Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»
(повне найменування вищого навчального закладу)

Відділення інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян
(назва відділення)

Циклова комісія комп'ютерної інженерії
(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра
(освітньо-професійного ступеня)

на тему: **Розробка програми "Nexus" для студентської ради ВСП «ТФК ТНТУ»**

Виконав: студент IV курсу, групи КІ-406
Спеціальності 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

_____ Денис ЛАЗАР
(ім'я та прізвище)

Керівник Ігор КАПАЦІЛА
(ім'я та прізвище)

Рецензент _____
(ім'я та прізвище)

Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО
УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ»

Відділення **інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян**
Циклова комісія **комп'ютерної інженерії**
Освітньо-професійний ступінь **фаховий молодший бакалавр**
Освітньо-професійна програма: **Обслуговування комп'ютерних систем і мереж**
Спеціальність: **123 Комп'ютерна інженерія**
Галузь знань: **12 Інформаційні технології**

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"30" березня 2026 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лазару Денису Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:**Розробка програми "Nexus" для студентської ради ВСП "ТФК ТНТУ"**
керівник роботи Капаціла Ігор Богданович

затверджені наказом Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя» від 27.03.2026р № 4/9-167.
2. Строк подання студентом роботи: 15 червня 2026 року.
3. Вихідні дані до роботи: технічне завдання на розробку вебсайту, мова програмування високого рівня, база даних, стандарти IEEE 29148-2018, IEEE 29119
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Охорона праці та безпека життєдіяльності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
- Структурна схема проєкту.
- Структура компонент.
- Текст програми.
- Таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Богдана МАРТИНЮК викладач		
Охорона праці та безпека життєдіяльності	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Приміт ка
1	Отримання і аналіз технічного завдання	01.04	
2	Збір і узагальнення інформації	08.05	
3	Написання першого розділу	15.05	
4	Розробка технічного та робочого проекту	22.05	
5	Написання спеціального розділу	28.05	
6	Розрахунок економічної частини	1.06	
7	Написання розділу охорони праці	3.06	
8	Виконання графічної частини	8.06	
9	Оформлення проєкту	10.06	
10	Погодження нормоконтролю	11.06	
11	Попередній захист роботи	12.06	
12	Захист кваліфікаційної роботи		

7. Дата видачі завдання: 31 березня 2026 року

Студент _____ Денис ЛАЗАР
(підпис) (ім'я та прізвище)

Керівник роботи _____ Ігор КАПАЦІЛА
(підпис) ім'я та прізвище)

АНОТАЦІЯ

Розробка інформаційної системи автоматизації діяльності студентської ради «Nexus» // Кваліфікаційна робота // Лазар Денис Олександрович // Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя», група КІ-406 // Тернопіль, 2026 // с. – 68 , рис. – 20, табл. – 9 , кресл. – 4 , додат. – 10.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, NEXT.JS, MONGODB, TELEGRAM, СТУДЕНТСЬКЕ САМОВРЯДУВАННЯ, ГЕЙМІФІКАЦІЯ.

Мета роботи — розробка інформаційної системи «Nexus» для автоматизації організаційної діяльності студентської ради, яка об'єднує інструменти управління контентом, ієрархічну систему прав доступу, відстеження завдань та механізми гейміфікації, з інтеграцією через Telegram Mini App.

Пояснювальна записка складається з п'яти розділів.

В першому розділі виконано аналітичний огляд існуючих рішень (систем управління проєктами, систем управління знаннями, месенджерів, освітніх платформ) та обґрунтовано доцільність створення єдиної спеціалізованої платформи. Описано призначення розробки, вимоги до програмного та апаратного забезпечення, ведення документації та технічне завдання.

В другому розділі дається загальний опис задачі та її специфічні особливості. Описано методи побудови системи з використанням вибраного стеку технологій (Next.js, MongoDB) та інтеграції з месенджером Telegram. Описано розробку структури бази даних, процес та логіку програмування клієнтської і серверної частин проекту.

В наступному розділі описано процедури з розміщення системи в Інтернеті, її обслуговування та наповнення, а також інструкцію з популяризації та підтримки системи.

В двох наступних розділах розраховано вартість розроблюваної системи, а також описано техніку безпеки та екологічні вимоги (охорону праці та безпеку життєдіяльності) під час виконання даного проекту.

ABSTRACT

Development of the "Nexus" Information System for Student Council Activity Automation // Qualification Thesis // Lazar Denys Oleksandrovych // Separate Structural Unit "Ternopil Professional College of Ivan Puluj Ternopil National Technical University", Group KI-406 // Ternopil, 2026 // 68 p., 20 figs., 9 tabs., 4 drawings, 10 apps.

Keywords: INFORMATION SYSTEM, NEXT.JS, MONGODB, TELEGRAM, STUDENT GOVERNANCE, GAMIFICATION.

The aim of the thesis is to develop the "Nexus" information system designed to automate the organizational activities of the student council. The system consolidates content management tools, a hierarchical access control system, task tracking, and gamification mechanisms, featuring integration via a Telegram Mini App.

The explanatory note consists of five chapters.

The first chapter provides an analytical review of existing solutions (project management systems, knowledge management systems, messengers, and educational platforms) and substantiates the feasibility of creating a unified specialized platform. It outlines the purpose of the development, hardware and software requirements, documentation management, and the technical specification.

The second chapter presents a general description of the problem and its specific features. It details the system architecture methods using the selected technology stack (Next.js, MongoDB) and integration with the Telegram messenger. Furthermore, it describes the database schema design, alongside the development process and programming logic for both the client-side and server-side components of the project.

The third chapter details the system deployment procedures on the Internet, its maintenance, and content population, as well as guidelines for system promotion and support.

The subsequent two chapters present the cost estimation for the developed system and outline occupational health, safety engineering, and environmental requirements during the project execution.

ЗМІСТ

ВСТУП

Студентські роки — це період активного формування лідерських якостей, становлення громадянської позиції та згуртування спільнот, здатних ініціювати реальні зміни у своєму середовищі. Рушійною силою таких процесів традиційно виступає студентське самоврядування — об’єднання небайдужої молоді, готової брати на себе відповідальність за організацію життя навчального закладу.

Водночас практика показує, що координація роботи органів студентського самоврядування часто супроводжується суттєвою організаційною неузгодженістю. Спроби впровадити сторонні корпоративні рішення — зокрема загальні месенджери чи класичні таск-трекери — для розподілу доручень і відстеження їх виконання здебільшого виявляються невдалими. Основна причина полягає в тому, що подібні системи є незвичними для більшості студентів: високий поріг входження, потреба створювати нові облікові записи та опановувати складні інтерфейси призводять до того, що користувачі уникають роботи з ними, а самі процеси швидко перетворюються на хаотичне листування в соціальних мережах.

Саме потреба впорядкувати командну взаємодію та підвищити ефективність організаційної діяльності стала підставою для розробки інформаційної системи Nexus, яка й визначила тему цієї кваліфікаційної роботи.

Nexus — це не черговий додаток для управління завданнями, а консолідована інформаційна платформа, спроєктована з урахуванням особливостей студентського середовища. Щоб подолати бар’єр незнайомої системи, Nexus не змушує користувачів кардинально змінювати усталені звички, а пропонує глибоку інтеграцію з популярним серед молоді месенджером Telegram (через технологію Mini App), надаючи доступ до всіх інструментів безпосередньо там, де студенти вже спілкуються щодня.

Метою кваліфікаційної роботи є створення єдиного гнучкого середовища для автоматизації організаційної діяльності студентської ради. Система має об’єднати інструменти управління новинами та контентом, забезпечити чітку

ієрархію рівнів доступу, впровадити прозору систему відстеження завдань і мотиваційні механізми гейміфікації. Nexus покликаний перетворити неструктуровану рутину на впорядкований і захопливий процес, у якому кожен учасник відчуває власну значущість та має змогу легко взаємодіяти з командою за допомогою зручних і знайомих технологій.

Об’єктом дослідження є процеси організації та координації діяльності органів студентського самоврядування. Предметом дослідження є методи та програмні засоби побудови вебзастосунку для автоматизації цих процесів із застосуванням сучасного стеку технологій Next.js[13], MongoDB та інтеграції з месенджером Telegram[18].

Практичне значення роботи полягає в тому, що розроблену систему може бути впроваджено в реальну діяльність студентської ради навчального закладу для підвищення прозорості, керованості та залученості учасників до спільної роботи.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

На сьогодні не існує універсального стандарту організації роботи органів студентського самоврядування. Через це студентські ради змушені поєднувати кілька різнорідних сервісів, що ускладнює адміністрування та обмін інформацією. Наявні інструменти умовно поділяють на чотири групи: системи управління проєктами (Trello, Asana, Jira), системи управління знаннями (Notion, Confluence), месенджери (Telegram, Discord, Slack) та освітні платформи (Moodle, Google Classroom).

Системи управління проєктами ефективні для візуалізації завдань і контролю за ходом їх виконання. До їхніх переваг належать наочність етапів роботи, можливість фіксувати дедлайни та розподіляти відповідальність між виконавцями. Проте ці інструменти розроблялися для корпоративного сектору: вони не враховують внутрішню ієрархію студентських рад, не підтримують гейміфікацію активності та не дають змоги інтегрувати специфічну систему оцінювання за дванадцятибальною шкалою.

Системи управління знаннями використовуються для створення внутрішніх баз даних і публікації новин. Їхньою сильною стороною є функціональний редактор тексту та гнучкість у налаштуванні сторінок. Однак саме через високу гнучкість виникають труднощі з контролем доступу: під час зміни складу ради адміністрування прав стає громіздким, а автоматизація створення робочих чатів для виконання завдань не підтримується взагалі.

Месенджери залишаються основним засобом оперативної комунікації завдяки високій швидкості обміну даними та зручності інтерфейсу. Проте в них складно структурувати звіти й календар завдань: інформація швидко

накопичується, що ускладнює її систематизацію та аналіз ефективності роботи окремих учасників.

Освітні системи застосовуються переважно для академічної звітності, мають журнали успішності та прив’язку до академічних груп. Водночас вони орієнтовані на викладачів, мають надмірно формалізований інтерфейс і не адаптовані до потреб самоврядування — у них відсутні інструменти для подання ініціатив чи внутрішні соціальні хаби.

Проведений аналіз засвідчив, що використання розрізнених сервісів спричиняє фрагментацію даних: статистика розпорошується між платформами, а управління правами доступу стає неефективним. Це обґрунтовує доцільність створення єдиної спеціалізованої платформи, яка поєднує переваги перелічених рішень і водночас усуває їхні недоліки в контексті студентського самоврядування.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Повне найменування розробки — інформаційна система автоматизації діяльності студентської ради «Nexus». Скорочена назва — система Nexus.

Областю застосування системи є організаційна діяльність органів студентського самоврядування закладів фахової передвищої та вищої освіти. Система призначена для планування громадської роботи, контролю за виконанням доручень, ведення статистики активності студентів і підвищення рівня їхньої залученості.

На відміну від наявних рішень, Nexus об’єднує ключові функції в єдиному середовищі та забезпечує:

– адаптивну ієрархію — підтримку семи рівнів доступу, що відповідає структурі навчального закладу та студентської ради;

– інтеграцію з Telegram — використання Telegram Mini App як клієнта системи, що автоматизує створення чатів для завдань і спрощує збір звітів у базу даних;

– спеціалізоване планування — режими календаря та стеку завдань, що дають змогу адміністрації оцінювати як дедлайни, так і загальне навантаження на кожного студента;

– соціальну взаємодію — поєднання конструктора сторінок із новинним хабом та системою коментарів для підвищення залученості студентів.

1.2.2 Призначення розробки

Експлуатаційним призначенням системи є автоматизація та оптимізація діяльності студентської ради шляхом створення консолідованого інформаційного середовища, яке усуває фрагментацію робочих процесів і замінює кілька розрізнених сервісів єдиною інтегрованою платформою.

Функціональним призначенням системи є забезпечення планування громадської діяльності, контролю за виконанням доручень, ведення об’єктивної статистики активності студентів, а також підвищення рівня залученості молоді через механізми гейміфікації та інтеграцію з месенджерами.

Платформа Nexus включає такі підсистеми:

– підсистема кросплатформної авторизації — забезпечує вхід через Telegram Widget або безпосередньо через Telegram Mini App, об’єднуючи дані користувача в єдиний профіль;

– підсистема ієрархічного контролю доступу (RBAC) — підтримує багаторівневу модель управління правами (сім рівнів: від адміністратора до звичайного студента), що дає змогу гнучко розмежовувати доступ до функціоналу;

– підсистема управління контентом — містить візуальний редактор сторінок на базі Puck[15] та текстовий редактор, адаптований для створення публікацій із підтримкою згадувань користувачів;

– підсистема ефемерних робочих просторів — реалізує інтеграцію з Telegram для автоматичного створення робочих груп під конкретні завдання та їх видалення після завершення роботи;

– підсистема гейміфікації та дисциплінарного контролю — веде таблицю лідерів з автоматичним нарахуванням балів і здійснює моніторинг дисципліни (фіксація попереджень з автоматичним реагуванням після трьох порушень);

– підсистема комунікації та сповіщень — забезпечує розсилання повідомлень через вебінтерфейс і Telegram-бота, а також дозволяє коментувати ініціативи та подавати пропозиції до студентської ради.

1.2.3 Вимоги до програмного забезпечення

Вхідні дані. У системі обробляються різні категорії даних, що надходять від користувачів та інтегрованих сервісів. Перелік і опис вхідних повідомлень наведено у таблиці 1.1.

