Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Інформаційна система планування подій та особистого розкладу
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *126 «Інформаційні системи та технології»*
Шифр, назва

Освітня програма *«Інформаційні системи та технології»*
Назва

Шифр КвРІСТ 220183.22.01.07 ПЗ

Виконав здобувач IV курсу, група ІСТ-22-1 — Підпис — Людмила ЯНТКОВА (Ініціали, прізвище)

Керівник канд.фіз.-мат.наук, доцент (Науковий ступінь, учене звання) — Підпис — Тетяна КИСІЛЬ (Ініціали, прізвище)

Нормоконтролер канд.фіз.-мат.наук, доц (Науковий ступінь, учене звання) — Підпис — Тетяна КИСІЛЬ (Ініціали, прізвище)

До захисту допускаю:
завідувач кафедри КІІС — Підпис — Ольга ПАВЛОВА (Ініціали, прізвище)
«01» червня 2026 р.
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 126 ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

Освітня програма «ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Янтковій Людмилі Сергіївні
Прізвище, ім`я, по батькові студента

1. Тема проекту (роботи) Інформаційна система планування подій та особистого розкладу

Керівник проекту (роботи) Кисіль Тетяна Миколаївна, к.ф-м.н., доц.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 10.01.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

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

Теоретичні основи досліджуваної інформаційної системи планування подій та особистого розкладу

Проєктування інформаційної системи планування подій та особистого розкладу

Реалізація та тестування інформаційної системи планування подій та особистого розкладу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура та структура ситеми

Алгоритми та процеси функціонування

Візуалізація інтерфейсу користувача системи

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір архітектурних рішень, структури даних та алгоритмів функціонування	01.04.2026	виконано
5	Робота над розділом 3 – проєктування інформаційної системи планування подій та особистого розкладу	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Людмила ЯНТКОВА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Тетяна КИСІЛЬ
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ 220183.22.01.07 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРІСТ 220183.22.01.07 Е8	Архітектура та структура системи	1		
3		КвРІСТ 220183.22.01.07 Е8	Алгоритми та процеси функціонування	1		
4		КвРІСТ 220183.22.01.07 Е8	Візуалізація інтерфейсу користувача системи	1		

Зм	Арк	№ докум	Підпис	Дата	КвРІСТ 220183.22.01.07 ВП		
Розробив		Янткова			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Кисіль				У	
Н. контр.		Кисіль				ХНУ, ІСТ-22-1	
Затв.		Павлова		21.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система планування подій та особистого розкладу».

Автор роботи: Людмила ЯНТКОВА.

Керівник роботи: Тетяна КИСІЛЬ.

Пояснювальна записка: 65 с., 26 рис., 7 табл., 3 дод., 40 джерел.

Графічна частина: 3 креслення.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ІНФОРМАЦІЙНА СИСТЕМА, МОНІТОРИНГ, ПЛАНУВАННЯ ПОДІЙ, РОЗКЛАД, REST API.

Кваліфікаційна робота присвячена розробці та дослідженню інформаційної системи для планування подій та організації особистого розкладу користувача. Актуальність теми обумовлена зростанням потреби в ефективному управлінні часом у сучасному цифровому середовищі, де користувачі взаємодіють з великою кількістю задач, подій та інформаційних потоків. Відсутність єдиного інструменту для централізованого планування призводить до втрати даних, неузгодженості розкладу та зниження продуктивності.

Метою роботи є проєктування, реалізація та тестування програмної системи, яка забезпечує створення, редагування, зберігання та відображення подій, а також дозволяє ефективно керувати особистим розкладом. У процесі виконання роботи проведено аналіз існуючих рішень, визначено їх переваги та недоліки, обґрунтовано вибір архітектури системи. Розроблено модель бази даних, спроєктовано структуру програмного забезпечення, реалізовано серверну та клієнтську частини застосунку.

Результатом роботи є функціональний вебзастосунок, який забезпечує ефективне планування подій, підвищує продуктивність користувача та може бути використаний у практичній діяльності з можливістю подальшого розвитку та розширення функціональних можливостей.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

У сучасних умовах розвитку інформаційних технологій значно зростає роль цифрових інструментів, що забезпечують ефективну організацію часу та управління повсякденними задачами. Активне впровадження інформаційних систем у різні сфери діяльності людини сприяє автоматизації процесів планування, підвищенню продуктивності праці та оптимізації використання ресурсів. Особливо актуальним це є для студентів, працівників IT-сфери, менеджерів та інших категорій користувачів, які мають справу з великим обсягом задач, подій та дедлайнів.

Однією з основних проблем сучасного користувача є ефективне управління особистим розкладом та планування подій. Традиційні підходи, такі як паперові щоденники або базові списки завдань, поступово втрачають свою актуальність через обмежений функціонал, відсутність інтеграції та складність у масштабуванні.

На сьогодні існує значна кількість програмних рішень для планування подій, таких як Google Calendar, Notion, Microsoft Outlook та інші. Однак більшість із них мають ряд обмежень, зокрема складність інтерфейсу, надлишковий функціонал або, навпаки, недостатню адаптацію під індивідуальні потреби користувача. Це створює передумови для розробки власних інформаційних систем, які враховують специфічні вимоги та забезпечують більш зручний і ефективний процес планування.

Актуальність теми дипломної роботи полягає у необхідності створення інформаційної системи, яка забезпечує зручне, гнучке та ефективне планування подій і управління особистим розкладом. Така система повинна поєднувати простоту використання, функціональність та можливість подальшого розширення.

Метою даної дипломної роботи є розробка інформаційної системи планування подій та особистого розкладу, яка дозволяє користувачу створювати, редагувати та управляти подіями, здійснювати їх фільтрацію, отримувати нагадування та ефективно організовувати власний час.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз предметної області та існуючих рішень планування подій;

– визначити основні функціональні вимоги до системи;

– розробити архітектуру інформаційної системи;

– спроєктувати структуру бази даних;

– реалізувати програмну частину системи із використанням сучасних технологій.

Об’єктом дослідження є процес планування подій та організації особистого розкладу користувача.

Предметом дослідження є методи та засоби розробки інформаційних систем для планування подій із використанням сучасних веб-технологій.

У процесі виконання дипломної роботи використовуються такі методи дослідження: аналіз і узагальнення наукових джерел, моделювання інформаційних систем, проєктування баз даних, а також методи програмної реалізації веб-додатків.

Практичне значення отриманих результатів полягає у створенні інформаційної системи, яка може бути використана для організації особистого часу, планування подій та оптимізації повсякденної діяльності користувачів. Розроблена система може бути адаптована під різні сфери застосування та розширена додатковими функціональними можливостями.

Структура дипломної роботи включає вступ, три основні розділи, висновки та список використаних джерел. У першому розділі проведено аналіз предметної області та існуючих рішень. У другому розділі виконано проєктування інформаційної системи. У третьому розділі описано процес реалізації програмного забезпечення та результати його тестування.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ПОДІЙ ТА ОСОБИСТОГО РОЗКЛАДУ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

В житті кожен із нас хоча б раз користувався планером: чи то звичайним паперовим, чи додатком у телефоні, адже досить важко тримати всі події, завдання чи зустрічі в голові [1]. Людина стикається з цим кожен день навіть у найелементарніших речах, таких як планування власного часу чи розпорядку дня. Здавалося б, це такі банальні речі, але наскільки планування дій допомагає раціонально використовувати особистий час і ресурси, уникати поспішних рішень. Адже коли є план і людина чітко бачить, що їй потрібно зробити, легше коригувати дії, відстежувати власний прогрес і не втрачати мотивацію до роботи [2].

Навіть у давнину, коли ще не існувало жодних гаджетів, люди, самі того не знаючи, почали вести власні щоденники та планувати події [3]. Ще у XV ст. в Європі з'явилися так звані ефемериди – це друковані таблиці із попередньо обчисленими координатами Сонця, Місяця, планет та інших астрономічних об'єктів у послідовні моменти часу (рис. 1.1).

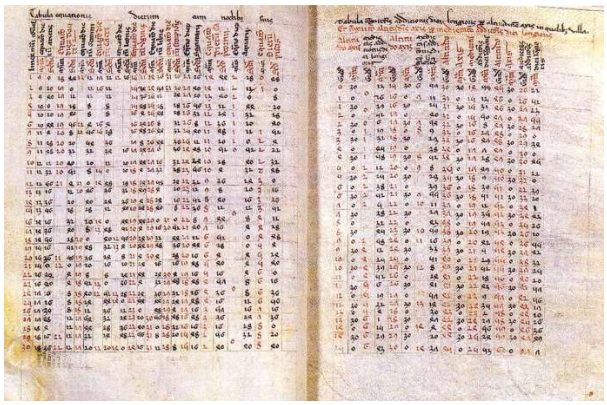


Рисунок 1.1 – Вигляд ефемериди [4]

Саме в них люди почали робити перші нотатки про важливі події. Трохи згодом вони почали вести щоденні записи про справи та зустрічі. Одним із

найбільш яскравих прикладів є англійський чиновник Samuel Pepys, який вів детальний щоденник, де записував події зі свого життя.

А ось першим електронним органайзером став портативний пристрій із функціями календаря, списку контактів і нотаток Psion Organiser (рис. 1.2), випущений у 1984 році. Він дозволяв створювати записи в календарі, зберігати контакти, планувати завдання, що стало важливим кроком до створення інформаційних систем планування.

Рисунок 1.2 – Psion Organiser [5]

Коли почали набирати популярності персональні комп'ютери, функції календаря почали інтегрувати в програмне забезпечення, що в 1990 році призвело до появи програми Microsoft Schedule+, яка стала аналогом усім відомого Microsoft Outlook, що об’єднав електронну пошту, календар, завдання та контакти в єдину систему [6].

Найбільшої популярності цифрові календарі набули з появою кишенькових комп’ютерів. Найвідомішим прикладом є Palm Pilot (рис. 1.3), який дозволяв синхронізувати календар із комп’ютером. З його допомогою користувачі могли встановлювати нагадування, створювати події, організовувати завдання. Це вже була повноцінна інформаційна система персонального планування.

Рисунок 1.3 – Palm Pilot [7]

Повертаючись до сьогодення, можна з впевненістю сказати, що всі ми плануємо та організовуємо свій час. Школярі, студенти, вчителі та педагоги щодня стикаються з проблемою правильного розподілу як навчального або робочого, так і вільного від основної активності часу [8]. В результаті це призводить до необхідності правильного планування свого графіка. Зараз існує дуже велика кількість інструментів та додатків для планування, проте користувачі часто стикаються із проблемами розподілу даних [9]. Частина з них може зберігатися в месенджерах, інша в електронній пошті чи окремих календарях. Саме відсутність єдиної інформаційної системи ускладнює роботу користувача, контроль виконання його завдань та створює ризики втрати важливої інформації чи її дублювання [10].

Ще одна проблема, з якою може стикатися користувач, – це спільні завдання чи події. Наприклад, команда планує підготовку до проведення заходу, тому необхідно узгодити дату зустрічі. І ось один із учасників має вільний час у понеділок ввечері, інший – тільки у вівторок зранку і так далі. В результаті обговорення та планування цієї зустрічі може зайняти декілька днів, поки не буде знайдено компроміс. У такому випадку за відсутності механізму перевірки зайнятості учасників починають виникати проблеми в розкладі, коли хтось просто забуде про подію чи неправильно вкаже свій вільний час [11].

Варто також звернути увагу на аспект зручності використання. Деякі з існуючих систем перевантажені функціями, які є зайвими для звичайного користувача. Інші ж, навпаки, надто обмежені у своїх можливостях і не дають змоги гнучко налаштовувати події, задавати пріоритети чи отримувати нагадування у зручному вигляді [12].

Ще одна проблема, яка може виникнути, полягає в тому, що люди по-різному підходять до планування. Якщо це студент, то для нього це розклад занять, підготовка до іспитів; якщо працівник - то це робочі завдання, графік зустрічей чи дотримання строків проєктів. В таких випадках підхід до кожного із користувачів має бути індивідуальним, а система планування має бути гнучкою, щоб кожен міг адаптувати її під себе та застосовувати ті функції, які потрібні саме для виконання його завдань [13].

Основним завданням після аналізу предметної області та всіх наявних проблем і завдань є створення єдиної інформаційної системи [14]. для планування, створення, редагування подій та організації гнучкого розкладу.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Після аналізу предметної області стало зрозуміло, що в нинішньому світі існує багато різноманітних програмних засобів та додатків, за допомогою яких можна планувати події та організовувати особистий розклад. Серед найпопулярніших можна виділити: Google Calendar, Trello, Notion та Todoist. Вивчення їхнього основного функціоналу, виявлення переваг та недоліків допоможе зробити інформаційну систему кращою і уникнути певних помилок.

Одним із найпопулярніших веб-орієнтованих сервісів для планування подій, який майже у кожної людини є в телефоні, – це Google Calendar [15]. З його допомогою можна створювати зустрічі, налаштовувати нагадування, відстежувати завдання, синхронізувати події з поштою тощо. Функціонал у

нього дуже різноманітний та доступний, саме тому він користується такою популярністю на ринку.

Серед основних його переваг простий та зрозумілий інтерфейс, автоматична синхронізація з іншими сервісами Google, гнучка система нагадувань та можливість запрошення учасників на події. Нижче (рис. 1.4) зображено меню перегляду завдань та подій.

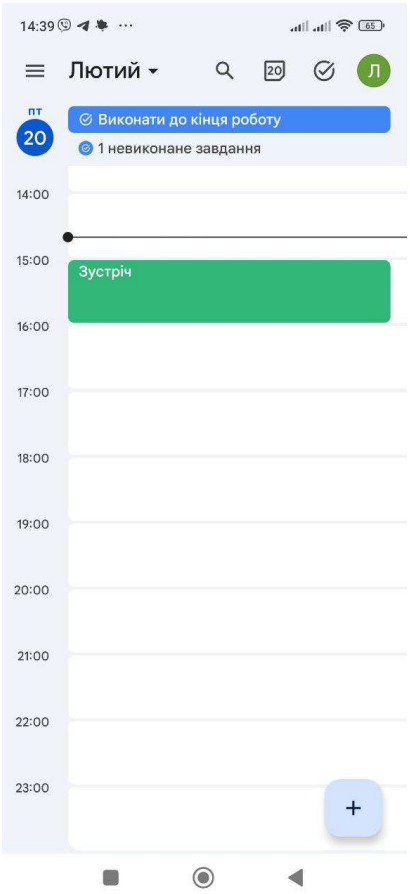


Рисунок 1.4 – Вигляд меню перегляду завдань та подій

Дуже серйозних недоліків немає, можна тільки виділити проблеми саме у випадках складної командної роботи:

– обмежені можливості в керуванні ролями учасників;

– брак розвинених інструментів для управління організацією подій;

– недостатня адаптованість до конкретних сценаріїв використання, таких як організація масштабних заходів.

Наступним програмним рішенням є додаток Trello [16]. Хоч у ньому і можна організовувати та керувати завданнями, проте основний його функціонал орієнтований саме на командну роботу та проєкти. Інтуїтивно зрозумілі функції Trello дають змогу кожній команді налаштувати й персоналізувати робочі процеси для будь-яких цілей.

Основними перевагами саме в плануванні подій та завдань є: зручна візуалізація завдань у форматі дошок, ефективне управління командною роботою та розподіл відповідальності серед учасників. Нижче (рис. 1.5) зображено додані завдання у додатку Trello.

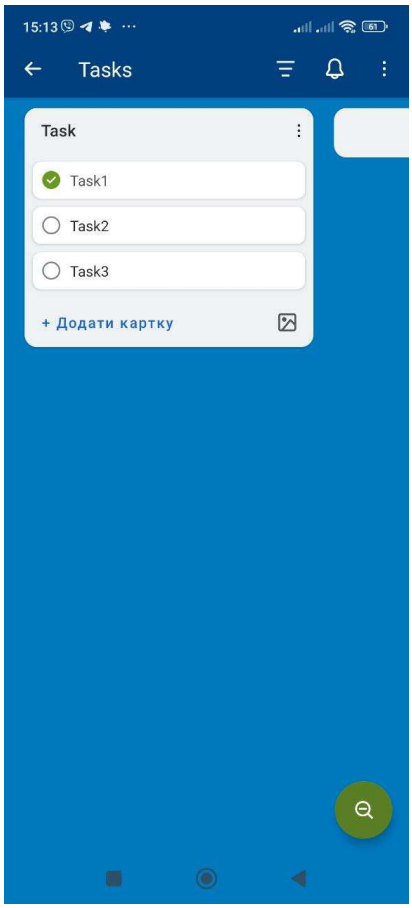


Рисунок 1.5 – Меню доданих завдань у додатку

Недоліки в першу чергу пов'язані з тим, що додаток орієнтований на командні проєкти:

– обмежені можливості для створення та ведення персонального розкладу;

– немає стандартного календарного інтерфейсу в базовій версії;

– необхідність додаткових інтеграцій для розширення функціоналу;

– для отримання більших можливостей необхідно придбати нові плани.

Ще одним прикладом системи планування подій є Notion [17]. У порівнянні з іншими додатками він є багатофункціональним середовищем для роботи з інформацією, яке постійно розвивається. Він об'єднує в собі можливості блокнота, бази даних, дошки для планування та wiki. Останні оновлення Notion додали чимало нових функцій, що роблять його ще більш ефективним і зручним інструментом як для роботи, так і для навчання.

Основними перевагами є: висока гнучкість і можливість налаштування, сумісність із базами даних та різними форматами подання інформації, здатність розробляти персоналізовані шаблони. Нижче (рис. 1.6) зображено приклад списку завдань та подій.

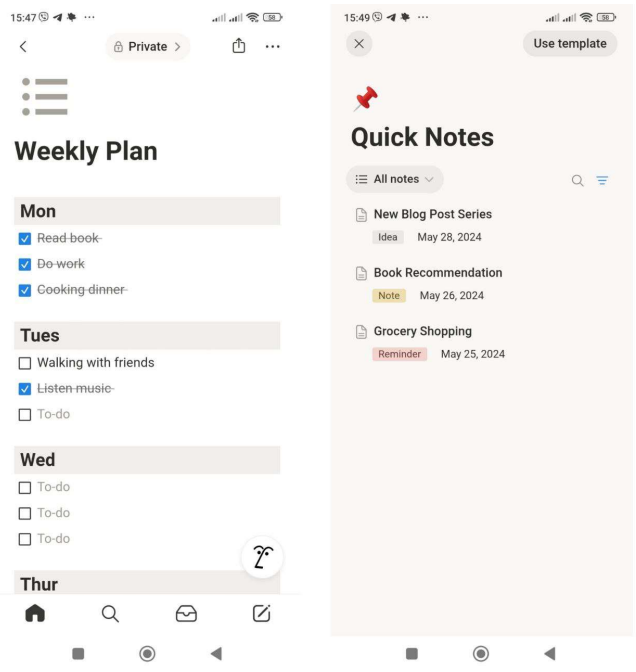


Рисунок 1.6 – Список завдань та подій в Notion

Із недоліків можна виділити:

– складнощі під час початкового налаштування;

– відсутність ефективних інструментів для автоматичного узгодження розкладу;

– надмірна кількість функцій, які ускладнюють використання для звичайних користувачів.

Наступним програмним рішенням, яке напряму пов'язане із плануванням подій та задач, є Todoist [18]. Це мобільний додаток-планувальник, спрямований на підвищення особистої продуктивності та ефективне планування повсякденних завдань. Todoist корисний інструмент для особистого тайм-менеджменту, але його можливостей не вистачає для комплексного управління подіями та синхронізації розкладів між кількома користувачами. Нижче (рис. 1.7) наведено інтерфейс додатка Todoist.

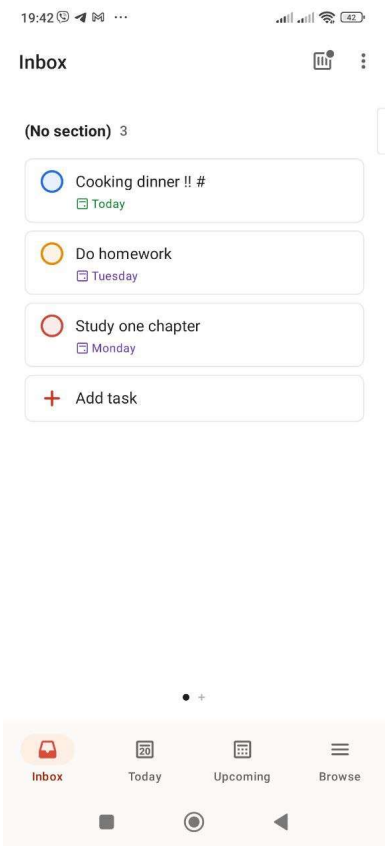


Рисунок 1.7 – Інтерфейс завдань у додатку Todoist

Переваги включають:

– зручне управління списками завдань із встановленням дедлайнів;

– підтримку пріоритетів і категорій;

– вбудовану систему нагадувань;

– можливість командної роботи над проєктами;

– кросплатформеність для вебу та мобільних додатків.

Щодо недоліків:

– недостатні інструменти для координації складних подій за участю кількох осіб;

– основний фокус на управлінні завданнями замість цілісного планування заходів;

– використання функціоналу доступне лише за платною підпискою.

Після аналізу існуючих рішень, таких як Google Calendar, Trello, Notion, Todoist, та визначення їх основних переваг і недоліків можна зробити висновки, що майже всі додатки мають обмежений функціонал, а також платний план, що обмежує повне використання додатка. Це показує, що жоден із них не може задовольнити всіх потенційних користувачів. Тому ще раз із впевненістю можна сказати про необхідність розробки схожої, але покращеної інформаційної системи, яка зможе виправити всі недоліки вже відомих нам рішень. Для кращого розуміння нижче наведена порівняльна характеристика всіх додатків (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика існуючих рішень

Критерій	**Google Calendar**	**Trello**	**Notion**	**Todoist**
Планування подій	Так	Частково	Так	Обмежено
Особистий розклад	Так	Обмежено	Частково	Так

Кінець таблиці 1.1

Нагадування	Так	Обмежено	Частково	Так
Командна робота	Обмежено	Так	Так	Частково
Гнучкість налаштувань	Обмежено	Так	Обмежено	Обмежено
Безкоштовний функціонал	Так	Обмежено	Обмежено	Обмежено

1.3 Підходи до вирішення задачі за темою дослідження

Так як потрібно створити інформаційну систему планування подій та особистого розкладу, найоптимальнішим рішенням буде використання веборієнтованої архітектури, яка зможе забезпечити доступ до системи з будь-яких пристроїв. Також можна застосувати клієнт-серверний принцип, який дозволить централізовано зберігати дані та здійснювати синхронізацію розкладу в реальному часі. Це дозволить централізовано керувати інформацією, синхронізувати її в реальному часі та зменшити ризик втрати даних.

Окремий акцент треба зробити саме на зручності користувацького інтерфейсу, адже саме клієнту повинно бути зручно користуватися цією системою. Тому вона повинна бути оснащена простими і зрозумілими інструментами, підтримувати перегляд розкладу, а також мати швидкий доступ до всіх основних функцій. Ще одним важливим моментом є розробка механізму сповіщень, за допомогою якого користувач зможе вчасно отримувати нагадування про майбутні події чи завдання та допоможе уникнути пропущених дедлайнів.

До базових аспектів також належить безпека даних, адже інформаційна система має пряме відношення до персональної інформації клієнтів. Тому необхідно забезпечити надійні методи автентифікації та захисту конфіденційних даних користувачів.

1.4 Постановка задачі

Система планування подій та організації особистого розкладу повинна реалізовувати основні функціональні можливості, зокрема створення, перегляд, редагування, збереження та видалення подій і завдань. Вона має забезпечувати формування індивідуального розкладу відповідно до потреб користувача з урахуванням часових обмежень, навантаження та пріоритетності запланованих дій. У межах розробки система повинна враховувати такі складові:

– характеристики подій і завдань, до яких належать дата, час початку та завершення, тривалість, рівень пріоритету, категорія, а також можливість повторення;

– параметри, що задаються користувачем, зокрема робочий графік, заплановані події, завдання та строки їх виконання;

– можливі обмеження, пов’язані з перевантаженням розкладу, відсутністю вільного часу або накладанням подій у межах одного періоду.

Система повинна забезпечувати зручне представлення розкладу у різних форматах, таких як денний, тижневий або місячний вигляд. Крім того, вона має підтримувати функцію нагадувань про заплановані події та завдання, що дозволяє своєчасно реагувати на наближення дедлайнів. У випадку виникнення конфліктів або некоректних даних система повинна інформувати користувача та пропонувати варіанти їх усунення.

Важливим елементом функціональності є підтримка спільних подій, яка дозволяє залучати інших користувачів, узгоджувати час проведення зустрічей та оперативно вносити зміни. Це сприяє більш ефективній взаємодії між учасниками та покращує процес організації спільної діяльності.

Особливу увагу необхідно приділити питанням безпеки, оскільки система працює з персональними даними. Для цього повинні бути реалізовані механізми автентифікації, авторизації та обмеження доступу, що гарантують захист інформації користувачів.

1.5 Висновки до розділу 1

У ході виконання першого розділу було здійснено комплексний аналіз предметної області, пов’язаної з плануванням подій та організацією особистого часу. Розглянуто етапи розвитку засобів планування – від традиційних паперових щоденників до сучасних цифрових рішень, що дозволило визначити основні тенденції розвитку даних систем.

Проведений аналіз показав, що сучасні користувачі постійно стикаються з необхідністю ефективного розподілу часу. При цьому наявні програмні рішення не завжди відповідають очікуванням щодо зручності, гнучкості та функціональних можливостей. Основними недоліками є фрагментація даних, складність організації спільних подій, обмеження у налаштуванні та надлишкова або недостатня функціональність окремих сервісів.

У підрозділі 1.2 було виконано порівняльний аналіз популярних систем, таких як Google Calendar, Trello, Notion та Todoist. У результаті визначено їх сильні та слабкі сторони, що дало змогу зробити висновок про відсутність універсального рішення, яке повністю задовольняє потреби користувачів. Зокрема, деякі системи орієнтовані на командну роботу, інші – на особисте планування, проте комплексного підходу до організації розкладу вони нажаль повністю не забезпечують.

У підрозділі 1.3 розглянуто основні підходи до розв’язання поставленої задачі. Обґрунтовано доцільність використання веборієнтованої клієнт-серверної архітектури, яка забезпечує доступність системи з різних пристроїв, централізоване зберігання інформації та її синхронізацію в реальному часі. Також наголошено на важливості зручного користувацького інтерфейсу, системи сповіщень та надійного захисту даних.

У підрозділі 1.4 сформульовано постановку задачі та визначено основні функціональні вимоги до системи. Зокрема, передбачено реалізацію повного циклу роботи з подіями, підтримку категорій, визначення пріоритетів, контроль

часових конфліктів та можливість організації спільних подій. Окрему увагу приділено забезпеченню безпеки та адаптивності системи під індивідуальні потреби користувачів.

Таким чином, у першому розділі було окреслено проблеми та сформовано вимоги до розроблюваної інформаційної системи. Отримані результати стали теоретичною базою для подальших етапів проєктування та реалізації програмного продукту, що розглядаються у наступних розділах роботи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ПОДІЙ ТА ОСОБИСТОГО РОЗКЛАДУ

2.1 Архітектура інформаційної системи планування подій та особистого розкладу

У процесі створення сучасних інформаційних систем велике значення має вибір архітектурного підходу, оскільки саме він визначає рівень гнучкості, масштабованості, продуктивності та можливості подальшого розвитку програмного продукту. Для реалізації інформаційної системи планування подій та особистого розкладу було обрано клієнт-серверну архітектуру, яка є загальноприйнятим стандартом для вебзастосунків і забезпечує чітке розмежування функціональних обов'язків між складовими системи [19].

Клієнт-серверна модель передбачає поділ системи на дві основні складові: клієнтську та серверну частини. Клієнт відповідає за взаємодію з користувачем, відображення інтерфейсу та обробку дій користувача, тоді як сервер забезпечує виконання бізнес-логіки, обробку запитів, взаємодію з базою даних і реалізацію механізмів безпеки [20].

У даній системі клієнтська частина реалізована із використанням бібліотеки React у поєднанні з мовою TypeScript [21]. Такий підхід дозволяє будувати інтерфейс за принципом компонентної структури, що полегшує масштабування та підтримку програмного коду. Застосування TypeScript забезпечує статичну типізацію, що сприяє зменшенню кількості помилок під час розробки та підвищує загальну надійність програмного забезпечення. Крім того, використання віртуального DOM у React дозволяє оптимізувати процес оновлення інтерфейсу, що позитивно впливає на швидкодію застосунку.

Серверна частина побудована на основі фреймворку Nest.js, який функціонує на платформі Node.js і підтримує сучасні підходи до організації коду. Зокрема, використовується модульна структура та принцип інверсії залежностей, що дозволяє розділити систему на окремі функціональні блоки:

модуль авторизації, модуль роботи з подіями, модуль категорій та модуль аналітики. Така організація сприяє підвищенню зручності супроводу та спрощує процес розширення функціональності системи.

Для зберігання інформації використовується реляційна система управління базами даних PostgreSQL, яка характеризується високою надійністю, підтримкою складних запитів і забезпеченням цілісності даних. Взаємодія з базою даних здійснюється за допомогою ORM Prisma, що дозволяє працювати з даними на рівні об’єктів, мінімізуючи необхідність прямого написання SQL-запитів і зменшуючи ймовірність виникнення помилок [22].

Передача даних між клієнтом і сервером реалізується через REST API із використанням протоколу HTTP. Дані передаються у форматі JSON, що забезпечує універсальність і сумісність між різними платформами. Кожен функціональний модуль має власний набір endpoint-ів, які відповідають за виконання конкретних операцій, наприклад створення подій, отримання списку записів або авторизацію користувача.

Значна увага в архітектурі системи приділяється забезпеченню безпеки. Для ідентифікації та автентифікації користувачів використовується технологія JSON Web Token (JWT), яка дозволяє реалізувати безпечний доступ до ресурсів. Після проходження авторизації користувач отримує токен, який використовується для подальших звернень до серверної частини. Такий підхід усуває необхідність зберігання сесій на сервері та сприяє підвищенню масштабованості системи [23].

Окрім цього, архітектура системи спроєктована з урахуванням можливості подальшого розширення. У перспективі передбачається інтеграція із зовнішніми сервісами, такими як Google Calendar або системи нагадувань, а також впровадження push-повідомлень. Завдяки використанню модульного підходу додавання нових функціональних можливостей може здійснюватися без значних змін у вже реалізованих компонентах.

Загальна структура архітектури інформаційної системи наведена на (рис. 2.1).

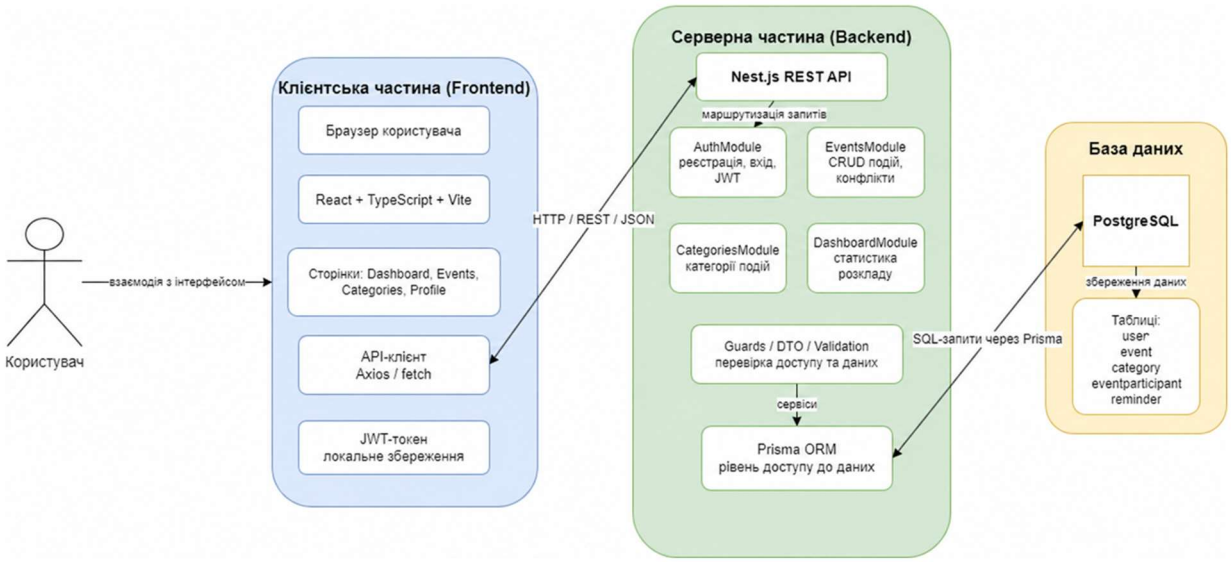


Рисунок 2.1 – Архітектура інформаційної системи планування подій та особистого розкладу

На представленому рисунку показано взаємодію між основними складовими системи, а саме клієнтською частиною, сервером та базою даних, а також відображено потоки передавання даних. Користувач здійснює взаємодію із системою через веббраузер, використовуючи інтерфейс застосунку. Усі дії користувача формують запити, які передаються на серверну частину, де вони обробляються відповідними модулями системи. Після обробки сервер повертає результати у форматі JSON, які далі відображаються у клієнтському інтерфейсі.

Запропонована архітектура забезпечує чіткий розподіл функціональності між компонентами, що сприяє ефективній взаємодії між ними. Такий підхід дозволяє реалізувати необхідний набір функцій та створює надійну основу для подальшого вдосконалення системи. Використання сучасних технологічних рішень і принципів проєктування позитивно впливає на якість програмного продукту та забезпечує його відповідність актуальним вимогам до вебзастосунків.

У результаті проведеного аналізу було визначено, що клієнт-серверна архітектура є найбільш доцільною для реалізації даної інформаційної системи. Застосування технологічного стеку, який включає React, Nest.js та PostgreSQL, дозволяє досягти високого рівня гнучкості, масштабованості та надійності.

Обрана архітектурна модель забезпечує ефективну реалізацію функціоналу системи та створює передумови для її подальшого розвитку й розширення.

2.2 Моделювання інформаційної системи

Процес проєктування інформаційної системи неможливий без попереднього моделювання, яке дозволяє формалізувати вимоги, визначити структуру системи та описати взаємодію її компонентів. Моделювання є важливим етапом розробки, оскільки воно забезпечує візуалізацію майбутньої системи, спрощує комунікацію між розробниками та дозволяє виявити потенційні проблеми ще до етапу реалізації.

Для моделювання інформаційної системи планування подій та особистого розкладу доцільно використовувати мову UML (Unified Modeling Language), яка є стандартом у сфері об’єктно-орієнтованого проєктування програмного забезпечення. UML надає набір діаграм, що дозволяють описати як статичну структуру системи, так і її поведінку [24].

На початковому етапі було побудовано діаграму варіантів використання (Use Case), зображеної на (рис. 2.2), яка відображає взаємодію користувача із системою. Основною діючою особою (актором) у системі є користувач, який виконує такі дії: реєстрація, авторизація, створення події, редагування події, видалення події, перегляд розкладу, управління категоріями та перегляд статистики. Дана діаграма дозволяє визначити основні функціональні можливості системи та межі її використання [25].

Діаграма варіантів використання також дозволяє чітко визначити межі системи та відокремити її від зовнішніх взаємодій. Завдяки цьому можна зрозуміти, які саме функції виконуються всередині системи, а які залежать від зовнішніх факторів або користувача. Важливою перевагою даної діаграми є можливість структурування функціоналу за допомогою зв’язків include та extend, що дозволяє уникнути дублювання логіки. Крім того, Use Case діаграма сприяє

кращому розумінню вимог замовника, оскільки вона відображає систему з точки зору кінцевого користувача. Це особливо важливо при розробці інтерфейсів, орієнтованих на зручність використання. Також дана діаграма є основою для подальшого створення інших UML-діаграм, зокрема діаграм послідовності та діяльності. Вона дозволяє визначити сценарії взаємодії, які будуть реалізовані у програмному забезпеченні. Окрім цього, Use Case діаграма допомагає виділити пріоритетні функції системи, що є важливим під час планування етапів розробки. Вона також сприяє виявленню додаткових вимог, які можуть виникнути у процесі аналізу. У результаті використання даного підходу формується більш повне уявлення про систему та її функціональні можливості. Це дозволяє мінімізувати ризики помилок на етапі реалізації та підвищує загальну якість розроблюваного програмного продукту.

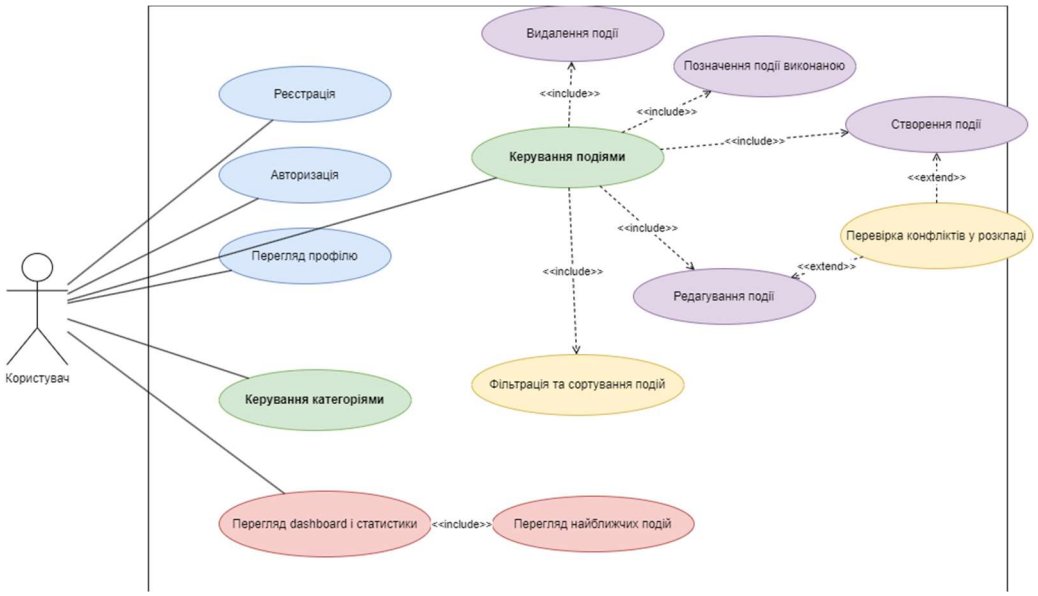


Рисунок 2.2 – Діаграма варіантів використання інформаційної системи

Для опису поведінки системи було використано діаграму послідовності (Sequence Diagram), зображена на (рис. 2.3), яка демонструє взаємодію між компонентами системи у часі. Наприклад, процес створення події включає взаємодію між користувачем, клієнтською частиною, сервером та базою даних. Користувач вводить дані у форму, після чого frontend відправляє запит на сервер,

сервер обробляє дані, перевіряє їх коректність та зберігає у базі даних. Після цього система повертає відповідь клієнту, і нова подія відображається у списку.

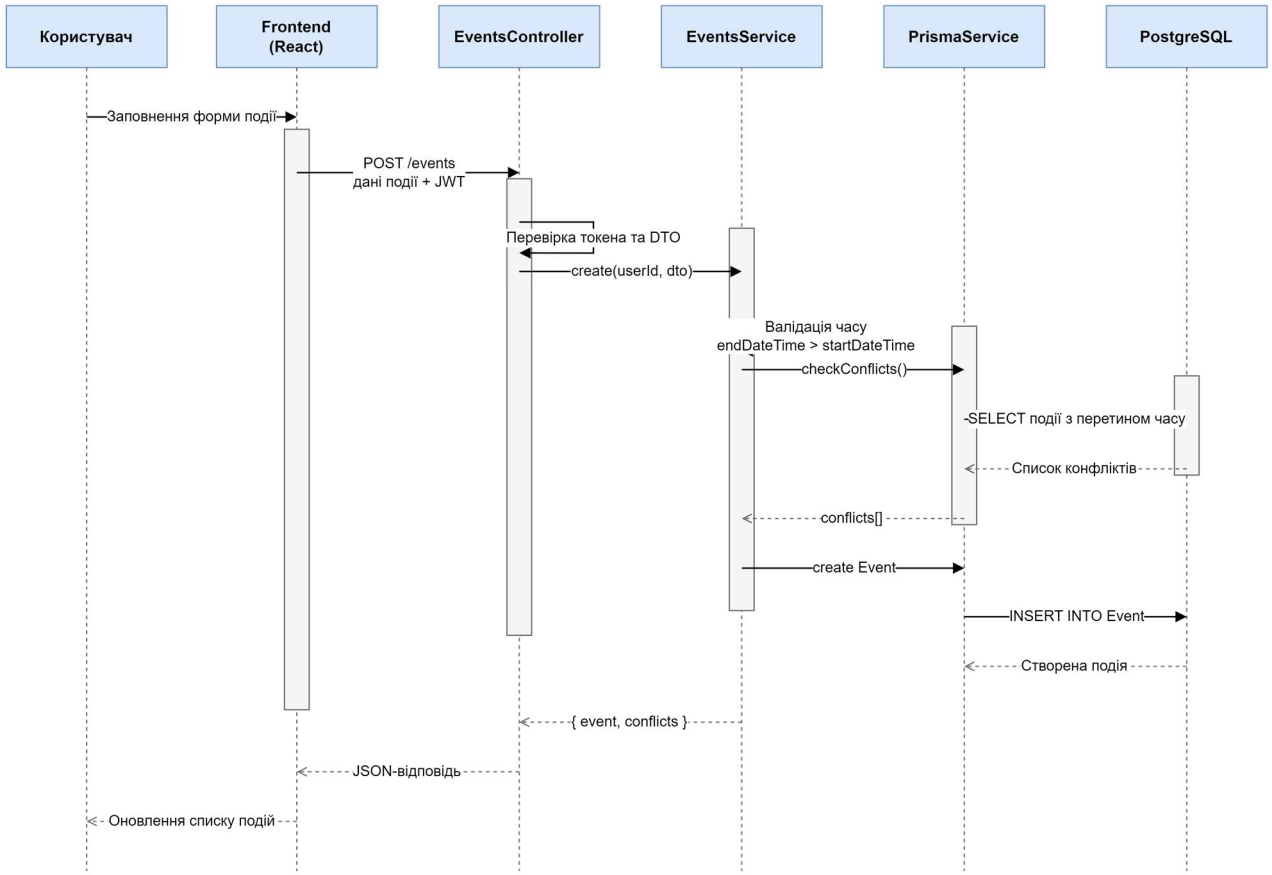


Рисунок 2.3 – Діаграма послідовності створення події

Крім того, важливим елементом моделювання є опис бізнес-логіки системи, зокрема алгоритму перевірки конфліктів у розкладі. При створенні або редагуванні події система повинна перевірити, чи не перетинається нова подія з уже існуючими. Це реалізується шляхом порівняння часових інтервалів подій. У випадку виявлення конфлікту система повідомляє користувача, але не блокує створення події, що забезпечує гнучкість використання [26].

Окрему увагу було приділено моделюванню станів подій. Кожна подія може мати один із кількох станів: активна, завершена або запланована. Це дозволяє більш ефективно організувати роботу з подіями та реалізувати фільтрацію у системі. Для цього може бути використана діаграма станів (State

Diagram), наведена на (рис. 2.4), яка відображає переходи між станами залежно від дій користувача.

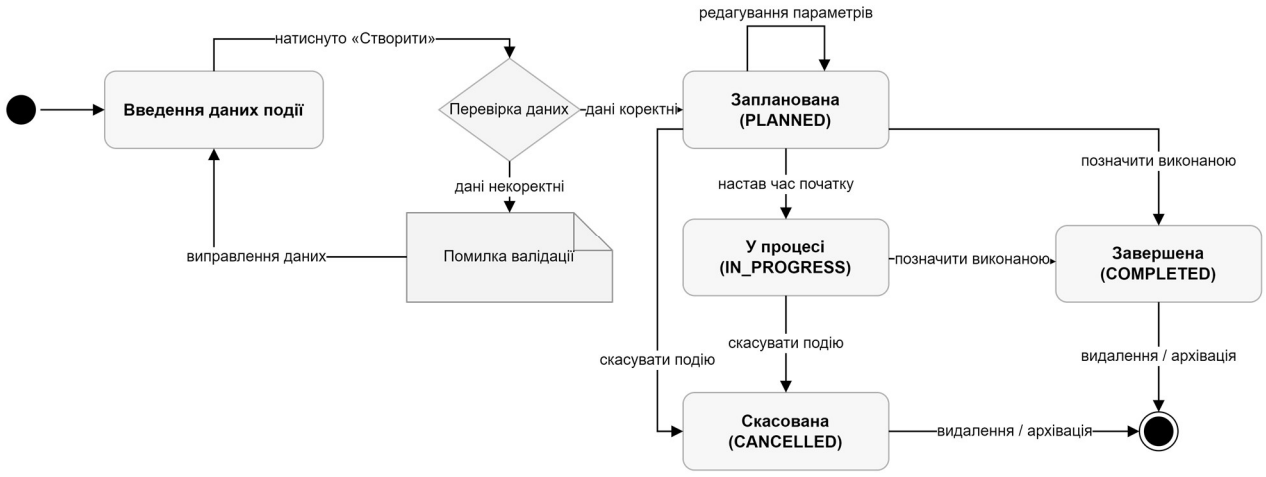


Рисунок 2.4 – Діаграма станів події

Використання UML-діаграм дозволило комплексно описати структуру та поведінку інформаційної системи. Це значно спростило процес подальшої реалізації, оскільки всі аспекти функціонування системи були формалізовані та візуалізовані на етапі проєктування.

У результаті моделювання інформаційної системи було сформовано її структурну та поведінкову модель із використанням UML-діаграм. Побудовані діаграми варіантів використання, послідовності та станів дозволили детально описати функціональність системи та взаємодію її компонентів. Отримані результати є основою для подальшої реалізації програмного забезпечення.

2.3 Проєктування бази даних інформаційної системи

Одним із основних етапів розробки інформаційної системи є проєктування бази даних, оскільки саме вона забезпечує збереження, обробку та структуризацію інформації. Від правильності побудови структури бази даних

залежить ефективність роботи всієї системи, швидкість виконання запитів, цілісність даних та можливість подальшого масштабування [27].

У межах даної дипломної роботи для реалізації бази даних було обрано реляційну модель, яка є доцільною для систем, де потрібно зберігати структуровані дані та забезпечувати чіткі зв’язки між сутностями. Такий підхід дозволяє визначити таблиці, їх поля, типи даних, первинні та зовнішні, а також правила взаємодії між записами. Як система управління базами даних використовується PostgreSQL, яка забезпечує надійність, підтримку транзакцій, цілісність даних та можливість виконання складних SQL-запитів.

Для спрощення взаємодії з базою даних використовується ORM Prisma. Вона дозволяє працювати з даними на рівні об’єктів, не виконуючи пряме написання SQL-запитів для кожної операції. Це прискорює розробку, зменшує ймовірність помилок та забезпечує кращу підтримку програмного коду. Крім того, Prisma дає змогу описати структуру бази даних у вигляді схеми, на основі якої можуть створюватися міграції та генеруватися типізовані методи для роботи з даними [28].

Основними сутностями бази даних інформаційної системи є користувач, подія, категорія, нагадування та учасник події. Відповідно до цього у базі даних передбачено п’ять основних таблиць: User, Event, Category, Reminder та EventParticipant. Така структура є більш повною, оскільки дозволяє не лише зберігати події користувача, а й підтримувати нагадування та можливість організації спільних подій.

Таблиця User призначена для збереження інформації про зареєстрованих користувачів системи. Вона містить унікальний ідентифікатор користувача, електронну пошту, хеш пароля, ім’я користувача, а також службові поля дати створення та оновлення запису. Для забезпечення безпеки пароль не зберігається у відкритому вигляді, а записується у вигляді хешу. Це унеможливлює його пряме відновлення навіть у випадку несанкціонованого доступу до бази даних.

Таблиця Event є центральною у системі, оскільки саме вона зберігає інформацію про заплановані події. До її основних полів належать назва події, опис, дата та час початку, дата та час завершення, статус, пріоритет, ознака завершеності, а також зв’язки з користувачем і категорією. Кожна подія обов’язково пов’язується з конкретним користувачем, що дозволяє розмежувати дані різних облікових записів.

Таблиця Category використовується для групування подій за певними ознаками. Наприклад, користувач може створити категорії “навчання”, “робота”, “особисте”, “зустрічі” тощо. Наявність категорій підвищує зручність використання системи, оскільки дозволяє швидше знаходити потрібні події, фільтрувати їх та візуально розділяти за типом.

Для реалізації механізму нагадувань у базі даних передбачено таблицю Reminder. Вона містить інформацію про час нагадування, тип нагадування та його стан. Виділення нагадувань в окрему таблицю є більш гнучким рішенням, оскільки одна подія може мати одне або декілька нагадувань. Це також спрощує подальше розширення системи, наприклад додавання email-сповіщень або push-повідомлень.

Таблиця EventParticipant призначена для підтримки спільних подій. Вона дозволяє зберігати інформацію про учасників, які запрошені до певної події, їх email та статус участі. Такий підхід створює основу для реалізації функціоналу спільного планування, коли одна подія може мати кількох учасників. Це особливо важливо для організації зустрічей, навчальних заходів або робочих подій.

Зв’язки між таблицями реалізуються за принципом “один-до-багатьох”. Один користувач може мати багато подій і категорій, одна категорія може бути пов’язана з багатьма подіями, а одна подія може мати декілька нагадувань та учасників. Така структура дозволяє ефективно організувати зберігання інформації та підтримувати цілісність даних [29].

Для кращого розуміння структури бази даних основні таблиці та їх поля наведено у (табл. 2.1 - 2.5.).

Таблиця 2.1 – Структура таблиці User

Поле	**Тип**	**Опис**
id	UUID	Унікальний ідентифікатор користувача
email	String	Електронна пошта користувача
passwordHash	String	Хеш пароля користувача
fullName	String	Ім’я користувача
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення запису

Таблиця User використовується для збереження інформації про зареєстрованих користувачів системи. Кожен запис містить унікальний ідентифікатор, email для входу в систему, хеш пароля та службові поля для фіксації часу створення й оновлення запису.

Таблиця 2.2 – Структура таблиці Event

Поле	**Тип даних**	**Опис**
id	UUID	Унікальний ідентифікатор події
title	String	Назва події
description	String	Опис події
startDateTime	DateTime	Дата та час початку події
endDateTime	DateTime	Дата та час завершення події
priority	String	Пріоритет події
status	String	Статус події
reminderMinutesBefore	Int	Час нагадування до початку події
isCompleted	Boolean	Ознака завершення події

Кінець таблиці 2.2

userId	UUID	Ідентифікатор користувача
categoryId	UUID	Ідентифікатор категорії події
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення запису

Таблиця Event є основною таблицею системи, оскільки саме в ній зберігаються всі заплановані події користувача. Вона містить часові параметри, статус, пріоритет, дані про нагадування та зв’язки з користувачем і категорією.

Таблиця 2.3 – Структура таблиці Category

Поле	**Тип даних**	**Опис**
id	UUID	Унікальний ідентифікатор категорії
name	String	Назва категорії
color	String	Колір категорії для візуального відображення
userId	UUID	Ідентифікатор користувача, якому належить категорія
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення запису

Таблиця Category призначена для групування подій за певними ознаками. Вона дозволяє користувачу створювати власні категорії, наприклад “Навчання”, “Робота”, “Особисте” або “Зустрічі”, що покращує зручність фільтрації та візуального сприйняття розкладу.

Таблиця 2.4 – Структура таблиці Reminder

Поле	**Тип даних**	**Опис**
id	UUID	Унікальний ідентифікатор нагадування
remindAt	DateTime	Дата та час спрацювання нагадування

Кінець таблиці 2.4

type	String	Тип нагадування
isSent	Boolean	Ознака надсилання нагадування
eventId	UUID	Ідентифікатор події
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення запису

Таблиця 2.5 – Структура таблиці EventParticipant

Поле	**Тип даних**	**Опис**
id	UUID	Унікальний ідентифікатор учасника
email	String	Електронна пошта учасника події
status	String	Статус участі
eventId	UUID	Ідентифікатор події
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення запису

При проєктуванні бази даних було враховано вимоги до нормалізації, що дозволило уникнути дублювання даних і забезпечити їх узгодженість. Наприклад, категорії не дублюються у таблиці подій, а зберігаються окремо та пов’язуються з подіями через зовнішній ключ. Аналогічно нагадування та учасники подій винесені в окремі таблиці, що забезпечує більшу гнучкість структури.

Використання зовнішніх ключів гарантує цілісність зв’язків між таблицями. Наприклад, кожна подія повинна бути пов’язана з існуючим користувачем, а кожне нагадування - з конкретною подією. Це дозволяє уникнути появи “відірваних” записів, які не мають логічного зв’язку з основними сутностями системи.

Окрему увагу приділено забезпеченню коректності даних. Зокрема, для подій передбачено перевірку часових параметрів, відповідно до якої дата та час завершення не можуть бути раніше дати та часу початку. Також використовується обмеження унікальності для електронної пошти користувача, що запобігає створенню дублюючих облікових записів.

Для підвищення швидкодії системи доцільно використовувати індекси для полів, які найчастіше застосовуються під час пошуку та фільтрації. До таких полів належать userId, categoryId, eventId та startDateTime. Це дозволяє швидше отримувати список подій конкретного користувача, фільтрувати записи за категоріями та формувати розклад на певний період [30].

Для підтримки повної узгодженості даних на рівні СУБД PostgreSQL було налаштовано правила каскадного оновлення та видалення зв'язаних записів за допомогою налаштування зв'язків між таблицями. Зокрема, у випадку видалення облікового запису користувача, усі створені ним події та персональні категорії автоматично вилучаються із системи. Це дозволяє уникнути появи «втрачених» записів у базі даних, які засмічують дисковий простір та порушують логіку бізнес-моделі. Аналогічно, видалення конкретної події автоматично ініціює очищення зв'язаних із нею нагадувань та списку запрошених учасників.

Розроблена структура бази даних є логічно узгодженою та відповідає вимогам інформаційної системи планування подій. Вона забезпечує зберігання основної інформації про користувачів, події, категорії, нагадування та учасників. Використання PostgreSQL та Prisma дозволяє спростити реалізацію доступу до даних, забезпечити їх цілісність та створити основу для подальшого розширення функціональних можливостей системи.

У результаті проєктування було сформовано структуру бази даних, яка включає основні сутності системи та зв’язки між ними. Обрана реляційна модель забезпечує ефективну організацію даних, підтримку складних запитів та надійне збереження інформації. Виділення додаткових таблиць Reminder та

EventParticipant дозволяє зробити систему більш повною і підготувати її до подальшого розвитку.

2.4 Алгоритми роботи інформаційної системи

Після завершення етапів проєктування архітектури та структури бази даних важливим кроком є визначення алгоритмів функціонування інформаційної системи. Саме алгоритми описують послідовність виконання операцій під час обробки запитів користувача та визначають логіку взаємодії між окремими компонентами системи. Чітке формалізування цих процесів дозволяє забезпечити правильну реалізацію програмного забезпечення та підвищує його надійність і стабільність роботи.

У межах розроблюваної системи увага приділяється таким процесам, як реєстрація та авторизація користувача, створення і редагування подій, виявлення конфліктів у розкладі, а також фільтрація й відображення інформації застосунку.

Процедура реєстрації користувача передбачає введення основних даних, зокрема електронної пошти, імені та пароля. Отримана інформація передається на сервер, де здійснюється її перевірка на коректність і відповідність встановленим вимогам. У разі успішної валідації пароль шифрується із використанням криптографічних методів, після чого дані користувача зберігаються у базі даних.

Алгоритм авторизації базується на перевірці введених облікових даних. Якщо зазначені користувачем email та пароль співпадають із даними у базі, система формує JWT-токен, який повертається клієнтській частині та використовується для подальшої взаємодії із сервером. У випадку некоректних даних користувач отримує відповідне повідомлення про помилку.

Однією з основних функцій системи є створення подій. Цей процес починається із введення користувачем необхідної інформації: назви події, опису,

часу початку та завершення, категорії та інших параметрів. Після цього дані надсилаються на сервер, де проходять етап перевірки.

На етапі валідації перевіряється:

– наявність обов’язкових полів;

– правильність формату введених дат;

– логічна узгодженість часових параметрів, зокрема те, що час завершення не може бути раніше часу початку.

Перед збереженням нової події система виконує сканування таблиці для поточного ідентифікатора користувача, де шукає записи, які задовольняють вищевказане часове обмеження. Завдяки попередньо налаштованим індексам на часові мітки, операція пошуку конфліктів виконується із високою швидкістю, що забезпечує миттєвий зворотний зв'язок для користувача під час інтерактивного керування розкладом. Якщо запит повертає записи, система ініціює алгоритм попередження, формуючи структуру даних із переліком ідентифікаторів конфліктних подій для подальшої візуалізації на стороні клієнтського інтерфейсу.

Після успішного проходження перевірок система переходить до етапу аналізу можливих конфліктів у розкладі. Якщо конфліктів не виявлено або користувач підтверджує створення події попри попередження, запис додається до бази даних і відображається у списку подій.

Виявлення конфліктів у розкладі є важливим функціональним елементом системи. Для цього виконується порівняння часових інтервалів нової події з уже існуючими записами користувача. Якщо відбувається перетин інтервалів, система фіксує конфлікт.

Формально така перевірка базується на наступних умовах:

– початок нової події відбувається раніше завершення існуючої;

– завершення нової події відбувається пізніше початку існуючої.

У разі виявлення накладання подій система інформує користувача про конфлікт. При цьому створення події не блокується повністю, що дозволяє зберегти гнучкість використання системи.

Важливою складовою функціонування алгоритмів системи є багаторівневий контроль доступу та захист даних користувачів. Під час авторизації та взаємодії клієнтської частини з сервером система чітко розмежовує права доступу до інформації. Кожен запит, що надходить на сервер для створення, редагування чи видалення подій, обов'язково проходить етап перевірки приналежності даних. Це унеможливлює ситуацію, коли один користувач може випадково або навмисно отримати доступ до розкладу, категорій чи нагадувань іншого учасника системи.

Редагування події передбачає зміну її параметрів із подальшою перевіркою правильності введених даних. Логіка цього процесу подібна до створення події, однак додатково включає отримання поточних значень та їх оновлення.

Видалення події реалізується шляхом надсилання відповідного запиту на сервер із передаванням ідентифікатора запису. Після перевірки прав доступу система видаляє запис із бази даних, а клієнтська частина оновлює список подій.

Для підвищення зручності користування система підтримує механізми фільтрації та сортування. Користувач має змогу відбирати події за різними критеріями, такими як дата, категорія, статус або пріоритет.

Алгоритм фільтрації полягає у формуванні запиту до бази даних з урахуванням заданих параметрів. Сервер обробляє запит та повертає лише ті записи, які відповідають встановленим умовам. Отримані результати відображаються у вигляді списку або календарного представлення.

Усі описані алгоритми наведені у додатку Б.

Таким чином, у межах даного підрозділу було сформовано алгоритмічне забезпечення основних функцій інформаційної системи. Розроблені алгоритми гарантують коректну обробку даних, ефективну взаємодію між користувачем і системою, а також реалізацію функцій, зокрема створення подій, контроль

конфліктів та управління розкладом. Це створює надійну основу для подальшої реалізації програмного забезпечення у наступному розділі.

2.5 Висновки до розділу 2

У другому розділі дипломної роботи було здійснено комплексне проєктування інформаційної системи планування подій та особистого розкладу, що є важливим етапом перед її програмною реалізацією. Отримані результати дозволили сформувати цілісне бачення структури системи, принципів її функціонування та взаємодії між окремими компонентами.

У підрозділі 2.1 було визначено та обґрунтовано доцільність використання клієнт-серверної архітектури як оптимального рішення для побудови веборієнтованої інформаційної системи. Описано основні складові системи - клієнтську частину, серверну логіку та базу даних, а також розглянуто принципи їх взаємодії. Використання сучасного технологічного стеку, зокрема React, Nest.js, PostgreSQL, Prisma та JWT, забезпечує необхідний рівень продуктивності, безпеки та масштабованості.

У підрозділі 2.2 було виконано моделювання системи за допомогою UML-діаграм. Побудовано діаграму варіантів використання, яка відображає взаємодію користувача із системою, діаграму послідовності, а також діаграму станів, які характеризують поведінку системи. Це дозволило формалізувати процеси та забезпечити наочне представлення логіки роботи системи.

У підрозділі 2.3 було розроблено модель бази даних на основі реляційного підходу. Визначено основні сутності, їх властивості та взаємозв’язки. Обрана структура забезпечує збереження цілісності даних, ефективну обробку інформації та створює умови для подальшого розвитку системи. Застосування PostgreSQL разом із ORM Prisma дозволяє значно спростити роботу з даними та підвищити надійність системи.

У підрозділі 2.4 було сформовано алгоритмічне забезпечення основних функцій системи. Описано алгоритми реєстрації та авторизації користувачів, створення, редагування та видалення подій, механізм виявлення конфліктів у розкладі, а також процеси фільтрації та відображення інформації. Розроблені алгоритми забезпечують коректну обробку даних та ефективну взаємодію користувача із системою.

У другому розділі було сформовано повну проєктну модель інформаційної системи, яка включає архітектурні рішення, структуру даних та алгоритми функціонування. Отримані результати є основою для подальшої реалізації програмного продукту, що розглядається у наступному розділі дипломної роботи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Обґрунтування вибору технологій для реалізації

Реалізація інформаційної системи планування подій та особистого розкладу потребує використання сучасних технологій, які забезпечують високу продуктивність, масштабованість, безпеку та зручність розробки. Враховуючи поставлені вимоги, було обрано стек технологій, що базується на веборієнтованій клієнт-серверній архітектурі із використанням REST API, реляційної бази даних та сучасних мов програмування.

Основною мовою програмування для реалізації системи є TypeScript, яка використовується як у клієнтській, так і у серверній частині. TypeScript є надбудовою над JavaScript [31] і забезпечує статичну типізацію, що дозволяє зменшити кількість помилок, підвищити якість коду та спростити підтримку проєктів. Використання єдиної мови програмування для frontend і backend дозволяє уніфікувати підхід до розробки та підвищити ефективність роботи [32].

Клієнтська частина інформаційної системи реалізована за допомогою бібліотеки React, наведена на (рис. 3.1), яка забезпечує компонентно-орієнтований підхід до розробки інтерфейсу користувача. Це дозволяє розбити інтерфейс на незалежні частини, що значно спрощує розробку, тестування та повторне використання коду. Для побудови оптимізованого середовища розробки використовується Vite, який забезпечує швидкий запуск застосунку та підтримку гарячого оновлення модулів (Hot Module Replacement) [33].

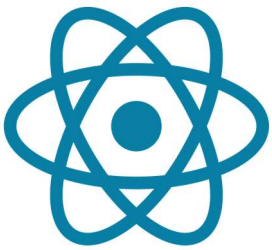


Рисунок 3.1 – Лого React

Для організації взаємодії між клієнтською та серверною частинами застосовується архітектурний підхід REST. REST API забезпечує впорядковану систему запитів до сервера, де кожна операція реалізується через відповідні HTTP-методи: GET, POST, PATCH/PUT, DELETE. Такий підхід широко використовується у сучасних вебзастосунках, оскільки забезпечує зручність інтеграції та можливість масштабування.

Серверна частина реалізована на базі фреймворку Nest.js, зображений на (рис. 3.2), який функціонує поверх платформи Node.js [34]. Даний фреймворк підтримує модульний підхід, що дозволяє розподілити систему на окремі логічні компоненти, зокрема модулі авторизації, управління подіями, категоріями та аналітики. Це забезпечує впорядкованість структури коду, спрощує його супровід і дає змогу легко розширювати функціональність системи [35].

Рисунок 3.2 – Лого Nest.js

У якості системи управління базами даних використовується PostgreSQL, яка є однією з найбільш надійних і продуктивних реляційних СУБД. PostgreSQL забезпечує підтримку транзакцій, цілісність даних, а також ефективну роботу з великими обсягами інформації. Вона добре підходить для систем, де важлива структурованість даних і наявність зв’язків між сутностями [36].

Для взаємодії з базою даних використовується ORM Prisma, зображена на (рис. 3.3), яка дозволяє описувати структуру бази даних у вигляді декларативної

схеми. Prisma автоматично генерує типізований клієнт для роботи з даними, що значно спрощує написання запитів та зменшує ймовірність помилок. А також підтримує міграції, що дозволяє контролювати зміни структури бази даних [37].

Рисунок 3.3 – Лого Prisma

Для реалізації механізму автентифікації використовується технологія JSON Web Token (JWT), яка дозволяє організувати безпечний доступ до ресурсів системи. Після успішної авторизації користувач отримує токен, який передається у заголовках HTTP-запитів і використовується для перевірки прав доступу. Це дозволяє уникнути зберігання сесій на сервері та підвищує масштабованість системи [38].

Для забезпечення безпеки паролів застосовується бібліотека bcrypt, яка виконує їх хешування перед збереженням у базі даних. Це забезпечує захист облікових даних користувачів навіть у випадку компрометації бази даних.

У клієнтській частині для виконання HTTP-запитів до серверу використовується бібліотека Axios, яка забезпечує зручний інтерфейс для роботи з REST API. Вона дозволяє налаштовувати заголовки запитів, обробляти помилки та працювати з асинхронними операціями [39].

Для реалізації маршрутизації використовується React Router, який дозволяє організувати навігацію між сторінками застосунку без перезавантаження сторінки. Це забезпечує зручність використання та покращує користувацький досвід [40].

Обраний стек технологій повністю відповідає вимогам до інформаційної системи. Використання REST API, PostgreSQL, Nest.js та React дозволяє створити сучасний, масштабований та безпечний вебзастосунок, який може бути використаний як повноцінний програмний продукт.

У результаті було обґрунтовано вибір технологій для реалізації інформаційної системи. Використання TypeScript як основної мови програмування, REST архітектури для взаємодії компонентів, PostgreSQL для збереження даних та сучасних фреймворків React і Nest.js забезпечує ефективну реалізацію системи. Обрані технології дозволяють створити надійний, продуктивний та зручний у використанні програмний продукт.

3.2 Проєктування та реалізація бази даних інформаційної системи

На етапі реалізації інформаційної системи планування подій та особистого розкладу було виконано проєктування та реалізацію бази даних із використанням реляційної моделі. У якості системи управління базами даних обрано PostgreSQL, а для взаємодії з нею використано ORM Prisma, що забезпечує високий рівень абстракції та спрощує роботу з даними.

База даних (рис. 3.4) реалізована на основі трьох основних сутностей: User, Event та Category. Взаємозв’язки між ними організовані за принципом “один-до-багатьох”, що відповідає логіці предметної області. Один користувач може мати декілька подій та категорій, а кожна подія належить конкретному користувачу та певній категорії.

Структура бази даних описується у файлі конфігурації Prisma, де визначаються моделі, їх поля, типи даних та зв’язки. Використання UUID як первинного ключа дозволяє забезпечити унікальність записів та зручність роботи у розподілених системах. Основні моделі представлені у файлі schema.prisma .

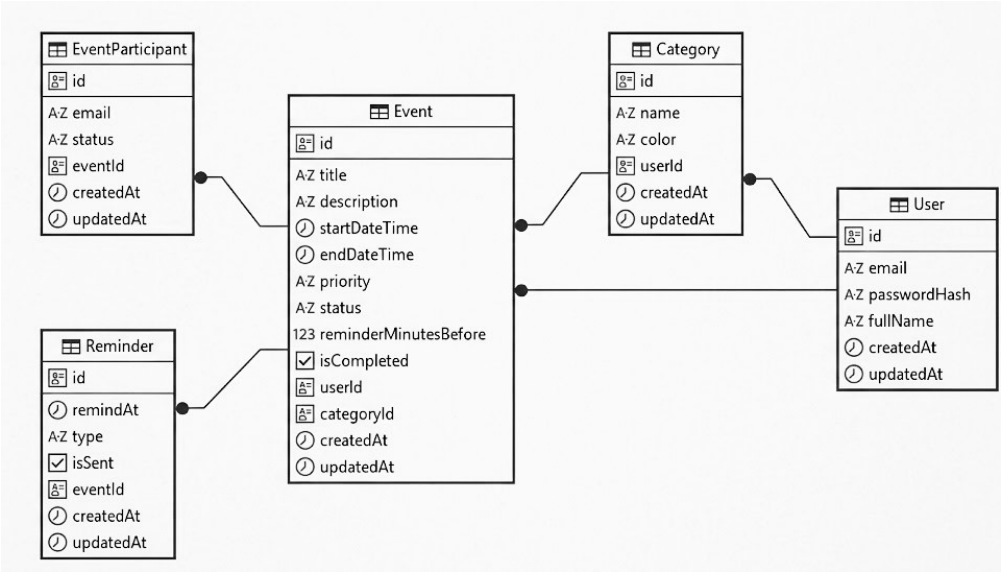


Рисунок 3.4 – ER-діаграма бази даних

Модель User містить інформацію про користувачів системи, зокрема електронну пошту, хеш пароля та ім’я. Для забезпечення унікальності використовується обмеження @unique для поля email. Додатково застосовуються поля createdAt та updatedAt для відстеження змін записів.

Модель Category призначена для групування подій. Вона містить назву категорії, колір та посилання на користувача. Використання складеного унікального ключа (@@unique([userId, name])) дозволяє уникнути дублювання категорій у межах одного користувача.

Модель Event є центральною у системі та описує події. Вона включає такі поля, як назва, опис, дата початку та завершення, пріоритет, статус, а також параметри нагадування. Для оптимізації запитів використовуються індекси, зокрема для полів userId та startDateTime, що дозволяє значно прискорити пошук подій.

Зв’язки між таблицями реалізовані за допомогою анотації @relation. Наприклад, кожна подія має зовнішні ключі userId та categoryId, які забезпечують зв’язок із відповідними таблицями. При видаленні користувача всі пов’язані події та категорії також видаляються, що гарантує цілісність даних.

Важливою частиною реалізації є початкове наповнення бази даних, яке здійснюється за допомогою файлу seed.ts . У ньому створюється тестовий користувач, базові категорії та декілька подій. Це дозволяє швидко протестувати функціональність системи без необхідності ручного введення даних.

Крім того, для роботи з базою даних використовуються типізовані запити Prisma Client. Наприклад, створення події виконується через метод create, отримання списку подій - через findMany, а фільтрація - із використанням умов у блоці where. Саме це дозволяє реалізувати складні запити без прямого використання SQL.

Особливу увагу приділено перевірці конфліктів подій. Для цього використовується запит, який перевіряє перетин часових інтервалів. Якщо нова подія починається раніше завершення існуючої і завершується пізніше її початку, система визначає конфлікт. Такий підхід дозволяє ефективно контролювати розклад користувача.

Міграції бази даних виконуються за допомогою інструменту Prisma Migrate. Команда npx prisma migrate dev створює SQL-скрипти, які синхронізують структуру бази даних зі схемою Prisma. Це забезпечує контроль змін та спрощує розгортання системи (рис. 3.5).

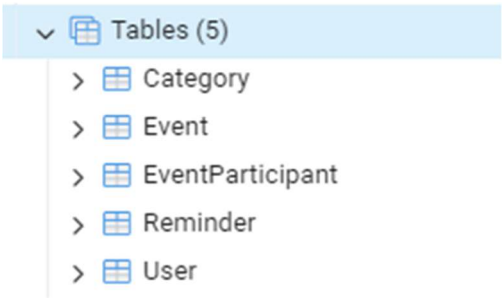


Рисунок 3.5 – Структура таблиць у PostgreSQL

Реалізована база даних відповідає всім вимогам інформаційної системи, забезпечує ефективне зберігання та обробку даних, а також дозволяє легко масштабувати систему у майбутньому.

У результаті було реалізовано базу даних інформаційної системи із використанням PostgreSQL та Prisma ORM. Визначено структуру основних сутностей, їх зв’язки та обмеження цілісності. Реалізація забезпечує ефективну роботу з даними, підтримку складних запитів та можливість подальшого розширення системи.

3.3 Реалізація серверної частини інформаційної системи

Реалізація серверної частини інформаційної системи планування подій та особистого розкладу виконана із використанням фреймворку Nest.js на базі платформи Node.js. Такий підхід дозволяє реалізувати масштабовану, модульну та підтримувану архітектуру, яка відповідає сучасним вимогам до вебзастосунків.

Серверна частина організована за модульним принципом і включає такі основні компоненти: модуль авторизації (auth), модуль роботи з подіями (events), модуль категорій (categories), модуль статистики (dashboard), а також модуль інтеграції з базою даних (prisma) . Така структура дозволяє чітко розділити функціональність системи та забезпечити зручність її подальшого розширення.

Модуль авторизації відповідає за реєстрацію, вхід користувача та перевірку доступу до захищених ресурсів. У контролері реалізовано endpoint-и для реєстрації (/auth/register), авторизації (/auth/login) та отримання інформації про поточного користувача (/auth/me).

У процесі реєстрації система перевіряє наявність користувача з таким самим email, після чого створює новий запис. При авторизації відбувається перевірка пароля за допомогою бібліотеки bcrypt, що забезпечує безпечне зберігання облікових даних.

Після успішного входу генерується JWT-токен, який використовується для доступу до захищених ресурсів системи. Для перевірки токена використовується стратегія JwtStrategy та захисний механізм JwtAuthGuard, що реалізує контроль доступу до endpoint-ів (рис. 3.6).

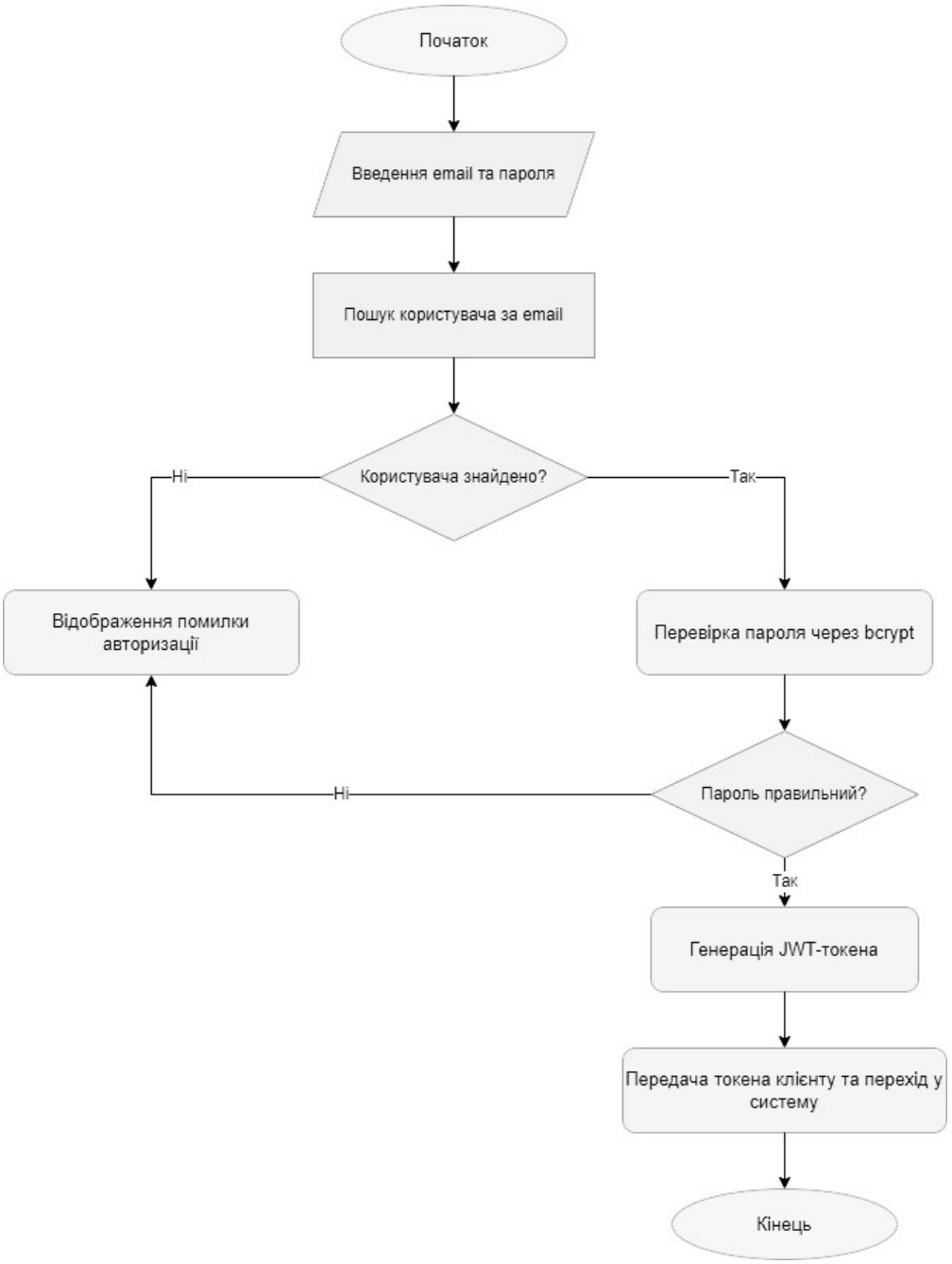


Рисунок 3.6 – Логіка авторизації користувача

Модуль роботи з подіями є центральним у системі, оскільки він забезпечує основний функціонал планування. У контролері реалізовано повний набір CRUD-операцій.

Сервіс подій містить бізнес-логіку обробки даних. Зокрема, при створенні події виконується нормалізація даних та перевірка конфліктів у розкладі.

Результатом створення події є об’єкт події разом із інформацією про можливі конфлікти, що дозволяє користувачу приймати рішення щодо планування.

Фільтрація подій реалізується через параметри запиту, такі як категорія, статус або пріоритет. Це дозволяє гнучко управляти відображенням даних та покращує зручність використання системи.

Модуль категорій забезпечує можливість створення, редагування, перегляду та видалення категорій. Кожна категорія прив’язана до конкретного користувача, що дозволяє персоналізувати структуру даних.

При видаленні категорії виконується перевірка на наявність пов’язаних подій. Якщо такі події існують, система не дозволяє видалити категорію та повертає повідомлення про помилку. Це забезпечує цілісність даних та запобігає втраті інформації.

Модуль статистики відповідає за формування агрегованих даних, які відображаються на головній сторінці застосунку. Зокрема, реалізовано підрахунок:

– загальної кількості подій;
– кількості завершених подій;
– кількості майбутніх подій.

Обчислення виконується за допомогою запитів до бази даних через Prisma Client, що дозволяє ефективно обробляти великі обсяги даних.

Для роботи з базою даних використовується сервіс PrismaService, який інкапсулює Prisma Client та забезпечує підключення до PostgreSQL. Усі операції з даними виконуються через типізовані методи, такі як findMany, create, update, delete, що значно спрощує розробку та підвищує надійність системи.

Для перевірки коректності вхідних даних використовуються DTO-об’єкти разом із бібліотекою class-validator. Це дозволяє перевіряти типи даних, їх формат та обов’язковість перед обробкою на сервері.

У системі реалізовано обробку помилок із використанням стандартних виключень Nest.js, таких як:

– BadRequestException;
– UnauthorizedException;
– NotFoundException.

Це дозволяє формувати зрозумілі повідомлення про помилки та підвищує якість взаємодії з користувачем.

Взаємодія між клієнтом і сервером здійснюється через REST API. Основні endpoint-и включають:

– /auth/login - авторизація користувача;
– /events - робота з подіями;
– /categories - управління категоріями;
– /dashboard/stats - отримання статистики.

Використання REST дозволяє забезпечити універсальність системи та можливість її інтеграції з іншими сервісами.

У результаті було реалізовано серверну частину інформаційної системи з використанням сучасного стеку технологій. Модульна архітектура, використання JWT для авторизації, інтеграція з базою даних через Prisma та реалізація REST API забезпечують високу продуктивність, безпеку та масштабованість системи. Отримане рішення повністю відповідає вимогам дипломної роботи та може бути використане як основа для подальшого розвитку програмного продукту.

3.4 Реалізація клієнтської частини інформаційної системи

Реалізація клієнтської частини інформаційної системи планування подій та особистого розкладу виконана з використанням бібліотеки React із застосуванням мови TypeScript. Такий підхід дозволяє створити сучасний односторінковий застосунок (SPA), який забезпечує швидку взаємодію з користувачем та ефективну роботу інтерфейсу.

Клієнтський застосунок побудований за компонентним підходом, що передбачає розбиття інтерфейсу на незалежні елементи (компоненти), які можна повторно використовувати. Це значно спрощує підтримку проєкту, дозволяє масштабувати систему та забезпечує високу читабельність коду.

Структура клієнтської частини організована за принципом розділення відповідальності. Основні директорії включають:

- components - багаторазові UI-компоненти;
- pages - сторінки застосунку;
- api - модулі для роботи з REST API;
- routes - налаштування маршрутизації;
- types - TypeScript-типи;
- utils - допоміжні функції.

Такий підхід дозволяє забезпечити логічну організацію коду та спростити процес розробки.

Для навігації між сторінками використовується бібліотека React Router. Вона дозволяє реалізувати багатосторінковий інтерфейс без перезавантаження сторінки, що покращує користувацький досвід.

Основні маршрути застосунку:

- /login - сторінка авторизації;
- /register - сторінка реєстрації;
- /dashboard - головна сторінка;
- /events - список подій;
- /events/create - створення події;
- /events/:id - перегляд події;
- /categories - управління категоріями.

Для обміну даними з серверною частиною використовується бібліотека Axios. Вона дозволяє виконувати HTTP-запити та обробляти відповіді сервера. Було реалізовано централізований API-клієнт, який автоматично додає JWT-

токен до заголовків запитів. Це дозволяє забезпечити доступ до захищених endpoint-ів без необхідності повторної авторизації.

Користувацький інтерфейс системи включає набір сторінок, які забезпечують повний функціонал роботи з подіями.

На сторінках входу та реєстрації (рис. 3.7) користувач може створити обліковий запис або увійти до системи. Дані передаються на сервер, після чого користувач отримує JWT-токен.

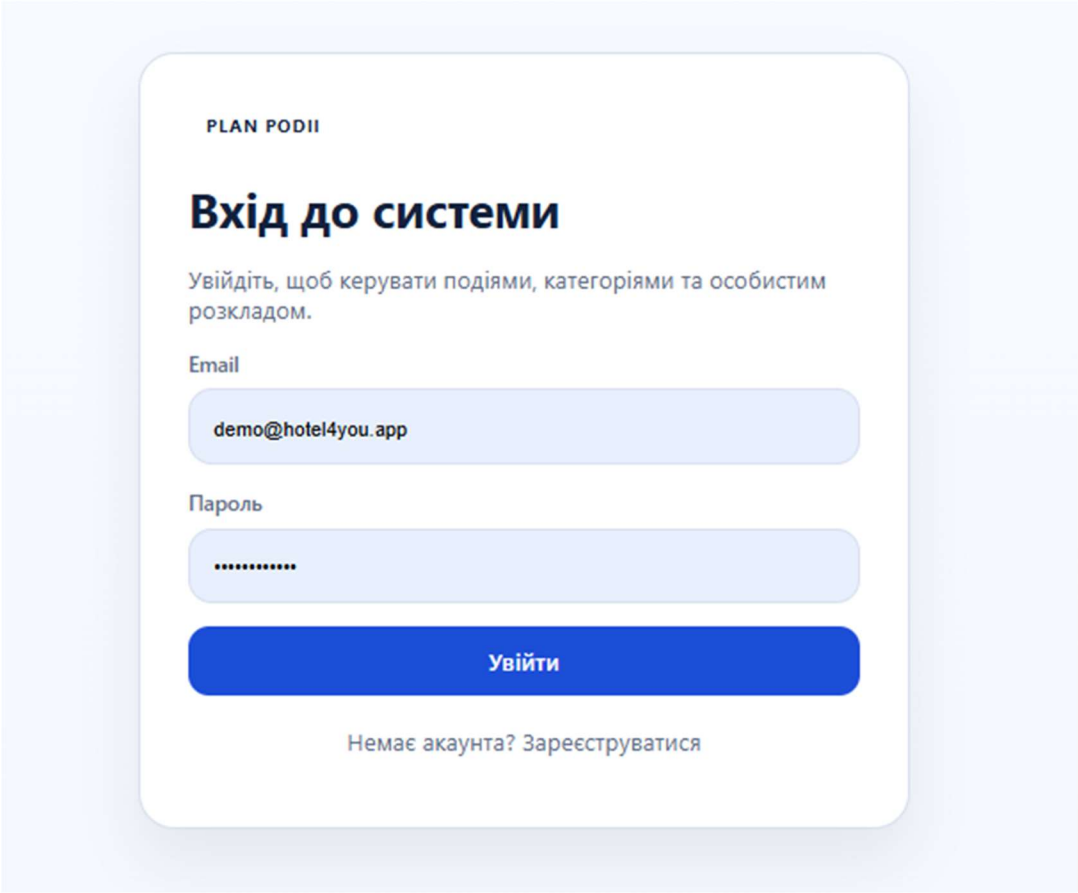


Рисунок 3.7 – Сторінка авторизації користувача

На даній сторінці відображається список усіх подій користувача. Передбачено можливість:

– перегляду подій;
– переходу до детальної інформації;
– редагування та видалення;

– фільтрації за різними параметрами.

Для створення події реалізована форма, наведена нижче (рис. 3.8), яка містить поля для введення назви, опису, дати та часу, категорії та інших параметрів. Перед відправкою дані проходять базову перевірку.

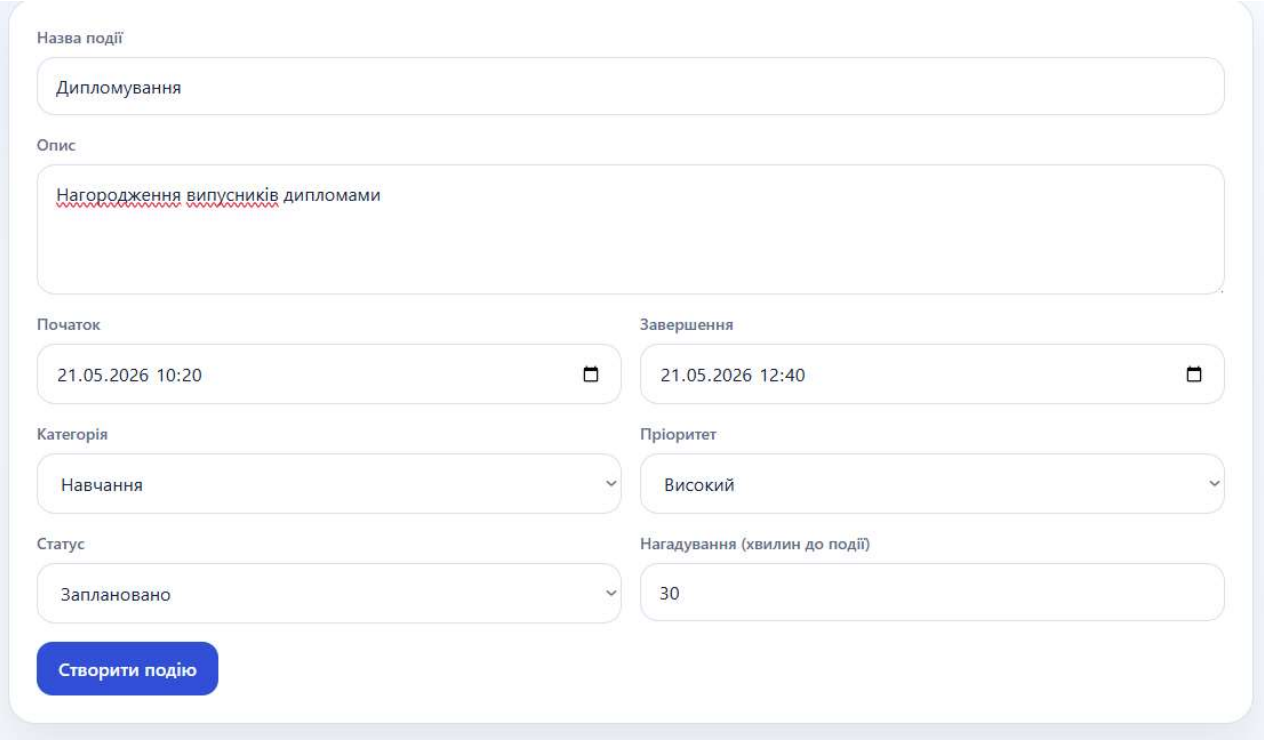


Рисунок 3.8 – Форма створення події

Користувач має можливість створювати власні категорії, редагувати їх та видаляти (рис. 3.9). Це дозволяє структурувати події та спрощує їх пошук.

Головна сторінка системи (рис. 3.10) відображає статистичну інформацію:

– загальну кількість подій;
– кількість виконаних подій;
– найближчі події;
– майбутні події.

Це дозволяє користувачу швидко оцінити свій розклад.

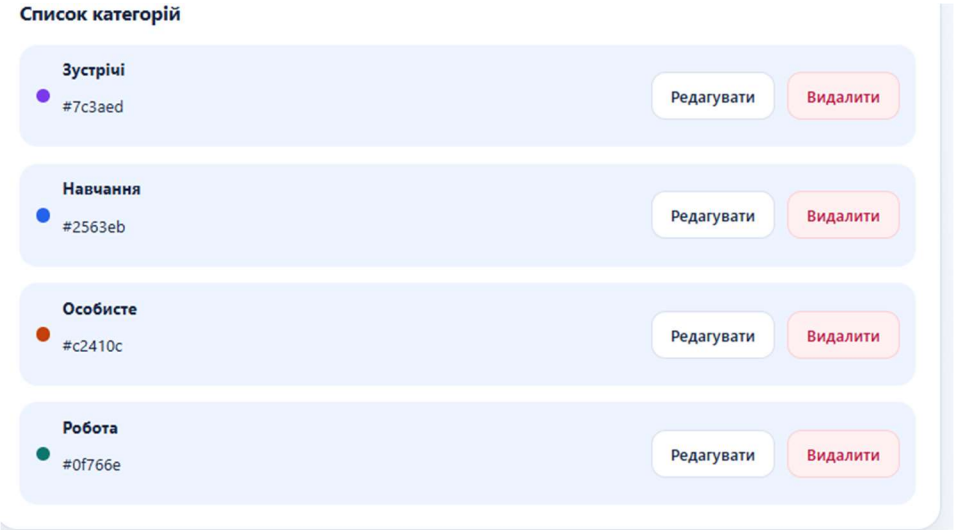


Рисунок 3.9 – Управління категоріями

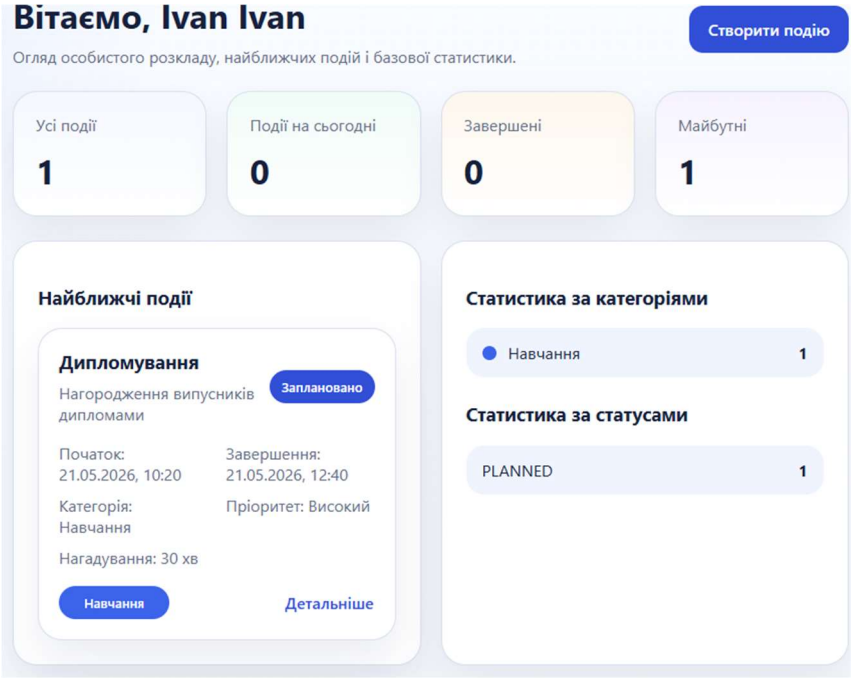


Рисунок 3.10 – Dashboard системи

Для управління станом застосунку використовуються стандартні механізми React, зокрема useState та useEffect. Дані отримуються з серверу через API та зберігаються у стані компонентів, після чого відображаються в інтерфейсі.

Такий підхід дозволяє забезпечити реактивність інтерфейсу, тобто автоматичне оновлення даних при їх зміні.

Особлива увага була приділена зручності користувацького інтерфейсу. Зокрема:

- реалізовано просту та зрозумілу навігацію;
- використано форми з валідацією;
- передбачено повідомлення про помилки;
- реалізовано адаптивний інтерфейс.

Це дозволяє зробити систему доступною для широкого кола користувачів. У результаті було реалізовано клієнтську частину інформаційної системи з використанням сучасних вебтехнологій. Застосування React, TypeScript та Axios дозволило створити зручний, швидкий та функціональний інтерфейс користувача. Реалізований frontend забезпечує повну взаємодію з серверною частиною системи та дозволяє ефективно та просто управляти подіями та особистим розкладом.

3.5 Тестування інформаційної системи планування подій та особистого розкладу

Тестування є завершальним етапом розробки інформаційної системи, який дозволяє перевірити відповідність реалізованого програмного забезпечення поставленим функціональним та нефункціональним вимогам. Основною метою тестування є виявлення можливих помилок у роботі системи, перевірка коректності обробки даних, стабільності взаємодії між клієнтською та серверною частинами, а також оцінка зручності використання застосунку.

У межах дипломної роботи тестування інформаційної системи виконувалося за кількома напрямами. Було перевірено роботу авторизації, реєстрації користувача, створення та редагування подій, видалення подій, управління категоріями, фільтрацію даних, роботу dashboard та перевірку конфліктів у розкладі. Такий підхід дозволив охопити основні сценарії використання системи та переконатися у працездатності її модулів.

Тестування проводилося вручну шляхом послідовного виконання типових дій користувача у вебінтерфейсі. Додатково перевірялися відповіді серверної частини через REST API, оскільки саме backend відповідає за обробку даних, валідацію, авторизацію та взаємодію з базою даних PostgreSQL. Під час перевірки особлива увага приділялася коректності обмеження доступу, тобто користувач повинен бачити лише власні події та категорії.

Першим етапом було перевірено запуск серверної та клієнтської частин застосунку. Серверна частина запускалася у режимі розробки, після чого виконувалася перевірка доступності endpoint-ів. Клієнтська частина запускалася окремо та взаємодіяла із сервером через REST API. У результаті було підтверджено, що обидві частини системи запускаються коректно та обмінюються даними.

Наступним етапом стало тестування модуля авторизації, наведеного на (рис. 3.11). Було перевірено створення нового користувача, його вхід до системи з правильними даними, а також обробку помилкових облікових даних. При введенні правильного email та пароля система успішно створювала JWT-токен і відкривала доступ до захищених сторінок.

Окремо було протестовано роботу з подіями. Перевірялися сценарії створення нової події, перегляду списку подій, редагування існуючої події, видалення та позначення події як виконаної. Усі дії виконувалися через інтерфейс користувача, після чого результат перевірявся у списку подій та у базі даних. При створенні події система коректно зберігала назву, опис, дату початку, дату завершення, категорію, пріоритет та нагадування. У ході перевірки цього модуля застосовувалися як позитивні, так і негативні сценарії тестування, що дозволило комплексно оцінити логіку обробки інформації в інтерфейсі та підтвердити правильність налаштування логіки обміну даними між клієнтом і сервером. Тестування роботи подій наведено нижче на (рис. 3.12).

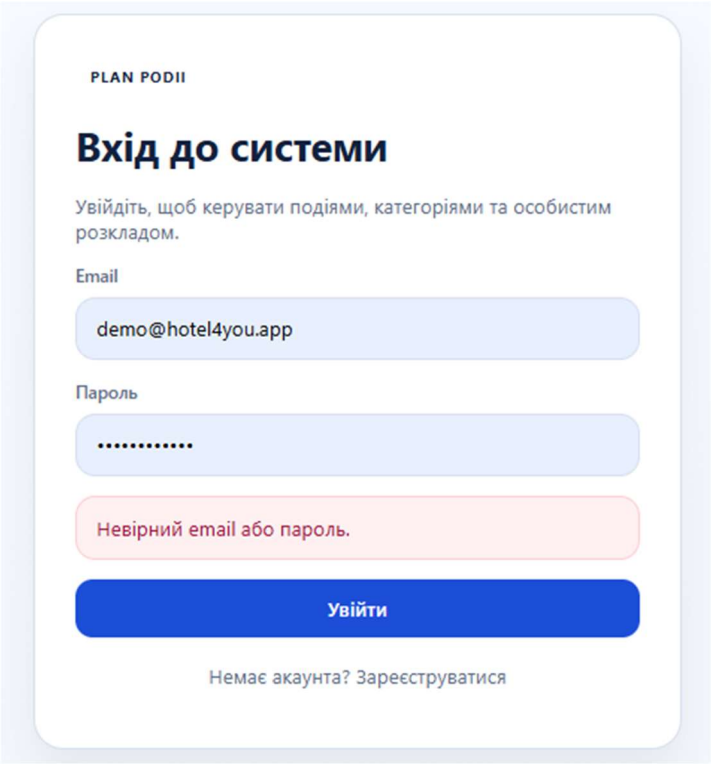


Рисунок 3.11 – Тестування сторінки авторизації користувача

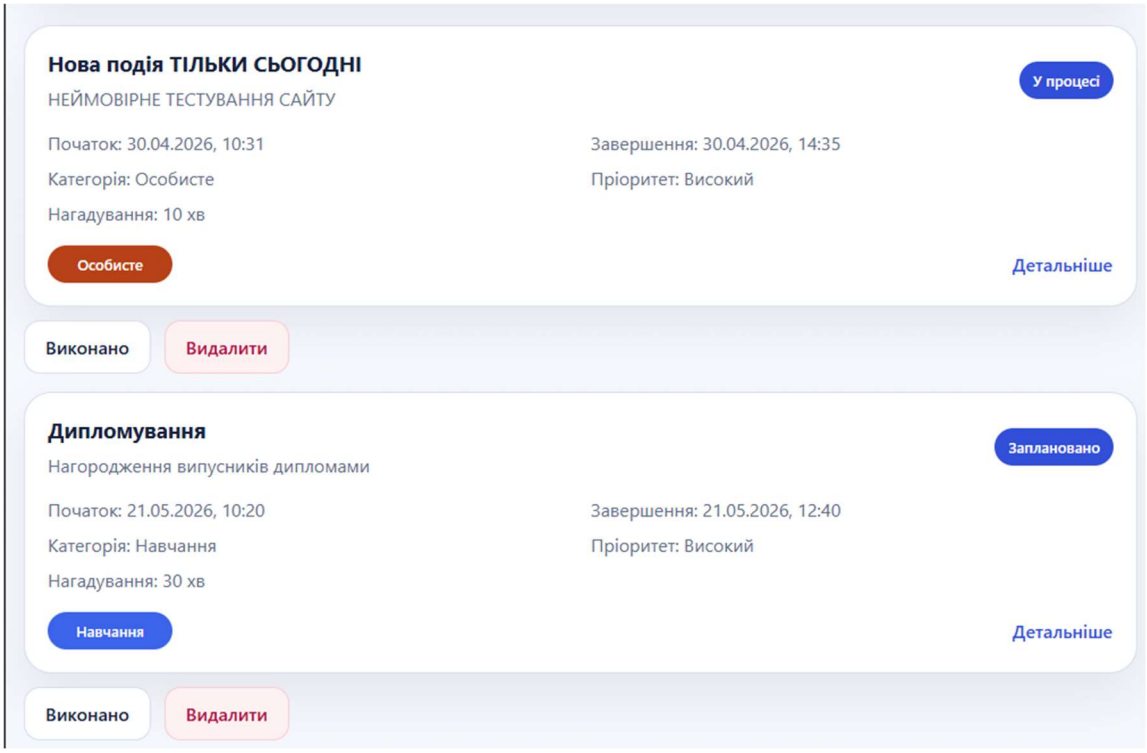


Рисунок 3.12 – Тестування створення нової події

Також було перевірено валідацію даних під час створення та редагування подій. Якщо користувач залишав обов'язкові поля порожніми або вказував некоректний часовий проміжок, система не дозволяла виконати операцію та відображала повідомлення про помилку. Це підтверджує коректну роботу механізмів перевірки даних як на клієнтській, так і на серверній стороні.

У ході тестування також перевірялася умова коректності введених даних, а саме те, що час завершення події не може бути раніше часу її початку. Приклад перевірки наведений нижче (рис. 3.13). Система успішно виконувала валідацію цих параметрів на рівні як клієнтської, так і серверної частини, що забезпечує подвійний контроль цілісності даних. У випадку виявлення логічної невідповідності часових інтервалів, спрацьовує механізм обробки виключень, і система миттєво відображає відповідне діалогове вікно з повідомленням про помилку.

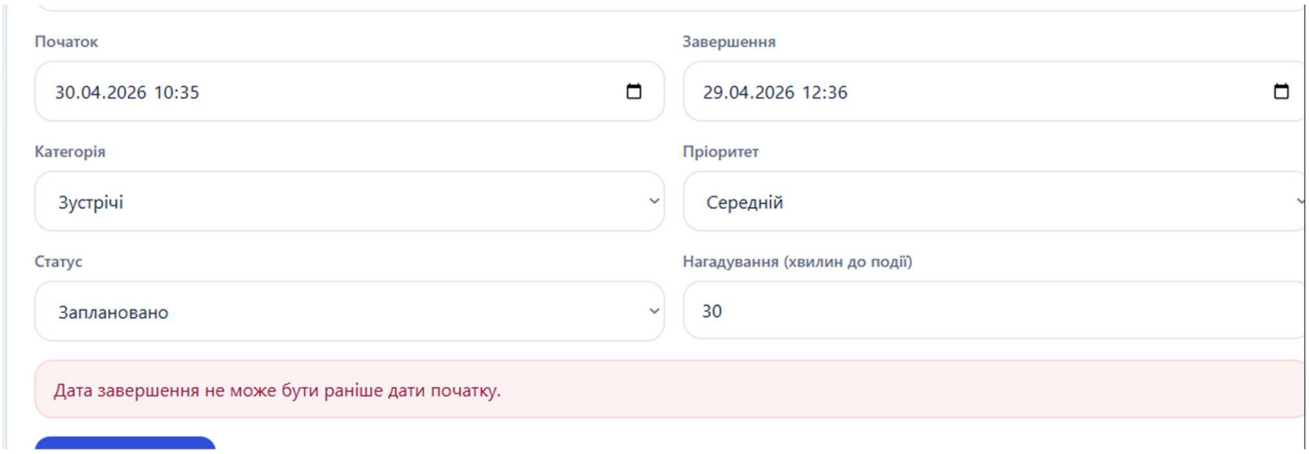


Рисунок 3.13 – Перевірка конфлікту початку події

Наступним етапом було протестовано управління категоріями (рис. 3.14). Користувач мав змогу створити нову категорію, змінити її назву або колір, а також видалити категорію. При цьому система не дозволяла видалити категорію, яка вже використовується у подіях, що забезпечує цілісність даних та запобігає втраті зв'язків між сутностями.

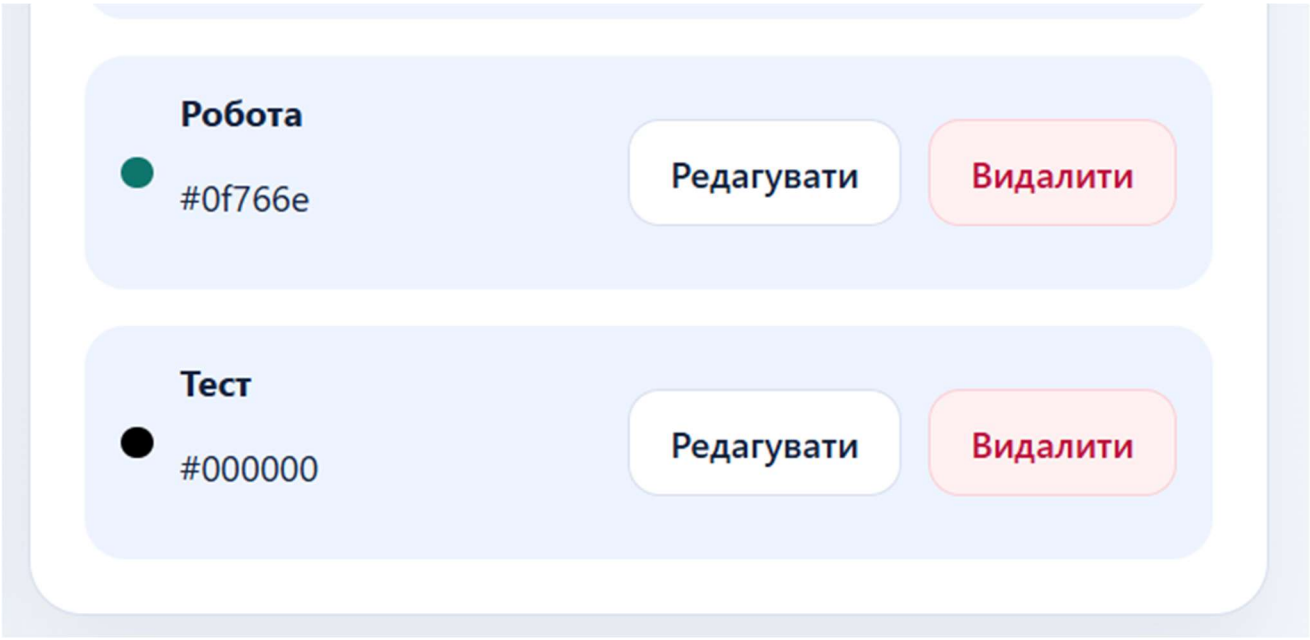


Рисунок 3.14 – Тестування управління категоріями

Окрема перевірка була виконана для dashboard, зображено нижче (рис. 3.15). Було протестовано відображення загальної кількості подій, кількості завершених та майбутніх подій. Після створення, завершення або видалення подій статистика оновлювалася відповідно до змін у базі даних.

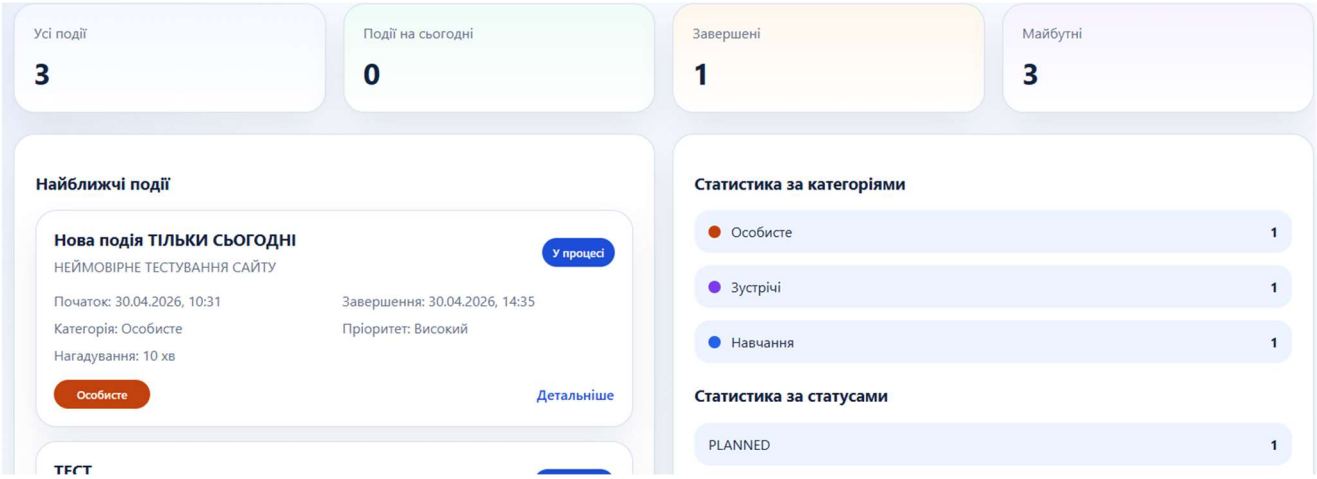


Рисунок 3.15 – Тестування dashboard системи

Для узагальнення результатів тестування було складено таблицю тестових сценаріїв. У (табл. 3.1) наведено основні функції системи, очікуваний результат та фактичний результат виконання.

Таблиця 3.1 – Результати функціонального тестування ІС

№	Тестовий сценарій	Вхідні дані / дія	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація користувача	Email, пароль, ПІБ	Користувач створений	Користувач створений, доступ надано	Успішно
2	Авторизація користувача	Коректний email і пароль	Вхід до системи	Створено JWT-токен, відкрито dashboard	Успішно
3	Авторизація з помилковим паролем	Невірний пароль	Повідомлення про помилку	Відображено повідомлення про помилку	Успішно
4	Створення події	Назва, дата, категорія	Подія збережена	Подія відображена у списку	Успішно
5	Створення події з некоректною датою	Дата завершення раніше початку	Помилка валідації	Операцію заблоковано	Успішно
6	Редагування події	Зміна назви або часу	Подія оновлена	Дані події оновлено	Успішно
7	Видалення події	Натискання кнопки видалення	Подія видалена	Подія зникла зі списку	Успішно
8	Завершення події	Позначити як виконану	Статус змінено	Подія отримала статус COMPLETED	Успішно
9	Фільтрація за категорією	Вибір категорії	Показано відповідні події	Список оновлено	Успішно

Кінець таблиці 3.1

10	Перевірка конфлікту	Події з однаковим часом	Попередження про конфлікт	Попередження відображено	Успішно
11	Створення категорії	Назва та колір	Категорія створена	Категорія з’явилась у списку	Успішно
12	Видалення категорії з подіями	Категорія використовується	Видалення заборонено	Виведено помилку	Успішно
13	Перегляд dashboard	Відкрити головну сторінку	Відображення статистики	Статистика завантажена	Успішно
14	Вихід з акаунта	Натискання кнопки виходу	Токен видалено	Користувача перенаправлено на login	Успішно

Під час тестування серверної частини окремо перевірялася робота REST API. Для цього виконувалися запити до основних endpoint-ів системи, зокрема авторизації, отримання списку подій, створення нової події, редагування та видалення записів. Така перевірка дозволила переконатися, що сервер коректно обробляє запити та повертає очікувані відповіді.

Особлива увага приділялася тестуванню захищених маршрутів. Якщо користувач не передає JWT-токен або передає некоректний токен, сервер повинен заборонити доступ до відповідних ресурсів. У результаті перевірки було встановлено, що захищені endpoint-и працюють коректно і не дозволяють неавторизованим користувачам отримувати доступ до подій, категорій або статистики.

Також було протестовано валідацію вхідних даних. Наприклад, при створенні події перевірялося, чи заповнена назва, чи правильно задані дата та час, а також чи не є час завершення ранішим за час початку. У випадку некоректних даних система повертала повідомлення про помилку, а запис не створювався у базі даних.

Окремо було перевірено роботу бази даних після виконання основних операцій. Після створення події вона з’являлася у відповідній таблиці, після редагування дані оновлювалися, а після видалення запис зникав із бази. Це підтверджує коректну взаємодію серверної частини з PostgreSQL через Prisma.

Крім того, було протестовано сценарії роботи з категоріями. Система дозволяла створювати нові категорії, редагувати їх і переглядати список. При спробі видалити категорію, яка вже використовується у подіях, система повертала помилку. Це свідчить про правильну реалізацію перевірки зв’язків між сутностями.

У результаті проведеного тестування було підтверджено, що система виконує основні функції відповідно до поставлених вимог. Виявлені під час перевірки незначні недоліки можуть бути усунені в межах подальшого вдосконалення системи, однак вони не впливають на працездатність основного функціоналу.

3.6 Висновки до розділу 3

У третьому розділі дипломної роботи було виконано практичну реалізацію інформаційної системи планування подій та особистого розкладу, а також проведено її тестування. Отримані результати підтверджують працездатність розробленого програмного забезпечення та його відповідність усім вище поставленим вимогам.

У підрозділі 3.1 було обґрунтовано вибір технологій для реалізації системи. Використання сучасного стеку, що включає TypeScript, React, Nest.js, PostgreSQL та Prisma, дозволило створити ефективну, масштабовану та зручну у підтримці систему. Застосування REST API забезпечило чітку взаємодію між клієнтською та серверною частинами, а використання JWT підвищило рівень безпеки доступу до ресурсів.

У підрозділі 3.2 було реалізовано базу даних системи. Визначено структуру основних сутностей, їх зв’язки та обмеження цілісності. Використання PostgreSQL у поєднанні з ORM Prisma дозволило забезпечити ефективну роботу з даними, підтримку складних запитів та зручне керування схемою бази даних.

У підрозділі 3.3 було реалізовано серверну частину застосунку. Створено модульну архітектуру backend із використанням Nest.js, що включає модулі авторизації, подій, категорій та статистики. Реалізовано повний набір CRUD-операцій, механізм автентифікації на основі JWT, валідацію даних та обробку помилок. Це забезпечило стабільну роботу серверної частини та коректну обробку запитів користувачів.

У підрозділі 3.4 було реалізовано клієнтську частину системи із використанням React. Створено зручний користувацький інтерфейс, який дозволяє виконувати всі основні операції з подіями та категоріями. Забезпечено інтеграцію з серверною частиною через REST API, що дозволило реалізувати повноцінну взаємодію між компонентами системи.

У підрозділі 3.5 було проведено тестування розробленого програмного забезпечення. Перевірено основні функціональні можливості системи, включаючи авторизацію, роботу з подіями, управління категоріями та формування статистики. Результати тестування показали, що система працює стабільно, коректно обробляє дані та відповідає вимогам користувача.

У третьому розділі було реалізовано повноцінну інформаційну систему планування подій, яка включає клієнтську та серверну частини, базу даних та механізми взаємодії між ними. Отриманий програмний продукт є функціонально завершеним, відповідає сучасним вимогам до вебзастосунків та може бути використаний у практичній діяльності або розширений у майбутньому.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено інформаційну систему планування подій та особистого розкладу, яка забезпечує автоматизацію процесів організації часу, підвищує ефективність управління задачами та дозволяє користувачу контролювати власну діяльність. У процесі виконання дипломної роботи було досягнуто поставленої мети, а також вирішено всі визначені задачі, що підтверджує цілісність і завершеність проведеного дослідження.

У першому розділі проведено ґрунтовний аналіз предметної області, що дозволило визначити актуальність розробки інформаційних систем планування подій у сучасних умовах цифровізації. Було розглянуто принципи організації особистого розкладу, досліджено існуючі програмні рішення, такі як календарні сервіси та системи управління задачами, а також виконано їх порівняльний аналіз. У результаті визначено їх основні недоліки, зокрема складність інтерфейсу, обмежену гнучкість або надлишковий функціонал, що не завжди відповідає потребам користувачів. Це дозволило сформувати чіткі вимоги до розроблюваної системи.

У другому розділі проведено комплексне проєктування інформаційної системи. Було обґрунтовано вибір клієнт-серверної архітектури, яка забезпечує розподіл функцій між клієнтською та серверною частинами та дозволяє досягти високої продуктивності і масштабованості. Виконано моделювання системи за допомогою UML-діаграм, що дозволило формалізувати функціональні можливості та взаємодію компонентів. Також було розроблено структуру бази даних із використанням реляційної моделі, визначено основні сутності, їх атрибути та зв’язки між ними. Додатково було сформовано алгоритми роботи усіх функцій системи, включаючи авторизацію користувача, створення подій, перевірку конфліктів та фільтрацію даних.

У третьому розділі виконано практичну реалізацію інформаційної системи із використанням сучасних вебтехнологій. Серверна частина була розроблена на основі фреймворку Nest.js із використанням REST API для взаємодії з клієнтом. Реалізовано механізм автентифікації користувачів за допомогою JWT, що забезпечує безпечний доступ до ресурсів системи. Для зберігання даних використано PostgreSQL у поєднанні з ORM Prisma, що дозволило ефективно працювати з базою даних та забезпечити цілісність інформації. Клієнтська частина реалізована з використанням React та TypeScript, що дозволило створити сучасний та зручний інтерфейс користувача. Також було проведено тестування системи, яке підтвердило коректність роботи всіх функціональних модулів.

У ході виконання роботи було отримано практичні результати у вигляді функціонуючого вебзастосунку, який реалізує основні можливості планування подій: створення, редагування, видалення подій, управління категоріями, фільтрацію даних та перегляд статистики. Особливістю розробленої системи є реалізація перевірки конфліктів у розкладі, що дозволяє уникати накладання подій та підвищує ефективність планування.

Розроблена інформаційна система має практичну цінність і може бути використана широким колом користувачів для організації особистого часу, планування навчальної або професійної діяльності. Вона також може бути адаптована для використання у корпоративному середовищі або розширена додатковими функціональними можливостями.

У результаті виконання дипломної роботи було створено сучасну інформаційну систему, яка відповідає актуальним вимогам до програмного забезпечення, має зручний інтерфейс, ефективну архітектуру та може бути використана у практичній діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ogbiti J. T., Jumai A. C. Design and implementation of an event management system. *Kasu journal of computer science*. 2024. Vol. 1, no. 4. P. 796–813. URL: https://doi.org/10.47514/kjcs/2024.1.4.007 (дата звернення: 13.02.2026).

2. Shah D. A., Vasudavan H., Razali N. F. Event management systems (EMS). *Journal of applied technology and innovation*. 2026. Vol. 7, no. 1. URL: https://doi.org/10.65136/jati.v7i1.122 (дата звернення: 13.02.2026).

3. Acharya K. Event management system project report. *SSRN electronic journal*. 2024. URL: https://doi.org/10.2139/ssrn.4846927 (дата звернення: 15.02.2026).

4. Astrology ephemeris for 9000+ years. *Astrodienst - The World's Best Horoscopes*. URL: https://www.astro.com/swisseph/swepha_r.htm (дата звернення: 15.02.2026).

5. Psion organiser 1 manual - jaap's psion organiser II page. *Jaap's Scratch Pad*. URL: https://www.jaapsch.net/psion/p1manorg.htm (дата звернення: 15.02.2026).

6. EventSync: event scheduler & coordination planner / M. Saraswat et al. *2025 IEEE international students' conference on electrical, electronics and computer science (SCEECS)*, Bhopal, India, 18–19 January 2025. 2025. P. 1–7. URL: https://doi.org/10.1109/sceecs64059.2025.10941662 (дата звернення: 16.02.2026).

7. The PalmPilot - CHM Revolution. *Home - CHM*. URL: https://www.computerhistory.org/revolution/mobile-computing/18/321 (дата звернення: 15.02.2026).

8. Bello A. H. "Eventful": revolutionizing event management through technology integration and user-centered design. *Saudi journal of engineering and technology*. 2024. Vol. 9, no. 03. P. 173–191. URL: https://doi.org/10.36348/sjet.2024.v09i03.008 (дата звернення: 16.02.2026).

9. Improving student organization: designing a digital planner app for college students / E. S. Caparos et al. *International journal of innovative science and research technology*. 2025. P. 1810–1817. URL: https://doi.org/10.38124/ijisrt/25may764 (дата звернення: 16.02.2026).

10. Personalized student AI scheduling assistant / V Abarna et al. *International research journal on advanced engineering and management (IRJAEM)*. 2025. Vol. 3, no. 03. P. 893–896. URL: https://doi.org/10.47392/irjaem.2025.0146 (дата звернення: 16.02.2026).

11. TellTime: an ai-augmented calendar with a voice interface for collecting time-use data / M. J. Hoefer et al. *IUI '25: 30th international conference on intelligent user interfaces*, Cagliari Italy. New York, NY, USA, 2025. P. 1366–1380. URL: https://doi.org/10.1145/3708359.3712116 (дата звернення: 16.02.2026).

12. Dawat event management system using progressive web application / M. Ahmed Khan et al. *Pakistan journal of engineering, technology and science*. 2024. Vol. 12, no. 1. P. 139–147. URL: https://doi.org/10.22555/pjets.v12i1.1051 (дата звернення: 16.02.2026).

13. Mohamad Zukriyani F. A., Azizan N. Student academic planner system: a review. *Malaysian journal of science health & technology*. 2023. Vol. 9, no. 1. P. 63–73. URL: https://doi.org/10.33102/mjosht.v9i1.326 (дата звернення: 18.02.2026).

14. Development of website-based sports event information system / Eva Ferdita Yuhantini et al. *World journal of advanced research and reviews*. 2025. Vol. 25, no. 3. P. 1659–1665. URL: https://doi.org/10.30574/wjarr.2025.25.3.0864 (дата звернення: 18.02.2026).

15. How to use google calendar to organize your university study. *Studwy – AI Study Planner & Pomodoro for Students*. URL: https://studwy.com/blog/how-to-use-google-calendar (дата звернення: 18.02.2026).

16. Capture, organize, and tackle your to-dos from anywhere. *Capture, organize, and tackle your to-dos from anywhere*. URL: https://trello.com (дата звернення: 18.02.2026).

17. The AI workspace that works for you. | notion. *Notion*. URL: https://www.notion.com (дата звернення: 18.02.2026).

18. Todoist | A to-do list to organize your work & life. *Todoist | A To-Do List to Organize Your Work & Life*. URL: https://www.todoist.com/ (дата звернення: 18.02.2026).

19. Geewax J. J. API design patterns. Manning Publications Co. LLC, 2021. 480 p.

20. Amundsen M. Restful web API patterns and practices cookbook: connecting and orchestrating microservices and distributed data. O'Reilly Media, Incorporated, 2022. 468 p.

21. JavaScript with syntax for types. *TypeScript: JavaScript With Syntax For Types.* URL: https://www.typescriptlang.org (дата звернення: 20.03.2026).

22. Practical SQL, 2nd edition: a beginner's guide to storytelling with data. No Starch Press, Incorporated, 2021. 392 p.

23. Richards M., Ford N. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020. 432 p.

24. Software architecture : the hard parts: modern tradeoff analysis for distributed architectures / M. Richards et al. O'Reilly Media, Incorporated, 2021. 450 p.

25. Introduction - OWASP cheat sheet series. *Introduction - OWASP Cheat Sheet Series*. URL: https://cheatsheetseries.owasp.org/index.html (дата звернення: 23.03.2026).

26. Farley D. Modern software engineering: doing what works to build better software faster. Pearson Education, Limited, 2022.

27. Typescript 4 design patterns and best practices: discover effective techniques and design patterns for every programming task. Packt Publishing, Limited, 2021.

28. Martin R. C. Clean craftsmanship: disciplines, standards, and ethics. Pearson Education, Limited, 2021.

29. Khononov V. Learning domain-driven design: aligning software architecture and business strategy. O'Reilly Media, Incorporated, 2021. 180 p.

30. Clements P., Bass L., Kazman R. Software architecture in practice. Pearson Education, Limited, 2021.

31. Powers S., Paxton J., Scott A. D. JavaScript cookbook: programming the web. O'Reilly Media, Incorporated, 2020. 650 p.

32. Greenwood R. Learn React with TypeScript - Third Edition: A beginner's guide to building real-world, production-ready web apps with React 19 and TypeScript by Carl Rippon on Ipad. ResearchHub Technologies, Inc., 2026. URL: https://doi.org/10.55277/researchhub.rc85wmkj.1 (дата звернення: 15.04.2026).

33. Vite. *vitejs*. URL: https://vite.dev (дата звернення: 26.04.2026).

34. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/en (дата звернення: 26.04.2026).

35. Documentation | NestJS - A progressive Node.js framework. *Documentation | NestJS - A progressive Node.js framework*. URL: https://docs.nestjs.com (дата звернення: 26.04.2026).

36. Roldán C. S. React 18 design patterns and best practices, 4e: design, build, and deploy production-ready web applications with react by leveraging industry-best practices. de Gruyter GmbH, Walter, 2023.

37. Prisma | serverless postgres, type-safe ORM, and database tools. *Prisma*. URL: https://www.prisma.io (дата звернення: 10.05.2026).

38. JSON web token introduction - jwt.io. *JSON Web Tokens - jwt.io*. URL: https://www.jwt.io/introduction#what-is-json-web-token (дата звернення: 10.05.2026).

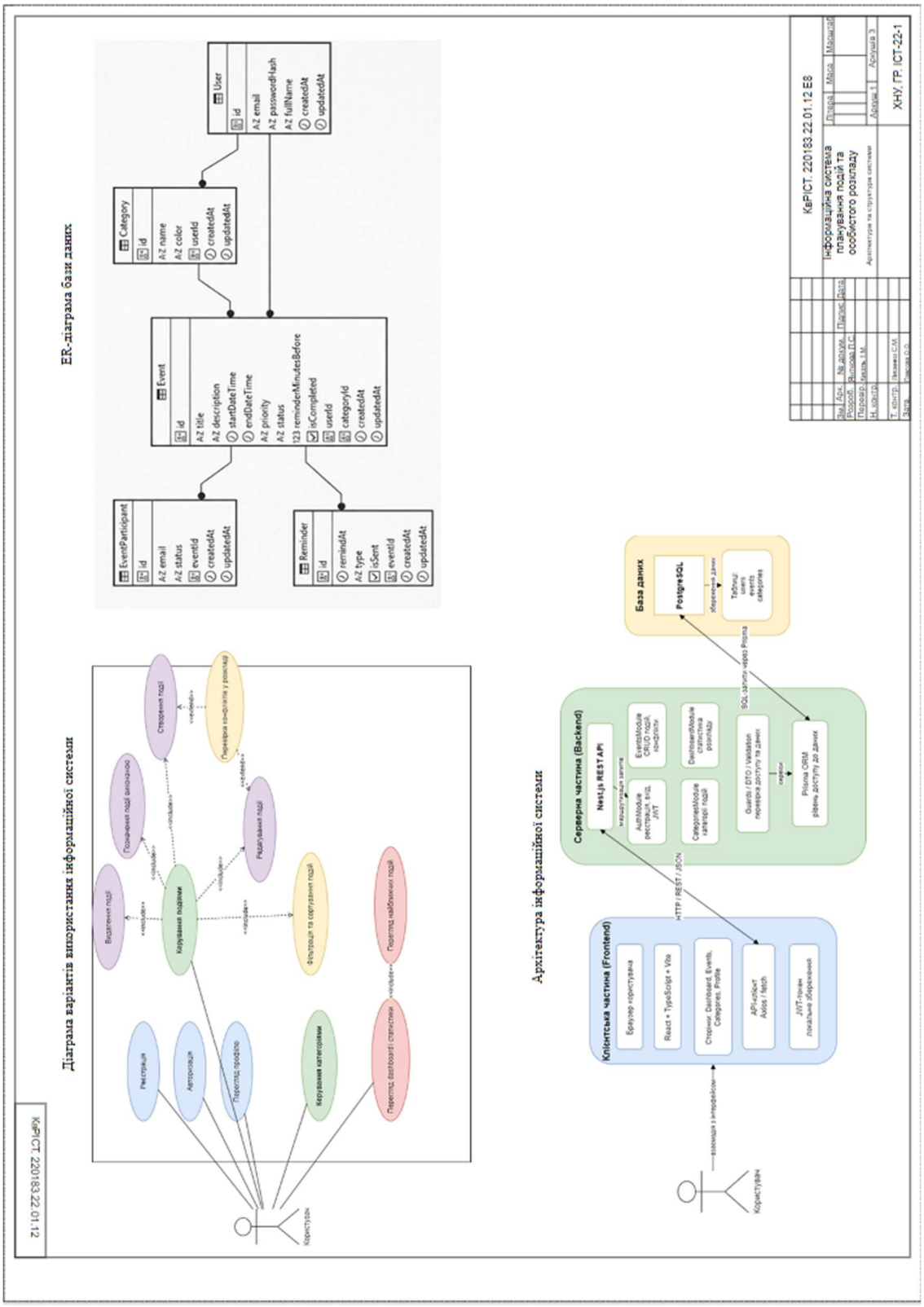
39. Axios | Promise based HTTP client. *axios | Promise based HTTP client*. URL: https://axios.rest (дата звернення: 10.05.2026).

40. React router official documentation. *React Router Official Documentation*. URL: https://reactrouter.com (дата звернення: 10.05.2026).

ДОДАТОК А

(обов’язковий)

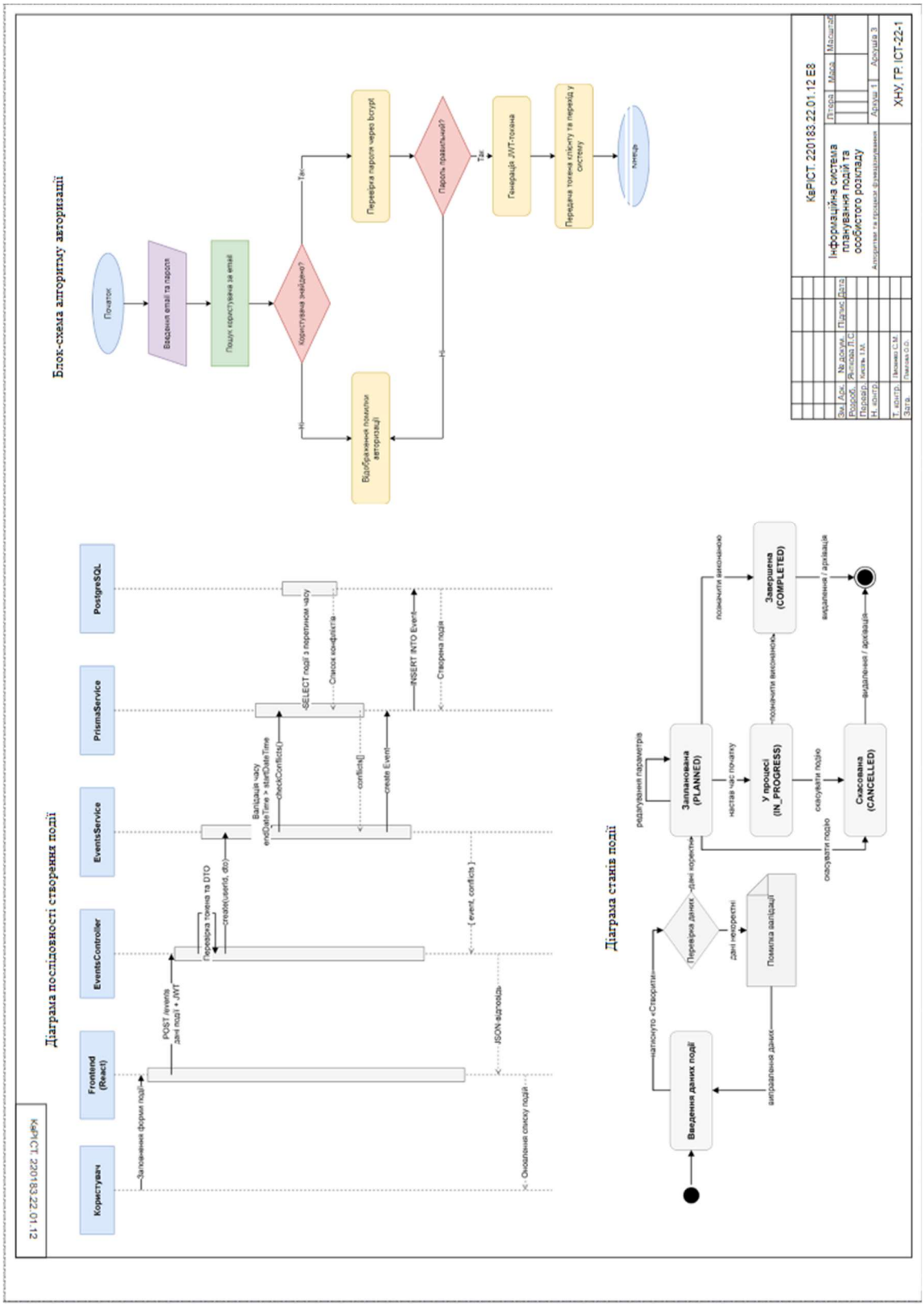
Копія креслення «Архітектура та структура системи»



ДОДАТОК Б

(обов’язковий)

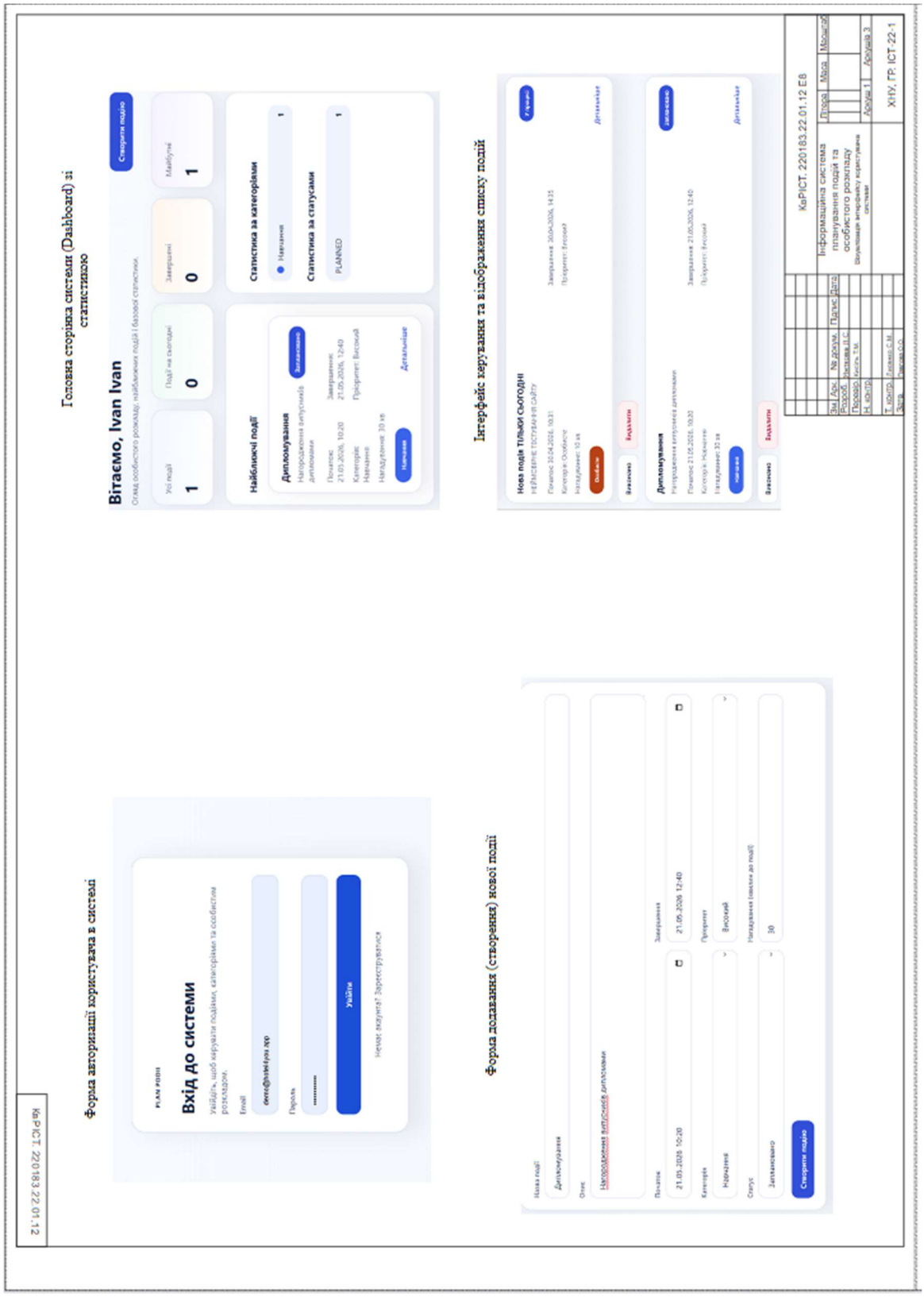
Копія креслення «Алгоритми та процеси функціонування»



ДОДАТОК В

(обов’язковий)

Копія креслення «Візуалізація інтерфейсу користувача системи»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Людмила ЯНТКОВА

Співавтор:

Назва: Інформаційна система планування подій та особистого розкладу

Експерт: Тетяна КИСІЛЬ

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:4.38%

Коефіцієнт подібності 2:0.38%

Мікропробіли: 162

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-27 20:58:38.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-28 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 12.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

ID: 272544 Назва: БКР Інформаційна система планування подій та особистого розкладу Додано в БД: 2026-05-28 Автора: Людмила ЯНТКОВА Керівники: Тетяна КИСІЛЬ Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	83104	702	11425 (14%)	99 (14%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269538	Назва: Звіт з ПДП Інформаційна система планування подій та особистого розкладу Додано в БД: 2026-02-26 Автора: Янткова Л. С. Керівники: Павлова О.О. Консультанти: Опоненти:	10059 (12.0%)	83 (12.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Янткова Людмила Сергіївна

Тема: Інформаційна система планування подій та особистого розкладу

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___65___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування та реалізація інформаційної системи планування подій та особистого розкладу.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У ході виконання першого розділу було здійснено комплексний аналіз предметної області, пов'язаної з плануванням подій та організацією особистого часу. Розглянуто етапи розвитку засобів планування – від традиційних паперових щоденників до сучасних цифрових рішень, що дозволило визначити основні тенденції розвитку даних систем. У другому розділі дипломної роботи було здійснено комплексне проєктування інформаційної системи планування подій та особистого розкладу, що є важливим етапом перед її програмною реалізацією. Було сформовано повну проєктну модель інформаційної системи, яка включає архітектурні рішення, структуру даних та алгоритми функціонування. Отримані результати є основою для подальшої реалізації програмного продукту. У третьому розділі дипломної роботи було виконано практичну реалізацію інформаційної системи планування подій та особистого розкладу. Було реалізовано повноцінну інформаційну систему, яка включає клієнтську та серверну частини, базу даних та механізми взаємодії між ними, а також проведено її тестування.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: —

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на високому технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно (А / 93)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Яшина О.М. доцент кафедри УТЗ

_____

"___" _____2026 р. _____(підпис)

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Людмила ЯНТКОВА

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи ІСТ-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Інформаційна система планування подій та особистого розкладу
Автор Людмила ЯНТКОВА
Освітня програма Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 Інформаційні системи та технології
Науковий керівник: к.ф-м.н., доцент Тетяна КИСІЛЬ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 4.38%; та системою Anti-Plagiarism складає 12.0%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис Ольга ПАВЛОВА Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Підпис Єлизавета ГНАТЧУК Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Підпис Тетяна КИСІЛЬ Ім'я, ПРІЗВИЩЕ