано мову програмування C#[18], яка відзначається сучасністю, високою продуктивністю та безпекою, що робить її придатною для розробки web-застосунків корпоративного рівня.

1. Фреймворк:

Основою став ASP.NET [4] Core [1] — потужний фреймворк від Microsoft, який забезпечує високу продуктивність, масштабованість та безпеку. Він чудово інтегрується з різними бібліотеками, дозволяє реалізувати MVC [16]-архітектуру і має велику спільноту підтримки.

2. Робота з базою даних:

Для ORM використано Entity Framework Core [2] — сучасний засіб для зручної роботи з реляційними БД, який дозволяє легко створювати і мігрувати схеми, працювати з LINQ та гарантує узгодженість даних.

3. База даних:

Для зберігання всієї інформації було обрано SQL Server. Це перевірена СУБД, яка забезпечує високу надійність, масштабованість та швидкий доступ до даних.

4. Інтерфейс користувача:

Для фронтенду використовувалася Razor Pages (або Views у рамках MVC [16]), стандартний підхід у ASP.NET [4],, що дозволяє швидко створювати динамічні сторінки.

5. Стилізація:

Для оформлення зовнішнього вигляду застосунку застосовувалася Bootstrap [12] — популярний CSS-фреймворк для адаптивного і привабливого інтерфейсу.

6. Аутентифікація та авторизація:

Стандартна система ASP.NET Identity — дозволяє швидко налаштувати реєстрацію, логін, керування ролями, захист персональних даних користувача.

Таблиця 4.1

Вибір основних інструментів та технологій для TimePlanner

Категорія	**Інструмент / Технологія**	**Опис, причина вибору**
Мова програмування	C#	Надійність, сучасність, підтримка Microsoft
Фреймворк	ASP.NET Core	Масштабованість, швидкість, безпека
ORM	Entity Framework Core	Зручність, робота з LINQ, швидке мігрування
База даних	SQL Server	Стабільність, висока продуктивність

Продовження таблиці 4.1

Вибір основних інструментів та технологій для TimePlanner

Категорія	**Інструмент / Технологія**	**Опис, причина вибору**
Фронтенд	Razor Pages / MVC Views	Швидка розробка динамічних сторінок
Стилізація	Bootstrap	Адаптивний, сучасний дизайн
Аутентифікація	ASP.NET Identity	Вбудована безпека, керування ролями

Обраний стек дозволяє забезпечити швидку і надійну роботу застосунку, легко впроваджувати нові функції, масштабувати систему та підтримувати її у довгостроковій перспективі. Крім того, використання знайомих і перевірених технологій знижує поріг входу для інших розробників, які можуть долучатися до проєкту в майбутньому.

4.2. Опис структури проекту

Структура проєкту TimePlanner побудована за принципами, які забезпечують логічну організацію коду, зручність підтримки та розширення, а також чітке розділення відповідальності між різними частинами системи. Проєкт створювався відповідно до шаблону ASP.NET Core MVC, що допомагає дотримуватися стандартних підходів у розробці сучасних web-застосунків.

Основні каталоги та їх призначення

1. Controllers рисунок 4.1

Містить контролери — основні компоненти, які обробляють запити користувача, керують логікою переходів між сторінками, взаємодіють з моделями та повертають відповіді у вигляді Views або JSON-даних (для AJAX).

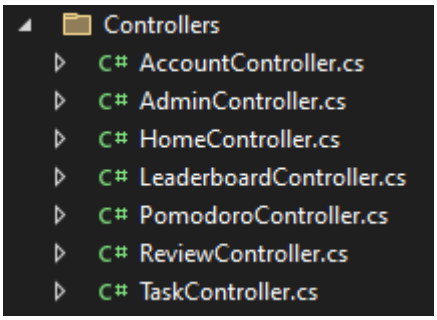


Рис. 4.1 Controllers

2. Models рисунок 4.2

Тут знаходяться всі моделі даних — класи, що відображають структуру таблиць у базі даних (наприклад, TaskItem, PomodoroSession, DailyReview, User тощо). Також можуть міститися моделі для валідації та допоміжних задач.

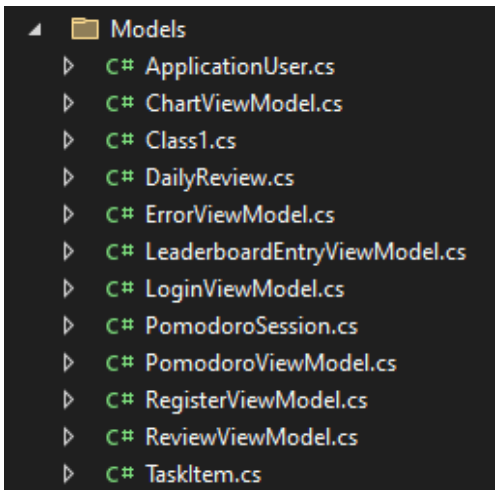


Рис. 4.2 Models

3. Views рисунок 4.3

   Каталог, де розміщені Razor-сторінки (файли .cshtml), що відповідають за відображення інтерфейсу користувача. Для кожного контролера зазвичай існує своя папка з відповідними представленнями: наприклад, Tasks, Pomodoro, Statistics, Account.

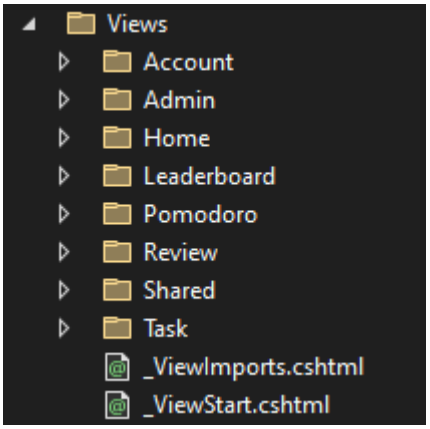


Рис. 4.3 Views

4. Data рисунок 4.4

   Тут міститься контекст бази даних (ApplicationDbContext) і класи для початкового заповнення, міграцій, конфігурації Entity Framework.

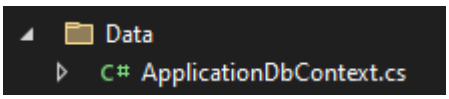


Рис. 4.4 Data

5. Migrations рисунок 4.5

   Містить файли міграцій, що дозволяють оновлювати структуру бази даних без втрати даних.

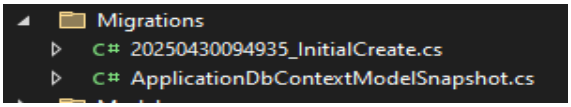


Рис. 4.5 Migrations

6. Properties, appsettings.json, Program.cs, Startup.cs рисунок 4.6

Файли конфігурації застосунку, точки входу в програму, налаштування залежностей, middleware, підключення до бази даних тощо.

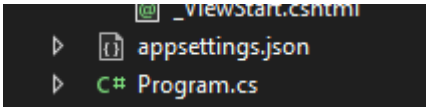


Рис. 4.6 Properties, appsettings.json, Program.cs, Startup.cs

4.3. Реалізація основних модулів

TasksController – керування завданнями

Це, мабуть, центральний контроллер усього застосунку, який реалізує логіку для роботи із завданнями користувача.

Ось основні можливості, які забезпечує TasksController:

1. Відображення списку завдань з фільтрацією по статусу (наприклад, “у роботі”, “завершені”), категорії чи дедлайну (метод Index).

   Це дозволяє користувачу швидко знаходити потрібні задачі, зосереджуватись на пріоритетних справах і планувати свій день максимально ефективно.
2. Календар завдань (метод Calendar) — вивід задач із дедлайнами у вигляді календаря, щоб бачити розподіл навантаження у часі.
3. Створення, редагування, видалення завдань (Create, Edit, Delete) — вся логіка перевірки прав користувача, валідації даних, збереження нових задач у базі, їх оновлення та видалення.

   Наприклад, при створенні задачі система одразу прив'язує її до поточного користувача через UserId — це виключає випадки втрати чи "плутанини" між обліковками.
4. Позначення задачі як виконаної (Complete) — користувач може одним кліком відмітити завершення завдання.

   Ця дія також впливає на статистику та лідерборд.

5. Перегляд деталей завдання (Details) — можна подивитися всю інформацію про задачу, навіть якщо вона вже виконана чи перенесена.

```
public async Task<IActionResult> Index(string status, string category, string due)
{
    var userId = _userManager.GetUserId(User);
    var tasksQuery = _context.TaskItems
        .Where(t => t.UserId == userId)
        .AsQueryable();

    if (!string.IsNullOrEmpty(status))
    {
        if (Enum.TryParse<WebApplication3.Models.TaskStatus>(status, out var parsedStatus))
        {
            tasksQuery = tasksQuery.Where(t => t.Status == parsedStatus);
        }
    }

    if (!string.IsNullOrEmpty(category))
    {
        tasksQuery = tasksQuery.Where(t => t.Category != null && t.Category.ToLower() == category.ToLower());
    }

    if (!string.IsNullOrEmpty(due))
    {
        var today = DateTime.Today;
        tasksQuery = due switch
        {
            "today" => tasksQuery.Where(t => t.Deadline.HasValue && t.Deadline.Value.Date == today),
            "upcoming" => tasksQuery.Where(t => t.Deadline.HasValue && t.Deadline.Value.Date > today),
            "overdue" => tasksQuery.Where(t => t.Deadline.HasValue && t.Deadline.Value.Date < today),
            _ => tasksQuery
        };
    }

    var tasks = await tasksQuery.ToListAsync();
    return View(tasks);
}
```

Рис 4.6 TasksController

Код контроллера охоплює всі базові операції CRUD (Create, Read, Update, Delete) і забезпечує зручну роботу з даними завдяки використанню Entity Framework та системи ідентифікації користувача через UserManager див.рис.4.6.