Таблиця 1.1 — Перелік і опис вхідних повідомлень

Назва вхідного повідомлення	**Форма подання**	**Періодичність надходження**	**Джерело даних**
Дані автентифікації та реєстрації	Вебформи (JSON), initData Telegram	Під час входу або реєстрації	Користувач, Telegram
Структурні макети сторінок	Дерево подібні JSON-структури (puckData)	Під час редагування та публікації	Редактор Puck (адміністратори)

Назва вхідного повідомлення	Форма подання	Періодичність надходження	Джерело даних
Медіафайли (аватари, обкладинки, відео)	Бінарні файли (multipart form data)	За потреби завантаження контенту	Користувачі та адміністратори
Звіти про виконання завдань	Текст, медіафайли	Залежно від дедлайнів	Користувачі (вебінтерфейс, бот)

Вихідна інформація. Результати розв'язання задач генеруються у вигляді вебкомпонентів, звітів або сповіщень. Перелік вихідних повідомлень подано у таблиці 1.2.

Таблиця 1.2 — Перелік і опис вихідних повідомлень

Назва вихідного повідомлення	Форма подання	Періодичність видання	Користувач інформації
Динамічні вебсторінки та новини	HTML / React-компоненти	На кожен запит клієнта	Усі користувачі
Аналітичні таблиці завдань	Графічний інтерфейс (календар, стек)	На запит адміністратора	Керівництво ради
Персональні профілі (DTO)	JSON-об'єкти без passwordHash	Під час перегляду профілів	Авторизовані користувачі
Системні сповіщення	Вебтости, повідомлення Telegram	У момент розсилання	Цільові групи студентів

Опис функцій та обмежень. Нижче наведено поетапний опис логіки ключових функцій системи із зазначенням вхідних даних, їх перетворення та результату.

– Функція кросплатформної авторизації та злиття профілів. Вхідними даними є облікові дані (логін, пароль) та маркери доступу сторонніх провайдерів. Система валідує дані за схемами Zod[20], виконує пошук користувача в MongoDB за унікальними ключами (login або telegramId) та зливає ідентифікатори з наявним записом або створює новий документ. Результат — згенерований JWT-токен сесії та публічний DTO-об’єкт користувача.

– Функція контролю доступу (RBAC). Вхідними даними є запит на зміну, індекс доступу актора та індекс цільового користувача. Виконується математичне порівняння ієрархії: операція дозволяється лише за умови суворого випередження та наявності делегованих прав. Результат — дозвіл на виконання операції або блокування з поверненням помилки доступу.

– Функція візуального конструювання сторінок. Вхідними даними є дії користувача в редакторі Puck та шлях URL. На клієнті формується JSON-дерево, а під час публікації сервер перевіряє унікальність URL. Результат — збережений документ Page у MongoDB. У разі збігу URL система повертає статус 409 Conflict без перезапису наявної сторінки.

– Функція розрахунку успішності. Вхідними даними є зараховані бали за виконані завдання. Виконується цілочислове підсумовування зароблених балів за весь період навчання, що унеможливлює похибки округлення. Результат — підсумковий рейтинг для виведення в таблицю лідерів.

Часові обмеження системи адаптовано під формат односторінкового застосунку (SPA): відгук клієнтського інтерфейсу на дії користувача — до 1 секунди; під час візуального редагування застосовується затримка (debouncing) у 400 мс перед оновленням макета; опитування сервера для отримання сповіщень відбувається кожні 60 секунд.

Для забезпечення надійного функціонування реалізовано ієрархічну структуру та ізоляцію (розгортання в контейнерах Docker[9]), жорстку валідацію

вводу на базі бібліотеки Zod[20], обробку виняткових ситуацій (перевірка конфліктів, статуси «не знайдено» та 409 Conflict), фоновий алгоритм очищення медіафайлів без зв'язків у базі даних, а також серіалізацію помилок у вигляді інформаційних повідомлень без виведення системного коду користувачеві.

Умови експлуатації. Взаємодія здійснюється за клієнт-серверною архітектурою через вебпереглядач або інтегрований Telegram Mini App[18]. Інтерфейс адаптовано для перегляду на моніторах і мобільних пристроях. Серверна частина вимагає ОС Linux (Ubuntu/Debian) із встановленим середовищем Docker; клієнтська частина сумісна із сучасними переглядачами з підтримкою CSS-змінних, а серверна базується на середовищі Node.js та СУБД MongoDB.

1.2.4 Вимоги до програмної документації

Документація системи Nexus є невіддільною частиною процесу розробки. Кожен клас, метод чи функція у вихідному коді документуються за стандартом JSDoc англійською мовою; описи мають бути стислими й розкривати не лише типи даних, але й побічні ефекти для бази даних. Архітектурні рішення, структури баз даних і специфікації функцій зберігаються безпосередньо в репозиторії у каталозі .ai/docs/ та оновлюються одночасно зі зміною коду, що виключає їх застарівання.

1.2.5 Техніко-економічні показники

Для реалізації системи Nexus виділено такий обсяг ресурсів. Розробка програмного комплексу здійснюється одним інженером-програмістом; загальний обсяг розрахункових трудовитрат на проєктування архітектури, кодування та тестування становить один людино-місяць. Для написання коду, локального розгортання контейнерів Docker, компіляції проєкту на Next.js та автоматизованого тестування витрачено близько 170 годин машинного часу.

Витрати на ліцензії відсутні, оскільки використовуються технології з відкритим кодом (MongoDB, Puck, Next.js під ліцензією MIT).

1.2.6 Стадії та етапи розробки

Розробку програмного комплексу виконано ітеративним методом із поділом на такі етапи:

1) налаштування монорепозиторію Next.js, контейнеризація (Docker), підключення MongoDB;

2) інтеграція візуального рушія Puck, налаштування глобального макета, динамічного фону та системи медіасховища;

3) розробка підсистеми кросплатформної авторизації (зокрема Telegram Mini App) та об’єднання профілів;

4) проєктування семирівневої моделі RBAC, створення таблиць відстеження завдань і системи гейміфікації;

5) програмування Telegram-бота для генерації робочих груп і парсингу звітів;

6) розробка текстового редактора та системи коментарів, фінальне тестування й розгортання.

1.2.7 Порядок контролю та прийому

Контроль працездатності програмного комплексу Nexus здійснюється у спеціально підготовленому Docker-середовищі, еквівалентному реальному серверу. Обсяг контрольного прикладу охоплює проходження повного користувацького сценарію: авторизація студента через Telegram Mini App, автоматичне створення профілю, візуальне конструювання новини адміністратором у редакторі Puck, делегування завдання та перевірка нарахування

балів. Надійність коду підтверджується запуском набору автоматизованих тестів, які охоплюють перевірку неможливості перевищення повноважень у семирівневій ієрархії, валідацію криптографічних підписів Telegram та коректність алгоритмів переміщення блоків у редакторі.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Розробка структури системи та вебсторінок

Створення сучасного вебпродукту починається з побудови його інформаційної моделі. Структура системи — це схема розташування сторінок, розділів і функціональних модулів, що відображає логічні зв’язки між ними. З технічної точки зору навігація ресурсу являє собою набір маршрутів (URL), розташованих у певній послідовності й об’єднаних спільною логікою доступу.

Архітектурно система Nexus побудована як монолітний застосунок на базі фреймворку Next.js[13], що поєднує клієнтську та серверну частини в межах єдиного монорепозиторію. Такий підхід дає змогу використовувати спільні типи даних і компоненти, спрощує розгортання та зменшує накладні витрати на підтримку окремих сервісів. Маршрутизація реалізована за допомогою App Router, що організовує сторінки у файловій ієрархії та підтримує як серверний (SSR), так і клієнтський (CSR) рендеринг.

Функціонально систему поділено на п’ять взаємопов’язаних доменів, кожен з яких відповідає за окрему предметну область: AuthDomain (профіль та автентифікація), TaskDomain (завдання), PageDomain (сторінки та новини), AccessControlDomain (адміністрування та контроль доступу) та спільна інфраструктура (MediaDomain, база даних, сесії). Центральною сутністю, навколо якої будуються всі зв’язки, є колекція користувачів users. Загальну схему зв’язків між модулями системи наведено на рисунку 2.1.

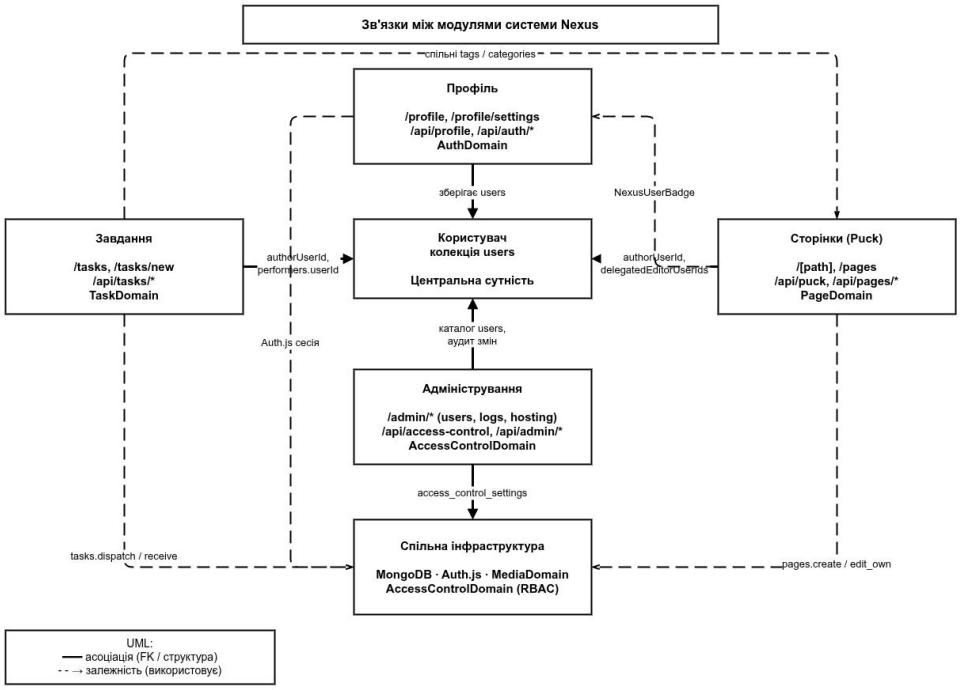

Рисунок 2.1 — Схема зв’язків між модулями системи Nexus

Як видно зі схеми, домен профілю збирає та зберігає записи користувачів, домен завдань посилається на користувачів через поля authorUserId та performers.userId, домен сторінок — через authorUserId і delegatedEditorUserIds, а домен адміністрування веде каталог користувачів і аудит змін. Усі домени спираються на спільну інфраструктуру, яка реалізує підключення до MongoDB, керування сесіями та централізований контроль доступу (RBAC).

Навігаційна структура системи передбачає чіткий поділ на загальнодоступну, авторизовану та адміністративну зони. Неавторизований відвідувач має доступ до публічних сторінок, створених у конструкторі, та до форм входу й реєстрації. Після авторизації користувач отримує доступ до особистого профілю, директорії учасників, стрічки завдань і новинного хабу. Адміністративна зона, ізольована захищеними маршрутами, доступна лише користувачам із достатнім рівнем ієрархії та містить інструменти керування користувачами, контролю доступу, перегляду журналів і налаштування хостингу.

2.2 Створення та верстка сторінок системи

Верстка — це перетворення макета інтерфейсу на робочий застосунок за допомогою програмного коду. Класичний підхід із використанням окремих файлів HTML і CSS погано масштабується для великих проєктів, тому для побудови клієнтської частини застосовано компонентний підхід на основі бібліотеки React[16] у поєднанні з утилітарним фреймворком Tailwind CSS[17]. Атомарні класи Tailwind, вбудовані безпосередньо в React-компоненти, роблять стилізацію модульною та запобігають конфліктам стилів, а вбудована адаптивність гарантує коректне відображення на екранах будь-якого розміру.

Особливістю системи Nexus є наявність двох клієнтських середовищ: повноцінного вебінтерфейсу для роботи у вебпереглядачі та полегшеного інтерфейсу Telegram Mini App, що відкривається безпосередньо у вікні месенджера. Спільні компоненти й логіка винесено в окремі модулі, що дає змогу повторно використовувати їх в обох середовищах і підтримувати єдиний стиль оформлення.

Базовий каркас застосунку формують глобальний макет (global_layout), що складається з адаптивної навігаційної панелі (header) із меню користувача та нижнього колонтитула (footer) із посиланнями та реквізитами. Глобальний макет зберігається у базі даних як окремий документ-сінглтон, що дає адміністраторам змогу змінювати навігацію та контактні дані без втручання в програмний код. На рисунку 2.2 зображено адмін панель зміни глобального макету.

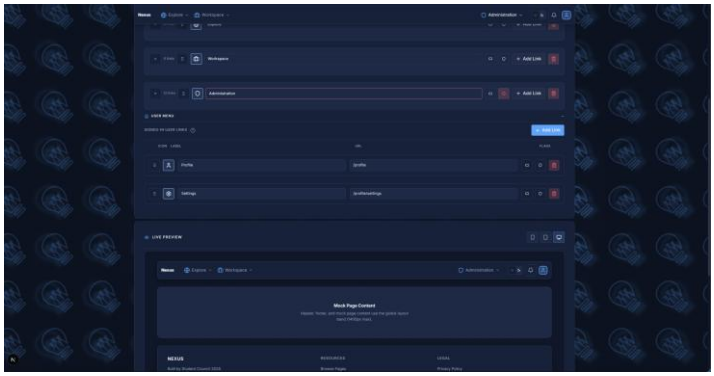

Рисунок 2.2 — панель керування вигляду навігації

Ключовою інновацією інтерфейсу є візуальний конструктор сторінок на базі рушія Puck[15]. Він дозволяє адміністраторам формувати публічні сторінки методом перетягування блоків (drag-and-drop) без знання програмування. Бібліотека Puck оперує деревоподібною JSON-структурою, у якій кожен блок описується типом і набором властивостей. Система Nexus розширює стандартний набір блоків власними компонентами: NexusSection («Секція-контейнер») (див. рис. 2.3) , NexusText («Форматований текст») (див. рис. 2.4) та NexusCarousel («Слайдер») (див. рис 2.5) .

