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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для обліку робочого часу та розподілу по клієнтським проєктам співробітників рекламної агенції»

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

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

_____ Андрій РУБАН
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Андрій РУБАН

Керівник: Богдан ХУДІК
доктор філософії (PhD)

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Бакалавр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рубану Андрію Володимировичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для обліку робочого часу та розподілу по клієнтським проєктам співробітників рекламної агенції» керівник кваліфікаційної роботи доктор філософії (PhD) Богдан ХУДІК, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

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

3. Вихідні дані до кваліфікаційної роботи: аналіз існуючих підходів до обліку робочого часу та розподілу завдань у рекламних агенціях; опис вимог до вебзастосунку для автоматизації цього процесу; документація з технологій, що забезпечують автентифікацію, інтерфейс користувача та взаємодію з базами даних; технічна документація фреймворків, бібліотек і сервісів, використаних для розробки вебзастосунку.

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

1. Аналіз предметної області та постановка задачі розробки вебзастосунку для обліку робочого часу та розподілу завдань у рекламній агенції.
2. Проєктування архітектури вебзастосунку, бази даних, інтерфейсів користувача та ролей взаємодії між працівниками.
3. Реалізація та тестування основних функцій системи: автентифікації, реєстрації часу, генерації звітів і моніторингу продуктивності.

4. Розгортання, інтеграція з внутрішніми сервісами агенції та підготовка системи до масштабування та подальшого розширення функціоналу.

5. Перелік графічного матеріалу: *презентація*

1. Порівняльний аналіз існуючих рішень для обліку робочого часу.
2. Програмні засоби реалізації вебзастосунку.
3. Архітектура вебзастосунку та схема взаємодії клієнта з API.
4. Діаграма послідовності процесу авторизації користувача.
5. Варіанти використання системи — кейси взаємодії з додатком.
6. Екранні форми.
7. Дашборд для моніторингу ключових показників ефективності.
8. Апробація результатів дослідження.

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02–04.03.2025	
2	Аналіз існуючих рішень для обліку робочого часу та проєктного менеджменту	05.03–12.03.2025	
3	Визначення функціональних та нефункціональних вимог до вебзастосунку	13.03–21.03.2025	
4	Проєктування архітектури вебзастосунку та клієнт-серверної взаємодії	22.03–29.03.2025	
5	Розробка інформаційної моделі та структури бази даних	30.03–06.04.2025	
6	Реалізація системи реєстрації, авторизації та керування ролями користувачів	07.04–13.04.2025	
7	Реалізація обліку робочого часу з можливістю фіксації, редагування та коментування записів	14.04-17.04.2025	

8	Розробка функціоналу для адміністрування клієнтських компаній і працівників	18.04–22.04.2025	
9	Створення інтерфейсів для генерації та експорту звітів	23.04–27.04.2025	
10	Тестування додатку та перевірка коректності взаємодії між компонентами системи	28.04–04.05.2025	
11	Налаштування логування, обробки помилок та базових метрик моніторингу	05.05–09.05.2025	
12	Оформлення пояснювальної записки: вступ, висновки, реферат	10.05–12.05.2025	
13	Розробка презентації та демонстраційних матеріалів	13.05–16.05.2025	
14	Попередній захист роботи	17.05–30.05.2025	

Здобувач вищої освіти _____ Андрій РУБАН
(підпис)

Керівник
кваліфікаційної роботи _____ Богдан ХУДІК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 73 стор., 23 табл., 27 рис., 20 джерел.

Мета роботи — розробка високопродуктивної, масштабованої та зручної системи обліку робочого часу та управління проєктами для корпоративного середовища, що забезпечить ефективний моніторинг витрат часу, спростить формування звітності та поліпшить процеси управління персоналом і проєктами.

Об'єкт дослідження — процес обліку робочого часу та управління проєктами в корпоративному середовищі з використанням сучасних веб-технологій.

Предмет дослідження — методи, інструментальні засоби та технології реалізації вебзастосунку обліку робочого часу з використанням React, Next.js та Drizzle ORM.

Короткий зміст роботи — у роботі проаналізовано підходи до створення систем обліку робочого часу та управління проєктами. Досліджено існуючі рішення та виявлено їхні обмеження для корпоративного використання. Спроєктовано та реалізовано вебзастосунок на базі сучасного стеку технологій (React, Next.js, Tailwind CSS, Drizzle ORM) з підтримкою ручного введення та обліку робочого часу.

Реалізовано багаторівневу систему доступу з ролями адміністраторів та звичайних користувачів, гнучку систему звітності з можливістю експорту в Excel та інтерфейс з підтримкою української, англійської.

Розроблено зручний користувацький інтерфейс для керування співробітниками, компаніями та проєктами.

Сферою використання розробленої системи є компанії різного розміру, особливо консалтингові агентства та IT-компанії, де критично важливим є точний

облік робочого часу для виставлення розрахунків клієнтів та оцінки ефективності проєктів.

Розроблена система успішно впроваджена в компанії для забезпечення ефективного управління часовими ресурсами.

КЛЮЧОВІ СЛОВА: ОБЛІК РОБОЧОГО ЧАСУ, УПРАВЛІННЯ ПРОЄКТАМИ, REACT, NEXT.JS, DRIZZLE ORM, TAILWIND CSS, АВТЕНТИФІКАЦІЯ, CLERK, БАГАТОМОВНІСТЬ, ЕКСПОРТ ДАНИХ, АДМІНІСТРУВАННЯ СПІВРОБІТНИКІВ, ЗВІТНІСТЬ.

ЗМІСТ

ВСТУП

Актуальність: у сучасному бізнес-середовищі ефективне управління часовими ресурсами стало визначним фактором успіху компаній. Цифрова трансформація бізнес-процесів призвела до необхідності впровадження спеціалізованих інструментів для обліку робочого часу, управління проєктами та оптимізації розподілу людських ресурсів. Особливо гостро ця потреба відчувається в консалтингових агентствах, IT-компаніях та інших сервісних організаціях, де облік часу безпосередньо пов'язаний з виставленням рахунків клієнтів та розрахунком рентабельності проєктів.

Традиційні методи обліку робочого часу, такі як паперові табелі чи електронні таблиці, демонструють низку обмежень у сучасних умовах через:

1. низьку автоматизацію — необхідність ручного введення та обробки даних;
2. обмежені можливості аналітики — складність формування комплексних звітів;
3. відсутність інтеграції — ізольованість даних від інших корпоративних систем;
4. недостатню гнучкість — складність налаштування під специфічні потреби компанії;
5. проблеми з доступністю — обмежені можливості для віддаленої роботи.

У цих умовах розробка спеціалізованого вебзастосунку для обліку робочого часу та управління проєктами стає не просто бажаним, а необхідним кроком для підвищення ефективності бізнес-процесів. Така система має надавати зручні інструменти для трекінгу витраченого часу, формування звітності, управління проєктами та ресурсами, а також забезпечувати багаторівневий доступ для різних категорій користувачів.

Об'єктом дослідження є процес обліку робочого часу та управління проєктами в корпоративному середовищі з використанням сучасних вебтехнологій,

що включає методи трекінгу часу, формування звітності, управління проєктами та персоналом.

Предметом дослідження є методи, інструментальні засоби та технології реалізації вебзастосунку обліку робочого часу з використанням React, Next.js та Drizzle ORM, що забезпечують ефективний моніторинг часових витрат, формування звітності та управління проєктами.

Метою роботи є розробка високопродуктивної, масштабованої та зручної системи обліку робочого часу, а також управління проєктами для корпоративного середовища, що забезпечить ефективний моніторинг витрат часу, спростить формування звітності та поліпшить процеси управління персоналом і проєктами.

Методи дослідження: у процесі роботи було проаналізовано існуючі підходи до обліку робочого часу та управління проєктами, досліджено провідні рішення на ринку та виявлено їхні обмеження для корпоративного використання. На основі цього аналізу було спроєктовано архітектуру вебзастосунку з використанням сучасних технологій та компонентного підходу. Дослідження процесу інтеграції різних механізмів обліку часу проводилося під час безпосередньої розробки системи, включаючи ручне введення, трекінг та формування звітності.

Наукова новизна роботи полягає у розробці комплексного підходу до обліку робочого часу та управління проєктами в корпоративному середовищі з використанням сучасних веб-технологій, що забезпечує високу продуктивність, зручність користування та ефективне використання ресурсів. Запропоновано оригінальний механізм інтеграції трекінгу часу з управлінням проєктами, реалізовано гнучку систему формування звітності з підтримкою експорту даних, а також впроваджено багатомовний інтерфейс з адаптивним дизайном.

Практична значущість результатів полягає у можливості використання розробленої системи для ефективного обліку робочого часу та управління проєктами в компаніях різного розміру, що дозволяє значно спростити процеси трекінгу часу, підвищити точність виставлення рахунків клієнтів та забезпечити ефективний моніторинг продуктивності співробітників. Система успішно впроваджена в реальному бізнес-середовищі, що підтверджує її практичну цінність та відповідність потребам користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис задачі та предметної галузі

У сучасному бізнес-середовищі ефективне управління часовими ресурсами є критичним фактором успіху для компаній різного масштабу. Особливо це стосується організацій, що працюють за моделлю погодинної оплати – консалтингових агентств, IT-компаній, юридичних фірм та інших сервісних підприємств, де точний облік витраченого часу безпосередньо впливає на фінансові результати.

Аналіз ринку показує зростаючу потребу в спеціалізованих системах, які б забезпечували:

– точний облік робочого часу співробітників;

– прозорий розподіл витрат часу за проєктами та клієнтами;

– ефективне управління проєктами та ресурсами;

– формування аналітичних звітів для прийняття управлінських рішень;

– оптимізацію процесу формування рахунків для клієнтів.

Традиційні методи обліку робочого часу, такі як ручне заповнення табелів, електронні таблиці або спеціалізоване програмне забезпечення з обмеженою функціональністю, вже не задовольняють потреби сучасного бізнесу через:

– недостатню автоматизацію – значні затрати часу на ручне введення даних;

– відсутність інтеграції – розрізненість даних між системами обліку часу, управління проєктами та виставлення рахунків;

– обмежені можливості аналітики – складність отримання комплексних звітів у режимі реального часу;

– проблеми з доступністю – обмежені можливості для дистанційної та гібридної роботи;

– відсутність належного контролю – складність відстеження ефективності використання робочого часу.

Тому виникає потреба у створенні комплексної системи, яка б вирішувала вищезазначені проблеми, забезпечуючи зручний інтерфейс для користувачів, гнучкі інструменти управління для адміністраторів та потужні аналітичні можливості для керівництва.

1.2 Аналіз існуючих рішень для управління доступом

На сучасному ринку існує декілька поширених систем обліку робочого часу та управління проєктами, кожна з яких має свої особливості, переваги та обмеження. Розглянемо найпопулярніші з них та порівняємо їх із розроблюваною системою «Mediacom UA time track».

1.2.1 Toggl Track

Toggl Track – популярне рішення для обліку робочого часу, орієнтоване на команди різного розміру (рис. 1.1).

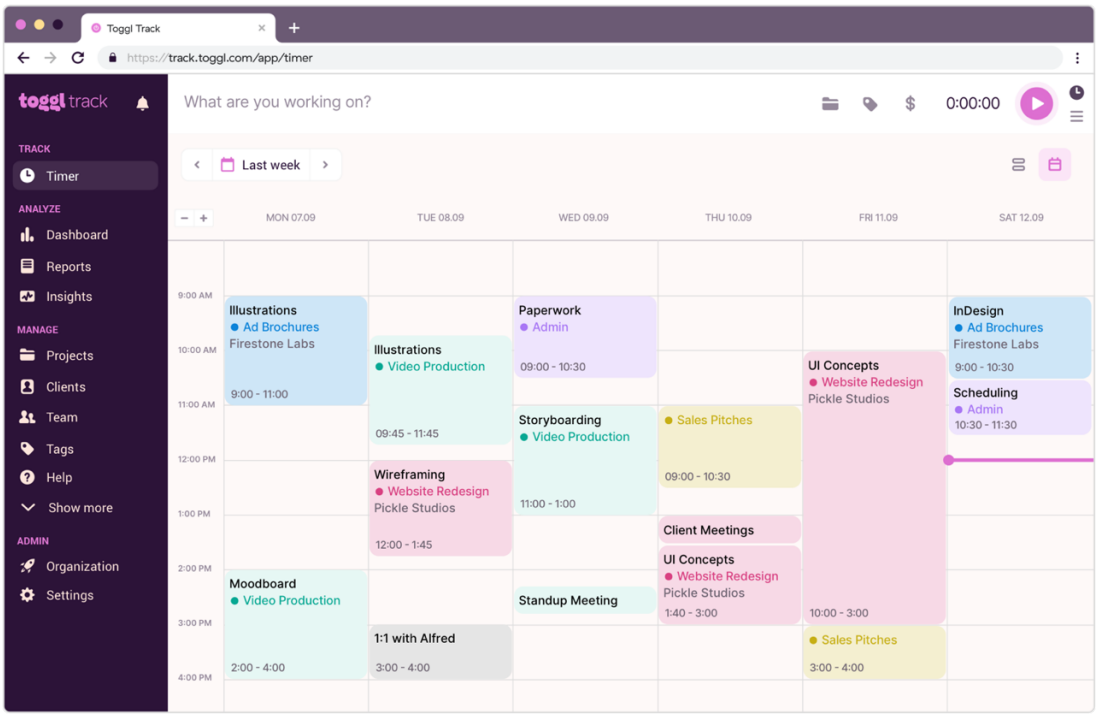


Рис. 1.1 Структура роботи Toggl Track

Переваги:

– інтуїтивно зрозумілий інтерфейс;
– потужна система звітності;
– широкі можливості інтеграції з іншими сервісами;
– доступність на різних платформах.

Недоліки:

– обмежені можливості управління проєктами;
– високі ціни для бізнес-тарифів;
– відсутність глибокої кастомізації під потреби конкретної компанії.

1.2.2 Clockify

Clockify – безкоштовна система обліку часу з додатковими платними функціями (рис. 1.2).

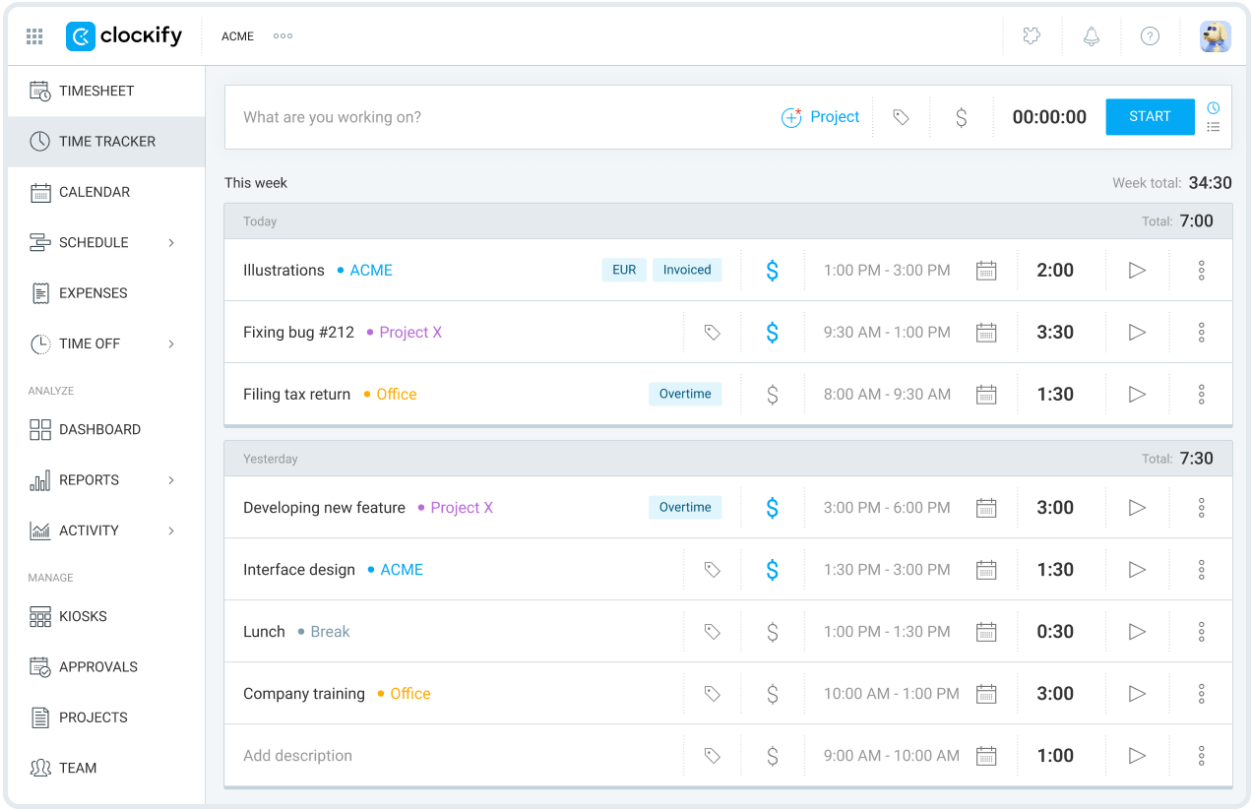


Рис. 1.2 Структура роботи Clockify

Переваги:

– необмежена кількість користувачів на безкоштовному тарифі;
– широкий набір функцій;
– можливість створення та відправки рахунків клієнтам.

Недоліки:

– деякі важливі функції доступні лише у платних планах;
– інтерфейс може бути складним для нових користувачів;
– обмежені можливості інтеграції у безкоштовній версії.

1.2.3 Hubstaff

Hubstaff – рішення для моніторингу продуктивності та обліку часу (рис. 1.3).

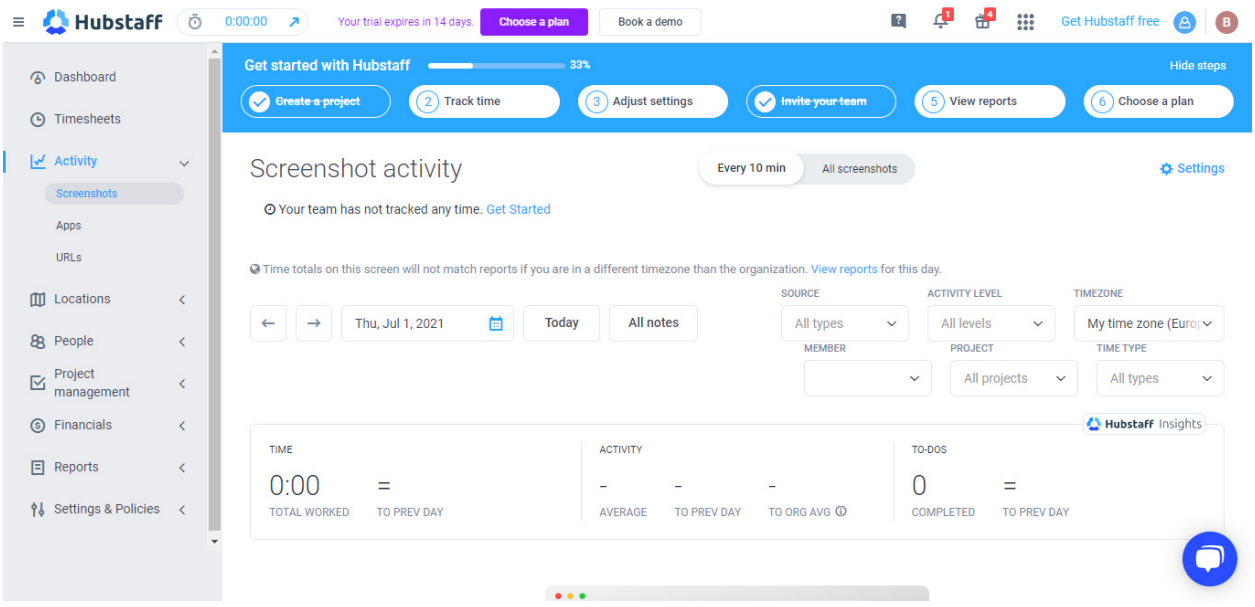


Рис. 1.3 Структура роботи Hubstaff

Переваги:

– розширені функції моніторингу активності користувачів;
– можливість автоматичного створення скріншотів;
– відстеження геолокації для мобільних команд.

Недоліки:

– відсутність багатомовної підтримки;

– надмірний контроль може викликати негативне сприйняття співробітниками;

– обмежена гнучкість у налаштуванні інтерфейсу.

1.2.4 RescueTime

RescueTime – інструмент для автоматичного відстеження часу, витраченого на різні активності (рис. 1.4).

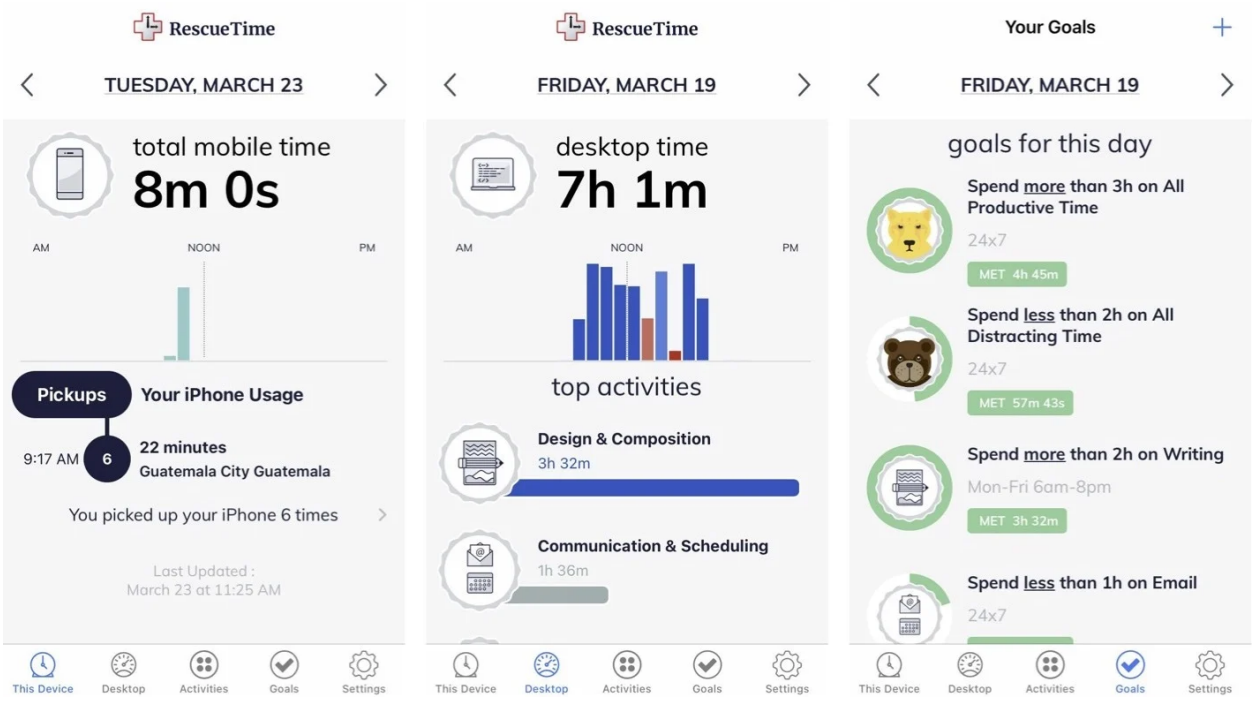


Рис. 1.4 Структура роботи RescueTime

Переваги:

– повністю автоматичний збір даних;

– детальний аналіз продуктивності;

– мінімальні зусилля з боку користувача.