2. LeaderboardController – мотивація та “змагальний” елемент

Особливість TimePlanner — це не просто “туду-лист”, а й інструмент для підвищення мотивац ії користувачів через гейміфікацію (див. рисунок 4.6).

```
public class LeaderboardController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly UserManager<ApplicationUser> _userManager;

    Ссылок: 0
    public LeaderboardController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)
    {
        _context = context;
        _userManager = userManager;
    }

    Ссылок: 0
    public async Task<IActionResult> Index()
    {
        var completedTasks = await _context.TaskItems
            .Where(t => t.Status == WebApplication3.Models.TaskStatus.Done)
            .GroupBy(t => t.UserId)
            .Select(g => new
            {
                UserId = g.Key,
                Count = g.Count(),
                LastCompleted = g.Max(t => t.Deadline)
            })
            .OrderByDescending(g => g.Count)
            .ToListAsync();

        var users = await _context.Users.ToListAsync();

        var leaderboard = completedTasks.Select(x => new LeaderboardEntryViewModel
        {
            UserName = users.FirstOrDefault(u => u.Id == x.UserId)?.Email ?? "Unknown",
            CompletedTasks = x.Count,
            LastCompletedDate = x.LastCompleted
        }).ToList();

        return View(leaderboard);
    }
}
```

Рис. 4.7 LeaderboardController

1. Головна задача LeaderboardController — формувати таблицю лідерів на основі кількості виконаних завдань кожного користувача (метод Index).
2. Логіка проста, але дієва: система підраховує всі завершені задачі для кожного користувача, визначає дату останньої виконаної задачі, формує модель для виводу і потім відображає лідерборд на сторінці.

Такий підхід дозволяє користувачам бачити свої досягнення у порівнянні з іншими, що часто стає додатковим стимулом не “відкладати справи на потім”.

3. AdminController – адміністративний контроль та безпека.

Для забезпечення порядку в системі й контролю за її роботою передбачений окремий модуль для адміністраторів (див. рисунок 4.7).

1. Список усіх користувачів (Index) — адміністратор може бачити всіх зареєстрованих користувачів, стежити за їх активністю.

2. Видалення користувача (Delete) — можливість видаляти обліковки (крім своєї), якщо це необхідно для підтримки безпеки або при порушенні правил.
3. Перегляд завдань конкретного користувача (Details) — адміністратор може подивитися, які задачі виконує кожен користувач.

   Ця функція корисна для розслідування спірних ситуацій або при обробці скарг.

Всі методи захищені атрибутом [Authorize(Roles = "Admin")], що виключає несанкціонований доступ.

```
private readonly UserManager<ApplicationUser> _userManager;
private readonly ApplicationDbContext _context;

Ссылок: 0
public AdminController(UserManager<ApplicationUser> userManager, ApplicationDbContext context)
{
    _userManager = userManager;
    _context = context;
}

Ссылок: 0
public IActionResult Index()
{
    var users = _userManager.Users.ToList();
    return View(users);
}

[HttpPost]
Ссылок: 0
public async Task<IActionResult> Delete(string id)
{
    var user = await _userManager.FindByIdAsync(id);
    if (user != null && user.Email != User.Identity?.Name)
    {
        await _userManager.DeleteAsync(user);
    }
    return RedirectToAction("Index");
}


Ссылок: 0
public async Task<IActionResult> Details(string id)
{
    if (id == null) return NotFound();

    var user = await _userManager.FindByIdAsync(id);
    if (user == null) return NotFound();

    var tasks = await _context.TaskItems
        .Where(t => t.UserId == id)
        .ToListAsync();

    ViewBag.UserEmail = user.Email;
    return View(tasks);
}
```

Рис. 4.8 AdminController

Коротко про інші модулі

3. У TimePlanner також є контролери для Pomodoro-сесій, статистики, профілю користувача, але центральними для роботи із завданнями та управління є саме описані вище.

4.4. Інтерфейс користувача

Інтерфейс користувача TimePlanner створений з урахуванням принципу “нічого зайвого”: усе найголовніше — під рукою, а кожен елемент допомагає швидко зорієнтуватися навіть новому користувачу, як на рисунку 4.9.

Ось короткий огляд основних розділів і сторінок із реальними скріншотами.

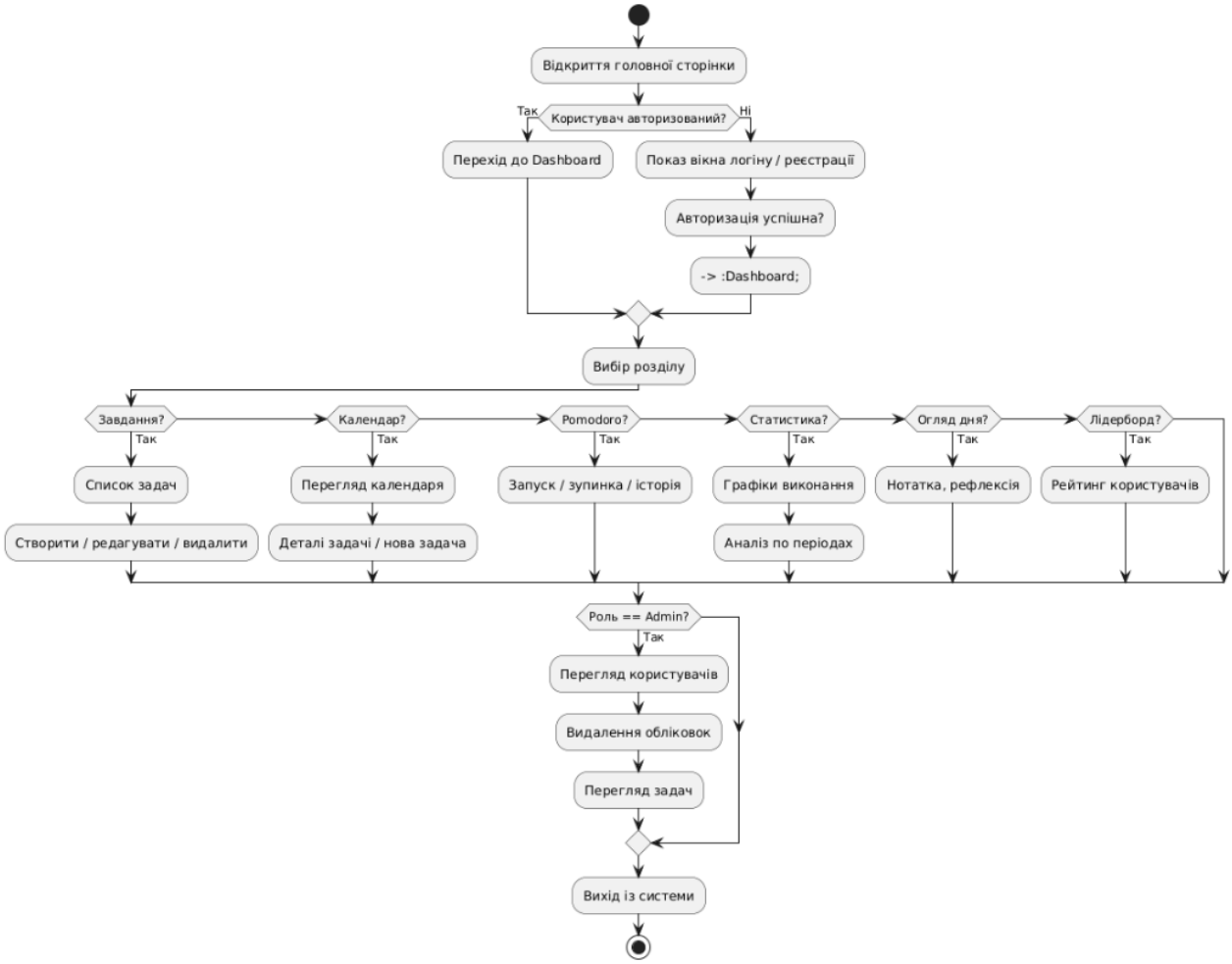


Рис. 4.9 Алгоритм роботи програми

Головна сторінка (Dashboard)

На головній сторінці користувач одразу бачить коротку статистику по задачах, мотиваційний лідерборд та актуальні події дня (див. рисунок 4.10).

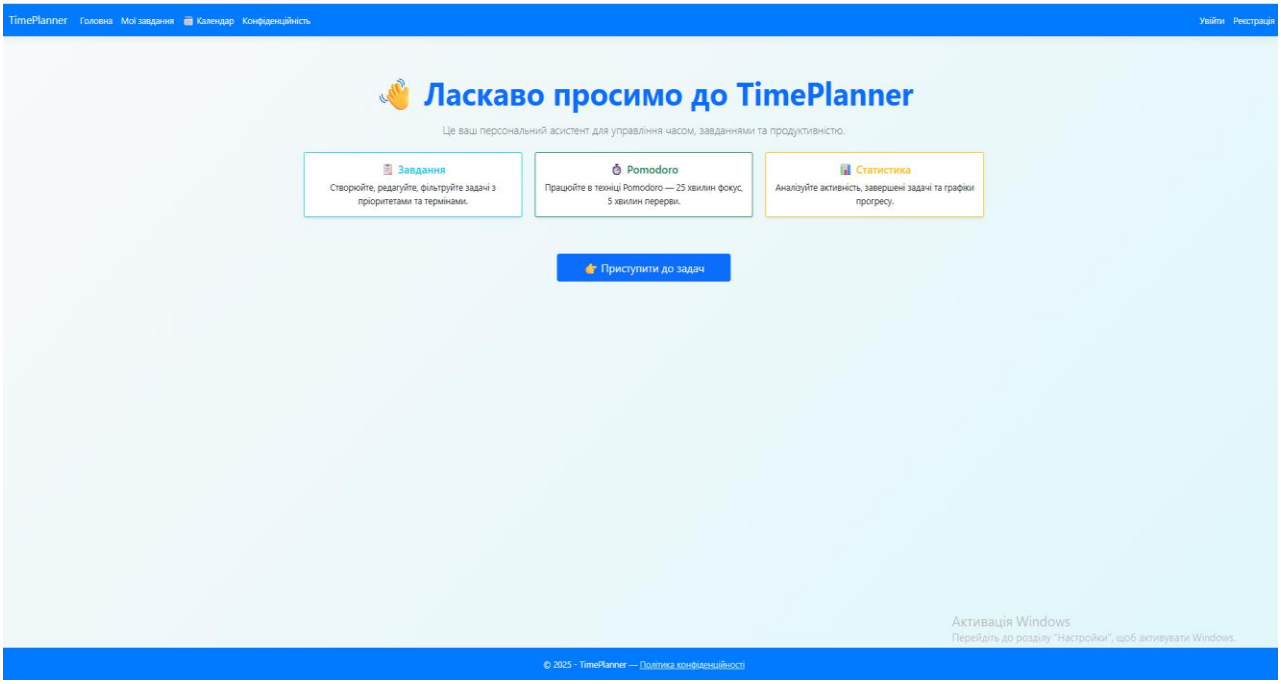


Рис. 4.10 Dashboard

Мої завдання (Task List)

Центральний елемент системи — список завдань. Тут можна створити нову задачу, відфільтрувати поточні по категорії, статусу або тегу, редагувати чи видаляти існуючі (див. рисунок 4.11).