Рисунок 2.3 — компонент «Секція-контейнер»

Рисунок 2.4 — компонент «Форматований текст»

Рисунок 2.5 — компонент «Слайдер»

Для ініціалізації проєкту використано пакетний менеджер npm та інструментарій Next.js. Розгортання базового каркасу виконується командою:

npx create-next-app@latest nexus --typescript --tailwind —app

Запуск середовища розробки з гарячим перезавантаженням модулів здійснюється командою npm run dev, а складання продакшн-версії — командою npm run build. Така організація забезпечує швидку роботу клієнтської частини як під час розробки, так і після фінальної компіляції для розміщення на хостингу.

2.3 Розробка структури бази даних

База даних — це структурована сукупність даних[23], що зберігається в електронному форматі та доступна для операцій обробки, пошуку й аналізу. Для системи Nexus, яка оперує різнорідними та слабко структурованими даними (профілі з безліччю необов’язкових полів, деревоподібні макети сторінок, вкладені масиви виконавців завдань), застосування суворої реляційної моделі призвело б до розгалуженої структури з великою кількістю таблиць і надмірного використання операцій об’єднання.

Таблиця 2.1 — Порівняльна характеристика реляційної СУБД та MongoDB

Критерій порівняння	**Реляційна СУБД (SQL)**	**MongoDB (NoSQL)**
1	**2**	**3**
Схема даних	Фіксована	Гнучка

Критерій порівняння	Реляційна СУБД (SQL)	MongoDB (NoSQL)
1	2	3
Модель даних	Таблиці та зв’язки	Документи (BSON)
Масштабування	Переважно вертикальне	Горизонтальне (шардинг)
Вкладені структури	Через окремі таблиці	Безпосередньо в документі
Мова доступу	SQL	Запити у форматі документів

Тому для зберігання даних обрано документоорієнтовану NoSQL базу даних MongoDB. Ця система працює з документами у форматі BSON (Binary JSON), що дозволяє зберігати масиви та вкладені структури безпосередньо в одному об’єкті. Порівняльну характеристику реляційної СУБД та MongoDB наведено в таблиці 2.1.

Для роботи з MongoDB з боку Node.js застосовано бібліотеку об’єктно-документного моделювання Mongoose[12], що дає змогу описувати схеми колекцій, валідувати дані та виконувати запити в типобезпечний спосіб. Хостинг бази даних реалізовано на офіційній хмарній платформі MongoDB Atlas[11], що забезпечує реплікацію, автоматичне резервне копіювання та моніторинг без потреби самостійно адмініструвати сервер бази даних.

Завдяки денормалізації структуру даних оптимізовано до шести базових колекцій: users (користувачі), tasks (завдання), pages (сторінки та новини), page_likes (вподобання сторінок), access_control_settings (налаштування контролю доступу) та user_directory_audits (аудит каталогу користувачів). Загальну схему

зв’язків між колекціями подано на рисунку 2.6 у вигляді діаграми «сутність — зв’язок».

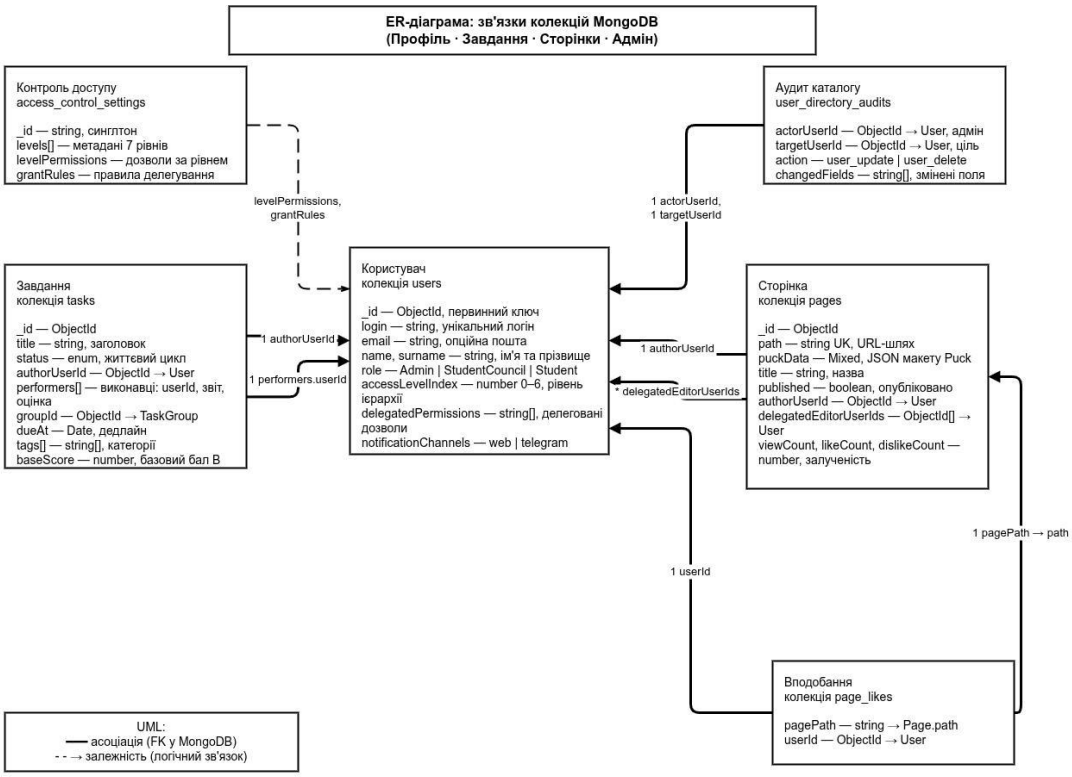

Рисунок 2.6 — ER-діаграма зв’язків колекцій бази даних

Центральною колекцією є users, на яку посилаються всі інші сутності. Колекція tasks містить поля authorUserId (автор) і масив performers (виконавці з полями userId, оцінка та звіт). Колекція pages посилається на автора (authorUserId) та делегованих редакторів (delegatedEditorUserIds), а колекція page_likes реалізує зв’язок «багато до багатьох» між сторінками й користувачами з унікальним складеним індексом. Колекція user_directory_audits фіксує дії адміністраторів над користувачами для подальшого аудиту.

Детальну структуру центральної колекції users наведено на рисунку 2.7. Документ користувача містить ідентифікаційні поля (login, email, телефон, ім’я, прізвище), зовнішні ідентифікатори провайдерів (googleId, appleId, telegramId), поле ролі (Admin, StudentCouncil, Student), числовий індекс рівня ієрархії

(accessLevelIndex), масив делегованих дозволів, а також поля профілю самоврядування, гейміфікації (зірки, попередження) та налаштувань сповіщень.

Повний лістинг моделі колекції користувачів засобами Mongoose наведено в додатку А.

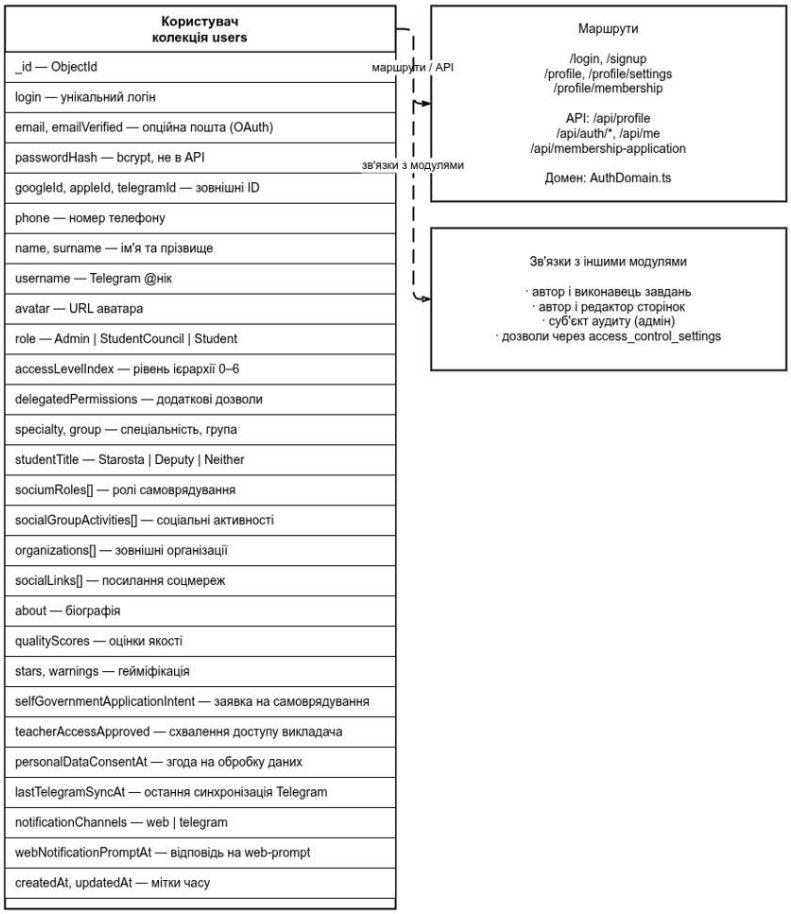

Рисунок 2.7 — Структура колекції користувачів users

Поле passwordHash позначено опцією select: false, тому воно ніколи не потрапляє у відповіді API за замовчуванням, що підвищує безпеку. Унікальні розріджені (sparse) індекси на полях telegramId та email дають змогу зберігати профілі без цих значень і водночас гарантувати унікальність за наявності.

2.4 Програмування системи

Програмування системи Nexus виконано [19], що додає до JavaScript статичну типізацію та підвищує надійність коду під час розробки. Завдяки монолітній архітектурі на Next.js клієнтська й серверна логіка перебувають у спільному репозиторії, проте чітко розмежовані: інтерфейсні компоненти виконуються на боці клієнта, а доменна логіка та робота з базою даних — на сервері через маршрути API[2].

2.4.1 Написання клієнтської частини

Клієнтську частину побудовано на основі компонентів React. Інтерфейс декомпозовано на ієрархію незалежних компонентів: глобальний макет , навігаційна панель (система дозволяє змініювати посилання у хедері та футері сайту через графічний інтерфейс), картки користувачів і завдань, форми введення даних та модальні вікна. Для керування станом застосунку та кешування серверних даних використано бібліотеку запитів, що автоматично оновлює інтерфейс під час зміни даних на сервері.

Взаємодія з сервером організована через асинхронні HTTP-запити. Клієнт через налаштований екземпляр HTTP-клієнта безпечно передає JWT-токени, динамічно отримує дані з бази та ініціює операції створення й оновлення сутностей. Для валідації форм на боці клієнта застосовано бібліотеку схем Zod[20], що дає змогу описати правила перевірки один раз і використати їх як на клієнті, так і на сервері.

Окремої уваги заслуговує реалізація візуального конструктора сторінок. Компонент редактора Puck отримує конфігурацію доступних блоків і поточний стан макета у вигляді JSON, відображає область редагування з можливістю перетягування блоків та панель налаштування їхніх властивостей. Під час кожної зміни макет зберігається у стані компонента із затримкою (debouncing) 400 мс, що

запобігає надмірному навантаженню. Повну конфігурацію блоків редактора Puck наведено в додатку З.

2.4.2 Написання серверної частини

Серверна частина реалізована як набір маршрутів API у межах Next.js. Кожен маршрут відповідає за окрему операцію та згрупований за доменами: /api/auth/* (автентифікація), /api/tasks/* (завдання), /api/pages/* і /api/puck (сторінки), /api/admin/* та /api/access-control (адміністрування). Доменна логіка винесена в окремі модулі (AuthDomain, TaskDomain, PageDomain, AccessControlDomain), що відокремлює обробку HTTP-запитів від бізнес-правил. Структура директорі /api/ зображена на рисунку 2.8.

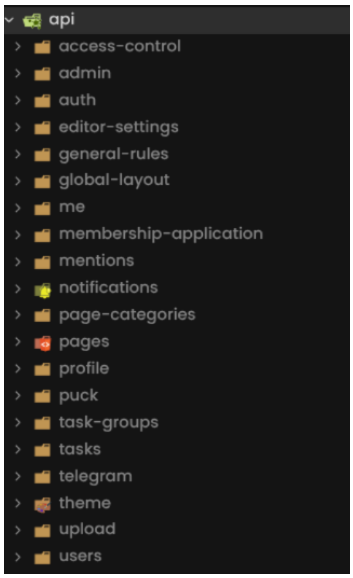

Рисунок 2.8 — Структура набору маршрутів

Структуру колекції завдань tasks та її життєвий цикл подано на рисунку 2.9. Документ завдання містить заголовок, опис (санітизований HTML), статус, масив виконавців, базовий бал, дедлайн, теги та зв’язок із групою завдань. Завдання

проходить життєвий цикл від чернетки (draft) через надсилання (dispatched), підтвердження (acknowledged), виконання (in_progress) і подання звіту (submitted) до завершення (completed); окремо передбачено статуси прострочення (overdue) та скасування (cancelled).

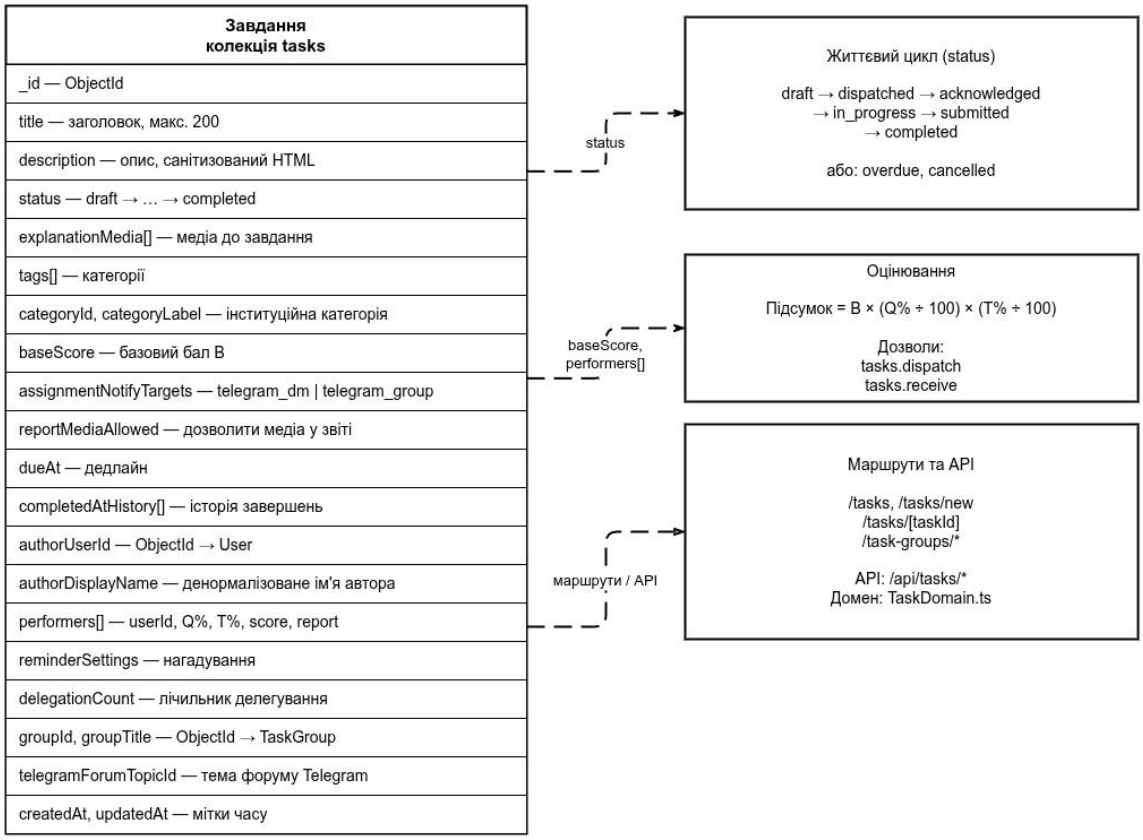

Рисунок 2.9 — Структура колекції завдань tasks та життєвий цикл

Підсумковий бал за виконане завдання обчислюється на основі базового балу та двох коефіцієнтів — якості виконання та своєчасності — за формулою

$$P = B \cdot (Q / 100) \cdot (T / 100), \qquad (2.1)$$

де P — підсумок; B — базовий бал завдання; Q — відсоток якості виконання (0…100); T — відсоток своєчасності, що враховує дотримання дедлайну.

Обчислення виконується виключно в цілочисловій арифметиці, що унеможливлює накопичення похибок округлення під час підсумовування балів за весь період навчання. Доступ до операцій диспетчеризації та приймання завдань регулюється ключами дозволів tasks.dispatch і tasks.receive. Лістинг функцій обчислення підсумкового балу та формування таблиці лідерів наведено в додатку Е.

Ключовою особливістю серверної логіки завдань є інтеграція з Telegram. Під час диспетчеризації завдання серверний модуль через Telegram-бота автоматично створює тему форуму (forum topic) у відповідній робочій групі, а ідентифікатор теми зберігається в полі telegramForumTopicId. Звіти, надіслані виконавцями в цю тему, бот парсить і зберігає у відповідні поля документа завдання. Після завершення роботи ефемерний робочий простір може бути автоматично прибрано.

Структуру колекцій сторінок pages і вподобань page_likes наведено на рисунку 2.11. Документ сторінки містить унікальний шлях path, макет puckData, заголовок, ознаку публікації, теги-категорії, зображення та лічильники залученості (перегляди, вподобання). Зв’язок зі сторінками вподобань реалізує реакції користувачів. Повний лістинг моделі сторінки наведено в додатку В. Лістинги моделей вподобань, аудиту каталогу та налаштувань контролю доступу наведено в додатку Г.

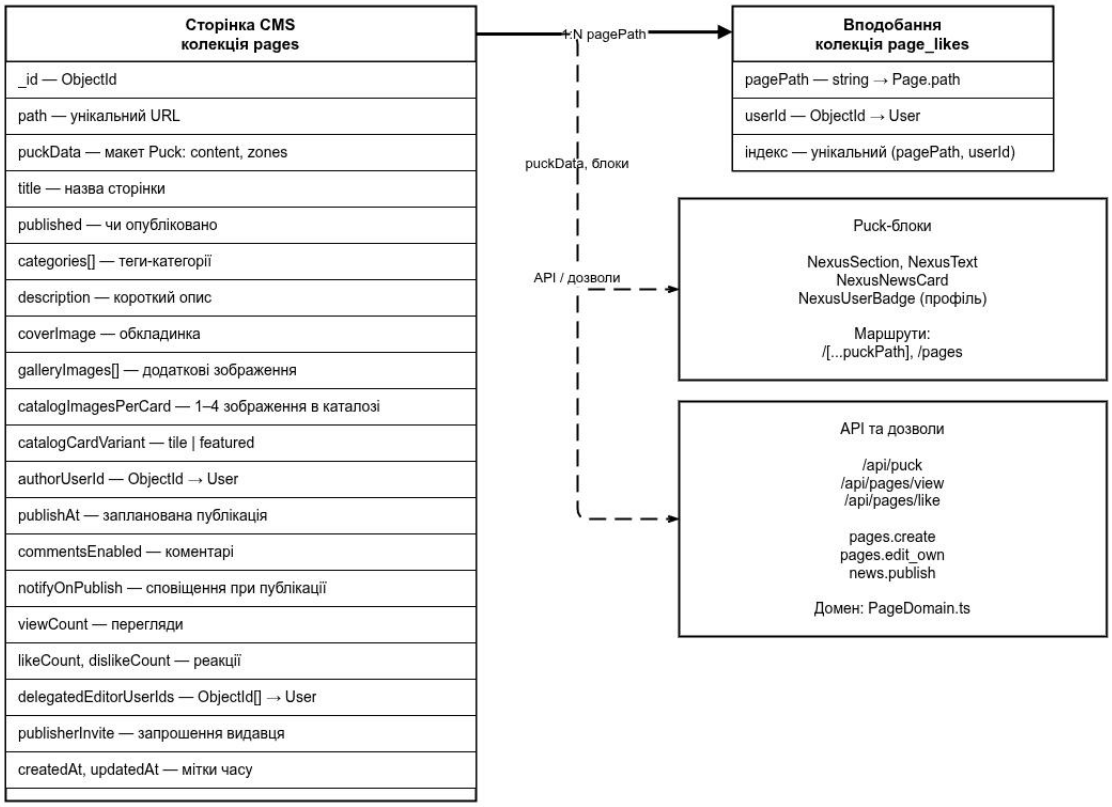

Рисунок 2.11 — Структура колекцій сторінок pages та вподобань page_likes

Під час публікації сторінки сервер перевіряє унікальність шляху path; повний лістинг обробника наведено в додатку Ж. Спрощений приклад має такий вигляд:

```
if (previousPath !== normalizedPath) {
  const conflict = await Page.findOne({ path: normalizedPath }).lean();
  if (conflict) {
    throw new PageDomainError(`The path "${normalizedPath}" is already taken.`, 409);
  }
}
```

Підсистему контролю доступу та адміністрування подано на рисунку 2.12. Налаштування зберігаються в колекції-сінглтоні access_control_settings, що містить метадані семи рівнів ієрархії, дозволи за рівнями (levelPermissions) та правила делегування (grantRules). Хаб адміністрування /admin об’єднує глобальний макет, каталог сторінок, керування доступом, журнали системи, каталог користувачів і заявки на членство.

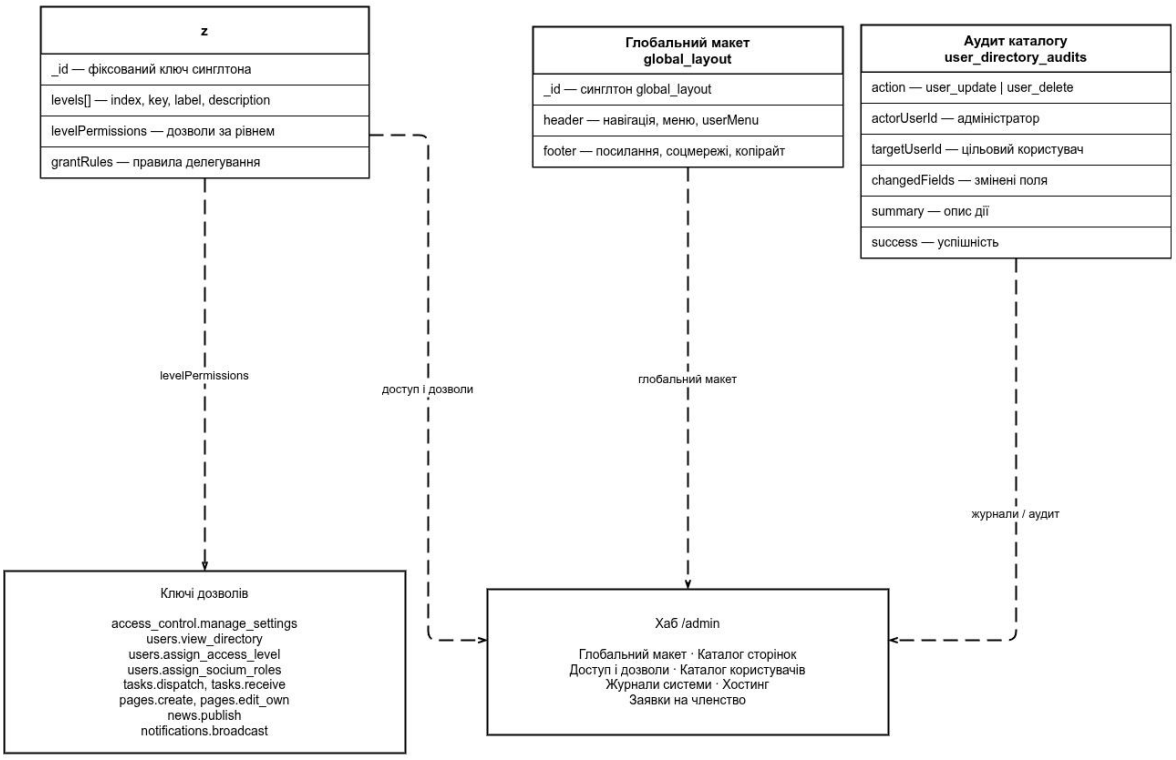

Рисунок 2.12 — Структура підсистеми адміністрування та контролю доступу

В основі контролю доступу лежить набір іменованих ключів дозволів (access_control.manage_settings, users.view_directory, users.assign_access_level, tasks.dispatch, pages.create, news.publish, notifications.broadcast тощо), які зіставляються з рівнями ієрархії. Перевірка прав виконується централізовано: операція над користувачем дозволяється лише за умови, що рівень актора суворо випереджає рівень цільового користувача та що актору надано відповідний ключ дозволу. (Лістинг функції перевірки прав доступу наведено в додатку Д.) Усі зміни над користувачами фіксуються в колекції аудиту user_directory_audits із зазначенням актора, цілі, змінених полів і результату операції.

2.5 Тестування системи

Тестування — це процес виконання програми з метою виявлення помилок та підтвердження відповідності її поведінки вимогам. Для системи Nexus застосовано поєднання модульного, інтеграційного та функціонального тестування, що дає змогу контролювати як окремі функції, так і взаємодію компонентів у цілому.

Модульні тести перевіряють коректність ізольованих функцій — насамперед обчислення підсумкового балу за формулою (2.1) та логіки порівняння рівнів ієрархії. Інтеграційні тести охоплюють взаємодію серверних маршрутів із базою даних: створення й оновлення сутностей, перевірку унікальності шляхів сторінок і повернення коректних статусів помилок. Окрему групу становлять тести безпеки, які підтверджують неможливість перевищення повноважень у семирівневій ієрархії та коректність перевірки криптографічних підписів даних Telegram.

Функціональне тестування виконано вручну за методом «чорної скриньки»[24] шляхом проходження повного користувацького сценарію: реєстрація та авторизація через Telegram Mini App, автоматичне створення профілю, конструювання новини в редакторі Puck, делегування завдання виконавцеві, подання звіту через бота та перевірка нарахування балів у таблиці лідерів. Результати тестування підтвердили відповідність системи вимогам технічного завдання та готовність до розгортання.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з розміщення системи в Інтернеті

На цьому етапі вже наявний протестований програмний комплекс Nexus, реалізований як монолітний застосунок на базі Next.js із базою даних MongoDB. Для публічного розміщення системи в мережі Інтернет обрано хмарну інфраструктуру Amazon Web Services (AWS)[7], яка забезпечує гнучке масштабування, високу доступність і широкий набір керованих сервісів. Базу даних розгорнуто на офіційній хмарній платформі MongoDB Atlas[11], що позбавляє від потреби самостійно адмініструвати сервер бази даних.

Загальна схема розгортання передбачає такі складники: обчислювальний інстанс Amazon EC2 для запуску застосунку в контейнері Docker, об’єктне сховище Amazon S3 для зберігання медіафайлів, мережу доставки контенту Amazon CloudFront для прискорення віддачі статичних ресурсів, а також зовнішній кластер MongoDB Atlas для зберігання даних. Перед розміщенням увесь вихідний код проєкту завантажується у віддалений репозиторій на платформі GitHub.

Послідовність розгортання системи така:

1) створити обліковий запис AWS та налаштувати інструмент керування доступом (IAM) із обмеженими правами для розгортання;

2) у консолі MongoDB Atlas створити кластер, налаштувати користувача бази даних і додати IP-адресу інстанса EC2 до списку дозволених підключень, після чого отримати рядок підключення (connection string);

3) у сервісі Amazon EC2 запустити інстанс на базі ОС Ubuntu Server, обравши тип інстанса t3.medium (2 віртуальні процесори, 4 ГБ оперативної пам’яті) як достатній для очікуваного навантаження;

4) підключитися до інстанса за протоколом SSH та встановити середовище Docker і Docker Compose[9];

5) створити в Amazon S3 сегмент (bucket) для медіафайлів і налаштувати політику доступу, а в CloudFront — дистрибуцію, що використовує цей сегмент як джерело;

6) клонувати репозиторій із GitHub та задати змінні середовища (рядок підключення до MongoDB Atlas, ключі доступу до S3, секрет JWT, токен Telegram-бота) у файлі оточення;

7) зібрати та запустити контейнери командою docker compose up.

Запуск застосунку в контейнерному середовищі виконується командою:

docker compose up -d —build

Для забезпечення захищеного з’єднання (HTTPS) перед застосунком встановлюється зворотний проксі-сервер, а сертифікат шифрування отримується автоматично через службу сертифікатів. Доменне ім’я спрямовується на публічну адресу інстанса EC2 за допомогою сервісу маршрутизації DNS. Після завершення цих кроків система стає доступною за обраним доменним іменем.

Така архітектура поєднує переваги контейнеризації (передбачуване середовище виконання, ізоляція залежностей) із керованими хмарними сервісами AWS і MongoDB Atlas. Винесення медіафайлів до S3 та їх роздавання через CloudFront знижує навантаження на обчислювальний інстанс і прискорює завантаження контенту для користувачів, а керована база даних Atlas забезпечує автоматичне резервне копіювання та реплікацію.

3.2 Інструкція з обслуговування та наповнення системи

Обслуговування системи Nexus спроєктовано так, щоб поточна робота з контентом, користувачами та завданнями не потребувала від адміністратора глибоких технічних знань. Уся взаємодія з наповненням платформи здійснюється через вебінтерфейс адміністративного хаба /admin, доступного користувачам із достатнім рівнем ієрархії.

Керування публічним контентом виконується через візуальний конструктор сторінок (див. рис. 3.1). Для створення або редагування сторінки адміністратор виконує такі дії:

1) у хабі адміністрування обирає розділ каталогу сторінок;

2) створює нову сторінку, задаючи її унікальний шлях (URL), заголовок і теги-категорії;

3) у візуальному редакторі Puck формує макет, перетягуючи потрібні блоки (секції, текст, картки новин, картки користувачів) та налаштовуючи їхні властивості;

4) за потреби завантажує зображення, які автоматично зберігаються в об’єктному сховищі S3;

5) публікує сторінку, після чого вона стає доступною за вказаним шляхом.

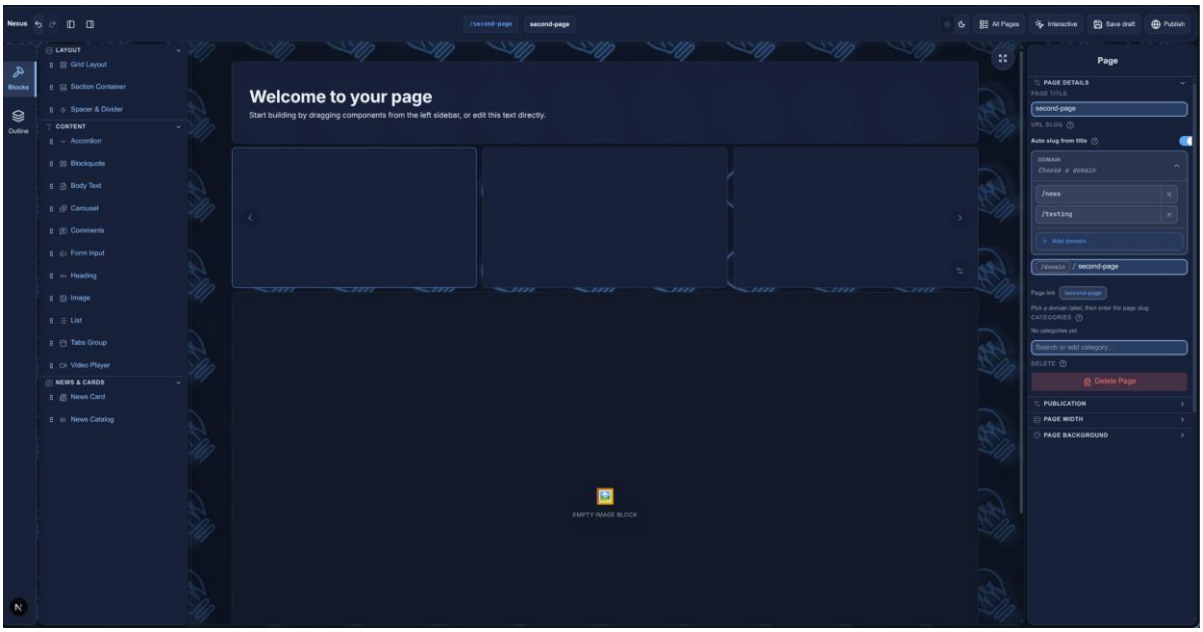

Рисунок 3.1 — Графічний редактор Puck

Керування користувачами здійснюється в каталозі учасників. (див. рис. 3.2) Адміністратор із відповідним рівнем доступу може переглядати профілі, призначати рівень ієрархії та ролі самоврядування, опрацьовувати заявки на членство, а також застосовувати дисциплінарні заходи (попередження). Усі дії над

користувачами фіксуються в журналі аудиту, що забезпечує прозорість і підзвітність адміністрування. (див. рис. 3.3)

Керування завданнями виконується у відповідному розділі системи: адміністратор створює завдання, задає базовий бал і дедлайн, призначає виконавців і диспетчеризує завдання (див. рис. 3.4). Після цього система автоматично створює робочий простір у Telegram, а надіслані виконавцями звіти збираються в базі даних. Оцінювання виконаних завдань відбувається шляхом задання коефіцієнтів якості та своєчасності, на основі яких система обчислює підсумковий бал за формулою (2.1) і оновлює таблицю лідерів.

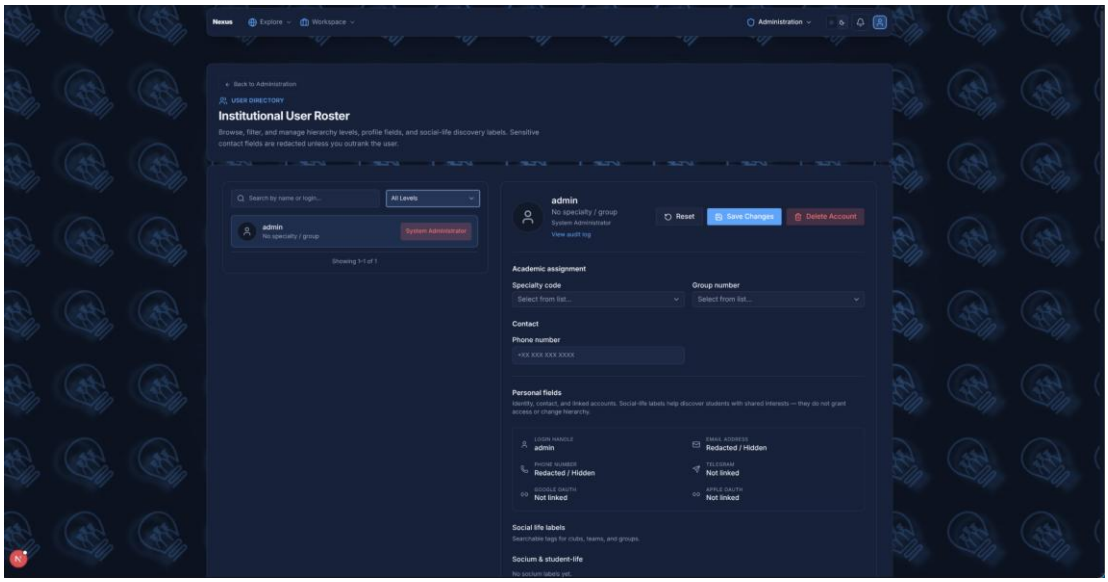

Рисунок 3.2 — Каталог користувачів

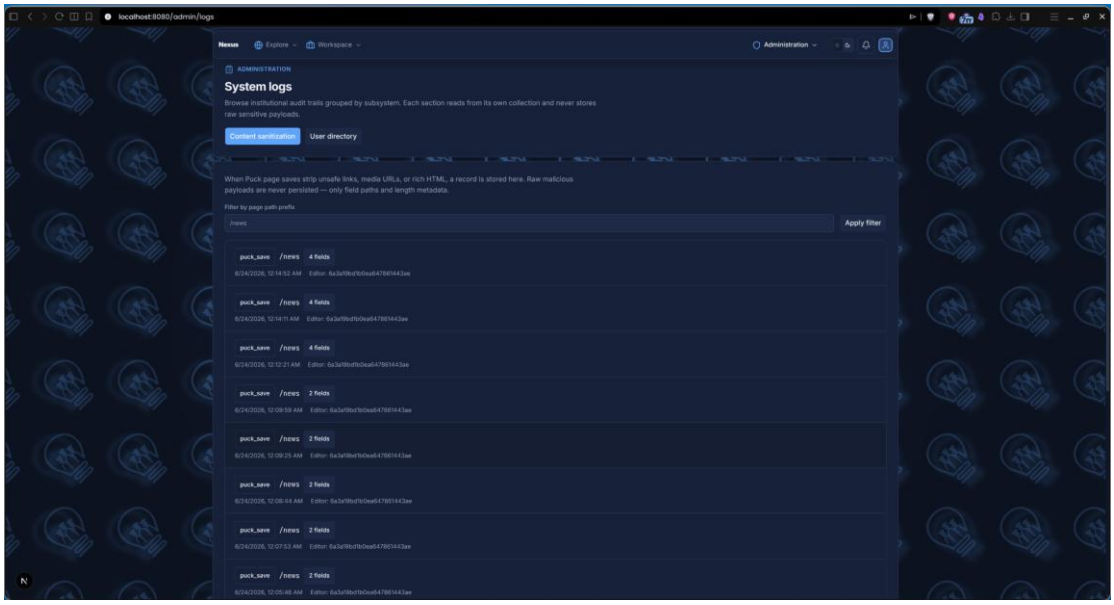

Рисунок 3.3 — Журнал аудиту

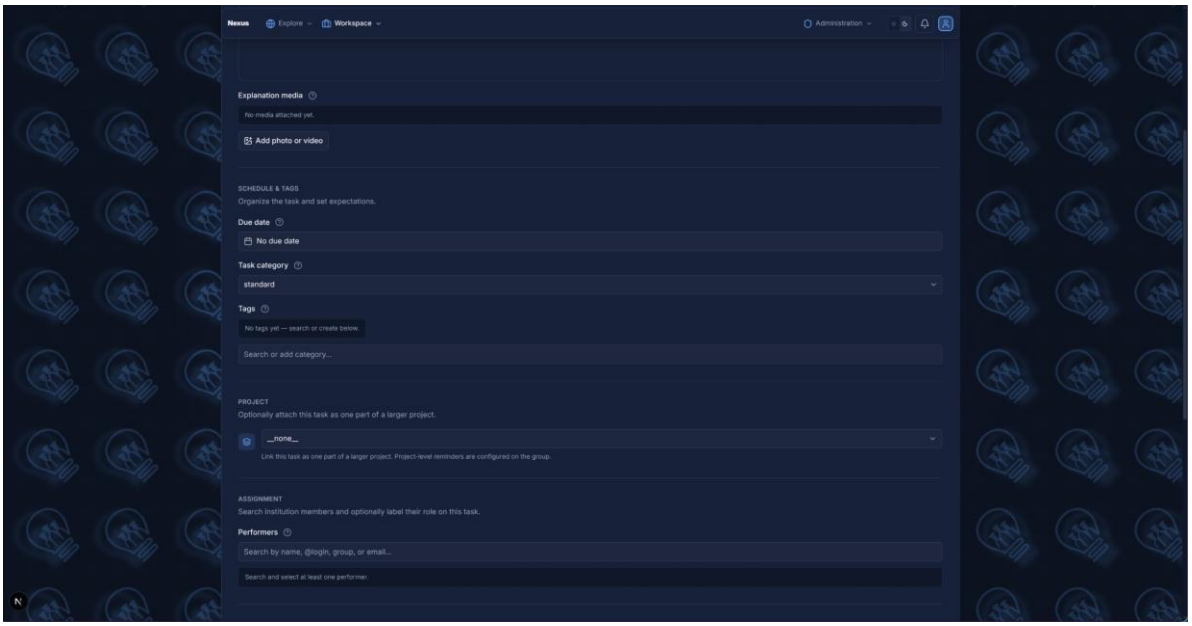

Рисунок 3.4 — Диспетчер завдань

3.3 Інструкція з популяризації та підтримки системи

Після розгортання системи в мережі перед адміністрацією постають завдання із залучення користувачів, підтримання активності та забезпечення технічної працездатності платформи.

Для популяризації системи серед студентів доцільно використовувати такі підходи:

– поширення посилання на Telegram Mini App у наявних студентських чатах і спільнотах, що завдяки знайомому середовищу месенджера знижує поріг входження нових користувачів;

– регулярне наповнення новинного хабу актуальними публікаціями про діяльність студентської ради, заходи та ініціативи, що формує звичку відвідувати платформу;

– залучення студентів до подання власних пропозицій та коментування ініціатив, що підвищує відчуття причетності до прийняття рішень. (див. рис. 3.6)

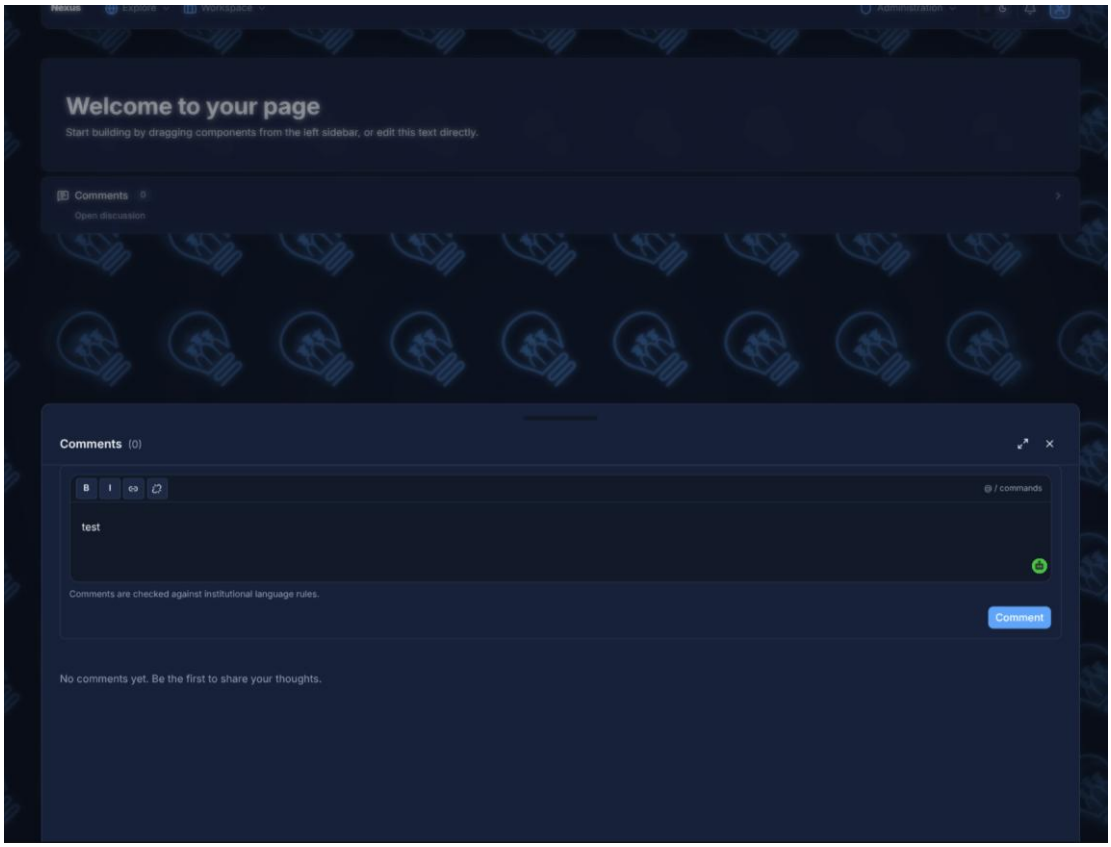

Рисунок 3.6 - Система обговорення новин

Для забезпечення безперервної роботи системи адміністратор має здійснювати регулярний технічний нагляд за ключовими вузлами інфраструктури:

– контроль стану обчислювального інстанса — періодична перевірка завантаженості процесора та пам’яті інстанса EC2 у консолі моніторингу AWS, а також перегляд журналів роботи контейнерів; (див. рис. 3.7)

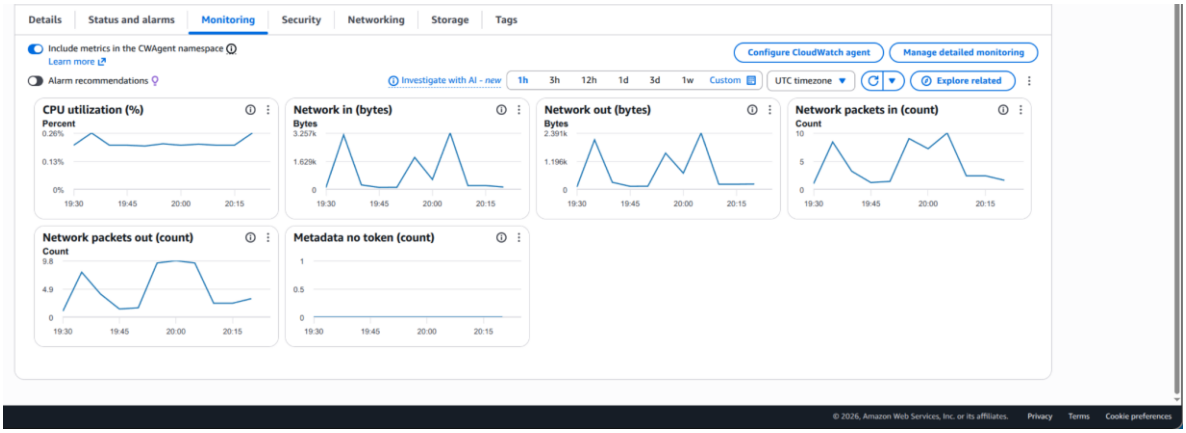

Рисунок 3.7 — Панель моніторингу EC2

– моніторинг бази даних — контроль рівня заповнення сховища та активності кластера MongoDB Atlas, перевірка налаштувань автоматичного резервного копіювання для запобігання втраті даних; (див. рис. 3.8)

– контроль медіасховища — спостереження за обсягом даних у сегменті S3 і роботою фонового алгоритму очищення медіафайлів, що не мають зв’язків у базі даних;

– оновлення системи — застосування оновлень коду шляхом завантаження стабільних змін до репозиторію та повторної збірки контейнерів, а також своєчасне оновлення секретних ключів і токенів доступу.

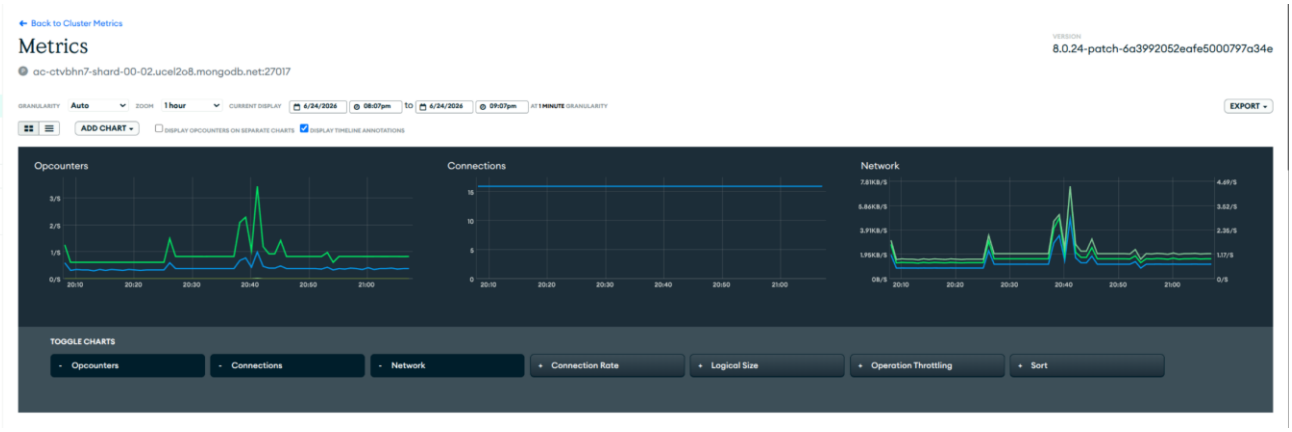

Рисунок 3.8 — Моніторинг роботи бази даних MongoDB

Дотримання наведених рекомендацій забезпечує стабільну роботу платформи, захист даних користувачів і поступове зростання залученості студентів до діяльності самоврядування.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічного розділу кваліфікаційної роботи є фінансово-економічне обґрунтування доцільності розробки інформаційної системи Nexus[5], визначення повної собівартості науково-дослідної роботи (НДР) та оцінювання економічної ефективності її впровадження. Проведені розрахунки є підставою для прийняття рішення щодо доцільності практичної реалізації програмного продукту.

Для визначення вартості НДР необхідно виконати такі етапи: описати технологічний процес розробки із зазначенням трудомісткості кожної операції; визначити витрати на оплату праці й відрахування на соціальні заходи; розрахувати матеріальні витрати, витрати на електроенергію та транспортні витрати; нарахувати амортизаційні відрахування та накладні витрати; скласти кошторис і визначити собівартість; розрахувати ціну НДР та оцінити економічну ефективність і термін окупності.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Витрати часу на виконання окремих технологічних операцій узагальнено й систематизовано у таблиці 4.1. Виконавцями стадій технологічного процесу є керівник проєкту, інженер-програміст та технік.

Таблиця 4.1 — Середній час виконання НДР та стадії технологічного процесу

№ з/п	Назва операції (стадії)	Виконавець	Середній час, год
1	2	3	4
1	Аналіз технічного завдання та збір вимог	Керівник проєкту	8
2	Проєктування архітектури та бази даних	Інженер-програміст	24
3	Розробка клієнтської частини	Інженер-програміст	56
4	Розробка серверної частини та інтеграцій	Інженер-програміст	60
5	Тестування системи	Технік	16
6	Розгортання та підготовка документації	Інженер-програміст	6
	Разом	—	170

Згідно з проведеними розрахунками, сумарний час, необхідний для виконання всіх етапів розробки системи Nexus, становить 170 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Відповідно до Закону України «Про оплату праці» заробітна плата — це винагорода, обчислена у грошовому виразі, яку за трудовим договором роботодавець виплачує працівникові за виконану роботу. Основна заробітна плата розраховується за формулою

$$Зосн = Тс \cdot Кг\ , \qquad (4.1)$$

де T_c — годинна тарифна ставка працівника, грн/год (приймаємо для керівника проєкту — 190 грн/год, інженера-програміста — 135 грн/год, техніка —

80 грн/год); $К_г$ — кількість відпрацьованих годин. Тарифні ставки визначено на основі аналізу актуальних вакансій IT-сектору за даними профільного порталу DOU[10].

$$Зосн = 190 \cdot 8 + 135 \cdot 146 + 80 \cdot 16 = 22510 \text{ грн.}$$

Додаткова заробітна плата охоплює доплати, надбавки та премії й приймається у межах 10–15 % від основної заробітної плати. Вона розраховується за формулою

$$Здод = Зосн \cdot Копл , \quad (4.2)$$

де $К_{опл}$ — коефіцієнт додаткових виплат працівникам (приймаємо 0,13).

$$Здод = 22510 \cdot 0,13 = 2926,30 \text{ грн.}$$

Загальні витрати на оплату праці визначаються за формулою

$$Во.п = Зосн + Здод , \quad (4.3)$$

$$Во.п = 22510 + 2926,30 = 25436,30 \text{ грн.}$$

Відрахування на соціальні заходи (єдиний соціальний внесок) становлять 22 % від фонду оплати праці та обчислюються за формулою

$$Вс.з = ФОП \cdot 0,22 , \quad (4.4)$$

де ФОП — фонд оплати праці, грн.

$$Вс.з = 25436,30 \cdot 0,22 = 5595,99 \text{ грн.}$$

Результати розрахунків зведено у таблицю 4.2.

Таблиця 4.2 — Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахування на ФОП, грн.	Всього витрати на оплату праці, грн.
		Тарифна ставка, грн.	К-сть відпрацьов. год.	Фактично нарах. з/пл., грн.			
1	Керівник проєкту	190	8	1520,00	197,60	–	–
2	Інженер-програміст	135	146	19710,00	2562,30	–	–
3	Технік	80	16	1280,00	166,40	–	–
Разом			170	22510,00	2926,30	5595,99	31032,29

Загальні витрати на оплату праці з урахуванням відрахувань на соціальні заходи становлять 31032,29 грн.

4.3 Розрахунок матеріальних витрат

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та ціни їх придбання за формулою

$$MBi = qi \cdot pi , \quad (4.5)$$

де q_i — кількість витраченого матеріалу і-го виду; p_i — ціна матеріалу і-го виду.

Загальна сума матеріальних витрат обчислюється за формулою

$$Зм.в = \Sigma MBi \, . \qquad (4.6)$$

Розрахунки наведено у таблиці 4.3.

Таблиця 4.3 — Зведені розрахунки матеріальних витрат

№	Найменування	Од.	К-сть	Ціна, грн	Сума, грн
1	Флешнакопичувач Kingston 16 ГБ	шт	1	400,00	400,00
2	Друк пояснювальної записки	арк	152	2,00	304,00
3	Друк плакатів графічної частини	шт	4	70,00	280,00
	Разом				984,00

Отже, загальна сума матеріальних витрат становить 984 грн.

4.4 Розрахунок витрат на електроенергію

Затрати на електроенергію визначаються за формулою

$$Зе = W \cdot T \cdot S\,, \qquad (4.7)$$

де W — необхідна потужність, кВт; Т — кількість годин роботи обладнання; S — вартість кіловат-години електроенергії.

Розробка системи здійснювалася з використанням одного ноутбука; номінальна потужність зарядного пристрою становить 0,15 кВт, час роботи — 170 год (згідно з табл. 4.1), вартість 1 кВт·год електроенергії — 15,94 грн.

$$Зе = 0{,}15 \cdot 170 \cdot 15{,}94 = 406{,}47 \text{ грн.}$$

4.5 Визначення транспортних затрат

Транспортні витрати прогнозуються у розмірі 8–10 % від загальної суми матеріальних затрат і розраховуються за формулою

$$Тв = Зм.в \cdot (0{,}08 \ldots 0{,}1)\,, \qquad (4.8)$$

де $T_в$ — транспортні витрати.

$$Тв = 984 \cdot 0{,}08 = 78{,}72 \text{ грн.}$$

4.6 Розрахунок суми амортизаційних відрахувань

Амортизація — це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення. Амортизаційні відрахування визначаються за формулою

$$A = БВ \cdot НА / 100\ \% \cdot Т\ , \qquad (4.9)$$

де А — амортизаційні відрахування, грн; БВ — балансова вартість обладнання, грн; НА — норма амортизації, %; Т — кількість годин роботи обладнання.

Розробка виконувалася на одному ноутбуці вартістю 32000 грн; загальний час експлуатації в межах проєкту — 170 год.

$$A = 32000 \cdot 0{,}04 \cdot 170 / 150 = 1450{,}67\ грн.$$

4.7 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням процесу розробки, утриманням обладнання та створенням умов праці. Залежно від організаційної форми вони становлять 20–60 % від суми основної та додаткової заробітної плати й обчислюються за формулою

$$Нв = Во.п \cdot (0{,}2 \ldots 0{,}6)\ , \qquad (4.10)$$

де $Н_в$ — накладні витрати.

$$Нв = 25436{,}30 \cdot 0{,}25 = 6359{,}07\ грн.$$

4.8 Складання кошторису витрат та визначення собівартості НДР

Результати проведених розрахунків зведено у таблицю 4.4.

Таблиця 4.4 — Кошторис витрат на НДР

Зміст витрат	Сума, грн	% до заг. суми
Витрати на оплату праці	25436,30	63,10
Відрахування на соціальні заходи	5595,99	13,88
Матеріальні витрати	984,00	2,44
Витрати на електроенергію	406,47	1,01
Транспортні витрати	78,72	0,20
Амортизаційні відрахування	1450,67	3,60
Накладні витрати	6359,07	15,77
Собівартість	40311,22	100,00

Собівартість НДР розраховується за формулою

$$Св = Во.п + Вс.з + Зм.в + Зе + Тв + А + Нв \quad (4.11)$$

$$Св = 25436,30 + 5595,99 + 984 + 406,47 + 78,72 + 1450,67 + 6359,07 = 40311,22 \text{ грн.}$$

4.9 Розрахунок ціни НДР

Ціна НДР визначається за формулою

$$Ц = Св \cdot (1 + Ррен) \cdot (1 + ПДВ) , \quad (4.12)$$

де $P_{рен}$ — рівень рентабельності (25 %); ПДВ — ставка податку на додану вартість (20 %).

$$Ц = 40311,22 \cdot (1 + 0,25) \cdot (1 + 0,2) = 60466,83 \text{ грн.}$$

4.10 Визначення економічної ефективності та терміну окупності

Ефективність розробки оцінюється шляхом розрахунку чистої теперішньої вартості (ЧТВ) та терміну окупності інвестованого капіталу. Оскільки система Nexus розгортається на хмарній інфраструктурі, до експлуатаційних витрат належить орендна плата за хмарні ресурси AWS та MongoDB Atlas[11]. Орієнтовну щомісячну вартість хмарної інфраструктури наведено в таблиці 4.5 (за курсом НБУ 44,90 грн за долар США станом на червень 2026 року).

Таблиця 4.5 — Орієнтовна щомісячна вартість хмарної інфраструктури

Сервіс	**Конфігурація**	**Вартість, $/міс**	**Вартість, грн/міс**
Amazon EC2	Інстанс t3.medium (2 vCPU, 4 ГБ)	30,37	1363,61
MongoDB Atlas	Кластер M10 (виділений)	57,00	2559,30
Amazon S3 + CloudFront	Сховище медіа та CDN	8,00	359,20
Разом	—	95,37	282,11

Таким чином, щомісячні витрати на хмарну інфраструктуру становлять близько 4282 грн, що в річному вимірі дорівнює приблизно 51385 грн. Завдяки використанню технологій з відкритим кодом ліцензійні відрахування відсутні, що суттєво знижує сукупну вартість володіння системою.

Чиста теперішня вартість визначається за формулою

$$ЧТВ = -КВ + \Sigma ГП / (1 + i)^t , \quad (4.13)$$

де КВ — затрати на проєкт; ГП — грошовий потік за t-ий рік; t — відповідний рік проєкту; і — величина дисконтної ставки (15 %).

Грошовий потік від упровадження системи формується за рахунок ефекту автоматизації організаційної діяльності — економії часу на адміністрування та усунення потреби в платних аналогах. Прийнято річний грошовий потік на рівні 20155,61 грн.

$$ЧТВ = -40311{,}22 + 20155{,}61/1{,}15 + 20155{,}61/1{,}15^2 + 20155{,}61/1{,}15^3 =$$
$$= 5708{,}58 \text{ грн.}$$

Оскільки ЧТВ ≥ 0, проєкт може бути рекомендований до впровадження. Термін окупності визначається за формулою

$$Ток = Тпв + НВ / ГПР , \quad (4.14)$$

де $T_{пв}$ — період до повного відшкодування витрат, років; НВ — невідшкодовані витрати на початок року, грн; ГПР — грошовий потік на початок року, грн.

Ток = 2 + 7544,06 / 20155,61 = 2,4 року.

Усі підсумкові показники зведено у таблицю 4.6.

Таблиця 4.6 — Техніко-економічні показники розробки

№ п/п	Показник	Значення
	Собівартість, грн	40311,22
	Плановий прибуток, грн	20155,61
	Ціна, грн	60466,83
	Чиста теперішня вартість, грн	5708,58
	Термін окупності, років	2,4

Враховуючи наведені показники, можна зробити висновок, що за терміну окупності 2,4 року та позитивної чистої теперішньої вартості розробка й упровадження системи Nexus є економічно доцільними. Основну частку собівартості становлять витрати на оплату праці розробника; для подальшого зниження вартості володіння доцільно оптимізувати конфігурацію хмарних ресурсів та використовувати моделі резервування потужностей AWS, які знижують вартість обчислювальних інстансів за умови довгострокового використання.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

5.1 Управління режимами праці та відпочинку для зниження психоемоційного напруження ІТ-фахівців

Професійна діяльність фахівців у сфері інформаційних технологій характеризується високим рівнем інтелектуального та психоемоційного навантаження. Розробка програмних продуктів супроводжується тривалою роботою за персональним комп'ютером, обробкою значних обсягів інформації, високою концентрацією уваги та жорсткими часовими обмеженнями. Сукупність цих чинників створює передумови для виникнення хронічної втоми, стресу, зниження працездатності та професійного вигорання.

Згідно з нормативно-правовими актами України з охорони праці, раціональна організація режимів праці та відпочинку є базовим превентивним заходом для збереження здоров'я персоналу[6]. Для розробників програмного забезпечення, чия діяльність належить до категорії з високим рівнем зорового та нервово-емоційного напруження, сумарна тривалість регламентованих перерв має становити не менше 50–60 хвилин за восьмигодинну зміну.

Для досягнення балансу між ефективною працею та відновленням ресурсів організму фахівцю доцільно дотримуватися таких організаційних заходів:

– встановити чіткий графік початку та завершення робочого дня;

– передбачити час для перерв, обіду та рухової активності (наприклад, 5 хвилин розминки кожні 45 хвилин);

– регулярно спілкуватися з колегами, що знижує рівень професійної ізоляції;

– вимикати робочі сповіщення в неробочий час і не перевіряти пошту пізно ввечері;

– вести облік власного стану для своєчасного виявлення накопиченої напруги.

Ефективність регламентованих перерв підвищується, якщо вони використовуються для активної зміни мікросередовища, а не для пасивного перегляду соціальних мереж на тому ж екрані. Під час перерв рекомендується виконувати гімнастику для очей, фізичну розминку для зняття статичного навантаження з м’язів шиї та спини, а також дихальні техніки для психологічного розвантаження.

На працездатність розробника суттєво впливають і параметри виробничого середовища. Оптимальними мікрокліматичними умовами вважають температуру повітря в межах 18–20 °C, відносну вологість 40–60 % та швидкість руху повітря 0,1–0,2 м/с. У разі дистанційного або гібридного режиму роботи додатково виникає ризик розмивання межі між робочим та особистим часом, тому важливо впроваджувати правила цифрової гігієни та поважати право працівника на відпочинок.

Таким чином, упровадження науково обґрунтованого режиму праці та відпочинку, підтримання оптимальних параметрів мікроклімату та захист особистого часу дають змогу зберегти продуктивність праці, мінімізувати кількість помилок у програмному коді та забезпечити безпечні умови життєдіяльності ІТ-фахівця.

5.2 Профілактика зорової втоми та нормалізація освітлення при роботі з відеодисплеями

Розробка та тестування інтерфейсу системи Nexus пов’язані з безперервним зоровим контролем інформації на екрані відеодисплея, що висуває підвищені вимоги до зорової системи розробника. Головними чинниками зорової втоми та розвитку комп’ютерного зорового синдрому є постійне фокусування погляду на близькій відстані, мерехтіння екрана, наявність відблисків та невідповідність параметрів освітлення нормативним вимогам.

Для запобігання зоровому перевтомленню першочергову увагу слід приділити нормалізації природного та штучного освітлення. Освітлення має бути рівномірним, не створювати різких тіней та відблисків на екрані монітора. Природне освітлення доцільно забезпечувати через віконні прорізи з боковим лівим розташуванням щодо робочого місця, а для регулювання світлового потоку в сонячні дні застосовувати жалюзі або щільні штори.

Штучне освітлення має бути комбінованим і складатися із загального та місцевого. Організація лише місцевого освітлення заборонена, оскільки різкий контраст між яскраво освітленим документом і темним приміщенням викликає швидку втому очей. Нормовані значення освітленості мають становити:

– на поверхні робочого столу в зоні розміщення документів — не менше 300–500 лк;

– на поверхні екрана монітора — не більше 300 лк, щоб уникнути засвічування та втрати контрастності.

Як джерела світла для загального освітлення доцільно використовувати світлодіодні світильники з тепло-білим або природно-білим спектром, розташовані паралельно лінії зору розробника. Важливу роль відіграє і правильне розташування монітора: оптимальна відстань від очей до екрана становить 60–70 см (не менше 50 см), а верхня кромка екрана має бути на рівні очей або трохи нижче.

Для індивідуальної профілактики зорової втоми рекомендується застосовувати правило «20–20–20»: кожні 20 хвилин роботи переводити погляд на предмет, що знаходиться на відстані близько 6 метрів, щонайменше на 20 секунд. Це дозволяє розслабити циліарний м’яз ока, який відповідає за акомодацію. Дотримання нормативних параметрів освітлення та ергономічних вимог до розміщення відеодисплея є надійним інструментом збереження зорового здоров’я ІТ-фахівця.

5.3 Ергономічне проєктування робочого простору розробника системи Nexus

Ергономічна організація робочого простору є ключовим чинником продуктивності праці, точності виконання завдань і збереження здоров'я розробника. Створення системи Nexus вимагає тривалого перебування в сидячому положенні та високої концентрації уваги, тому неправильно організоване робоче місце призводить до статичного перевантаження опорно-рухового апарату, порушення постави та розвитку тунельних синдромів кистей рук.

Оскільки розробка виконувалася на портативному комп'ютері, ергономічне проєктування простору має свої особливості. Фіксоване з'єднання екрана та клавіатури в ноутбуці змушує користувача нахиляти голову вперед і сутулитися. Для усунення цього чинника робоче місце доцільно укомплектувати похилою підставкою під ноутбук або зовнішнім монітором, а також окремою клавіатурою та ергономічною мишею.

Проєктування робочої зони базується на компонуванні меблів з урахуванням антропометричних характеристик тіла людини:

– висота робочої поверхні столу має становити 725–750 мм, а сама поверхня — бути матовою, щоб унеможливити відблиски; простір для ніг повинен мати висоту не менше 600 мм і ширину не менше 500 мм;

– крісло має бути підйомно-поворотним, із регулюванням висоти сидіння в межах 400–550 мм, анатомічною спинкою з валиком у зоні поперекового прогину та підлокітниками з регулюванням висоти;

– при правильній посадці кут у колінних та ліктьових суглобах має становити приблизно 90–100 градусів, а стопи мають повністю стояти на підлозі або на спеціальній підставці.

Організація робочих зон на столі здійснюється за принципом частоти використання предметів: у ближній зоні (30–40 см) розміщують клавіатуру та мишу, у середній (40–50 см) — ноутбук на підставці, у дальній (понад 50 см) — предмети, що використовуються рідко. Для зниження зорового і нервового напруження під час тривалого кодування доцільно використовувати темні теми в середовищах розробки та налаштовувати комфортний масштаб шрифтів за оптимальної яскравості й контрастності екрана.

Таким чином, комплексне ергономічне проєктування робочого простору мінімізує фізичне навантаження на організм розробника, запобігає появі втоми та створює оптимальні умови для успішного виконання професійних обов’язків.

ВИСНОВКИ

У межах кваліфікаційної роботи розроблено інформаційну систему автоматизації діяльності студентської ради Nexus на основі сучасного стеку технологій. Клієнтську та серверну частини реалізовано на базі фреймворку Next.js із застосуванням мови TypeScript, бібліотеки React і фреймворку Tailwind CSS, а для зберігання даних використано документоорієнтовану базу даних MongoDB.

В аналітичній частині роботи проведено дослідження наявних на ринку рішень — систем управління проєктами, систем управління знаннями, месенджерів та освітніх платформ. Виявлено, що жодне з них повністю не задовольняє потреб студентського самоврядування, що обґрунтувало доцільність створення власної спеціалізованої платформи. На основі аналізу сформульовано технічне завдання.

У процесі розробки спроєктовано модульну архітектуру з п’яти доменів, побудовано структуру бази даних із шести колекцій та реалізовано ключові підсистеми: кросплатформну авторизацію з інтеграцією Telegram Mini App, семирівневу модель контролю доступу (RBAC), візуальний конструктор сторінок на базі рушія Puck, систему відстеження завдань із гейміфікацією та механізм автоматичного створення робочих просторів у Telegram.

Систему розгорнуто на хмарній інфраструктурі Amazon Web Services із використанням обчислювального інстанса EC2, об’єктного сховища S3 та мережі доставки контенту CloudFront, а базу даних — на офіційній платформі MongoDB Atlas. Контейнеризація за допомогою Docker забезпечила передбачуване середовище виконання та спростила розгортання.

У фінансовій частині розраховано повну собівартість розробки, яка становить 40311,22 грн, та визначено ціну НДР — 60466,83 грн. За позитивної чистої теперішньої вартості 5708,58 грн та терміну окупності 2,57 року впровадження системи є економічно доцільним.

У частині охорони праці обґрунтовано раціональний режим праці та відпочинку, нормалізовано параметри освітлення та спроєктовано ергономічний робочий простір для тривалої роботи розробника.

Розроблена система повністю відповідає вимогам технічного завдання та дає змогу автоматизувати організаційну діяльність студентської ради, підвищити прозорість управління і рівень залученості студентів завдяки інтеграції зі звичним для них середовищем месенджера Telegram.

ПЕРЕЛІК ПОСИЛАНЬ

1) Bayer J. Next.js Quick Start Guide: Server-side rendering done right. Birmingham : Packt Publishing, 2018. 198 с.

2) Hahn E. Express.js in Action: Writing and testing Node applications. Shelter Island : Manning Publications, 2016. 236 с.

3) Wilson E. MERN Quick Start Guide: Build web applications with MongoDB, Express, React, and Node. Birmingham : Packt Publishing, 2018. 281 с.

4) Грибан В. Г., Фоменко А. Є., Казначеєв Д. Г. Безпека життєдіяльності та охорона праці : підручник. Дніпро : Дніпроп. держ. ун-т внутр. справ, 2022. 388 с.

5) Карпюк Г. І. Основи підприємництва : навч. посіб. Київ : Міністерство освіти і науки України, 2021. 105 с.

6) Кошель В. І., Сав'юк Г. П., Дзундза Б. С. Основи охорони праці : навчально-методичний посібник. Івано-Франківськ : НАІР, 2020. 182 с.

7) Amazon Web Services : офіц. сайт. URL: https://aws.amazon.com/ (дата звернення: 10.06.2026).

8) Amazon EC2 Pricing : офіц. сайт. URL: https://aws.amazon.com/ec2/pricing/ (дата звернення: 10.06.2026).

9) Docker : офіц. сайт. URL: https://www.docker.com/ (дата звернення: 05.06.2026).

10) DOU.ua — портал спільноти розробників та аналітики IT-ринку України : вебсайт. URL: https://dou.ua/ (дата звернення: 06.06.2026).

11) MongoDB Atlas : офіц. сайт. URL: https://www.mongodb.com/atlas (дата звернення: 28.05.2026).

12) Mongoose ODM : офіц. сайт. URL: https://mongoosejs.com/ (дата звернення: 28.05.2026).

13) Next.js : офіц. сайт. URL: https://nextjs.org/ (дата звернення: 23.05.2026).

14) Node.js : офіц. сайт. URL: https://nodejs.org/ (дата звернення: 23.05.2026).

15) Puck — visual editor for React : офіц. сайт. URL: https://puckeditor.com/ (дата звернення: 24.05.2026).

16) React : офіц. сайт. URL: https://react.dev/ (дата звернення: 23.05.2026).

17) Tailwind CSS : офіц. сайт. URL: https://tailwindcss.com/ (дата звернення: 23.05.2026).

18) Telegram Mini Apps : документація. URL: https://core.telegram.org/bots/webapps (дата звернення: 25.05.2026).

19) TypeScript : офіц. сайт. URL: https://www.typescriptlang.org/ (дата звернення: 23.05.2026).

20) Zod — TypeScript-first schema validation : офіц. сайт. URL: https://zod.dev/ (дата звернення: 24.05.2026).

21) Емоційне вигорання на роботі: що робити. Комп'ютерна академія IT STEP : блог. URL: https://itstep.org/ (дата звернення: 06.06.2026).

22) Мікроклімат виробничих приміщень. Державна служба України з питань праці : офіц. сайт. URL: https://dsp.gov.ua/ (дата звернення: 06.06.2026).

23) Що таке бази даних. Sigma Software University : вебсайт. URL: https://university.sigma.software/ (дата звернення: 23.05.2026).

24) Що таке тестування за методом «чорної скриньки». Sigma Software University : вебсайт. URL: https://university.sigma.software/black-box-testing/ (дата звернення: 01.06.2026).

ДОДАТКИ

Додаток А. Лістинг моделі користувача (user.model.ts)

```
import { Schema, model, models } from 'mongoose';
 const userSchema = new Schema(
  {
   login: { type: String, unique: true, required: true,
    match: /^[a-z0-9._-]+$/ },
   email: { type: String, sparse: true, lowercase: true },
   emailVerified: { type: Boolean, default: false },
   passwordHash: { type: String, select: false },
   googleId: { type: String, sparse: true },
   appleId: { type: String, sparse: true },
   telegramId: { type: String, sparse: true, index: true },
   username: String,
   name: String,
   surname: String,
   avatar: String,
   phone: String,
   role: { type: String,
    enum: ['Admin', 'StudentCouncil', 'Student'],
    default: 'Student' },
   accessLevelIndex: { type: Number, min: 0, max: 6, default: 6 },
   delegatedPermissions: { type: [String], default: [] },
   specialty: String,
   group: String,
   studentTitle: { type: String,
    enum: ['Starosta', 'Deputy', 'Neither'], default: 'Neither' },
   sociumRoles: { type: [String], default: [] },
   organizations: { type: [String], default: [] },
   socialLinks: { type: [String], default: [] },
```

```
    about: String,
    stars: { type: Number, default: 0 },
    warnings: { type: Number, default: 0 },
    selfGovernmentApplicationIntent: { type: Boolean, default: false },
    teacherAccessApproved: { type: Boolean, default: false },
    personalDataConsentAt: Date,
    lastTelegramSyncAt: Date,
    notificationChannels: { type: [String], default: ['web'] },
    webNotificationPromptAt: Date,
  },
  { timestamps: true }
);

export const User = models.User || model('User', userSchema);
```

Додаток Б. Лістинг моделі завдання (task.model.ts)

```
import { Schema, model, models } from 'mongoose';

const performerSchema = new Schema({
  userId: { type: Schema.Types.ObjectId, ref: 'User', required: true },
  qualityPercent: { type: Number, min: 0, max: 100 },
  timelinessPercent: { type: Number, min: 0, max: 100 },
  score: { type: Number, default: 0 },
  report: String,
}, { _id: false });

const taskSchema = new Schema(
  {
    title: { type: String, required: true, maxlength: 200 },
    description: String,
    status: { type: String,
      enum: ['draft', 'dispatched', 'acknowledged', 'in_progress',
        'submitted', 'completed', 'overdue', 'cancelled'],
      default: 'draft' },
    explanationMedia: { type: [String], default: [] },
    tags: { type: [String], default: [] },
    categoryId: String,
    categoryLabel: String,
    baseScore: { type: Number, default: 0 },
    authorUserId: { type: Schema.Types.ObjectId, ref: 'User' },
    authorDisplayName: String,
    performers: { type: [performerSchema], default: [] },
    dueAt: Date,
    delegationCount: { type: Number, default: 0 },
    groupId: { type: Schema.Types.ObjectId, ref: 'TaskGroup' },
    groupTitle: String,
    telegramForumTopicId: Number,
  },
  { timestamps: true }
);
```

```
export const Task = models.Task || model('Task', taskSchema);
```

Додаток В. Лістинг моделі сторінки (page.model.ts)

```
import { Schema, model, models } from 'mongoose';

const pageSchema = new Schema(
  {
    path: { type: String, unique: true, required: true },
    puckData: { type: Schema.Types.Mixed, default: {} },
    title: { type: String, required: true },
    published: { type: Boolean, default: false },
    categories: { type: [String], default: [] },
    description: String,
    coverImage: String,
    galleryImages: { type: [String], default: [] },
    catalogImagesPerCard: { type: Number, min: 1, max: 4, default: 1 },
    catalogCardVariant: { type: String,
      enum: ['tile', 'featured'], default: 'tile' },
    authorUserId: { type: Schema.Types.ObjectId, ref: 'User' },
    delegatedEditorUserIds: {
      type: [Schema.Types.ObjectId], ref: 'User', default: [] },
    publishAt: Date,
    commentsEnabled: { type: Boolean, default: true },
    notifyOnPublish: { type: Boolean, default: false },
    viewCount: { type: Number, default: 0 },
    likeCount: { type: Number, default: 0 },
    dislikeCount: { type: Number, default: 0 },
  },
  { timestamps: true }
);

export const Page = models.Page || model('Page', pageSchema);
```

Додаток Г. Лістинг моделей вподобань, аудиту та контролю доступу

```
// page_like.model.ts
const pageLikeSchema = new Schema({
  pagePath: { type: String, required: true },
  userId: { type: Schema.Types.ObjectId, ref: 'User', required: true },
});
pageLikeSchema.index({ pagePath: 1, userId: 1 }, { unique: true });

// user_directory_audit.model.ts
const auditSchema = new Schema({
  action: { type: String, enum: ['user_update', 'user_delete'] },
  actorUserId: { type: Schema.Types.ObjectId, ref: 'User' },
  targetUserId: { type: Schema.Types.ObjectId, ref: 'User' },
  changedFields: { type: [String], default: [] },
  summary: String,
  success: { type: Boolean, default: true },
}, { timestamps: true });

// access_control_settings.model.ts (сінглтон)
const accessControlSchema = new Schema({
  _id: { type: String, default: 'access_control_settings' },
  levels: { type: Array, default: [] },
  levelPermissions: { type: Schema.Types.Mixed, default: {} },
  grantRules: { type: Schema.Types.Mixed, default: {} },
});
```

Додаток Д. Лістинг функції перевірки прав доступу (access-control.ts)

```
import { AccessControlSettings } from "./models";

/**
 * Перевіряє, чи може актор виконати дію над цільовим користувачем.
 * @param actor      користувач, що ініціює операцію
 * @param target     цільовий користувач
 * @param permission ключ дозволу (напр. "users.assign_access_level")
 * @returns true, якщо операція дозволена
 */
export async function canActOn(actor, target, permission) {
  const settings = await AccessControlSettings.findById(
    "access_control_settings");

  // 1) у актора має бути відповідний ключ дозволу
  const allowed = settings.levelPermissions[actor.accessLevelIndex]
    ?.includes(permission)
    || actor.delegatedPermissions.includes(permission);
  if (!allowed) return false;

  // 2) рівень актора має суворо випереджати рівень цілі
  if (actor.accessLevelIndex >= target.accessLevelIndex) return false;

  return true;
}
```

Додаток Е. Лістинг функції обчислення підсумкового балу (score.ts)

```
/**
 * Обчислює підсумковий бал виконавця за завдання.
 * Формула: score = baseScore * (quality/100) * (timeliness/100).
 * Використовується цілочислова арифметика без чисел з рухомою комою.
 */
export function calcScore(baseScore, qualityPercent, timelinessPercent) {
  const q = Math.max(0, Math.min(100, qualityPercent | 0));
  const t = Math.max(0, Math.min(100, timelinessPercent | 0));
  return Math.round((baseScore * q * t) / 10000);
}

/**
 * Сумує всі бали користувача за період навчання для таблиці лідерів.
 */
export function totalStars(tasks, userId) {
  return tasks.reduce((sum, task) => {
    const p = task.performers.find(
      (x) => String(x.userId) === String(userId));
    return sum + (p ? p.score : 0);
  }, 0);
}
```

Додаток Ж. Лістинг маршруту публікації сторінки (api/pages/route.ts)

```
import { NextResponse } from "next/server";
import { Page } from "@/models/page.model";
import { pageSchema } from "@/validation/page.schema";

export async function POST(req) {
  const body = await req.json();

  // валідація вхідних даних за схемою Zod
  const parsed = pageSchema.safeParse(body);
  if (!parsed.success) {
    return NextResponse.json(
      { error: parsed.error.flatten() }, { status: 400 });
  }

  // перевірка унікальності URL
  const exists = await Page.findOne({ path: parsed.data.path });
  if (exists) {
    return NextResponse.json(
      { error: "URL already exists" }, { status: 409 });
  }

  const page = await Page.create(parsed.data);
  return NextResponse.json(page, { status: 201 });
}
```

Додаток 3. Лістинг конфігурації блоків візуального редактора (puck.config.tsx)

```
export const puckConfig = {
root: PageRoot as any,
components: {
NexusSection: shellBlock("NexusSection", NexusSection as any,
ROOT_SHELL_SPACING),
NexusGrid: shellBlock("NexusGrid", NexusGrid as any),
NexusSpacer: shellBlock("NexusSpacer", NexusSpacer as any),

NexusHeading: shellBlock("NexusHeading", NexusHeading as any),
NexusText: shellBlock("NexusText", NexusText as any),
NexusImage: shellBlock("NexusImage", NexusImage as any),
NexusQuote: shellBlock("NexusQuote", NexusQuote as any),
NexusVideo: shellBlock("NexusVideo", NexusVideo as any),
NexusAccordion: shellBlock("NexusAccordion", NexusAccordion as any),
NexusList: shellBlock("NexusList", NexusList as any),
NexusTabs: shellBlock("NexusTabs", NexusTabs as any),
NexusCarousel: shellBlock("NexusCarousel", NexusCarousel as any),
NexusInput: shellBlock("NexusInput", NexusInput as any),
NexusComments: shellBlock("NexusComments", NexusComments as any),

NexusNewsCard: shellBlock("NexusNewsCard", NexusNewsCard as any),
NexusNewsCatalog: shellBlock("NexusNewsCatalog", NexusNewsCatalog as
any),
},
categories: {
layout: {
title: "Layout",
defaultExpanded: false,
components: [
"NexusGrid",
```

```
"NexusSection",
"NexusSpacer",
],
},
content: {
title: "Content",
defaultExpanded: false,
components: [
"NexusAccordion",
"NexusQuote",
"NexusText",
"NexusCarousel",
"NexusComments",
"NexusInput",
"NexusHeading",
"NexusImage",
"NexusList",
"NexusTabs",
"NexusVideo",
],
},
news: {
title: "News & Cards",
defaultExpanded: false,
components: ["NexusNewsCard", "NexusNewsCatalog"],
},
},
} satisfies Config;

export default puckConfig;
```