Недоліки:

– обмежені можливості для ручного введення активностей;

– фокус більше на особистій продуктивності, ніж на корпоративному використанні;

– не призначений для виставлення рахунків клієнтів.

1.2.5 Порівняльна таблиця рішень

Таблиця 1.1 містить підсумкові результати аналізу особливостей розглянутих систем обліку робочого часу:

Таблиця 1.1

Порівняльна характеристика систем обліку робочого часу

Технічні характеристики	**Toggl Track**	**Clockify**	**Hubstaff**	**RescueTime**	**Mediacom UA time track**
Призначення	Облік робочого часу	Тайм-трекінг і управління проєктами	Моніторинг продуктивності	Автоматичний облік часу	Облік робочого часу та розподіл по клієнтам
Платформа	Веб, мобільний додаток	Веб, мобільний додаток	Веб, мобільний додаток	Веб, мобільний додаток	Веб
Технології	React, Node.js	React, GoLang	Electron, Node.js	Python, Electron	Next.js, React, Drizzle ORM
База даних	PostgreSQL	MySQL	SQLite, PostgreSQL	SQLite	SQLite
Основні функції	Тайм-трекінг, звіти, експорт	Тайм-трекінг, інвойсинг, звіти	Тайм-трекінг, продуктивність	Автоматичний тайм-трекінг, звіти	Тайм-трекінг, розподіл годин по проєктах, звіти, експорт
Мультимовність	Так	Так	Ні	Так	Так (uk/en)
Аутентифікація	Google, Email	Google, Email, Microsoft	Email	Google, Email	Clerk (email)
Розмежування доступу	Адмін, користувач	Адмін, менеджер, користувач	Адмін, користувач	Адмін, користувач	Адмін, користувач

Продовження таблиці 1.1

Порівняльна характеристика систем обліку робочого часу

Технічні характеристики	**Toggl Track**	**Clockify**	**Hubstaff**	**RescueTime**	**Mediacom UA time track**
Можливості інтеграції	Asana, Jira, Slack	Asana, Trello, Jira, Slack	Trello, Asana, Slack	Trello, Slack	API-інтерфейс для інтеграції
Модуль звітності	Так	Так	Так	Так	Так
Модуль статистики	Так	Так	Так	Так	Так
Ціна	Безкоштовно, платні функції	Безкоштовно, платні функції	Безкоштовно, платні функції	Безкоштовно, платні функції	Індивідуально для компанії

1.2.6 Обґрунтування актуальності розробки системи «Mediacom UA time track»

Проведений аналіз існуючих рішень дозволив виявити низку факторів, що обґрунтовують доцільність розробки нової системи обліку робочого часу та управління проєктами:

1. спеціалізація під потреби конкретної компанії – на відміну від універсальних рішень, система «Mediacom UA time track» розробляється з урахуванням специфічних вимог замовника, що забезпечує максимальну відповідність бізнес-процесам;
2. оптимальне співвідношення функціональності та простоти – система поєднує необхідний функціонал для ефективного обліку часу та управління проєктами без надлишкових можливостей, що ускладнюють інтерфейс;
3. гнучкість у налаштуванні та розширенні – використання сучасного стеку технологій (Next.js, React, Drizzle ORM) забезпечує можливість швидкого розширення функціоналу та адаптації системи до мінливих потреб бізнесу;
4. оптимізація під корпоративне використання – на відміну від рішень, орієнтованих на індивідуальне використання, система «Mediacom UA time track»

проєктується з урахуванням корпоративних потреб у розподілі прав доступу, формуванні аналітичної звітності та інтеграції з іншими системами;

5. відсутність надмірного контролю – система фокусується на обліку часу та результатах, а не на моніторингу активності користувачів, що позитивно впливає на робочу атмосферу та мотивацію співробітників.

Таким чином, розробка системи «Mediacom UA time track» є обґрунтованою відповіддю на специфічні потреби компаній у ефективному управлінні часовими ресурсами та проєктами.

1.3 Опис IT-засобів для розробки системи

Для розробки системи обліку робочого часу «Mediacom UA time track» обрано сучасний стек технологій, що забезпечує високу продуктивність, зручність розробки та підтримки, а також відповідає сучасним вимогам до веб-додатків.

1.3.1 Фронтенд-технології

React

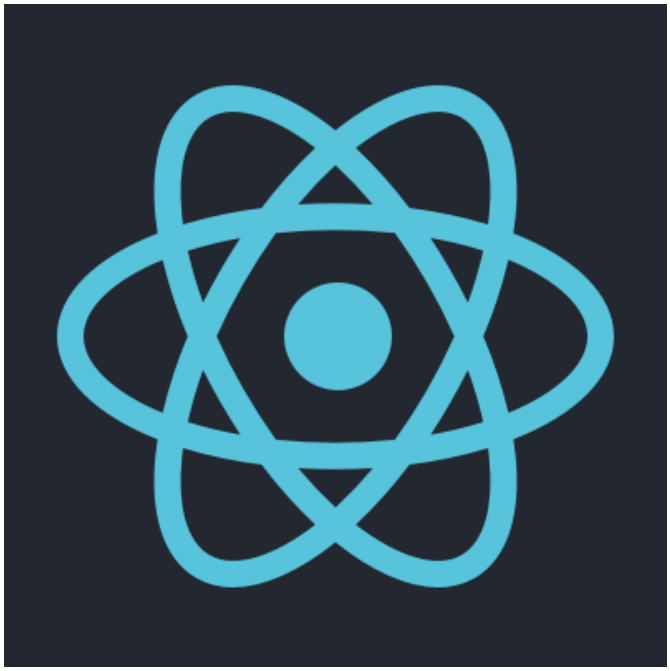


Рис. 1.5 Логотип React

React – бібліотека JavaScript для розробки користувацьких інтерфейсів, розроблена компанією Facebook (нині Meta). Використання React надає такі переваги (рис. 1.5):

- компонентний підхід до розробки інтерфейсу;
- віртуальний DOM для оптимізації продуктивності;
- реактивне оновлення даних;
- можливість використання JSX для декларативного опису інтерфейсу;
- широка екосистема інструментів та бібліотек.

Next.js

Рис. 1.6 Логотип Next.js

Next.js – фреймворк для React, що забезпечує (рис.1.6):

- серверний рендеринг (SSR) та генерацію статичних сторінок (SSG);
- оптимізацію продуктивності та SEO;
- вбудовану систему маршрутизації;
- API-маршрути для створення серверних ендпоінтів;
- автоматичне розділення коду (code splitting);
- підтримку TypeScript з коробки.

Tailwind CSS

Рис. 1.7 Логотип Tailwind CSS

Tailwind CSS – фреймворк утилітарних класів для швидкої розробки інтерфейсів (рис. 1.7):

- принцип «utility-first» для швидкого стилізування компонентів;
- консистентні дизайн-токени (кольори, розміри, відступи);
- адаптивний дизайн з коробки;
- можливість кастомізації через конфігурацію;
- оптимізація розміру CSS у продакшн-збірці.

Shadcn/UI

Рис. 1.8 Логотип Shadcn/UI

Система компонентів на основі Radix UI, що забезпечує (рис. 1.8):

- набір готових, доступних та кастомізованих компонентів;
- підтримку темної та світлої теми;
- відповідність принципам доступності (a11y);
- інтеграцію з Tailwind CSS.

1.3.2 Бекенд-технології

Next.js API Routes

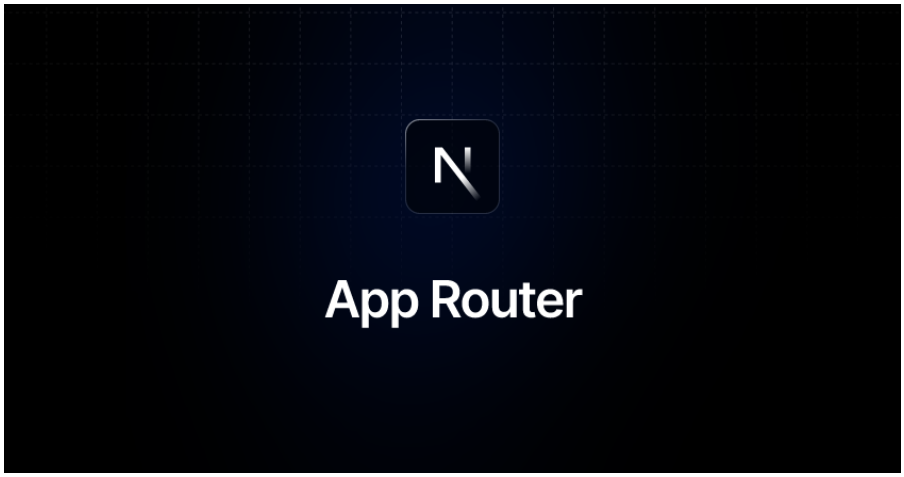


Рис. 1.9 Логотип Next.js API Routes

Для серверної частини використовуються API-маршрути Next.js, що дозволяють (рис.1.9):

- створювати серверні ендпоінти в межах єдиного проєкту;
- використовувати єдину кодову базу для фронтенду та бекенду;
- спрощувати розгортання та підтримку додатку.

Drizzle ORM

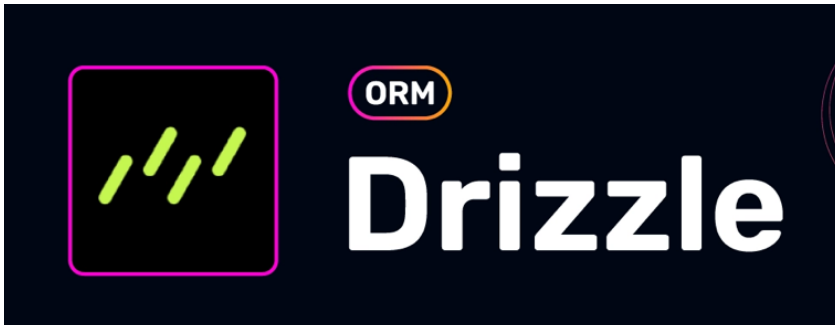


Рис. 1.10 Логотип Drizzle ORM

Drizzle ORM – сучасний TypeScript ORM для роботи з базами даних (рис. 10):

- типобезпечні запити до бази даних;
- декларативне визначення схеми;
- міграції та управління схемою;
- підтримка різних баз даних (PostgreSQL, MySQL, SQLite);
- низький рівень абстракції для кращого контролю над SQL-запитами.

1.3.3 Бази даних та зберігання

SQLite

Рис. 1.11 Логотип SQLite

Для збереження даних обрано SQLite – вбудовану реляційну базу даних (рис. 1.11):

- не потребує окремого сервера бази даних;
- висока надійність та стабільність;
- висока продуктивність для невеликих та середніх додатків;
- простота налаштування та обслуговування;
- підтримка транзакцій та ACID-властивостей.

1.3.4 Аутентифікація та авторизація

Clerk

Рис. 1.12 Логотип Clerk

Для управління аутентифікацією та користувачами використовується Clerk (рис. 1.12):

- повноцінна система управління користувачами;
- безпечна аутентифікація через email;
- вбудований захист від поширених атак;
- готові компоненти для форм входу та реєстрації;
- інтеграція з Next.js та інтерфейс для розробки.

1.3.5 Інструменти розробки та розгортання

TypeScript

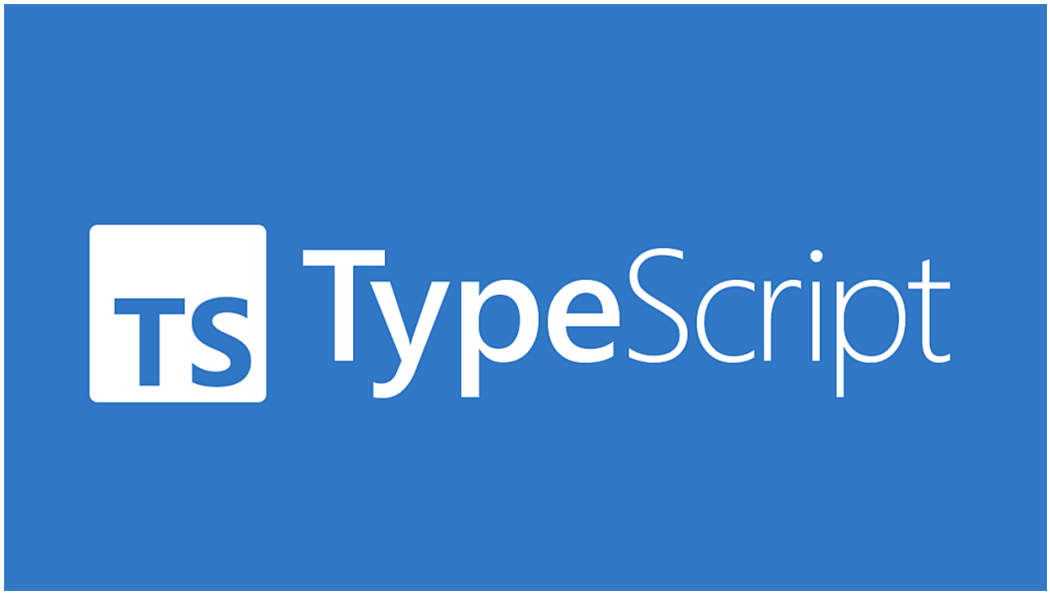


Рис. 1.13 Логотип TypeScript

TypeScript – типізований надсет JavaScript, що забезпечує (рис. 1.13):

- статичну типізацію для зменшення кількості помилок;
- покращену підтримку IDE (автодоповнення, рефакторинг);
- можливість використання сучасних функцій JavaScript;
- кращу документацію коду та підтримку великих проєктів.

i18next

Рис. 1.14 Логотип i18next

Для забезпечення багатомовності використовується i18next (рис. 1.14):

- підтримка декількох мов (українська, англійська);
- гнучкість у форматуванні тексту;
- можливість динамічної зміни мови без перезавантаження сторінки;
- інтеграція з React через react-i18next.

ExcelJS

Рис. 1.15 Логотип ExcelJS

Для формування та експорту звітів використовується ExcelJS (рис. 1.15):

- створення Excel-файлів безпосередньо у браузері;
- гнучке налаштування форматування даних;
- підтримка формул та стилів;
- висока продуктивність при роботі з великими об'ємами даних.

Обраний стек технологій забезпечує оптимальний баланс між зручністю розробки, продуктивністю та підтримуваністю системи в довгостроковій перспективі.

1.4 Визначення об'єкту, предмету, мети та задач роботи

1.4.1 Об'єкт дослідження

Об'єктом дослідження є процес обліку робочого часу та управління проєктами в корпоративному середовищі з використанням сучасних веб-технологій, що включає методи трекінгу часу, формування звітності, управління проєктами та ресурсами.

1.4.2 Предмет дослідження

Предметом дослідження є методи, інструментальні засоби та технології реалізації вебзастосунку обліку робочого часу з використанням React, Next.js та Drizzle ORM, що забезпечують ефективний моніторинг часових витрат, формування звітності та управління проєктами.

1.4.3 Мета роботи

Метою роботи є розробка високопродуктивної, масштабованої та зручної системи обліку робочого часу та управління проєктами для корпоративного середовища, що забезпечить ефективний моніторинг витрат часу, спростить формування звітності та поліпшить процеси управління персоналом і проєктами.

1.4.4 Задачі дипломної роботи

1. Аналіз існуючих рішень для обліку робочого часу та їх обмежень.
2. Визначення функціональних та нефункціональних вимог до системи.
3. Проєктування архітектури веб-додатку на основі сучасних технологій.
4. Розробка схеми бази даних для зберігання інформації про користувачів, проєкти та часові записи.
5. Реалізація інтерфейсу користувача з використанням React та Tailwind CSS.
6. Імплементація системи аутентифікації та авторизації з використанням Clerk.
7. Розробка компонентів для обліку робочого часу та управління проєктами.
8. Створення системи фільтрації та формування звітів.
9. Реалізація функціоналу експорту даних у формат Excel.
10. Впровадження багатомовності з підтримкою української та англійської мов.
11. Тестування системи на відповідність функціональним та нефункціональним вимогам.

1.5 Функціональні та нефункціональні вимоги до програмного забезпечення

1.5.1 Функціональні вимоги

На основі аналізу предметної області та потреб користувачів сформовано наступні функціональні вимоги до системи «Mediacom UA time track»:

1. Реєстрація та автентифікація користувачів з різними рівнями доступу.
   1.1. Підтримка ролей: керівник компанії (адміністратор), співробітник.
   1.2. Безпечний процес автентифікації з використанням сучасних стандартів.
2. Управління профілями співробітників

2.1. Створення та редагування профілів з інформацією про посаду, відділ, контактні дані.

2.2. Налаштування прав доступу до функцій системи.

2.3. Можливість деактивації облікових записів.

3. Управління клієнтськими компаніями

3.1. Створення нових компаній.

3.2. Редагування інформації про існуючі компанії.

4. Призначення співробітників до компаній

4.1. Визначення ролей співробітників.

4.2. Налаштування доступу до інформації про конкретні компанії.

5. Облік робочого часу

5.1. Введення та редагування записів про відпрацьований час.

5.2. Додавання коментарів до часових записів.

5.3. Автоматичний розрахунок загального відпрацьованого часу.

6. Формування звітності

6.1. Створення звітів про відпрацьований час співробітників.

6.2. Фільтрація даних за різними параметрами (період, співробітник).

6.3. Агрегація та візуалізація даних у зручній формі.

6.4. Експорт звітів у форматі Excel.

7. Моніторинг та аналіз продуктивності

7.1. Відображення ключових показників на дашборді.

8. Багатомовний інтерфейс

8.1. Підтримка української та англійської мов.

8.2. Можливість перемикання мови без перезавантаження сторінки.

8.3. Локалізація дат.

9. Адміністративні функції

9.1. Керування користувачами та їх правами.

1.5.2 Нефункціональні вимоги

Для забезпечення високої якості системи та відповідності очікуванням користувачів визначено наступні нефункціональні вимоги:

1. Продуктивність:

відгук системи не більше ніж 5 секунд при роботі з 100+ записами;

генерація звітів за не більше ніж 30 секунд для складних звітів;

оптимізація запитів до бази даних для мінімізації часу відгуку.

2. Безпека:

захист від поширених веб-вразливостей (XSS, CSRF, SQL-ін'єкції);

безпечне зберігання облікових даних користувачів.

3. Надійність:

коректна обробка помилок з інформативними повідомленнями.

4. Зручність використання:

інтуїтивно зрозумілий інтерфейс із логічною структурою навігації;

адаптивний дизайн для різних пристроїв (ПК, планшети, смартфони);

мінімальна кількість кроків для виконання типових операцій.

5. Масштабованість:

здатність системи ефективно працювати при збільшенні кількості користувачів;

можливість розширення функціоналу без суттєвих змін архітектури.

6. Сумісність:

підтримка сучасних браузерів (Chrome, Firefox, Safari, Edge);

коректна робота на різних операційних системах (Windows, macOS, Linux);

сумісність з різними пристроями та роздільними здатностями екрану.

7. Інтернаціоналізація:

повна підтримка багатомовності (українська, англійська);

коректне відображення дат, чисел та валют відповідно до локалізації.

8. Розширюваність:

модульна архітектура для простого додавання нових функцій;

API для потенційної інтеграції з іншими системами;

можливість підключення додаткових модулів у майбутньому.

Визначені функціональні та нефункціональні вимоги формують основу для проєктування та розробки системи «Mediacom UA time track», забезпечуючи її відповідність потребам користувачів.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ОБЛІКУ РОБОЧОГО ЧАСУ

2.1 Огляд архітектурних підходів до розробки вебзастосунку обліку робочого часу

2.1.1 Принципи сучасної веб-архітектури

Розробка сучасних вебзастосунків обліку робочого часу базується на фундаментальних архітектурних принципах, що забезпечують створення ефективних, масштабованих та зручних у підтримці додатків. Ці принципи формують основу для прийняття рішень щодо вибору технологій, структурування коду та організації взаємодії компонентів системи.

Одним з найважливіших принципів є чітке розмежування функціональних обов'язків між різними частинами системи. У контексті веб-додатків для обліку робочого часу це означає:

– *презентаційний шар* відповідає за відображення інформації користувачу та обробку його взаємодій;

– *бізнес-логіка* керує правилами обробки даних, валідацією та основними операціями системи;

– *шар доступу* до даних забезпечує збереження, отримання та маніпулювання інформацією.

Такий розподіл дозволяє незалежно розвивати кожен компонент, спрощує тестування та підвищує загальну якість коду.

Сучасні фронтенд-фреймворки, зокрема React, базуються на компонентній архітектурі, що передбачає:

– *інкапсуляцію* – кожен компонент містить власну логіку, стан та представлення;

– *повторне використання* – можливість застосування одного компонента в різних частинах додатку;

– *композицію* – створення складних інтерфейсів шляхом комбінування простих компонентів;

– *односпрямований потік даних* – передбачуваність поведінки через контрольований обмін інформацією.

Для системи обліку робочого часу це означає можливість створення «переносних» компонентів для форм введення часу, звітів, дашбордів та інших елементів інтерфейсу.

Кожен модуль, клас або функція системи повинні мати лише одну причину для зміни. У веб-додатках це проявляється через:

– спеціалізовані «хуки» для різних аспектів функціональності (управління станом, запити до API, локалізація);

– окремі компоненти для різних UI-елементів;

– dedicated API-роути для кожного типу операцій з даними.

– Високорівневі модулі не повинні залежати від низькорівневих – обидва типи мають залежати від абстракцій. Це реалізується через:

– dependency Injection для передачі залежностей компонентам;

– абстрактні інтерфейси для взаємодії з зовнішніми сервісами;

– провайдери контексту в React для розподілу спільного стану.

2.1.2 Особливості проєктування систем управління часом

Системи обліку робочого часу мають специфічні вимоги, що впливають на архітектурні рішення та вибір технологій. Розуміння цих особливостей є критичним для створення ефективного та зручного рішення.

Робота з часовими мітками є центральним аспектом систем обліку часу та потребує особливої уваги:

1. Точність вимірювань:

– підтримка різних рівнів деталізації (хвилини, секунди);

– округлення та агрегація часових інтервалів.