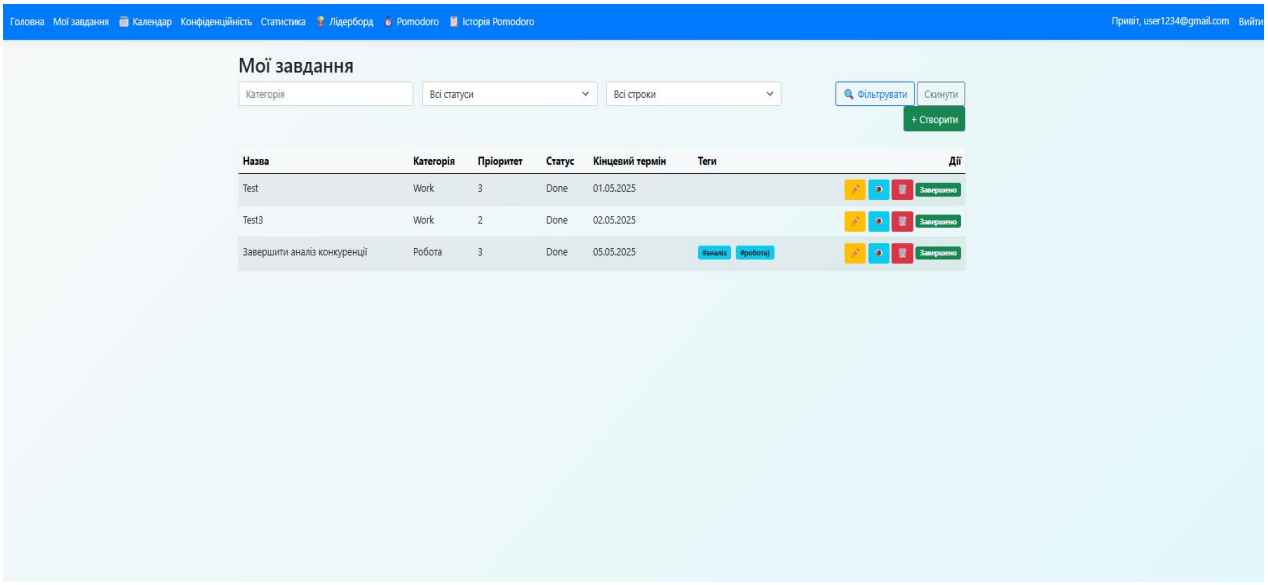


Рис. 4.11 Task List

Список задач: інтуїтивна фільтрація, кольорові позначки пріоритету, швидкі дії над кожною задачею.

Створення/редагування задачі

Користувач може швидко додати нову задачу, вказавши всі необхідні поля — від назви до дедлайну та тегів. Усе оформлено просто та зрозуміло, щоб не витрачати час на зайві формальності (див. рисунки 4.12-4.13).

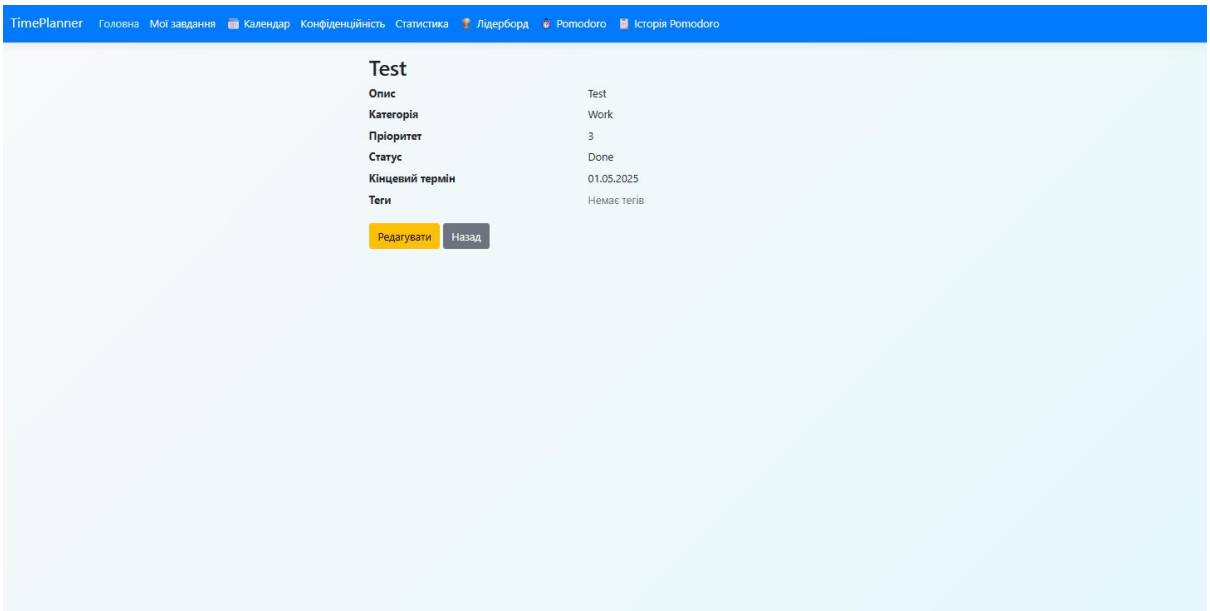


Рис. 4.12 Вікно редагування задачі

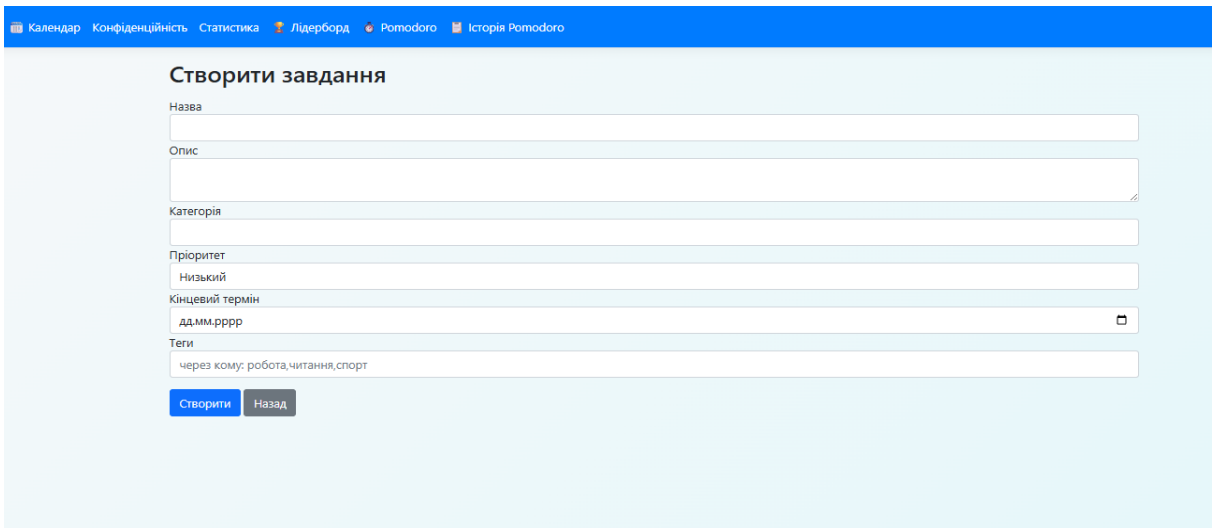


Рис. 4.13 Вікно редагування задачі

Календар завдань

На рисунку 4.14 надано календар, що дає можливість побачити розподіл задач у часі: що заплановано на сьогодні, завтра, наступний тиждень. Усі задачі з дедлайнами автоматично потрапляють у календар.

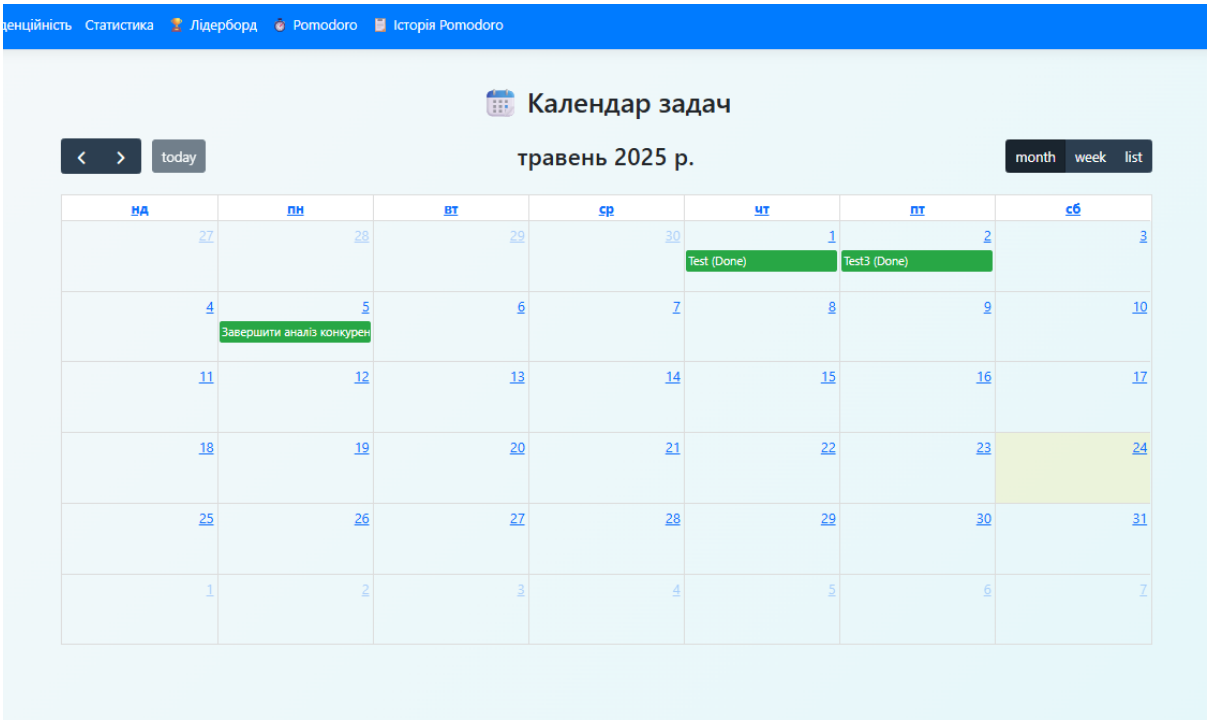


Рис. 4.14 Календар задач: наочне планування, повна картина завантаження на будь-який день.

Статистика і аналітика

Окремий розділ для тих, хто хоче бачити свій прогрес — тут можна переглядати графіки виконаних задач, активність за останній тиждень/місяць, порівнювати результати, зображено на рисунку 4.15.

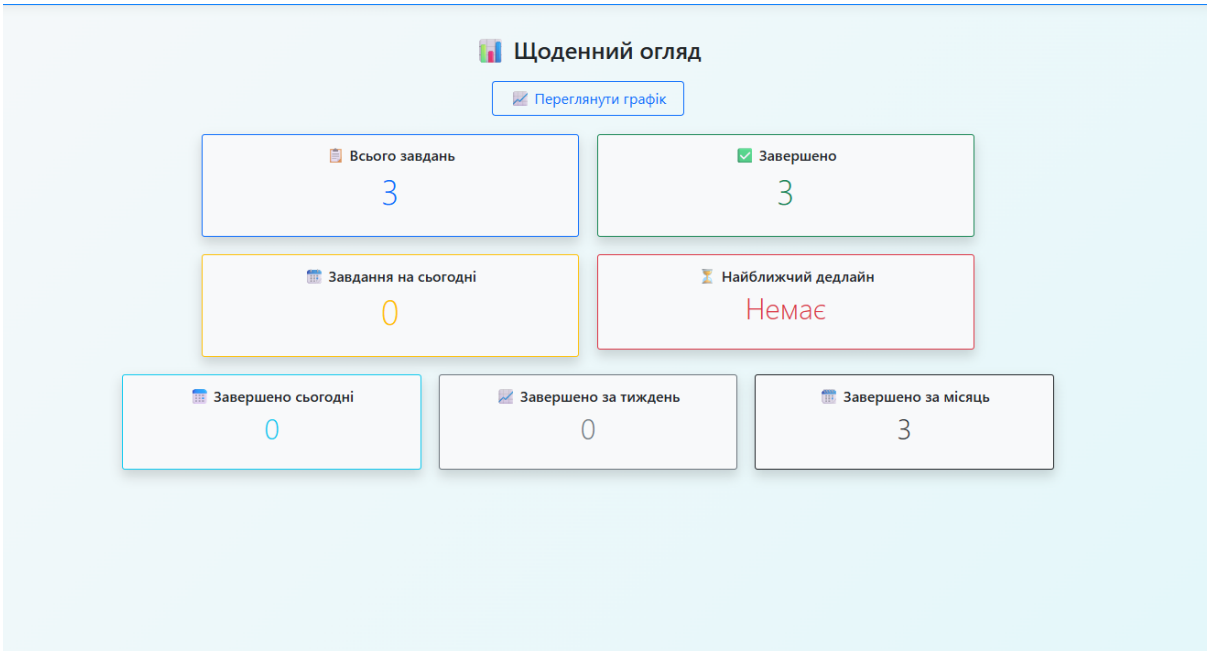


Рис. 4.15 Розділ статистики: динаміка виконання задач, графіки та візуалізація особистої продуктивності.

Лідерборд

Мотиваційна “дошка пошани” — тут користувач бачить себе у порівнянні з іншими: хто скільки задач виконав, хто лідирує за період (див. рисунок 4.16).

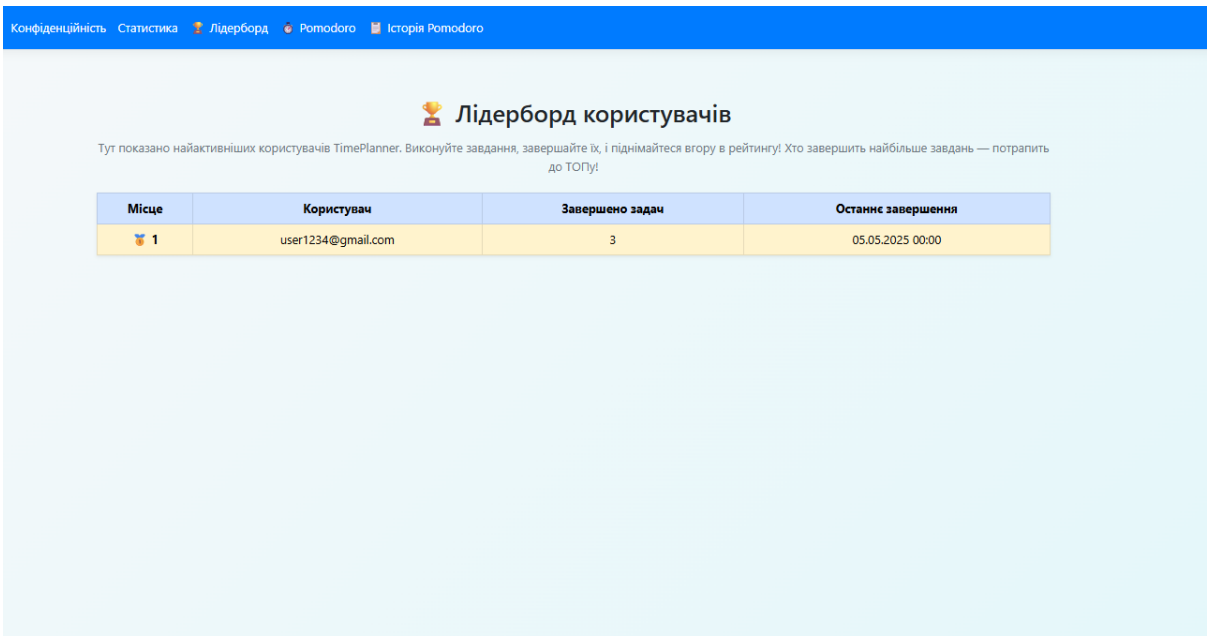


Рис. 4.16 Лідерборд: ігровий підхід до продуктивності, мотивація досягати більшого разом.

Pomodoro-таймер

Фокус для продуктивної роботи: тут можна запустити Pomodoro-сесію, побачити історію виконаних циклів і не забути зробити паузу (див. рисунок 4.17).

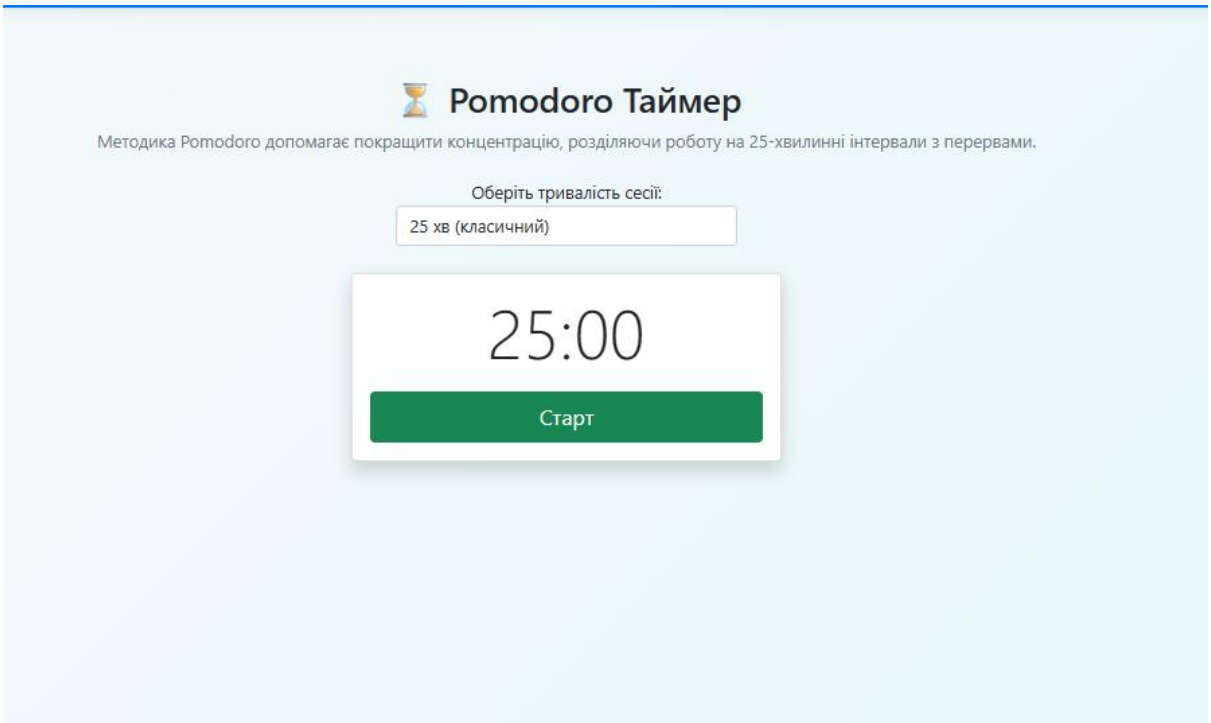


Рис. 4.17 Pomodoro-таймер: керування фокусом, облік ефективності і перерв

Адміністративна панель

На рисунку 4.18 доступна окрема панель для адміністратора з усіма користувачами, переглядом їх задач та можливістю керування акаунтами.

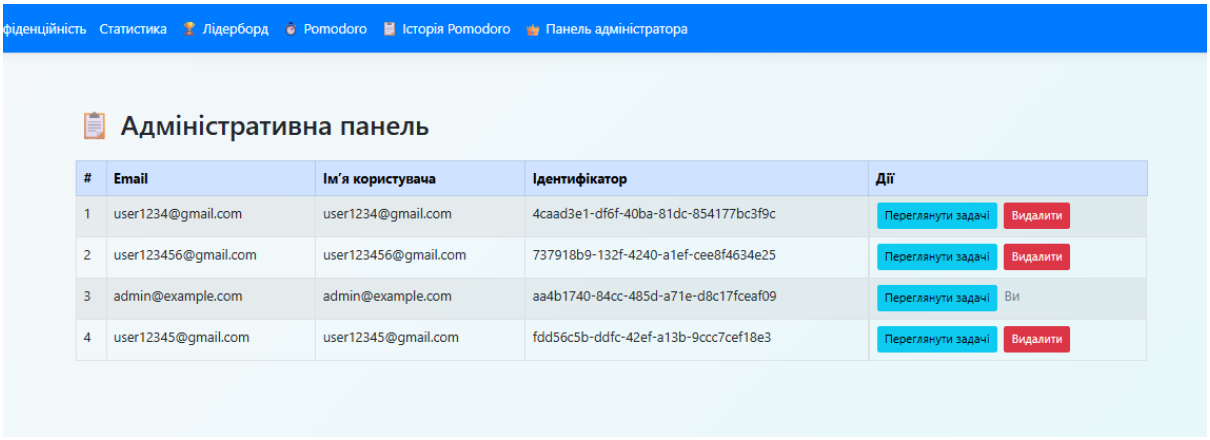


Рис. 4.18 Адмін-панель: повний контроль над системою, безпека та підтримка порядку

5 ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ

5.1 Методика тестування

Під час розробки TimePlanner основна увага приділялась практичному тестуванню функціоналу “на місці” — тобто у реальних сценаріях роботи із застосунком. Основна мета тестування — впевнитися, що всі ключові модулі працюють коректно, дані не губляться, а інтерфейс залишається зручним незалежно від обставин.

Для перевірки функціональності застосовувався підхід ручного тестування, під час якого кожна нова функція тестувалася в браузері — додавання завдань, зміна статусів, запуск Pomodoro, аналіз статистики, видалення облікових записів. Особливу увагу приділяли тестуванню механізмів доступу, щоб забезпечити захист даних користувачів та обмежити адміністративні можливості відповідно до ролей.

5.2. Приклади тестових сценаріїв

Для впевненості у працездатності TimePlanner було проведено перевірку основних користувацьких сценаріїв, зокрема:

1. Створення, редагування та видалення завдань:

Перевіряв, чи зберігається нова задача, чи можна її змінити або видалити, чи оновлюється список без помилок.

2. Фільтрація і сортування задач:

Тестував різні комбінації фільтрів (за категорією, статусом, дедлайном), щоб упевнитися, що користувач завжди бачить саме те, що йому потрібно.

3. Робота з Pomodoro-таймером:

Перевіряв запуск і зупинку таймера, збереження сесій у базі, правильність відображення історії.

4. Статистика та лідерборд:

Дивився, чи правильно підраховується кількість виконаних задач, чи коректно формується рейтинг користувачів.

5. Реєстрація, вхід, адміністративний доступ:

Тестував створення нових акаунтів, логін/лог-аут, доступність адмін-розділів тільки для адміністратора, видалення та перегляд обліковок.

6. Обмеження доступу:

Окремо перевіряв, що один користувач не може побачити чи змінити задачі іншого.

5.3.Аналіз результатів тестування

У процесі тестування більшість основних функцій працювали стабільно та відповідно до очікувань.

Під час ручної перевірки були виявлені окремі дрібні недоліки — наприклад, не завжди коректно оновлювався статус задачі при втраті інтернету, або іноді не зберігались окремі теги при створенні задачі. Ці помилки були оперативно виправлені під час розробки.

Загалом додаток показав себе як достатньо стійкий навіть при одночасній роботі кількох користувачів, а всі основні сценарії виконувались без серйозних збоїв.

5.4. Оцінка якості роботи застосунку

З урахуванням результатів тестування можна зробити висновок, що TimePlanner вже на першому етапі готовий для повсякденного використання. Головні переваги:

1. зручний та інтуїтивний інтерфейс,
2. стабільна робота основних модулів,
3. захищеність даних і чіткий розподіл прав доступу.

У майбутньому планується впровадження більш глибокого автоматизованого тестування для ще більшої впевненості у якості системи та підвищення її надійності при розширенні функціоналу.

ВИСНОВКИ

Досягнуті результати

Робота над TimePlanner вийшла за межі навчального проєкту і стала важливим етапом здобуття практичного досвіду. Було підтверджено, що навіть індивідуальний розробник може створити сучасний і функціональний інструмент для організації особистого часу за умови наявності мотивації та чіткого розуміння цілей. У ході розробки вдалося реалізувати усі ключові функції, які були заплановані на початку. Застосунок дозволяє:

1. створювати, редагувати та видаляти задачі, групувати їх за категоріями, встановлювати пріоритети та дедлайни;
2. користуватися Pomodoro-таймером для підвищення концентрації та фіксувати історію робочих сесій;
3. аналізувати свою продуктивність через детальну статистику, вести щоденні огляди і бачити свої сильні та слабкі сторони;
4. мотивувати себе та інших завдяки лідерборду і легкому “змагальному” елементу;
5. забезпечити високий рівень безпеки та конфіденційності даних, чітко розмежувавши права доступу між звичайними користувачами та адміністраторами.

Слід окремо відзначити, що TimePlanner — це не просто набір web-сторінока, а повноцінний сервіс, який можна використовувати як у навчанні, так і в роботі, для особистих чи командних цілей. Його структура дозволяє легко масштабувати проєкт, додавати нові функції та інтегрувати з іншими сервісами у майбутньому.

За час розробки було значно поглиблено знання у сфері web-розробки, вдосконалено навички роботи з C#, ASP.NET Core, Entity Framework та SQL Server, а також отримано досвід проектування зручного інтерфейсу користувача та проведення практичного тестування.

Перспективи подальшого розвитку

Будь-який програмний продукт має розвиватися відповідно до змін у потребах користувачів. Уже наявні ідеї для реалізації нових функцій у наступних версіях застосунку, що дозволить підвищити його ефективність і зручність.

TimePlanner:

1. Інтеграція з календарями Google чи Outlook — щоб користувач міг синхронізувати свої задачі із зовнішніми подіями.
2. Push-сповіщення та email-нагадування — для ще більшої залученості та зниження ймовірності забути про важливі завдання.
3. Більш глибока аналітика — автоматичний аналіз продуктивності, рекомендації щодо розподілу часу, візуалізації звичок.
4. Підтримка спільних проектів — можливість працювати над задачами у групі, призначати ролі та відповідальних.
5. Мобільний застосунок або адаптивний PWA — для зручної роботи з будь-якого пристрою.
6. Розширення системи мотивації — додаткові рівні, бейджі, особисті цілі, що надихатимуть користувачів не зупинятись на досягнутому.

Крім того, планую поступово впроваджувати автоматизоване тестування, що дозволить підтримувати високу якість системи навіть при додаванні нового функціоналу.

Головне — зберігати простоту та зручність для користувача, не перевантажуючи інтерфейс зайвими деталями. Вірю, що TimePlanner стане незамінним помічником для тих, хто хоче організувати своє життя краще та не втрачати важливе серед повсякденної рутини.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Корляков Д.С. Розробка web-застосунку для планування особистого часу як міждисциплінарний іт-проєкт/Андрусевич Н.В., Корляков Д.С.// Матеріали IX Міжнародної наукової конференції «Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досліджень». Збірник тез. 13.06.2025, м. Луцьк. (подано до друку).

2. Корляков Д.С. Аналіз існуючих рішень у сфері web-планування особистого часу /Андрусевич Н.В., Корляков Д.С.// Матеріали IX Міжнародної студентської конференції Теоретичне та практичне застосування результатів сучасної науки. Збірник тез. 13.06.2025, м. Умань. (подано до друку).

ПЕРЕЛІК ПОСИЛАНЬ

1. Microsoft Learn. Офіційна документація та кращі практики ASP.NET Core. Доступно за посиланням: https://learn.microsoft.com/uk-ua/aspnet/core/

2. Microsoft Learn. Що нового у Entity Framework Core 9. Доступно за посиланням: https://learn.microsoft.com/en-us/ef/core/what-is-new/ef-core-9.0/whatsnew

3. Code Maze. Entity Framework Core Best Practices. Доступно за посиланням: https://code-maze.com/entity-framework-core-best-practices/

4. Dot Net Tutorials. .NET Developer Roadmap for 2024. Доступно за посиланням: https://dotnettutorials.net/lesson/net-developer-roadmap/

5. Toxigon. Кращі практики ASP.NET Core у 2024 році. Доступно за посиланням: https://toxigon.com/asp-net-core-best-practices-2024

6. MoldStud. 10 основних практик для Entity Framework Core. Доступно за посиланням: https://moldstud.com/articles/p-essential-best-practices-for-entity-framework-core-that-every-net-developer-needs-to-master

7. Eleken Blog. Як спроєктувати time-tracking застосунок для продуктивності. Доступно за посиланням: https://www.eleken.co/blog-posts/time-tracking-app-design-how-to-make-an-app-that-increases-productivity

8. Codedex. Як створити застосунок Pomodoro з HTML, CSS, JS. Доступно за посиланням: https://www.codedex.io/projects/build-a-pomodoro-app-with-html-css-js

9. Freshman.tech. Помодоро-таймер на JavaScript: покрокова інструкція. Доступно за посиланням: https://freshman.tech/pomodoro-timer/

10. Toggl. Кращі інструменти тайм-менеджменту 2024. Доступно за посиланням: https://toggl.com/blog/best-time-management-apps

11. ClientVenue. Кращі сервіси для відстеження часу для команд у 2024. Доступно за посиланням: https://clientvenue.com/blog/time-tracking-software-for-web-design-agencies

12. Cloudwards. Топ-10 інструментів тайм-менеджменту 2025. Доступно за посиланням: https://www.cloudwards.net/best-time-management-tools/

13. Traqq Blog. Статистика тайм-менеджменту працівників на 2024 рік. Доступно за посиланням: https://traqq.com/blog/employee-time-management-statistics-for-2024/

14. Beck K. *Test-Driven Development: By Example*. – Addison-Wesley, 2002.

15. Nielsen J. *Usability Engineering*. – Academic Press, 1993.

16. Gamma E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. – Addison-Wesley, 1994.

17. Freeman A., Sanderson A. *Pro ASP.NET Core MVC 2*. – Apress, 2017.

18. Troelsen A., Japikse P. *Pro C# 10 with .NET 6*. – Apress, 2022.

19. Microsoft Learn. ASP.NET Core Documentation – https://learn.microsoft.com/aspnet/core

20. ISO/IEC 25010:2011 – *Software Engineering – System and software quality models*. – https://iso.org

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для планування особистого часу

Виконав студент 4 курсу
групи ПД-41
Корляков Дмитро Сергійович
Керівник роботи
старший викладач кафедри ІПЗ Андрусевич Наталя Володимирівна

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи – створення web-застосунку для ефективного планування особистого часу з використанням сучасних технологій C#, ASP.NET Core та SQL Server.
- Об’єкт дослідження – процес організації та планування особистого часу користувачів.
- Предмет дослідження - програмне забезпечення для планування часу.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати сучасні підходи до тайм-менеджменту та існуючі застосунки.
2. Розробити архітектуру web-застосунку.
3. Реалізувати основний функціонал: календар, нагадування, створення задач.
4. Забезпечити адаптивний та зручний інтерфейс користувача.
5. Провести тестування застосунку.

3

АНАЛІЗ АНАЛОГІВ

Функції / Сервіси	Google Calendar	Microsoft To Do	Trello	Todoist	Notion	TimePlanner
Календар	Так	Обмежено	Через інтеграцію	Так	Так	Так
Управління задачами	Лише події	Так	Так	Так	Так	Так
Pomodoro таймер	Немає	Немає	Немає	Немає	Через шаблони	Вбудовано
Пріоритетність задач	Обмежено	Так	Через теги	Так	Через бази даних	Так
Теги / Категорії	Немає	Так	Так	Так	Так	Так
Командна робота	Так	Так	Так	Так	Так	Немає
Статистика / Звіти	Немає	Базова	Через плагіни	Так	Через шаблони	Є
Панель адміністратора	Немає	Немає	Через інтеграцію	Немає	Можливо налаштувати	Є
Локалізація українською	Часткова	Так	Так	Так	Так	Повна
Простота інтерфейсу	Так	Так	Інтуїтивно	Так	Потребує звикання	Зрозумілий і простий

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

- Функціональні:
  - додавати, редагувати, видаляти завдання;
  - встановлювати пріоритет для кожного завдання;
  - групувати завдання за категоріями або тегами;
  - переглядати свої завдання у вигляді календаря;
  - відзначати виконані завдання;
  - переглядати статистику виконання (наприклад, кількість виконаних/прострочених завдань);
  - можливість створювати повторювані завдання;
  - шукати завдання за ключовими словами;
  - легкий та зрозумілий інтерфейс для взаємодії з системою.

5

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

- Нефункціональні:
  - швидка робота інтерфейсу;
  - безпека зберігання даних користувача;
  - можливість масштабування при зростанні кількості користувачів;
  - адаптивність під різні пристрої (комп’ютер, планшет, смартфон);
  - надійність (мінімізація ризику втрати даних);
  - простота впровадження та супроводу.

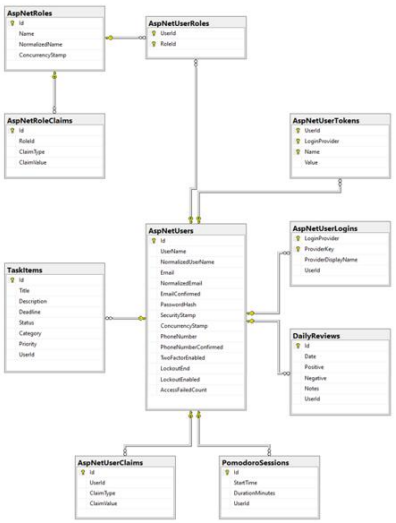
6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

- Мова програмування: C#
- Framework: ASP.NET Core
- Frontend: Razor Pages / MVC Views
- База даних: SQL Server
- Інші інструменти: Git, Figma

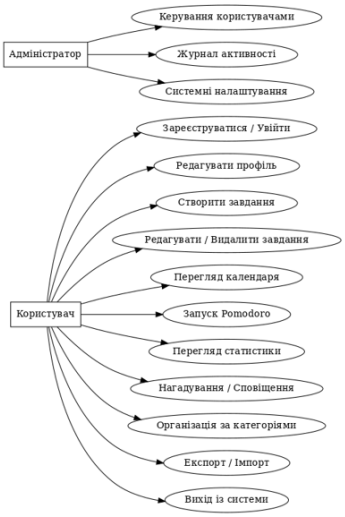
7

ER-ДІАГРАМА ЗВ’ЯЗКІВ



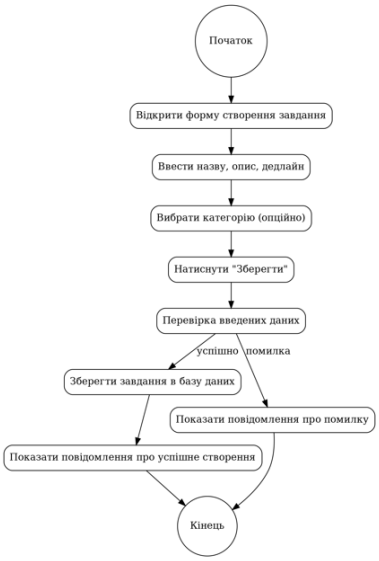
8

ДІАГРАМА ПРЕЦЕДЕНТІВ



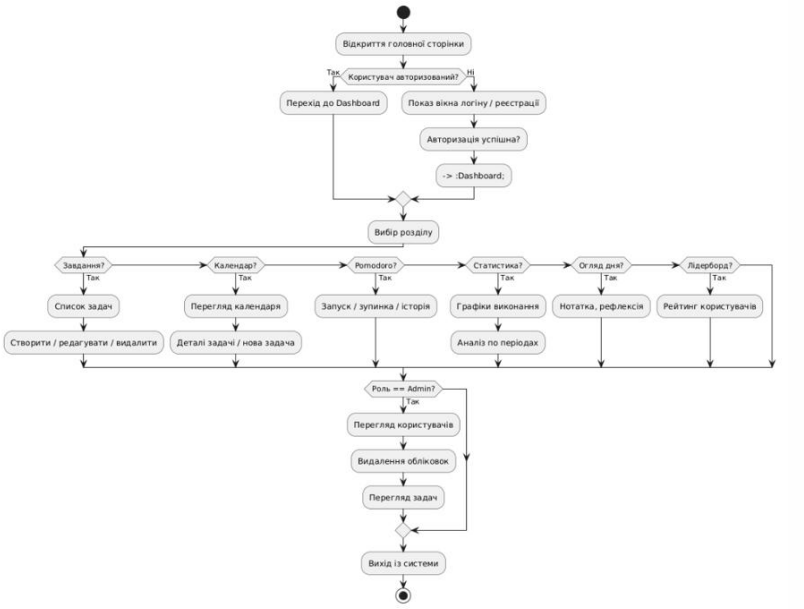
9

Діаграма діяльності для процесу додавання завдання в TimePlanner

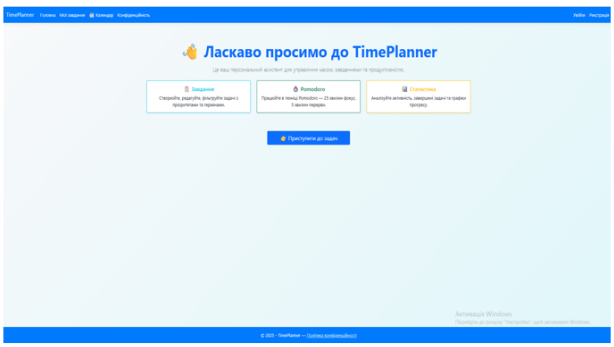


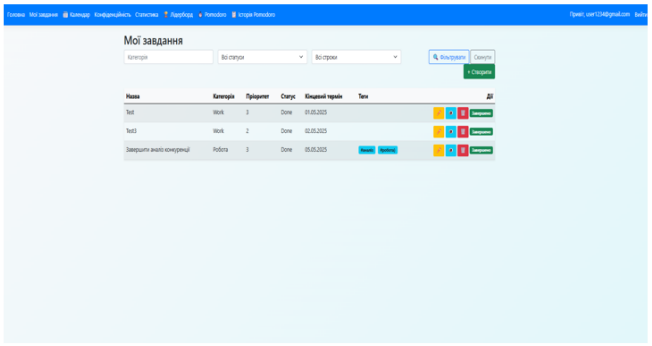
10

Алгоритм роботи програми

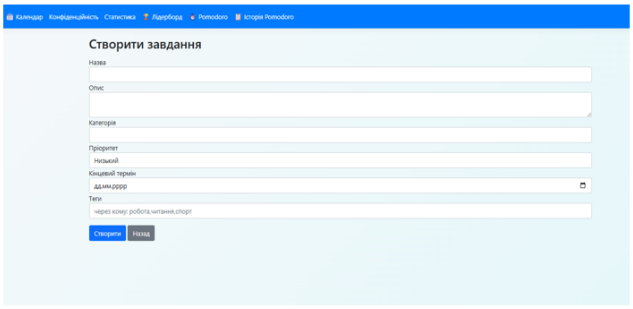


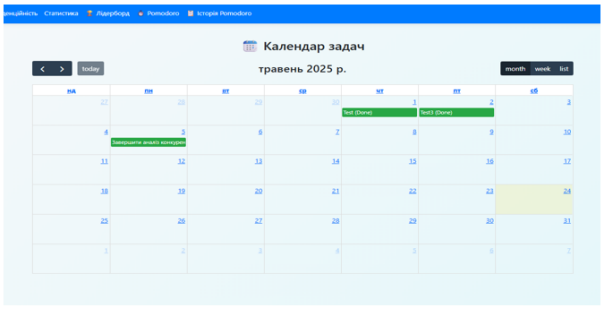
ЕКРАННІ ФОРМИ





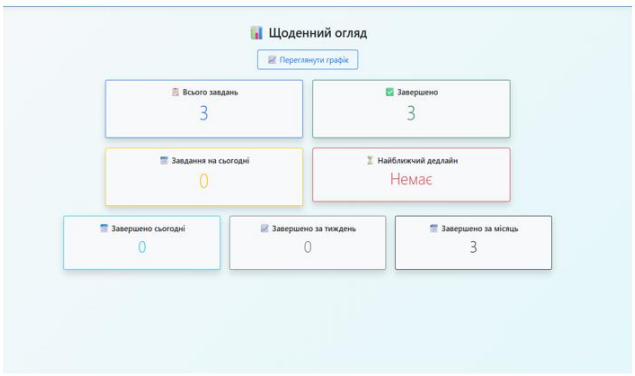
ЕКРАННІ ФОРМИ





13

ЕКРАННІ ФОРМИ



14

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Після тестування збиралися коментарі, оцінки зручності інтерфейсу та повідомлення про помилки.

Показник	Результат
Успішність реєстрації	100% (30/30)
Успішне створення завдань	97% (29/30)
Коректна робота Pomodoro	93% (28/30)
Задоволеність інтерфейсом	90% (27/30) позитивних відгуків
Частота помилок	2 випадки некоректного збереження
Середній час на створення завдання	1-2 хвилини

Виявлені недоліки

У 2 випадках завдання не збереглися через тимчасові проблеми з інтернет-з'єднанням.
Деякі користувачі зазначили, що хотіли б мати більше налаштувань для персоналізації інтерфейсу.
Пропозиції щодо додавання функції синхронізації з календарями Google/Outlook.

15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Корляков Д.С. Розробка web-застосунку для планування особистого часу як міждисциплінарний іт-проєкт/Андрусевич Н.В., Корляков Д.С.// Матеріали IX Міжнародної наукової конференції «Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досліджень». Збірник тез. 13.06.2025, м. Луцьк. (подано до друку).
- Корляков Д.С. Аналіз існуючих рішень у сфері web-планування особистого часу /Андрусевич Н.В., Корляков Д.С.// Матеріали IX Міжнародної студентської конференції Теоретичне та практичне застосування результатів сучасної науки. Збірник тез. 13.06.2025, м. Умань. (подано до друку).

16

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були успішно реалізовані всі поставлені завдання:
1. Аналіз сучасних підходів до тайм-менеджменту та існуючих застосунків дозволив виявити ключові потреби користувачів та типові недоліки аналогічних систем. На основі проведеного дослідження сформульовано вимоги до власного продукту та визначено функціональні пріоритети.
2. Архітектура web-застосунку була спроєктована з урахуванням вимог до масштабованості, підтримки, безпеки та розширюваності. Було реалізовано розподіл на клієнтську і серверну частини, використано перевірені технології — C#, ASP.NET Core, Razor Pages, Entity Framework Core, SQL Server.
3. Основний функціонал, зокрема створення задач, робота з календарем, налаштування нагадувань, був повністю реалізований. Застосунок підтримує CRUD-операції для завдань, дозволяє працювати з категоріями та фільтрами, інтегрує таймер Pomodoro та формує базову статистику активності.
4. Інтерфейс користувача розроблено з орієнтацією на простоту, мінімалізм і адаптивність. Особлива увага приділялася зручності взаємодії — від форми створення завдання до візуального подання розкладу.
5. Тестування застосунку, проведене із залученням 30 користувачів, підтвердило стабільність роботи, зрозумілість інтерфейсу та актуальність реалізованого функціоналу. Було зібрано цінні відгуки, які вже враховано у планах подальшого розвитку.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using WebApplication3.Data;
using WebApplication3.Models;
using Microsoft.Extensions.DependencyInjection;

var builder = WebApplication.CreateBuilder(args);

// Подключаем EF + SQL Server
builder.Services.AddDbContext<ApplicationDbCo
ntext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionS
tring("DefaultConnection")));

// Добавляем Identity + роли
builder.Services.AddDefaultIdentity<ApplicationU
ser>(options =>
{
    options.SignIn.RequireConfirmedAccount =
false;
})
.AddRoles<IdentityRole>()
.AddEntityFrameworkStores<ApplicationDbConte
xt>();

//
builder.Services.ConfigureApplicationCookie(optio
ns =>
{
    options.LoginPath = "/Account/Login";
    options.AccessDeniedPath = "/Account/Login";
});

// Только MVC, без Razor Pages
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Middleware
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

// Роутинг
app.MapControllerRoute(
    name: "default",
    pattern:
"{controller=Home}/{action=Index}/{id?}");

// 👑 Сидирование администратора
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var roleManager =
services.GetRequiredService<RoleManager<IdentityRole>>(
);
    var userManager =
services.GetRequiredService<UserManager<ApplicationUser
>>();

    string adminEmail = "admin@example.com";
    string adminPassword = "Admin123!";

    if (!await
roleManager.RoleExistsAsync("Admin"))
        await roleManager.CreateAsync(new
IdentityRole("Admin"));

    if (await
userManager.FindByEmailAsync(adminEmail) == null)
    {
        var admin = new ApplicationUser {
UserName = adminEmail, Email = adminEmail };
        var result = await
userManager.CreateAsync(admin, adminPassword);
        if (result.Succeeded)
        {
            await
userManager.AddToRoleAsync(admin, "Admin");
        }
    }
}

app.Run();


using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebApplication3.Data;
using WebApplication3.Models;

namespace WebApplication3.Controllers
{
    [Authorize(Roles = "Admin")]
    public class AdminController : Controller
    {
        private readonly
UserManager<ApplicationUser> _userManager;
        private readonly ApplicationDbContext
_context;

        public
AdminController(UserManager<ApplicationUser>
userManager, ApplicationDbContext context)
        {
            _userManager = userManager;
            _context = context;
        }

        public IActionResult Index()
        {
            var users = _userManager.Users.ToList();
            return View(users);
```

```
        }

        [HttpPost]
        public async Task<IActionResult>
Delete(string id)
        {
            var user = await
_userManager.FindByIdAsync(id);
            if (user != null && user.Email !=
User.Identity?.Name)
            {
                await _userManager.DeleteAsync(user);
            }
            return RedirectToAction("Index");
        }

        // GET: /Admin/Details/id
        public async Task<IActionResult>
Details(string id)
        {
            if (id == null) return NotFound();

            var user = await
_userManager.FindByIdAsync(id);
            if (user == null) return NotFound();

            var tasks = await _context.TaskItems
                .Where(t => t.UserId == id)
                .ToListAsync();

            ViewBag.UserEmail = user.Email;
            return View(tasks);
        }
    }
}


@model
WebApplication3.Models.ChartViewModel

@{
    ViewData["Title"] = "Графік задач";
}

<div class="container mt-5 text-center">
    <h2 class="mb-4">📈 Графік завершених
задач</h2>

    <p class="text-muted mb-4">
        Цей графік показує динаміку завершення
задач за датою дедлайну. Вісь X — дати, вісь Y —
кількість завершених задач.
    </p>

    <canvas id="tasksChart" width="400"
height="200"></canvas>

    <div class="mt-4 d-flex justify-content-center
gap-3">
        <a asp-action="Daily" class="btn btn-outline-
secondary">← Назад до огляду</a>
        <button onclick="downloadChart()"
class="btn btn-primary">⬇️ Завантажити графік</button>
    </div>
</div>

@section Scripts {
    <script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script>
        const ctx =
document.getElementById('tasksChart').getContext('2d');

        const gradient = ctx.createLinearGradient(0, 0,
0, 400);
        gradient.addColorStop(0, 'rgba(0, 123, 255,
0.4)');
        gradient.addColorStop(1, 'rgba(0, 123, 255,
0)');

        const tasksChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels:
@Html.Raw(Json.Serialize(Model.Dates)),
                datasets: [{
                    label: 'Завершено задач',
                    data:
@Html.Raw(Json.Serialize(Model.CompletedCounts)),
                    backgroundColor: gradient,
                    borderColor: 'rgba(0, 123, 255, 1)',
                    borderWidth: 2,
                    pointRadius: 4,
                    pointHoverRadius: 6,
                    tension: 0.35,
                    fill: true
                }]
            },
            options: {
                responsive: true,
                animation: {
                    duration: 1200,
                    easing: 'easeOutQuart'
                },
                scales: {
                    y: {
                        beginAtZero: true,
                        title: {
                            display: true,
                            text: 'Кількість задач'
                        },
                        ticks: {
                            stepSize: 1
                        }
                    },
                    x: {
                        title: {
                            display: true,
                            text: 'Дата'
                        }
                    }
                },
                plugins: {
                    legend: {
                        display: true,
                        position: 'top'
                    },
                    tooltip: {
                        callbacks: {
                            label: (ctx) => ` ${ctx.parsed.y}
завершено`
                        }
                    }
                }
            }
        });

        function downloadChart() {
            const link = document.createElement('a');
            link.download = 'tasks-chart.png';
```

```
                link.href = tasksChart.toBase64Image();
                link.click();
            }
        </script>
    }


    @using Microsoft.AspNetCore.Identity
    @using WebApplication3.Models
    @inject SignInManager<ApplicationUser>
SignInManager
    @inject UserManager<ApplicationUser>
UserManager

    <!DOCTYPE html>
    <html lang="uk">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
        <title>@ViewData["Title"] -
TimePlanner</title>
        <link rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
        <link rel="stylesheet" href="~/css/site.css" asp-
append-version="true" />
        <link rel="stylesheet"
href="~/WebApplication3.styles.css" asp-append-
version="true" />
        <style>
            html, body {
                height: 100%;
                margin: 0;
            }

            .page-wrapper {
                display: flex;
                flex-direction: column;
                min-height: 100vh;
            }

            main {
                flex: 1;
            }

            body {
                background: linear-gradient(135deg,
#f8f9fa, #e0f7fa);
            }

            .navbar {
                background: #007bff !important;
            }

                .navbar .nav-link,
                .navbar .navbar-brand {
                    color: #ffffff !important;
                }

                    .navbar .nav-link:hover {
                        color: #d1ecf1 !important;
                    }

            footer {
                background: #007bff;
                color: white;
            }

                footer a {
                    color: #d1ecf1;
                }

                    footer a:hover {
                        color: #ffffff;
                    }
        </style>
    </head>
    <body>
        <div class="page-wrapper">
            <header>
                <nav class="navbar navbar-expand-sm
navbar-toggleable-sm navbar-dark border-bottom box-
shadow mb-3">
                    <div class="container-fluid">
                        <a class="navbar-brand" asp-area=""
asp-controller="Home" asp-
action="Index">TimePlanner</a>
                        <button class="navbar-toggler"
type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"
                                aria-
controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
                            <span class="navbar-toggler-
icon"></span>
                        </button>
                        <div class="navbar-collapse collapse
d-sm-inline-flex justify-content-between">
                            <ul class="navbar-nav flex-grow-
1">
                                <li class="nav-item"><a
class="nav-link" asp-controller="Home" asp-
action="Index">Головна</a></li>
                                <li class="nav-item"><a
class="nav-link" asp-controller="Task" asp-
action="Index">Мої завдання</a></li>
                                <li class="nav-item"><a
class="nav-link" asp-controller="Task" asp-
action="Calendar">🗓 Календар</a></li>
                                <li class="nav-item"><a
class="nav-link" asp-controller="Home" asp-
action="Privacy">Конфіденційність</a></li>
                                @if
(SignInManager.IsSignedIn(User))
                                {
                                    <li class="nav-item"><a
class="nav-link" asp-controller="Review" asp-
action="Daily">Статистика</a></li>
                                    <li class="nav-item"><a
class="nav-link" asp-controller="Leaderboard" asp-
action="Index">🏆 Лідерборд</a></li>
                                    <li class="nav-item"><a
class="nav-link" asp-controller="Pomodoro" asp-
action="Index">⏱ Pomodoro</a></li>
                                    <li class="nav-item"><a
class="nav-link" asp-controller="Pomodoro" asp-
action="History">📋 Історія Pomodoro</a></li>
                                    @if (User.IsInRole("Admin"))
                                    {
                                        <li class="nav-item"><a
class="nav-link" asp-controller="Admin" asp-
action="Index">👑 Панель адміністратора</a></li>
                                    }
                                }
                            </ul>
                            <ul class="navbar-nav">
                                @if
(SignInManager.IsSignedIn(User))
                                {
```

```
                        <li class="nav-item"><span
class="nav-link text-light">Привіт,
@User.Identity?.Name</span></li>
                        <li class="nav-item">
                            <form asp-
controller="Account" asp-action="Logout" method="post"
class="form-inline">
                                <button type="submit"
class="nav-link btn btn-link text-light">Вийти</button>
                            </form>
                        </li>
                    }
                    else
                    {
                        <li class="nav-item"><a
class="nav-link text-light" asp-controller="Account" asp-
action="Login">Увійти</a></li>
                        <li class="nav-item"><a
class="nav-link text-light" asp-controller="Account" asp-
action="Register">Реєстрація</a></li>
                    }
                </ul>
            </div>
        </div>
    </nav>
</header>

<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top py-3 mt-auto">
    <div class="container text-center">
        &copy; 2025 - TimePlanner — <a asp-
controller="Home" asp-action="Privacy">Політика
конфіденційності</a>
    </div>
</footer>
</div>

<script
src="~/lib/jquery/dist/jquery.min.js"></script>
<script
src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script
>
<script src="~/js/site.js" asp-append-
version="true"></script>
@await RenderSectionAsync("Scripts", required:
false)
</body>
</html>


@model
WebApplication3.Models.ReviewViewModel

@{
    ViewData["Title"] = "Щоденний огляд";
}

<div class="container mt-4">
    <h2 class="mb-4 text-center">📊 Щоденний
огляд</h2>

    <div class="text-center mb-4">
        <a asp-action="Chart" class="btn btn-outline-
primary btn-lg px-4">
            📈 Переглянути графік
        </a>
    </div>

    <div class="row g-4 justify-content-center">

        <div class="col-md-5">
            <div class="card shadow bg-light border-
primary">
                <div class="card-body text-center">
                    <h5 class="card-title">📋 Всього
завдань</h5>
                    <p class="display-5 text-
primary">@Model.TotalTasks</p>
                </div>
            </div>
        </div>

        <div class="col-md-5">
            <div class="card shadow bg-light border-
success">
                <div class="card-body text-center">
                    <h5 class="card-title">✅
Завершено</h5>
                    <p class="display-5 text-
success">@Model.CompletedTasks</p>
                </div>
            </div>
        </div>

        <div class="col-md-5">
            <div class="card shadow bg-light border-
warning">
                <div class="card-body text-center">
                    <h5 class="card-title">🗓️ Завдання
на сьогодні</h5>
                    <p class="display-5 text-
warning">@Model.TodayTasks</p>
                </div>
            </div>
        </div>

        <div class="col-md-5">
            <div class="card shadow bg-light border-
danger">
                <div class="card-body text-center">
                    <h5 class="card-title">⏳
Найближчий дедлайн</h5>
                    <p class="display-6 text-danger">
                        @(Model.NearestDeadline ??
"Немає")
                    </p>
                </div>
            </div>
        </div>

        <!-- Нові картки -->

        <div class="col-md-4">
            <div class="card shadow bg-light border-
info">
                <div class="card-body text-center">
                    <h5 class="card-title">🗓️ Завершено
сьогодні</h5>
                    <p class="display-6 text-
info">@Model.CompletedToday</p>
                </div>
            </div>
        </div>
```

```
                <div class="col-md-4">
                    <div class="card shadow bg-light border-
secondary">
                        <div class="card-body text-center">
                            <h5 class="card-title">📈 Завершено
за тиждень</h5>
                            <p class="display-6 text-
secondary">@Model.CompletedThisWeek</p>
                        </div>
                    </div>
                </div>

                <div class="col-md-4">
                    <div class="card shadow bg-light border-
dark">
                        <div class="card-body text-center">
                            <h5 class="card-title">🗓️ Завершено
за місяць</h5>
                            <p class="display-6 text-
dark">@Model.CompletedThisMonth</p>
                        </div>
                    </div>
                </div>

            </div>
        </div>


        @model WebApplication3.Models.TaskItem

        @{
            ViewData["Title"] = "Деталі завдання";
        }

        <h2>@Model.Title</h2>

        <dl class="row">
            <dt class="col-sm-3">Опис</dt>
            <dd class="col-sm-
9">@Model.Description</dd>

            <dt class="col-sm-3">Категорія</dt>
            <dd class="col-sm-9">@Model.Category</dd>

            <dt class="col-sm-3">Пріоритет</dt>
            <dd class="col-sm-9">@Model.Priority</dd>

            <dt class="col-sm-3">Статус</dt>
            <dd class="col-sm-9">@Model.Status</dd>

            <dt class="col-sm-3">Кінцевий термін</dt>
            <dd class="col-sm-
9">@Model.Deadline?.ToShortDateString()</dd>

            <dt class="col-sm-3">Теги</dt>
            <dd class="col-sm-9">
                @{
                    var tags = (Model.Description ?? "")
                    .Split(' ',
StringSplitOptions.RemoveEmptyEntries)
                    .Where(t => t.StartsWith("#"))
                    .Distinct();

                    if (tags.Any())
                    {
                        foreach (var tag in tags)
                        {
                            <span class="badge bg-info text-dark
me-1">@tag</span>
                        }
                    }
                    else
                    {
                        <span class="text-muted">Немає
тегів</span>
                    }
                }
            </dd>
        </dl>

        <a asp-action="Edit" asp-route-id="@Model.Id"
class="btn btn-warning">Редагувати</a>
        <a asp-action="Index" class="btn btn-
secondary">Назад</a>


        @model
IEnumerable<WebApplication3.Models.TaskItem>

        @{
            ViewData["Title"] = "Мої завдання";
        }

        <h2>Мої завдання</h2>

        <form asp-action="Index" method="get"
class="row g-3 mb-4">
            <div class="col-md-3">
                <input type="text" name="category"
class="form-control" placeholder="Категорія" />
            </div>

            <div class="col-md-3">
                <select name="status" class="form-select">
                    <option value="">Всі статуси</option>
                    <option value="Todo">До
виконання</option>
                    <option value="InProgress">В
процесі</option>
                    <option
value="Done">Завершено</option>
                </select>
            </div>

            <div class="col-md-3">
                <select name="due" class="form-select">
                    <option value="">Всі строки</option>
                    <option value="today">Сьогодні</option>
                    <option
value="upcoming">Наближаються</option>
                    <option
value="overdue">Прострочені</option>
                </select>
            </div>

            <div class="col-md-3 text-end">
                <button type="submit" class="btn btn-outline-
primary">🔍 Фільтрувати</button>
                <a asp-action="Index" class="btn btn-outline-
secondary">Скинути</a>
                <a asp-action="Create" class="btn btn-success
ms-2">+ Створити</a>
            </div>
        </form>

        <table class="table table-striped table-hover
shadow-sm">
            <thead class="table-light">
                <tr>
                    <th>Назва</th>
```

```
                <th>Категорія</th>
                <th>Пріоритет</th>
                <th>Статус</th>
                <th>Кінцевий термін</th>
                <th>Теги</th>
                <th class="text-end">Дії</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
                    <td>@item.Title</td>
                    <td>@item.Category</td>
                    <td>@item.Priority</td>
                    <td>@item.Status</td>

<td>@item.Deadline?.ToString("dd.MM.yyyy")</td>
                    <td>
                        @{
                            var tags = (item.Description ?? "")
                            .Split(' ',
StringSplitOptions.RemoveEmptyEntries)
                            .Where(x => x.StartsWith("#"))
                            .Distinct()
                            .ToList();
                        }
                        @foreach (var tag in tags)
                        {
                            <span class="badge bg-info text-
dark me-1">@tag</span>
                        }
                    </td>
                    <td class="text-end">
                        <a asp-action="Edit" asp-route-
id="@item.Id" class="btn btn-sm btn-warning">🖍️</a>
                        <a asp-action="Details" asp-route-
id="@item.Id" class="btn btn-sm btn-info">👁</a>
                        <a asp-action="Delete" asp-route-
id="@item.Id" class="btn btn-sm btn-danger">🗑</a>
                        @if (item.Status !=
WebApplication3.Models.TaskStatus.Done)
                        {
                            <form asp-action="Complete" asp-
route-id="@item.Id" method="post" style="display:inline;">
                                <button type="submit" class="btn
btn-sm btn-success">✅ Завершити</button>
                            </form>
                        }
                        else
                        {
                            <span class="badge bg-
success">Завершено</span>
                        }
                    </td>
                </tr>
            }
        </tbody>
    </table>


    @model WebApplication3.Models.TaskItem

    @{
        ViewData["Title"] = "Редагувати завдання";
    }

    <h2>Редагувати завдання</h2>

    <form asp-action="Edit">
        <input type="hidden" asp-for="Id" />

        <div class="form-group">
            <label asp-for="Title">Назва</label>
            <input asp-for="Title" class="form-control" />
        </div>

        <div class="form-group">
            <label asp-for="Description">Опис</label>
            <textarea asp-for="Description" class="form-
control"></textarea>
        </div>

        <div class="form-group">
            <label asp-for="Category">Категорія</label>
            <input asp-for="Category" class="form-
control" />
        </div>

        <div class="form-group">
            <label asp-for="Priority">Пріоритет</label>
            <select asp-for="Priority" class="form-
control">
                <option value="1">Низький</option>
                <option value="2">Середній</option>
                <option value="3">Високий</option>
            </select>
        </div>

        <div class="form-group">
            <label asp-for="Deadline">Кінцевий
термін</label>
            <input asp-for="Deadline" class="form-
control" type="date" />
        </div>

        <div class="form-group">
            <label asp-for="Status">Статус</label>
            <select asp-for="Status" class="form-
control">
                <option value="0">До
виконання</option>
                <option value="1">В процесі</option>
                <option value="2">Завершено</option>
            </select>
        </div>

        <div class="form-group">
            <label for="Tags">Теги (через кому)</label>
            <input name="Tags" class="form-control"
value="@Context.Request.Form["Tags"]" />
        </div>

        <button type="submit" class="btn btn-success
mt-3">Зберегти</button>
        <a asp-action="Index" class="btn btn-secondary
mt-3">Скасувати</a>
    </form>
```