МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА З ОСОБИСТОГО ТАЙМ-МЕНЕДЖМЕНТУ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувачка _____ Анастасія ВОЛОШКО

«___»_____ 2026 р.

Керівник канд. техн. наук, доцент _____ Інесса КУЛАКОВСЬКА

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувачки

Волошко Анастасія Сергіївна
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система з особистого тайм-менеджменту».

Керівник роботи: Кулаковська Інесса Василівна, доцент кафедри інтелектуальних інформаційних систем, кандидат фіз.-мат. наук.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: розроблена інтелектуальна вебсистема з особистого тайм-менеджменту з адаптивним інтерфейсом, що забезпечує планування завдань, автоматичне розміщення задач у розкладі та генерацію персональних рекомендацій, реалізована на основі Laravel, MySQL та Chart.js.

4. Перелік питань, що підлягають розробці: аналіз проблематики персонального тайм-менеджменту та потреб цільової аудиторії; огляд існуючих систем планування часу; дослідження методів інтелектуального аналізу розкладу та алгоритмів автоматичного планування; розробка математичної моделі оцінки часових слотів; проєктування архітектури системи за патерном MVC; розробка структури бази даних з дотриманням нормалізації; реалізація серверної частини на Laravel 10 з безпековими механізмами; створення адаптивного користувацького інтерфейсу; розробка модулів інтелектуального радника та автопланера задач; тестування функціональності та продуктивності системи.

5. Перелік графічних матеріалів: таблиці, рисунки, презентація.

Керівник роботи _____ Інесса КУЛАКОВСЬКА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувачка _____ Анастасія ВОЛОШКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «21» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система з особистого тайм-менеджменту.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	Виконано
2	Аналіз предметної області та постановка задачі	25.12.2025	30.01.2026	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем щодо вебзастосунків з тайм-менеджменту	31.01.2026	01.03.2026	Виконано
4	Огляд існуючих архітектур вебзастосунків та технологій для реалізації систем планування часу й автоматизації задач	02.03.2026	01.04.2026	Виконано
5	Реалізація обраних технологій із аналізом отриманих результатів у вебзастосунку для особистого тайм-менеджменту	02.04.2026	24.05.2026	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
7	Корегування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
9	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Інесса КУЛАКОВСЬКА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувачка _____ Анастасія ВОЛОШКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 401 ЧНУ ім. Петра Могили

Волошко Анастасії Сергіївни

на тему: «**ІНТЕЛЕКТУАЛЬНА СИСТЕМА З ОСОБИСТОГО ТАЙМ-МЕНЕДЖМЕНТУ**»

Актуальність даного дослідження полягає в необхідності створення інтелектуальних інструментів планування часу, які не лише фіксують задачі користувача, а й активно допомагають у прийнятті рішень щодо їх розміщення в розкладі. У сучасних умовах зростаючого інформаційного навантаження ефективне управління часом стало однією з ключових складових особистого успіху, проте більшість наявних рішень обмежуються функціями календаря або списку задач без розумних рекомендацій та автоматизованого планування.

Об'єктом роботи є процеси персонального планування часу та інтелектуального аналізу розкладу користувача.

Предметом роботи є методи інтелектуального аналізу розкладу (статистичний аналіз, виявлення продуктивних годин, аналіз навантаження, пошук вільних слотів, правилова система генерації порад), алгоритм автопланування задач з оцінкою часових слотів за функцією score(hour, preferMorning), а також технології Laravel 10, PHP 8.4, MySQL 8.4, JavaScript (ES6+) і Chart.js для реалізації інтелектуального вебзастосунку тайм-менеджменту.

Метою роботи є розробка інтелектуальної вебсистеми, що забезпечить ефективне планування завдань користувача, автоматичне розміщення задач у розкладі на основі пріоритетів та зайнятості, а також надання персоналізованих рекомендацій з управління часом.

Пояснювальна записка кваліфікаційної роботи складається зі вступу, чотирьох розділів, висновків і 2 додатків. У першому розділі розглянуто предметну область, проаналізовано цільову аудиторію та існуючі аналоги (Worksection, Google Calendar, TickTick, Notion, Any.do), сформульовано постановку задачі. У другому розділі досліджено методи інтелектуального аналізу розкладу, розроблено

алгоритм автопланера задач, побудовано математичну модель оцінки часових слотів, обґрунтовано вибір технологій (Laravel 10, MySQL 8.4, Bootstrap 5, Chart.js). У третьому розділі представлено проєктування системи: описано сценарії використання, архітектуру MVC, структуру бази даних, моделювання восьми функціональних модулів, створено макети дизайну системи. У четвертому розділі здійснено програмну реалізацію вебзастосунку, наведено приклади реалізації серверної та клієнтської частин, описано адаптивний інтерфейс із підтримкою світлої та темної теми, виконано тестування системи.

Кваліфікаційна робота містить 92 сторінки, 26 рисунків, 13 таблиць, 30 джерел посилання та 2 додатки.

Ключові слова: тайм-менеджмент, інтелектуальна система, автопланер, Laravel, PHP, MySQL, MVC, аналітика продуктивності, адаптивний вебзастосунок.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Voloshko Anastasiia

“INTELLIGENT SYSTEM FOR PERSONAL TIME MANAGEMENT”

The relevance of this research lies in the need to create intelligent time-planning tools that not only record user tasks but actively assist in decision-making regarding their placement in the schedule. In conditions of increasing information overload, effective time management has become one of the key components of personal success, yet most existing solutions are limited to calendar or task-list functions without smart recommendations and automated planning.

The object of the work is the processes of personal time planning and intelligent schedule analysis.

The subject of the work is the methods of intelligent schedule analysis (statistical analysis, productive hours detection, workload analysis, free slot search, rule-based advice generation), a task auto-scheduling algorithm with time slot scoring via the score(hour, preferMorning) function, and the technologies Laravel 10, PHP 8.4, MySQL 8.4, JavaScript (ES6+), and Chart.js for implementing an intelligent time management web application.

The goal of the work is to develop an intelligent web system that ensures effective task planning, automatic placement of tasks in the schedule based on priorities and availability, and the generation of personalized time-management recommendations.

The explanatory note of the qualifying work consists of an introduction, four chapters, conclusions, and 2 appendices. The first chapter examines the subject area, analyzes the target audience and existing analogues (Worksection, Google Calendar, TickTick, Notion, Any.do), and formulates the task statement. The second chapter investigates methods of intelligent schedule analysis, develops the auto-planner algorithm, constructs a mathematical model for time-slot scoring, and justifies the choice of technologies (Laravel 10, MySQL 8.4, Bootstrap 5, Chart.js). The third chapter

presents the system design: use-case scenarios, MVC architecture, database structure, modeling of eight functional modules are described, and system design mockups are created. The fourth chapter covers the software implementation of the web application, including examples of server-side and client-side code, an adaptive user interface with light and dark theme support, and system testing.

The qualifying work contains 92 pages, 26 figures, 13 tables, 30 references, and 2 appendices.

Keywords: time management, intelligent system, auto-planner, Laravel, PHP, MySQL, MVC, productivity analytics, adaptive web application.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

API – Application Programming Interface (інтерфейс програмування додатків)

CRUD – Create, Read, Update, Delete (створення, читання, оновлення, видалення)

CSRF – Cross-Site Request Forgery (міжсайтова підробка запиту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

JSON – JavaScript Object Notation

MVC – Model-View-Controller

ORM – Object-Relational Mapping (об'єктно-реляційне відображення)

PHP – Hypertext Preprocessor

SQL – Structured Query Language (мова структурованих запитів)

UI – User Interface (інтерфейс користувача)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

ВСТУП

Актуальність. У сучасних умовах, коли обсяг інформації зростає щодня разом із збільшенням кількості завдань, управління часом відіграє важливу роль у досягненні успіху. Аспект управління часом є ще одним важливим елементом для покращення та підвищення продуктивності, зниження рівня стресу, а також допомагає краще відпочивати. Але інструменти, які ми маємо сьогодні – паперові планувальники, календарі, списки завдань – зазвичай неповні: вони вимагають постійного самоконтролю користувача, не адаптуються до його особистих особливостей і не допомагають користувачам орієнтуватися у найкращому розподілі їхнього часу.

Інтелектуальні системи управління часом, що використовують алгоритми аналізу розкладу, автоматизоване розміщення завдань та генерацію персоналізованих рекомендацій, можуть значно спростити процес планування. Ці системи не просто підтримують список справ; вони допомагають користувачеві знайти найкращий час для виконання роботи, визначити, які частини перевантажені або недовантажені, рекомендують перерозподілити роботу, вимірюють рівні продуктивності в різний час доби та протягом тижня. Саме тому розробка інтелектуальної системи особистого управління часом є цікавим завданням, оскільки вона може автоматизувати рішення про планування та надавати користувачеві зрозумілі метрики для моніторингу та підвищення їхньої продуктивності.

Мета роботи – розробка інтелектуальної вебсистеми з особистого тайм-менеджменту, яка забезпечить ефективне планування завдань користувача, автоматичне розміщення задач у розкладі на основі пріоритетів та зайнятості, а також надання персоналізованих рекомендацій з управління часом.

Об'єкт роботи – процеси персонального планування часу та розробки інтелектуальних систем підтримки прийняття рішень у сфері тайм-менеджменту.

Предмет роботи – методи інтелектуального аналізу розкладу, алгоритм автопланування задач із математичною моделлю оцінки часових слотів, а також

технологічний стек на основі Laravel 10 (PHP 8.4), MySQL 8.4 та JavaScript (ES6+) з Chart.js для побудови інтелектуальної вебсистеми особистого тайм-менеджменту.

Для досягнення вищезазначеного необхідно вирішити такі **завдання**:

– вивчити обмеження особистого та маркетингового управління часом з урахуванням цільової аудиторії та їхніх вимог – дослідити та зрозуміти сучасну систему планування часу та її переваги та недоліки – вивчити підходи, методи інтелектуального розкладу та алгоритму автоматичного планування завдань;

– обґрунтувати вибір технологічного стека;

– розробити архітектуру системи, структуру бази даних;

– розробити модулі для управління розкладом, повторюваними завданнями, категоріями, аналітикою, інтелектуальними рекомендаціями;

– реалізувати інтуїтивно зрозумілий адаптивний інтерфейс користувача з підтримкою світлої та темної тем;

– провести належне тестування системи та результатів.

Практичне значення роботи полягає у створенні готового до використання інтелектуального інструменту, який може застосовуватися студентами, фрілансерами, керівниками та будь-якими особами, що прагнуть підвищити власну продуктивність через системне планування часу.

Декларація про використання ШІ. Під час підготовки наукової роботи було використано інструмент Claude Opus 4.8 від Anthropic. Його було застосовано для таких допоміжних завдань: генерація ідей для структури розділів; оптимізація великих частин коду; перефразування речень у третьому та четвертому розділі задля академічного стилю тексту. Весь основний текст, аналіз, інтерпретація даних та висновки є результатом моєї власної інтелектуальної праці. Я перевірила всю інформацію, отриману від ШІ, на точність та достовірність і несу повну відповідальність за зміст цієї роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ. ПОСТАНОВКА ЗАДАЧІ

Сучасний стан життя означає, що управління часом стало основною компетенцією. Велика кількість завдань, обов'язків та постійний потік інформації вимагають існування систем, які роблять більше, ніж просто фіксують плани – вони дозволяють їх логічно організувати [1].

1.1 Актуальність задачі персонального тайм-менеджменту

Управління часом як предмет розширює продуктивність часу, як з боку індивіда, так і організації. Старі методи планування (матриця Ейзенхауера, техніка «Pomodoro», метод GTD (Getting Things Done)) є дуже ефективними; однак, вони також вимагають постійного ручного управління користувачем. У цифрову епоху багато з цих інструментів пропонують певний рівень автоматизації в плануванні, але в більшості випадків ці інструменти є лише списками завдань та функціями календаря. Але в цей час в історії користувача проблема сучасного користувача полягає в тому, що інтелектуалізація є меншою в сучасну епоху. Замість цього календар є записом, і немає пропозицій щодо розподілу навантаження, немає попередження про перевантажені дні, немає часу для нового завдання з урахуванням існуючих зобов'язань [2, 3].

Інтелектуальна система управління часом повинна надавати рішення для кількох завдань:

– централізоване зберігання завдань, включаючи категоризацію, пріоритети та загальні шаблони;

– автоматичний пошук вільного часу для нових завдань на основі пріоритетів та навантаження;

– аналіз продуктивності користувача та ідентифікація поведінкових патернів;

– генерація персоналізованих рекомендацій щодо часу та активності;

– легка візуалізація розкладу та статистики.

Створення такого вебзастосунку також має кілька переваг перед нативними мобільними або настільними додатками; він може бути доступний з різних пристроїв з доступом до Інтернету, установка не потрібна, збереження даних може бути централізованим, синхронізація даних може застосовуватися між різними пристроями користувача [4].

Отже, проєктування інтелектуальної вебсистеми управління часом є важливим завданням, яке відображає попит сучасного клієнта на високоякісний, доступний та «розумний» додаток для управління часом.

1.2 Аналіз цільової аудиторії та потреб користувачів

Створення якісної системи управління часом вимагає детального розуміння аудиторії, для якої вона призначена. Існує багато типів користувачів системи планування на основі демографічних характеристик, типів завдань, стилів роботи тощо. Деякі категорії потенційних користувачів інтелектуальної системи управління часом на основі аналізу ринку та запитів користувачів є:

– студенти та школярі – з чергуванням занять, такими як домашні завдання, підготовка до іспитів; чітко окреслюють свій розклад відповідно до розкладу занять та дедлайнів;

– фрілансери та віддалені працівники – планують весь свій робочий день самостійно, мають різні проєкти, пріоритети; ефективний розподіл завдань та клієнтів;

– менеджери та керівники – багато зустрічей, дедлайнів, паралельних завдань; аналітика продуктивності, інтелектуальне автоматичне планування;

– фахівці у творчих та інтелектуальних професіях – планування в години пікової продуктивності;

– звичайні користувачі – бажання організувати особисті справи, домашні завдання, хобі та цілі.

Згідно з аналізом поведінки аудиторії та оглядом відгуків користувачів популярних додатків для управління часом, основні вимоги користувачів визначені наступним чином:

– легке введення завдань з мінімальною кількістю кліків;

– візуальне відображення розкладу у вигляді графіка на часовій шкалі або тижневого календаря;

– підтримка повторюваних завдань (щоденні звички, щотижневі рутини);

– категоризація завдань з візуальним виділенням, як кольори або теги;

– система пріоритетів для ранжування завдань за важливістю;

– автоматичні пропозиції для вибору найкращого часу для виконання завдання;

– відображення продуктивності (графіки або карти активності);

– адаптивний інтерфейс для пристроїв, розроблений з урахуванням різних розмірів;

– світла та темна тема для зручної роботи в будь-який час доби;

– стабільна аутентифікація та безпека особистих даних [5].

Тому дослідження та аналіз цільової аудиторії визначають основні вимоги системи наступним чином: система може бути легко використана для повсякденного життя з розумними рекомендаціями, візуалізацією статистики та доступністю на пристроях [6].

1.3 Огляд та аналіз існуючих аналогів систем планування

Для того, щоб вирішити, чи є рішення достатньо хорошим, проводиться оцінка та аналіз існуючих аналогів систем планування в особистій системі управління часом для оцінки найкращого функціонування майбутньої системи. Провідні сервіси, з різними методологіями організації завдань, були розглянуті, щоб оцінити найкращі практики та характерні недоліки.

Таблиця 1.1 надає порівняльне дослідження доступних рішень.

Таблиця 1.1 – Порівняльний аналіз існуючих систем тайм-менеджменту

Система	Переваги	Недоліки
Worksection [7]	– український продукт з локалізованим інтерфейсом; – комбінація задач, проєктів і тайм-трекера в одному сервісі; – підтримка діаграми Ґанта для візуалізації термінів; – облік часу виконання задач; – вбудована комунікація та коментування; – зручна категоризація.	– орієнтований насамперед на команди та бізнес, а не на особисте використання; – складна крива навчання для одного користувача; – надлишковий функціонал для особистого тайм-менеджменту (звіти клієнтам, фінанси, ролі).
Google Calendar [8]	– безкоштовність; – надійна синхронізація; – зручне представлення часу; – інтеграція з екосистемою Google.	– відсутній інтелектуальний помічник; – немає системи порад; – обмежена статистика; – фокус лише на подіях, а не задачах.
TickTick [9]	– комбінація to-do та календаря; – вбудований таймер Pomodoro; – звички; – Eisenhower Matrix.	– більшість інтелектуальних функцій доступні лише в Premium; – інтерфейс перевантажений для нових користувачів.
Notion [10]	– гнучка кастомізація; – можливість створення власних систем; – сумісність з командною роботою.	– складна крива навчання; немає готового автопланера; – потребує самостійного налаштування.

Кінець таблиці 1.1

Система	Переваги	Недоліки
Any.do [11]	– мінімалістичний дизайн; – швидке внесення задач; функція «Мій день» для планування дня.	– відсутні розумні рекомендації; – обмежена аналітика; – функція плану дня працює статично, без аналізу звичок.

На рис. 1.1–1.5 зображено скриншоти розглянутих систем планування.

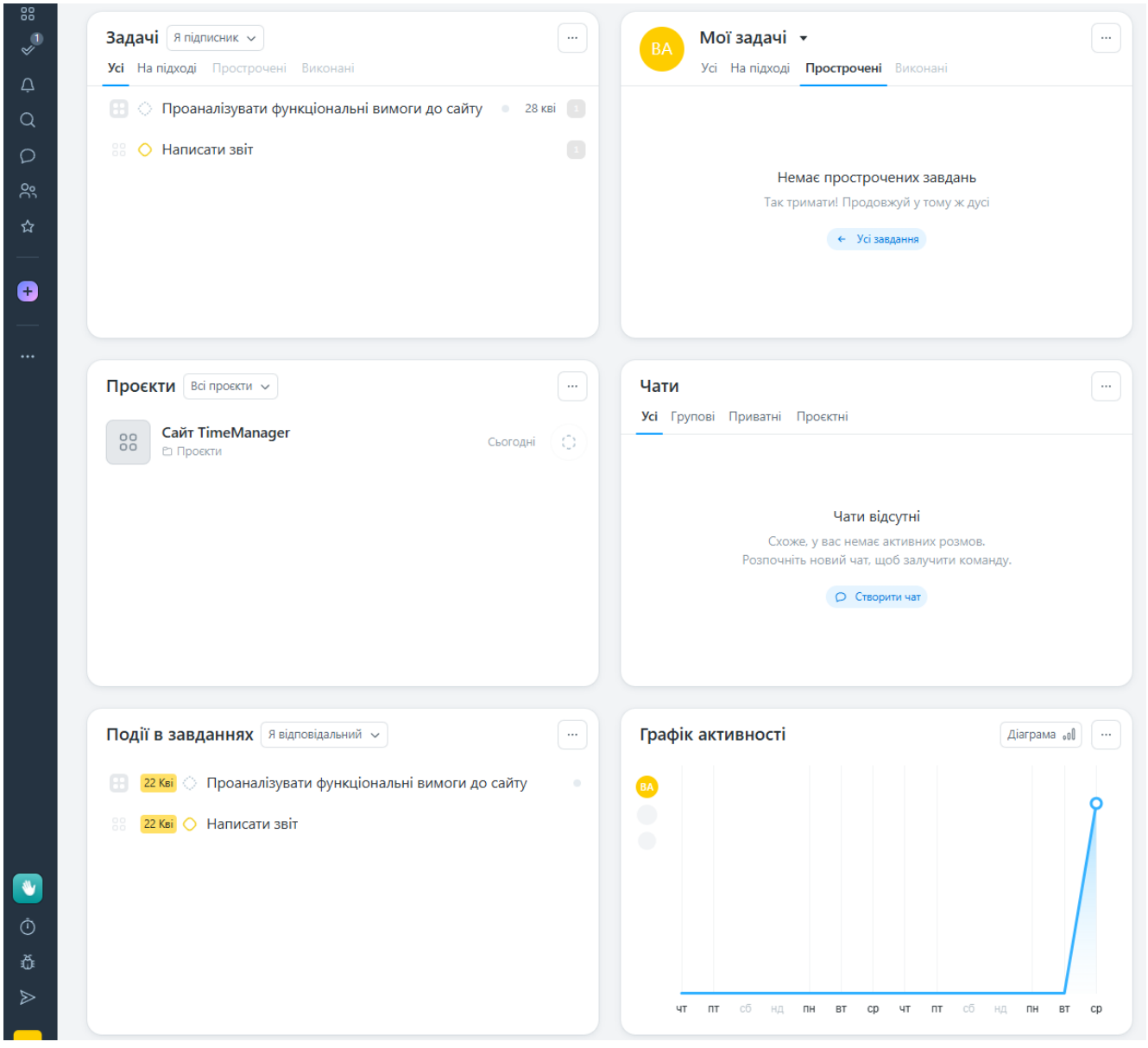

Рисунок 1.1 – Worksection [7]

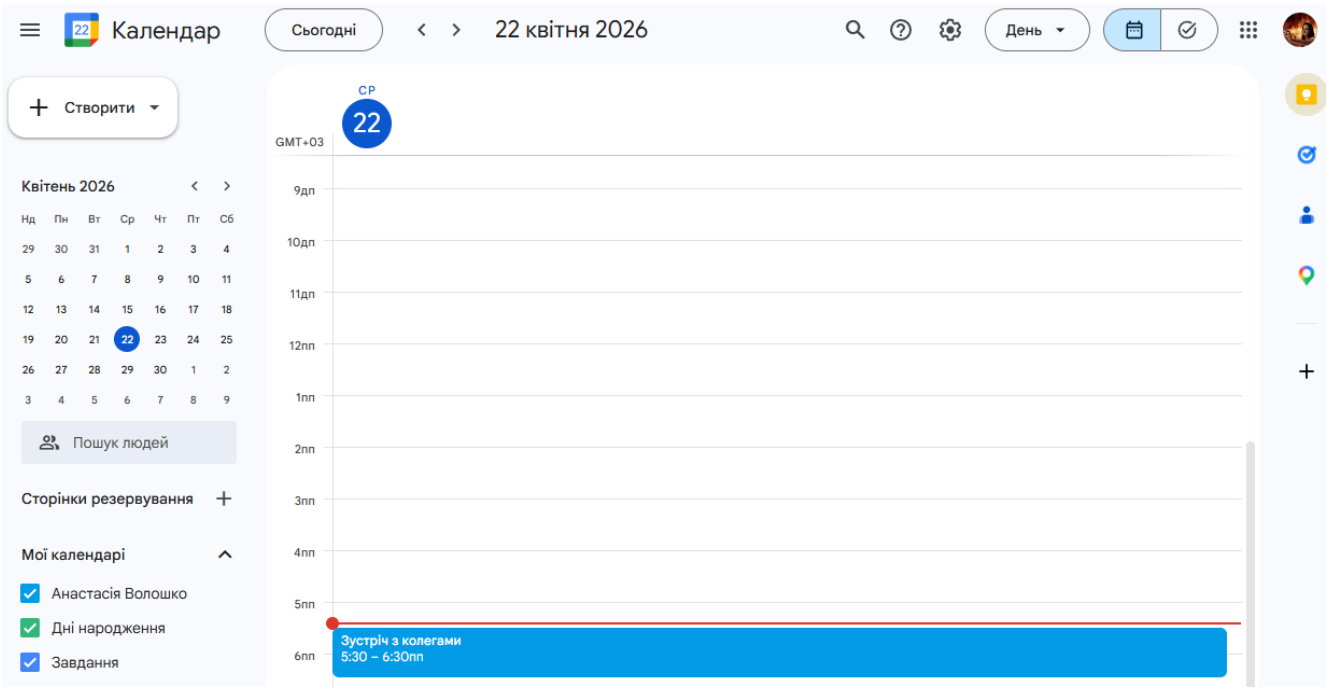

Рисунок 1.2 – Google Calendar [8]

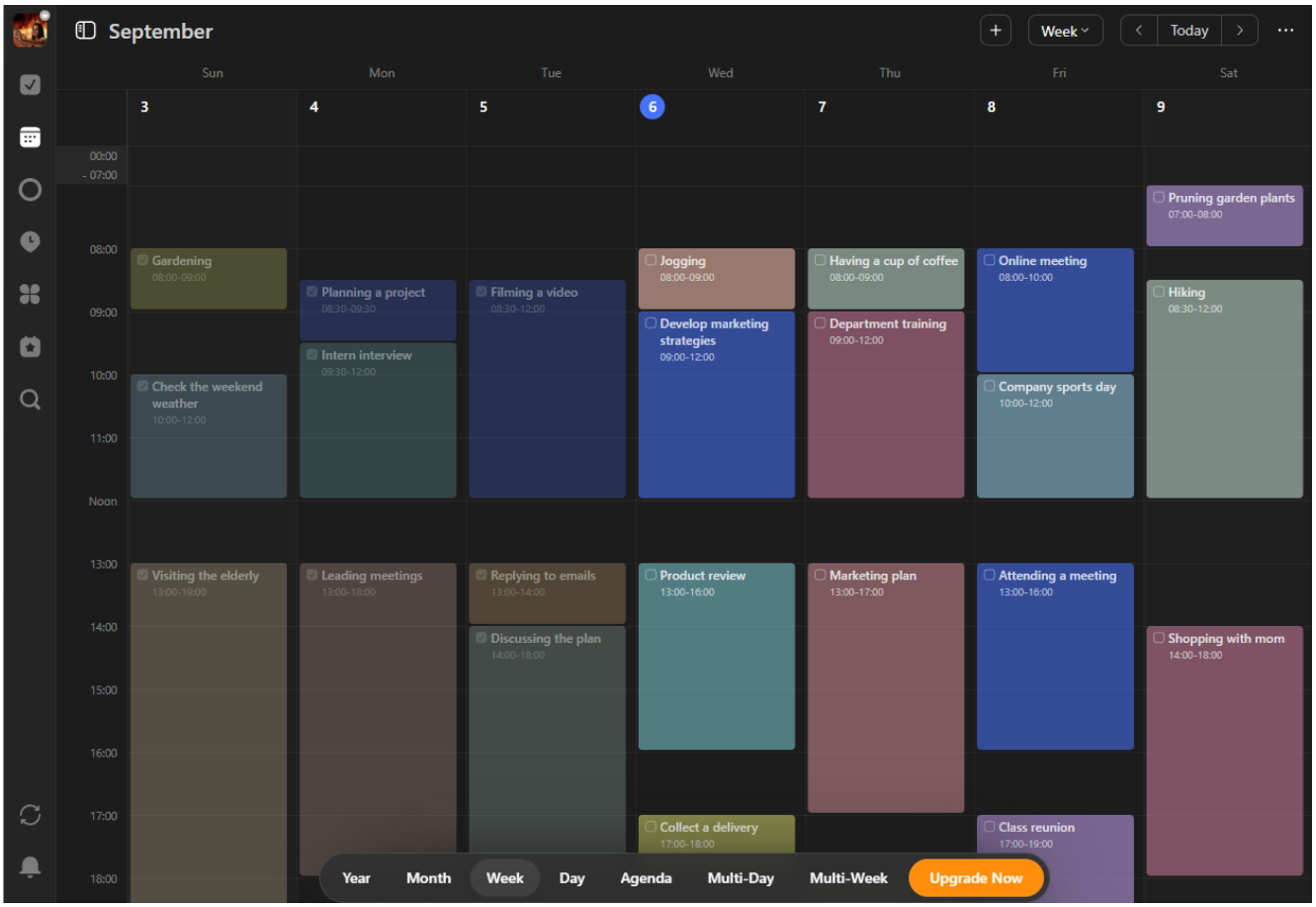

Рисунок 1.3 – TickTick [9]

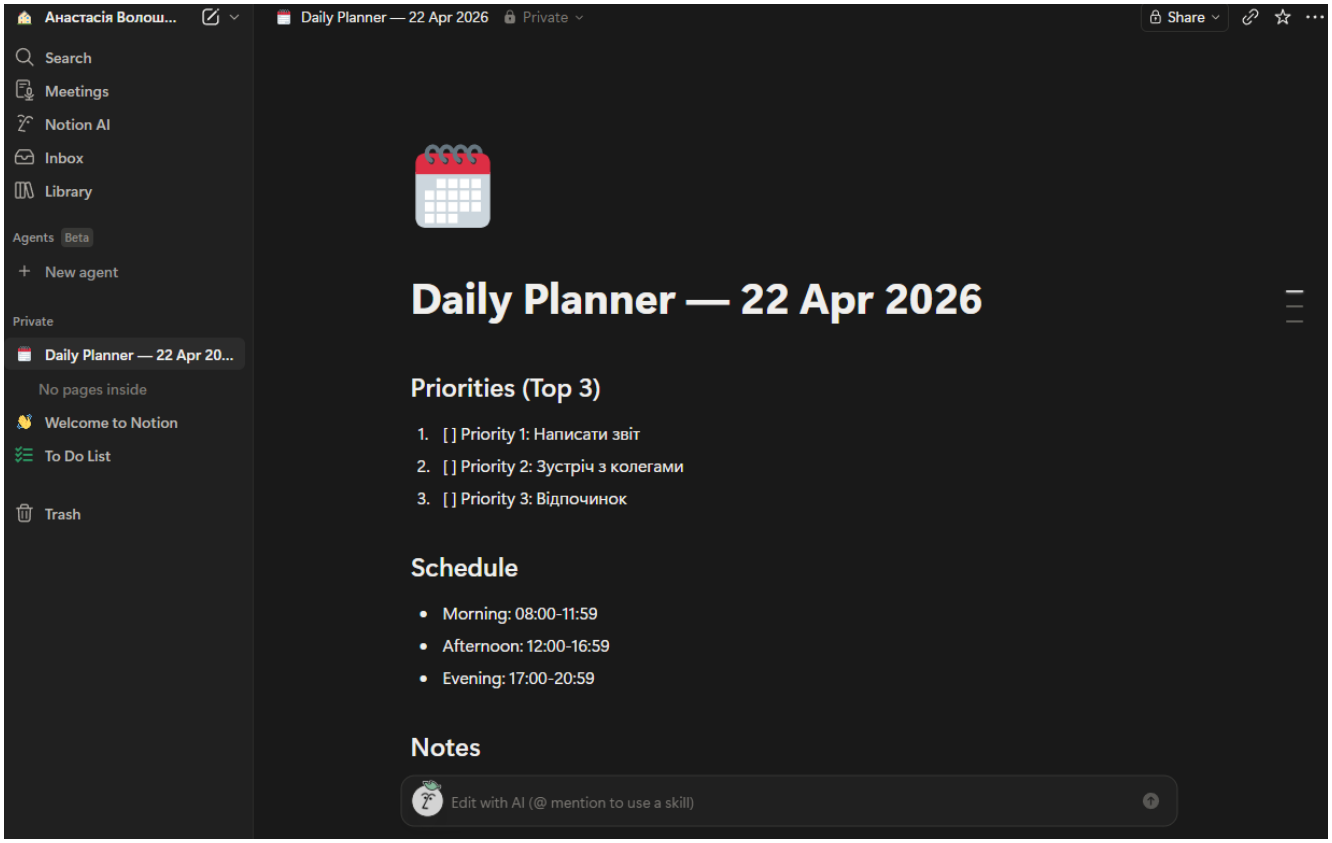

Рисунок 1.4 – Notion [10]

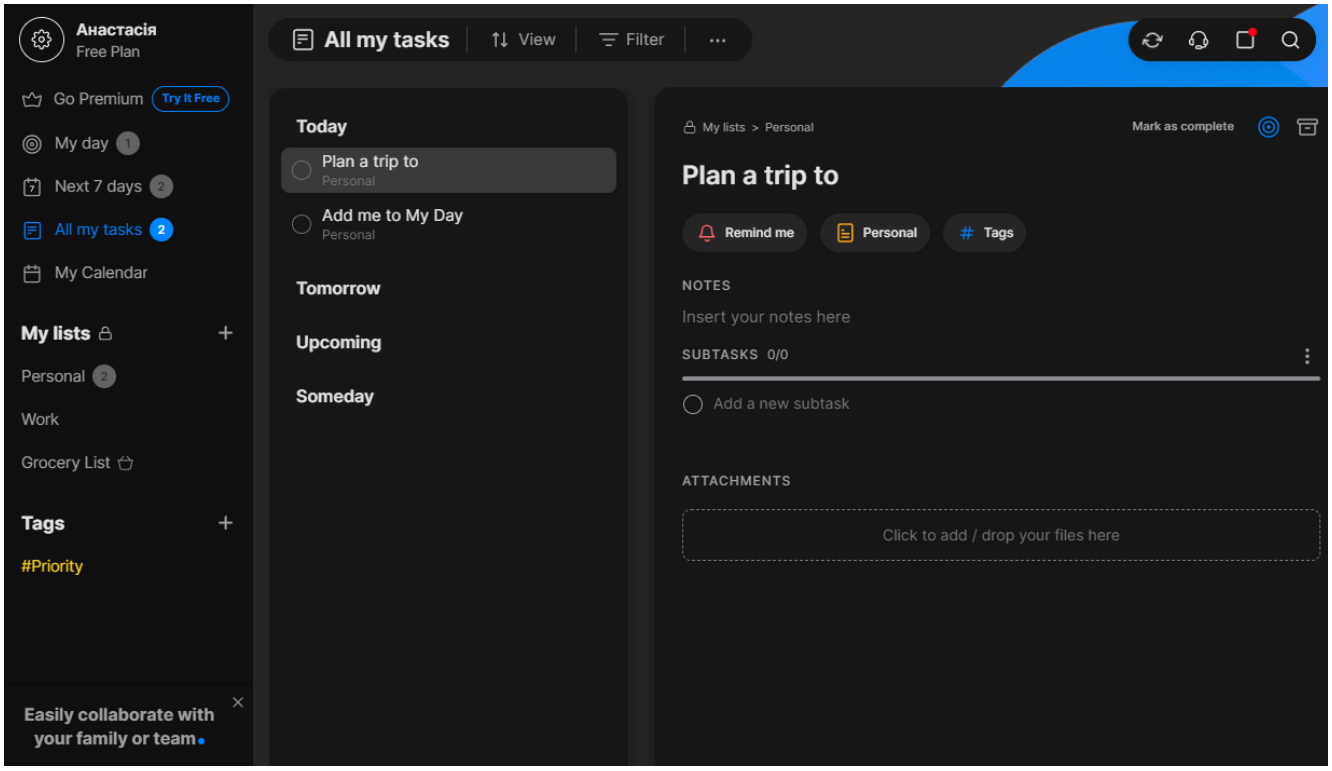

Рисунок 1.5 – Any.do [11]

За результатами аналізу аналогів виявлено наступні **типові недоліки** існуючих систем:

– немає інтелектуального автопланера, який може автоматично визначити, де є нові вільні слоти для майбутніх завдань;

– немає системи розумних порад на основі аналізу поведінки користувача;

– відсутність аналітики продуктивності – дуже обмежена аналітика продуктивності;

– розумні функції іноді доступні лише у платній версії;

– немає планування на основі пікових часів користувача, як зазначено вище.

Серед **позитивних тенденцій** у розробці подібних систем можна виділити:

– легкий для розуміння інтерфейс і найшвидший спосіб введення завдань;

– підтримка повторюваних завдань з можливістю вибору налаштувань;

– класифікація та кольори;

– налаштування для різних пристроїв;

– також підтримується темний колір для комфортної роботи.

На основі цього аналізу можемо вивести основну цінність розробленої системи як інтеграцію традиційного управління часом (розклад, завдання, категорії) та інтелектуальних функцій (автопланер, розумні поради, аналіз продуктивності) в одне безкоштовне вебрішення з гарним адаптивним інтерфейсом.

1.4 Постановка задачі

Актуальність. На основі аналізу цільової аудиторії та існуючих аналогів, актуальним завданням буде створення інтелектуальної вебсистеми для особистого управління часом шляхом інтеграції традиційних функцій планування з інтелектуальними алгоритмами для автоматизованого розміщення завдань та генерації персоналізованих порад.

Мета роботи – створити вебдодаток та покращити доступ користувача до планування часу за допомогою автоматичного пошуку вільних слотів, розумного радника, аналітики продуктивності зображень. Робота стосується особистого планування часу та інтелектуального аналізу розкладу користувача.

Об'єкт роботи – процеси персонального планування часу та інтелектуального аналізу розкладу користувача.

Предмет роботи – методи інтелектуального аналізу розкладу (статистичний аналіз, виявлення продуктивних годин, аналіз навантаження, пошук вільних слотів, правилова система генерації порад), алгоритм автопланування задач на основі оцінки часових слотів за функцією score(hour, preferMorning), а також технології реалізації вебзастосунку: PHP 8.4 / Laravel 10 на серверній стороні, MySQL 8.4 як сховище даних та JavaScript (ES6+) з Chart.js на клієнтській стороні.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

– дослідити існуючі системи управління часом та функціональні недоліки;

– розробити інтелектуальний алгоритм автопланування завдань з урахуванням існуючих уподобань користувача та розкладу;

– створити систему на основі правил для генерації індивідуальних рекомендацій;

– побудувати дизайн системи для масштабування та забезпечення безпеки;

– створити вебдодаток з адаптивним інтерфейсом та підтримкою тем;

– провести тестування системи.

Функціональні вимоги до системи подано в табл. 1.2.

Таблиця 1.2 – Функціональні вимоги до системи

Модуль	Функція
Автентифікація	– реєстрація нових користувачів; – вхід у систему/вихід; – захист сесії;

Кінець таблиці 1.2

Модуль	Функція
	– редагування профілю, зміна паролю, завантаження фото профілю.
Розклад	– створення, редагування, видалення задач; – встановлення дати, часу початку та кінця; – призначення категорії та пріоритету; – задання кольору; – зміна статусу (очікує / виконано / скасовано); – перегляд розкладу на день та тиждень.
Повторювані задачі	– створення шаблонів повторюваних задач (щодня, по днях тижня, по днях місяця); – ввімкнення/вимкнення повторення; – відмітка виконання на конкретну дату.
Категорії	– створення, редагування, видалення категорій; – призначення кольору кожній категорії; – облік кількості задач у категорії.
Аналітика	– відображення статистики по періодах (тиждень/місяць/рік); – графіки задач по днях, днях тижня, годинах; – розподіл за пріоритетами та категоріями; – відстеження серії продуктивних днів.
Інтелектуальний радник	– генерація персоналізованих порад на основі аналізу поведінкових патернів користувача; – виявлення перевантажених днів, прострочених задач, пікових годин продуктивності.
Автопланер	– автоматичне знаходження вільних часових слотів для нової задачі на основі тривалості, діапазону дат, робочого часу та переваг користувача.
Налаштування	– перемикання між світлою та темною темою зі збереженням вибору в акаунті.

Наведені функціональні вимоги стосуються восьми модулів системи та представляють основні потреби користувачів системи з планування, а також інтелектуальні можливості, які відрізнятимуть розроблений додаток від існуючого аналога. Інтеграція саме цього функціонального набору забезпечить кінцевий додаток для управління вашим особистим часом, автоматизуючи рутинні планувальні дії.

Нефункціональні вимоги до системи подано в табл. 1.3.

Таблиця 1.3 – Нефункціональні вимоги до системи

Модуль	Функція
Продуктивність	– час завантаження сторінки – до 2 секунд при стабільному з'єднанні.
Безпека	– захист від SQL-ін'єкцій, XSS (Cross-Site Scripting), CSRF-атак; – хешування паролів із використанням bcrypt; – розмежування даних між користувачами.
Зручність	– інтуїтивно зрозумілий інтерфейс без потреби окремих інструкцій.
Адаптивність	– повноцінна робота на пристроях з діагоналлю від 4,7 до 27 дюймів.
Надійність	– коректна робота при одночасному використанні кількома користувачами; цілісність даних при виконанні транзакцій.
Підтримуваність	– модульна архітектура за патерном MVC; чітке розмежування логіки; – коментовані ділянки коду.

Забезпечення дотримання зазначених нефункціональних вимог є важливим для того, щоб система була надійною, якісною та зручною у використанні в реальних умовах. Ця комплексна специфікація була підготовлена шляхом роботи

над функціональними та нефункціональними вимогами, і це буде основою для подальшого архітектурного проєктування та розробки системи.

Висновки до розділу 1

У першій частині було вивчено змістову область управління особистим часом та заплановано робочий стан.

Було обґрунтовано, наскільки інтелектуальна вебсистема є актуальною для управління особистим часом; багато сучасних інструментів мають або функціональність календаря, або список завдань, що означає, що вони не надають розумніших пропозицій або автоматизації планування.

Також було проведено аналіз цільової аудиторії, який виявив п'ять основних типів користувачів (студенти, фрілансери, менеджери, творчі спеціалісти, звичайні користувачі) та їхні потреби в системі планування.

Було оцінено п'ять основних аналогів (Worksection, Google Calendar, TickTick, Notion, Any.do), і виявилися ті ж обмеження – відсутність автопланера, недостатня аналітична здатність, інтелектуальні функції у платних версіях. Цей процес можна використовувати для визначення основної цінності кінцевої системи: інтеграція інтелектуального автопланування, розумних порад та візуальної аналітики в одному рішенні.

Було розроблено завдання, об'єкт та предмет роботи. Встановлено функціональні вимоги до системи, які вимагають 8 модулів, та нефункціональні вимоги до продуктивності, безпеки, зручності, адаптивності, надійності, підтримуваності, які вплинуть на проєктування та реалізацію в майбутньому.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

У цьому розділі розглянуто теоретичну основу системи, що розробляється, яка складається з різних способів розуміння розкладу користувача та розробки алгоритмів для автоматичного планування завдань, математичної моделі оцінки часових слотів та логіки вибору технологічного стеку, який буде основою для реалізації вебдодатку.

2.1 Методи інтелектуального аналізу розкладу

Одним з ключових компонентів системи є обробка розкладу користувача інтелектуальним способом. Це дозволяє зберігати завдання на диску, а також активно враховувати їх у планувальному аналізі шляхом аналізу поведінкових патернів та генерації рекомендацій [12].

У розроблюваній системі застосовано комплекс методів, кожен з яких вирішує певну підзадачу.

2.1.1 Статистичний аналіз розкладу

Аналіз зводиться до агрегації даних про задачі користувача в різні часові зрізи – день, тиждень, місяць, рік. Підраховуються такі показники:

- загальна кількість задач за період;
- кількість виконаних, прострочених, скасованих задач;
- відсоток виконання;
- розподіл задач по годинах доби;
- розподіл задач по днях тижня;
- розподіл задач по категоріях та пріоритетах.

Результати агрегації візуалізуються у вигляді стовпчикових діаграм, кільцевих діаграм та лінійних графіків, що дозволяє користувачеві візуально оцінити свою продуктивність.

Такий підхід перетворює сухі цифри з бази даних у наочну інформацію, що дозволяє користувачеві виявити власні поведінкові патерни – найбільш завантажені дні тижня, недоопрацьовані категорії задач або періоди зі зниженою продуктивністю. На основі цих спостережень користувач може коригувати власний розклад та планувати майбутні справи з урахуванням реальних звичок.

2.1.2 Метод виявлення продуктивних годин (peak hours detection)

В основі цього методу лежить збір кількості завдань для кожної години дня за певний період (останні 30 днів). Година з найбільшою кількістю виконань вважається годиною найвищої продуктивності користувача. На основі цієї інформації ми застосовуємо два модулі:

– у модулі радника – для підказки користувачеві про його найпродуктивніший час;

– у модулі автопланера – для оцінки часових слотів за продуктивністю.

2.1.3 Аналіз навантаження на день (day load analysis)

Запропонована техніка визначає, чи є певний день користувача перевантаженим. Поріг визначається наступним чином:

– кількість завдань кожного дня (поріг ≥ 10 вважається перевантаженням);

– загальна тривалість запланованих завдань (включаючи повторювані);

– кількість завдань високого пріоритету.

Якщо день вважається зайнятим, надається відповідна порада з рекомендацією перемістити певні завдання або перевірити їх пріоритет.

2.1.4 Метод пошуку вільних слотів (free-slot finding)

Для виконання завдання вводяться тривалість завдання, діапазон дат та робочі години, які повертають список доступних часових інтервалів. Деталі реалізації розглядаються у п. 2.2.

Цей метод є ядром автоматизованого модуля планувальника і використовується кожного разу, коли користувач звертається до системи за допомогою для встановлення нового завдання. У той час як класичний календар, де один вручну шукає вільне «вікно» зверху в розкладі, система робить все автоматично, враховуючи доступні зобов'язання.

2.1.5 Система радника на основі правил (rule-based advisor)

Система використовує модель продукційних знань, яка є набором правил у форматі «ЯКЩО умова – ТО порада». Розклади оцінюються за правилами для генерації нових рекомендацій. Приклади правил:

– «ЯКЩО кількість прострочених завдань > 0 ТО показати пораду про необхідність їх перегляду»;

– «ЯКЩО заплановано більше 10 завдань на сьогодні ТО надати попередження про перевантаження»;

– «ЯКЩО відсоток виконання за тиждень ≥ 80% ТО надати похвальну пораду»;

– «ЯКЩО відсоток виконання за тиждень < 40% ТО запропонувати планувати менше завдань».

Нарешті, до системи впроваджено десять правил, які включають такі категорії поведінки користувача, як прострочені завдання, перевантаження дня, порожні дні, тижнева продуктивність, пікові години, дотримання повторюваних завдань, нічні завдання, завдання високого пріоритету, відсутність категоризації, загальна оцінка стану розкладу.

Комбінація цих підходів є повною інтелектуальною системою, яка надає користувачу більше, ніж просто запис даних, а також допомагає користувачу ефективно управляти своїм часом.

2.2 Алгоритм автопланування задач

Автопланер є ключовим інтелектуальним компонентом системи. Його задача – знайти оптимальний час для нової задачі в існуючому розкладі користувача. На вхід алгоритм отримує набір параметрів [13]:

- назва задачі, опис, пріоритет, категорія;
- тривалість задачі (у хвилинах);
- діапазон дат, у якому здійснюється пошук;
- робочий час (година початку та закінчення пошуку);
- прапорець переваги ранкових годин.

На виході алгоритм повертає відранжований список часових слотів (максимум 8), кожен з яких містить дату, час початку, час закінчення та оцінку пріоритетності.

Загальна схема алгоритму:

1) для кожного дня з діапазону отримати список зайнятих часових інтервалів (окремі задачі + повторювані задачі, що припадають на цей день);

2) пройти по робочому часу дня з кроком 15 хвилин, перевіряючи можливість розміщення задачі заданої тривалості;

3) для кожного потенційного слоту перевірити, чи не конфліктує він із зайнятими інтервалами;

4) для вільних слотів обчислити оцінку (score) на основі часу доби та преференцій користувача;

5) відсортувати всі знайдені слоти за спаданням оцінки та повернути топ-8.

Псевдокод алгоритму наведено нижче:

```
function findFreeSlots(userId, dateFrom, dateTo, hFrom, hTo, durationMin,
                       preferMorning):
    slots = []
    for date in range(dateFrom, dateTo):
        busy = getBusyRanges(userId, date)
```

```
t = date + hFrom:00
while t + durationMin <= date + hTo:00:
if not conflictsWithBusy(t, t+durationMin, busy):
score = scoreSlot(t.hour, preferMorning)
slots.append({date, start:t, end:t+durationMin, score})
t = t + 15 minutes
return sortByScoreDesc(slots)[:8]
```

Функція getBusyRanges надає список зайнятих інтервалів з двох джерел: одноразових завдань у таблиці завдань та повторюваних завдань у таблиці recurring_tasks, які припадають на вказану дату. Функція occursOn використовує параметри повторення (щоденно, за днями тижня, за днями місяця), щоб визначити, чи виконується завдання в цей день.

Функція conflictsWithBusy перевіряє, чи новий слот перекривається з будь-яким із зайнятих інтервалів. Два інтервали [s1, e1] та [s2, e2] перекриваються, тоді і тільки тоді, якщо s1 < e2 та e1 > s2.

Функцію scoreSlot використовуємо для створення оцінки слота, яка залежить від часу початку та уподобань користувача (описано в 2.3).

Складність алгоритму. Нехай n – кількість днів у діапазоні, h – кількість робочих годин на день, b – середня кількість зайнятих інтервалів на день. Отже, загальна обчислювальна складність O (n × h × 4 × b), оскільки кожен день має h × 4 слоти (кроки по 15 хвилин) і для кожного з них потрібно перевірити b зайнятих інтервалів. І для типових значень (n = 7, h = 12, b = 10) це близько 3360 операцій, які виконуються за мілісекунди.

2.3 Математична модель оцінки часових слотів

Функція scoreSlot(hour, preferMorning) обчислює пріоритетність часового слоту на основі двох параметрів: години доби та налаштування користувача щодо переваги ранкових годин. Модель спирається на дослідження ритмів

продуктивності людини, зокрема на концепцію «ультрадіанних циклів» та досліджень денної продуктивності [14].

Базова оцінка для всіх слотів – 50 балів. До неї додається (або віднімається) модифікатор, що залежить від години доби. У разі увімкнення користувачем переваги ранкових годин – модифікатори для ранкових годин збільшуються, а для вечірніх – зменшуються.

Формальна функція оцінки наведена нижче:

$$score(hour, preferMorning) = 50 + modifier(hour, preferMorning)$$

Значення модифікатора для різних годин наведено в табл. 2.1.

Таблиця 2.1 – Оцінка часових слотів залежно від години та преференцій

Година початку	**Режим «без переваги»**	**Режим «ранкова перевага»**
0:00–7:59	50 – 5 = 45	50 – 10 = 40
8:00–8:59	50 + 10 = 60	50 + 30 = 80
9:00–9:59	50 + 10 = 60	50 + 30 = 80
10:00–11:59	50 + 25 = 75 (пік)	50 + 30 = 80 (пік)
12:00–14:59	50 + 10 = 60	50 + 10 = 60
15:00–17:59	50 + 20 = 70	50 + 5 = 55
18:00–20:59	50 + 5 = 55	50 – 10 = 40
21:00–23:59	50 – 5 = 45	50 – 10 = 40

Обґрунтування вибору значень:

– години 10:00–12:00 та 15:00–17:00 вважаються піковими за замовчуванням, оскільки більшість людей є продуктивними в цей час;

– ранкові години (до 8:00) та пізні нічні (після 21:00) отримують негативний модифікатор, оскільки завдання, заплановані на цей час, частіше відкладаються;

– коли увімкнено перевагу ранкових годин, зростає акцент на 8:00–11:00 – підходить для користувачів, які є більш продуктивними вранці;

– обідній час (12:00–14:00) отримує нейтральну оцінку, оскільки продуктивність у цей період є досить змінною.

При розрахунку оцінки, всі існуючі слоти сортуються в порядку спадання. Спочатку слоти з найвищими оцінками будуть розміщені першими у списку рекомендацій, відповідно до їх найбільш продуктивних годин. Легко вибрати будь-який з рекомендованих слотів або відхилити все і вручну ввести завдання.

Цю модель легко застосувати, і водночас вона надає змістовні рекомендації на основі загальних моделей продуктивності. У подальшому розвитку модель може бути покращена шляхом врахування специфічної статистики користувача (адаптивна система), а для базових можливостей підхід є оптимальним.

2.4 Огляд та обґрунтування вибору технологій розробки

Для реалізації вебсистеми необхідно обрати оптимальний стек технологій, що забезпечить виконання всіх вимог та дозволить ефективно підтримувати й розвивати застосунок у майбутньому.

Аналіз наявних вебтехнологій. Сучасна веброзробка має великий вибір технологій, які можна розділити на кілька категорій:

1) фронтенд-технології (клієнтська частина):

– HTML – основа будь-якої вебсторінки, визначає структуру документа;

– CSS – стилізація вебсторінок, з підтримкою анімацій, змінних та адаптивного дизайну;

– JavaScript – додавання інтерактивності та динамічної взаємодії;

– фреймворки/бібліотеки: React, Vue, Angular для SPA-застосунків; Bootstrap для швидкого адаптивного дизайну;

– бібліотеки візуалізації: Chart.js, D3.js, Highcharts для побудови графіків і діаграм;

2) бекенд-технології (серверна частина):

– PHP з фреймворком Laravel – зріла популярна екосистема для веброзробки;

– Node.js з Express.js – JavaScript на сервері, асинхронна обробка запитів;

– Python з Django/Flask – швидка розробка з потужним ORM та адмін-панеллю;

– Ruby on Rails – висока продуктивність розробки з філософією convention over configuration;

3) бази даних:

– MySQL – найпопулярніша реляційна СУБД (система управління базами даних) з відкритим кодом;

– PostgreSQL – потужна альтернатива з підтримкою складних запитів та типів даних;

– MongoDB – документоорієнтована NoSQL база для неструктурованих даних;

– SQLite – легка файлова БД для невеликих проєктів.

Вибір технологій для розробки системи. Для реалізації інтелектуальної системи з особистого тайм-менеджменту обрано такі технології:

1) фронтенд:

– HTML5 – семантична розмітка сторінок;

– CSS3 – стилізація з використанням CSS-змінних для підтримки тем;

– JavaScript (ES6+) – динаміка інтерфейсу, AJAX-запити, робота з модальними вікнами;

– Bootstrap 5 – сітка та базові компоненти для адаптивного дизайну;

– Bootstrap Icons – набір іконок, що узгоджується з Bootstrap;

– Chart.js – побудова діаграм і графіків для модуля аналітики;

2) бекенд:

– PHP 8.4 – мова серверної частини;

– Laravel 10 – MVC-фреймворк з Eloquent ORM, системою міграцій, Blade-шаблонізатором, вбудованою автентифікацією, захистом від CSRF та XSS;

– Composer – пакетний менеджер PHP;

3) база даних:

– MySQL 8.4 – реляційна СУБД;

– phpMyAdmin – вебінтерфейс для адміністрування БД;

4) середовище розробки:

– OpenServer – локальний сервер для Windows з Apache, PHP, MySQL;

– Visual Studio Code – редактор коду.

Обґрунтування вибору:

– Laravel 10 забезпечує готові рішення для всіх необхідних функцій – автентифікація, валідація, міграції, Blade-шаблони, ORM, захист від CSRF, сесії, хешування паролів, що значно прискорює розробку;

– Eloquent ORM дозволяє працювати з БД на рівні об'єктів, не пишучи SQL-запити вручну, що знижує ризик помилок та SQL-ін'єкцій;

– MySQL є перевіреним рішенням для реляційних даних з чіткою структурою, які переважають у системі (задачі, категорії, користувачі);

– Chart.js має легкий API, дає змогу швидко створювати різні типи діаграм без додаткової конфігурації;

– Bootstrap 5 пришвидшує розробку адаптивного інтерфейсу завдяки готовій системі сітки та компонентів;

– CSS-змінні (custom properties) дозволяють реалізувати систему тем без перебудови всього CSS – потрібно лише змінити значення змінних у корені документа;

– обраний стек не потребує додаткових комерційних ліцензій – усі технології безкоштовні й з відкритим кодом.

Такий вибір технологій забезпечує оптимальне співвідношення швидкості розробки, продуктивності, безпеки та підтримуваності системи [15, 16].

Висновки до розділу 2

У другому розділі виконано дослідження методів та технологій, що будуть використані для вирішення поставленої задачі.

Описано п'ять методів інтелектуального аналізу розкладу: статистичний аналіз, виявлення продуктивних годин, аналіз навантаження на день, пошук вільних слотів та правилова система генерації порад. Кожен метод має чітко окреслену задачу та результат.

Розроблено алгоритм автопланування задач, який на вхід приймає тривалість задачі, діапазон дат та преференції користувача, а на виході повертає відранжований список вільних слотів. Проаналізовано обчислювальну складність алгоритму.

Побудовано математичну модель оцінки часових слотів у вигляді функції scoreSlot(hour, preferMorning), яка базується на концепції пікових годин продуктивності. Наведено таблицю модифікаторів для різних годин доби з обґрунтуванням вибору значень.

Проведено огляд технологій веброзробки та обґрунтовано вибір технологічного стеку: PHP 8.4 + Laravel 10, MySQL 8.4, Bootstrap 5, Chart.js, JavaScript ES6+. Вибір зумовлений зрілістю технологій, наявністю готових рішень для необхідних задач та відсутністю комерційних ліцензій.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У цьому розділі подано процес проєктування інтелектуальної системи тайм-менеджменту «TimeManager»: сценарії використання, архітектуру, структуру бази даних, моделювання окремих модулів та аналіз отриманих результатів.

3.1 Сценарії використання системи

Для коректного проєктування інтерфейсів і логіки взаємодії розроблено основні сценарії використання системи «TimeManager». Сценарії сформульовано на основі вимог, визначених у розділі 1, та охоплюють ключові дії користувача в системі (табл. 3.1–3.7).

Таблиця 3.1 – Сценарій 1: реєстрація та вхід у систему

Актор	Новий користувач
Передумови	Користувач відвідав стартову сторінку системи
Основний потік	1. Користувач обирає «Створити акаунт». 2. Система відображає форму реєстрації. 3. Користувач вводить ім'я, email, пароль та підтвердження паролю. 4. Система валідує введені дані. 5. Система створює запис у таблиці users, хешує пароль та створює п’ять стандартних категорій для нового користувача. 6. Система автоматично авторизує користувача та перенаправляє на дашборд.
Альтернативний потік	1. Якщо email вже існує, система показує повідомлення про помилку. 2. Якщо паролі не збігаються, система показує відповідну помилку.
Постумови	Користувач авторизований у системі та має акаунт з базовими категоріями.

Таблиця 3.2 – Сценарій 2: додавання нової задачі в розклад

Актор	Авторизований користувач
Передумови	Користувач перебуває на сторінці розкладу
Основний потік	1. Користувач натискає кнопку «Додати» у верхній панелі або клацає на вільну годину. 2. Відкривається модальне вікно з формою. 3. Користувач вводить назву задачі, опис, обирає дату, час початку та закінчення, пріоритет, категорію, колір. 4. Користувач натискає «Зберегти». 5. Система валідує дані та створює запис у таблиці tasks. 6. Розклад оновлюється, і нова задача з'являється у відповідному часовому слоті.
Альтернативний потік	1. Користувач може використати автопланер, відкривши сторінку «Поради»: заповнити параметри задачі та тривалість, отримати пропозиції слотів і прийняти будь-яку з них.
Постумови	Задача додана до розкладу та видна користувачеві.

Таблиця 3.3 – Сценарій 3: перегляд аналітики та порад

Актор	Авторизований користувач
Передумови	Користувач має історію використання системи
Основний потік	1. Користувач переходить у розділ «Аналітика». 2. Обирає період (тиждень/місяць/рік). 3. Система агрегує дані та відображає: 4 статистичних блоки (всього, виконано, серія, повторювані), графік задач по днях, кільцеву діаграму пріоритетів, графік активності по годинах, графік по днях тижня, розподіл по категоріях. 4. Користувач переходить у розділ «Поради», де бачить список персональних порад від системи.
Альтернативний потік	1. Користувач може змінити період – дані перерахуються.
Постумови	Користувач отримав візуальне уявлення про свою продуктивність та рекомендації щодо покращення.

Таблиця 3.4 – Сценарій 4: використання автопланера

Актор	Авторизований користувач
Передумови	Користувач має заплановані задачі в розкладі
Основний потік	1. Користувач відкриває розділ «Поради». 2. Заповнює форму автопланера: назва задачі, тривалість, пріоритет, категорія, діапазон дат, робочий час, прапорець «Ранкова перевага». 3. Натискає «Знайти вільний час». 4. Система виконує алгоритм findFreeSlots та повертає до 8 запропонованих слотів. 5. Користувач вибирає один зі слотів клацанням. 6. Система підтверджує вибір та створює задачу в розкладі.
Альтернативний потік	1. Якщо вільних слотів не знайдено, система виводить відповідне повідомлення з пропозицією розширити діапазон.
Постумови	Задача додана в розклад на автоматично підібраний оптимальний час.

Таблиця 3.5 – Сценарій 5: позначення задачі як виконаної

Актор	Авторизований користувач
Передумови	Користувач має хоча б одну заплановану задачу зі статусом «Очікує»
Основний потік	1. Користувач переходить на дашборд або сторінку розкладу. 2. Наводить курсор на задачу та натискає кнопку «Виконано». 3. Система оновлює статус задачі на «completed» у таблиці tasks. 4. Задача візуально змінюється: назва перекреслюється, прозорість зменшується. 5. Статистика прогресу дня автоматично оновлюється.
Альтернативний потік	1. Для повторюваних задач позначка виконання зберігається не в полі status, а додається запис у таблицю recurring_done_log з поточною датою.
Постумови	Задача позначена як виконана, прогрес користувача оновлено.

Таблиця 3.6 – Сценарій 6: робота з повторюваними задачами

Актор	Авторизований користувач
Передумови	Користувач має регулярні задачі (звички, рутини)
Основний потік	1. Користувач переходить у розділ «Повторювані задачі». 2. Натискає «Додати шаблон». 3. Вводить назву, опис, час початку, пріоритет, категорію. 4. Обирає тип повторення: щодня, по днях тижня (обирає конкретні дні), по днях місяця (1–31). 5. Зберігає шаблон. 6. Система створює запис у recurring_tasks. 7. Шаблон автоматично з'являється в розкладі на всі підходящі дати.
Альтернативний потік	1. Користувач може тимчасово вимкнути шаблон через toggle; він не з'являтиметься в розкладі, але не видалятиметься. 2. Користувач може позначити повторювану задачу виконаною на конкретну дату – запис додається до recurring_done_log.
Постумови	Повторювана задача налаштована та автоматично відображається у розкладі.

Таблиця 3.7 – Сценарій 7: персоналізація та налаштування профілю

Актор	Авторизований користувач
Передумови	Користувач має активний обліковий запис у системі
Основний потік	1. Користувач переходить у розділ «Мій профіль» через бічне меню. 2. Система відображає сторінку з поточними даними користувача: ім'я, email, дата реєстрації, фото профілю. 3. Користувач клацає на область аватара та обирає нове зображення з пристрою. 4. Система валідує файл (формат, розмір до 10 МБ), зберігає його в public/uploads/avatars/ та оновлює поле avatar у таблиці users. 5. Користувач редагує ім'я та email у відповідних полях форми. 6. Користувач натискає «Зберегти зміни». 7. Система оновлює запис у таблиці users та показує

Кінець таблиці 3.7

	повідомлення про успіх. 8. Користувач переходить у розділ «Налаштування». 9. Обирає темну тему оформлення через прев'ю-картки. 10. Система миттєво застосовує обрану тему (через зміну атрибута data-theme), зберігає її у localStorage та надсилає AJAX-запит до /settings/theme для оновлення поля theme в БД.
Альтернативний потік	1. Користувач може змінити пароль, заповнивши окрему форму (поточний пароль + новий + підтвердження). Система перевіряє поточний пароль, хешує новий через bcrypt та оновлює запис. 2. Користувач може видалити поточне фото профілю – система очищає поле avatar та видаляє файл з диску. 3. Для швидкого перемикання теми користувач може використовувати іконку сонця/місяця у верхній панелі без переходу в налаштування.
Постумови	Профіль користувача оновлений, тема оформлення застосована та синхронізована між пристроями через збереження в БД.

Розроблені сценарії використання охоплюють усі ключові функції системи та будуть основою для реалізації відповідних сторінок і компонентів.

3.2 Проєктування архітектури системи за патерном MVC

Для побудови вебзастосунку обрано архітектурний патерн MVC (Model-View-Controller), що забезпечує чіткий розподіл відповідальності між компонентами та спрощує подальшу підтримку й розширення системи [17].

На рис. 3.1 представлено схему архітектури системи.

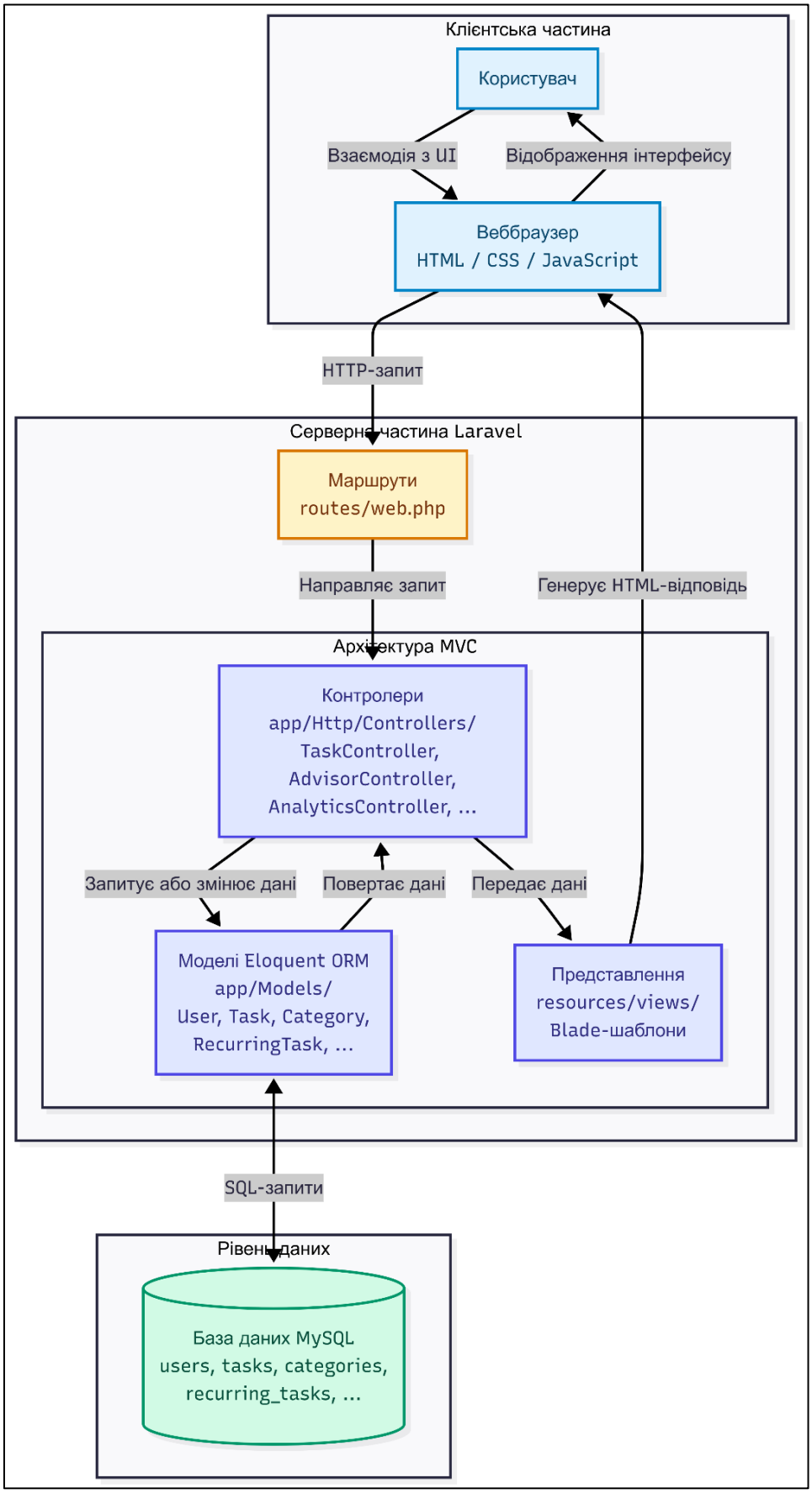

Рисунок 3.1 – Загальна схема архітектури системи

Потік даних та взаємодія компонентів:

– користувач взаємодіє з системою через веббраузер, надсилаючи HTTP-запити (HyperText Transfer Protocol);

– система маршрутизації Laravel (routes/web.php) аналізує URL-адресу й передає запит відповідному контролеру;

– контролер обробляє запит, взаємодіючи з моделями для отримання або зміни даних;

– моделі Eloquent виконують операції з базою даних, повертаючи результати контролеру;

– контролер передає дані у Blade-шаблон, який генерує HTML-відповідь;

– клієнтська частина отримує HTML, CSS та JavaScript і відображає інтерфейс.

Основні компоненти архітектури:

– моделі (Models) – класи Eloquent, що представляють сутності системи та забезпечують операції з БД;

– представлення (Views) – Blade-шаблони, що генерують HTML на основі переданих даних;

– контролери (Controllers) – класи, що обробляють HTTP-запити, взаємодіють з моделями та повертають представлення.

На рис. 3.2 представлено діаграму моделей та їх зв’язки.

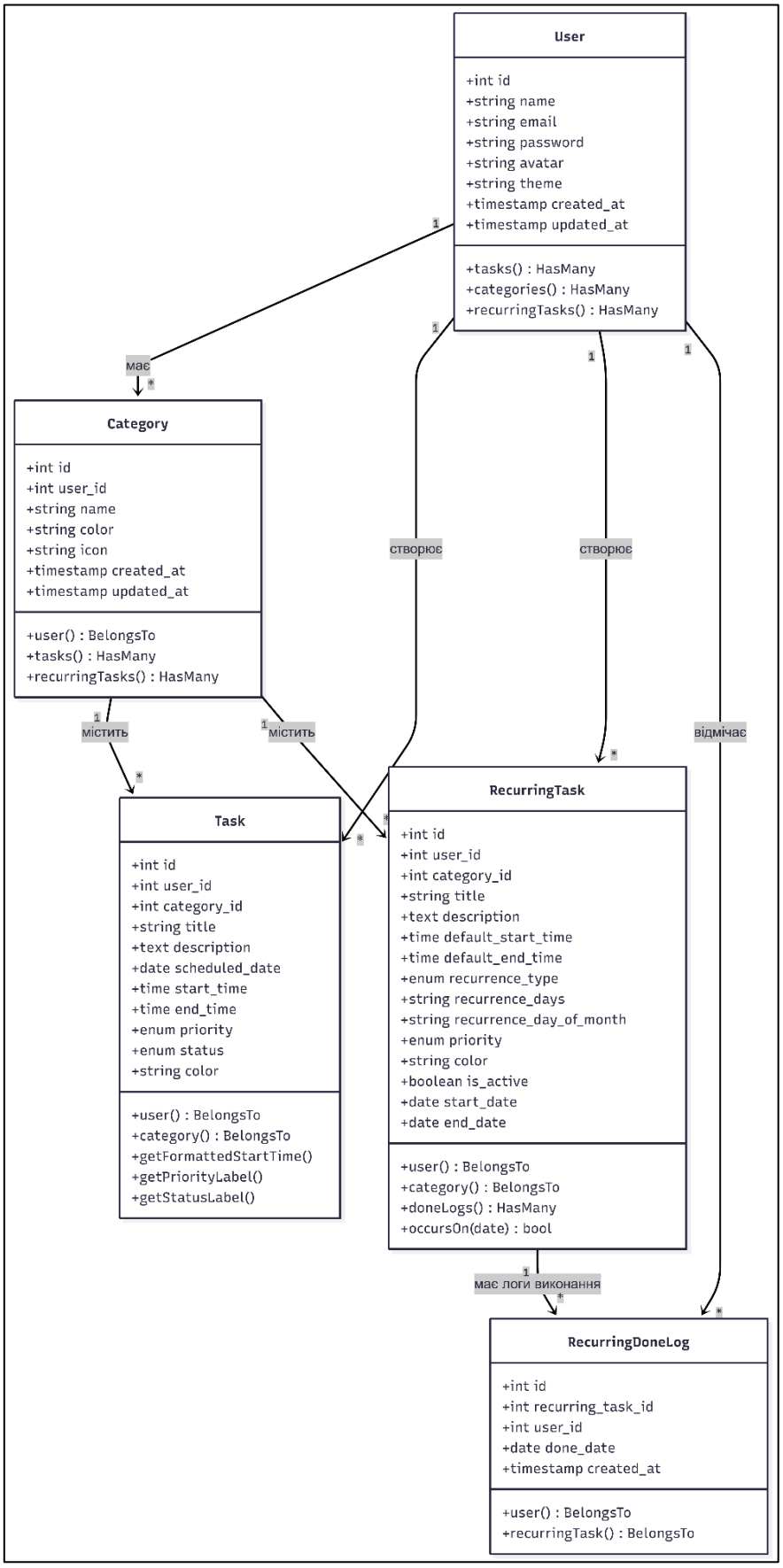

Рисунок 3.2 – Діаграма моделей та їх зв'язків

Контролери системи згруповано за функціональними розділами. Детально їх описано в табл. 3.8.

Таблиця 3.8 – Контролери системи та їх призначення

Контролер	Призначення	Основні методи
AuthController	Автентифікація	showLogin, login, showRegister, register, logout
DashboardController	Головна сторінка	index (з параметром date для навігації по днях)
TaskController	Задачі та розклад	schedule, week, store, update, updateStatus, destroy, getTasksJson
RecurringTaskController	Повторювані задачі	index, store, update, toggleActive, markDone, destroy
CategoryController	Категорії	index, store, update, destroy, pluralTasks (helper)
AnalyticsController	Аналітика	index (з параметром period), calculateStreak, getPeriodRange
AdvisorController	Радник та автопланер	index, plan, accept, generateTips, findFreeSlots, scoreSlot
ProfileController	Профіль користувача	edit, update, updatePassword, uploadAvatar, removeAvatar, destroy
SettingsController	Налаштування	index, updateTheme

Blade-шаблони представлення організовані за функціональними розділами і описані в табл. 3.9.

Таблиця 3.9 – Blade-шаблони системи

Blade-шаблон	Призначення
layouts/app.blade.php	Основний layout з sidebar, topbar, theme-toggle; підключає CSS/JS (JavaScript); діє як оболонка для всіх автентифікованих сторінок
auth/login.blade.php	Сторінка входу з розділеним layout та брендованою лівою панеллю
auth/register.blade.php	Сторінка реєстрації в тому ж стилі, що й вхід
dashboard/index.blade.php	Головна сторінка з 4 stat-картками, розкладом на день з навігацією, прогресом дня, найближчими задачами, категоріями
tasks/schedule.blade.php	Погодинний розклад на день з 24 рядками годин, задачами у вигляді task-item карток, модальними вікнами додавання/редагування
tasks/week.blade.php	Тижневий розклад у вигляді таблиці 7 × 24 з підсвічуванням поточного дня
analytics/index.blade.php	Аналітика з вибором періоду, 4 stat-блоками, 4 графіками Chart.js, розподілом по категоріях
tips/index.blade.php	Сторінка порад з переліком tip-карток зліва та формою автопланера справа
recurring/index.blade.php	Список шаблонів повторюваних задач з toggle-кнопками активності
recurring/_form.blade.php	Partial-форма для створення/редагування повторюваної задачі
categories/index.blade.php	Картки категорій з лічильниками задач та модальним вікном редагування
profile/edit.blade.php	Сторінка профілю з аватаром, формами редагування даних та пароля, статистикою акаунту, зоною видалення
settings/index.blade.php	Налаштування з вибором теми оформлення через превью-картки

Додаткові архітектурні компоненти:

– маршрути (Routes) – файл routes/web.php містить усі маршрути застосунку, згруповані під middleware «auth» для захищених сторінок і окремо для гостьових;

– міграції (Migrations) – файли в database/migrations/ описують структуру БД та дозволяють версіонувати зміни;

– сідер (DatabaseSeeder) – наповнює БД демонстраційними даними для тестування;

– публічні ресурси – CSS у public/css/, JS у public/js/, завантажені аватари у public/uploads/avatars/.

Комплексна архітектура з використанням MVC та додаткових компонентів Laravel забезпечує гнучкість, масштабованість і зручність підтримки системи.

3.3 Проєктування бази даних

База даних є центральним компонентом системи «TimeManager», оскільки вона зберігає всі дані користувачів: задачі, повторювані шаблони, категорії, логи виконання, налаштування профілю [18]. Для проєктування використано принципи нормалізації (3NF) для мінімізації дублювання та забезпечення цілісності даних [19].

На рис. 3.3 представлено ER-діаграму бази даних.

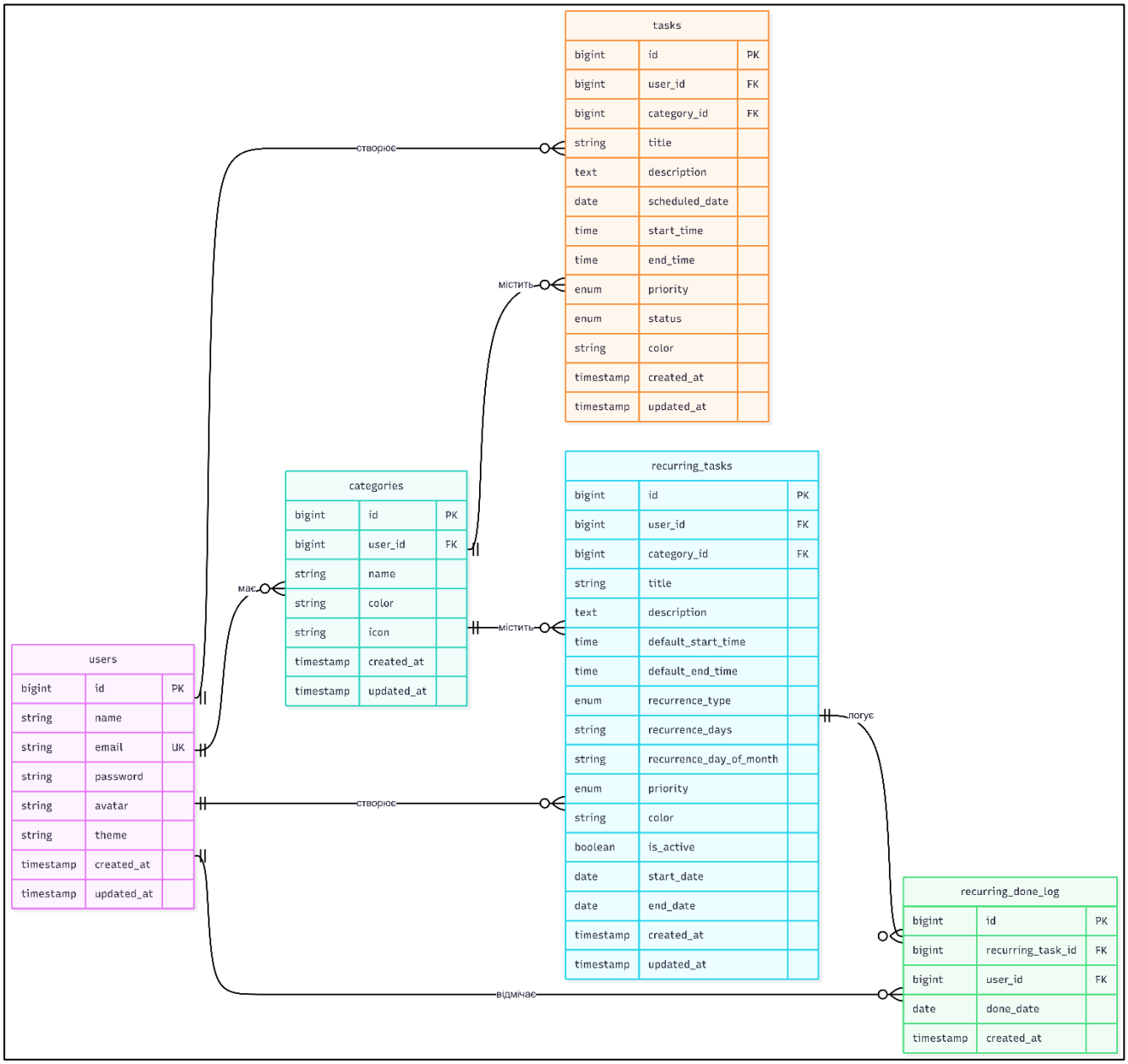

Рисунок 3.3 – ER-діаграма бази даних

Структура таблиць бази даних описана в табл. 3.10.

Таблиця 3.10 – Таблиці бази даних та їх призначення

Таблиця	Призначення
users	Облікові записи користувачів: ім'я, email, хешований пароль, шлях до аватара, обрана тема оформлення

Кінець таблиці 3.10

Таблиця	Призначення
categories	Категорії для групування задач (наприклад, Робота, Особисте, Навчання) з назвою та кольором; належать конкретному користувачу
tasks	Окремі задачі користувача: назва, опис, дата, час початку/закінчення, пріоритет (low/medium/high), статус (pending/in_progress/completed/cancelled), колір, категорія
recurring_tasks	Шаблони повторюваних задач: назва, час, пріоритет, категорія, тип повторення (daily/weekly/monthly), дні тижня, дні місяця, період дії, ознака активності
recurring_done_log	Лог виконання повторюваних задач: пара «recurring_task_id + done_date» вказує, що конкретна повторювана задача виконана в конкретну дату
sessions	Сесії користувачів, створюється автоматично Laravel
password_reset_tokens	Токени для відновлення паролю, створюється автоматично
migrations	Історія виконаних міграцій, створюється автоматично

Ключові зв'язки між таблицями:

– users → tasks (один до багатьох): користувач має багато задач, кожна задача належить одному користувачу;

– users → categories (один до багатьох): користувач має власні категорії;

– users → recurring_tasks (один до багатьох): користувач має власні шаблони повторюваних задач;

– categories → tasks (один до багатьох): одна категорія охоплює багато задач;

– categories → recurring_tasks (один до багатьох);

– recurring_tasks → recurring_done_log (один до багатьох): для одного шаблону – багато відміток виконання на різні дати;

– users → recurring_done_log (один до багатьох): для швидкого пошуку виконань по user_id.

Індексація: для оптимізації запитів створено індекси на зовнішніх ключах (user_id, category_id, recurring_task_id) та на полях, за якими часто здійснюється пошук (scheduled_date, done_date, status).

Цілісність даних забезпечується каскадним видаленням: при видаленні користувача автоматично видаляються всі його задачі, категорії, повторювані шаблони та логи. При видаленні категорії поле category_id у задачах встановлюється в NULL (onDelete('set null')), а задача залишається.

Така структура БД забезпечує ефективне зберігання даних, легкість вибірки інформації для всіх модулів системи та цілісність при змінах.

3.4 Моделювання модулів системи

Система «TimeManager» складається з низки функціональних модулів, кожен з яких відповідає за певну частину логіки. Нижче розглянуто основні модулі та їх взаємодію.

1. Модуль автентифікації. Відповідає за реєстрацію, вхід, вихід користувачів. Використовує вбудовані механізми Laravel: хешування паролів через bcrypt, middleware auth для захисту маршрутів, CSRF-токени для форм. При реєстрації автоматично створюються п’ять стандартних категорій для нового користувача (Робота, Особисте, Навчання).

2. Модуль розкладу. Центральний модуль системи. Надає два подання:

– погодинний розклад на день – 24 рядки-години з можливістю вставити задачу клацанням на слот;

– тижневий розклад – таблиця 7 днів × 24 години з відображенням усіх задач.

У модулі реалізовано спеціальну логіку для відображення багатогодинних задач: задача, що триває понад годину, позиціонується абсолютно всередині

стартової години з обчисленою висотою. Це гарантує коректне візуальне відображення без «дрейфу» по вертикалі.

3. Модуль повторюваних задач. Забезпечує шаблони, що автоматично відображаються в розкладі у відповідні дати. Підтримує три типи повторення:

– щоденне (daily);

– по конкретних днях тижня (weekly) – обираються зі списку «ПН ВТ СР ЧТ ПТ СБ НД»;

– по конкретних днях місяця (monthly) – обираються числа 1–31 з сіткою.

У моделі RecurringTask метод occursOn визначає, чи буде завдання повторюватися на певну дату, враховуючи період активності (start_date, end_date). Модуль recurring_done_log дозволяє відзначати завершення на конкретну дату без зміни шаблону.

4. Модуль категорій. Простий модуль CRUD для управління категоріями. Кожна категорія має назву та колір. Колір автоматично застосовується до всіх завдань у цій категорії, якщо завдання не має власного набору.

5. Модуль аналітики. Агрегує дані про задачі й повторювані задачі за обраний період (тиждень/місяць/рік). Обчислює:

– загальну кількість задач, кількість виконаних, % виконання;

– серію продуктивних днів (streak) – кількість днів поспіль з виконаними задачами;

– розподіл задач по годинах доби;

– розподіл по днях тижня;

– розподіл по пріоритетах та категоріях;

– пікову годину продуктивності.

Результати візуалізуються через Chart.js: стовпчикові діаграми, лінійні графіки, кільцева діаграма, прогрес-бари по категоріях.

6. Модуль інтелектуального радника. Включає 10 правил (які реалізуються методом generateTips контролера AdvisorController), що аналізують шаблони поведінки користувача для створення персоналізованих порад. Ці поради потім

класифікуються за рівнем важливості: інформація, успіх, попередження, небезпека. Кожна порада може включати посилання на дію (наприклад, перейти до розкладу для перегляду прострочених завдань).

7. Модуль автопланера. Реалізує алгоритм findFreeSlots з розділу 2.2. Окремо – метод accept, який приймає обраний слот і створює з нього повноцінну задачу в розкладі.

8. Модуль налаштувань. Дозволяє користувачу переключати тему (світла/темна). Вибір зберігається в полі users.theme та в localStorage для швидкого застосування при завантаженні сторінки. CSS-змінні (кастомні властивості CSS) дозволяють перемикання, і перемикаються за допомогою атрибута data-theme на елементі html.

Чітке розмежування функціоналу між модулями спрощує розробку, тестування та подальше розширення системи.

3.5 Макети сторінок сайту

Перед програмною реалізацією розроблено дизайн-систему з уніфікованими компонентами, що забезпечує візуальну послідовність усіх сторінок та спрощує розробку. Нижче розглянуто основні елементи дизайн-системи.

Кольорова схема (світла тема):

- основний акцентний колір: #4f46e5 (індиго);
- колір успіху: #10b981 (зелений);
- колір попередження: #f59e0b (бурштиновий);
- колір небезпеки: #ef4444 (червоний);
- колір тексту: #374151 (темно-сірий);
- колір заголовків: #111827 (майже чорний);
- фон сторінки: #f8faff (світло-блакитний);
- фон карток: #ffffff (білий);
- колір рамок: #e5e7eb (світло-сірий).

Кольорова схема (темна тема):

– основний акцентний колір: #818cf8 (світло-індиговий);

– фон сторінки: #0f172a (глибокий темно-синій);

– фон карток: #1e293b (темно-сірий);

– колір тексту: #d1d5db (світло-сірий);

– колір заголовків: #f3f4f6 (майже білий).

Типографіка:

– основний шрифт – Inter (sans-serif), що забезпечує чітке читання на всіх пристроях;

– розміри: базовий 15 px, заголовки – від 0.88 rem до 2.4 rem.

Компоненти інтерфейсу:

– кнопки (первинні, вторинні, контурні, іконки);

– форми введення (текстові поля, селектори, time-picker, color-picker);

– картки (stat-картки дашборду, картки категорій, картки повторюваних задач, задачі в розкладі);

– бейджі (пріоритет, статус, категорія);

– модальні вікна для створення/редагування;

– навігаційна бічна панель (sidebar) з іконками.

Для забезпечення візуальної цілісності розроблено макети основних сторінок системи у Figma, враховуючи принципи сучасного UX/UI-дизайну та потреби цільової аудиторії, визначені в розділі 1.2.

На макеті дашборду представлено 4 статистичні картки у верхній частині (задачі на день, виконано, задачі за тиждень, прострочені), блок з розкладом на обраний день зліва та колонку найближчих задач і категорій справа (рис. 3.4).

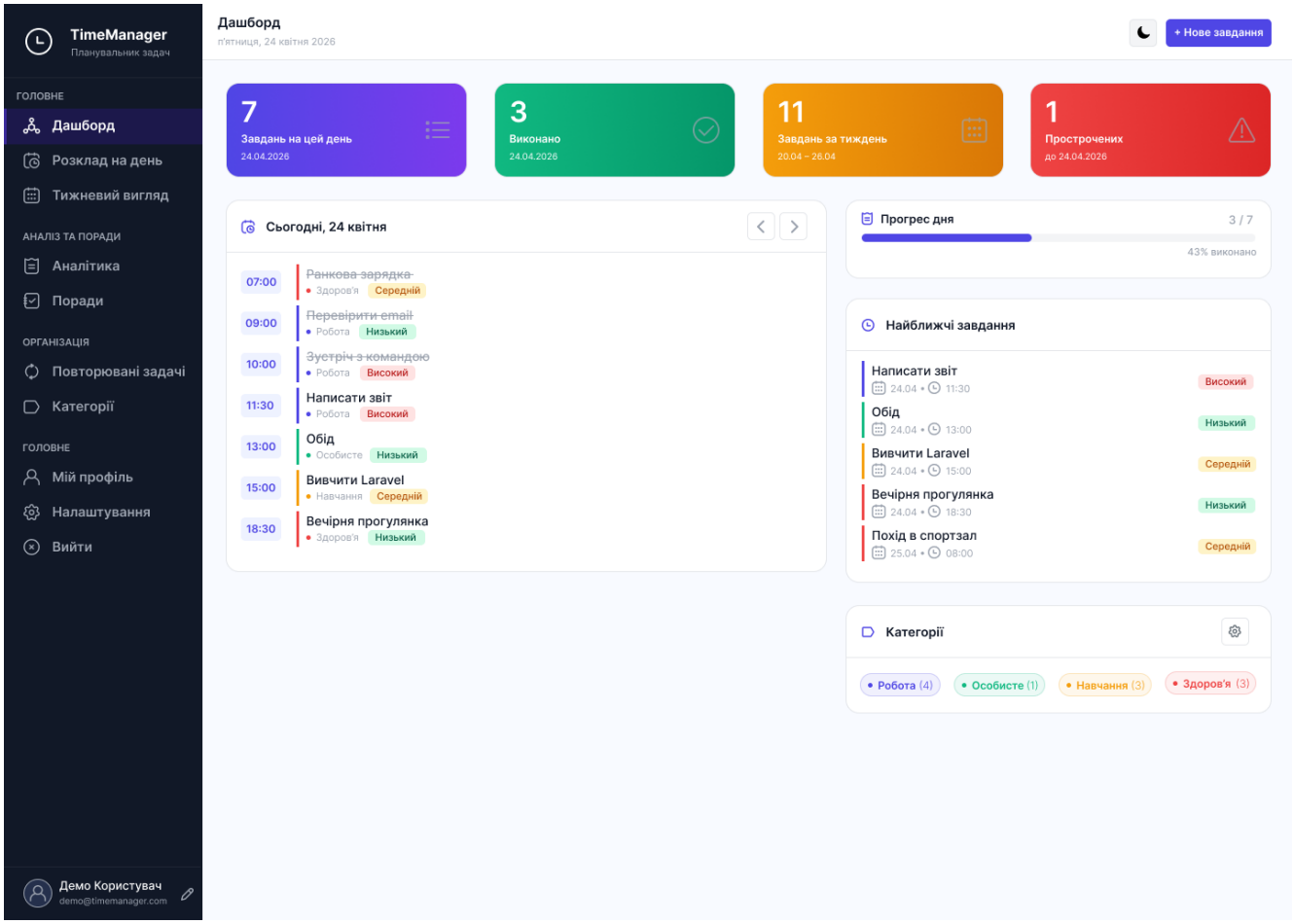

Рисунок 3.4 – Макет головної сторінки (дашборд)

Сторінка розкладу побудована як вертикальний таймлайн з 24 годинами. Кожна задача відображається як картка з часом, назвою, категорією, пріоритетом та кнопками дій (рис. 3.5).

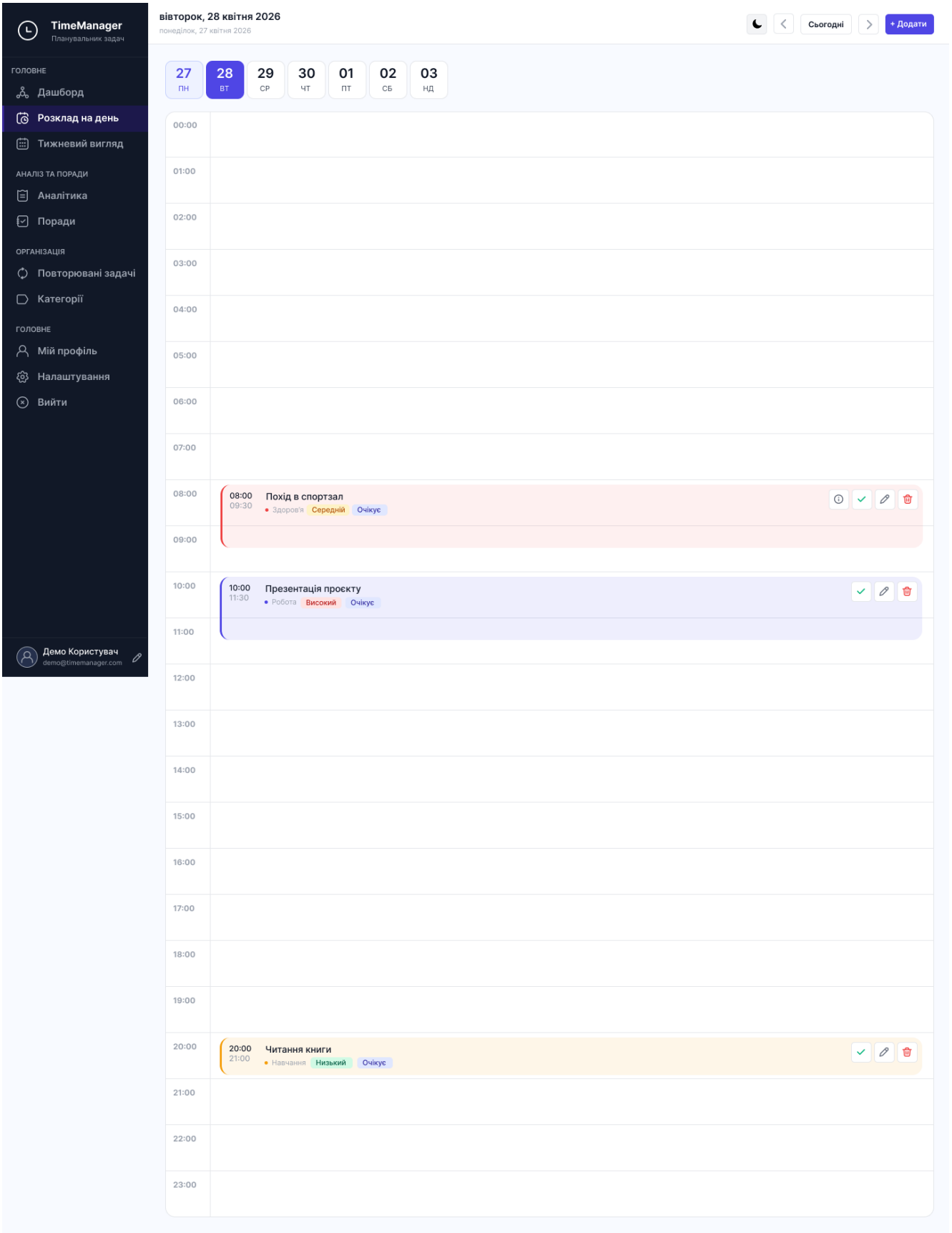

Рисунок 3.5 – Макет сторінки розкладу на день

На сторінці аналітики розміщено перемикач періоду (тиждень/місяць/рік), 4 статистичні блоки, стовпчиковий графік задач по днях, кільцеву діаграму пріоритетів, лінійний графік активності по годинах (рис. 3.6).

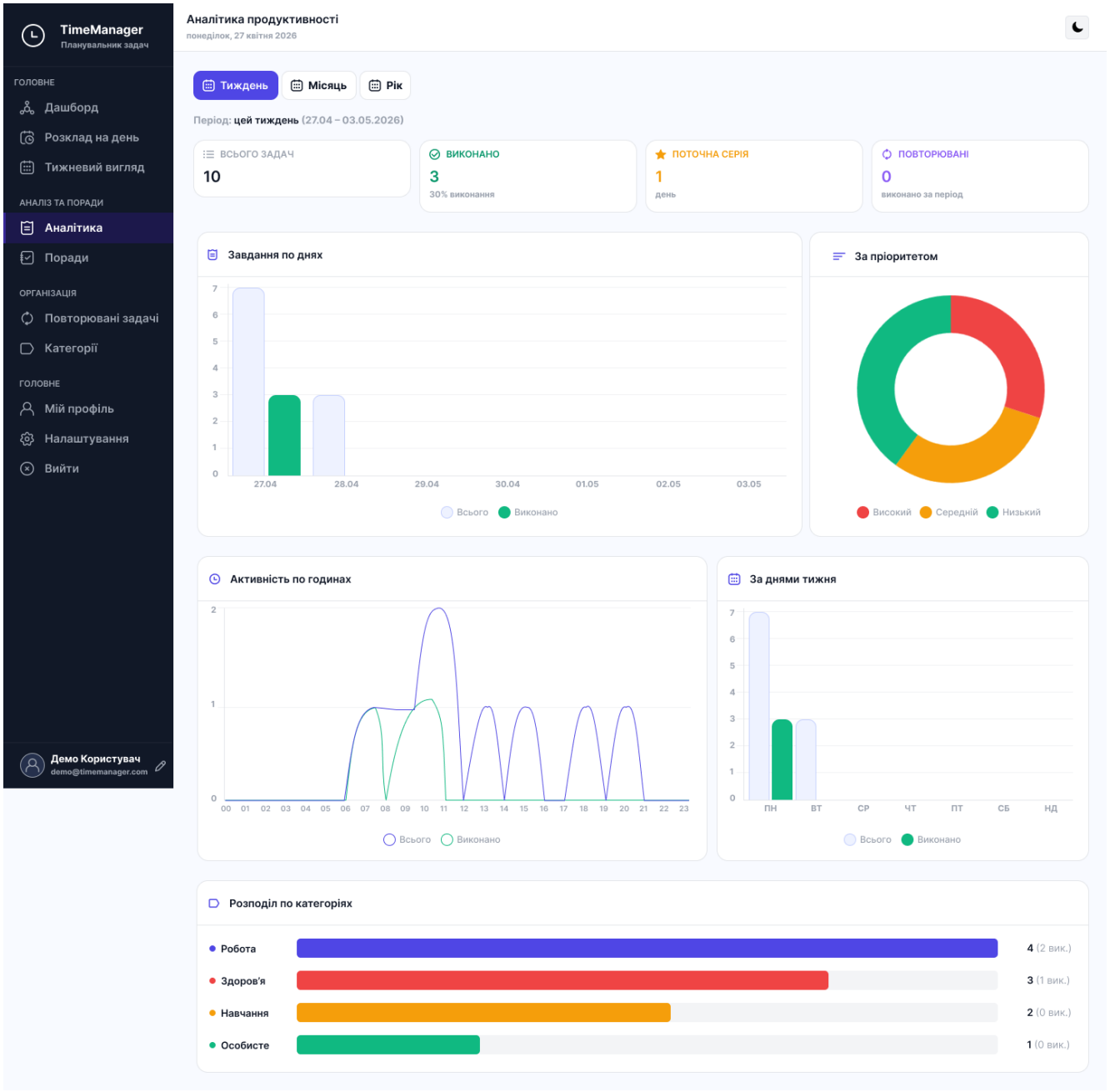

Рисунок 3.6 – Макет сторінки аналітики

Наведено зображення головної сторінки у темній темі, що демонструє роботу системи CSS-змінних (рис. 3.7).

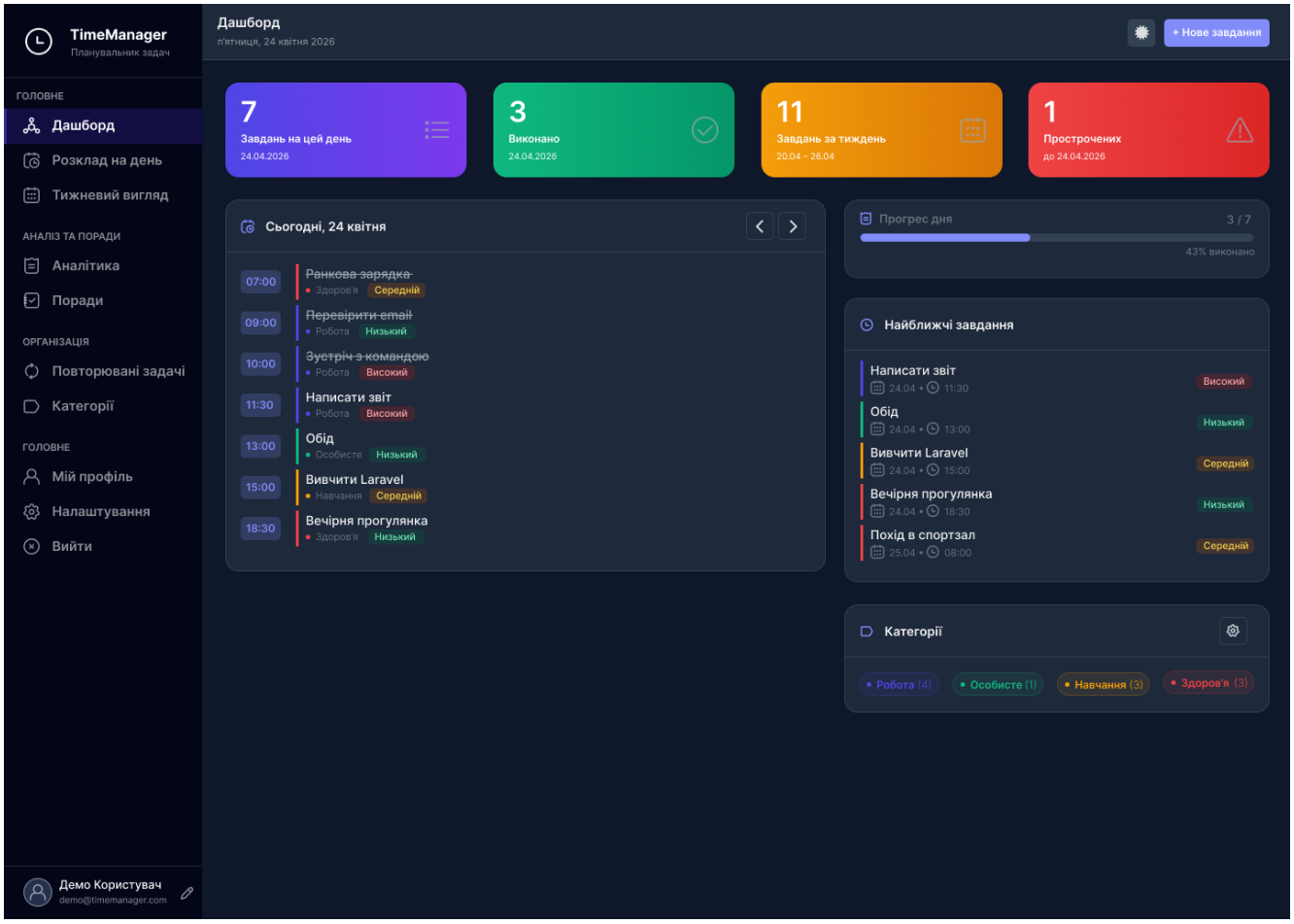

Рисунок 3.7 – Макет головної сторінки (темна тема)

Розроблені макети дозволили перейти до реалізації з чітким розумінням кінцевого вигляду системи та забезпечили узгодженість стилів між усіма сторінками.

3.6 Аналіз отриманих результатів

За результатами проєктування та реалізації системи «TimeManager» отримано повнофункціональний вебзастосунок, що відповідає всім функціональним і нефункціональним вимогам, поставленим у розділі 1.4.

Функціональні досягнення:

– реалізовано всі 8 запланованих модулів (автентифікація, розклад, повторювані задачі, категорії, аналітика, радник, автопланер, налаштування);

– створено зручний інтерфейс з адаптивним дизайном і підтримкою світлої/темної теми;

– реалізовано інтелектуальний автопланер, що знаходить оптимальні слоти для нових задач за мілісекунди;

– система розумного радника генерує до 10 типів персональних порад.

Реалізований функціонал виходить за межі класичного планувальника та забезпечує користувачеві активну допомогу в управлінні часом, що відповідає головній меті дослідження. Ключові інтелектуальні можливості – автопланер та радник – працюють у режимі реального часу та не потребують від користувача додаткових налаштувань.

Технічні досягнення:

– розроблено чітку архітектуру MVC з 9 контролерами, 5 моделями та 13 Blade-шаблонами;

– спроєктовано нормалізовану базу даних з 8 таблиць;

– реалізовано CSS-систему з CSS-змінними для підтримки тем без дублювання стилів;

– використано Chart.js для візуалізації аналітики;

– впроваджено безпекові механізми Laravel: CSRF-захист, хешування паролів, Eloquent ORM для захисту від SQL-ін'єкцій, валідація даних.

Відповідність нефункціональним вимогам:

– продуктивність – типова сторінка завантажується за < 1.5 сек на локальному сервері;

– безпека – виконано вимоги до захисту даних;

– зручність – інтерфейс інтуїтивно зрозумілий, навігація логічна;

– адаптивність – інтерфейс коректно відображається на екранах від 480 px (телефони) до 1920+ px (монітори);

– надійність – завдяки використанню транзакцій Laravel та ORM.

Таким чином, результати проєктування показують, що обраний підхід (MVC, Eloquent, Chart.js, CSS custom properties) дозволив побудувати систему, яка не лише

задовольняє базові вимоги, а й пропонує інтелектуальні функції, що вирізняють її серед аналогів.

Висновки до розділу 3

У третьому розділі виконано проєктування інтелектуальної системи «TimeManager» та аналіз отриманих результатів.

Розроблено п'ять основних сценаріїв використання (реєстрація та вхід, додавання задачі, використання автопланера, перегляд аналітики та порад, робота з повторюваними задачами), що охоплюють ключові функції системи.

Спроєктовано архітектуру системи на основі патерну MVC з використанням Laravel 10. Визначено роль та задачі кожного з 9 контролерів та 13 Blade-шаблонів.

Спроєктовано структуру бази даних з 8 таблиць (users, categories, tasks, recurring_tasks, recurring_done_log, sessions, password_reset_tokens, migrations). Визначено зв'язки між таблицями та правила цілісності.

Описано моделі 8 модулів системи: автентифікації, розкладу, повторюваних задач, категорій, аналітики, інтелектуального радника, автопланера та налаштувань.

Розроблено дизайн-систему з уніфікованими компонентами (кольорова схема, типографіка, компоненти інтерфейсу) та макети основних сторінок системи у Figma, що забезпечили візуальну цілісність реалізованого застосунку.

Проведено аналіз отриманих результатів: підтверджено відповідність функціональним та нефункціональним вимогам, виявлено переваги обраного технологічного підходу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

У цьому розділі описано практичну реалізацію інтелектуальної системи «TimeManager» відповідно до проєктних рішень, розроблених у попередніх розділах. Висвітлено процес розробки та результати тестування системи. Приклади програмного коду основних файлів винесені у додаток А.

4.1 Підготовка середовища розробки

Для ефективної розробки системи було підготовлено локальне середовище на основі OpenServer – популярного для Windows програмного комплексу з вбудованими Apache, PHP, MySQL та phpMyAdmin.

Конфігурація середовища:

– локальний сервер: OpenServer 5.4 з PHP 8.4, MySQL 8.4, Apache 2.4;

– редактор коду: Visual Studio Code з розширеннями PHP Intelephense, Laravel Blade Snippets, Prettier;

– пакетний менеджер: Composer 2.6 [20].

Встановлення проєкту:

– через Composer встановлено Laravel 10: composer create-project laravel/laravel timemanager;

– створено базу даних «timemanager» через phpMyAdmin;

– налаштовано файл .env: параметри підключення до БД, встановлено APP_TIMEZONE=Europe/Kyiv;

– виконано команду php artisan key:generate для генерації ключа шифрування;

– встановлено CDN-залежності: Bootstrap 5.3, Bootstrap Icons 1.11, Chart.js 4.4, Google Fonts (Inter).

4.2 Реалізація бази даних та міграцій Laravel

Структура бази даних реалізована через систему міграцій Laravel, що забезпечує версіонування схеми БД та дозволяє легко застосовувати зміни.

Для кожної таблиці створено окрему міграцію.

Перелік реалізованих міграцій:

– 000002_create_tasks_table – створює tasks з усіма полями задачі та зовнішніми ключами до users і categories;

– 000003_create_categories_table – створює таблицю categories з полями id, user_id, name, color, created_at, updated_at;

– 000004_add_category_to_tasks – додає зовнішній ключ category_id до таблиці tasks;

– 000005_create_recurring_tasks_table – створює таблицю шаблонів повторюваних задач;

– 000006_create_recurring_done_log_table – створює лог виконання повторюваних задач;

– 000007_add_avatar_to_users – додає поле avatar до таблиці users;

– 000008_change_recurrence_day_of_month_to_string – змінює тип поля recurrence_day_of_month з int на string, щоб підтримувати CSV-формат для кількох днів місяця;

– 000099_ensure_avatar_column_on_users – захисна міграція з перевіркою через Schema::hasColumn();

– 000100_add_theme_to_users – додає поле theme зі значенням за замовчуванням 'light'.

Міграції застосовуються командою php artisan migrate. Використано механізм зовнішніх ключів з каскадним видаленням для забезпечення цілісності.

Моделі Eloquent (app/Models/):

– User – з $fillable включно з avatar та theme (критично для роботи оновлення цих полів через update());

– Category – зв'язок hasMany з Task та RecurringTask;

– Task – зв'язки belongsTo з User та Category; accessor-атрибути formatted_start_time, formatted_end_time, priority_label, status_label;

– RecurringTask – зв'язки belongsTo з User та Category; має метод occursOn($date), accessor recurrence_days_array;

– RecurringDoneLog – просте відображення логу виконання [21].

Для наповнення БД тестовими даними реалізовано DatabaseSeeder, який створює демонстраційного користувача (email demo@timemanager.com, пароль password), кілька категорій, набір задач на поточний тиждень та кілька повторюваних шаблонів.

4.3 Реалізація бекенду

Серверна частина системи реалізована на Laravel 10 з дотриманням принципів MVC, модульності та безпеки. Реалізовано 9 контролерів та всі необхідні маршрути [22].

Система маршрутизації (routes/web.php) структурована наступним чином:

– гостьові маршрути (middleware 'guest'): /login, /register для входу/реєстрації;

– захищені маршрути (middleware 'auth'): всі основні функції системи;

– API-маршрут /api/tasks – повертає JSON для AJAX-запитів при редагуванні задач.

Усі маршрути іменовані, що дозволяє використовувати route() helper для генерації URL та спрощує зміну структури URL у майбутньому.

Ключові функціональні блоки бекенду описано нижче.

1. Авторизація та реєстрація. Реалізована у AuthController без використання сторонніх пакетів – безпосередньо через Laravel Auth facade. При реєстрації валідуються дані (email – унікальний, пароль – мінімум 8 символів, підтвердження), пароль хешується bcrypt'ом, і створюються п’ять стандартних категорій для нового користувача.

2. CRUD задач. У TaskController реалізовано методи store, update, destroy, updateStatus. Перед кожною операцією перевіряється, чи належить задача поточному користувачу (за $task->user_id === Auth::id()). Інакше повертається 403 Forbidden. Це гарантує розмежування даних між користувачами.

3. Модуль розкладу. Методи schedule() і week() в TaskController агрегують одноразові задачі (з таблиці tasks) та повторювані (через recurring_tasks + виклик occursOn() для кожного дня), сортують їх за часом та передають у представлення. Для методу week() додатково повертається карта виконань повторюваних задач (recurring_done_log) за тиждень.

4. Модуль аналітики. AnalyticsController::index приймає параметр period (week/month/year) і відповідно до нього визначає діапазон дат. Всі агрегації виконуються в PHP над об'єктом Collection для гнучкості (groupBy, map, filter). Для потужних виборок можна було б перейти на SQL-агрегації, але поточний підхід дає достатню швидкість при розмірах даних, типових для особистого використання.

5. Модуль інтелектуального радника. AdvisorController::generateTips реалізує 10 правил у вигляді послідовних умовних конструкцій. Кожне правило перевіряє певну умову (наприклад, $overdue > 0) та, у разі її виконання, додає пораду з визначеним severity, title, text та опційним action. Метод повертає масив порад, що далі рендериться у Blade-шаблоні.

6. Автопланер. Реалізований у методах plan() і accept() AdvisorController. plan() приймає JSON-дані з форми, викликає findFreeSlots() з потрібними параметрами та повертає JSON з до 8 найкращими слотами. accept() створює повноцінну задачу в таблиці tasks на основі обраного слоту.

Безпека:

- CSRF-токени автоматично включені в усі форми через @csrf директиву;
- Eloquent ORM захищає від SQL-ін'єкцій через параметризовані запити;
- валідація вхідних даних – через $request->validate() з правилами;
- хешування паролів – bcrypt через Hash::make();
- розмежування даних – перевірка user_id у кожному контролері [23].

4.4 Реалізація фронтенду та адаптивного інтерфейсу

Клієнтська частина системи розроблена з орієнтацією на створення сучасного, зручного й адаптивного інтерфейсу. Використано Blade-шаблонізатор Laravel для генерації HTML, Bootstrap 5 для сітки й базових компонентів, власну CSS-систему з підтримкою тем, JavaScript для інтерактивності [24, 25].

Структура CSS-файлів (public/css/):

– app.css – глобальні стилі, CSS-змінні для обох тем;

– dark-overrides.css – специфічні перевизначення для темної теми;

– schedule.css – стилі розкладу дня;

– dashboard.css – стилі дашборду;

– analytics.css – стилі аналітики;

– tips.css – стилі порад та автопланера;

– categories.css – стилі сторінки категорій;

– auth.css – стилі сторінки входу та реєстрації;

– profile.css – стилі сторінки профілю користувача;

– recurring.css – стилі повторюваних задач;

– settings.css – стилі сторінки налаштувань;

– week.css – стилі сторінки тижневого розкладу.

Структура JavaScript-файлів (public/js/):

– theme.js – завантажується в <head> для моментального застосування теми (уникнення спалаху світлого контенту);

– app.js – глобальні утиліти: CSRF helper, керування sidebar, color picker;

– schedule.js – логіка сторінки розкладу: time picker, модальні вікна, AJAX-операції;

– dashboard.js – inline дії з задачами на дашборді.

Система тем оформлення. Реалізована через CSS-змінні:

```
:root { --primary: #4f46e5; --bg: #f8faff; --text: #374151; ... }
```

[data-theme="dark"] { --primary: #818cf8; --bg: #0f172a; --text: #d1d5db; ... }

Перемикання теми здійснюється зміною атрибута data-theme на елементі html. JavaScript зберігає вибір у localStorage для моментального застосування при наступних відвідуваннях, а також надсилає AJAX-запит до /settings/theme для збереження в БД [26].

Адаптивний дизайн. Реалізовано через чотири основні breakpoint'и:

– > 1280 px – повна розкладка, sidebar 260 px;

– 1024–1280 px – ноутбук, sidebar 230 px, тісніші відступи;

– 768–1024 px – планшет, sidebar 220 px, дрібніший шрифт;

– < 768 px – мобільний, sidebar стає overlay з backdrop, з'являється кнопка-гамбургер;

– < 480 px – маленький телефон, ховаються підписи, бейджі скорочуються [27, 28].

Візуалізація даних. На сторінці аналітики використано Chart.js для побудови:

– стовпчикової діаграми задач по днях (всього / виконано);

– кільцевої діаграми розподілу по пріоритетах;

– лінійного графіка активності по годинах;

– стовпчикової діаграми по днях тижня [29].

Розподіл по категоріях реалізовано як прогрес-бари без Chart.js – це дає більш детальне візуальне представлення з кольорами категорій.

4.5 Реалізація ключових алгоритмів системи

У цьому підрозділі детально розглянуто реалізацію двох найскладніших алгоритмів системи – автопланера задач та інтелектуального радника, які складають інтелектуальне ядро застосунку.

Алгоритм пошуку вільних часових слотів. Метод findFreeSlots у класі AdvisorController реалізує логіку, описану в розділі 2.2. Алгоритм проходить по

кожному дню в заданому діапазоні дат із кроком 15 хвилин у межах робочих годин користувача та для кожного потенційного слоту перевіряє відсутність конфлікту з існуючими задачами.

Послідовність дій алгоритму така:

– ітерація по кожному дню в діапазоні від date_from до date_to;

– для поточного дня формується список зайнятих часових інтервалів – окремі задачі з таблиці tasks плюс повторювані задачі, що припадають на цей день;

– у межах робочих годин (від hour_from до hour_to) перебираються потенційні слоти з кроком 15 хвилин;

– для кожного слоту перевіряється, чи не накладається він на жоден із зайнятих інтервалів;

– вільні слоти отримують оцінку від функції scoreSlot та додаються до результуючого списку;

– усі знайдені слоти сортуються за спаданням оцінки, і повертаються 8 найкращих.

Особлива складність алгоритму – **врахування повторюваних задач**. Звичайні задачі легко вибрати запитом до таблиці tasks, а повторювані потребують програмної перевірки кожного шаблону через метод occursOn. Цей метод визначає, чи припадає шаблон на конкретну дату з урахуванням типу повторення (щодня / по днях тижня / по днях місяця) та періоду дії шаблону.

Перевірка конфлікту виконується класичним перетином інтервалів: два інтервали часу перетинаються тоді й лише тоді, коли початок одного раніший за кінець іншого, і навпаки. Ця проста умова дозволяє швидко відсіювати непридатні слоти.

Функція оцінки слотів. Метод scoreSlot обчислює пріоритетність кожного знайденого вільного слоту на основі математичної моделі з розділу 2.3. Базова оцінка – 50 балів, до якої додається модифікатор залежно від години доби та преференцій користувача. Найвищі оцінки отримують пікові години продуктивності (10:00–12:00 та 15:00–17:00 у звичайному режимі або 8:00–11:00 у

режимі ранкової переваги). Після обчислення оцінок усі знайдені слоти сортуються за спаданням, і користувачеві повертаються найкращі варіанти.

Реалізація системи правил радника. Метод generateTips побудований за принципом продукційної моделі знань: кожне з 10 правил перевіряє певну умову й, у разі її виконання, додає пораду в результуючий масив.

Перелік правил, що перевіряються методом generateTips:

– наявність прострочених задач – генерується порада з посиланням на розклад;

– перевантажений день (більше 10 задач) – пропонується перенести частину на інші дні;

– відсутність задач на робочий день – пропонується додати задачу або відпочити;

– високий відсоток виконання за тиждень – генерується схвальна порада;

– низький відсоток виконання за тиждень – пропонується планувати менше задач;

– виявлення пікової години продуктивності – підказка про найпродуктивніший час;

– низьке дотримання повторюваних задач – пропозиція переглянути або вимкнути їх;

– багато задач на нічні години – нагадування про важливість сну;

– велика кількість незавершених задач з високим пріоритетом – рекомендація переглянути критичність;

– мала частка категоризованих задач – пропозиція використовувати категоризацію.

Кожна порада має чотири можливі рівні критичності – info, success, warning, danger, які відображаються відповідним кольором у Blade-шаблоні. Якщо жодне правило не спрацювало, генерується дефолтна порада-похвала про те, що користувач підтримує гарний баланс.

Метод визначення серії продуктивних днів. Окремої уваги заслуговує алгоритм обчислення streak – кількості послідовних днів, у які користувач виконав хоча б одну задачу. Реалізований у AnalyticsController::calculateStreak як цикл назад від поточної дати: для кожного дня перевіряється наявність хоча б однієї виконаної задачі або відмітки виконання повторюваної задачі. Якщо такі є – лічильник збільшується, і алгоритм переходить до попереднього дня. Якщо ні – цикл припиняється, окрім випадку поточного дня (його можна пропустити, оскільки сьогодні задачі ще можуть бути виконані пізніше). Максимальна глибина пошуку – 365 днів.

Особливістю цього алгоритму є дозвіл пропустити поточний день, якщо в ньому ще немає виконаних задач – це дає змогу не обнулити серію в перших годинах нового дня.

Реалізовані алгоритми забезпечують основні інтелектуальні функції системи, причому їх обчислювальна складність є лінійною щодо кількості задач користувача, що гарантує швидкий відгук системи навіть при значному обсязі даних.

4.6 Керівництво користувача

Для початку роботи з системою «TimeManager» користувачеві потрібно виконати кілька простих кроків.

1. Реєстрація та вхід. На стартовій сторінці натиснути «Створити акаунт», заповнити форму (ім'я, email, пароль ≥ 8 символів, підтвердження паролю) та натиснути «Зареєструватися». Після реєстрації користувач автоматично авторизується та перенаправляється на дашборд. Для повторного входу достатньо ввести email та пароль.

2. Дашборд. Головна сторінка системи. Відображає: чотири статистичні картки (задачі на день, виконано, задачі за тиждень, прострочені), задачі на обраний день з можливістю перемикання стрілками, прогрес дня, найближчі задачі, картки категорій. Stat-картки оновлюються відповідно до обраного дня.

3. Додавання задачі. Перейти у «Розклад на день», клацнути на вільну годину або натиснути «Додати». У модальному вікні ввести назву, опис, дату, час початку і закінчення, пріоритет, категорію та колір. Після збереження задача з'являється у відповідному часовому слоті. Для редагування – клацнути по задачі; для видалення – навести курсор і натиснути іконку корзини.

4. Робота з повторюваними задачами. Розділ «Повторювані задачі». Для створення шаблону натиснути «Додати шаблон», ввести параметри й обрати тип повторення: щодня, по днях тижня (обрати ПН–НД) або по днях місяця (обрати числа 1–31). Шаблон автоматично з'являтиметься в розкладі у всі підходящі дати. Перемикач на картці шаблону дозволяє тимчасово відключити його без видалення.

5. Використання автопланера. Розділ «Поради». У формі автопланера ввести назву задачі, тривалість, пріоритет, категорію, діапазон дат, робочий час. За бажанням увімкнути «Ранкова перевага». Натиснути «Знайти вільний час». Система пропонує до 8 оптимальних слотів, відсортованих за пріоритетністю. Клацнути на потрібний – задача додасться в розклад.

6. Перегляд аналітики. Розділ «Аналітика». Вибрати період (тиждень/місяць/рік). Відображаються статистика та графіки: кількість задач, відсоток виконання, поточна серія, пікова година продуктивності, розподіли по різних часових зрізах та категоріях.

7. Керування профілем. Розділ «Мій профіль». Можна змінити ім'я та email, завантажити фото профілю (клацнути по аватару), змінити пароль. Внизу сторінки – статистика акаунту: загальна кількість задач, виконано, повторюваних, категорій.

8. Налаштування теми. Швидке перемикання – іконка сонця/місяця в правому верхньому куті. Детально – розділ «Налаштування» з вибором теми через прев’ю-картки.

4.7 Тестування функціоналу та оцінка продуктивності

Для забезпечення якості розробленої системи проведено комплексне тестування за кількома напрямками [30].

Функціональне тестування:

– реєстрація та вхід – перевірено валідацію email'ів, паролів, відображення помилок;

– CRUD-операції задач – створення, читання, оновлення, видалення, зміна статусу;

– повторювані задачі – коректність визначення дати повторення (occursOn), робота toggle активності, відмітки виконання;

– категорії – CRUD, каскадна зміна при видаленні категорії (set null у задачах);

– автопланер – знаходження слотів, коректне обминання зайнятих інтервалів, правильне сортування за оцінкою;

– радник – коректність спрацьовування всіх 10 правил у різних ситуаціях;

– аналітика – перевірка обчислень по періодах, обчислення серії продуктивних днів;

– профіль – завантаження, оновлення та видалення аватара;

– теми оформлення – моментальне перемикання без перезавантаження, збереження вибору.

Тестування інтерфейсу:

– перевірка коректності відображення на різних розширеннях екрану (від 480 до 1920 px);

– тестування світлої та темної тем на всіх сторінках;

– перевірка навігації, sidebar на мобільних пристроях (overlay + backdrop);

– тестування модальних вікон, time-picker'ів, color-picker'ів.

Тестування безпеки:

– перевірено розмежування даних між користувачами (спроба звернутися до чужої задачі повертає 403);

– перевірено захист від CSRF (форми без токена не проходять);

– перевірено хешування паролів – у БД зберігається тільки хеш.

Результати:

– усі функціональні вимоги виконуються коректно;
– інтерфейс адекватно відображається на всіх розширеннях екрану;
– типове завантаження сторінки – менше 1 секунди на локальному сервері;
– виявлені в процесі тестування недоліки (зокрема, коректне завантаження аватара через fillable, правильне позиціонування багатогодинних задач у розкладі, адаптивна поведінка sidebar на мобільних пристроях) – були усунені.

Скриншоти працездатності системи винесено в додаток Б.

Висновки до розділу 4

У четвертому розділі описано процес програмної реалізації та тестування інтелектуальної системи «TimeManager».

Підготовлено середовище розробки на базі OpenServer з PHP 8.4, MySQL 8.4, Laravel 10, Visual Studio Code. Налаштовано проєкт Laravel, створено БД, встановлено усі залежності.

Реалізовано базу даних через систему міграцій Laravel: 8 міграцій для створення таблиць та додавання нових полів. Створено 5 моделей Eloquent з правильно налаштованими $fillable, зв'язками та accessor-атрибутами.

Реалізовано серверну частину: 9 контролерів з повним CRUD-функціоналом, система маршрутизації, алгоритми автопланера та радника. Впроваджено безпекові механізми Laravel: CSRF-захист, хешування паролів, валідацію, розмежування даних за user_id.

Реалізовано клієнтську частину: 13 Blade-шаблонів, 7 CSS-файлів з підтримкою світлої та темної теми через CSS-змінні, 4 JavaScript-файли. Створено адаптивний інтерфейс з чотирма breakpoint'ами.

Складено докладне керівництво користувача, що охоплює всі основні операції з системою.

Проведено функціональне, UI та безпекове тестування системи. Результати показують, що всі вимоги виконуються, інтерфейс адекватно відображається на різних пристроях, типове завантаження сторінки займає менше секунди.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра розроблено інтелектуальну вебсистему з особистого тайм-менеджменту «TimeManager», що відповідає всім поставленим функціональним та нефункціональним вимогам.

У ході виконання роботи досягнуто наступного:

1. Проаналізовано проблематику персонального тайм-менеджменту та цільову аудиторію. Виявлено, що сучасні користувачі потребують не просто списку задач, а інтелектуального помічника, здатного пропонувати рішення.

2. Виконано порівняльний аналіз п'яти провідних аналогів (Worksection, Google Calendar, TickTick, Notion, Any.do). Виявлено типові недоліки: відсутність інтелектуального автопланера, обмежена аналітика, розумні функції лише у платних версіях. На основі цього сформульовано ключову цінність розроблюваної системи.

3. Розроблено математичну модель та алгоритми інтелектуального аналізу розкладу: алгоритм пошуку вільних слотів, функцію оцінки часових слотів на основі пікових годин продуктивності, систему з 10 правил для генерації персоналізованих порад.

4. Обґрунтовано вибір технологічного стеку: PHP 8.4 + Laravel 10, MySQL 8.4, Bootstrap 5, Chart.js, JavaScript ES6+. Використані технології є зрілими, безкоштовними й забезпечують всі необхідні функції.

5. Спроєктовано архітектуру системи за патерном MVC. Розроблено 9 контролерів, 5 моделей Eloquent, 13 Blade-шаблонів та структуру БД з 8 таблиць.

6. Реалізовано 8 функціональних модулів системи: автентифікація, розклад дня, тижневий розклад, повторювані задачі, категорії, аналітика, інтелектуальний радник, автопланер, налаштування теми.

7. Створено адаптивний інтерфейс користувача з підтримкою світлої та темної теми оформлення. Інтерфейс коректно відображається на пристроях з різними розширеннями екрану – від телефонів (480 px) до великих моніторів (1920+ px).

8. Проведено комплексне тестування системи за функціональним, UI та безпековим напрямками. Результати демонструють коректну роботу всіх функцій, відповідність вимогам продуктивності та безпеки.

Таким чином, розроблений вебзастосунок «TimeManager» є цілісним інтелектуальним рішенням для персонального тайм-менеджменту, що поєднує класичні функції планування з інноваційними можливостями автопланування та розумних рекомендацій. Система готова до практичного використання та має потенціал для подальшого розширення (наприклад, впровадження машинного навчання для персоналізації моделі оцінки часових слотів, додавання мобільного застосунку через REST API, інтеграція із зовнішніми календарями).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Allen D. Getting Things Done: The Art of Stress-Free Productivity. Revised ed. New York : Penguin Books, 2015. 352 p.

2. Covey S. R. The 7 Habits of Highly Effective People. 30th anniversary ed. New York : Simon & Schuster, 2020. 464 p.

3. Cirillo F. The Pomodoro Technique: The Acclaimed Time-Management System That Has Transformed How We Work. New York : Currency, 2018. 160 p.

4. Tracy B. Eat That Frog! 21 Great Ways to Stop Procrastinating and Get More Done in Less Time. 3rd ed. Oakland : Berrett-Koehler Publishers, 2017. 144 p.

5. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 216 p.

6. RescueTime. The State of Work Life Balance : report. URL: https://blog.rescuetime.com/work-life-balance-study-2019/ (Last accessed: 12.01.2026).

7. Worksection : офіційний сайт. URL: https://worksection.com/ua/ (дата звернення: 26.02.2026).

8. Google Calendar : Official Website. URL: https://calendar.google.com/ (дата звернення: 26.02.2026).

9. TickTick : Official Website. URL: https://ticktick.com/ (Last accessed: 26.02.2026).

10. Notion : Official Website. URL: https://www.notion.so/ (Last accessed: 26.02.2026).

11. Any.do : Official Website. URL: https://www.any.do/ (Last accessed: 26.02.2026).

12. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2020. 1136 p.

13. Skiena S. The Algorithm Design Manual. 3rd ed. New York : Springer, 2020. 793 p.

14. Kleitman K. P., Weitzer S. L. Productivity Peaks and Ultradian Rhythms. Journal of Circadian Rhythms. 2019. Vol. 17, No. 1. P. 1–12.

15. Stauffer M. Laravel: Up & Running. 3rd ed. Sebastopol : O'Reilly Media, 2023. 568 p.

16. Laravel Documentation. Laravel : website. URL: https://laravel.com/docs/10.x (Last accessed: 23.03.2026).

17. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

18. MySQL 8.4 Reference Manual. MySQL : website. URL: https://dev.mysql.com/doc/refman/8.4/en/ (Last accessed: 23.03.2026).

19. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Addison-Wesley, 2020. 1024 p.

20. PHP 8.4 Manual. PHP : website. URL: https://www.php.net/manual/en/ (Last accessed: 25.03.2026).

21. Eloquent ORM Documentation. Laravel : website. URL: https://laravel.com/docs/10.x/eloquent (Last accessed: 25.03.2026).

22. McConnell S. Code Complete: A Practical Handbook of Software Construction. 2nd ed. Redmond : Microsoft Press, 2004. 960 p.

23. OWASP Top Ten 2021. OWASP Foundation : website. URL: https://owasp.org/www-project-top-ten/ (Last accessed: 28.03.2026).

24. Bootstrap 5 Documentation. Bootstrap : website. URL: https://getbootstrap.com/docs/5.3/ (Last accessed: 28.03.2026).

25. Blade Templates Documentation. Laravel : website. URL: https://laravel.com/docs/10.x/blade (Last accessed: 28.03.2026).

26. MDN Web Docs : CSS Custom Properties (Variables). Mozilla : website. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (Last accessed: 08.04.2026).

27. Marcotte E. Responsive Web Design. 2nd ed. New York : A Book Apart, 2014. 153 p.

28. Frain B. Responsive Web Design with HTML5 and CSS. 4th ed. Birmingham : Packt Publishing, 2022. 408 p.

29. Chart.js Documentation. Chart.js : website. URL: https://www.chartjs.org/docs/latest/ (Last accessed: 16.04.2026).

30. Nielsen J. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group : website. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (Last accessed: 27.04.2026).

ДОДАТОК А

Лістинг коду

А.1 Blade-компоненти

resources\views\layouts\app.blade.php:

```
<!DOCTYPE html>
<html lang="uk" data-server-theme="{{ Auth::check() ? (Auth::user()->theme ?? 'light') :
'light' }}">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-scale=1.0,  viewport-
fit=cover">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>@yield('title', 'TimeManager')</title>

    <script src="{{ asset('js/theme.js') }}"></script>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link   href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.min.css" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;800&display=
swap" rel="stylesheet">
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('css/dark-overrides.css') }}" rel="stylesheet">

    @stack('styles')
</head>
<body>

<aside class="sidebar" id="sidebar">
    <div class="sidebar-header">
        <a href="{{ route('dashboard') }}" class="sidebar-brand">
            <div class="brand-icon"><i class="bi bi-clock-history"></i></div>
            <div>
                <div class="brand-text">TimeManager</div>
                <span class="brand-sub">Планувальник задач</span>
            </div>
        </a>
    </div>
    <nav class="sidebar-nav">
        <div class="nav-group-label">Головне</div>
        <a href="{{ route('dashboard') }}"
           class="sidebar-link {{ request()->routeIs('dashboard') ? 'active' : '' }}">
            <i class="bi bi-grid-fill"></i> Дашборд
        </a>
        <a href="{{ route('schedule') }}"
           class="sidebar-link  {{  request()->routeIs('schedule')  &&  !request()-
>routeIs('schedule.week') ? 'active' : '' }}">
            <i class="bi bi-calendar-day-fill"></i> Розклад на день
        </a>
        <a href="{{ route('schedule.week') }}"
           class="sidebar-link {{ request()->routeIs('schedule.week') ? 'active' : '' }}">
            <i class="bi bi-calendar-week-fill"></i> Тижневий вигляд
```

```
        </a>
        <div class="nav-group-label">Аналіз та поради</div>
        <a href="{{ route('analytics.index') }}"
           class="sidebar-link {{ request()->routeIs('analytics*') ? 'active' : '' }}">
            <i class="bi bi-bar-chart-fill"></i> Аналітика
        </a>
        <a href="{{ route('tips.index') }}"
           class="sidebar-link {{ request()->routeIs('tips*') ? 'active' : '' }}">
            <i class="bi bi-lightbulb-fill"></i> Поради
        </a>
        <div class="nav-group-label">Організація</div>
        <a href="{{ route('recurring.index') }}"
           class="sidebar-link {{ request()->routeIs('recurring*') ? 'active' : '' }}">
            <i class="bi bi-arrow-repeat"></i> Повторювані задачі
        </a>
        <a href="{{ route('categories.index') }}"
           class="sidebar-link {{ request()->routeIs('categories*') ? 'active' : '' }}">
            <i class="bi bi-tags-fill"></i> Категорії
        </a>
        <div class="nav-group-label">Акаунт</div>
        <a href="{{ route('profile.edit') }}"
           class="sidebar-link {{ request()->routeIs('profile*') ? 'active' : '' }}">
            <i class="bi bi-person-fill"></i> Мій профіль
        </a>
        <a href="{{ route('settings.index') }}"
           class="sidebar-link {{ request()->routeIs('settings*') ? 'active' : '' }}">
            <i class="bi bi-gear-fill"></i> Налаштування
        </a>
        <form action="{{ route('logout') }}" method="POST" class="logout-form">
            @csrf
            <button type="submit" class="sidebar-link">
                <i class="bi bi-box-arrow-left"></i> Вийти
            </button>
        </form>
    </nav>
    @auth
    <div class="sidebar-footer">
        <div class="user-block">
            <div class="sidebar-avatar">
                @if(Auth::user()->avatar && file_exists(public_path(Auth::user()->avatar)))
                    <img  src="{{  asset(Auth::user()->avatar)  }}?v={{
filemtime(public_path(Auth::user()->avatar)) }}" alt="Фото">
                @else
                    <i class="bi bi-person-fill"></i>
                @endif
            </div>
            <div class="user-block-info">
                <div class="user-name">{{ Auth::user()->name }}</div>
                <div class="user-email">{{ Auth::user()->email }}</div>
            </div>
            <a href="{{ route('profile.edit') }}" class="user-block-edit" title="Профіль">
                <i class="bi bi-pencil-square"></i>
            </a>
        </div>
    </div>
    @endauth
</aside>
{{-- Backdrop for mobile sidebar --}}
<div class="sidebar-backdrop" id="sidebarBackdrop"></div>

<div class="main-wrap">
```

```
    <header class="topbar">
        <div class="topbar-left">
            <button class="sidebar-toggle-btn" id="sidebarToggle" aria-label="Меню">
                <i class="bi bi-list btn-list-icon"></i>
            </button>
            <div class="topbar-text-min">
                <div class="topbar-title">@yield('page-title', 'TimeManager')</div>
                @php
                    $ukDaysL           =
['неділя','понеділок','вівторок','середа','четвер','п\'ятниця','субота'];
                    $ukMonthsL     =
['січня','лютого','березня','квітня','травня','червня','липня','серпня','вересня','жовтня'
,'листопада','грудня'];
                    $nowL      = \Carbon\Carbon::now();
                @endphp
                <div class="topbar-date">
                    {{ ucfirst($ukDaysL[$nowL->dayOfWeek]) }}, {{ $nowL->day }} {{
$ukMonthsL[$nowL->month - 1] }} {{ $nowL->year }}
                </div>
            </div>
        </div>
        <div class="topbar-right">
            <button class="theme-toggle" onclick="toggleTheme()" title="Перемкнути тему">
                <i class="bi {{ (Auth::check() && Auth::user()->theme === 'dark') ? 'bi-
sun-fill' : 'bi-moon-fill' }}"></i>
            </button>
            @yield('topbar-actions')
        </div>
    </header>

    <main class="page-body">
        @if(session('success'))
        <div class="alert alert-success alert-dismissible mb-4">
            <i class="bi bi-check-circle-fill"></i>
            {{ session('success') }}
            <button type="button" class="btn-close ms-auto alert-close-sm" data-bs-
dismiss="alert"></button>
        </div>
        @endif
        @if(session('error'))
        <div class="alert alert-danger alert-dismissible mb-4">
            <i class="bi bi-exclamation-circle-fill"></i>
            {{ session('error') }}
            <button type="button" class="btn-close ms-auto alert-close-sm" data-bs-
dismiss="alert"></button>
        </div>
        @endif

        @yield('content')
    </main>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></scrip
t>
<script src="{{ asset('js/app.js') }}"></script>
@stack('scripts')
</body>
</html>
```

A.2 Models

app\Models\Category.php:

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Category extends Model
{
    use HasFactory;
    protected $fillable = ['user_id', 'name', 'color'];
    public function user()
    {
        return $this->belongsTo(User::class);
    }
    public function tasks()
    {
        return $this->hasMany(Task::class);
    }
}
```

A.3 Controllers

app\Http\Controllers\SettingsController.php:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class SettingsController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        return view('settings.index', ['user' => Auth::user()]);
    }

    public function updateTheme(Request $request)
    {
        $request->validate([
            'theme' => 'required|in:light,dark',
        ]);

        Auth::user()->update(['theme' => $request->theme]);

        if ($request->wantsJson() || $request->ajax()) {
            return response()->json(['theme' => $request->theme]);
        }

        return back()->with('success', 'Тему змінено на ' . ($request->theme === 'dark' ?
'темну' : 'світлу') . '.');
    }
}
```

A.4 Routes

routes\web.php:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\TaskController;
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\RecurringTaskController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\AnalyticsController;
use App\Http\Controllers\AdvisorController;
use App\Http\Controllers\SettingsController;
Route::get('/login',            [AuthController::class,     'showLogin'])->name('login')-
>middleware('guest');
Route::post('/login',   [AuthController::class, 'login'])->middleware('guest');
Route::get('/register',     [AuthController::class,     'showRegister'])->name('register')-
>middleware('guest');
Route::post('/register',[AuthController::class, 'register'])->middleware('guest');
Route::post('/logout',  [AuthController::class, 'logout'])->name('logout');
Route::middleware('auth')->group(function () {
    Route::get('/', [DashboardController::class, 'index'])->name('dashboard');
    Route::get('/schedule',      [TaskController::class, 'schedule'])->name('schedule');
    Route::get('/schedule/week', [TaskController::class, 'week'])->name('schedule.week');

    Route::post('/tasks',                        [TaskController::class,  'store'])-
>name('tasks.store');
    Route::put('/tasks/{task}',                 [TaskController::class,  'update'])-
>name('tasks.update');
    Route::patch('/tasks/{task}/status',  [TaskController::class,  'updateStatus'])-
>name('tasks.status');
    Route::delete('/tasks/{task}',             [TaskController::class,  'destroy'])-
>name('tasks.destroy');
    Route::get('/api/tasks',                 [TaskController::class,  'getTasksJson'])-
>name('tasks.json');
    Route::get('/recurring',                            [RecurringTaskController::class,
'index'])->name('recurring.index');
    Route::post('/recurring',                           [RecurringTaskController::class,
'store'])->name('recurring.store');
    Route::put('/recurring/{recurringTask}',           [RecurringTaskController::class,
'update'])->name('recurring.update');
    Route::patch('/recurring/{recurringTask}/toggle',  [RecurringTaskController::class,
'toggleActive'])->name('recurring.toggle');
    Route::post('/recurring/{recurringTask}/done',     [RecurringTaskController::class,
'markDone'])->name('recurring.done');
    Route::delete('/recurring/{recurringTask}',        [RecurringTaskController::class,
'destroy'])->name('recurring.destroy');
    Route::get('/api/recurring/{recurringTask}',   function(\App\Models\RecurringTask
$recurringTask) {
        if ($recurringTask->user_id !== auth()->id()) abort(403);
        return response()->json($recurringTask);
    })->name('api.recurring.show');
    Route::get('/categories',                   [CategoryController::class,  'index'])-
>name('categories.index');
    Route::post('/categories',                  [CategoryController::class,  'store'])-
>name('categories.store');
    Route::put('/categories/{category}',      [CategoryController::class,  'update'])-
>name('categories.update');
```

```
    Route::delete('/categories/{category}',  [CategoryController::class,  'destroy'])-
>name('categories.destroy');
    Route::get('/analytics',      [AnalyticsController::class,      'index'])-
>name('analytics.index');
    Route::get('/tips',          [AdvisorController::class, 'index'])->name('tips.index');
    Route::post('/tips/plan',    [AdvisorController::class, 'plan'])->name('tips.plan');
    Route::post('/tips/accept',      [AdvisorController::class,     'accept'])-
>name('tips.accept');
    Route::get('/profile',                    [ProfileController::class,   'edit'])-
>name('profile.edit');
    Route::put('/profile',                   [ProfileController::class,   'update'])-
>name('profile.update');
    Route::put('/profile/password',   [ProfileController::class,  'updatePassword'])-
>name('profile.password');
    Route::post('/profile/avatar',      [ProfileController::class,   'uploadAvatar'])-
>name('profile.avatar');
    Route::delete('/profile/avatar',   [ProfileController::class,   'removeAvatar'])-
>name('profile.avatar.remove');
    Route::delete('/profile',              [ProfileController::class,   'destroy'])-
>name('profile.destroy');
    Route::get('/settings',                 [SettingsController::class,   'index'])-
>name('settings.index');
    Route::post('/settings/theme',    [SettingsController::class,   'updateTheme'])-
>name('settings.theme');
});
```

A.5 Migrations

database\migrations\000003_create_categories_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration

{
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->string('name');
            $table->string('color', 7)->default('#4f46e5');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('categories');
    }

};
```

A.6 Seeders

database\seeders\DatabaseSeeder.php:

```
class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $user = User::create([
            'name'     => 'Демо Користувач',
            'email'    => 'demo@timemanager.com',
            'password' => Hash::make('password'),
            'theme'    => 'light',
        ]);
$cats = [
    'work'    => Category::create(['user_id' => $user->id, 'name' => 'Робота',    'color'
=> '#4f46e5']),
    'home'    => Category::create(['user_id' => $user->id, 'name' => 'Особисте',  'color'
=> '#10b981']),
    'study'   => Category::create(['user_id' => $user->id, 'name' => 'Навчання',  'color'
=> '#f59e0b']),
    'health'  => Category::create(['user_id' => $user->id, 'name' => 'Здоров\'я', 'color'
=> '#ef4444']),
    'fun'     => Category::create(['user_id' => $user->id, 'name' => 'Розваги',   'color'
=> '#8b5cf6']),
    'project' => Category::create(['user_id' => $user->id, 'name' => 'Проєкти',   'color'
=> '#06b6d4']),
];
        $morningExercise = RecurringTask::create([
            'user_id'            => $user->id,
            'category_id'        => $cats['health']->id,
            'title'              => 'Ранкова зарядка',
            'description'        => 'Щоденна 20-хвилинна розминка для бадьорості',
            'default_start_time' => '07:00',
            'default_end_time'   => '07:20',
            'recurrence_type'    => 'daily',
            'priority'           => 'medium',
            'color'              => '#ef4444',
            'is_active'          => true,
            'start_date'         => Carbon::today()->subDays(60)->toDateString(),
        ]);
        $standUp = RecurringTask::create([
            'user_id'            => $user->id,
            'category_id'        => $cats['work']->id,
            'title'              => 'Stand-up зустріч команди',
            'description'        => 'Щоденна синхронізація команди',
            'default_start_time' => '09:30',
            'default_end_time'   => '09:45',
            'recurrence_type'    => 'weekly',
            'recurrence_days'    => '1,2,3,4,5',
            'priority'           => 'high',
            'color'              => '#4f46e5',
            'is_active'          => true,
            'start_date'         => Carbon::today()->subDays(30)->toDateString(),
        ]);
        RecurringTask::create([
            'user_id'            => $user->id,
            'category_id'        => $cats['health']->id,
            'title'              => 'Медитація',
            'description'        => 'Тимчасово вимкнено — повернутись пізніше',
```

```
            'default_start_time' => '21:00',
            'default_end_time'   => '21:15',
            'recurrence_type'    => 'daily',
            'priority'           => 'low',
            'color'              => '#10b981',
            'is_active'          => false,
            'start_date'         => Carbon::today()->subDays(20)->toDateString(),
        ]);
        for ($i = 1; $i <= 30; $i++) {
            if ($i % 6 === 0) continue;
            RecurringDoneLog::create([
                'recurring_task_id' => $morningExercise->id,
                'user_id'           => $user->id,
                'done_date'         => Carbon::today()->subDays($i)->toDateString(),
            ]);
        }
        $lastWeekTasks = [
            ['day' => -7, 'cat' => 'work',    'title' => 'Підготовка квартального звіту',
's' => '10:00', 'e' => '12:00', 'p' => 'high'],
            ['day' => -7, 'cat' => 'work',    'title' => 'Презентація для клієнта',
 's' => '14:00', 'e' => '15:30', 'p' => 'high'],
            ['day' => -4, 'cat' => 'home',    'title' => 'Покупки в магазині',
's' => '17:00', 'e' => '18:00', 'p' => 'low'],
            ['day' => -3, 'cat' => 'family',  'title' => 'Зустріч з батьками',
's' => '15:00', 'e' => '17:00', 'p' => 'medium'],
            ['day' => -1, 'cat' => 'fun',     'title' => 'Кіно з друзями',
's' => '20:00', 'e' => '22:30', 'p' => 'low'],
        ];
        foreach ($lastWeekTasks as $t) {
            Task::create([
                'user_id'        => $user->id,
                'category_id'    => $cats[$t['cat']]->id,
                'title'          => $t['title'],
                'scheduled_date' => Carbon::today()->addDays($t['day'])->toDateString(),
                'start_time'     => $t['s'],
                'end_time'       => $t['e'],
                'priority'       => $t['p'],
                'status'         => 'completed',
                'color'          => $cats[$t['cat']]->color,
            ]);
        }
        Task::create([
            'user_id'        => $user->id,
            'category_id'    => $cats['work']->id,
            'title'          => 'Здати фінансовий звіт (прострочено)',
            'description'    => 'Здати в бухгалтерію',
            'scheduled_date' => Carbon::today()->subDays(3)->toDateString(),
            'start_time'     => '14:00',
            'end_time'       => '15:00',
            'priority'       => 'high',
            'status'         => 'pending',
            'color'          => '#4f46e5',
        ]);
        Task::create([
            'user_id'        => $user->id,
            'category_id'    => $cats['study']->id,
            'title'          => 'Написати есе з курсу',
            'scheduled_date' => Carbon::today()->subDays(5)->toDateString(),
            'start_time'     => '20:00',
            'end_time'       => '22:00',
            'priority'       => 'medium',
```

```
            'status'         => 'pending',
            'color'          => '#f59e0b',
        ]);
        $todayTasks = [
            ['cat' => 'work',    'title' => 'Перевірити пошту',                's' =>
'08:30', 'e' => '09:00', 'p' => 'low',    'st' => 'completed'],
            ['cat' => 'home',    'title' => 'Обід',                            's' =>
'13:00', 'e' => '14:00', 'p' => 'low',    'st' => 'pending'],
            ['cat' => 'study',   'title' => 'Курс з Laravel: тема Routing',   's' =>
'15:00', 'e' => '16:30', 'p' => 'medium', 'st' => 'pending'],
            ['cat' => 'health',  'title' => 'Прогулянка в парку',              's' =>
'18:00', 'e' => '18:45', 'p' => 'low',    'st' => 'pending'],
            ['cat' => 'fun',     'title' => 'Серіал перед сном',               's' =>
'21:30', 'e' => '22:30', 'p' => 'low',    'st' => 'pending'],
        ];
        foreach ($todayTasks as $t) {
            Task::create([
                'user_id'        => $user->id,
                'category_id'    => $cats[$t['cat']]->id,
                'title'          => $t['title'],
                'scheduled_date' => Carbon::today()->toDateString(),
                'start_time'     => $t['s'],
                'end_time'       => $t['e'],
                'priority'       => $t['p'],
                'status'         => $t['st'],
                'color'          => $cats[$t['cat']]->color,
            ]);
        }
        $upcomingTasks = [
            ['day' => 1,  'cat' => 'health',  'title' => 'Похід до стоматолога',
's' => '14:00', 'e' => '15:00', 'p' => 'medium'],
            ['day' => 2,  'cat' => 'project', 'title' => 'Code review нового функціоналу',
's' => '11:00', 'e' => '12:30', 'p' => 'high'],
            ['day' => 2,  'cat' => 'study',   'title' => 'Вебінар з UX/UI',
 's' => '19:00', 'e' => '20:30', 'p' => 'medium'],
            ['day' => 3,  'cat' => 'family',  'title' => 'День народження мами',
's' => '18:00', 'e' => '21:00', 'p' => 'high'],
            ['day' => 4,  'cat' => 'work',    'title' => 'Демонстрація прототипу',
's' => '15:00', 'e' => '16:00', 'p' => 'high'],
            ['day' => 5,  'cat' => 'fun',     'title' => 'Концерт улюбленого виконавця',
's' => '20:00', 'e' => '23:00', 'p' => 'low'],
            ['day' => 6,  'cat' => 'home',    'title' => 'Прибирання вдома',
's' => '11:00', 'e' => '13:00', 'p' => 'medium'],
            ['day' => 7,  'cat' => 'health',  'title' => 'Біг у парку',
 's' => '08:00', 'e' => '09:00', 'p' => 'low'],
        ];
        foreach ($upcomingTasks as $t) {
            Task::create([
                'user_id'        => $user->id,
                'category_id'    => $cats[$t['cat']]->id,
                'title'          => $t['title'],
                'scheduled_date' => Carbon::today()->addDays($t['day'])->toDateString(),
                'start_time'     => $t['s'],
                'end_time'       => $t['e'],
                'priority'       => $t['p'],
                'status'         => 'pending',
                'color'          => $cats[$t['cat']]->color,
            ]);
        }
    }
}
```

ДОДАТОК Б

Результати працездатності системи

У додатку Б наведено скриншоти працездатності розробленої системи «TimeManager» у різних режимах використання (рис. Б.1–Б.14).

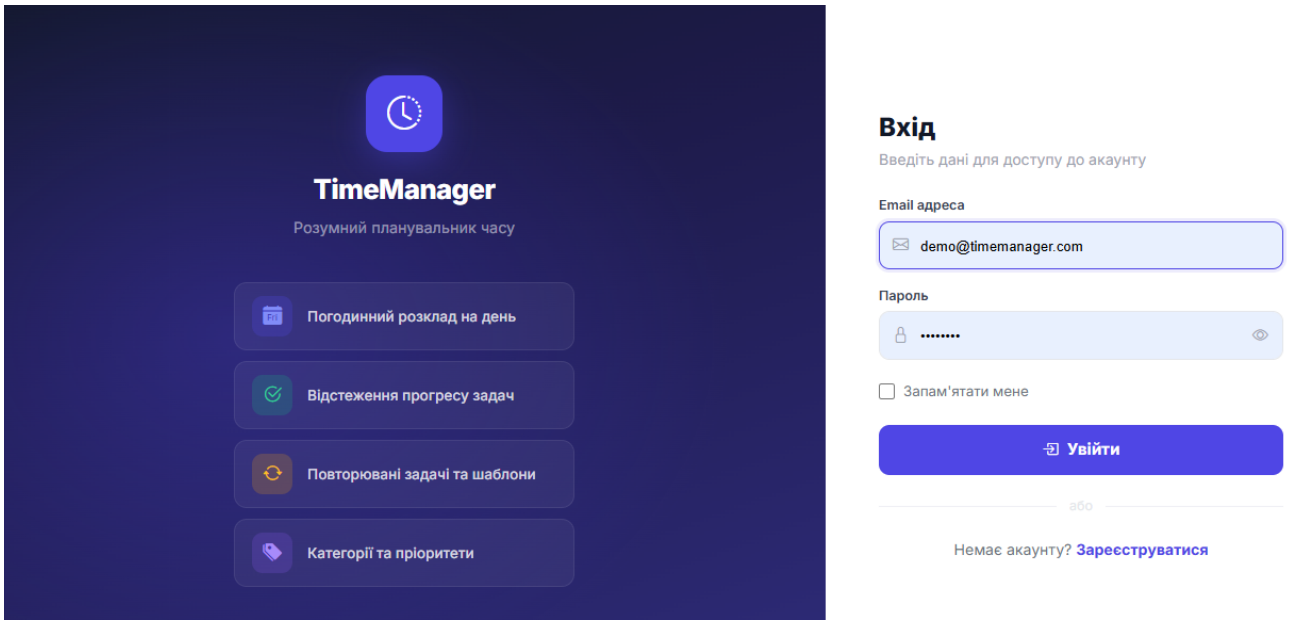

Рисунок Б.1 – Сторінка реєстрації

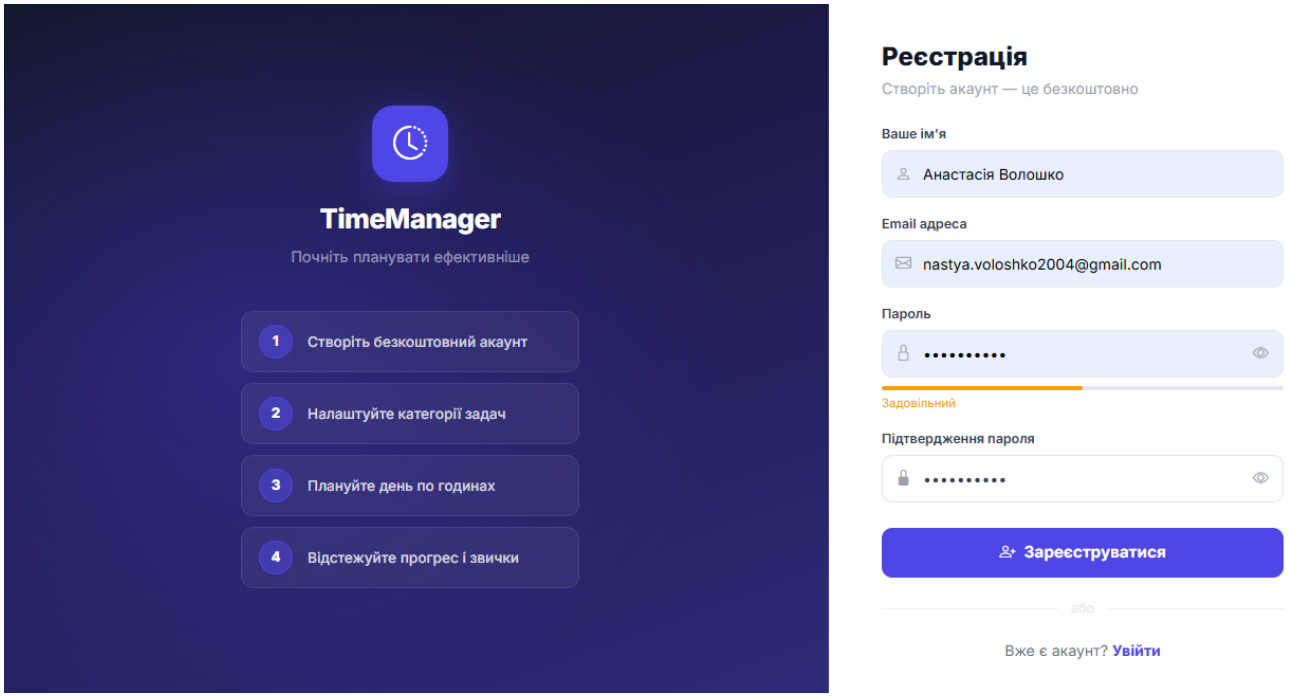

Рисунок Б.2 – Сторінка входу в систему

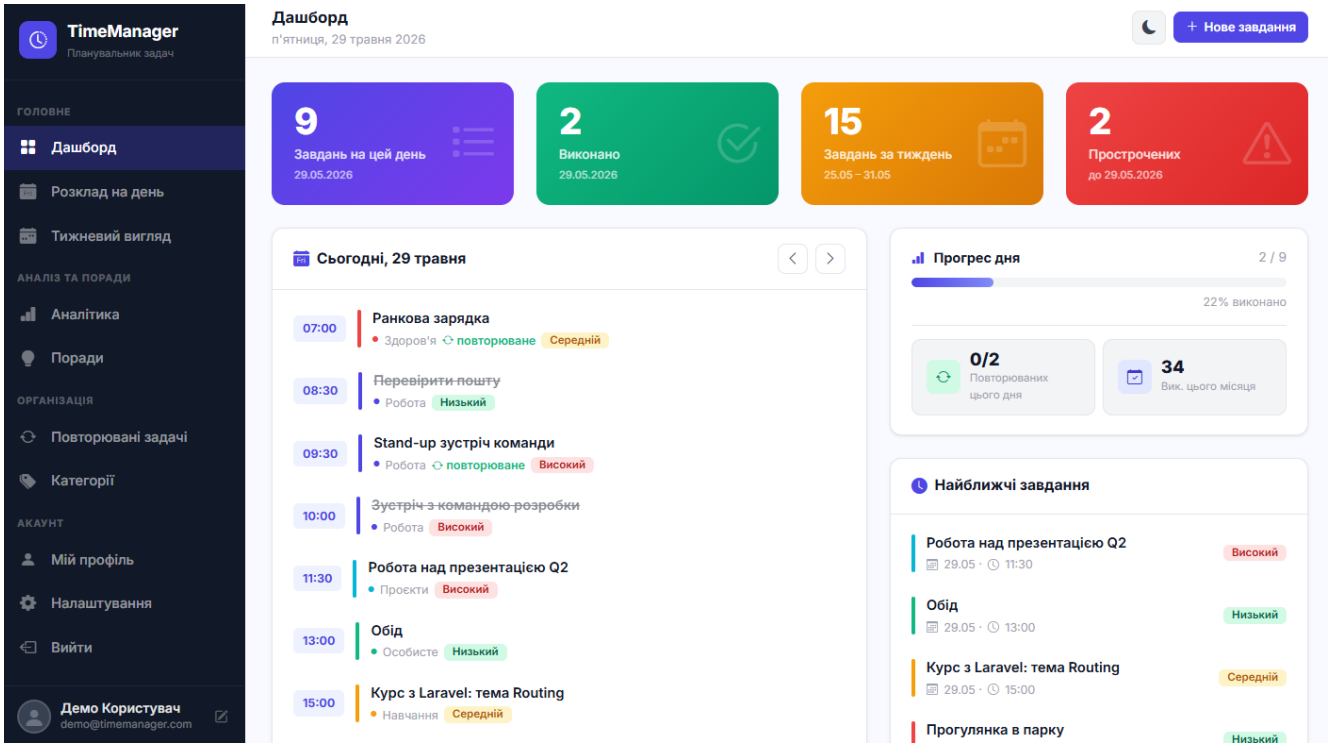

Рисунок Б.3 – Головна сторінка (дашборд) зі статистичними картками, розкладом на день та найближчими задачами

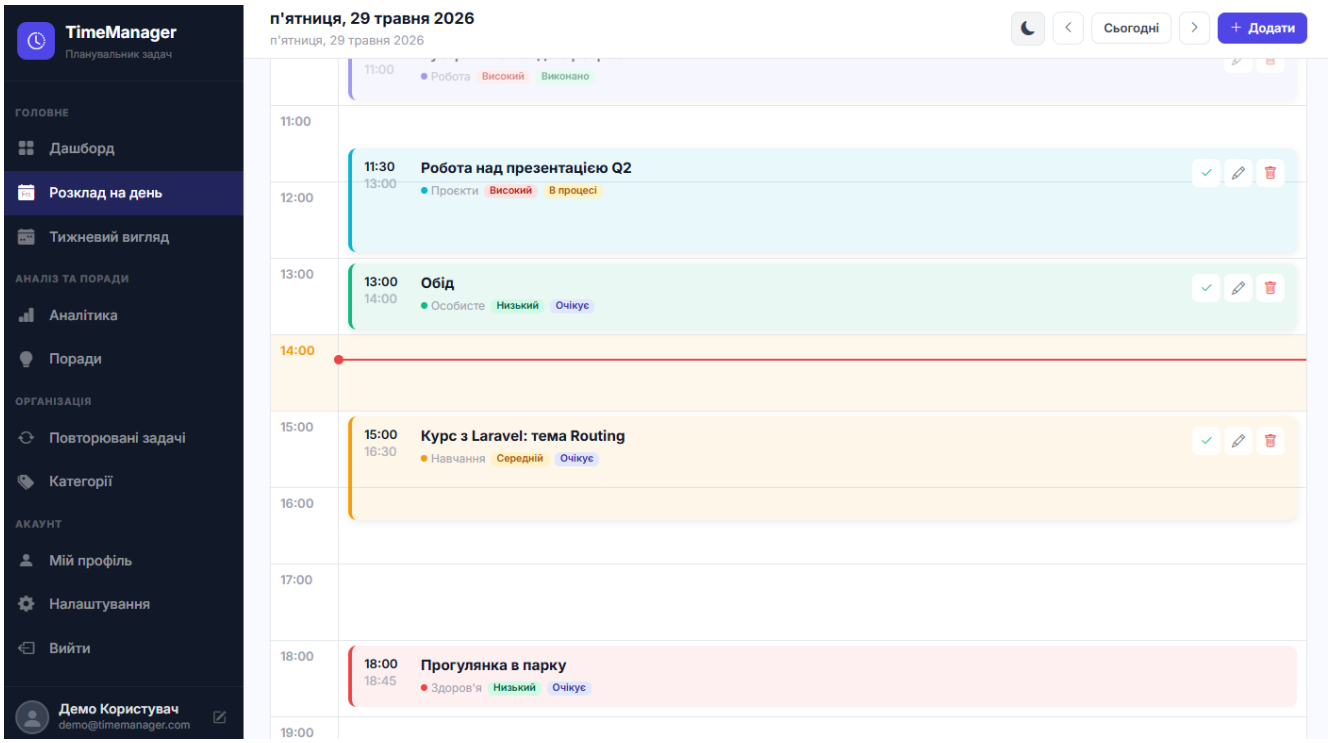

Рисунок Б.4 – Погодинний розклад на день зі задачами різних типів (одногодинні, багатогодинні, повторювані)

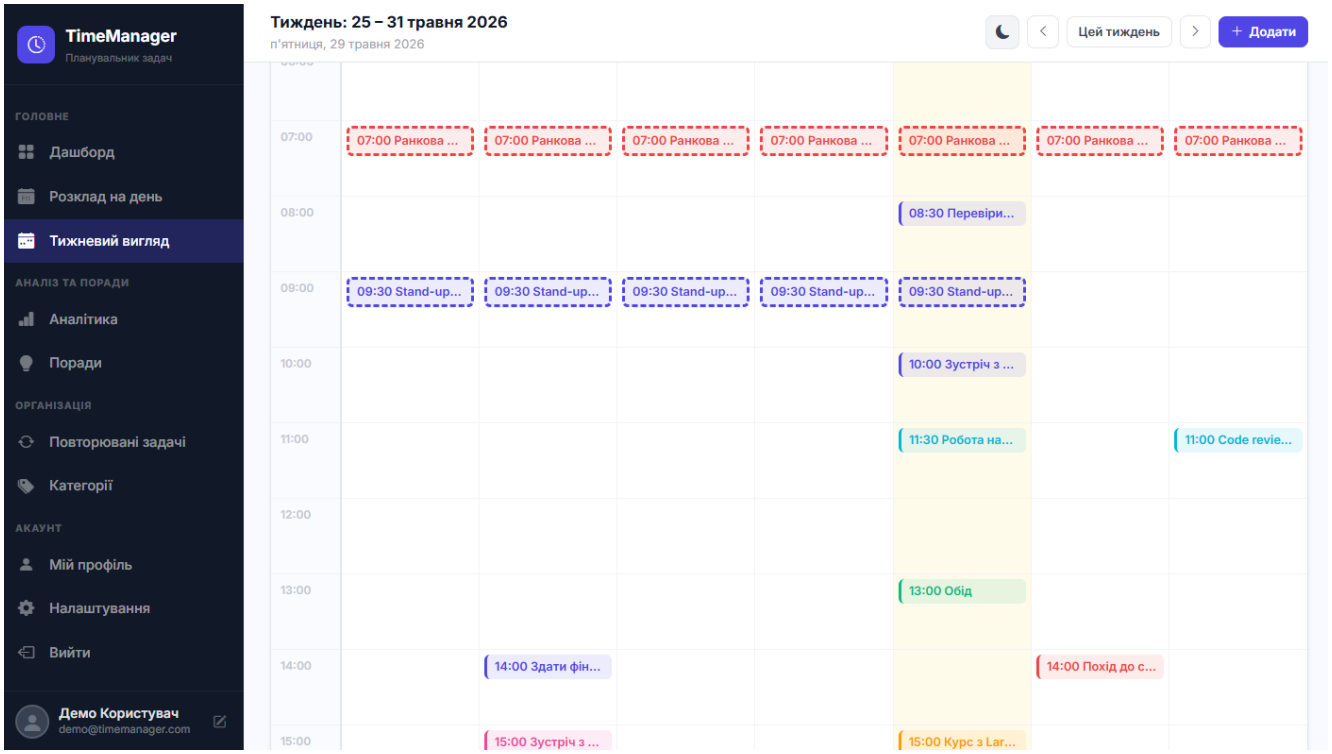

Рисунок Б.5 – Тижневий розклад у вигляді таблиці 7 днів × 24 години

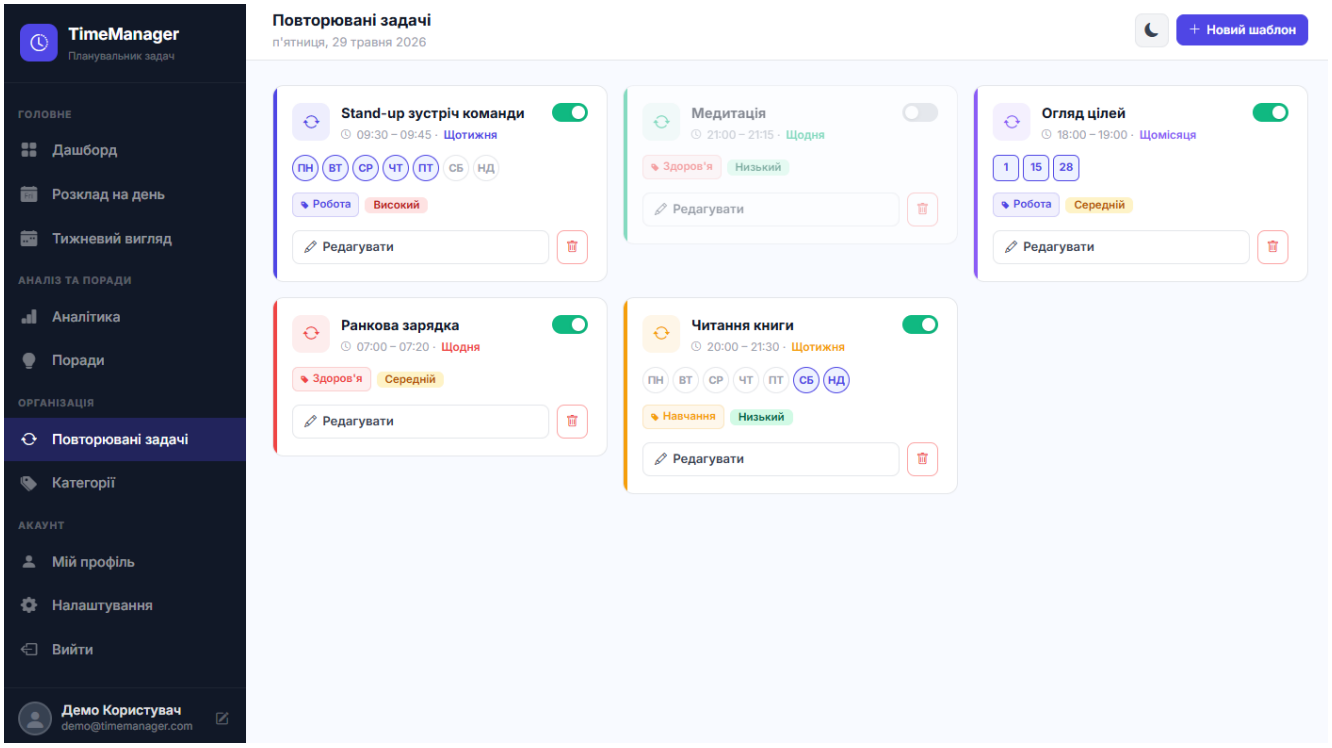

Рисунок Б.6 – Сторінка повторюваних задач з картками шаблонів

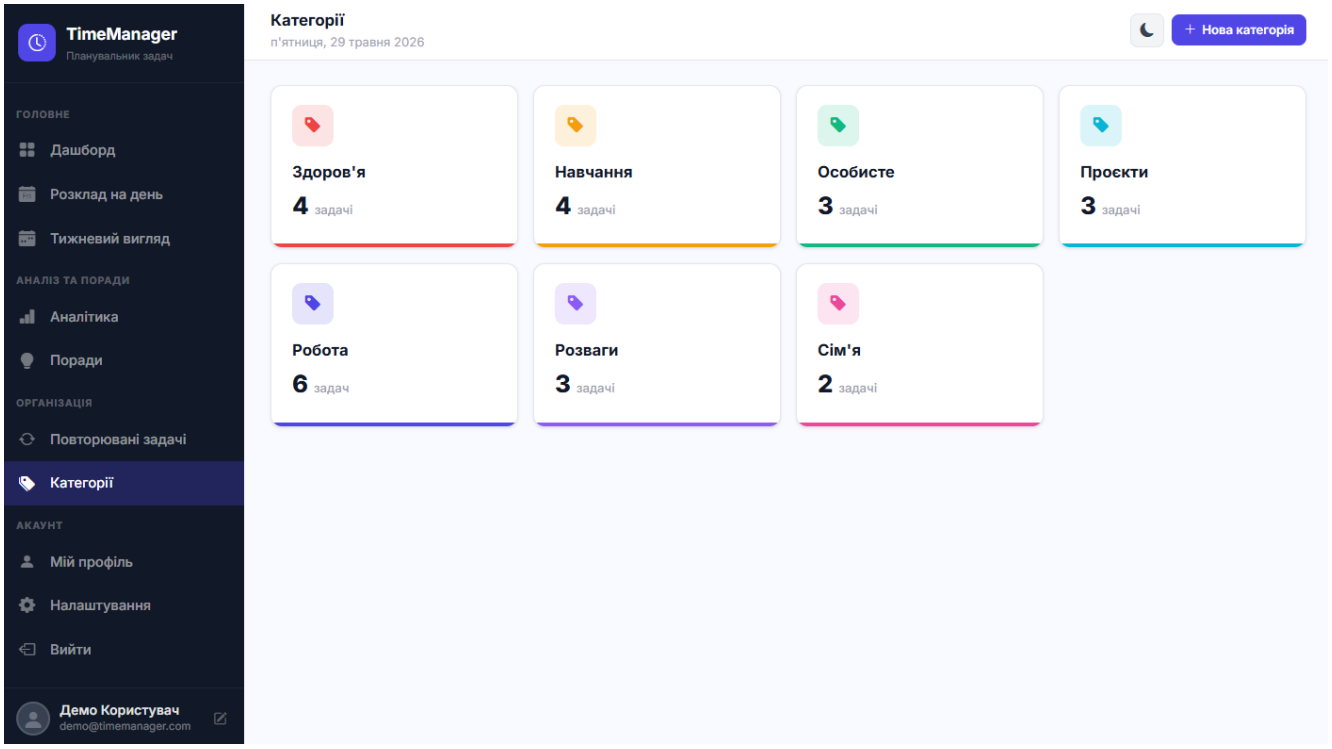

Рисунок Б.7 – Сторінка категорій з лічильниками задач

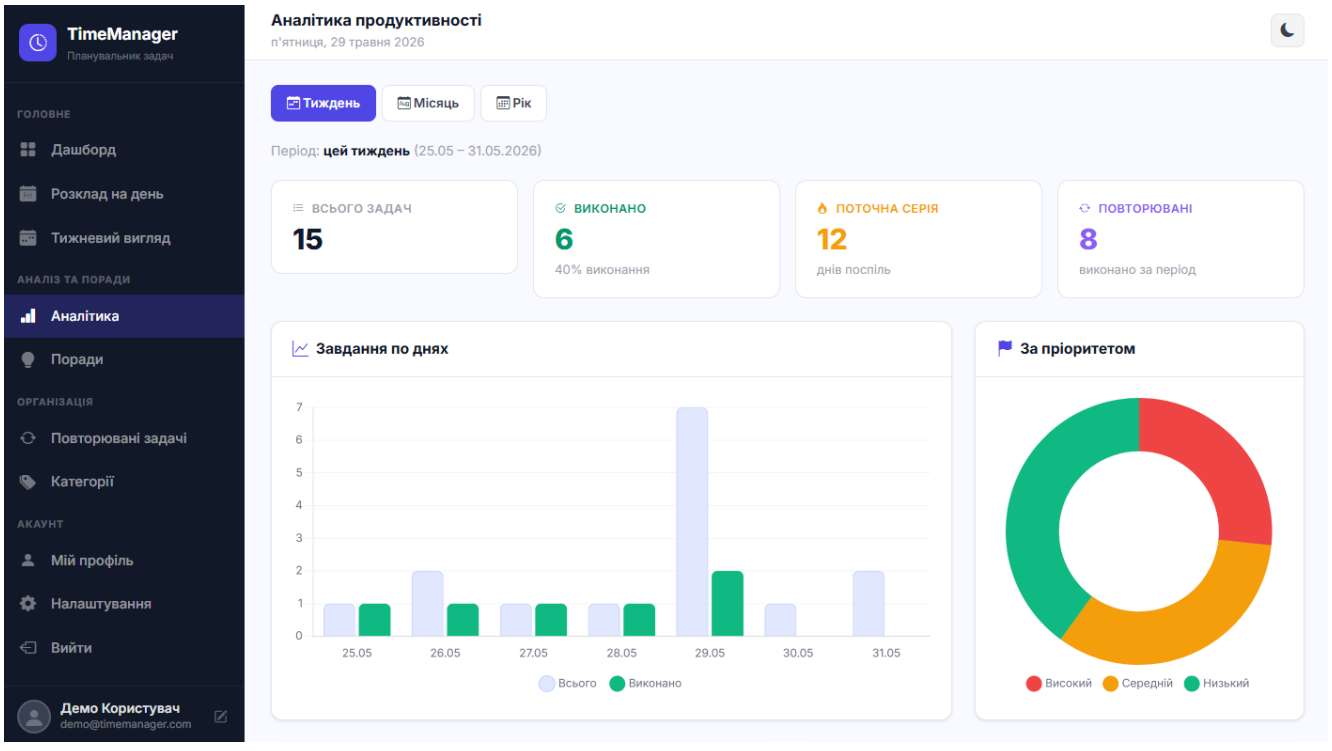

Рисунок Б.8 – Сторінка аналітики з Chart.js діаграмами та статистикою

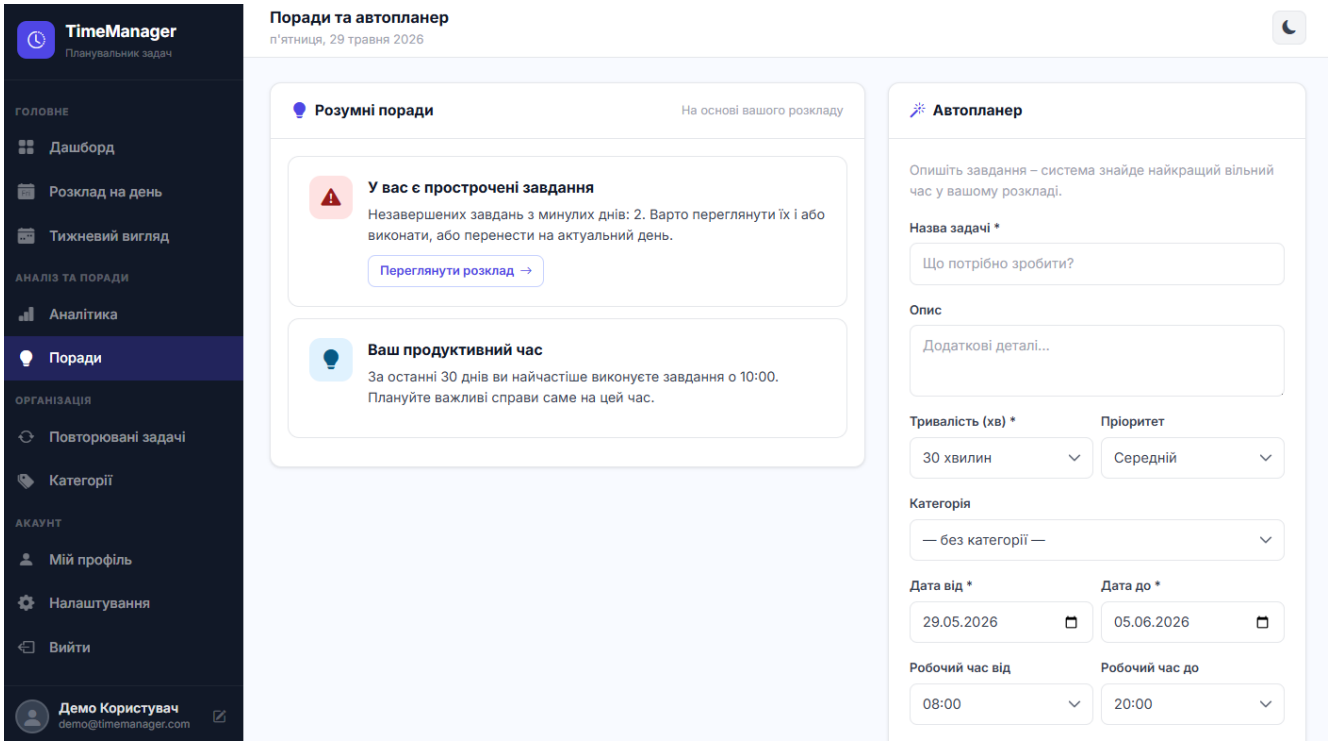

Рисунок Б.9 – Сторінка порад з інтелектуальним радником та формою автопланера

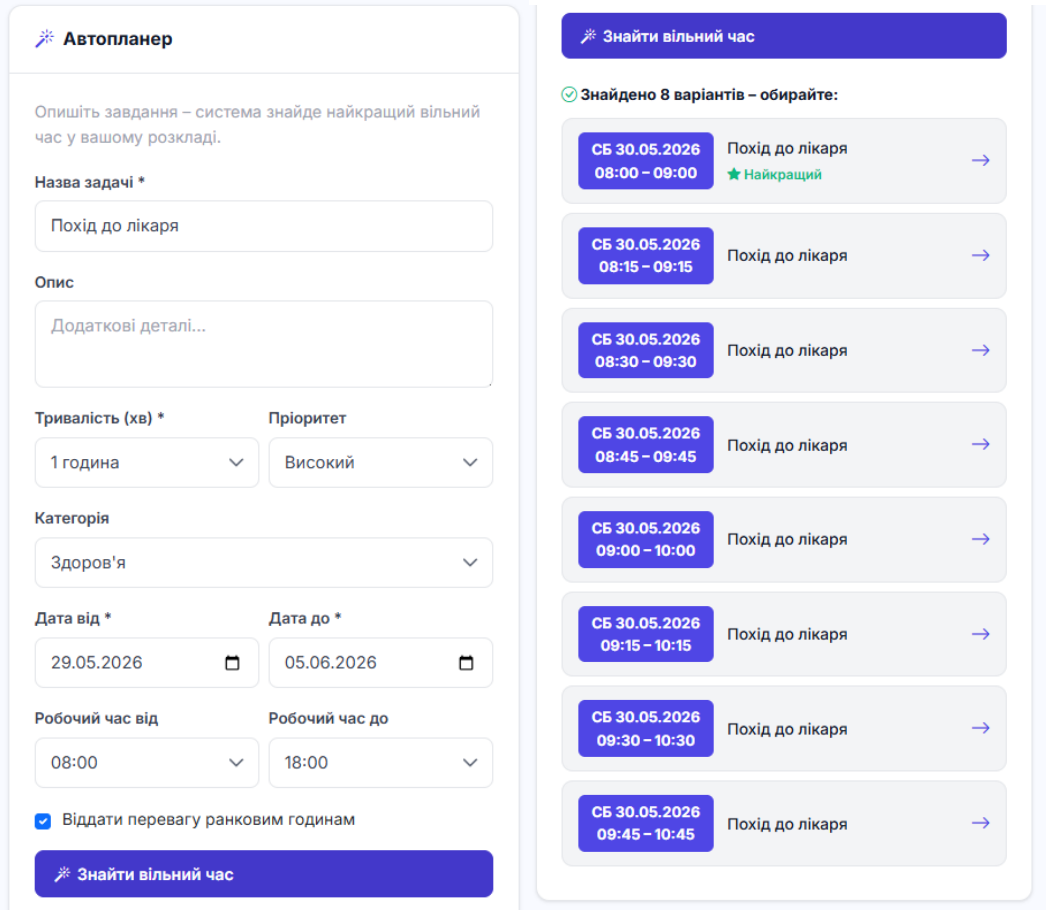

Рисунок Б.10 – Результат роботи автопланера (список знайдених вільних слотів)

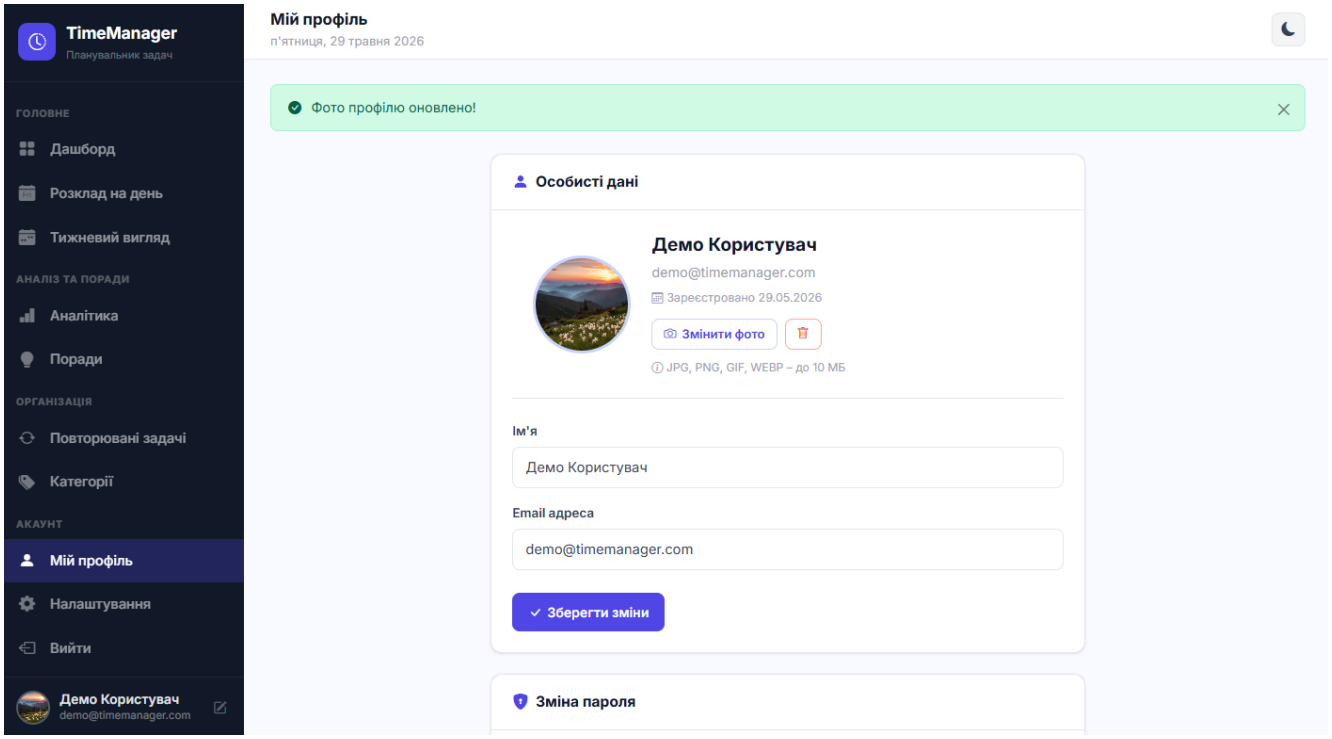

Рисунок Б.11 – Сторінка профілю користувача з завантаженням аватара

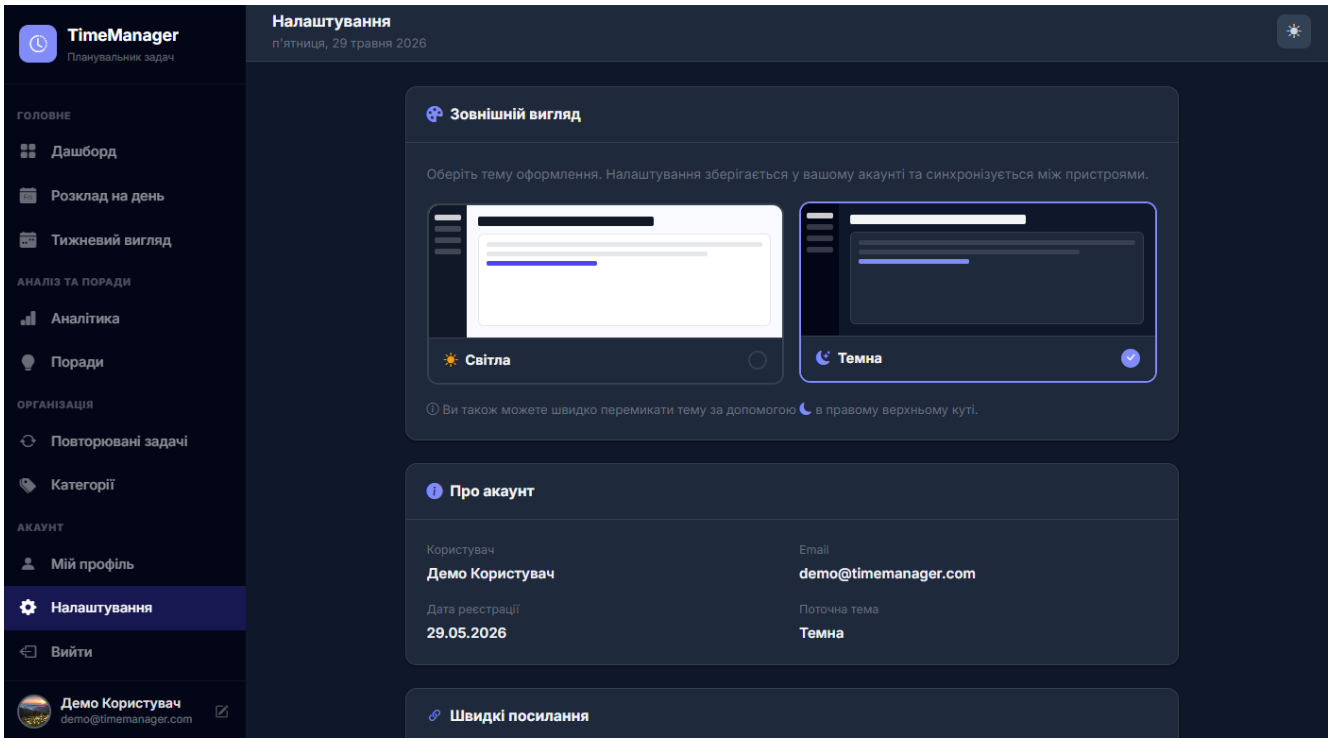

Рисунок Б.12 – Сторінка налаштувань з вибором теми оформлення

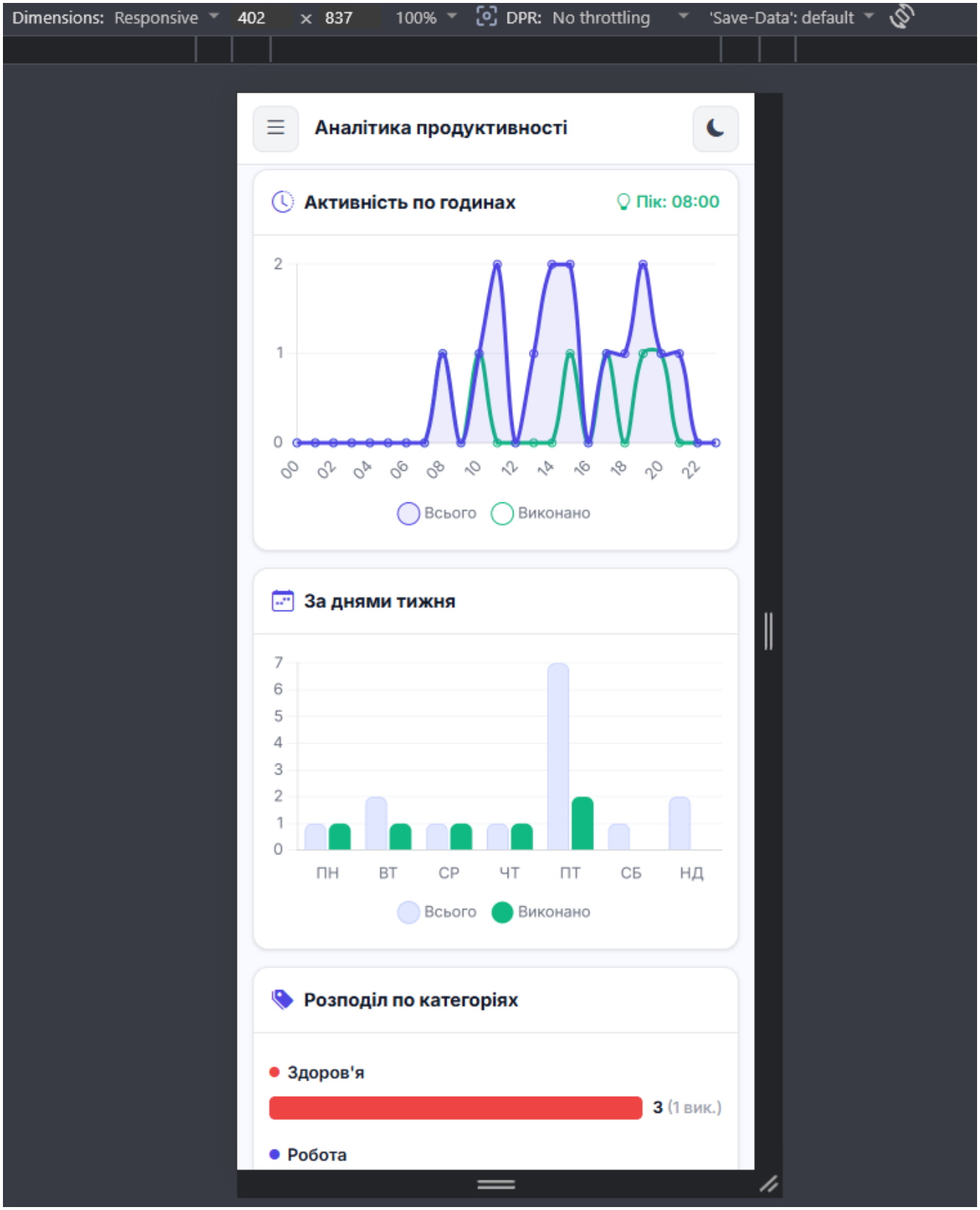

Рисунок Б.13 – Адаптивне відображення на мобільному пристрої (sidebar overlay)

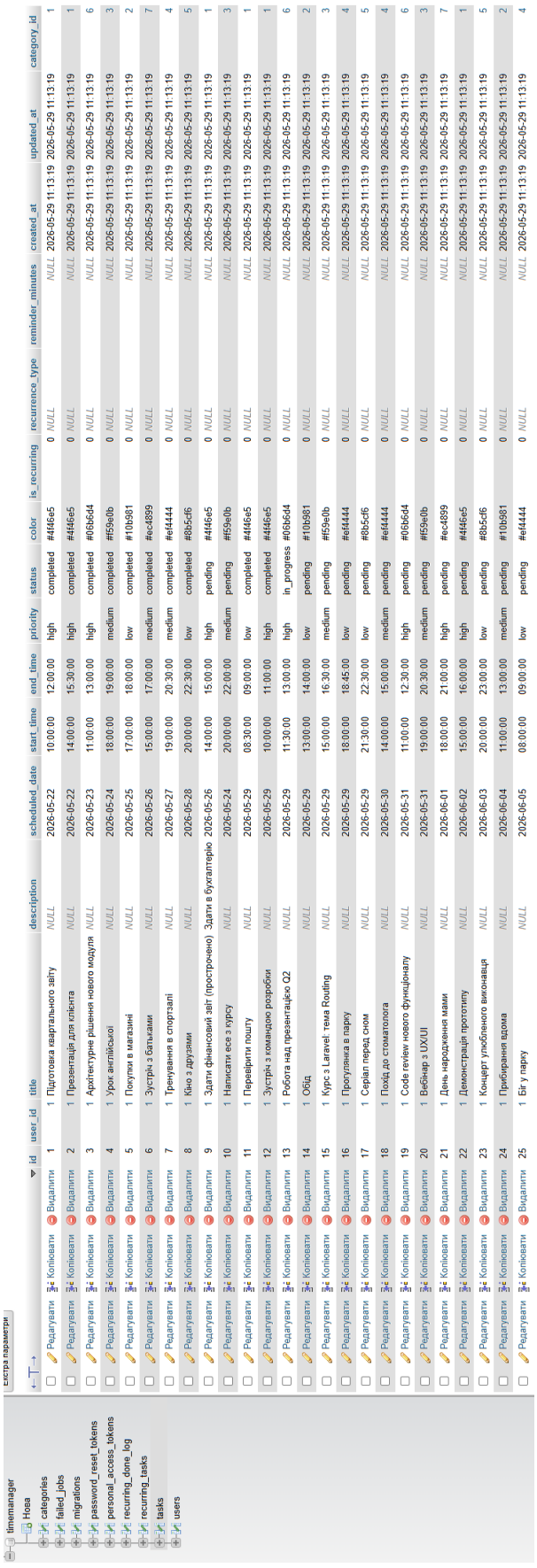

Рисунок Б.14 – Таблиця із завданнями у БД