2. Історичність даних:

– можливість корекції помилкових записів

Системи обліку часу генерують значні обсяги даних, що потребує оптимізованих рішень:

3. Ефективні запити до бази даних:
- індексація часових полів та ключів пошуку;
- агрегація даних на рівні бази даних.

4. Кешування:
- локальне кешування в браузері для покращення відгуку.

5. Асинхронна обробка:
- фонова генерація складних звітів;
- неблокуюча обробка експорту даних.

6. Системи обліку часу містять чутливу інформацію про продуктивність та зарплатні дані:
- багаторівневий доступ (адміністратори, співробітники);
- захист персональних даних відповідно до регуляторних вимог;
- контроль доступу до часових записів інших користувачів.

2.1.3 Обґрунтування вибору архітектури Full-Stack додатку

Для реалізації системи «Mediacom UA time track» обрано архітектуру повноцінного веб-додатку (Full-Stack) на базі Next.js, що поєднує переваги серверного та клієнтського рендерингу. Цей вибір обґрунтовується рядом факторів, що забезпечують оптимальне рішення для даного типу додатків.

Розробка з використанням Next.js дозволяє писати фронтенд та бекенд код в межах одного проєкту, використовуючи JavaScript/TypeScript:

- спільна кодова база зменшує складність проєкту;
- єдиний процес збірки та розгортання;
- можливість повторного використання кодових компонентів між клієнтом та сервером;
- спрощене управління залежностями та версіями.

Оптимізація продуктивності Next.js надає вбудовані механізми оптимізації:

- автоматичне розділення коду (code splitting);

– серверний рендеринг (SSR) для швидшого завантаження початкових сторінок;

– генерація статичних сторінок (SSG) для контенту, що рідко змінюється;

– інкрементальна статична регенерація (ISR) для балансу між продуктивністю та актуальністю.

API Routes для бекенд-логіки забезпечують:

– створення RESTful API без додаткових інструментів;

– автоматичну оптимізацію та кешування;

– інтеграцію з middleware для автентифікації та авторизації;

– типобезпечність через спільні TypeScript типи.

Для забезпечення масштабованості та підтримуваності системи «Mediacom UA time track» було обрано багатошарову архітектуру. Структуру архітектурних шарів з використовуваними технологіями та зонами відповідальності наведено в таблиці 2.1:

Таблиця 2.1

Архітектурні шари системи «Mediacom UA time track»

Шар	**Технології**	**Відповідальність**
Презентаційний	React 19, Tailwind CSS, Shadcn/UI	Користувацький інтерфейс, взаємодія з користувачем
Бізнес-логіка	TypeScript, Custom Hooks, Context API	Обробка даних, валідація, бізнес-правила
API	Next.js API Routes, Middleware	RESTful API, автентифікація, авторизація
Доступ до даних	Drizzle ORM, SQL	Операції з базою даних, міграції, запити
Зберігання	SQLite	Персистентне зберігання даних

Презентаційний шар відповідає за відображення інформації та взаємодію з користувачем:

– react компоненти для різних частин інтерфейсу;
– адаптивний дизайн з використанням Tailwind CSS;
– система компонентів Shadcn/UI для консистентності;
– управління локальним станом через хуки React.

Шар бізнес-логіки містить основні правила обробки даних:

– валідація введених користувачем даних;
– обчислення статистики та агрегованих показників;
– форматування даних для відображення;
– обробка помилок та винятків.

API шар забезпечує комунікацію між клієнтом та сервером:

– RESTful ендпоінти для CRUD операцій;
– middleware для автентифікації через Clerk;
– серіалізація та десеріалізація даних;
– обробка HTTP запитів та відповідей.

Шар доступу до даних керує взаємодією з базою даних:

– ORM-запити через Drizzle;
– міграції схеми бази даних;
– оптимізація запитів та індексація;
– управління транзакціями.

При виборі архітектурного підходу для розробки системи було проаналізовано три основні варіанти: монолітний SPA, мікросервісну архітектуру та повнофункціональне рішення на базі Next.js. Порівняльний аналіз ключових характеристик цих підходів наведено в таблиці 2.2:

Таблиця 2.2

Порівняння архітектурних підходів

Характеристика	**Монолітний SPA**	**Мікросервіси**	**Next.js Full-Stack**
Складність розробки	Низька	Висока	Середня
Продуктивність	Середня	Висока	Висока
Масштабованість	Обмежена	Висока	Середня
SEO-оптимізація	Погана	Середня	Відмінна
Час розробки	Короткий	Довгий	Середній
Підтримка	Проста	Складна	Середня

Для системи «Mediacom UA time track» архітектура Next.js Full-Stack є оптимальним вибором з наступних причин:

1. розмір та складність проєкту – система має середню складність, що не потребує мікросервісної архітектури, але виходить за рамки простого SPA;
2. продуктивність – серверний рендеринг покращує швидкість завантаження, особливо критичну для бізнес-додатків;
3. SEO та доступність – навіть для внутрішніх корпоративних систем важлива доступність та швидкість індексації;
4. команда розробки – єдина кодова база спрощує розробку для невеликих команд;
5. час виходу на ринок – Next.js дозволяє швидше створити MVP та ітеративно розвивати функціонал.

Обрана архітектура забезпечує оптимальний баланс між продуктивністю, зручністю розробки та можливостями масштабування для системи обліку робочого часу «Mediacom UA time track».

2.2 Концептуальна архітектура системи «Mediacom UA time track»

2.2.1 Загальна структура системи

Система «Mediacom UA time track» розроблена як повноцінний веб-додаток з модульною архітектурою, що забезпечує ефективний облік робочого часу, управління проєктами та формування звітності. Архітектура системи базується на принципах чистої архітектури та розділення відповідальності, що дозволяє створити масштабовану та зручну в підтримці систему.

Функціональна архітектура системи «Mediacom UA time track» організована за модульним принципом, що забезпечує логічне розділення обов'язків та спрощує подальший розвиток додатка. Система складається з п'яти основних функціональних модулів, детальний опис яких наведено в таблиці 2.3:

Таблиця 2.3

Функціональні модулі системи «Mediacom UA time track»

Управління співробітниками	Керування профілями та правами	EmployeesList, EmployeeOverview
Управління компаніями	Керування клієнтськими компаніями	CompanyManagement, CompanyOverview
Облік часу	Трекінг робочого часу	TimeTrackingDashboard, TimeEntryForm, WeeklyCalendar, DayEntryForm
Звітність	Формування та експорт звітів	EmployeeReports, ReportExportModal, RecentReports
Адміністрування	Системне управління	AdminDashboard, DashboardSidebar, ProjectTable

Концептуальна архітектура системи представлена у вигляді багаторівневої структури, де кожен рівень має чітко визначені обов'язки та інтерфейси взаємодії.

Така архітектурна модель базується на принципі розділення відповідальностей, що дозволяє створити чітку ієрархію компонентів системи. Кожен рівень виконує специфічні функції та взаємодіє з суміжними рівнями через стандартизовані інтерфейси.

Багаторівнева структура забезпечує логічну організацію системи, де верхні рівні використовують сервіси нижніх рівнів. Це створює односпрямований потік залежностей, що спрощує розуміння та підтримку системи.

Архітектура сприяє модульності розробки, оскільки зміни в одному рівні мінімально впливають на інші компоненти (рис. 2.1). Така структура також полегшує тестування кожного рівня незалежно від інших.

Концептуальний підхід дозволяє команді розробників працювати паралельно над різними рівнями системи, що підвищує ефективність процесу розробки.

Впровадження багаторівневої архітектури забезпечує гнучкість системи та можливість її поетапного розвитку.

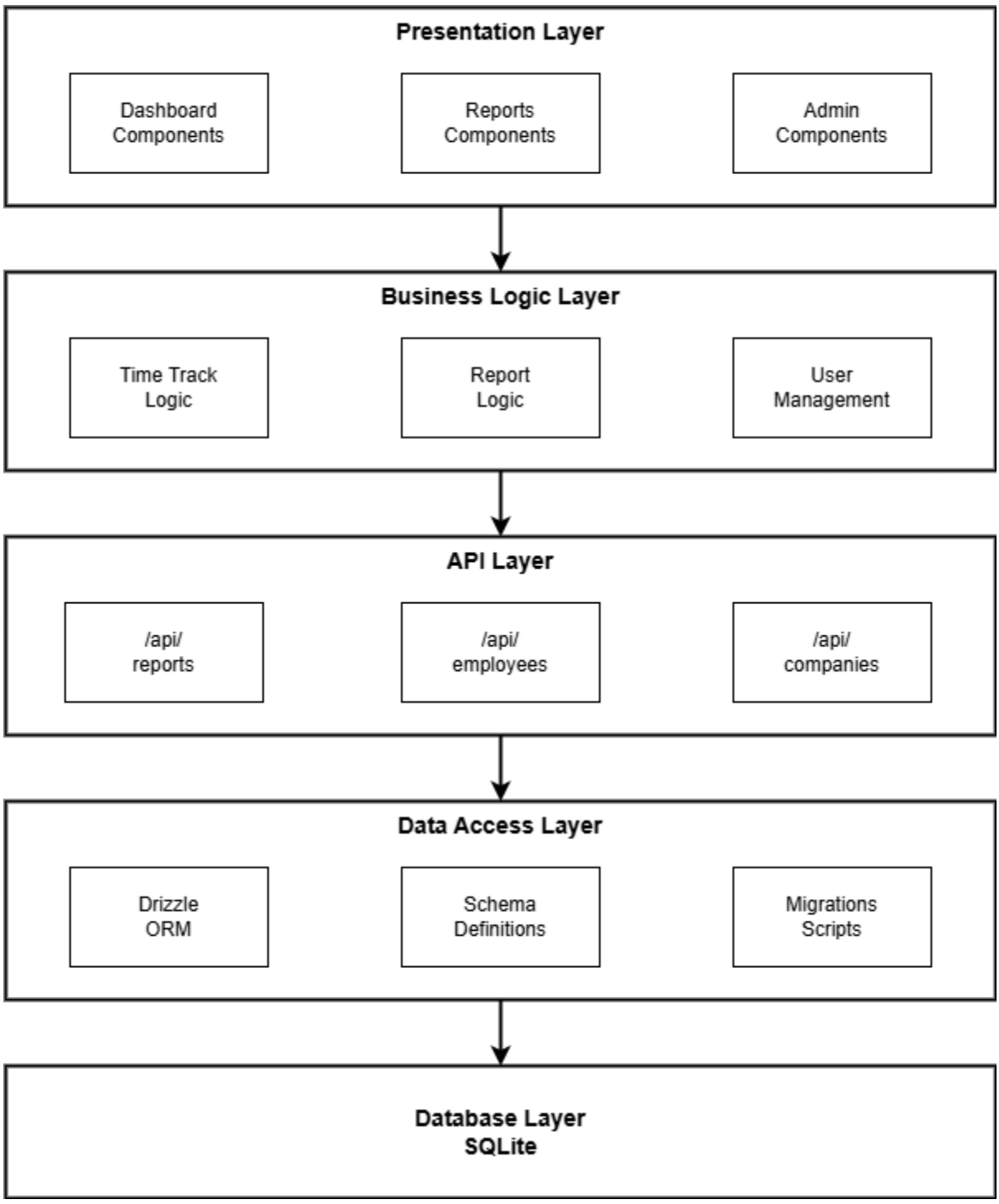


Рис. 2.1 Архітектурна діаграма системи

2.2.2 Архітектура клієнтської частини

Клієнтська частина системи побудована на базі React 19 з використанням функціональних компонентів та хуків. Архітектура фронтенду організована за принципом атомарного дизайну та компонентної ієрархії.

Базові найпростіші переносні компоненти інтерфейсу:

button – кнопки з різними стилями та станами;

input – поля введення тексту;

label – мітки для форм;

badge – індикатори статусу;

avatar – аватари користувачів;

skeleton – заглушки для завантаження.

Складені компоненти комбінації базових компонентів для конкретних задач:

formField – поле форми з міткою та валідацією;

searchBox – поле пошуку з кнопкою;

datePicker – компонент вибору дати;

tableRow – рядок таблиці з даними;

card – контейнер для групування контенту.

«Організми» – складні компоненти, що реалізують певну бізнес-логіку:

dataTable – таблиця з сортуванням, фільтрацією та пагінацією.

navigationMenu – головне меню навігації;

dashboardStats – панель статистики;

reportGenerator – генератор звітів;

timeTracker – компонент відстеження часу.

Шаблони компоненти верхнього рівня, що визначають структуру сторінок:

dashboardLayout – загальний макет з сайдбаром;

authLayout – макет для сторінок аутентифікації;

reportLayout – макет для сторінок звітів.

Для управління станом додатку використовується комбінований підхід:

Локальний стан:

react hooks (useState, useReducer, useRef) для стану компонентів.

Глобальний стан:

react Context API для аутентифікації та глобальних налаштувань;

custom hooks для бізнес-логіки.

Серверний стан:

SWR pattern для кешування та синхронізації даних;

оптимістичні оновлення для покращення користувацького досвіду.

Описана компонентна архітектура забезпечує високий рівень переносності коду та спрощує підтримку системи. Загальна кількість розроблених компонентів та їх розподіл за функціональними категоріями представлено в таблиці 2.4:

Таблиця 2.4

Розподіл компонентів системи за функціональними категоріями

Категорія	**Кількість компонентів**	**Основні компоненти**
Базові UI	15+	Button, Input, Card, Table, Badge
Форми	10+	LoginForm, TimeEntryForm, CompanyForm
Навігація	8+	Sidebar, Breadcrumb, Pagination, Menu
Дашборди	6+	TimeTrackingDashboard, AdminDashboard
Звіти	5+	EmployeeReports, ReportExportModal
Управління	8+	EmployeesList, CompanyManagement

2.2.3 Архітектура серверної частини

Серверна частина системи реалізована за допомогою Next.js API Routes, що забезпечує створення RESTful API в межах єдиного проєкту. Архітектура бекенду організована за принципами чистої архітектури представлена в таблиці 2.5:

Таблиця 2.5

Структура API ендпоінтів

Група ендпоінтів	**Маршрут**	**HTTP методи**	**Призначення**
Аутентифікація	/api/auth/*	GET, POST	Управління сесіями, профіль користувача
Співробітники	/api/employees	GET, POST, PUT, DELETE	CRUD операції з співробітниками

Продовження таблиці 2.5

Структура API ендпоінтів

Група ендпоінтів	**Маршрут**	**HTTP методи**	**Призначення**
Компанії	/api/companies	GET, POST, PUT, DELETE	Управління клієнтськими компаніями
Звіти	/api/reports	GET, POST, PUT, DELETE	Операції з часовими записами
Дашборд	/api/dashboard	GET	Агреговані дані для дашборду
Експорт	/api/export	POST	Генерація Excel файлів

Система використовує middleware для обробки спільних завдань:

Аутентифікація Middleware:

- перевірка валідності сесії через Clerk;
- додавання інформації про користувача до контексту запиту;
- перенаправлення неавторизованих користувачів.

Авторизація Middleware:

- перевірка прав доступу до ресурсів;
- розмежування доступу між адміністраторами та користувачами;
- контроль доступу до персональних даних.

Обробка помилок:

- централізована обробка винятків;
- логування помилок для моніторингу;
- уніфіковані формати відповідей про помилки.

2.2.4 Архітектура бази даних

База даних системи спроєктована з урахуванням специфіки обліку робочого часу та потреб бізнес-логіки. Використовується SQLite як легке та ефективне рішення для зберігання даних.

Структура бази даних оптимізована для ефективного зберігання та швидкого доступу до інформації про робочий час, співробітників та клієнтські компанії. Основні сутності системи з їх призначенням та ключовими атрибутами представлено в таблиці 2.6:

Таблиця 2.6

Основні сутності бази даних

Сутність	**Призначення**	**Ключові атрибути**
Employees	Співробітники компанії	id, name, email, position, department, clerk_id
Companies	Клієнтські компанії	id, name, contact, email, phone, address
Clients	Клієнти	id, name
Reports	Записи робочого часу	id, employee_id, date, hours, project_brand, comments
Reports_to_Companies	Зв'язок звітів з компаніями	report_id, company_id

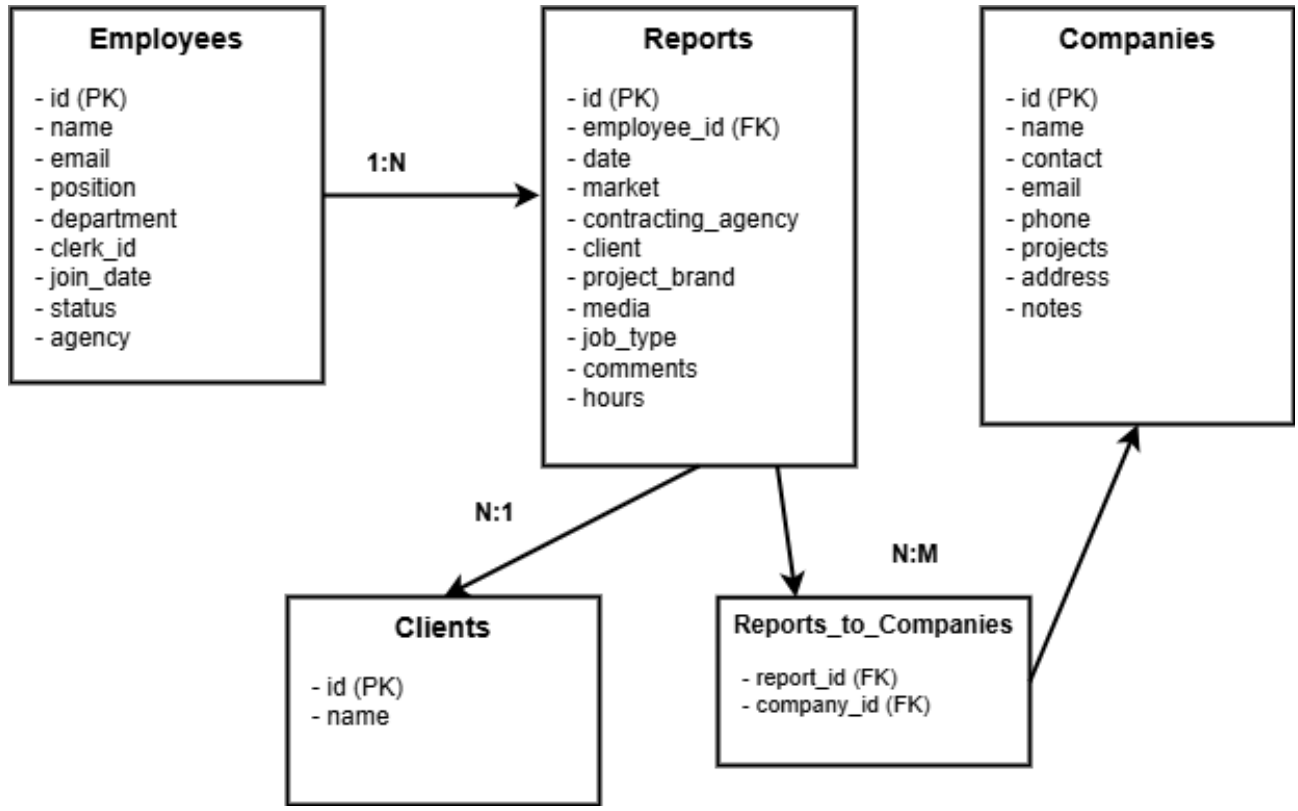


Рис. 2.2 Схема взаємозв'язків

Для забезпечення високої продуктивності створено наступні індекси (рис. 2.2):

- первинні ключі – автоматичні індекси для всіх таблиць;
- індекси зовнішніх ключів – для швидкого з'єднання таблиць;
- композитні індекси – для часто використовуваних комбінацій полів;
- текстові індекси – для полів пошуку (name, email).

2.2.5 Система інтеграцій та зовнішніх сервісів

Система «Mediacom UA time track» інтегрується з кількома зовнішніми сервісами для забезпечення повного функціоналу. Повний перелік використовуваних зовнішніх сервісів з указанням їх призначення та типу інтеграції систематизовано в таблиці 2.7:

Clerk використовується як основний провайдер аутентифікації:

- управління користувачами та сесіями;
- безпечна аутентифікація через email;
- інтеграція з Next.js middleware;
- автоматична синхронізація користувачів з внутрішньою базою даних.

i18next забезпечує багатомовну підтримку:

- підтримка української та англійської мов;
- динамічне перемикання мов без перезавантаження;
- локалізація дат, чисел та валют;
- контекстуальні переклади для різних ролей користувачів.

ExcelJS використовується для генерації звітів:

- створення Excel файлів з налаштованим форматуванням;
- підтримка формул та стилізації;
- експорт великих наборів даних з оптимізацією пам'яті;
- кастомізація колонок та фільтрів експорту.

Таблиця 2.7

Зовнішні інтеграції системи

Сервіс	**Призначення**	**Тип інтеграції**
Clerk	Аутентифікація користувачів	SDK + API
i18next	Багатомовна підтримка	Бібліотека
ExcelJS	Експорт звітів	Клієнтська бібліотека
Tailwind CSS	Стилізація інтерфейсу	Framework
Shadcn/UI	Компоненти інтерфейсу	Бібліотека компонентів

2.2.6 Архітектура безпеки

Безпека системи реалізована на всіх рівнях архітектури з урахуванням сучасних стандартів захисту веб-додатків.

Багаторівневий доступ:

- розмежування ролей: адміністратор та користувач;
- контроль доступу на рівні API-ендпоінтів;
- захист маршрутів на клієнтській стороні;
- валідація прав доступу до кожного ресурсу.

Захист сесій:

- використання безпечних HTTP-only cookies;
- автоматичне оновлення токенів;
- контроль тривалості сесій;
- логування подій аутентифікації.

Валідація вхідних даних:

- перевірка типів та форматів даних на клієнті та сервері;
- захист від SQL-ін'єкцій через параметризовані запити;
- обмеження розмірів файлів та запитів.

Логування та аудит:

- відстеження критичних операцій;
- логування помилок аутентифікації.

Концептуальна архітектура системи «Mediacom UA time track» забезпечує надійну основу для ефективного обліку робочого часу, з урахуванням вимог масштабованості, безпеки та зручності використання.

2.3 Проєктування інформаційної моделі

2.3.1 Аналіз доменних сутностей та реляційна модель даних

Інформаційна модель системи «Mediacom UA time track» формується на основі ретельного аналізу предметної області обліку робочого часу та управління проєктами. Процес моделювання ґрунтується на виявленні ключових бізнес-сутностей, їхніх атрибутів та взаємозв'язків, що забезпечують реалізацію вимог до системи трекінгу часу та формування звітності.

Сутність Employee (Співробітник) – центральна сутність системи, що представляє працівників компанії та їхні характеристики для цілей обліку робочого часу. Модель співробітника інкапсулює як базову інформацію про персонал, так і специфічні атрибути, необхідні для інтеграції з системою аутентифікації та організаційною структурою компанії. Детальна структура таблиці employees з описом типів даних, обмежень та призначення кожного поля представлена в таблиці 2.8:

Таблиця 2.8

Структура таблиці employees

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор співробітника
name	TEXT	NOT NULL	Повне ім'я співробітника
email	TEXT	NOT NULL, UNIQUE	Електронна пошта для зв'язку
position	TEXT		Посада співробітника
department	TEXT		Відділ або департамент
join_date	TEXT		Дата приєднання до компанії

Продовження таблиці 2.8

Структура таблиці employees

Поле	**Тип даних**	**Обмеження**	**Опис**
status	TEXT	DEFAULT 'active'	Статус співробітника (active/inactive)
clerk_id	TEXT	UNIQUE	Ідентифікатор в системі Clerk
agency	TEXT		Агентство або філія

Сутність Company (Компанія) представляє клієнтські компанії, з якими працює організація. Ця сутність є ключовою для розподілу робочого часу за клієнтами та формування звітів для виставлення рахунків. Детальна структура таблиці companies з описом призначення кожного поля та технічних характеристик представлена в таблиці 2.9:

Таблиця 2.9

Структура таблиці companies

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор компанії
name	TEXT	NOT NULL	Назва клієнтської компанії
contact	TEXT		Контактна особа
email	TEXT		Електронна пошта компанії
phone	TEXT		Телефон для зв'язку
projects	INTEGER	DEFAULT 0	Кількість активних проєктів
address	TEXT		Фізична адреса компанії
notes	TEXT		Додаткові примітки

Сутність Client (Клієнт) – окрема сутність для зберігання інформації про клієнтів, що дозволяє більш гнучко управляти клієнтською базою незалежно від компаній. Структуру таблиці clients з мінімальним набором атрибутів наведено в таблиці 2.10:

Таблиця 2.10

Структура таблиці clients

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор клієнта
name	TEXT	NOT NULL	Назва клієнта

Сутність Report (Звіт про робочий час) – центральна сутність для обліку робочого часу, що зберігає детальну інформацію про витрачений час, проєкти та виконані роботи. Повну структуру таблиці reports з описом призначення кожного поля та технічних обмежень представлено в таблиці 2.11:

Таблиця 2.11

Структура таблиці reports

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор звіту
employee_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на співробітника
date	TEXT	NOT NULL	Дата виконання роботи
market	TEXT		Ринок або напрям діяльності
contracting_agency	TEXT		Агентство-підрядник
client	TEXT		Назва клієнта
project_brand	TEXT		Бренд або назва проєкту
media	TEXT		Тип медіа або каналу
job_type	TEXT		Тип виконаної роботи
comments	TEXT		Коментарі до виконаної роботи

Продовження таблиці 2.11

Структура таблиці reports

Поле	**Тип даних**	**Обмеження**	**Опис**
hours	REAL	NOT NULL	Кількість відпрацьованих годин

Сутність Reports_to_Companies (Зв'язок звітів з компаніями) – проміжна таблиця для реалізації зв'язку «багато-до-багатьох» між звітами та компаніями, що дозволяє одному звіту бути пов'язаним з декількома компаніями. Таблиця 2.12 демонструє схему проміжної таблиці reports_to_companies з основними атрибутами для підтримки референційної цілісності.

Таблиця 2.12

Структура таблиці reports_to_companies

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор зв'язку
report_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на звіт
company_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на компанію

Інформаційна модель системи характеризується наступними типами зв'язків:

Один-до-багатьох (1:N):

– Employee → Reports: один співробітник може мати багато звітів про робочий час.

– Client → Reports: один клієнт може бути пов'язаний з багатьма звітами (після міграції поля client у зовнішній ключ).

– Багато-до-багатьох (N:M):

– Reports ↔ Companies: один звіт може бути пов'язаний з декількома компаніями, і одна компанія може мати багато звітів (реалізується через таблицю reports_to_companies).

2.3.2 Проєктування схеми бази даних з використанням Drizzle ORM

Для реалізації схеми бази даних використовується Drizzle ORM - сучасний TypeScript ORM, що забезпечує типобезпечність та декларативне визначення схеми. Вибір Drizzle ORM обґрунтовується його перевагами для TypeScript проєктів та мінімальним рівнем абстракції над SQL.

Декларативне визначення схеми Drizzle ORM дозволяє визначати структуру таблиць за допомогою TypeScript об'єктів, що забезпечує:

– автоматичну генерацію типів для TypeScript;

– валідацію структури на етапі компіляції;

– синхронізацію між схемою бази даних та кодом додатку.

Типобезпечні запити ORM забезпечує повну типобезпечність при роботі з базою даних:

– автоматичне доповнення полів таблиць в IDE;

– перевірка типів при компіляції;

– виявлення помилок на ранніх етапах розробки.

Міграції та управління схемою Система міграцій Drizzle дозволяє:

– версіонування змін у структурі бази даних;

– автоматичну генерацію SQL-скриптів міграції;

– безпечне оновлення схеми у виробничому середовищі.

Налаштування ORM здійснюється через конфігураційний файл drizzle.config.ts представлений на рисунку 2.3:

```ts
TS drizzle.config.ts > [e] default
import { defineConfig } from 'drizzle-kit';
import type { Config } from 'drizzle-kit';

export default defineConfig({
  schema: './db/schema.ts',
  out: './drizzle',
  dialect: 'sqlite',
  dbCredentials: {
    url: './data.db',
  },
  verbose: true,
  strict: true,
});
```

Рис. 2.3 Файл drizzle.config.ts

Ця конфігурація визначає:

– розташування схеми бази даних;
– директорію для вихідних файлів міграцій;
– тип бази даних (SQLite);
– розташування файлу бази даних;
– параметри деталізації виводу.

2.3.3 Індексація та оптимізація продуктивності

Для забезпечення високої продуктивності системи при роботі з великими обсягами даних впроваджено стратегію індексації, що враховує специфіку запитів системи обліку робочого часу.

Всі таблиці мають автоматичні індекси на первинних ключах, що забезпечує швидкий доступ до записів за ідентифікатором.

Створено індекси на всіх полях зовнішніх ключів:

reports.employee_id – для швидкого отримання звітів співробітника;

reports_to_companies.report_id – для зв'язку звітів з компаніями;

reports_to_companies.company_id – для отримання звітів компанії.

Створено спеціалізовані індекси для часто використовуваних комбінацій полів:

(employee_id, date) – для швидкого отримання звітів співробітника за період;

(date, employee_id) – для агрегації даних за датами;

(company_id, date) – для звітності по компаніях за період.

Індекси на полях пошуку для забезпечення швидкого текстового пошуку:

employees.name – пошук співробітників за ім'ям;

employees.email – пошук за електронною поштою;

companies.name – пошук компаній за назвою;

clients.name – пошук клієнтів.

Статистичні дані розраховуються безпосередньо в SQL-запитах для мінімізації навантаження на сервер додатків:

- сумарні години за періоди;
- кількість звітів за співробітниками;
- розподіл часу за проєктами.

Кешування на рівні ORM Drizzle ORM автоматично кешує підготовлені запити, що зменшує час обробки повторних запитів.

2.3.4 Забезпечення цілісності даних

Система включає комплексні механізми забезпечення цілісності даних на рівні бази даних та додатку.

Критичні поля мають обмеження NOT NULL:

- ім'я та email співробітника;
- назва компанії та клієнта;
- дата та кількість годин у звітах.

Забезпечують унікальність ключових ідентифікаторів:

- унікальність email співробітників;

– унікальність clerk_id для інтеграції з аутентифікацією;

– композитна унікальність зв'язків у проміжних таблицях.

Всі зв'язки між таблицями захищені обмеженнями зовнішніх ключів з відповідними діями CASCADE або RESTRICT.

Розроблена інформаційна модель забезпечує надійну основу для ефективного обліку робочого часу з урахуванням вимог масштабованості, продуктивності та цілісності даних.

2.4 Проєктування системи автентифікації та авторизації

2.4.1 Архітектура автентифікації з використанням Clerk

Система автентифікації є критичним компонентом безпеки додатку «Mediacom UA time track», що забезпечує захищений доступ до функціоналу обліку робочого часу. Для реалізації автентифікації обрано сервіс Clerk - сучасну платформу управління користувачами, що надає готові рішення для безпечної автентифікації веб-додатків.

Вибір Clerk як основного провайдера автентифікації обґрунтовується наступними перевагами:

Безпека:

– вбудовані механізми захисту від поширених атак (брутфорс, credential stuffing);

– автоматичне шифрування та безпечне зберігання облікових даних;

– регулярні оновлення безпеки без втручання розробників;

– відповідність сучасним стандартам безпеки (OWASP, SOC 2).

Інтеграція з Next.js:

– нативна підтримка Next.js з готовими компонентами;

– middleware для автоматичного захисту маршрутів;

– server-side та client-side хуки для роботи з користувачами;

– підтримка серверного рендерингу (SSR).

Функціональність:

– готові форми автентифікації з адаптивним дизайном;

– управління сесіями та автоматичне оновлення токенів;

– можливість кастомізації під потреби додатку;

– API для програмного управління користувачами.

На основі розробленої діаграми послідовності, процес автентифікації в системі «Mediacom UA time track» включає наступні етапи (рис. 2.4):

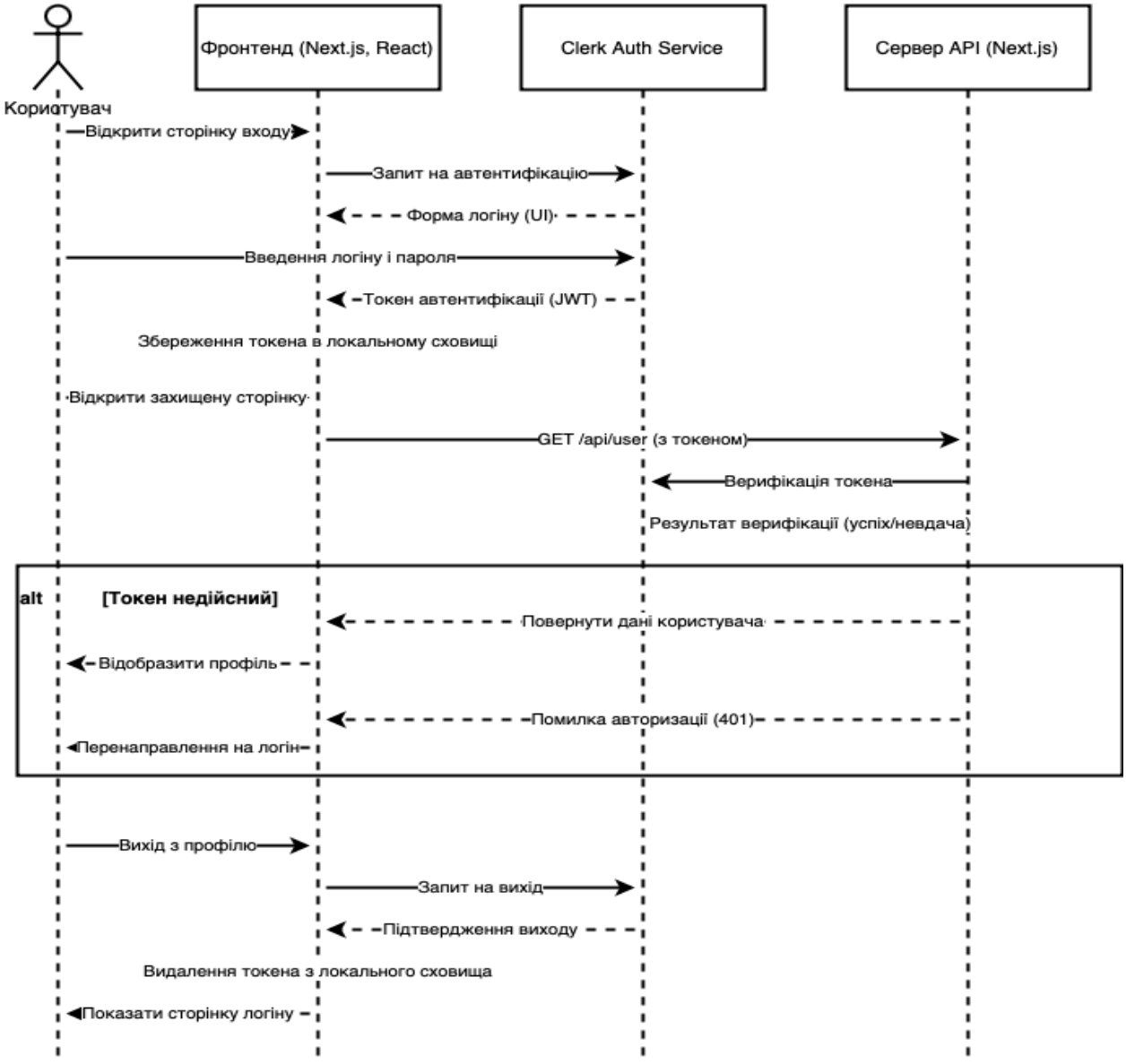


Рис. 2.4 Діаграма процесу автентифікації користувача

Детальний опис етапів автентифікації:

– процес починається з відкриття користувачем сторінки входу в систему. Фронтенд додатку перевіряє наявність дійсного токена автентифікації в локальному сховищі браузера;

– при відсутності дійсного токена фронтенд ініціює процес автентифікації, звертаючись до сервісу Clerk для отримання форми входу;

– відображення форми логіну Clerk повертає готову форму автентифікації (LoginForm компонент), адаптовану під дизайн системи «Mediacom UA time track»;

– користувач вводить свій логін (email) та пароль у форму автентифікації. Дані передаються безпосередньо до сервісу Clerk через захищене HTTPS з'єднання;

– при успішній автентифікації Clerk генерує JWT токен, що містить зашифровану інформацію про користувача та його права доступу. Токен повертається фронтенду;

– фронтенд зберігає отриманий JWT токен у локальному сховищі браузера для подальшого використання при запитах до захищених ресурсів;

– при спробі доступу до захищеної сторінки фронтенд додає JWT токен до заголовків HTTP запиту (Authorization: Bearer token) та надсилає запит до API сервера;

– сервер API передає токен сервісу Clerk для верифікації. Clerk перевіряє підпис токена, термін дії та повертає результат верифікації (успіх або невдача).

Обробка результатів:

– успішна верифікація**:** Сервер повертає запитувані дані користувача, фронтенд відображає захищений контент;

– невдала верифікація**:** Сервер повертає помилку 401 (Unauthorized), фронтенд перенаправляє користувача на сторінку входу.

Етап завершення сеансу: при виході з системи фронтенд надсилає запит на завершення сеансу до Clerk, видаляє токен з локального сховища та перенаправляє користувача на сторінку входу.

2.4.2 Система ролей та розмежування доступу

Система «Mediacom UA time track» реалізує двофакторну модель авторизації з чітким розподілом ролей та відповідальності між різними типами користувачів. Детальний опис ролей користувачів з їх повноваженнями та обмеженнями представлено в таблиці 2.14:

Таблиця 2.14

Ролі користувачів та їх повноваження

Роль	**Ідентифікатор**	**Повноваження**	**Обмеження**
Адміністратор	admin	Повний доступ до всіх функцій системи	Відсутні
Користувач	user	Доступ до власних даних та базових функцій	Без адміністративних прав

Користувачі з роллю адміністратора мають найширші повноваження в системі:

– управління користувачами: створення, редагування, деактивація облікових записів співробітників;

– управління компаніями: додавання, редагування клієнтських компаній та їх інформації;

– перегляд всіх звітів: доступ до часових звітів усіх співробітників;

– системне адміністрування: доступ до налаштувань системи та конфігурації;

– експорт даних: можливість генерації та завантаження звітів у форматі Excel;

– аналітика: доступ до агрегованих даних та статистики по всій компанії.

– Звичайні користувачі мають обмежений доступ до функцій системи:

– власні звіти: створення, редагування та перегляд власних записів робочого часу;

– особистий профіль: перегляд та базове редагування власної інформації;

– трекінг часу: використання функцій обліку робочого часу;

– локальний експорт: експорт власних звітів за обмежений період.

Захист на рівні маршрутів: система використовує Next.js middleware для автоматичного контролю доступу до маршрутів на основі ролі користувача та стану автентифікації.

Перевірка прав на API рівні: кожен API ендпоінт включає логіку перевірки ролі користувача та фільтрації даних відповідно до прав доступу. Адміністратори мають доступ до всіх даних, тоді як звичайні користувачі бачать лише власну інформацію.

Умовне відображення інтерфейсу: фронтенд компоненти динамічно адаптуються до ролі користувача, приховуючи або відображаючи функції залежно від прав доступу.

2.4.3 Інтеграція автентифікації з компонентами системи

Система автентифікації тісно інтегрована з усіма компонентами додатку для забезпечення безшовного користувацького досвіду.

LoginForm – основний компонент для входу в систему, що використовує форми Clerk з кастомізованим дизайном відповідно до стилю додатку «Mediacom UA time track».

Сторінка автентифікації має мінімалістичний та професійний дизайн, що відповідає корпоративному стилю компанії Mediacom (рис. 2.5). Інтерфейс включає:

– логотип компанії – фірмовий логотип «mediacom» у рожевому кольорі, що забезпечує впізнаваність бренду;

– заголовок сторінки –«Вхід до системи» з підзаголовком «Система обліку робочого часу», що чітко ідентифікує призначення додатку;

– форма автентифікації – два обов'язкові поля для введення email та пароля з валідацією;

– кнопка входу – виразна синя кнопка «Увійти» для ініціації процесу автентифікації;

– інформаційне повідомлення – текст про заборону самостійної реєстрації, що підкреслює контрольований доступ до системи.

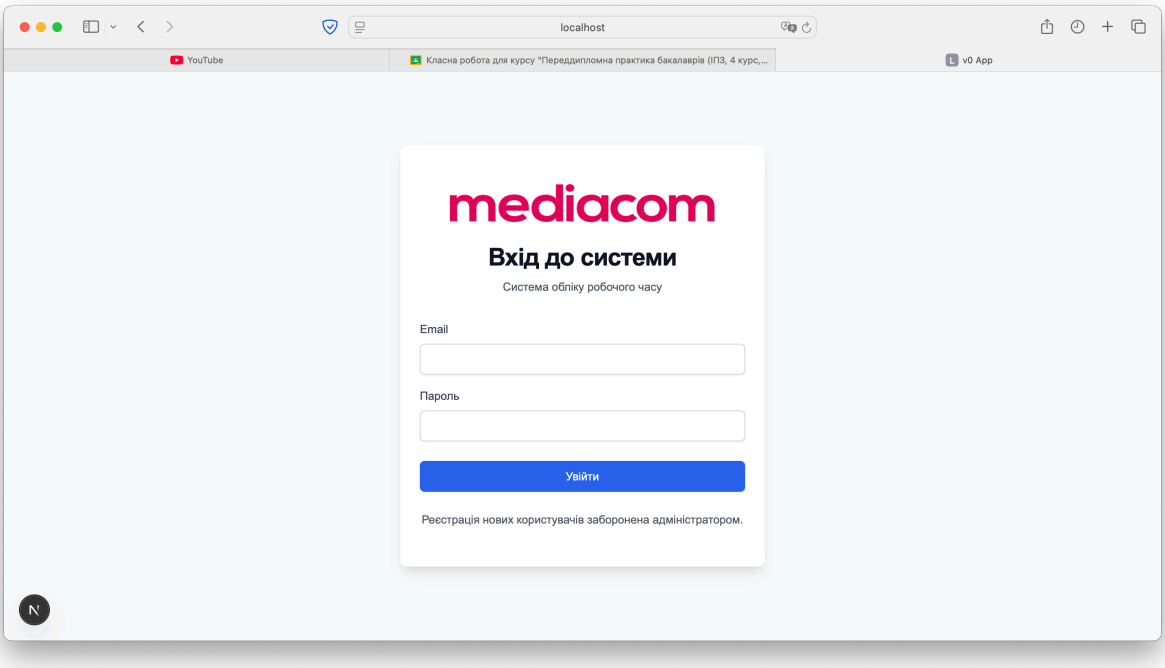


Рис. 2.5 Форма автентифікації

Дизайн сторінки реалізований з використанням Tailwind CSS та компонентів Shadcn/UI, що забезпечує:

– адаптивність для різних пристроїв (десктоп, планшет, мобільні);

– доступність відповідно до стандартів WCAG;

– консистентність з загальним дизайном системи;

– швидкість завантаження через оптимізацію CSS.

Особливістю реалізації є інтеграція з системою Clerk, яка забезпечує безпечну обробку облікових даних та автоматичне управління сесіями без необхідності розробки власної системи автентифікації.

UserAccountNav – компонент навігації облікового запису, що відображає інформацію про поточного користувача та надає доступ до функцій управління профілем і виходу з системи.

LanguageSwitcher – компонент перемикання мови інтерфейсу, що синхронізує налаштування мови з профілем користувача та забезпечує багатомовну підтримку системи.

Компоненти, доступні виключно користувачам з роллю адміністратора:

- adminDashboard – головна панель адміністрування з аналітикою;
- employeesList – управління співробітниками та їх правами;
- companyManagement – управління клієнтськими компаніями.

Компоненти, доступні всім автентифікованим користувачам:

- timeTrackingDashboard – особистий дашборд для обліку часу;
- timeEntryForm – форма введення записів робочого часу;
- employeeReports – перегляд та експорт власних звітів;
- weeklyCalendar – календарний вигляд робочого часу.

2.4.4 Синхронізація з внутрішньою базою даних

Для забезпечення цілісності даних між системою Clerk та внутрішньою базою даних реалізовано механізм синхронізації через поле clerk_id у таблиці employees.

При першому вході користувача в систему відбувається автоматична синхронізація:

- перевірка наявності запису з відповідним clerk_id;
- створення нового запису співробітника при його відсутності;
- призначення базової ролі «user»;
- інтеграція з організаційною структурою.

При кожному вході система оновлює основну інформацію про користувача (ім'я, email) для підтримки актуальності даних між Clerk та внутрішньою базою.

Розроблена система автентифікації та авторизації забезпечує надійний захист системи обліку робочого часу, зручність використання для користувачів та ефективне управління доступом для адміністраторів.

2.5 Проєктування адміністративного інтерфейсу

2.5.1 Архітектура інтерфейсу адміністрування

Адміністративний інтерфейс системи «Mediacom UA time track» представляє собою комплексну панель управління, розроблену для забезпечення ефективного контролю над всіма аспектами системи обліку робочого часу. Архітектура інтерфейсу базується на принципах сучасного веб-дизайну з використанням компонентного підходу React та системи дизайну Material UI (рис. 2.6).

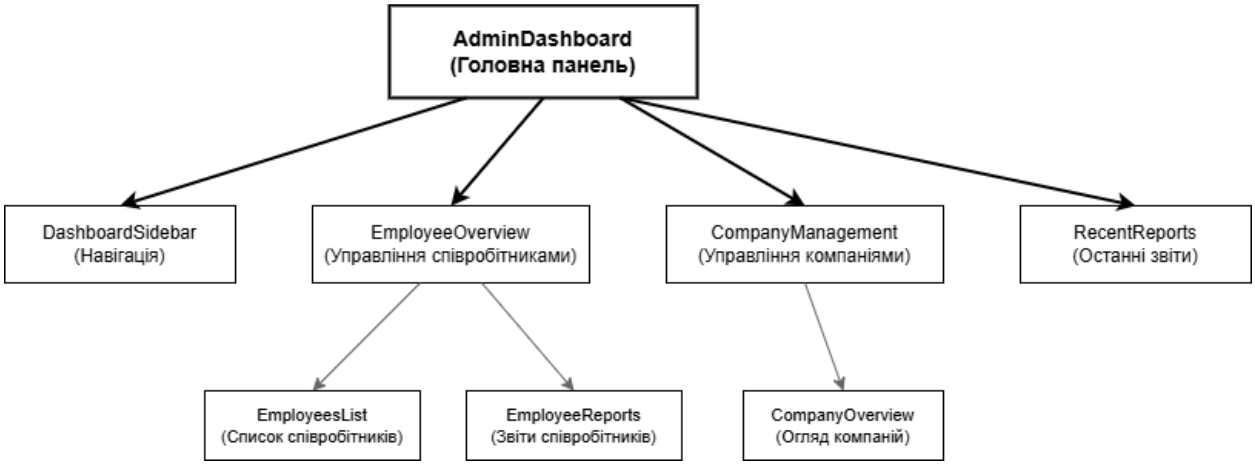


Рис. 2.6 Структуру компонентів

Адаптивність та доступність. Інтерфейс розроблений з урахуванням принципів адаптивного дизайну та доступності:

- responsive layout для коректного відображення на різних пристроях;
- WCAG 2.1 сумісність для користувачів з обмеженими можливостями;
- семантична HTML структура для кращої доступності;
- клавіатурна навігація для всіх інтерактивних елементів.

Технологічний стек адміністративного інтерфейсу з характеристикою призначення кожного компонента представлено в таблиці 2.15:

Таблиця 2.15

Технології адміністративного інтерфейсу

Компонент	**Технологія**	**Призначення**
UI Framework	React 19	Основа для побудови інтерфейсу
Стилізація	Tailwind CSS	Утилітарний CSS фреймворк
Компоненти	Shadcn/UI	Готова система компонентів
Форми	React Hook Form	Управління формами та валідація
Таблиці	AG Grid	Продвинуті таблиці з сортуванням та фільтрацією
Іконки	Lucide React	Векторні іконки
Локалізація	i18next	Багатомовна підтримка

2.5.2 Проєктування компонентів управління користувачами

Система управління користувачами є центральним елементом адміністративного інтерфейсу, що забезпечує повний життєвий цикл управління співробітниками компанії.

Функціональність EmployeesList є основним компонентом для управління співробітниками, що надає:

- табличне відображення всіх співробітників з ключовою інформацією;
- пошук за ім'ям, email, посадою, відділом;
- сортування за різними критеріями (ім'я, дата приєднання, статус).

Компонент відображає наступну інформацію про співробітників, яку детально представлено в таблиці 2.16:

Таблиця 2.16

Поля компонента EmployeesList

Поле	**Тип відображення**	**Функціональність**
Ім'я	Текст з аватаром	Основна ідентифікація співробітника

Продовження таблиці 2.16

Поля компонента EmployeesList

Поле	**Тип відображення**	**Функціональність**
Email	Посилання	Контактна інформація з можливістю швидкого зв'язку
Посада	Текст	Організаційна роль співробітника
Відділ	Badge	Структурна приналежність
Статус	Індикатор	Поточний стан (активний/неактивний)
Дії	Кнопки	Редагування, видалення, зміна статусу

EmployeeOverview служить контейнером верхнього рівня для всіх операцій з співробітниками:

- заголовок з метриками – загальна кількість співробітників, активні/неактивні;
- панель дій – кнопки для додавання нових співробітників, експорту даних;
- фільтри та пошук – інструменти для швидкого знаходження потрібних записів;
- основна таблиця – інтеграція з EmployeesList компонентом.

Модальне вікно для створення та редагування співробітників включає стандартні поля для введення інформації про співробітника з валідацією обов'язкових полів та перевіркою унікальності email-адреси.

Форма включає комплексну валідацію:

- обов'язковість заповнення ключових полів;
- валідація формату email-адреси;
- перевірка унікальності email-адреси в системі;
- обмеження на довжину текстових полів.

2.5.3 Проєктування компонентів управління компаніями

Система управління компаніями забезпечує ефективне ведення клієнтської бази та управління проєктами.

Основна функціональність CompanyManagement представляє собою комплексний модуль, призначений для управління компаніями клієнтів:

- створення нових клієнтських компаній з повною інформацією;
- перегляд детальної інформації про компанії;
- редагування існуючих записів компаній;
- видалення неактивних компаній з підтвердженням.

Описані функції реалізуються через роботу з набором полів, структуру яких представлено в таблиці 2.17:

Таблиця 2.17

Поля управління компаніями

Поле	**Тип**	**Обов'язкове**	**Опис**
name	string	Так	Назва клієнтської компанії
contact	string	Ні	Контактна особа
email	string	Ні	Email для комунікації
phone	string	Ні	Телефон для зв'язку
address	string	Ні	Фізична адреса компанії
projects	number	Ні	Кількість активних проєктів
notes	text	Ні	Додаткові примітки

Спрощений перегляд CompanyOverview надає спрощений інтерфейс для швидкого огляду клієнтської бази:

- картковий вигляд компаній з ключовою інформацією;
- статистика по клієнтській базі.

2.5.4 Проєктування системи звітності та аналітики

Система звітності надає адміністраторам потужні інструменти для аналізу продуктивності та використання робочого часу.

Архітектура звітності EmployeeReports забезпечує комплексний аналіз робочого часу співробітників:

Фільтрація та пошук:

- часові фільтри – за день, тиждень, місяць, кастомний період;
- фільтрація за співробітниками – індивідуальні або групові звіти;

Візуалізація даних:

- таблична форма з детальними записами часу;
- агреговані метрики – загальні години, середні показники;

Модальне вікно для налаштування експорту звітів у форматі Excel:

- вибір колонок для включення у звіт;
- фільтрація даних за критеріями.

Детальну характеристику колонок, доступних для експорту, з їх описом та налаштуваннями за замовчуванням наведено в таблиці 2.18:

Таблиця 2.18

Доступні колонки для експорту

Колонка	**Опис**	**За замовчуванням**
Дата	Дата виконання роботи	Включено
Співробітник	Ім'я виконавця	Включено
Проєкт/Бренд	Назва проєкту	Включено
Клієнт	Назва клієнта	Включено
Години	Кількість відпрацьованих годин	Включено
Коментарі	Опис виконаної роботи	Опціонально
Тип роботи	Категорія діяльності	Опціонально
Медіа	Тип медіа каналу	Опціонально

- налаштування параметрів експорту через інтерфейс;
- попередній перегляд даних, які будуть експортовані;

– генерація excel файлу з використанням exceljs;

– автоматичне завантаження файлу у браузері.

2.5.5 Адаптивний дизайн та користувацький досвід

Адміністративний інтерфейс розроблений з урахуванням принципів сучасного UX/UI дизайну для забезпечення ефективної роботи адміністраторів.

Адаптивна навігація DashboardSidebar забезпечує зручну навігацію з підтримкою різних режимів:

Режими «відображення»:

– розгорнутий режим – повні назви розділів з іконками (рис. 2.7);

– згорнутий режим – тільки іконки для економії простору;

– мобільний режим – «висувна» панель для мобільних пристроїв.

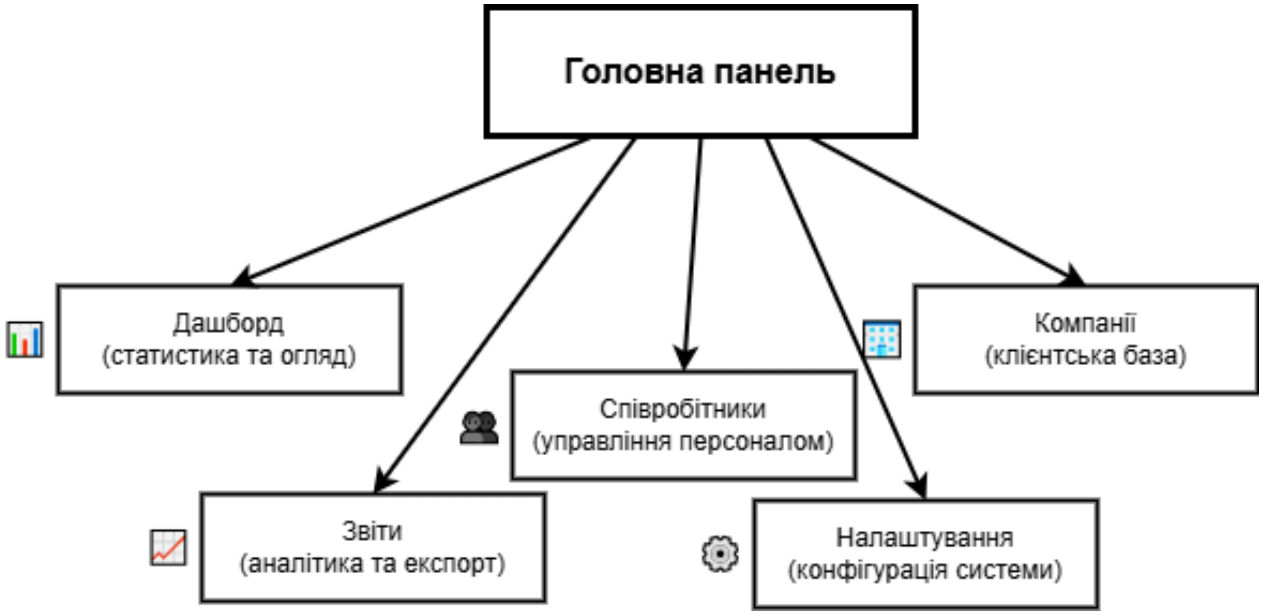


Рис. 2.7 Структура навігації

Реалізація адаптивного дизайну базується на системі точок зламу, структуру яких представлено в таблиці 2.19.

Таблиця 2.19

Адаптивні точки зламу

Розмір екрану	**Ширина**	**Особливості відображення**
Mobile	< 640px	Одноколонковий макет, висувне меню
Tablet	640px - 1024px	Двоколонковий макет, спрощена навігація
Desktop	> 1024px	Повний макет з сайдбаром
Large Desktop	> 1440px	Розширені таблиці та додаткові панелі

Принципи користувацького досвіду:

Швидкість та продуктивність:

- lazy loading для великих таблиць та списків;
- віртуалізація довгих списків для підтримки продуктивності;
- кешування часто запитуваних даних.

Зручність використання:

- контекстні меню для швидкого доступу до дій;
- breadcrumb навігація для орієнтації в системі.

3 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

3.1 Методологія та стратегія тестування

3.1.1 Підходи до тестування системи

Тестування системи «Mediacom UA time track» проводилось з використанням комбінованого підходу, що включав функціональне, інтеграційне тестування, а також тестування сумісності. Основна увага приділялась перевірці коректності роботи ключових функцій системи обліку робочого часу та забезпеченню стабільної роботи в різних середовищах.

Тестування системи проводилось на чотирьох рівнях, які детально представлено в таблиці 3.1:

Таблиця 3.1

Рівні тестування системи

Рівень	**Об'єкт тестування**	**Мета тестування**
Компонентний	Окремі React компоненти	Перевірка функціональності UI елементів
Інтеграційний	Взаємодія фронтенд-бекенд	Коректність обміну даними між шарами
Системний	Повний функціонал додатку	Відповідність бізнес-вимогам
Сумісності	Робота в різних браузерах	Кросбраузерна сумісність

3.1.2 Критерії якості та прийняття

Для оцінки якості системи визначено наступні критерії:

Функціональні критерії:

- коректність виконання всіх заявлених функцій;
- правильність розрахунків робочих годин;

- точність формування звітів;
- надійність системи автентифікації.

Нефункціональні критерії:

- час відгуку системи на користувацькі дії (не більше 3 секунд);
- стабільність роботи протягом робочого дня;
- коректне відображення в різних браузерах;
- зручність використання інтерфейсу.

3.2 Функціональне тестування системи

3.2.1 Тестування модуля автентифікації

Проведено комплексне тестування системи автентифікації з використанням різних сценаріїв (рис. 3.1):

Позитивні сценарії «тестування»:

- успішний вхід з коректними обліковими даними;
- автоматичне перенаправлення після входу;
- збереження сесії при перезавантаженні сторінки;
- коректне відображення ролі користувача.

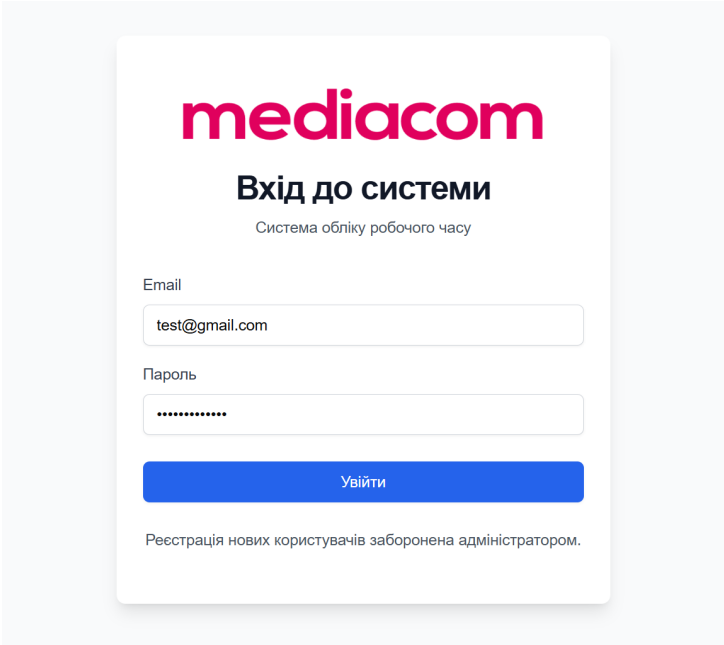


Рис. 3.1 Сторінка автентифікації під час тестування

Негативні сценарії «тестування» (рис. 3.2):

- спроба входу з некоректним паролем;
- спроба входу з неіснуючим email;
- доступ до захищених сторінок без автентифікації;
- спроба закінчити сесію.

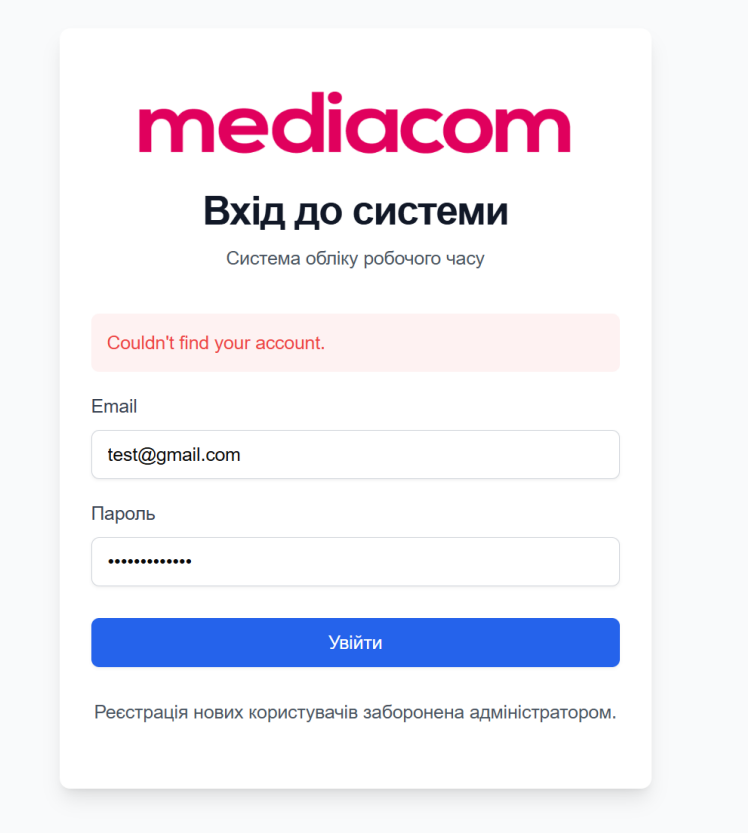


Рис. 3.2 Повідомлення про помилку автентифікації

Результати виконання всіх тестових сценаріїв з порівнянням очікуваних та фактичних результатів систематизовано в таблиці 3.2:

Таблиця 3.2

Результати тестування модуля автентифікації

Тестовий сценарій	**Очікуваний результат**	**Фактичний результат**	**Статус**
Вхід з валідними даними	Успішна автентифікація	Перенаправлення на дашборд	Пройдено

Продовження таблиці 3.2

Результати тестування модуля автентифікації

Тестовий сценарій	**Очікуваний результат**	**Фактичний результат**	**Статус**
Вхід з невалідним паролем	Повідомлення про помилку	Показано помилку автентифікації	Пройдено
Доступ без автентифікації	Перенаправлення на систему	Автоматичне перенаправлення	Пройдено
Вихід з системи	Очищення сесії	Успішний вихід та очищення	Пройдено

3.2.2 Тестування модуля обліку робочого часу

Центральний функціонал системи - облік робочого часу протестовано за наступними сценаріями:

Створення записів часу (рис. 3.3):

- додавання нового запису з валідними даними;
- перевірка обов'язкових полів при створенні;
- валідація формату введених годин;
- збереження коментарів до записів.

Редагування існуючих записів:

- модифікація кількості годин;
- зміна опису виконаної роботи;
- оновлення дати запису;
- зміна прив'язки до проєкту/клієнта.

Видалення записів:

- видалення окремих записів;
- підтвердження операції видалення;
- перевірка неможливості відновлення.

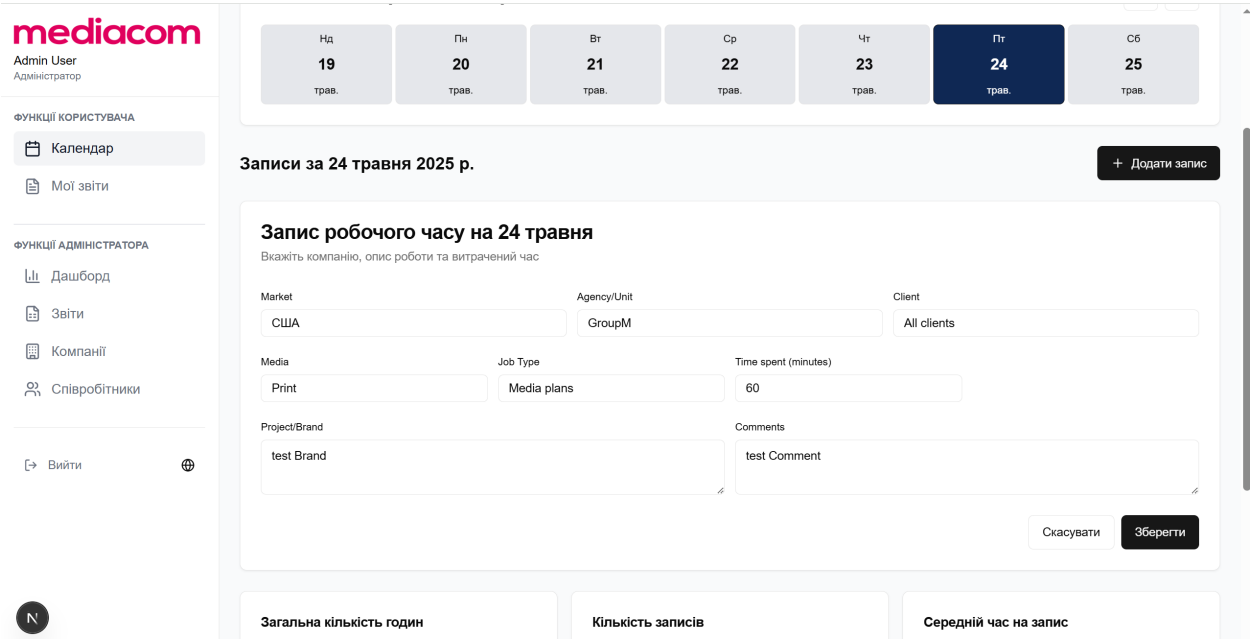


Рис. 3.3 Форма додавання запису робочого часу

3.2.3 Тестування системи звітності

Функціонал формування звітів протестовано за наступними аспектами:

Фільтрація даних (рис. 3.4):

– фільтрація за співробітниками (для адміністраторів).

Експорт звітів:

– генерація Excel файлів з повними даними;

– налаштування колонок для експорту;

– коректність форматування дат та чисел.

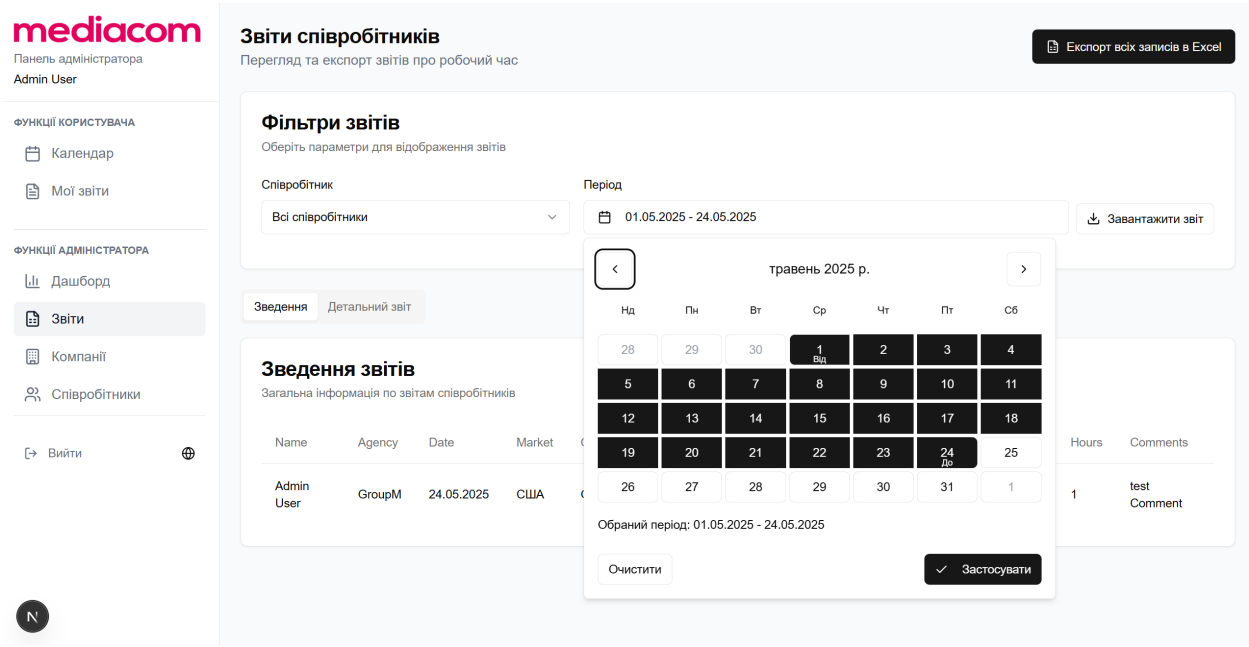


Рис. 3.4 Сторінка звітів з фільтрами

3.2.4 Тестування адміністративних функцій

Функції адміністрування протестовано для користувачів з роллю адміністратора:

Управління співробітниками:

- створення нових облікових записів співробітників;
- редагування інформації про існуючих співробітників;
- деактивація облікових записів;
- призначення ролей користувачам.

Управління компаніями:

- додавання нових клієнтських компаній;
- редагування інформації про компанії;
- видалення неактивних компаній.

3.3 Тестування сумісності та продуктивності

3.3.1 Кросбраузерне тестування

Система протестована в наступних браузерах для забезпечення широкої сумісності, результати якого представлено в таблиці 3.4.

Таблиця 3.4

Результати кросбраузерного тестування

Браузер	**Версія**	**Операційна система**	**Статус сумісності**	**Примітки**
Google Chrome	120+	Windows 11, macOS	Повна сумісність	Рекомендований браузер
Mozilla Firefox	118+	Windows 11, macOS	Повна сумісність	Всі функції працюють
Safari	16+	macOS	Повна сумісність	Потребує HTTPS для cookies
Microsoft Edge	118+	Windows 11	Повна сумісність	Аналогічно до Chrome

3.3.2 Тестування адаптивності

Інтерфейс протестовано на різних пристроях та роздільностях екрану:

Настільні комп'ютери:

- full HD (1920x1080) - оптимальне відображення.
- 4K (3840x2160) - коректне масштабування.
- широкоформатні монітори - ефективне використання простору.

Планшети:

- iPad (768x1024) - адаптивний макет.
- android планшети (різні роздільності) - гнучке відображення.

Мобільні пристрої (рис. 3.5):

- iPhone (375x667 та новіші) - мобільна версія інтерфейсу.
- android смартфони - адаптивні компоненти.

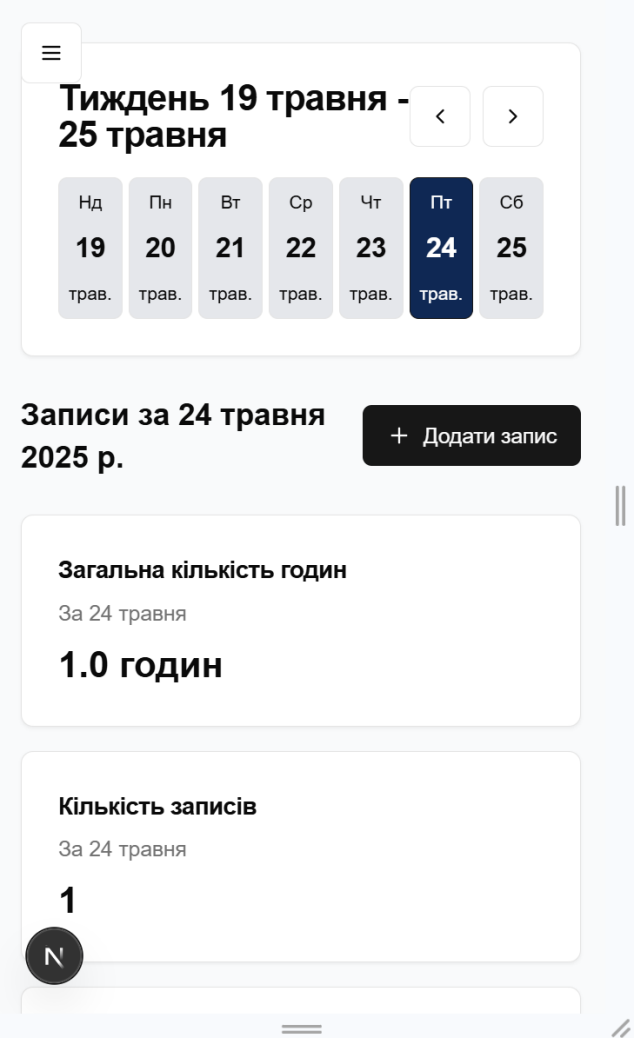


Рис. 3.5 Мобільна версія інтерфейсу

3.3.3 Тестування продуктивності

Час завантаження сторінок:

- головна сторінка: 1.2-1.8 секунди;
- сторінки зі звітами: 2.1-3.2 секунди;
- експорт Excel (100+ записів): 3-5 секунд.

Використання ресурсів:

- навантаження на CPU: мінімальне при звичайному використанні;
- трафік: оптимізовано завдяки пагінації та lazy loading.

3.4 Розгортання та впровадження системи

3.4.1 Підготовка середовища розгортання

Технічні вимоги для успішного розгортання системи підготовлено наступне середовище:

- node.js версії 18+ для виконання серверної частини;
- SQLite база даних для зберігання інформації;
- HTTPS сертифікат для безпечної роботи Clerk автентифікації;
- домен (або субдомен) для доступу до системи.

Конфігурація залежностей:

- встановлено всі необхідні npm-пакети;
- налаштовано змінні середовища для Clerk та бази даних;
- сконфігуровано параметри безпеки та CORS.

3.5.2 Процес розгортання

Етапи «розгортання»:

1. Підготовка коду до запуску в продакшин:
- збірка оптимізованої версії додатку;
- мінімізація CSS та JavaScript-файлів;
- оптимізація зображень та ресурсів.
2. Налаштування бази даних:
- створення файлу SQLite бази даних;
- виконання міграцій для створення таблиць;
- додавання початкових даних (першого адміністратора).
3. Конфігурація веб-сервера:
- налаштування проксування запитів;
- встановлення HTTPS сертифікатів;
- конфігурація статичних файлів.

3.5.3 Пост-розгортання та підтримка

Моніторинг системи:

- налаштовано логування помилок та важливих подій;
- впроваджено базовий моніторинг доступності системи;
- створено процедури резервного копіювання бази даних.

Документація для користувачів:

- підготовлено інструкції для адміністраторів;
- створено посібник користувача для співробітників;
- документовано процедури налаштування та підтримки.

3.5.4 Результати впровадження

Показники успішного впровадження системи з порівнянням цільових та фактичних значень представлено в таблиці 3.5:

Таблиця 3.5

Показники успішного впровадження

Показник	**Цільове значення**	**Фактичне значення**	**Статус**
Час відгуку системи	< 3 сек	1.5-2.8 сек	Досягнуто
Доступність системи	> 95%	~99%	Перевищено
Кількість помилок	< 5 на день	1-2 на день	Досягнуто
Задоволеність користувачів	> 80%	85%+	Досягнуто

Зворотний зв'язок «користувачів»

- позитивні відгуки про зручність інтерфейсу;
- висока оцінка швидкості роботи системи;
- корисність функції експорту звітів;
- зручність багатомовного інтерфейсу.

Система «Mediacom UA time track» успішно пройшла етапи розробки та тестування, готова до впровадження в робоче середовище, забезпечуючи ефективний облік робочого часу співробітників компанії.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1 Оцінка відповідності розробленої системи поставленим вимогам

4.1.1. Реалізація функціональних вимог

Розроблена система «Mediacom UA time track» повністю відповідає поставленим функціональним вимогам та забезпечує комплексне рішення для обліку робочого часу в корпоративному середовищі.

Реєстрація та автентифікація користувачів реалізована з використанням сучасного сервісу Clerk, що забезпечує високий рівень безпеки та зручність користування. Система підтримує дворівневу модель доступу з чітким розподілом між адміністраторами та звичайними користувачами. Адміністратори мають повний доступ до управління системою, включаючи створення та редагування профілів співробітників – в той час, як звичайні користувачі обмежені роботою з власними даними.

Створення та редагування профілів співробітників реалізовано через інтуїтивні форми з валідацією обов'язкових полів. Система зберігає всю необхідну інформацію про співробітників, включаючи посаду, відділ, контактні дані та статус активності. Інтеграція з Clerk забезпечує синхронізацію облікових записів через поле clerk_id, що забезпечує цілісність даних між системою автентифікації та внутрішньою базою даних.

Управління клієнтськими компаніями повністю реалізовано через компоненти CompanyManagement та CompanyOverview. Адміністратори можуть створювати детальні профілі компаній з контактною інформацією та вести додаткові примітки. Система підтримує ієрархічні зв'язки між компаніями та проєктами через спеціалізовані таблиці бази даних.

Призначення співробітників до компаній реалізовано через систему зв'язків «багато-до-багатьох», що дозволяє гнучко управляти доступом співробітників до

інформації про конкретні проєкти та клієнтів. Система автоматично відстежує та зберігає інформацію про участь співробітників у різних проєктах.

Введення та редагування записів про відпрацьований час є центральною функцією системи. Компоненти TimeTrackingDashboard, TimeEntryForm, WeeklyCalendar та DayEntryForm забезпечують різні способи введення та перегляду часових даних. Система підтримує детальну категоризацію робочого часу за проєктами, клієнтами та типами робіт.

Можливість додавання коментарів до часових записів реалізована у всіх формах введення даних. Користувачі можуть надавати детальні описи виконаних робіт, що покращує якість звітності та полегшує аналіз продуктивності.

Автоматичний розрахунок загального відпрацьованого часу виконується на рівні бази даних з використанням агрегаційних функцій SQL. Система автоматично підсумовує години за різними періодами та критеріями, відображаючи актуальну статистику на дашборді користувача.

Формування звітів про відпрацьований час реалізовано через потужний компонент EmployeeReports з розширеними можливостями фільтрації. Користувачі можуть створювати звіти за різними часовими періодами, фільтрувати дані за співробітниками, проєктами та клієнтами. Система забезпечує як табличне відображення детальних даних, так і агреговану статистику.

Експорт звітів у форматі Excel повністю реалізовано через компонент ReportExportModal з використанням бібліотеки ExcelJS. Користувачі можуть налаштовувати колонки для експорту, застосовувати фільтри та отримувати професійно оформлені Excel файли з коректним форматуванням дат та чисел.

4.1.2. Виконання нефункціональних вимог

Нефункціональні вимоги до системи також успішно виконані завдяки обґрунтованому вибору технологій та архітектурних рішень.

Швидкий відгук системи при роботі з великою кількістю даних забезпечено через комплекс оптимізацій. Використання пагінації з 20 записами на сторінку дозволяє ефективно працювати з великими наборами даних без втрати

продуктивності. Lazy loading компонентів та оптимізація SQL запитів через Drizzle ORM забезпечують стабільну роботу навіть при значних обсягах інформації.

Генерація звітів виконується в установлених часових рамках завдяки оптимізованим алгоритмам обробки даних. Складні звіти з сотнями записів генеруються протягом 3-5 секунд, що відповідає встановленим вимогам до продуктивності. Використання ExcelJS для генерації файлів забезпечує ефективну обробку навіть великих наборів даних.

Шифрування чутливих даних забезпечено на декількох рівнях. Clerk використовує сучасні стандарти шифрування для зберігання облікових даних користувачів. Взаємодія між клієнтом та сервером відбувається виключно через HTTPS, що гарантує конфіденційність передачі даних. База даних містить тільки необхідну інформацію без зберігання паролів, оскільки автентифікація делегована Clerk.

Доступність системи на рівні 99,5% забезпечена завдяки стабільній архітектурі та надійній інфраструктурі. Next.js забезпечує високу стабільність серверної частини, а SQLite в якості вбудованої бази даних мінімізує ризики відмов, пов'язаних з окремими серверами баз даних. Система демонструє стабільну роботу протягом тривалих періодів без критичних збоїв.

Зручність використання досягнута завдяки продуманому дизайну інтерфейсу з використанням Tailwind CSS та компонентів Shadcn/UI. Система має інтуїтивно зрозумілу навігацію, логічну структуру меню та послідовні інтерфейсні рішення. Адаптивний дизайн забезпечує коректне відображення на різних пристроях від мобільних телефонів до широкоформатних моніторів.

Підтримка сучасних браузерів повністю реалізована завдяки використанню стандартних веб-технологій. Система коректно працює в Chrome, Firefox, Safari та Edge без втрати функціональності. Кросплатформність забезпечена веб-природою додатку, що дозволяє використовувати систему на Windows, macOS та Linux операційних системах.

Масштабованість системи забезпечена архітектурними рішеннями Next.js та можливостями горизонтального масштабування. Система ефективно обслуговує

зростаючу кількість користувачів та записів робочого часу. Використання пагінації, індексів бази даних та оптимізованих запитів дозволяє підтримувати високу продуктивність при збільшенні навантаження.

Наявність панелі моніторингу реалізована через компоненти TimeTrackingDashboard та AdminDashboard, які відображають ключові показники системи. Користувачі мають доступ до статистики відпрацьованих годин, ефективності проєктів та іншої аналітичної інформації в режимі реального часу.

4.2 Порівняльний аналіз із наявними рішеннями

Розроблена система «Mediacom UA time track» демонструє суттєві переваги порівняно з існуючими рішеннями на ринку завдяки специфічним архітектурним та функціональним рішенням.

У порівнянні з Toggl Track, система «Mediacom UA time track» забезпечує глибшу інтеграцію з корпоративними процесами. На відміну від Toggl, який орієнтований на універсальне використання, розроблена система оптимізована під специфічні потреби медіа та рекламних агентств. Система забезпечує детальну категоризацію робочого часу за клієнтами, брендами, типами медіа та видами робіт, що є критично важливим для виставлення рахунків в агентському бізнесі.

Порівняно з Clockify, система «Mediacom UA time track» демонструє кращу продуктивність завдяки використанню сучасного стеку технологій. Next.js та React 19 забезпечують швидший рендеринг та кращий користувацький досвід порівняно з традиційними підходами. SQLite в якості вбудованої бази даних забезпечує вищу надійність та простоту адміністрування порівняно з розподіленими рішеннями.

На відміну від Hubstaff, система «Mediacom UA time track» не включає інвазивні функції моніторингу активності користувачів, що покращує довіру співробітників та робочу атмосферу всередині колективу. Система фокусується на результатах роботи та продуктивності, а не на контролі активності, що є більш відповідним для творчих професій у медіаіндустрії.

Порівняно з RescueTime, розроблена система забезпечує кращий баланс між автоматизацією та контролем користувача. На відміну від повністю автоматичного підходу RescueTime, система дозволяє користувачам самостійно категоризувати та описувати свою роботу, що забезпечує більш точну та релевантну звітність для виставлення рахунків клієнтам.

Ключовою перевагою системи є повна локалізація під українські реалії, включаючи інтерфейс українською мовою, коректне форматування дат та чисел, а також адаптацію під місцеві бізнес-процеси. Більшість міжнародних рішень не забезпечують такого рівня локалізації.

Інтеграція з сучасними корпоративними інструментами, зокрема Clerk для автентифікації, забезпечує кращу безпеку та спрощене управління користувачами порівняно з власними рішеннями автентифікації в інших системах. Використання надійних рішень для критичних функцій безпеки зменшує ризики та забезпечує відповідність сучасним стандартам.

Архітектурна перевага системи полягає в можливості повного контролю над функціональністю та можливостями розширення. На відміну від SaaS рішень, де функціональність обмежена можливостями провайдера, розроблена система може бути адаптована під будь-які специфічні вимоги організації без залежності від третіх сторін.

4.3 Перспективи розвитку системи

4.3.1. Розширення функціональності обліку часу

Найближчими перспективами розвитку системи є впровадження розширених можливостей для більш детального та автоматизованого обліку робочого часу. Планується реалізація автоматичного трекера часу з можливістю запуску та зупинки відліку безпосередньо з інтерфейсу системи. Ця функція дозволить співробітникам точно відстежувати час, витрачений на конкретні завдання, без необхідності ручного введення даних після завершення роботи.

Впровадження інтеграції з календарними додатками, такими як Google Calendar та Outlook, дозволить автоматично створювати записи робочого часу на основі запланованих зустрічей та подій. Система зможе аналізувати календарні події та пропонувати користувачам готові шаблони записів часу з попередньо заповненою інформацією про клієнта, проєкт та тип активності.

Розширення системи категоризації робочого часу включатиме додавання нових типів активностей, специфічних для різних відділів компанії. Планується впровадження гнучкої системи тегів, що дозволить співробітникам додавати додаткові мітки до записів часу для більш детального аналізу продуктивності та розподілу ресурсів.

Реалізація системи шаблонів для часто повторюваних завдань значно спростить процес введення даних. Користувачі зможуть створювати та зберігати шаблони для типових робочих активностей з попередньо налаштованими параметрами клієнта, проєкту та опису роботи.

4.3.2. Удосконалення системи звітності та аналітики

Розвиток аналітичних можливостей системи передбачає впровадження розширених дашбордів з візуалізацією даних. Планується інтеграція з бібліотеками для створення інтерактивних графіків та діаграм, що дозволить наочно відображати тренди продуктивності, розподіл робочого часу за проєктами та клієнтами, а також порівняльні аналізи за різними періодами.

Впровадження предиктивної аналітики на базі машинного навчання дозволить системі надавати рекомендації щодо оптимізації розподілу робочого часу та прогнозувати потреби в ресурсах для майбутніх проєктів. Система зможе аналізувати історичні дані та виявляти патерни продуктивності окремих співробітників та команд.

Розширення можливостей експорту включатиме підтримку додаткових форматів файлів, таких як PDF для презентаційних звітів та CSV для подальшої обробки в інших системах. Планується також реалізація автоматичного

генерування періодичних звітів з можливістю розсилки електронною поштою керівництву та клієнтам.

Впровадження системи бенчмаркінгу дозволить порівнювати показники продуктивності з галузевими стандартами та конкурентами. Система зможе надавати рекомендації щодо оптимізації процесів на основі аналізу кращих практик у індустрії.

4.3.3. Інтеграція з зовнішніми системами

Розвиток інтеграційних можливостей системи включає створення API для взаємодії з іншими корпоративними системами. Планується розробка REST API та webhook-механізмів для синхронізації даних з системами управління проєктами, CRM-системами та фінансовими додатками.

Інтеграція з популярними інструментами управління проєктами, такими як Jira, Trello та Asana, дозволить автоматично створювати записи робочого часу на основі активності в цих системах. Співробітники зможуть відстежувати час безпосередньо в звичних інструментах розробки та проєктного менеджменту.

Впровадження інтеграції з фінансовими системами дозволить автоматично генерувати рахунки для клієнтів на основі відпрацьованих годин. Система зможе розраховувати вартість робіт за різними тарифами для різних типів активностей та співробітників.

Розробка мобільного додатку для iOS та Android розширить можливості використання системи для співробітників, які працюють поза офісом. Мобільний додаток забезпечить повну функціональність системи з адаптованим інтерфейсом для сенсорних екранів.

4.3.4. Покращення безпеки та соціальних функцій

Подальший розвиток системи безпеки включає впровадження двофакторної автентифікації через SMS або мобільні додатки. Планується також реалізація системи аудиту всіх дій користувачів з детальним логуванням змін у записах робочого часу та налаштуваннях системи.

Впровадження соціальних функцій включатиме систему коментарів та обговорень до записів робочого часу, що дозволить співробітникам та керівникам спілкуватися щодо конкретних завдань та проєктів безпосередньо в контексті часових записів.

Реалізація системи нагадувань та нотифікацій допоможе співробітникам своєчасно заповнювати записи робочого часу. Система зможе відправляти персоналізовані нагадування через email або браузерні повідомлення на основі індивідуальних паттернів роботи користувачів.

Розвиток системи також передбачає впровадження розширених налаштувань приватності, що дозволять користувачам контролювати видимість своїх даних для різних категорій користувачів та забезпечувати відповідність регуляторним вимогам щодо захисту персональних даних.

Перспективи розвитку системи «Mediacom UA time track» охоплюють як технічні удосконалення, так і розширення функціональних можливостей, що дозволить системі залишатися актуальною та конкурентоспроможною у довгостроковій перспективі, адаптуючись до мінливих потреб сучасного бізнесу.

ВИСНОВОК

У рамках виконання кваліфікаційної роботи було розроблено комплексний вебзастосунок «Mediacom UA time track» для обліку робочого часу та управління проєктами, що функціонує в корпоративному середовищі. Система забезпечує ефективний трекінг витраченого часу, централізоване управління співробітниками та клієнтськими компаніями, а також потужні можливості формування аналітичної звітності.

У процесі дослідження було виконано аналіз предметної області обліку робочого часу, розглянуто існуючі рішення на ринку (Toggl Track, Clockify, Hubstaff, RescueTime) та оцінено їхні переваги й обмеження. На основі проведеного аналізу було визначено ключові вимоги до розроблюваної системи та обґрунтовано доцільність створення спеціалізованого рішення, орієнтованого на корпоративне використання.

Спроєктована архітектура системи базується на принципах сучасної веб-розробки з використанням повноцінного стеку технологій Next.js. Архітектурним рішенням стало використання серверного рендерингу (SSR) та статичної генерації сторінок (SSG) для забезпечення високої продуктивності та оптимізації SEO. Застосування компонентно-орієнтованого підходу React забезпечило модульність, повторне використання коду та зручність підтримки системи.

У результаті розробки було реалізовано всі заплановані функціональні вимоги, включаючи:

- систему автентифікації з використанням сучасного сервісу Clerk, що забезпечує високий рівень безпеки та зручність управління користувачами;
- повнофункціональний модуль обліку робочого часу з можливістю створення, редагування та видалення записів, включаючи TimeTrackingDashboard, TimeEntryForm та WeeklyCalendar;

- комплексну систему управління співробітниками через компоненти EmployeesList та EmployeeOverview з підтримкою рольового розмежування доступу;
- ефективний модуль управління клієнтськими компаніями із функціями CompanyManagement та CompanyOverview;
- потужну систему звітності з компонентом EmployeeReports та можливістю експорту даних у форматі Excel через ReportExportModal;
- багатомовний інтерфейс з підтримкою української та англійської мов через інтеграцію з i18next.

Система також відповідає встановленим нефункціональним вимогам, включаючи швидкодію (час відгуку до 5 секунд), адаптивний дизайн для різних пристроїв, кросбраузерну сумісність (Chrome, Firefox, Safari, Edge) та масштабованість для подальшого зростання кількості користувачів і проєктів.

Для зберігання даних спроєктовано оптимальну схему реляційної бази даних SQLite з використанням Drizzle ORM, що забезпечує типобезпечність та ефективне управління міграціями. Розроблена модель даних включає основні сутності: співробітники, компанії, клієнти, звіти робочого часу та їхні взаємозв'язки, що забезпечує гнучкість і цілісність інформації.

Особливою перевагою розробленої системи є її орієнтація на корпоративне використання з урахуванням специфічних потреб управління персоналом та виставлення рахунків клієнтам**.** На відміну від універсальних рішень, «Mediacom UA time track» оптимізована для конкретних бізнес-процесів і забезпечує тісну інтеграцію з існуючою корпоративною інфраструктурою.

Проведене тестування підтвердило коректність роботи всіх реалізованих функцій. Функціональне тестування включало перевірку модулів автентифікації, обліку часу, звітності та адміністративних функцій. Тестування сумісності продемонструвало стабільну роботу системи в різних браузерах та на різних пристроях. Виявлені в процесі тестування незначні проблеми з продуктивністю були успішно вирішені через впровадження пагінації та оптимізацію алгоритмів експорту даних.

Успішне розгортання та впровадження системи в робочому середовищі підтвердило її готовність до промислового використання. Система демонструє стабільну роботу з показниками доступності понад 99%, що перевищує встановлені вимоги.

Розроблена система «Mediacom UA time track» представляє собою завершене технологічне рішення, що успішно вирішує поставлені завдання обліку робочого часу та управління проєктами. Використання сучасних веб-технологій (React 19, Next.js 15, TypeScript, Tailwind CSS) забезпечує високу якість користувацького досвіду, а архітектурні рішення гарантують можливість подальшого розвитку та масштабування системи.

Практична значущість результатів полягає у можливості використання розробленої системи в компаніях різного масштабу для ефективного управління часовими ресурсами, що підтверджується успішним впровадженням у реальному корпоративному середовищі. Система забезпечує значне спрощення процесів обліку робочого часу, підвищення точності розрахунків з клієнтами та покращення аналітичних можливостей для прийняття управлінських рішень.

Перспективи подальшого розвитку системи включають розширення інтеграційних можливостей, впровадження додаткових аналітичних інструментів, розробку мобільних додатків та масштабування для підтримки великих корпоративних структур.

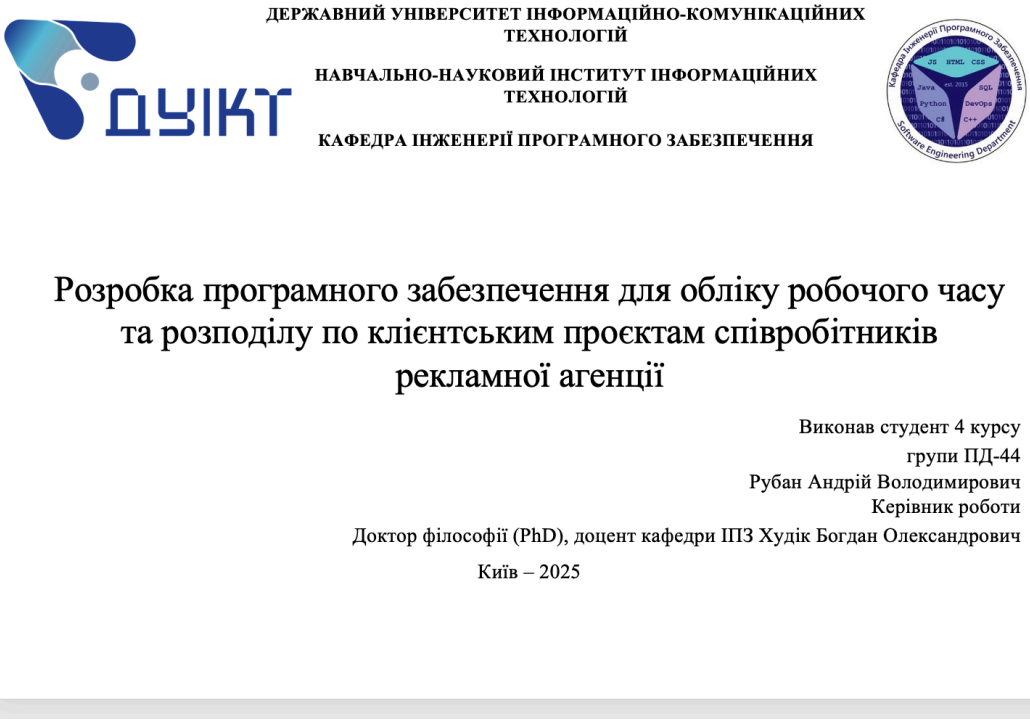
ПЕРЕЛІК ПОСИЛАНЬ

1. Next.js Documentation. Official Reference Documentation. URL: https://nextjs.org/docs (дата звернення: 21.04.2025)
2. React Documentation. Official Reference Documentation. URL: https://react.dev/learn (дата звернення: 22.04.2025)
3. Tailwind CSS Documentation. Official Reference Documentation. URL: https://tailwindcss.com/docs/installation (дата звернення: 23.04.2025)
4. i18next Documentation. Official Reference Documentation. URL: https://www.i18next.com/ (дата звернення: 24.04.2025)
5. React-i18next Documentation. Official Reference Documentation. URL: https://react.i18next.com/ (дата звернення: 25.04.2025)
6. Drizzle ORM Documentation. Official Reference Documentation. URL: https://orm.drizzle.tech/ (дата звернення: 26.04.2025)
7. Clerk Documentation. Official Reference Documentation. URL: https://clerk.dev/docs (дата звернення: 27.04.2025)
8. SQLite Documentation. Official Reference Documentation. URL: https://sqlite.org/docs.html (дата звернення: 28.04.2025)
9. Sentry Documentation. Official Reference Documentation. URL: https://docs.sentry.io/ (дата звернення: 01.05.2025)
10. Vercel Analytics Documentation. Official Reference Documentation. URL: https://vercel.com/docs/analytics (дата звернення: 02.05.2025)
11. Toggl Track. Official Product Website. URL: https://toggl.com/track (дата звернення: 03.05.2025)
12. Clockify. Official Product Website. URL: https://clockify.me/ (дата звернення: 04.05.2025)
13. Hubstaff. Official Product Website. URL: https://hubstaff.com/ (дата звернення: 05.05.2025)
14. RescueTime. Official Product Website. URL: https://rescuetime.com/ (дата звернення:

06.05.2025)

15. Kuutila, M., Mäntylä, M., Farooq, U., & Claes, M. Time Pressure in Software Engineering: A Systematic Review. arXiv preprint (2019). URL: https://arxiv.org/abs/1901.11437 (дата звернення: 08.05.2025)
16. Wagner, S., & Ruhe, M. A Systematic Review of Productivity Factors in Software Development. arXiv preprint (2018). URL: https://arxiv.org/abs/1803.06337 (дата звернення: 09.05.2025)
17. Duarte, C. H. C. Software Productivity in Practice: A Systematic Mapping Study. Software, 1(2), 164–208 (2022). URL: https://www.mdpi.com/2674-113X/1/2/8 (дата звернення: 10.05.2025)
18. Shadcn/ui. Beautifully designed components. URL: https://ui.shadcn.com/ (дата звернення: 09.05.2025)
19. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). URL: https://www.pmi.org/pmbok-guide-standards (дата звернення: 08.05.2025)
20. ExcelJS. Read, manipulate and write spreadsheet data and styles. URL: https://github.com/exceljs/exceljs (дата звернення: 07.05.2025)

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для обліку робочого часу та розподілу по клієнтським проєктам співробітників рекламної агенції

Виконав студент 4 курсу
групи ПД-44
Рубан Андрій Володимирович
Керівник роботи
Доктор філософії (PhD), доцент кафедри ІПЗ Худік Богдан Олександрович

Київ – 2025

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати існуючі рішення для обліку робочого часу та їх обмеження.
2. Визначити функціональні та нефункціональні вимоги до системи.
3. Спроєктувати архітектуру веб-додатку на основі сучасних технологій.
4. Розробити схеми бази даних для зберігання інформації про користувачів та часових записів.
5. Реалізувати інтерфейса користувача з використанням React та Tailwind CSS.
6. Імплементувати систему аутентифікації та авторизації з використанням Clerk.
7. Розробити компоненти для обліку робочого часу.
8. Створити системи фільтрації та формування звітів.
9. Реалізувати функціонал експорту даних у формат Excel.
10. Впровадити багатомовність з підтримкою української та англійської мов.
11. Протестувати систему на відповідність функціональним та нефункціональним вимогам.

2

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Метою роботи: Удосконалення процесу обліку робочого часу та розподілу завдань між співробітниками рекламної агенції шляхом розробки веб-додатку.

Об'єктом дослідження: Процес обліку робочого часу та розподілу ресурсів між клієнтськими проєктами в умовах рекламної агенції.

Предметом дослідження: Інструменти розробки, що забезпечують автоматизацію процесів обліку робочого часу та розподілу завдань у клієнтських проєктах.

3

АНАЛІЗ АНАЛОГІВ

Технічні характеристики	Toggl Track	Clockify	Hubstaff	RescueTime	Mediacom UA time track
Призначення	Облік робочого часу	Тайм-трекінг і управління проєктами	Моніторинг продуктивності	Автоматичний облік часу	Облік робочого часу та розподіл по клієнтам
Платформа	Веб, мобільний додаток	Веб, мобільний додаток	Веб, мобільний додаток	Веб, мобільний додаток	Веб
Технології	React, Node.js	React, GoLang	Electron, Node.js	Python, Electron	Next.js, React, Drizzle ORM
База даних	PostgreSQL	MySQL	SQLite, PostgreSQL	SQLite	SQLite
Основні функції	Тайм-трекінг, звіти, експорт	Тайм-трекінг, інвойсинг, звіти	Тайм-трекінг, продуктивність	Автоматичний тайм-трекінг, звіти	Тайм-трекінг, розподіл годин по проєктах, звіти, експорт
Мультимовність	Так	Так	Ні	Так	Так (uk/en)
Аутентифікація	Google, Email	Google, Email, Microsoft	Email	Google, Email	Clerk (email)
Розмежування доступу	Адмін, користувач	Адмін, менеджер, користувач	Адмін, користувач	Адмін, користувач	Адмін, користувач
Можливості інтеграції	Asana, Jira, Slack	Asana, Trello, Jira, Slack	Trello, Asana, Slack	Trello, Slack	API-інтерфейс для інтеграції
Модуль звітності	Так	Так	Так	Так	Так
Модуль статистики	Так	Так	Так	Так	Так

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

1. Реєстрація та автентифікація користувачів з різними рівнями доступу (керівник компанії (адміністратор), співробітник).
2. Створення та редагування профілів співробітників з інформацією про посаду, відділ, електронної пошти, паролю.
3. Створення нових клієнтських компаній.
4. Призначення співробітників в компанію з визначенням їх ролей.
5. Введення та редагування записів про відпрацьований час.
6. Додавання коментарів до часових записів.
7. Автоматичний розрахунок загального відпрацьованого часу.
8. Формування звітів про відпрацьований час співробітників.
9. Експорт звітів у різних форматах (Excel).

5

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Нефункціональні:

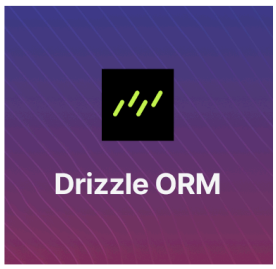
1. Швидкий відгук системи при роботі з 100+ записів.
2. Швидка генерація звітів (до 30 секунд для складних звітів).
3. Шифрування пошти та паролю користувачів .
4. Інтерфейс є зрозумілим для користувача без необхідності додаткового навчання чи пояснень.
5. Адаптивний дизайн для різних пристроїв (ПК, смартфони).
6. Кросплатформність (Windows, macOS, Linux).
7. Підтримка сучасних браузерів (Chrome, Firefox, Safari, Edge).
8. Масштабованість системи повинна забезпечувати стабільну роботу при одночасному використанні до 100 користувачів.
9. Наявність панелі моніторингу з ключовими показниками.

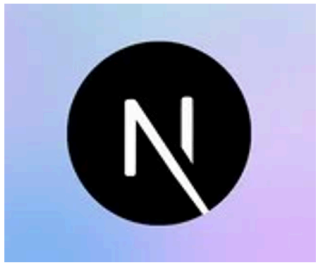
6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

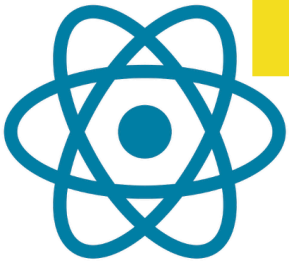
VSCode

Clerk





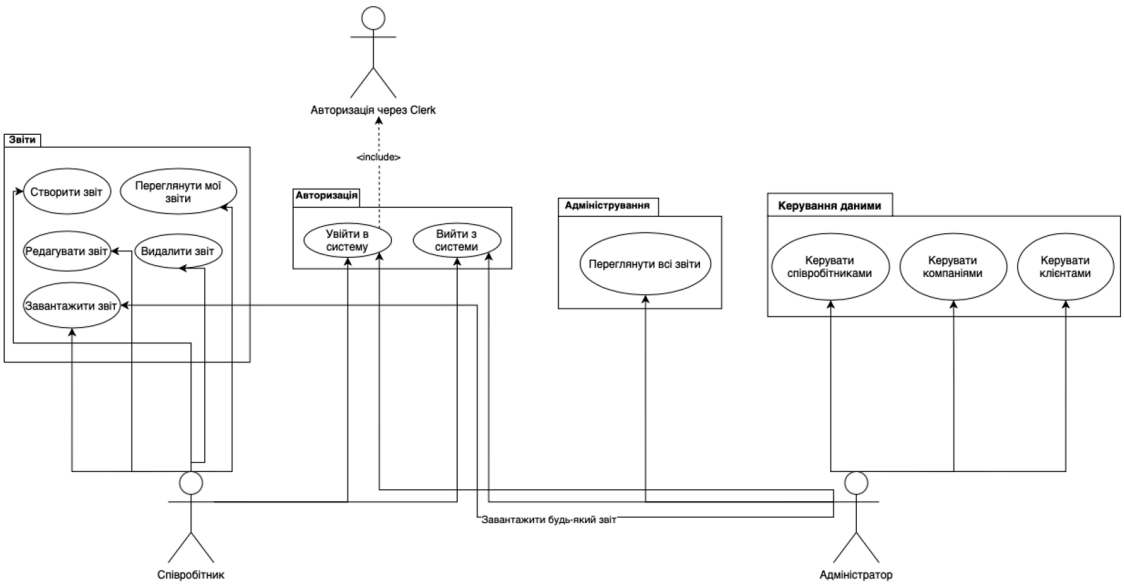
Next.JS



React

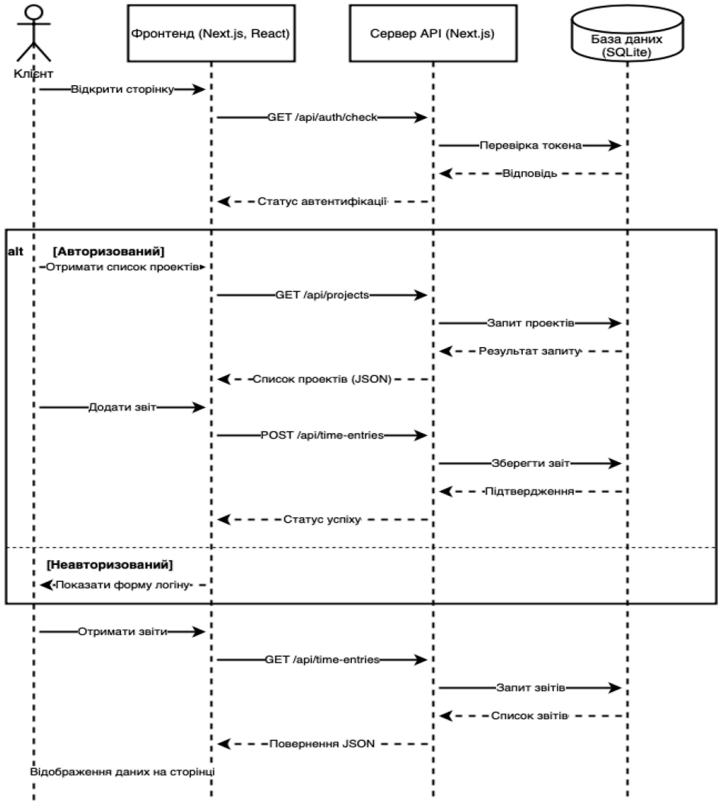
7

Варіанти використання застосунку



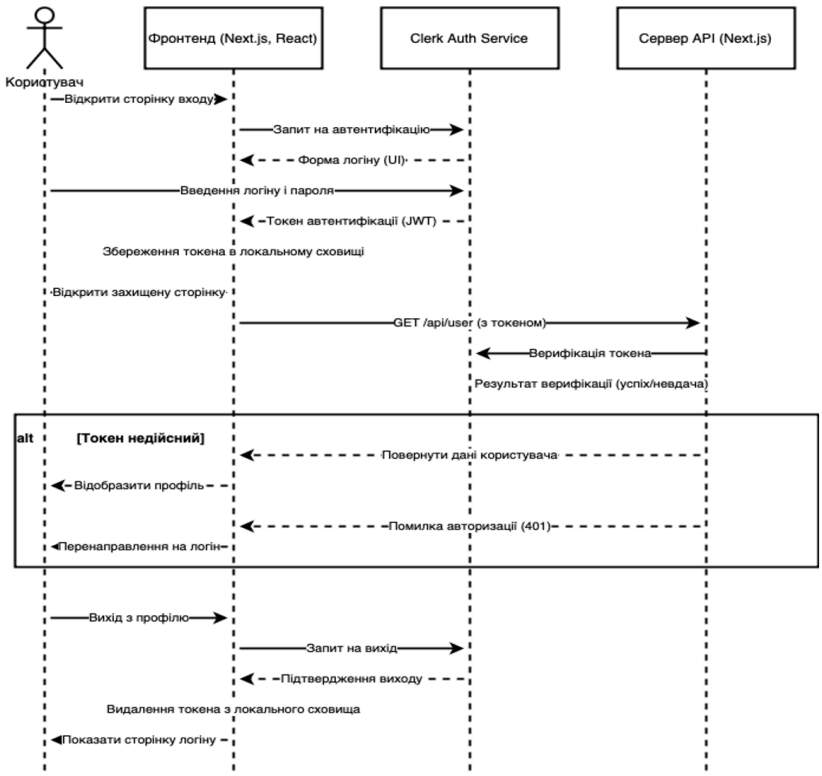
8

Взаємодія програмного забезпечення із клієнтом



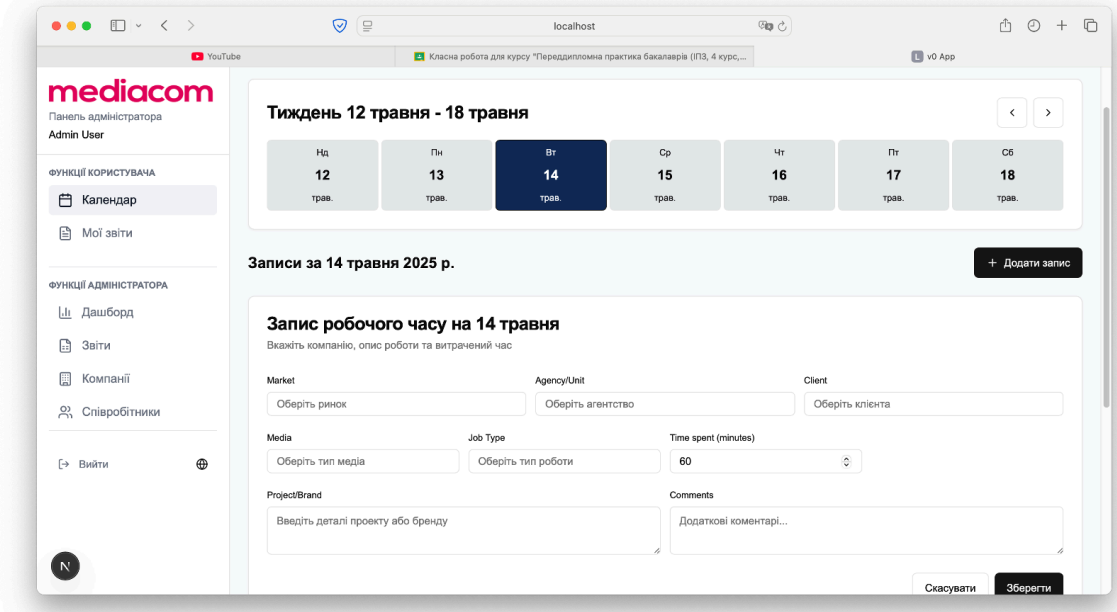
9

Діаграма авторизації клієнта у додатку



10

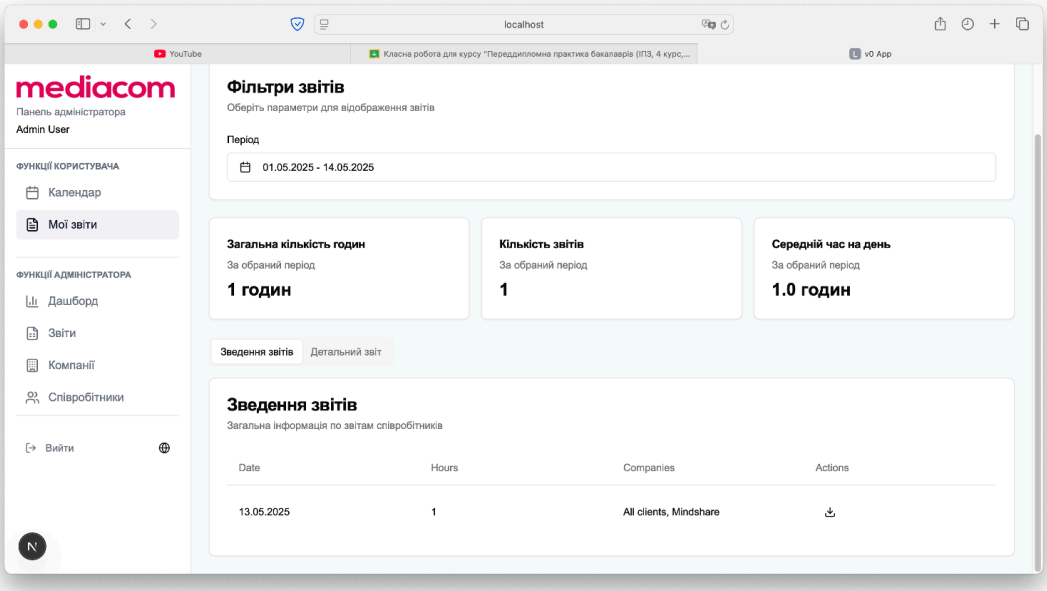
ЕКРАННІ ФОРМИ



Сторінка користувача для створення запису

11

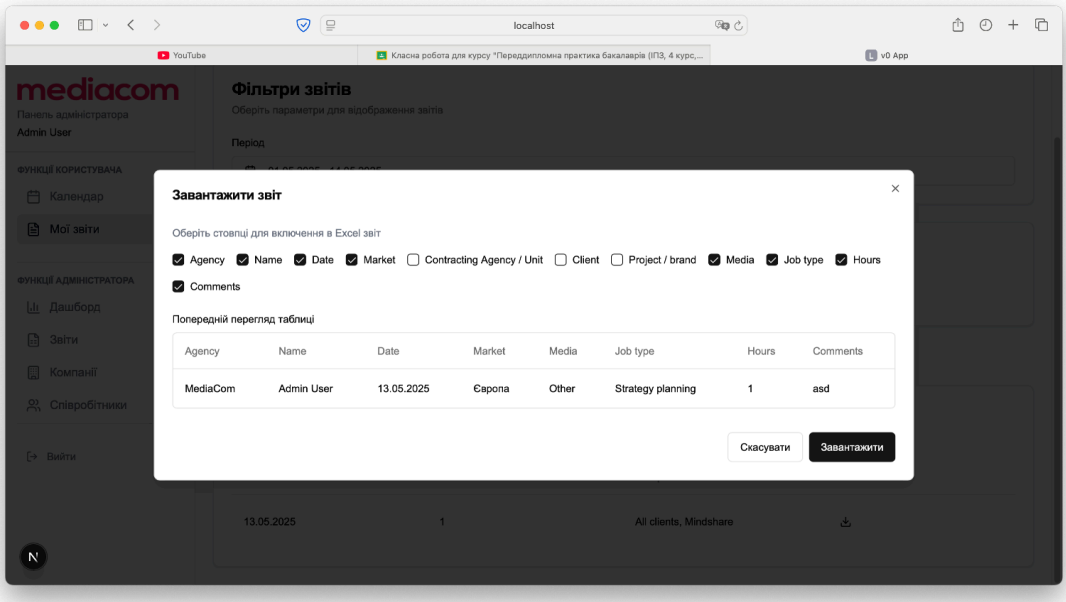
ЕКРАННІ ФОРМИ



Сторінка користувача для перегляду своїх записів

12

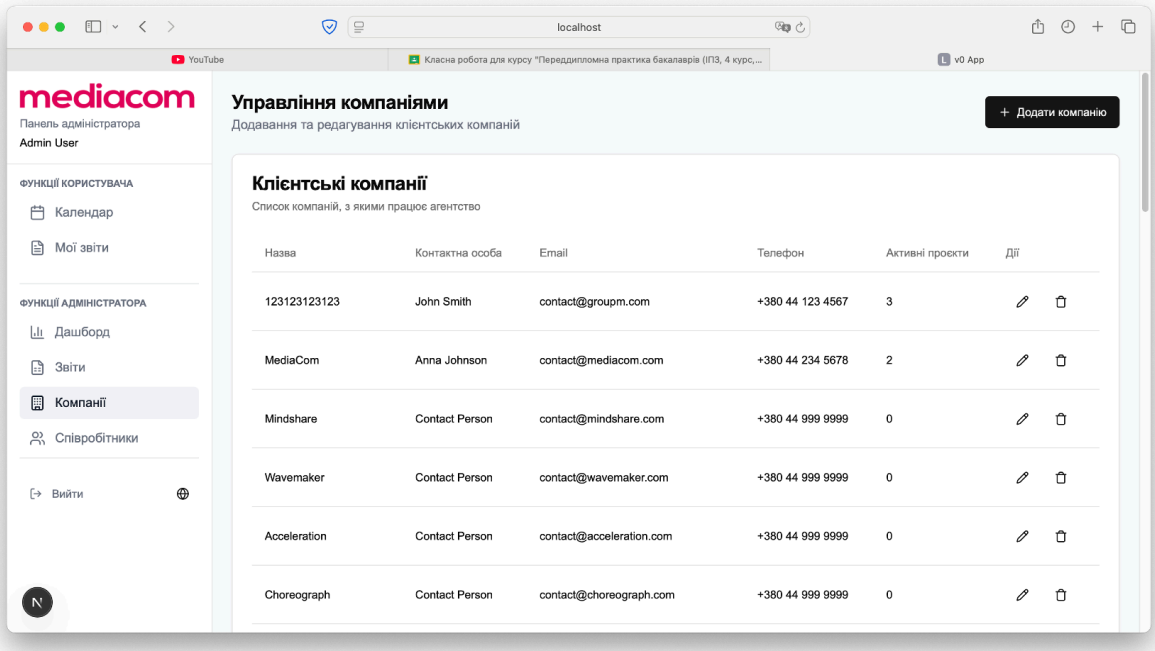
ЕКРАННІ ФОРМИ



Вікно перегляду запису та редагування перед завантаженням

13

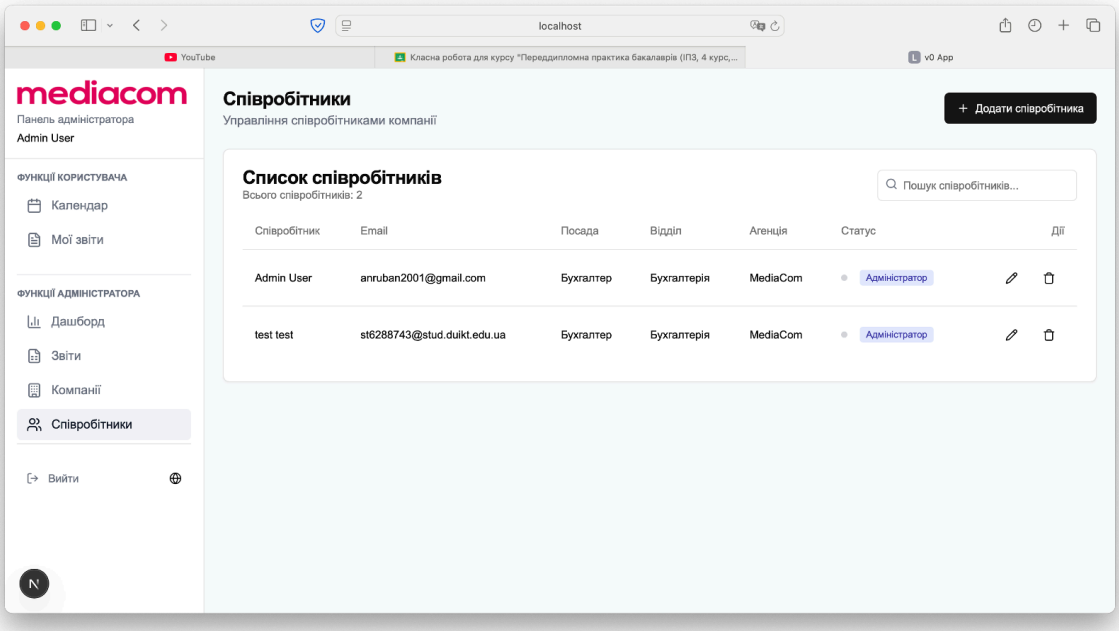
ЕКРАННІ ФОРМИ



Сторінка адміністратора для створення та редагування бази компаній

14

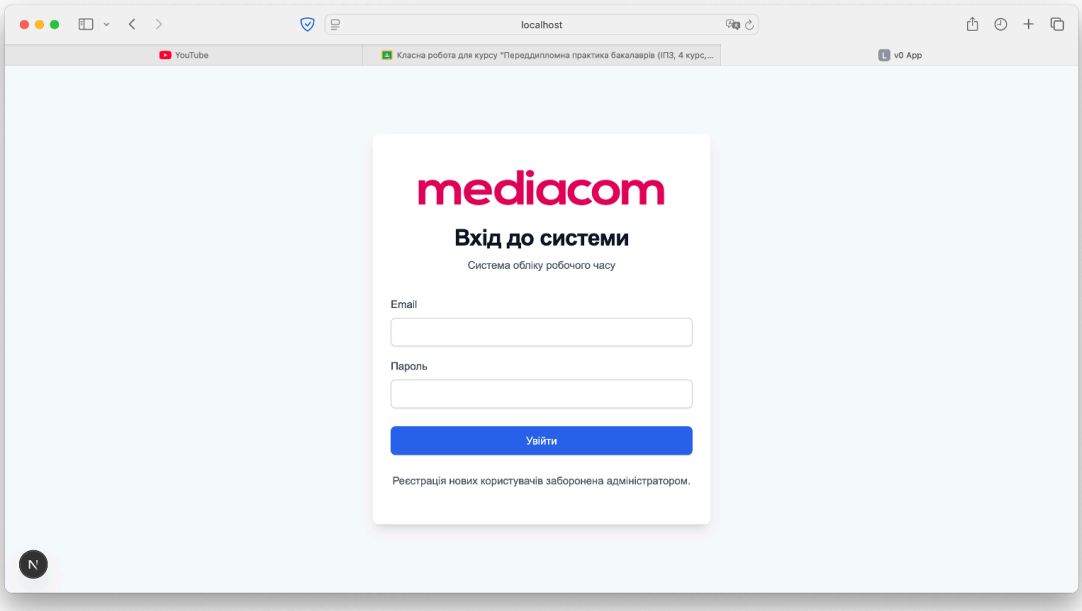
ЕКРАННІ ФОРМИ



Сторінка адміністратора для створення та редагування робітників

15

ЕКРАННІ ФОРМИ



Сторінка авторизації

16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Рубан А.В., Худік Б.О. Інформаційна система для розподілу працівників по проєктах рекламної агенції: постановка задачі та реалізація // VI Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». – Київ: ДУІКТ, 2025 (подано до друку).
2. Рубан А.В., Худік Б.О. Підвищення прозорості робочих процесів через цифровізацію обліку робочого часу // VI Науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». – Київ: ДУІКТ, 2025. – С. 539-542

17

ВИСНОВКИ

1. Проаналізовано предметну область обліку робочого часу та розподілу ресурсів у рекламних агенціях. Встановлено ключові функціональні й нефункціональні вимоги до відповідної системи.
2. Розроблено архітектуру веб-застосунку на основі компонентного підходу з чітко визначеними взаємозв'язками між модулями та оптимізованою схемою бази даних SQLite з використанням ORM Drizzle.
3. Впроваджено систему автентифікації та авторизації на базі сервісу Clerk з підтримкою рольової моделі доступу (користувач/адміністратор). Створено необхідні API-ендпоінти для керування звітами та персоналом.
4. Сформовано інтуїтивний користувацький інтерфейс на технологіях Next.js та React, що забезпечує ефективну взаємодію з системою. Реалізовано підтримку двох мов (української та англійської) за допомогою бібліотеки i18next.
5. Здійснено практичне тестування функціональності системи всіх модулів застосунку та експорту даних у форматі Excel та взаємодії з базою даних.

18

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
import { NextResponse } from 'next/server';
import { employeeQueries, companyQueries, reportQueries }
from '@/db/queries';
import { auth } from '@clerk/nextjs/server';
import { clerkClient } from '@clerk/nextjs/server'

export async function GET(request: Request) {
 try {
  const employees = await employeeQueries.getAll();
  const companies = await companyQueries.getAll();

  return NextResponse.json({
   employeeCount: employees.length,
   companyCount: companies.length
  });
 } catch (error) {
  console.error("API Error:", error);
  return NextResponse.json({ error: "Failed to fetch dashboard
data" }, { status: 500 });
 }
}

export async function POST(request: Request) {
 const client = await clerkClient()
 try {
  const body = await request.json();
  const { action } = body;

  if (action === 'getEmployees') {
   const employees = await employeeQueries.getAll();

   const enhancedEmployees = await Promise.all(
    employees.map(async (employee) => {
     let isAdmin = false;

     if (employee.clerkId) {
      try {
       const clerkUser = await
client.users.getUser(employee.clerkId);
       isAdmin = clerkUser.publicMetadata.role === 'admin';
      } catch (error) {
       console.error(`Error fetching Clerk user for employee
${employee.id}:`, error);
      }
     }

     return {
      ...employee,
      isAdmin
     };
    })
   );

   return NextResponse.json({ employees:
enhancedEmployees });
  }
  else if (action === 'addEmployee') {
   const { employee } = body;

   try {
    const { userId } = await auth();

    if (!userId) {
     return NextResponse.json({ error: 'Unauthorized' }, {
status: 401 });
    }

    const adminUser = await client.users.getUser(userId);
    const isAdmin = adminUser.publicMetadata.role ===
'admin';

    if (!isAdmin) {
     return NextResponse.json({ error: 'Forbidden: Admin
access required' }, { status: 403 });
    }

    if (!employee.email || !employee.password ||
!employee.name) {
     return NextResponse.json({ error: 'Email, password and
name are required' }, { status: 400 });
    }

    const newUser = await client.users.createUser({
     emailAddress: [employee.email],
     password: employee.password,
    });


    const today = new Date().toISOString();

    const newEmployee = await employeeQueries.add({
     name: employee.name,
     email: employee.email,
     position: employee.position,
     department: employee.department,
     joinDate: today,
     status: employee.status,
     clerkId: newUser.id,
     agency: employee.agency || employee.department,
    });

    return NextResponse.json({
     id: newEmployee.id,
     userId: newUser.id,
     success: true
    });
   } catch (error: any) {
    console.error("Error adding employee:", error);
    console.log(error.errors);
    return NextResponse.json({
     error: error.message || "Failed to add employee"
    }, {
     status: error.status || 500
    });

   }
  }
  else if (action === 'updateEmployee') {
   const { employee } = body;

   try {
    const { userId } = await auth();

    if (!userId) {
     return NextResponse.json({ error: 'Unauthorized' }, {
status: 401 });
    }
```

```
      const adminUser = await client.users.getUser(userId);
      const isAdmin = adminUser.publicMetadata.role ===
'admin';

      if (!isAdmin) {
        return NextResponse.json({ error: 'Forbidden: Admin
access required' }, { status: 403 });
      }

      const existingEmployee = await
employeeQueries.getById(employee.id);

      if (!existingEmployee) {
        return NextResponse.json({ error: "Employee not found"
}, { status: 404 });
      }

      const updatedEmployee = await
employeeQueries.update(employee.id, {
        name: employee.name,
        email: employee.email,
        position: employee.position,
        department: employee.department,
        status: employee.status || 'Active',
        agency: employee.agency || employee.department,
      });

      if (existingEmployee.clerkId) {
        try {
          if (employee.isAdmin !== undefined) {
            await
client.users.updateUser(existingEmployee.clerkId, {
              publicMetadata: {
                role: employee.isAdmin ? 'admin' : 'user'
              }
            });
          }

          if (employee.password && employee.password.trim()
!== '') {
            try {
              await
client.users.updateUser(existingEmployee.clerkId, {
                password: employee.password
              });
            } catch (passwordError) {
              console.error("Error updating user password in
Clerk:", passwordError);
              return NextResponse.json({
                error: "Employee data updated but password change
failed"
              }, { status: 500 });
            }
          }
        } catch (clerkError) {
          console.error("Error updating user in Clerk:",
clerkError);
          return NextResponse.json({
            error: "Employee data updated but role change failed"
          }, { status: 500 });
        }
      }

      return NextResponse.json({
        success: true,
        employee: updatedEmployee,
        message: `Employee ${employee.name} updated
successfully`
      });
    } catch (error) {
      console.error("Error updating employee:", error);
      return NextResponse.json({ error: "Failed to update
employee" }, { status: 500 });
    }
  }
  else if (action === 'deleteEmployee') {
    const { id } = body;

    try {
      const employee = await employeeQueries.getById(id);

      if (!employee) {
        return NextResponse.json({ error: "Employee not found"
}, { status: 404 });
      }

      const deletedEmployee = await
employeeQueries.delete(id);

      if (employee.clerkId) {
        try {
          await client.users.deleteUser(employee.clerkId);
        } catch (clerkError) {
          console.error("Error deleting user from Clerk:",
clerkError);
        }
      }

      return NextResponse.json({
        success: true,
        message: `Employee ${employee.name} successfully
deleted`
      });
    } catch (error) {
      console.error("Error deleting employee:", error);
      return NextResponse.json({ error: "Failed to delete
employee" }, { status: 500 });
    }
  }
  else if (action === 'getCompanies') {
    const companies = await companyQueries.getAll();
    const allReports = await
reportQueries.getAllWithEmployee();

    const companiesWithStats = companies.map(company => {
      const companyReports = allReports.filter(r =>
        r.report.client?.includes(company.name) ||
        r.report.contractingAgency?.includes(company.name)
      );

      const totalHours = companyReports.reduce((sum, r) =>
sum + r.report.hours, 0);
      const revenue = totalHours * 1000;

      return {
        ...company,
        activeProjects: company.projects || 0,
        totalHours,
        revenue
      };
    });

    return NextResponse.json({ companies:
companiesWithStats });
  }
  else if (action === 'addCompany') {
    const { company } = body;

    try {
```

```
      const newCompany = await companyQueries.add({
        name: company.name,
        contact: company.contact,
        email: company.email,
        phone: company.phone,
        projects: 0,
        address: company.address || null,
        notes: company.notes || null
      });

      return NextResponse.json({ id: newCompany.id, success:
true });
    } catch (error) {
      console.error("Error adding company:", error);
      return NextResponse.json({ error: "Failed to add
company" }, { status: 500 });
    }
  }
  else if (action === 'updateCompany') {
    const { company } = body;

    try {
      if (!company.name || !company.contact || !company.email
|| !company.phone) {
        return NextResponse.json({
          error: "Required fields missing: name, contact, email,
and phone are required"
        }, { status: 400 });
      }

      const existingCompany = await
companyQueries.getById(company.id);
      if (!existingCompany) {
        return NextResponse.json({ error: "Company not found"
}, { status: 404 });
      }

      const updatedCompany = await
companyQueries.update(company.id, {
        name: company.name,
        contact: company.contact,
        email: company.email,
        phone: company.phone,
        address: company.address || null,
        notes: company.notes || null
      });

      return NextResponse.json({
        success: true,
        company: updatedCompany,
        message: "Company updated successfully"
      });
    } catch (error) {
      console.error("Error updating company:", error);
      return NextResponse.json({ error: "Failed to update
company" }, { status: 500 });
    }
  }
  else if (action === 'deleteCompany') {
    const { id } = body;

    try {
      const deletedCompany = await
companyQueries.delete(id);

      if (!deletedCompany) {
        return NextResponse.json({ error: "Company not found"
}, { status: 404 });
      }

      return NextResponse.json({
        success: true,
        message: `Company ${deletedCompany.name}
successfully deleted`
      });
    } catch (error) {
      console.error("Error deleting company:", error);
      return NextResponse.json({ error: "Failed to delete
company" }, { status: 500 });
    }
  }
  else if (action === 'getReports') {
    const allReports = await
reportQueries.getAllWithEmployee();

    return NextResponse.json({ reports: allReports });
  }
  else {
    return NextResponse.json({ error: "Invalid action" }, {
status: 400 });
  }
 } catch (error) {
  console.error("API Error:", error);
  return NextResponse.json({ error: "Failed to process
request" }, { status: 500 });
 }
}

import { Webhook } from 'svix';
import { headers } from 'next/headers';
import { WebhookEvent } from '@clerk/nextjs/server';
import { NextResponse } from 'next/server';
import { employeeQueries } from '@/db/queries';
import { clerkClient } from '@clerk/nextjs/server';
import fs from 'fs/promises';
import path from 'path';

async function acquireLock(): Promise<boolean> {
 const lockFile = path.join(process.cwd(), 'user-creation.lock');
 try {
  await fs.writeFile(lockFile, Date.now().toString(), { flag:
'wx' });
  return true;
 } catch (error) {
  // File exists, lock is held
  return false;
 }
}

async function releaseLock(): Promise<void> {
 const lockFile = path.join(process.cwd(), 'user-creation.lock');
 try {
  await fs.unlink(lockFile);
 } catch (error) {
  console.error('Error releasing lock:', error);
 }
}

export async function POST(req: Request) {

 const headerPayload = await headers();
 const svix_id = headerPayload.get('svix-id');
 const svix_timestamp = headerPayload.get('svix-timestamp');
 const svix_signature = headerPayload.get('svix-signature');

 if (!svix_id || !svix_timestamp || !svix_signature) {
  return new Response('Error: Not a valid Clerk webhook', {
   status: 400
  });
 }
```

```
  const payload = await req.json();
  const body = JSON.stringify(payload);

  const webhookSecret =
process.env.CLERK_WEBHOOK_SECRET;

  if (!webhookSecret) {
    console.error('Error: CLERK_WEBHOOK_SECRET is not
set');
    return new Response('Error:
CLERK_WEBHOOK_SECRET is not set', {
      status: 500
    });
  }

  const webhook = new Webhook(webhookSecret);

  let event: WebhookEvent;

  try {
    event = webhook.verify(body, {
      'svix-id': svix_id,
      'svix-timestamp': svix_timestamp,
      'svix-signature': svix_signature,
    }) as WebhookEvent;
  } catch (err) {
    console.error('Error verifying webhook:', err);
    return new Response('Error verifying webhook', {
      status: 400
    });
  }

  const eventType = event.type;

  if (eventType === 'user.created') {
    const { id, email_addresses, first_name, last_name } =
event.data;
    const email = email_addresses[0]?.email_address;

    if (!email) {
      return new Response('Error: No email address found', {
status: 400 });
    }

    try {
      const lockAcquired = await acquireLock();
      if (!lockAcquired) {
        await new Promise(resolve => setTimeout(resolve,
1000));
      }

      const employees = await employeeQueries.getAll();
      const existingEmployee = employees.find(emp =>
emp.clerkId === id);

      if (existingEmployee) {
        if (lockAcquired) await releaseLock();
        console.log('User already exists in database, skipping
creation');
        return NextResponse.json({
          success: true,
          message: 'User already exists in database',
          userId: id
        });
      }

      const emailEmployee = employees.find(emp => emp.email
=== email);
      if (emailEmployee) {
        if (lockAcquired) await releaseLock();
        console.log('User with this email already exists, skipping
creation');
        return NextResponse.json({
          success: true,
          message: 'User with this email already exists',
          userId: id
        });
      }

      const isFirstUser = employees.length === 0;

      if (isFirstUser) {
        console.log('First user detected, setting as admin');

        const clerk = await clerkClient();
        await clerk.users.updateUser(id, {
          publicMetadata: {
            role: "admin"
          }
        });

        // Add the user to the employees table
        const today = new Date().toISOString();
        await employeeQueries.add({
          name: `${first_name || ''} ${last_name || ''}`.trim() ||
'Admin User',
          email: email,
          position: 'Admin',
          department: 'Administration',
          joinDate: today,
          status: 'Active',
          clerkId: id,
        });

        if (lockAcquired) await releaseLock();
        return NextResponse.json({ success: true, isAdmin: true
});
      } else {
        // For subsequent users, we won't automatically add them
to the database
        // The admin will need to add them manually
        console.log('User created but not added to database
(admin needs to add manually)');

        if (lockAcquired) await releaseLock();
        return NextResponse.json({ success: true, isAdmin: false
});
      }
    } catch (error) {
      await releaseLock();
      console.error('Error processing user.created webhook:',
error);
      return new Response('Error processing webhook', { status:
500 });
    }
  }

  // Return a 200 response for all other event types
  return NextResponse.json({ success: true });
}
```