МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

_____

к.т.н, доц. Михелєв І.Л.

"22" червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: **Розробка веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу**

Виконав: студент групи 4144

_____ *Саєнко Д.С.*
(підпис)

Керівник роботи:

к.т.н., доцент
(посада, науковий ступінь вчене звання)

_____ *Гайдаєнко О.В.*
(підпис)

м. Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 126 "Інформаційні системи та технології"

Освітня програма " Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

_____

к.т.н, доц. Книрік Н.Р.

"25" березня 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття ступеня вищої освіти "бакалавр"

Студенту Саєнко Дмитру Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу

Керівник роботи к.т.н., доцент Гайдаєнко О.В.

Затверджені наказом ректора № 270-уч від "31" березня 2026 року.

2. Термін подання роботи: 22 червня 2026 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем.

4. Перелік питань, що належать до розробки (найменування розділів) Аналіз предметної області та постановка задачі. Проєктування веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу. Розробка веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу. Охорона праці.

5. Перелік презентаційних матеріалів Актуальність, концепція системи, Порівняльний аналіз систем, Діаграма Use Case, Діаграма IDEF0, ER-діаграми інформаційної бази даних, Структура технічних засобів інформаційної системи, Екранні форми розробки, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайдаєнко О.В.	02.02.2026	23.02.2026
2	Гайдаєнко О.В.	23.02.2026	20.03.2026
3	Гайдаєнко О.В.	20.03.2026	24.04.2026
4	Гайдаєнко О.В.	24.04.2026	01.06.2026

7. Дата видачі завдання 02 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	02.02.2026	
2	Постановка задачі	16.02.2026	
3	Розробка концепції	23.02.2026	
4	Розробка проєкту ІС	23.03.2026	
5	Реалізація проєкту	27.04.2026	
6	Розробка питань з охорони праці	01.06.2026	
7	Розробка графічних матеріалів	05.06.2026	
8	Подання роботи на перевірку на плагіат	08.06.2026	
9	Подання роботи рецензенту	15.06.2026	
10	Подання роботи на захист	22.06.2026	

Студент _____ Саєнко Д.С.
(підпис) (ПІБ)

Керівник роботи _____ Гайдаєнко О.В.
(підпис) (ПІБ)

АНОТАЦІЯ

У дипломній роботі розглянуто процес розроблення веб-застосунку на основі сучасних підходів UX/UI-дизайну користувацького інтерфейсу. Актуальність теми зумовлена зростанням вимог до зручності, доступності та ефективності веб-систем, що використовуються у різних сферах діяльності. Основна увага приділена аналізу предметної області, визначенню проблем існуючих веб-рішень та формуванню концепції системи, орієнтованої на кінцевого користувача.

У роботі виконано проєктування архітектури веб-застосунку, побудовано UML-діаграми варіантів використання, послідовності та розгортання. Також спроєктовано базу даних, розроблено концептуальну та логічну моделі, визначено основні сутності та їх взаємозв'язки. Обґрунтовано вибір програмного та апаратного забезпечення для реалізації проєкту, описано вимоги до безпеки та надійності системи.

Результатом роботи є створення прототипу веб-застосунку з адаптивним інтерфейсом, що відповідає принципам UX/UI-дизайну та забезпечує ефективну взаємодію користувача з системою. Отримані результати можуть бути використані як основа для розроблення реальних веб-продуктів, а також для подальших досліджень у сфері веб-розробки та проєктування інтерфейсів.

Робота складається з 68 сторінок машинописного тексту, 13 рисунків, 8 таблиць, 10 літературних джерел та трьох додатків.

Ключові слова: веб-застосунок, UX/UI-дизайн, користувацький інтерфейс, прототипування, веб-розробка, база даних, UML-діаграми, архітектура системи, адаптивний дизайн, користувацький досвід.

ABSTRACT

This thesis examines the process of developing a web application based on modern UX/UI design approaches for user interface development. The relevance of the topic is determined by the growing requirements for usability, accessibility, and efficiency of web systems used in various fields. Special attention is paid to the analysis of the subject area, identification of existing web solution problems, and the formation of a user-centered system concept.

The thesis includes the design of the web application architecture and the development of UML diagrams, including use case, sequence, and deployment diagrams. A database structure was also designed, including conceptual and logical models, with the definition of key entities and their relationships. The selection of software and hardware requirements for system implementation is justified, and the main requirements for security and reliability are described.

As a result of the work, a prototype web application with an adaptive interface was developed. The system follows UX/UI design principles and ensures effective user interaction. The obtained results can be used as a basis for developing real web products as well as for further research in the field of web development and interface design.

The work consists of 68 pages of typewritten text, 13 figures, 8 tables, 10 literary sources, and three appendices.

Keywords: web application, UX/UI design, user interface, prototyping, web development, database, UML diagrams, system architecture, responsive design, user experience.

ВСТУП

У сучасних умовах цифровізації суспільства веб-застосунки стали невід'ємною частиною повсякденного життя та професійної діяльності. Вони активно використовуються у сфері освіти, бізнесу, державного управління, медицини, торгівлі та багатьох інших галузях. Високий рівень конкуренції серед програмних продуктів спричиняє підвищення вимог не лише до функціональності веб-застосунків, але й до якості їх користувацького інтерфейсу. Від того, наскільки зручним, зрозумілим та привабливим є інтерфейс, залежить швидкість освоєння системи, ефективність роботи користувача та загальне сприйняття продукту.

Одним із ключових чинників успішності веб-застосунку є застосування сучасних підходів UX/UI-дизайну. UX User Experience визначає користувацький досвід взаємодії з системою, включаючи логіку навігації, доступність, зручність виконання операцій і рівень задоволеності користувача. UI User Interface охоплює візуальне оформлення інтерфейсу, структуру сторінок, компонування елементів, кольорову палітру та типографіку. Поєднання UX та UI підходів дозволяє створювати веб-застосунки, які є не лише функціональними, але й інтуїтивно зрозумілими та ефективними у використанні.

Актуальність даної дипломної роботи зумовлена тим, що значна частина сучасних веб-систем має недоліки, пов'язані з перевантаженістю інтерфейсу, складною структурою меню, відсутністю адаптивного дизайну, недостатнім рівнем доступності для різних категорій користувачів та неефективною взаємодією з інформаційними компонентами. Такі проблеми негативно впливають на зручність користування системою, збільшують кількість помилок і можуть призводити до зниження продуктивності користувача. Тому розроблення веб-застосунку з орієнтацією на UX/UI-дизайн є важливим завданням, що дозволяє підвищити якість програмного продукту та відповідність сучасним стандартам.

Метою дипломної роботи є розроблення веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу, який забезпечить зручну взаємодію користувача з системою, логічну структуру сторінок, адаптивність та доступність веб-інтерфейсу.

Для досягнення поставленої мети необхідно вирішити такі завдання проаналізувати предметну область веб-розробки та сучасні підходи UX/UI-дизайну. Визначити основні недоліки існуючих веб-застосунків та обґрунтувати необхідність удосконалення інтерфейсу. Сформувати концепцію системи та визначити її функціональні можливості. Спроєктувати архітектуру веб-застосунку та структуру бази даних. Розробити UML-діаграми для опису взаємодії компонентів системи. Створити прототипи та макети користувацького інтерфейсу. Реалізувати веб-застосунок із використанням сучасних технологій. Визначити вимоги до програмного та апаратного забезпечення. Провести тестування функціональності та оцінювання зручності використання.

Об’єктом дослідження є процес розроблення веб-застосунків як програмних систем, що забезпечують взаємодію користувача з інформаційним середовищем.

Предметом дослідження є методи та засоби UX/UI-дизайну, що застосовуються для проєктування користувацького інтерфейсу та підвищення ефективності взаємодії користувача з веб-застосунком.

Практичне значення роботи полягає у створенні веб-застосунку з сучасним інтерфейсом, який може бути використаний як навчальний або демонстраційний проєкт, а також як основа для подальшого вдосконалення та впровадження у реальні інформаційні системи.

Таким чином, виконання дипломної роботи спрямоване на розроблення веб-застосунку, який відповідає сучасним вимогам до UX/UI-дизайну та забезпечує високу якість користувацького досвіду, що є важливим чинником ефективності та конкурентоспроможності програмного продукту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

У сучасних умовах розвитку цифрових технологій веб-застосунки стали одним із основних інструментів взаємодії користувачів з інформаційними сервісами. Практично кожна сфера діяльності — освіта, бізнес, медицина, державне управління, торгівля та розваги — використовує веб-рішення для автоматизації процесів, надання послуг і підвищення ефективності роботи. При цьому зростає роль не лише технічної реалізації програмного продукту, а й його зручності, доступності та відповідності очікуванням користувачів. Саме тому UX/UI-дизайн стає ключовим фактором успішності сучасного веб-застосунку.

Предметна область даної бакалаврської роботи охоплює процеси розроблення веб-застосунків із врахуванням принципів UX/UI-дизайну. User Experience UX— це користувацький досвід, що включає зручність взаємодії, логіку навігації, швидкість виконання дій, інтуїтивність та загальне враження користувача від роботи із системою. User Interface UI — це зовнішній вигляд інтерфейсу, графічні елементи, компонування сторінок, кольорові схеми, типографіка та візуальні засоби взаємодії. Таким чином, UX/UI-дизайн поєднує функціональність, ергономіку та естетичні характеристики інтерфейсу.

Розроблення веб-застосунку на основі UX/UI-дизайну передбачає використання підходу, орієнтованого на користувача. Це означає, що процес проєктування починається з дослідження потреб цільової аудиторії, аналізу сценаріїв використання, формування вимог до функціоналу та створення прототипів інтерфейсу. На основі прототипів проводиться тестування зручності usability testing, після чого інтерфейс удосконалюється до фінальної версії, яка реалізується засобами веб-програмування.

У межах предметної області важливу роль відіграють такі компоненти:

– користувачі веб-застосунку, які взаємодіють із системою для отримання інформації або виконання певних дій;

– інтерфейс взаємодії, який забезпечує доступ до функцій системи через веб-браузер;

– веб-технології, що використовуються для реалізації front-end та back-end частин застосунку;

– база даних, яка зберігає інформацію та забезпечує її обробку;

– сценарії взаємодії користувача з системою, які визначають логіку виконання операцій.

Основною проблемою предметної області є те, що багато веб-застосунків розробляються з акцентом лише на функціональність, без урахування зручності користування. Це призводить до складної навігації, перевантажених сторінок, неінтуїтивних форм введення даних та низької ефективності взаємодії користувача із системою. У результаті користувачі можуть відмовлятися від використання продукту навіть у разі наявності корисного функціоналу. Таким чином, забезпечення якісного UX/UI-дизайну є критично важливим етапом створення сучасного веб-застосунку.

Важливою складовою UX/UI-проєктування є дотримання принципів доступності accessibility. Веб-застосунок повинен бути зручним для користувачів із різними потребами, зокрема для людей з обмеженнями зору, слуху або моторики. Для цього використовуються стандарти Web Content Accessibility Guidelines, WCAG, адаптивний дизайн та підтримка роботи на мобільних пристроях.

Також у предметній області враховується тенденція розвитку сучасних технологій веб-розробки, зокрема використання фреймворків React, Angular, Vue, компонентного підходу, дизайн-систем Material Design, Bootstrap, а також інтеграції з API та хмарними сервісами. Це дозволяє створювати швидкі, масштабовані та інтерактивні веб-застосунки, що відповідають сучасним вимогам користувачів.

Отже, предметна область дослідження охоплює комплекс процесів проєктування та реалізації веб-застосунку з орієнтацією на UX/UI-дизайн. Її особливістю є необхідність поєднання технічних аспектів програмування із психологічними та ергономічними характеристиками взаємодії користувача з системою. Застосування UX/UI-підходів дозволяє підвищити ефективність роботи веб-застосунку, забезпечити позитивний користувацький досвід та зробити програмний продукт конкурентоспроможним.

1.2 Недоліки роботи та існуючі варіанти вирішення недоліків

У процесі розроблення веб-застосунків однією з найбільш поширених проблем є недостатня увага до UX/UI-дизайну, що негативно впливає на якість користувацького досвіду та ефективність взаємодії користувача із системою. Навіть за наявності потужного функціоналу веб-застосунок може бути малоефективним через складну навігацію, перевантажений інтерфейс або неочевидну логіку виконання операцій. Саме тому актуальним є аналіз типових недоліків сучасних веб-рішень та розгляд підходів до їх усунення.

Одним із ключових недоліків є *низький рівень зручності користування usability.* У багатьох веб-застосунках відсутня чітка структура сторінок, логіка переходів між розділами не відповідає очікуванням користувачів, а основні функції розміщені незрозуміло або приховані. Це призводить до того, що користувач витрачає більше часу на пошук потрібної інформації або виконання дій, що знижує загальну ефективність роботи із системою.

Існуючим варіантом вирішення цієї проблеми є застосування *підходу User-Centered Design UCD,* який передбачає орієнтацію на потреби користувачів на всіх етапах розробки. Зокрема, проводиться аналіз цільової аудиторії, створюються сценарії використання, формуються user stories та здійснюється тестування прототипів із реальними користувачами.

Ще одним поширеним недоліком є *перевантаженість інтерфейсу елементами та інформацією.* Деякі веб-застосунки містять надмірну кількість кнопок, полів введення та інформаційних блоків на одній сторінці, що ускладнює сприйняття та знижує швидкість виконання операцій. Такі

інтерфейси викликають візуальну втому і збільшують ймовірність помилок користувача.

Для усунення цього недоліку використовується принцип *мінімалізму та ієрархії інформації*, що реалізується через правильне групування елементів, використання вкладених меню, табів, модальних вікон та поступове розкриття інформації progressive disclosure. Також ефективним рішенням є застосування дизайн-систем та компонентного підходу, що дозволяє забезпечити стандартизований вигляд і логіку елементів.

Суттєвою проблемою сучасних веб-застосунків є *відсутність адаптивності інтерфейсу* для мобільних пристроїв. З огляду на зростання кількості користувачів, які працюють із веб-застосунками зі смартфонів або планшетів, невідповідність інтерфейсу різним розмірам екрану стає критичним недоліком. Це проявляється у некоректному відображенні елементів, складності введення даних або неможливості використання певних функцій.

Існуючими способами вирішення є застосування *адаптивного дизайну responsive design* із використанням CSS Grid/Flexbox, медіа-запитів та сучасних UI-фреймворків, таких як Bootstrap або Tailwind CSS. Додатково ефективним рішенням є проєктування інтерфейсу за принципом *Mobile First*, коли мобільна версія створюється першочергово, а потім розширюється для більших екранів.

Окрему увагу слід приділити проблемі *недостатньої доступності веб-інтерфейсів* для людей з обмеженими можливостями. Часто розробники не враховують контрастність кольорів, можливість навігації з клавіатури, відсутність альтернативного тексту для зображень або некоректну роботу з екранними читачами. Це обмежує частину користувачів у доступі до функціоналу системи.

Рішенням є використання стандартів *WCAG,* впровадження семантичної HTML-розмітки, ARIA-атрибутів та проведення тестування

доступності із використанням спеціалізованих інструментів наприклад, Lighthouse або WAVE.

Ще одним недоліком є *відсутність зворотного зв'язку між системою та користувачем*. У багатьох веб-застосунках користувач не отримує повідомлення про успішність виконання операції, помилку введення даних або стан завантаження. Це може викликати плутанину, повторні натискання кнопок або втрату даних.

Для вирішення цієї проблеми застосовуються механізми UX-підтримки, зокрема: повідомлення про помилки, підказки, сповіщення notifications, індикатори завантаження, а також підтвердження важливих дій. Це дозволяє зробити взаємодію користувача із системою більш зрозумілою та безпечною.

Також важливим недоліком є *низька швидкодія веб-застосунку*, яка може бути пов'язана з неефективною роботою frontend або backend частини, неоптимізованими зображеннями, великою кількістю запитів до сервера та відсутністю кешування. Повільне завантаження сторінок знижує рівень задоволеності користувача і може призвести до відмови від використання системи.

Вирішенням цієї проблеми є оптимізація коду, використання CDN, кешування даних, мінімізація файлів JavaScript та CSS, а також застосування сучасних технологій, таких як lazy loading, SSR server-side rendering або PWA progressive web application.

Отже, проведений аналіз показав, що основні недоліки сучасних веб-застосунків пов'язані з відсутністю орієнтації на користувача, слабкою адаптивністю, низькою доступністю, перевантаженістю інтерфейсу та недостатнім рівнем взаємодії між системою і користувачем. Для усунення цих недоліків доцільно застосовувати сучасні підходи UX/UI-проєктування, стандарти доступності, адаптивний дизайн та методи тестування зручності. Використання зазначених рішень дозволяє створити веб-застосунок, який буде ефективним, інтуїтивним і конкурентоспроможним.

1.3 Концепція системи

У сучасних умовах активного розвитку цифрових сервісів веб-застосунки є основним інструментом взаємодії користувачів з інформаційними ресурсами. Проте значна частина існуючих веб-рішень створюється з акцентом на реалізацію функціоналу, тоді як питання зручності, зрозумілості та ефективності взаємодії користувача із системою часто залишаються другорядними. Це призводить до того, що навіть якісно реалізований програмний продукт може бути складним у використанні, що знижує його популярність та ефективність. У зв'язку з цим концепція даної бакалаврської роботи полягає у розробленні веб-застосунку з використанням підходів UX/UI-дизайну, який забезпечить покращення користувацького досвіду, зручність навігації та ефективність виконання основних операцій.

Традиційно процес розробки веб-застосунку відбувається за схемою, коли першочергово формуються технічні вимоги та визначається функціонал, після чого програмісти реалізують інтерфейс на основі власного бачення або стандартних шаблонів. У такому випадку дизайн часто створюється вже після реалізації основної частини функціоналу або використовується типове оформлення без глибокого аналізу потреб користувача.

У результаті виникають такі проблеми:

- структура сторінок формується без чіткої логіки та ієрархії інформації;
- користувачеві важко знайти потрібну функцію або розділ;
- відсутня єдина стильова система, що робить інтерфейс неузгодженим;
- форма введення даних може бути складною, перевантаженою або незручною;
- інтерфейс не адаптований до мобільних пристроїв;
- відсутні механізми зворотного зв'язку (повідомлення про помилки, підтвердження дій, індикатори завантаження);

– не враховуються стандарти доступності (WCAG), що обмежує можливість використання системи для людей з особливими потребами.

Таким чином, у традиційному підході основна увага приділяється програмній реалізації, а UX/UI розглядається як другорядний елемент, що негативно впливає на загальну якість веб-застосунку.

Запропонована концепція системи передбачає побудову веб-застосунку за принципом User-Centered Design, тобто проєктування системи на основі реальних потреб користувача. Основна ідея полягає у тому, що інтерфейс та логіка взаємодії повинні бути інтуїтивно зрозумілими, а виконання основних дій — швидким і мінімально навантаженим.

Концепція вдосконалення включає такі ключові етапи:

1. Визначення цільової аудиторії та аналіз потреб користувачів. На початковому етапі проводиться аналіз потенційних користувачів, визначаються їх цілі, проблеми та сценарії використання веб-застосунку. