

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ МОТИВАЦІЇ ТА УПРАВЛІННЯ
КОМАНДНОЇ РОЗРОБКИ ІТ-СТАРТАПУ
SOFTWARE IMPLEMENTATION OF METHODS FOR MOTIVATION AND
MANAGEMENT OF AN IT-STARTUP TEAM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

БІДЕНКО І.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ДУБРОВІН В.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БІДЕНКА Іллі Віталійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація методів мотивації та управління командної розробки ІТ-стартапу. Software Implementation of Methods for Motivation and Management of an IT-Startup Team

керівник проєкту (роботи) канд.техн.наук, проф. ДУБРОВІН Валерій Іванович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ДУБРОВІН В.І., професор		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання 15 квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області та постановка завдання.	1 тиждень	Розділ 1
3	Вибір та обґрунтування системи, яка проєктується.	2-3 тиждень	Розділ 2
4	Основні рішення щодо реалізації компонентів системи.	3-4 тиждень	Розділ 3
5	Експлуатація, тестування та експериментальне дослідження програми.	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

Ілля БІДЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Валерій ДУБРОВІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 61 сторінки, 25 рисунків, 6 таблиці, 2 додатки, 17 джерел.

ІТ-СТАРТАП, МОТИВАЦІЯ КОМАНДИ, УПРАВЛІННЯ КОМАНДОЮ, ОЦІНЮВАННЯ РОБОТИ КОМАНДИ.

Об'єкт дослідження – процес програмної розробки методів мотивації і управління командою.

Предмет дослідження – методи управління командою розробки.

Мета роботи – створення програмної системи для мотивації та управління командою розробки ІТ-стартапу.

Матеріали, методи та технічні засоби: бібліотека React на основі мови програмування JavaScript, програмна платформа Node.js та Express.js, реляційна база даних PostgreSQL, персональний комп'ютер з операційною системою Microsoft Windows 10 Pro та процесором Intel Core I5 3570k.

Результат. Результатом роботи став повністю працюючий web-застосунок з зручним інтерфейсом, надійним сервером, інтеграцією з зовнішніми застосунками та зберіганням даних у базі даних.

Висновки. Було проаналізовано особливості ІТ-стартапів та керування командою розробки, розглянуто методи керування співробітників у ІТ-стартапі, створено програмний web-застосунок для автоматизації методів мотивації та управління командою розробки ІТ-стартапу.

Галузь використання – ІТ-стартапи, або компанії які мають ІТ-відділ(и).

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 61 pages, 25 figures, 6 table, 2 appendixes, 17 sources.

IT-STARTUP, TEEM MOTIVATION, TEEM MENEDGEMENT, PERFORMANCE TEEM EVALUATION.

The object of research is the process of software development of motivation methods and team management.

The subject of research is managing methods of a development team.

The purpose of the work is to create a software system for the automation of motivation methods and management of an IT-startup development team.

Materials, methods, and tools: React library based on the JavaScript programming language, Node.js and Express.js software platforms, PostgreSQL relational database, personal computer with the Microsoft Windows 10 Pro operation system and an Intel Core I5 3570k processor.

Results. The result of the work was a fully functional web application with a user-friendly interface, a reliable server, integration with external relations and data storage in the database.

Conclusions. The features of IT startups and development team management were analyzed, methods of managing employees in an IT startup were considered, a software web application was created to automate methods of motivating and managing the development team of an IT startup.

Field of use - IT startups, or companies that have an IT department(s).

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області та постановка завдання	10
1.1 Поняття та аналіз особливостей стартапів	10
1.2 Аналіз управління командою розробки	12
1.3 Постановка завдання роботи	14
1.4 Висновки до розділу	15
2 Вибір та обґрунтування структури системи, яка проєктується	16
2.1 Архітектурна модель	16
2.2 Популярні методи управління командою розробки	16
2.3 Структура модулів системи	20
2.4 Обґрунтування вибору структури	21
2.5 Висновки до розділу	21
3 Основні рішення щодо реалізації компонентів системи	22
3.1 Розробка програмного коду серверу	22
3.2 Розробка інтерфейсу додатку	22
3.3 Висновки до розділу	28
4 Експлуатація, тестування та експериментальне дослідження програми	30
4.1 Тестування інтерфейсу додатку	30
4.2 Тестування сервера	31
4.3 Висновок за розділом	32
Висновки	33
Перелік джерел посилання	34
Додаток А Текст програми	35
Додаток Б Слайди презентації	53

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
REST – Representational State Transfer;
HR – Human Resources;
WIP – Work In Progress;
IT – Інформаційні технології;
WEB – англ. павутиння;
UI – user interface;
БД – база даних;
СУБД – система управління базами даних.

ВСТУП

Поняття «стартап» за останні десять-двадцять років стало асоціюватися у нас в головах як прорив, котрий майже зі стовідсотковою вірогідністю пов'язаний з новими технологіями або розробками. Стартап (від англійської Start-up - запускати), - нещодавно створена компанія, що будує свій бізнес на основі інновацій або інноваційних технологій, не вийшла на ринок або почала на нього виходити і що володіє обмеженими ресурсами[1]. Доволі умовно стартапи поділяються на глобальні та локальні. Глобальні – абсолютна інновація у всьому світі, а локальні – інновація в окремій, наприклад, країні, але в інших країнах ця технологія вже не є інноваційною. Останніми роками з'явилась добра тенденція – стратегічне управління, зазвичай керівник команди з ІТ розробляє особисто стратегію, бо без цієї стратегії керівник команди буде стикатися з дуже великою кількістю проблем. Згідно статистики, лише 10% стартапів стають успішними[1]. Серед 90% неуспішних одна з основних причин неуспішності – відсутність чітко сформульованої ІТ-стратегії та втрата мотивації команди розробників. На цьому факті треба зацентрувати увагу, так як грамотна ІТ-стратегія прямо впливає з чіткого моделювання практичних задач управління. Керівник зобов'язаний регулярно аналізувати ефективність роботи кожного з своїх підлеглих для того щоб побудувати подальшу успішну стратегію. Неякісна стратегія впливає на інші фактори успішності, наприклад – неправильний підбір персоналу та, як наслідок, неякісна робота та/або вигорання, незатребуваний на ринку товар, неправильне використання бюджету.

Як висновок до вступу, хочу зазначити що розробка системи аналізу ефективності праці та мотивації співробітників є актуальною проблемою, вирішення якої допоможе розробникам оптимізувати свою роботу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Поняття та аналіз особливостей стартапів

1.1.1 Поняття ІТ-стартапу

ІТ-стартапи дещо відрізняються від стартапів, до яких ми звикли. ІТ-стартапи не прив'язані до агресивного маркетингу для набуття широкої популярності, ІТ-стартапи можуть створюватися тривалий термін.

Велика кількість нині популярних та відомих компаній колись також були стартапами – Google, YouTube, Apple. Наразі стадію коли ці компанії були стартапами називають «гаражною», бо вони, в прямому сенсі цього слова, починали з «офісу» в гаражі[1].

Згідно класифікації стадій розвитку стартапів їх існує 5, а саме:

- Посівна стадія (seed stage);
- Запуск продукту (startup stage);
- Стадія зростання (growth stage);
- Стадія розширення (expansion stage);
- Стадія виходу (exit stage)[2].

Будь-який стартап відрізняється від звичайної нової компанії однією рисою – стартап це повністю новий продукт, котрого до цього на ринку не було, саме в цьому і полягає інноваційна основа.

1.1.2 Особливості управління командою в ІТ-сфері

Менеджмент є основною та невід'ємною частиною розвитку продуктів.

Менеджмент – це управління (над продуктом, як у нашому випадку).

Управління продуктом або проектом – це методологія організації, планування, керівництва, координації трудових, фінансових та матеріально-технічних ресурсів протягом проектного циклу, спрямована на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки та технології управління для досягнення певних результатів щодо складу та обсягу робіт, вартості, часу, якості та задоволення учасників проекту[3].

Управління ІТ-проектами складніше, ніж управління проектами інших типів. Термін «ІТ-проект» можна описати у вигляді загального терміна «проект» та з деякими важливими доповненнями, пов'язаними, перш за все, з тим, що ІТ-проект має на увазі собою діяльність, спрямовану на створення або використання інформаційних технологій. Зазвичай ІТ-проекти замовляються бізнес-відділом компанії, а впроваджуються – або властим ІТ-відділом, або самостійною ІТ-компанією. Через те що ІТ-сфера є досить специфічною, можливе виникнення труднощів у визначенні точних бізнес-вимог та загальних очікувань від розроблюваного і впроваджуваного програмного забезпечення. Тобто обговорення перших етапів реалізації проекту між замовником та керівником розробників, бо на такому етапі дуже розмиті ідеї, котрі фіксуються у технічному завданні наспіх, а потім можуть змінитись. Друга особливість ІТ-проекту – рівноцінний розподіл провини на учасників при його неблагополучній реалізації. Тобто відповідальність за результат лягає не тільки на керівника розробників (за можливу неправильну реалізацію) або не тільки на замовника (за можливу помилку в формуванні вимог), а на всіх учасників разом, тому що, так чи інакше, вони повинні створювати між собою ефективну комунікацію, мінімізуючи можливий суб'єктивізм в подальшому. Третьою особливістю можна вважати доволі високу вартість такого проекту. Зазвичай контракти підписуються на початку формування ідеї, але як показала практика – в ІТ-проектах регулярні зміни це нормальний процес. Це тягне за собою «запит на

зміни у контракті», котрий так само вимагає кошти. Велике число запитів має на увазі під собою і велику вартість, саме через цю причину ІТ-проекти вважаються одними з найдорожчих видів проєктів.

1.2 Аналіз управління командою розробки

1.2.1 Управління командою розробки

В наш час вже було і є багато компаній, які своїми прикладами показували як потрібно діяти і як не потрібно, тому майже кожен керівник усвідомлює значення співробітників та їх рівня професіоналізму (наприклад у вирішенні поставлених задач).

Саме методи управління командою стають тим стовпом надійності та стійкої ефективної роботи організації або підприємства, навіть за умови посилення конкуренції. Дуже важливо щоб на підприємстві була побудована система управління співробітниками, яка допоможе вирішувати задачі підприємства. Це і буде показником ефективної роботи керівництва.

Найважливіший зараз, мабуть, напрямок діяльності багатьох організацій це управління трудовими ресурсами або HR(англійською human resources, тобто людські ресурси). Якщо раніше керівники звертали увагу на технічне вдосконалення та/або технічні інновації, то зараз видно значний поворот в сторону людського фактора, бо саме люди визначають сутність та ефект підприємства. Саме від HR-менеджера або HR-відділу залежить ефективність використання людських ресурсів

1.2.2 Аналіз мотивації розробників

Мотивація є однією з провідних функцій управління, оскільки досягнення основної мети залежить від злагодженості роботи людей[4]. Кожен ефективний керівник намагається переконати співробітників працювати краще, створити у них внутрішні спонуки до активної трудової діяльності, підтримувати зацікавленість у праці, ініціювати переживання задоволення від отриманих результатів[4]. Важливо також, щоб працівники домагалися досягнення цілей організації добровільно і творчо[4].

Якщо уважно проаналізувати мотивацію співробітників команди то можна легко спрогнозувати їхню реакцію на дії керівника або керівництва. Через мотивацію можна легко зрозуміти чому саме ця людина вчинила саме так. Це стосується і конкретного керівника: чим зумовлений його вибір тактики поведінки при розв'язанні управлінського завдання, стилю управління, манери ділового спілкування тощо[4]. Саме через цю причину мотиваційна функція є найважливішою в управлінні.

Загалом поняття «мотивація» еволюціонувало з політики «батога і пряника»(покарання і винагороди) до «концепції людських ресурсів» (сприйняття співробітника як ключової фігури на виробництві, від якої залежать кінцеві результати виробничої діяльності)[5].

Філософія управління співробітниками – це формування поведінки окремих співробітників по відношенню до цілей розвитку проєкту. В проєктах, схожий на наш, мотивація розглядається як процес що активізує мотиви працівників – це внутрішня мотивація, та створення стимулів для спонукання до ефективної праці – це зовнішня мотивація. В сучасному розумінні, мотивація – рушійна сила людської поведінки.

1.2.3 Методи оцінки команди розробки

Успіх будь-якого ІТ-проєкту залежить не лише від кваліфікації працівників команди підтримки виконання замовлень, але і від якісної моделі управління цією командою. Оцінка персоналу – це регулярна перевірка співробітників на відповідність займаній посаді, професійних компетенцій та потенціалу для подальшого розвитку[6]. Саме такі перевірки дозволяють виявити сильні та слабкі сторони співробітників, та краще зрозуміти потреби у підвищенні кваліфікації та/або навчанні. Серед її основних цілей можна зазначити – виявлення працівників з потенціалом у кар'єрному зрості, встановити наскільки співробітник відповідає вимогам його/її посаді, визначити особливі якості співробітників та надавання чіткого та конструктивного зворотнього зв'язку. Впровадження методів оцінювання співробітників, зазвичай, надає компаніям тільки переваги, такі як – можливість створення кадрових резервів, мотивація

співробітників, створення цілеспрямованої програми кар'єрного розвитку співробітників та багато іншого.

Всього виділяють п'ять основних пунктів для оцінки команди:

1. Підготовка програми оцінки;
2. Аналіз діяльності та формування критеріїв оцінки;
3. Конструювання процедур оцінки;
4. Проведення оцінки;
5. Аналіз[6].

Також задля ефективної роботи системи оцінювання потрібно мати критерії, основні з них – об'єктивність, чіткість, прозорість, мотиваційність[6]. Підхід з саме такими критеріями є найефективнішим та справедливим, бо він враховує особливості кожного співробітника та його роботи.

1.3 Постановка завдання роботи

Завданням роботи є розробка інтерактивної системи мотивації співробітників команди IT-стартапу, котра буде поєднувати в собі функціональність управління задачами (за допомогою Kanban), систему гейміфікації та інтеграцію з зовнішніми сервісами (наприклад Trello). Система має бути web-додатком для зручності (можливість оновити свій статус будь-де та будь з чого, що має вихід до інтернету), має мати інтуїтивно зрозумілий інтерфейс та систему авторизації. Всі ці пункти будуть сприяти підвищенню командної мотивації, відповідальності та ефективності.

1.4 Висновки до розділу

В цьому розділі було описано основні поняття предметної області. Було розглянуто поняття та аналіз особливостей ІТ-стартапів, було проведено аналіз управління командою розробки ІТ-стартапу. Під час роботи над дипломним проєктом стало зрозуміло наскільки важливо якісно та ефективно управляти командою розробки ІТ-стартапу. Також дуже важливо приділяти увагу мотивації кожного окремого співробітника ІТ-стартапу. Якісне виконання всіх вищеописаних кроків доволі сильно полегшать роботу колективу, підніметься ефективність праці та буде запобігати професійному вигоранню співробітників.

2 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Архітектурна модель

Архітектурна модель доволі проста, але є дієвою. Вона складається з клієнтської частини, серверної частини, бази даних та зовнішньої інтеграції. Клієнтська частина (надалі фронтенд) реалізована з використанням бібліотеки React. React був обраний серед інших аналогів тому що він забезпечує динамічну побудову інтерфейсу, швидкий рендеринг та гнучке оновлення стану за допомогою хуків. Серверна частина (надалі бекенд) реалізована за допомогою Node.js з фреймворком Express. Node.js+Express був обраний серед інших аналогів тому що ця комбінація дозволяє обробляти HTTP-запити, працювати з базою даних, керувати авторизацією та реалізовувати REST API. Як базу даних буде використана PostgreSQL як надійна реляційна СУБД для зберігання облікових даних користувачів, їхніх завдань, статистики активності та прогресу. Щодо зовнішньої інтеграції – система інтегрується з API сервісу Trello для автоматичного імпорту завдань із Kanban-дошок, що дозволяє очільнику групи централізовано керувати завданнями, а користувачам — синхронізувати особисті плани з командною роботою. Далі на рисунку ілюстративно показана архітектурна модель (рис.2.1.1).



Рисунок 2.1 – архітектурна модель

2.2 Популярні методи управління командою розробки

2.2.1 Аналіз популярних методів

Ще нещодавно життєвий цикл ІТ-проектів розглядався як традиційна водоспадна модель (англійською waterfall). В основу цієї моделі покладено послідовний розвиток проекту[7]. Тобто наступна задача реалізується тільки тоді, коли закінчиться попередня. Водоспадна модель застосовується зазвичай для короткострокових та невеликих за масштабами проектів. Але зараз великої популярності набувають «гнучкі» методи управління ІТ-проектами та ІТ-стартапами.

Зокрема Agile-менеджмент є гнучким підходом, який базується на декількох похідних методиках розробки, зокрема, Scrum, Kanban, котрі ми розглянемо нижче. Суть методології полягає у розділенні процесу реалізації продукту на дрібні етапи для мінімізації ризиків. Тривалість цих фаз може варіюватися, в основному, термінами від одного тижня до місяця. Цей інструмент дозволяє

визначати послідовність виконання проекту, передбачати зміни та доповнення, виставляти пріоритети та постійно комунікувати з клієнтом.

Першочергово розглянемо метод Scrum. Перед початком розробки відбувається його планування, до якого входить – терміни здачі проекту, визначення пріоритетів та спільні обговорення ключових завдань. Під словом «спільні» мається на увазі, так звана, Scrum-команда, до складу якої входять: власник продукту, проєктний менеджер та команда розробників. По завершенню кожного етапу розробки, згідно з планом, відбувається демонстрація вже напрацьованих матеріалів. Після цієї демонстрації відбувається аналіз та вдосконалення продукту[7]. Але важливо відзначити, що Scrum — це лише одна частина гігантської головоломки Agile [8].

Другим основним методом є Kanban(з японської сигнальна дошка). Метод Kanban є доволі гнучким та більш спрямований на невеликі проєкти, планування яких не займає багато часу. Постановка завдань відбувається за допомогою kanban-дошки, бо саме на цій дошці розміщуються список усіх задач та виконавців. Система є дуже прозорою, аже кожен співробітник має доступ до всіх завдань та статусу їх виконання[9]. Виділяють три основні фази виконання:

1. To Do (почати роботу над завданням);
2. Doing (завдання в процесі виконання);
3. Done (завдання виконано)[9].

Саме такий простий метод розподілення надає команді можливість спостерігати за процесом розробки.

2.2.2 Порівняння інструментів та методів управління

Таблиця 2.1 - порівняння інструментів та методів управління.

Пункти порівняння	Scrum	Kanban
Структура	Чітко визначена структура	Гнучка
Ролі	Власник продукту, проєктний менеджер, команда розробки	Немає обов'язкових ролей
Планування	Чітке фіксоване планування	Без фіксованого планування
Часові обмеження	Так	Ні
Візуалізація процесу	Scrum-дошка з поділом задач на спринт	Канбан-дошка з колонками (To Do, In Progress, Done)
Зміни під час виконання	Небажані	Дозволені будь-коли
Покращення процесу	Ретроспектива після кожного спринту	Постійне вдосконалення
Обсяг роботи	Визначається на початку	Обмежується «WIP»
Для кого підходить	Для команд із стабільним ритмом	Для гнучких процесів, підтримки, обслуговування систем

2.2.3 Недоліки та проблеми інструментів та методів управління

Таблиця 2.2 - недоліки та проблеми інструментів та методів управління

Пункти порівняння	Scrum	Kanban
Складність впровадження	Вимагає значних зусиль для навчання ролей і процесів	Здається доволі простим, але потребує зрілості команди для ефективного використання
Гнучкість	Обмежена	Занадто велика
Чіткість дедлайнів	Спринт має чіткий дедлайн, але може спричинити стрес(наприклад через дуже короткий проміжок часу)	Відсутність дедлайнів може знизити мотивацію
Контроль за обсягом роботи	Доволі складне оцінювання під час планування	Потрібне чітке «WIP», бо без нього існує ризик перевантаження
Фокус на команді	Постійна командна робота, тому як наслідок – важко працювати в ізоляції	Менше командної взаємодії може мати як наслідок зменшення командного духу
Фіксація обсягу задач	Під час спринту вже неможливо додати нові задачі	Постійне додавання задач є ризиком втрати пріоритетів
Відстеження процесу	Вимагає постійних мітингів та звіти	Більше самодисципліни, аніж відстеження

2.3 Структура модулів системи

Всього в структурі можна виділити 5 основних модулів:

1. Модуль аутентифікації та авторизації
2. Модуль управління завданнями
3. Модуль синхронізації з Trello
4. Модуль статистики користувача
5. Модуль інтерфейсу користувача

Модуль аутентифікації та авторизації є першим модулем та він є безумовно важливим, бо без нього неможливо буде структурувати процес виконання задач. Як коротку характеристику модулю аутентифікації та авторизації можна зазначити наступне:

- Модуль забезпечує реєстрацію та вхід користувачів.
- Модуль реалізований на бекенді через обробку POST-запитів `/api/register` та `/api/login`.
- Модуль відповідає за перевірку паролів, збереження користувача в базі даних та за базову валідацію.

Модуль управління завданнями це другий модуль, котрий також є дуже важливим для цього web-застосунку. Завдяки ньому користувач може переглядати, починати виконання або завершувати виконання задачі або завдання. У свою чергу завдання поділяються на типи: «Щоденні», «Тижневі», «Місячні» (що є аналогом колонок Kanban). Також завдання мають статуси: «До виконання», «Виконую», «Виконано».

Далі йде модуль синхронізації з Trello. Trello – це онлайн платформа для управління проектами та задачами [10]. Учасників можна додавати не тільки в команди, а й на конкретні дошки і окремі картки та додавати на кожну задачу/картку можна встановити свій дедлайн [11]. Тож завдяки цьому модулю можливе отримання списку карток з певного списку. Також для кожного типу завдань/задач у системі відповідає колонці на Trello-дошці. А імпортовані завдання додаються в локальний список задач користувача.

Модуль статистики користувача фіксує виконання завдань та нараховує бали за активність, це напряду підвищує статус користувача (1. Новачок, 2. Просунутий, 3. Експерт). Ця статистика зберігається як локально, так і на сервері в Базі Даних.

Модуль інтерфейсу користувача побудований на Tailwind CSS для швидкого стилізування компонентів. В ньому реалізована адаптивність, темна тема, прогрес-бар та кнопки взаємодії. Для анімацій та динамічного оновлення UI використовується Framer Motion.

2.4 Обґрунтування вибору структури

Я обрав цю систему бо:

- Завдяки ній можна розширювати функціонал без зміни загальної архітектури.
- Має зрозумілу логіку та межу розділення відповідальностей (фронтент відповідає за UI, а бекенд – за загальну безпеку та за безпеку даних користувача).
- Має зручну інтеграцію з зовнішніми сервісами
- Має можливість автономної роботи (за рахунок локального збереження даних). Навіть якщо з сервером або з базою даних виникнуть неполадки – дані завжди будуть збережені.
- Має командну орієнтацію, бо Trello-інтеграція дозволяє координувати завдання між учасниками проєкту.

2.5 Висновки до розділу

У висновку хочу зазначити, що особисто для мене підхід до вибору всіх інструментів є доволі зрозумілим та аргументованим. Я працюю один, більш-менш чіткий план web-застосунку є, але він буде постійно вдосконалюватись під час написання дипломної роботи.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка програмного коду серверу

Як зазначав раніше основу для REST API складає Node.js + Express.js, бо ця «двійка» дозволяє легко створювати RESTful API та масштабувати проєкт у майбутньому.

Для обґрунтування вибору Node.js для бекенду хочу порівняти його з аналогом у вигляді Django (на основі мови програмування Python) у таблиці 3.1:

Таблиця 3.1 – Порівняння Node.js [12] з Django [13]

Пункти порівняння	Node.js	Django
Продуктивність	Висока.	Нижче.
Гнучкість	Максимальна.	Строгий фреймворк із фіксованою структурою.
Масштабованість	Добре масштабується.	Більше монолітний.
Сфера застосування	Чудово підходить для реального часу. Наприклад: чати або Trello-аналоги.	Частіше використовується у, наприклад, CRM або у блогах.

Основна СУБД це PostgreSQL бо це реляційна система, котра є зручною для структурованих даних і є доволі хорошим вибором для зберігання даних користувачів та статистики. Для обґрунтування вибору PostgreSQL як СУБД хочу порівняти його з аналогом у вигляді MongoDB у таблиці 3.2:

Таблиця 3.2 – Порівняння PostgreSQL [14] з MongoDB [15]

Пункти порівняння	PostgreSQL	MongoDB
Тип СУБД	Реляційна (SQL).	Документоорієнтована (NoSQL).
Схема даних	Чітка схема.	Схеми гнучкі, але немає цілісності зв'язків.
Продуктивність при складних запитах	Висока.	Слабша.
Сфера застосування	Підходить для задач з чіткими структурами, наприклад: користувачі, задачі, аналітика.	Підходить для гнучких схем даних.
Інструменти аналітики	Потужні інструменти агрегування.	Обмежені інструменти, складні для багаторівневого аналізу.

Як короткий та ємкий опис бекенду, надам основні API-маршрути:

- POST /api/register – відповідає за реєстрацію нового користувача
- POST /api/login – відповідає за авторизацію вже існуючого користувача
- GET /api/tasks – відповідає за отримання задач
- PATCH /api/tasks/:id – відповідає за оновлення статусу задач
- GET /api/user-stats – відповідає за зчитування статистики
- POST /api/user-stats – відповідає за збереження статистики

3.2 Розробка інтерфейсу додатку

Основною бібліотекою для побудови користувацького інтерфейсу я вибрав React.js. Ця бібліотека є доволі ємкою, цікавою та є доволі простою як для спеціаліста, так і для новачка. Вона забезпечує високу продуктивність та компонентну структуру. Це все дуже сильно полегшує розробку інтерфейсу з динамічним оновленням даних, наприклад – стан завдань, прогрес, статистики, тощо).

Під час написання дипломної роботи мій особистий вибір стояв між JavaScript-бібліотекою React та JavaScript-фреймворком Vue. Задля аргументації вибору JavaScript-бібліотеки React нижче наведена таблиця 3.3.

Таблиця 3.3 – Порівняння React.js [16] та Vue.js [17]

Пункт порівняння	JavaScript-бібліотека React	JavaScript-фреймворк Vue
Популярність	Є найпопулярнішою бібліотекою, котра активно підтримується великою спільнотою та корпорацією Meta.	Є менш популярним, котра підтримується спільнотою.
Гнучкість	Надзвичайно гнучка (можливість обирання бібліотек для маршрутизації, стану тощо).	Менш гнучка при нестандартних рішеннях.
Компонентна модель	JSX дає повний контроль над логікою та структурою компонента.	HTML-шаблони простіші, але менш виразні.
Продуктивність	Хуки дозволяють ефективно керувати станом без класів.	Більш складна логіка при складних сценаріях.
Кросплатформеність	React Native є стабільним та популярним.	Vue Native є, але не славиться стабільністю та популярністю.

Для швидкого та адаптивного стилювання я використав Tailwind CSS, бо ця бібліотека дає можливість писати стилі в inline-форматі, не відволікаючись на створення окремих CSS-файлів, це значно пришвидшує швидкість написання програми.

Для анімації Kanban-карток і плавного інтерфейсу був використаний Framer Motion, бо він є доволі простим та зрозумілим майже для кожного.

Функціональність додатку:

- Реалізована автентифікація користувача, а саме:
 - Реєстрація (рис.3.1)

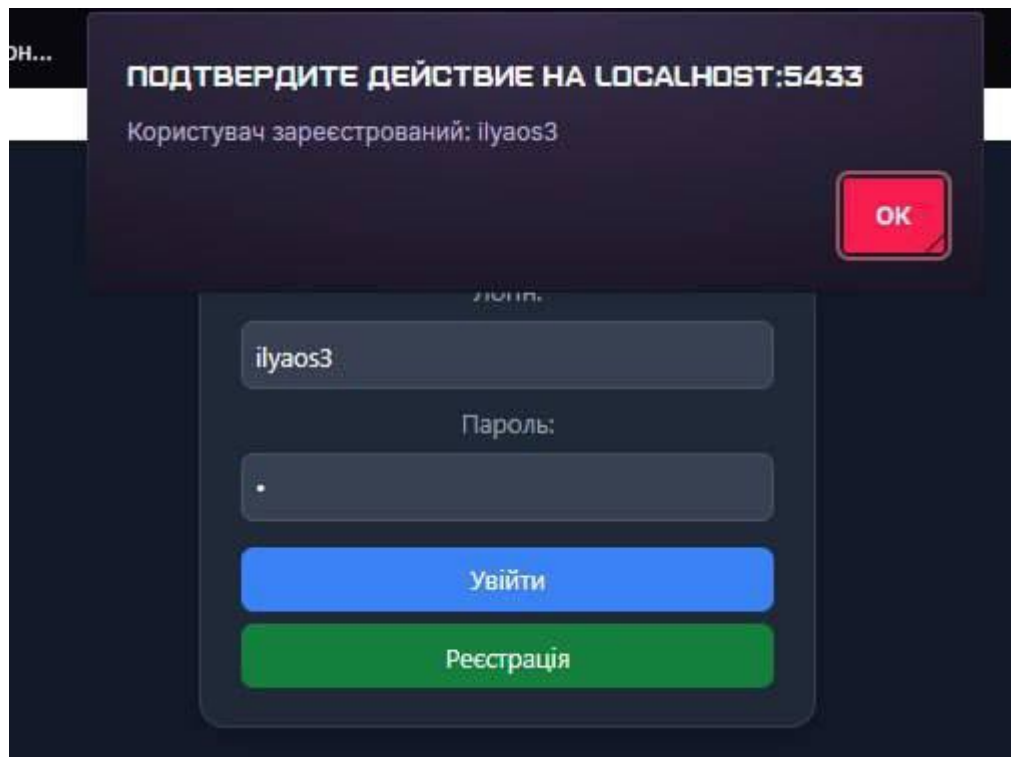


Рисунок 3.1 – реєстрація користувача

- Вхід
- Перевірка на реєстрацію вже існуючого логіну(рис.3.2)

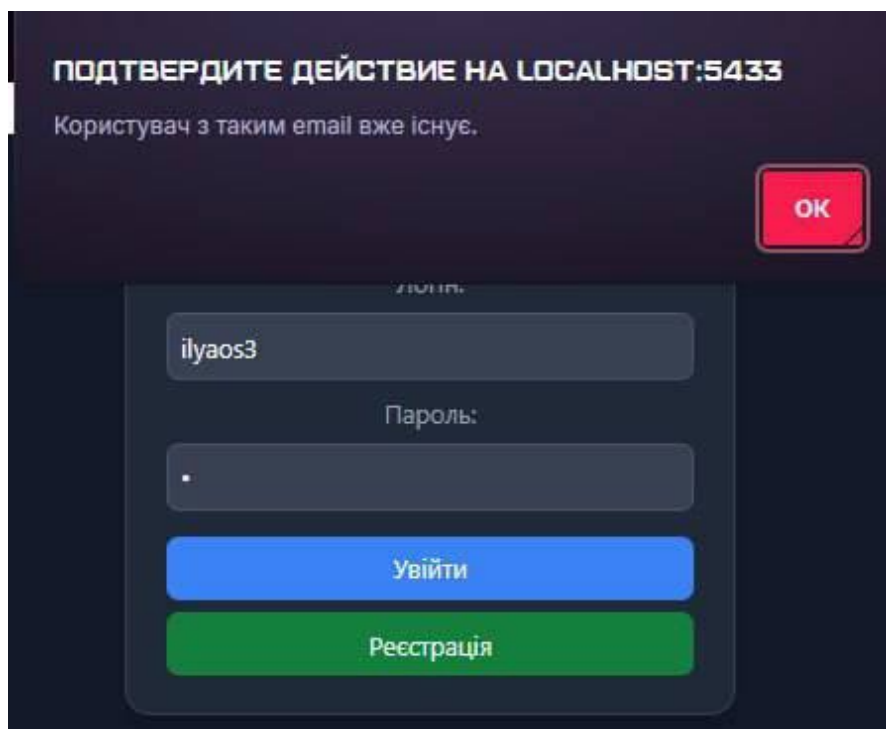


Рисунок 3.2 – перевірка на реєстрацію вже існуючого логіну

- Вибір типу задач (щоденні, тижневі, місячні). (рис.3.3)

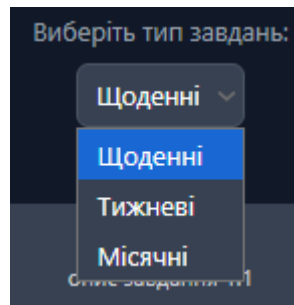


Рисунок 3.3 – вибір типу задач

- Перегляд та зміна статусів задач у вигляді Kanban-дошки. (рис.3.4-3.6)

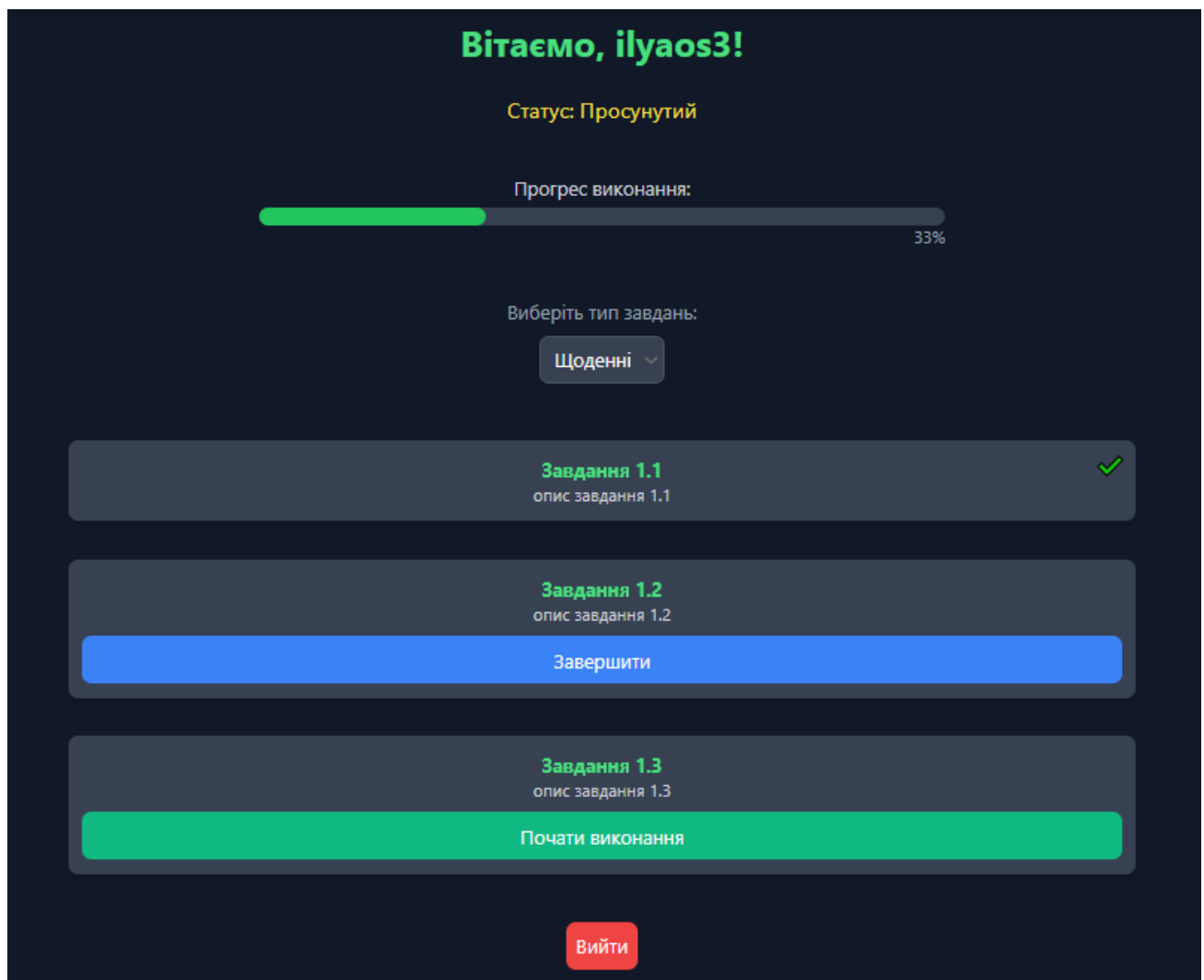


Рисунок 3.4 – зміна статусів Kanban-дошки

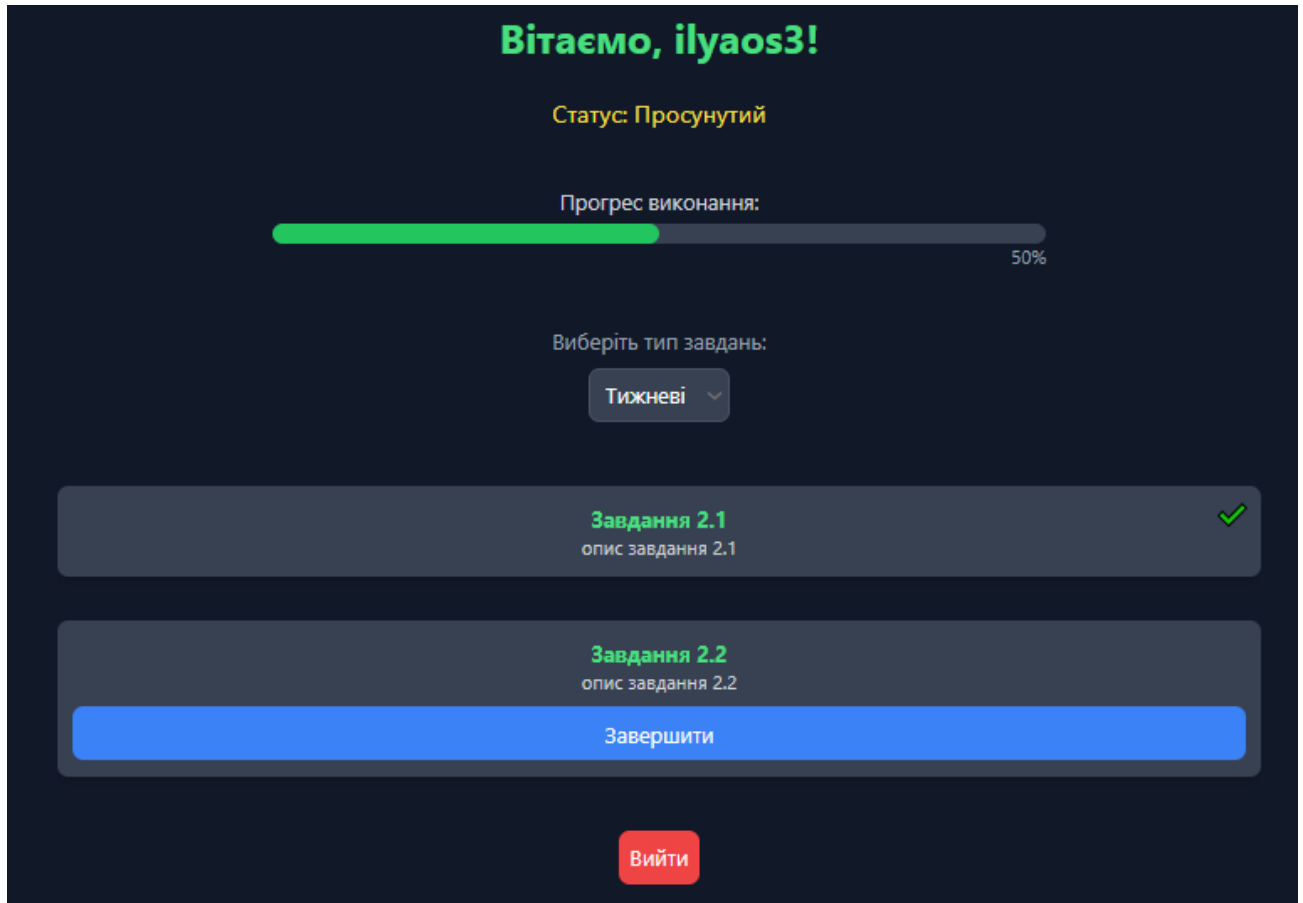


Рисунок 3.5 – зміна статусів Kanban-дошки

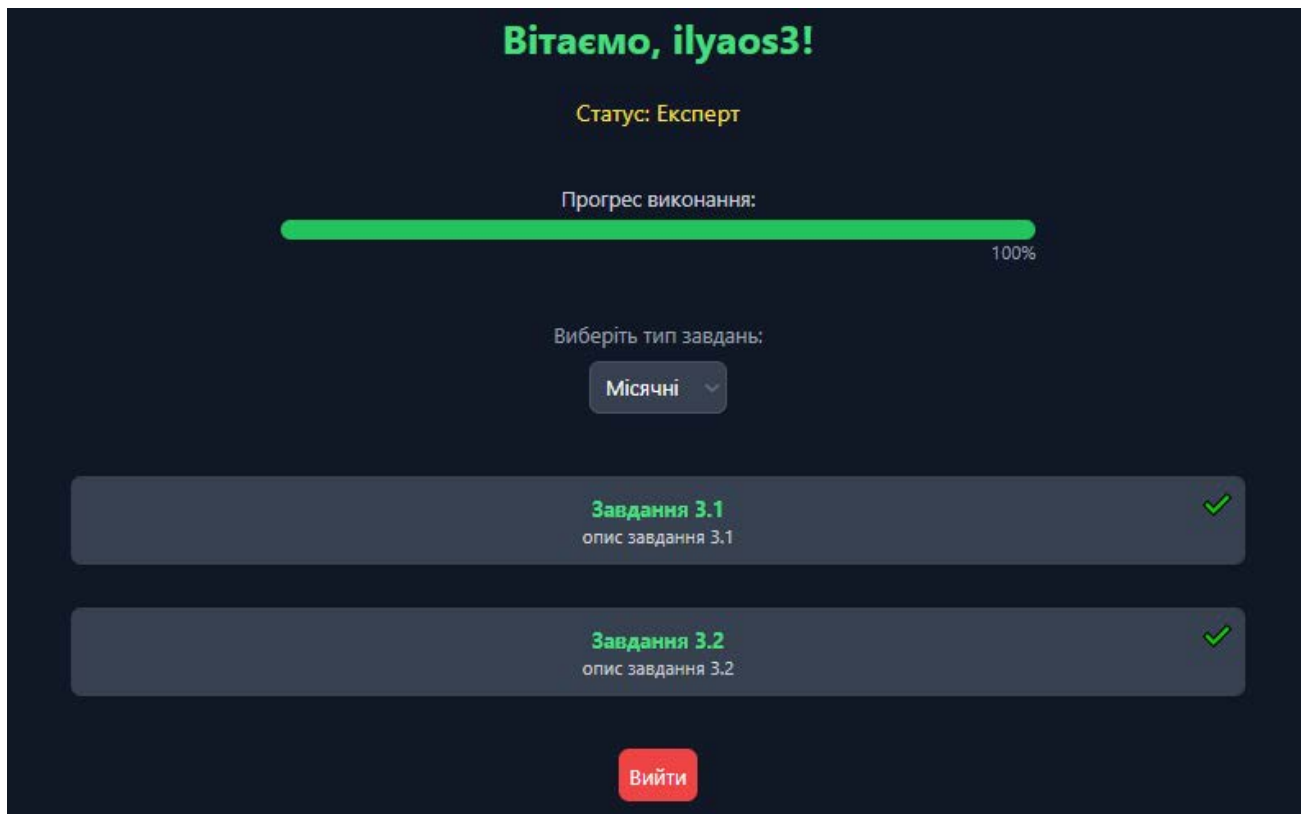


Рисунок 3.6 – зміна статусів Kanban-дошки

- Прогрес-бар виконання задач(рис.3.7).

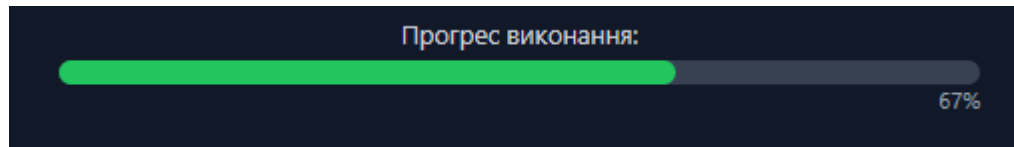


Рисунок 3.7 – прогрес-бар виконання задач

- Реєстрація іншого користувача та його прогрес + таблиця з БД(рис.3.8 - 3.10)

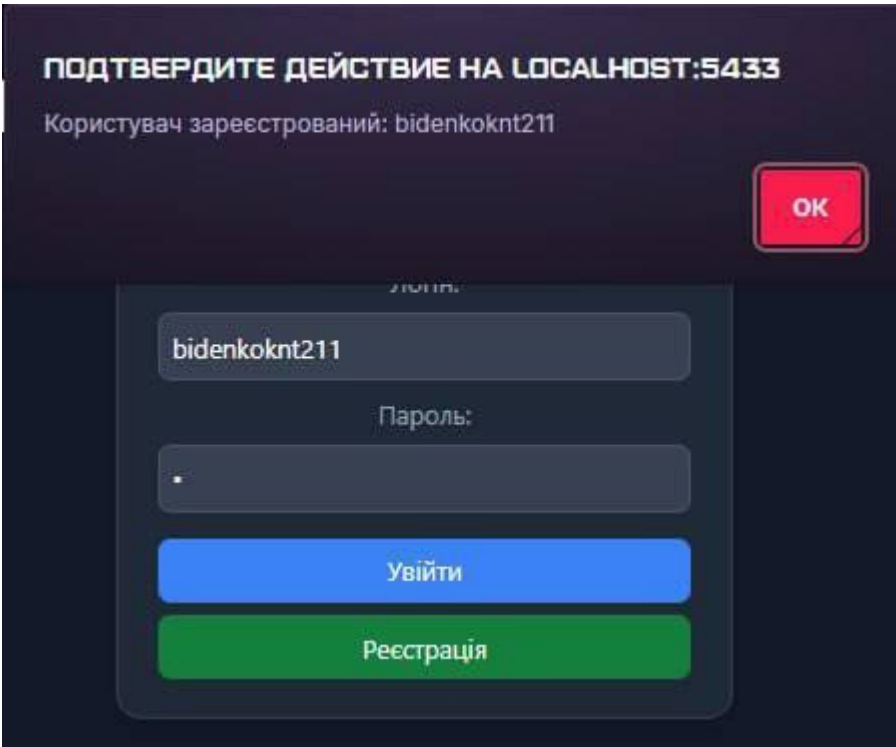
A registration form titled "ПОДТВЕРДИТЕ ДЕЙСТВИЕ НА LOCALHOST:5433". It states "Користувач зареєстрований: bidenkoknt211". The form includes a text input field with "bidenkoknt211", a password field labeled "Пароль:" with a single character visible, and two buttons: "Увійти" (blue) and "Реєстрація" (green). An "OK" button is in the top right corner.

Рисунок 3.8 – реєстрація іншого користувача

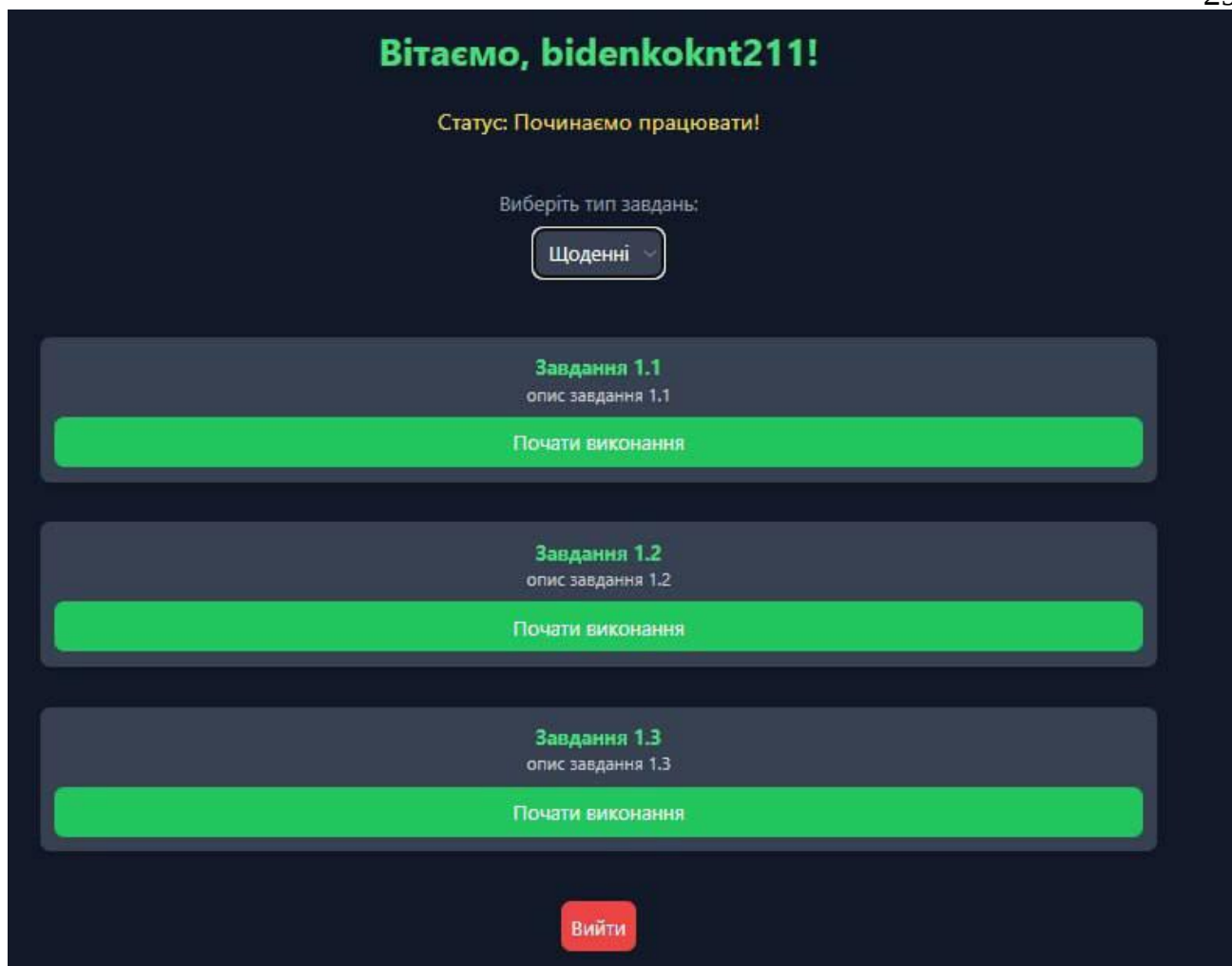


Рисунок 3.9 – прогрес нового користувача

```
postgres=# SELECT * FROM users;
```

id	email	password	score	tasks_completed	rank_position
1	ilyaos3	1	0	0	Новачок
2	bidenkoknt211	2	0	0	Новачок

(2 rows)

Рисунок 3.10 – таблиця у БД

- Показ статусів користувача на основі результатів(рис.3.11).

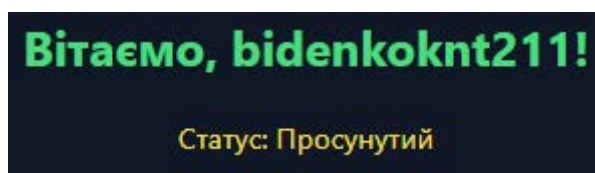


Рисунок 3.11 – показ статусу користувача

- Також, для гнучкості у використанні, web-застосунок було адаптовано для всіх типів екранів, що забезпечує можливість користуватися сайтом з будь якого пристрою (рис. 3.12).

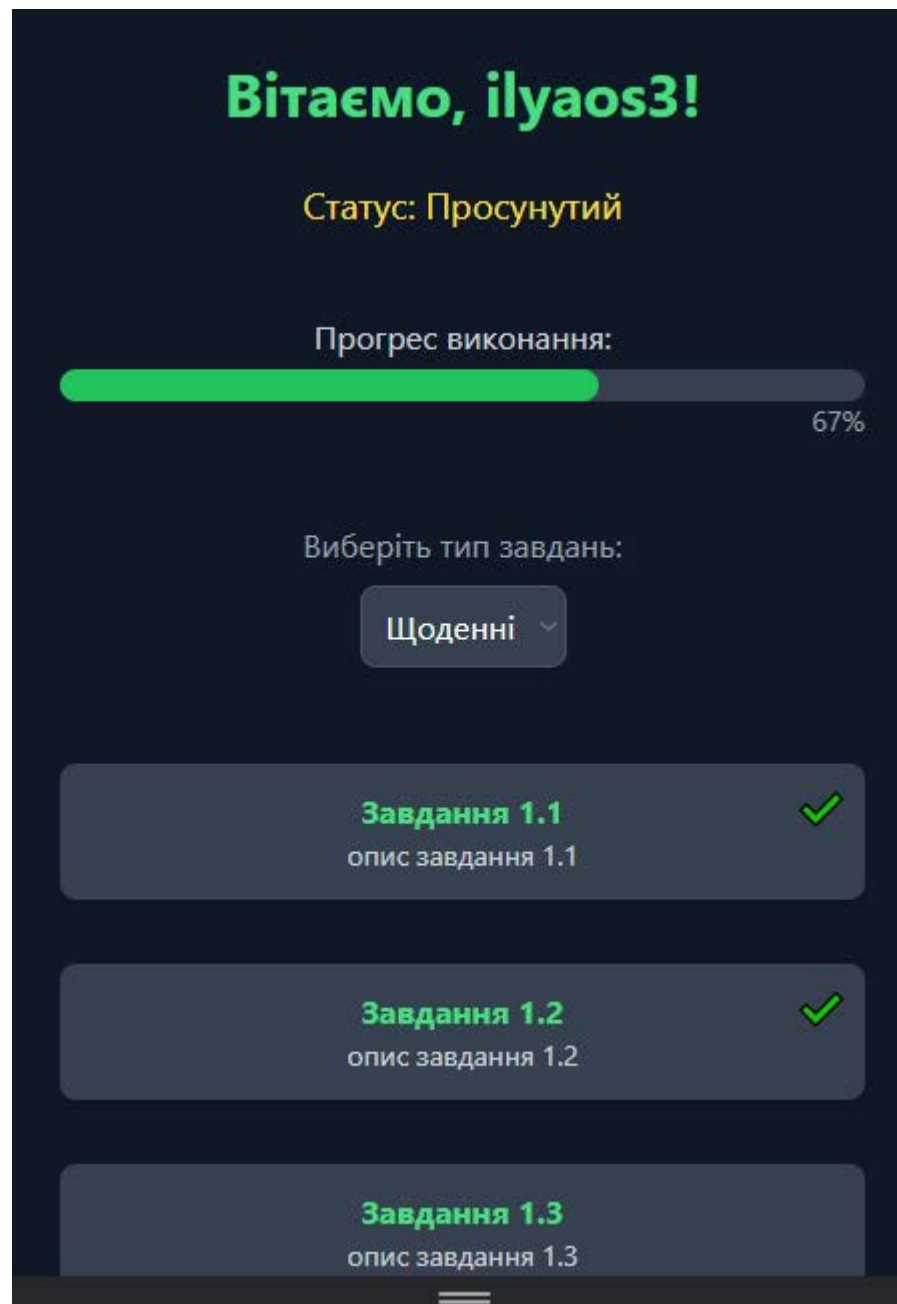


Рисунок 3.2.12 – web-застосунок у мобільному браузері

3.3 Висновки до розділу

В даному розділі було описано програмну реалізацію додатку. Його основною функцією є: активне відстежування виконання задачі, тобто на якому етапі перебуває кожен співробітник та його статус для подальшого розуміння

професіоналізму співробітника. Було вирішено реалізувати окремий web-застосунок з інтуїтивно зрозумілим та зовнішньо приємним інтерфейсом користувача. Фронтенд був реалізований завдяки бібліотеці React.js, а бекенд – Node.js+Express.js. Реалізований web-застосунок був написаний у середовищі розробки Microsoft Visual Studio 2022.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Тестування інтерфейсу додатку

Одразу після завершення розробки web-застосунку потребується його ретельне тестування. Основне це тестування Web UI. Спочатку потрібно перевірити роботоспроможність застосунку у всіх браузерях, приділяючи увагу основним – Google Chrome, Opera, Mozilla Firefox, Microsoft Edge та Safari.

Щоб протестувати Web UI я використав Chrome dev tools. За допомогою цього інструменту я перевіряв наскільки застосунок адаптований до мобільних телефонів, планшетів та інших популярних електронних пристроїв.

Під час перевірки Web UI через браузер Опера були виділені основні пункти, котрі зазначені у таблиці 4.1:

Таблиця 4.1 – основні пункти перевірки Web UI

№	Крок	Очікуваний результат	Фактичний результат	Статус
1	Зареєструвати нового користувача та пройти авторизацію	Користувач буде зареєстрований та успішно пройде авторизацію	Дані нового користувача, вказані при реєстрації, було успішно збережено у БД	Успішно виконано
2	Спробувати змінити тип задач	Список задач змінюється	Список задач успішно змінюється відповідно до його типу	Успішно виконано
3	Оновити задачу у застосунку	Дані змінюються	Отримано підтвердження (Задача успішно оновлена на сервері)	Успішно виконано
3	Перевірити, чи відображено новий статус без reload	Статус задачі змінено на «Виконую»	Змінився одразу після відповіді сервера	Успішно виконано
4	Оновити сторінку	Дані залишаються після перезавантаження	Дані збереглися, відвантажено з БД	Успішно виконано

Під час тестування Web UI через браузер Opera та Chrome dev tools у консолі не було помилок та все працювало коректно (рис. 4.1):

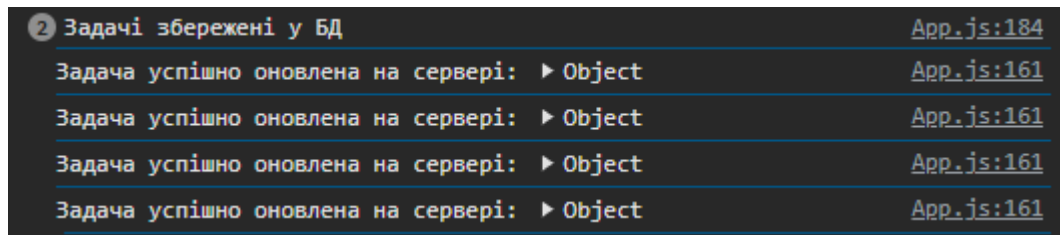


Рисунок 4.1 – консоль під час тестування Web UI

4.2 Тестування сервера

Мій сервер відповідає за запуск вебінтерфейсу застосунку та зв'язку з даними, що обробляються та записуються до потрібних файлів за допомогою скрипту. Для тестування роботи мого сервера у Docker середовищі, я спочатку перевіряв що Docker контейнер успішно запущений за допомогою команди `sudo docker ps` –а. Далі потрібно чи доступний сервер за адресою `localhost:5000`. Для цього я відкрив веббраузер Opera та ввів адресу `localhost:5000` у рядок пошуку та побачив інтерфейс мого web-застосунку. Саме це свідчить про те, що сервер успішно функціонує (рис.4.2).



Рисунок 4.2– скриншот localhost:5000

4.3 Висновки за розділом

Щоб протестувати web-застосунок мені знадобились наступні інструменти:

- Веббраузер Opera (або можливий будь-який інший веббраузер)
- Google Chrome dev tools
- Docker команди.

Під час тестування web-застосунок успішно пройшов перевірку на

працездатність Web UI та працездатність серверу. В ході перевірки не було знайдено помилок, котрі могли би вплинути на основні функції додатку та їх роботу.

ВИСНОВКИ

Під час розробки дипломного проєкту було створено web-застосунок для активної мотивації та управління командою розробки ІТ-стартапу. В цьому проєкті було використані наступні технології: бібліотека React на основі мови програмування JavaScript, програмна платформа Node.js та Express.js, реляційна база даних PostgreSQL та персональний комп'ютер з операційною системою Microsoft Windows 10 Pro та процесором Intel Core I5 3570-k.

Основною функцією розробленого web-застосунку є те, що, завдяки Kanban-дошці, процес став максимально інтуїтивно зрозумілим та не вимагаючим великої кількості часу, щоб була змога більш зконцентрувавшись на виконанні задач. Керівник команди розробки, в свою чергу, може переглядати хто зі співробітників перебуває на якому етапі та формувати таблицю лідерів з подальшими винагородами для ще більшої мотивації співробітників.

В результаті дослідження було проаналізовано особливості ІТ-стартапів та керування командою розробки, розглянуто методи керування співробітників у ІТ-стартапі, створено програмний web-застосунок для автоматизації методів мотивації та управління командою розробки ІТ-стартапу.

Галуззю використання є майбутні або вже існуючі ІТ-стартапи, також ІТ-компанії котрі мають у своєму складі відділи ІТ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Стартап — визначення, особливості, розвиток. – [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Стартап>
2. Менеджмент стартап-проектів: навчальний посібник / КПІ ім. Ігоря Сікорського. – [Електронний ресурс]. – Режим доступу: https://marketing.kpi.ua/files/studentam/metodichki/Менеджмент_стартап_проектів.pdf
3. Поняття стартапу як бізнес-моделі. – [Електронний ресурс]. – Режим доступу: <https://buklib.net/books/34104/>
4. Управління стартапами: теоретичні аспекти. – [Електронний ресурс]. – Режим доступу: <https://studentam.net.ua/content/view/3329/97/>
5. Теоретичні основи управління стартап-проектами / Науковий збірник Острозької академії. – [Електронний ресурс]. – Режим доступу: <https://eprints.oa.edu.ua/id/eprint/557/1/teoret.pdf>
6. Оцінка персоналу: методи, які повинен знати кожен HR. – [Електронний ресурс]. – Режим доступу: <https://hurma.work/blog/oczinka-personalu-metody-yaki-povinen-znati-kozhnij-hr>
7. Методи управління IT-проектами. – [Електронний ресурс]. – Режим доступу: <https://armedsoft.com/ua/blog/metody-upravlinnya-it-proektamy>
8. What is Scrum methodology? – [Електронний ресурс]. – Режим доступу: <https://career.softserveinc.com/uk-ua/stories/what-is-scrum-methodology>
9. Методологія Kanban: можливості та принципи роботи. – [Електронний ресурс]. – Режим доступу: <https://para.school/blog/management/metodologija-kanban-vozmozhnosti-i-printsipu-rabotu>
10. Що таке Trello? – [Електронний ресурс]. – Режим доступу: <https://hostiq.ua/blog/ukr/what-is-trello-2/>
11. Що таке Trello. – [Електронний ресурс]. – Режим доступу: <https://para.school/blog/management/chto-takoe-trello>
12. Node.js – [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Node.js>
13. Django – [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Django>

14. PostgreSQL – [Электронный ресурс]. – Режим доступа:

<https://uk.wikipedia.org/wiki/PostgreSQL>

15. MongoDB – [Электронный ресурс]. – Режим доступа:

<https://uk.wikipedia.org/wiki/MongoDB>

16. React – [Электронный ресурс]. – Режим доступа:

<https://uk.wikipedia.org/wiki/React>

17. Vue.js – [Электронный ресурс]. – Режим доступа:

<https://uk.wikipedia.org/wiki/Vue.js>

ДОДАТОК А

Текст програми

А.1 Скрипт app.js

```
import { useState, useEffect } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import axios from 'axios';

const usersDB = new Map();
const userStats = new Map();

export default function AuthApp() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [tasks, setTasks] = useState([]);
  const [loggedIn, setLoggedIn] = useState(false);
  const [currentUser, setCurrentUser] = useState("");
  const [selectedTaskType, setSelectedTaskType] = useState('Щоденні');

  useEffect(() => {
    if (currentUser) {
      fetchUserStatsFromBackend(currentUser);
      fetchTasksFromBackend(currentUser);
    }
  }, [currentUser]);

  useEffect(() => {
    if (currentUser) {
      localStorage.setItem(`${currentUser}_tasks`, JSON.stringify(tasks));
    }
  }, [tasks, currentUser]);
```

```
const fetchTasksFromBackend = async (email) => {
  try {
    const response = await axios.get('http://localhost:5000/api/tasks', {
      params: { email }
    });
    setTasks(response.data);
  } catch (error) {
    console.error('Помилка при отриманні завдань:', error);
  }
};
```

```
const fetchUserStatsFromBackend = async (email) => {
  try {
    const res = await axios.get('http://localhost:5000/api/user-stats', {
      params: { email }
    });
    if (res.data) {
      userStats.set(email, res.data);
      localStorage.setItem(email, JSON.stringify(res.data));
    }
  } catch (err) {
    console.error("Помилка отримання статистики:", err);
  }
};
```

```
const fetchTrelloTasks = async (taskType, currentUserEmail) => {
  if (!currentUserEmail) {
    console.error('Помилка: немає email для користувача!');
    return;
  }
}
```

```
const apiKey = '34e31a388c298cb6b4dc151ff0e68e2e';
```

```

const token =
'ATTA0df4fcf73a68b6e346f5081c88f0a3838f2bba40ca22008736e48f5ebc709895EA
FADE83';

const listIdMap = {
  'Щоденні': '67eed019a68801a13522101f',
  'Тижневі': '67eed9b54cf65108876059eb',
  'Місячні': '67eed9c2fa9637141cd4af4d'
};

try {
  const response = await
fetch(`https://api.trello.com/1/lists/${listIdMap[taskType]}/cards?key=${apiKey}&to
ken=${token}`);
  const data = await response.json();

  const formattedTasks = data.map(task => ({
    id: task.id,
    title: task.name,
    description: task.desc,
    status: 'До виконання',
    assignedTo: null,
    completed: false,
    user_email: currentUserEmail
  }));

  setTasks(formattedTasks);
  await saveTasksToBackend(formattedTasks);
} catch (error) {
  console.error('Помилка при отриманні завдань з Trello:', error);
}
};

```

```

const handleRegister = async () => {
  try {
    const response = await axios.post('http://localhost:5000/api/register', { email,
password });
    if (response.data.success) {
      alert(`Користувач зареєстрований: ${email}`);
      const initialStats = { score: 0, tasksCompleted: 0, badge: 'Починаємо
працювати!' };
      userStats.set(email, initialStats);
      await axios.post('http://localhost:5000/api/user-stats', { email, stats:
initialStats });
      localStorage.setItem(email, JSON.stringify(initialStats));
    } else {
      alert(response.data.message || 'Помилка при реєстрації');
    }
  } catch (error) {
    console.error('Помилка при реєстрації:', error);
    alert('Помилка під час реєстрації');
  }
};

```

```

const handleLogin = async () => {
  try {
    const response = await axios.post('http://localhost:5000/api/login', { email,
password });
    if (response.data.success) {
      setLoggedIn(true);
      setCurrentUser(email);
    } else {
      alert(response.data.message || 'Невірний логін або пароль!');
    }
  } catch (error) {

```

```

        console.error('Помилка при вході:', error);
        alert('Помилка при з'єднанні з сервером');
    }
};

```

```

const handleLogout = () => {
    setLoggedIn(false);
    setCurrentUser("");
    setTasks([]);
};

```

```

const handleTaskTypeChange = (event) => {
    const newTaskType = event.target.value;
    setSelectedTaskType(newTaskType);
    if (currentUser) {
        fetchTrelloTasks(newTaskType, currentUser);
    }
};

```

```

const updateTaskStatus = (taskId, newStatus) => {
    setTasks(prevTasks =>
        prevTasks.map(task =>
            task.id === taskId && (task.assignedTo === null || task.assignedTo ===
currentUser)
            ? {
                ...task,
                status: newStatus,
                assignedTo: newStatus === 'Виконую' ? currentUser : null,
                completed: newStatus === 'Виконано'
            }
            : task
        )
    )
};

```

```
);
if (newStatus === 'Виконано') updateUserStats(taskId);
updateTaskStatusOnBackend(taskId, newStatus);
};
```

```
const updateTaskStatusOnBackend = (taskId, newStatus) => {
  axios.patch(`http://localhost:5000/api/tasks/${taskId}`, {
    status: newStatus,
    completed: newStatus === 'Виконано',
    user_email: currentUser
  })
  .then(response => {
    console.log('Задача успішно оновлена на сервері:', response.data);
  })
  .catch(error => {
    console.error('Помилка при оновленні задачі на сервері:', error);
  });
};
```

```
const saveTasksToBackend = async (tasksToSave) => {
  if (!currentUser) {
    console.error('Помилка: користувач не авторизований! Невідомий user_email.');
```

```
    return;
```

```
  }
```

```
const tasksWithUserEmail = tasksToSave.map(task => {
  if (!task.user_email) {
    task.user_email = currentUser;
  }
  return task;
});
```

```

    try {
      await axios.post('http://localhost:5000/api/tasks/bulk', { tasks:
tasksWithUserEmail });
      console.log("Задачі збережені у БД");
    } catch (error) {
      console.error("Помилка при збереженні задач у БД:", error);
    }
  };

const updateUserStats = async (taskId) => {
  const task = tasks.find(t => t.id === taskId);
  if (!task) return;

  let points = selectedTaskType === 'Щоденні' ? 10 : selectedTaskType ===
'Тижневі' ? 50 : 100;

  let user = userStats.get(currentUser) || { score: 0, tasksCompleted: 0, badge:
'Починаємо працювати!' };

  user.score += points;
  user.tasksCompleted += 1;

  if (user.score >= 500) user.badge = 'Експерт';
  else if (user.score >= 250) user.badge = 'Просунутий';
  else if (user.score >= 100) user.badge = 'Новачок';

  userStats.set(currentUser, user);
  localStorage.setItem(currentUser, JSON.stringify(user));

  await axios.post('http://localhost:5000/api/user-stats', {
    email: currentUser,
    stats: user
  });
};

```

```
});
```

```
};
```

```
const getTaskButtonText = (task) => {
  if (task.completed) return 'Виконано';
  if (task.status === 'До виконання') return 'Почати виконання';
  return 'Завершити';
};
```

```
const getStatusColor = (status) => {
  if (status === 'До виконання') return '#10B981';
  if (status === 'Виконую') return '#3B82F6';
  if (status === 'Виконано') return '#8B5CF6';
  return '#6B7280';
};
```

```
const getCompletionProgress = () => {
  if (tasks.length === 0) return 0;
  const completed = tasks.filter(t => t.completed).length;
  return Math.round((completed / tasks.length) * 100);
};
```

```
return (
  <div className="flex flex-col items-center gap-6 p-6 bg-gray-900 text-white
min-h-screen">
    {loggedIn ? (
      <>
        <h2 className="text-3xl font-bold text-green-400">Вітаємо,
{currentUser}!</h2>
        <p className="text-lg text-yellow-300">Статус:
{userStats.get(currentUser)?.badge || 'Починаємо працювати!'}</p>
```

```

<div className="w-full max-w-xl mt-4">
  <label className="text-gray-300">Прогрес виконання:</label>
  <div className="w-full bg-gray-700 rounded-full h-4 mt-1">
    <div className="bg-green-500 h-4 rounded-full transition-all
duration-500" style={{ width: `${getCompletionProgress()}%` }} />
  </div>
  <p className="text-right text-sm text-gray-
400">{getCompletionProgress()}%</p>
</div>

<div className="mt-4">
  <label className="block text-gray-400">Виберіть тип
завдань:</label>
  <select
    value={selectedTaskType}
    onChange={handleTaskTypeChange}
    className="p-2 border border-gray-600 rounded-lg bg-gray-700
text-white mt-2"
  >
    <option value="Щоденні">Щоденні</option>
    <option value="Тижневі">Тижневі</option>
    <option value="Місячні">Місячні</option>
  </select>
</div>

<div className="flex flex-wrap justify-center gap-6 w-full max-w-4xl
mt-4">
  <AnimatePresence>
    {tasks.map(task => (
      <motion.div
        key={task.id}
        initial={{ opacity: 0, scale: 0.95 }}

```

```

        animate={{ opacity: 1, scale: 1 }}
        exit={{ opacity: 0, scale: 0.95 }}
        transition={{ duration: 0.3 }}
        className="p-3 bg-gray-700 rounded-lg mt-2 shadow-md w-
full relative"
      >
        <h4 className="text-md font-bold text-green-
400">{task.title}</h4>
        <p className="text-sm text-gray-300">{task.description}</p>
        {task.completed && <span className="absolute top-2 right-2
text-green-400 text-xl">✔</span>}
        {task.status !== 'Виконано' && (
          <motion.button
            layout
            whileTap={{ scale: 0.95 }}
            animate={{ backgroundColor: getStatusColor(task.status)
}}
            transition={{ duration: 0.3 }}
            onClick={() => updateTaskStatus(task.id, task.status ===
'До виконання' ? 'Виконую' : 'Виконано')}}
            className="text-white p-2 rounded-lg mt-2 w-full"
          >
            {getTaskButtonText(task)}
          </motion.button>
        )}
      </motion.div>
    )}
  </AnimatePresence>
</div>

  <button className="bg-red-500 hover:bg-red-700 transition duration-
300 text-white p-2 rounded-lg mt-4" onClick={handleLogout}>

```

```

        Вийти
    </button>
</>
):(
    <div className="w-96 p-6 bg-gray-800 shadow-lg rounded-2xl border
border-gray-700">
        <h2 className="text-2xl font-bold text-center text-blue-
300">Авторизація</h2>
        <label className="block text-gray-400">Логін:</label>
        <input
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            placeholder="Логін"
            className="p-2 border border-gray-600 rounded-lg w-full bg-gray-
700 text-white my-2"
        />
        <label className="block text-gray-400">Пароль:</label>
        <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            placeholder="Пароль"
            className="p-2 border border-gray-600 rounded-lg w-full bg-gray-
700 text-white my-2"
        />
        <button className="bg-blue-500 hover:bg-blue-700 transition duration-
300 text-white p-2 rounded-lg w-full mt-2" onClick={handleLogin}>
            Увійти
        </button>
        <button className="bg-green-500 hover:bg-green-700 transition
duration-300 text-white p-2 rounded-lg w-full mt-2" onClick={handleRegister}>
            Реєстрація

```

```

        </button>
    </div>
  })
</div>
);
}

```

A.2 Скрипт server.js

```

const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const { Pool } = require('pg');

const app = express();
const port = 5000;

app.use(cors({
  origin: 'http://localhost:5433',
  methods: ['GET', 'POST', 'PATCH', 'DELETE'],
  allowedHeaders: ['Content-Type']
}));

app.use(bodyParser.json());

const pool = new Pool({
  user: 'postgres',
  host: 'localhost',
  database: 'postgres',
  password: 'vavle4real',
  port: 5432,
});

```

```
app.get('/', (req, res) => {
  res.send('Сервер працює!');
});
```

```
app.post('/api/register', async (req, res) => {
  const { email, password } = req.body;
  try {
    const checkUser = await pool.query('SELECT * FROM users WHERE email = $1', [email]);
    if (checkUser.rows.length > 0) {
      return res.json({ success: false, message: 'Користувач з таким email вже існує.' });
    }
    await pool.query('INSERT INTO users (email, password) VALUES ($1, $2)', [email, password]);
    res.json({ success: true });
  } catch (error) {
    console.error('Помилка при реєстрації:', error);
    res.status(500).json({ success: false, message: 'Помилка сервера' });
  }
});
```

```
app.post('/api/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await pool.query('SELECT * FROM users WHERE email = $1 AND password = $2', [email, password]);
    if (user.rows.length === 1) {
      res.json({ success: true });
    } else {
      res.json({ success: false, message: 'Неправильний логін або пароль' });
    }
  }
});
```

```

    }
  } catch (error) {
    console.error('Помилка при вході:', error);
    res.status(500).json({ success: false, message: 'Помилка сервера' });
  }
});

app.get('/api/tasks', async (req, res) => {
  const userEmail = req.query.email;
  try {
    const result = await pool.query('SELECT * FROM tasks WHERE user_email = $1', [userEmail]);
    res.json(result.rows);
  } catch (error) {
    console.error('Помилка при отриманні задач:', error);
    res.status(500).json({ error: 'Не вдалося отримати задач' });
  }
});

app.get('/api/user-stats', async (req, res) => {
  const { email } = req.query;
  try {
    const result = await pool.query('SELECT * FROM user_stats WHERE email = $1', [email]);
    if (result.rows.length > 0) {
      res.json(result.rows[0]);
    } else {
      res.status(404).json({ success: false, message: 'Статистика не знайдена' });
    }
  } catch (error) {
    console.error('Помилка отримання статистики:', error);
  }
});

```

```

    res.status(500).json({ success: false, message: 'Помилка при отриманні
статистики' });
  }
});

```

```

app.post('/api/tasks/bulk', async (req, res) => {

```

```

  const tasks = req.body.tasks;

```

```

  if (!Array.isArray(tasks)) {

```

```

    return res.status(400).json({ error: 'Очікується масив задач' });
  }

```

```

  const client = await pool.connect();

```

```

  try {

```

```

    await client.query('BEGIN');

```

```

    for (const task of tasks) {

```

```

      const { id, title, description, status, assignedTo, completed, email } = task;

```

```

      const exists = await client.query('SELECT 1 FROM tasks WHERE id = $1',
[id]);

```

```

      if (exists.rows.length === 0) {

```

```

        await client.query(

```

```

          `INSERT INTO tasks (id, title, description, status, completed,
assigned_to, user_email)

```

```

            VALUES ($1, $2, $3, $4, $5, $6, $7)`,

```

```

            [id, title, description, status, completed, assignedTo, email]

```

```

          );

```

```

        }

```

```

      }

```

```

    await client.query('COMMIT');

```

```

    res.status(200).json({ message: 'Успішно збережено задачі з Trello' });

    } catch (error) {
        await client.query('ROLLBACK');
        console.error('Помилка збереження задач:', error);
        res.status(500).json({ error: 'Помилка при збереженні задач' });
    } finally {
        client.release();
    }
});

app.post('/api/user-stats', async (req, res) => {
    const { email, stats } = req.body;
    try {
        const existing = await pool.query('SELECT * FROM user_stats WHERE email = $1', [email]);
        if (existing.rows.length > 0) {
            await pool.query('UPDATE user_stats SET score = $1, tasks_completed = $2, badge = $3 WHERE email = $4',
                [stats.score, stats.tasksCompleted, stats.badge, email]);
        } else {
            await pool.query('INSERT INTO user_stats (email, score, tasks_completed, badge) VALUES ($1, $2, $3, $4)',
                [email, stats.score, stats.tasksCompleted, stats.badge]);
        }
        res.json({ success: true });
    } catch (error) {
        console.error('Помилка збереження статистики:', error);
        res.status(500).json({ success: false, message: 'Помилка збереження статистики' });
    }
});

```

```

app.patch('/api/tasks/:id', async (req, res) => {
  const taskId = req.params.id;
  const { status, completed } = req.body;

  try {
    const result = await pool.query(
      'UPDATE tasks SET status = $1, completed = $2 WHERE id = $3
RETURNING *',
      [status, completed, taskId]
    );

    res.json(result.rows[0]);
  } catch (error) {
    console.error('Помилка при оновленні задачі:', error);
    res.status(500).json({ error: 'Не вдалося оновити задачу' });
  }
});

pool.connect((err, client, release) => {
  if (err) {
    return console.error('Помилка підключення до PostgreSQL:', err.stack);
  }
  console.log('Підключено до PostgreSQL!');
  release();
});

app.listen(port, () => {
  console.log(`Сервер API працює на http://localhost:${port}`);
});

```

A.3 Скрипт db.js

```
import pkg from 'pg';  
const { Pool } = pkg;  
  
export const pool = new Pool({  
  user: 'postgres',  
  host: 'localhost',  
  database: 'postgres',  
  password: 'vavle4real',  
  port: 5432,  
});
```

ДОДАТОК Б
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЗАПОРІЗЬКА ПОЛІТЕХНІКА"
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ
ДИПЛОМНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

**Програмна реалізація методів мотивації та
управління командної розробки ІТ-стартапу**
**Software Implementation of Methods for Motivation
and Management of an IT-Startup Team**

Виконав:
студент групи КНТ-211

Ілля БІДЕНКО

Керівник:
професор, к.т.н.

Валерій ДУБРОВІН

Запоріжжя, 2025

Рисунок Б.1 – Слайд №1

МЕТА, ОБ'ЄКТ І ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Об'єкт дослідження – процес програмної
розробки методів мотивації і управління
командою.**

**Предмет дослідження – методи управління
командою розробки.**

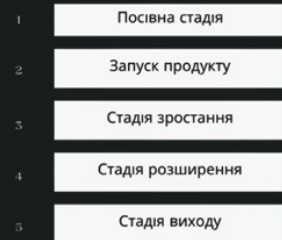
**Мета роботи – створення програмної
системи для мотивації та управління
командою розробки ІТ-стартапу.**

Рисунок Б.2 – Слайд №2

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

- Поняття та аналіз особливостей стартапів
- Аналіз управління командою розробки
- Постановка завдання роботи

Стадії розвитку стартапів



3

Рисунок Б.3 – Слайд №3

АРХІТЕКТУРНА МОДЕЛЬ



4

Рисунок Б.4 – Слайд №4

АНАЛІЗ ПОПУЛЯРНИХ МЕТОДІВ УПРАВЛІННЯ КОМАНДОЮ РОЗРОБКИ ПОРІВНЯННЯ ІНСТРУМЕНТІВ ТА МЕТОДІВ УПРАВЛІННЯ НЕДОЛІКИ ТА ПРОБЛЕМИ ІНСТРУМЕНТІВ ТА МЕТОДІВ УПРАВЛІННЯ

Пункти порівняння	Scrum	Kanban	Пункти порівняння	Scrum	Kanban
Структура	Чітко визначена структура	Гнучка	Складність впровадження	Вимагає значних зусиль для навчання ролей і процесів	Здається доволі простим, але потребує зрілості команди для ефективного використання
Ролі	Власник продукту, проєктний менеджер, команда розробки	Немає обов'язкових ролей	Гнучкість	Обмежена	Занадто велика
Планування	Чітке фіксоване планування	Без фіксованого планування	Чіткість дедлайнів	Спринт має чіткий дедлайн, але може спричинити стрес(наприклад через дуже короткий проміжок часу)	Відсутність дедлайнів може знизити мотивацію
Часові обмеження	Так	Ні	Контроль за обсягом роботи	Доволі складне оцінювання під час планування	Потрібне чітке «WIP», бо без нього існує ризик перевантаження
Візуалізація процесу	Scrum-дошка з поділом задач на спринт	Канбан-дошка з колонками (To Do, In Progress, Done)	Фокус на команді	Постійна командна робота, тому як наслідок – важко працювати в ізоляції	Менше командної взаємодії може мати як наслідок зменшення командного
Зміни під час виконання	Небажані	Дозволені будь-коли	Фіксація обсягу задач	Під час спринту вже неможливо додати нові задачі	Постійне додавання задач є ризиком втрати пріоритетів
Покращення процесу	Ретроспектива після кожного спринту	Постійне вдосконалення	Відстеження процесу	Вимагає постійних мітингів та звітів	Більше самодисципліни, аніж відстеження
Обсяг роботи	Визначається на початку	Обмежується «WIP»			
Для кого підходить	Для команд із стабільним ритмом	Для гнучких процесів, підтримки, обслуговування систем			

Рисунок Б.5 – Слайд №5

СТРУКТУРА МОДУЛІВ СИСТЕМИ



Рисунок Б.6 – Слайд №6

ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

			Пункти порівняння	PostgreSQL	MongoDB
Пункти порівняння	Node.js	Django	Тип СУБД	Реляційна (SQL).	Документоорієнтована (NoSQL).
Продуктивність	Висока.	Нижче.	Схема даних	Чітка схема.	Схеми гнучкі, але немає цілісності зв'язків.
Гнучкість	Максимальна.	Строгий фреймворк із фіксованою структурою.	Продуктивність при складних запитах	Висока.	Слабша.
Масштабованість	Добре масштабується.	Більше монолітний.	Сфера застосування	Підходить для задач з чіткими структурами, наприклад: користувачі, задачі, аналітика.	Підходить для гнучких схем даних.
Сфера застосування	Чудово підходить для реального часу. Наприклад: чати або Trello-аналоги.	Частіше використовується у, наприклад, CRM або у блогах.	Інструменти аналітики	Потужні інструменти агрегування.	Обмежені інструменти, складні для багаторівневого аналізу.

Пункт порівняння	JavaScript-бібліотека React	JavaScript-фреймворк Vue
Популярність	Є найпопулярнішою бібліотекою, котра активно підтримується великою спільнотою та корпорацією Meta.	Є менш популярним, котра підтримується спільнотою.
Гнучкість	Надзвичайно гнучка (можливість обирання бібліотек для маршрутизації, стану тощо).	Менш гнучка при нестандартних рішеннях.
Компонентна модель	JSX дає повний контроль над логікою та структурою компонента.	HTML-шаблони простіші, але менш виразні.
Продуктивність	Хуки дозволяють ефективно керувати станом без класів.	Більш складна логіка при складних сценаріях.
Кросплатформеність	React Native є стабільним та популярним.	Vue Native є, але не славиться стабільністю та популярністю.

Рисунок Б.7 – Слайд №7

ФУНКЦІОНАЛЬНІСТЬ ДОДАТКУ

ПОДТВЕРДИТЕ ДЕЙСТВИЕ НА LOCALHOST:5433

Користувач зареєстрований: iIyaos3

Виберіть тип завдання:

Щоденні

Щоденні

Тижневі

Місячні

Вітаємо, iIyaos3!

Статус: Прогнутий

Прогрес виконання: 50%

Виберіть тип завдання: Тижневі

Завдання 2.1

Завдання 2.2

```
postgres=# SELECT * FROM users;
 id | email | password | score | tasks_completed | rank_position
----+-----+-----+-----+-----+-----
  1 | iIyaos3 | 1 | 0 | 0 | Новачок
  2 | bidenkoknt211 | 2 | 0 | 0 | Новачок
(2 rows)
```

Рисунок Б.8 – Слайд №8

ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

№	Крок	Очікуваний результат	Фактичний результат	Статус
1	Зареєструвати нового користувача та пройти авторизацію	Користувач буде зареєстрований та успішно пройде авторизацію	Дані нового користувача, вказані при реєстрації, було успішно збережено у БД	Успішно виконано
2	Спробувати змінити тип задач	Список задач змінюється	Список задач успішно змінюється відповідно до його типу	Успішно виконано
3	Оновити задачу у застосунку	Дані змінюються	Отримано підтвердження (Задача успішно оновлена на сервері)	Успішно виконано
3	Перевірити, чи відображено новий статус без reload	Статус задачі змінено на «Виконую»	Змінився одразу після відповіді сервера	Успішно виконано
4	Оновити сторінку	Дані залишаються після перезавантаження	Дані збереглися, відвантажено з БД	Успішно виконано

```

2 Задачі збережені у БД
Задача успішно оновлена на сервері: ▶ Object App.js:184
Задача успішно оновлена на сервері: ▶ Object App.js:161
Задача успішно оновлена на сервері: ▶ Object App.js:161
Задача успішно оновлена на сервері: ▶ Object App.js:161
Задача успішно оновлена на сервері: ▶ Object App.js:161

```

< > ↺ VPN 🌐 localhost:5000

Сервер працює!

Рисунок Б.9 – Слайд №9

ВИСНОВКИ

Під час розробки дипломного проекту було створено web-застосунок для активної мотивації та управління командою розробки IT-стартапу. В цьому проекті було використані наступні технології: бібліотека React на основі мови програмування JavaScript, програмна платформа Node.js та Express.js, реляційна база даних PostgreSQL та персональний комп'ютер з операційною системою Microsoft Windows 10 Pro та процесором Intel Core I5 3570-k.

Основною функцією розробленого web-застосунку є те, що, завдяки Kanban-дошці, процес став максимально інтуїтивно зрозумілим та не вимагаючим великої кількості часу, щоб була змога більш зконцентруватися на виконанні задач. Керівник команди розробки, в свою чергу, може переглядати хто зі співробітників перебуває на якому етапі та формувати таблицю лідерів з подальшими винагородами для ще більшої мотивації співробітників.

В результаті дослідження було проаналізовано особливості IT-стартапів та керування командою розробки, розглянуто методи керування співробітників у IT-стартапі, створено програмний web-застосунок для автоматизації методів мотивації та управління командою розробки IT-стартапу.

Рисунок Б.10 – Слайд №10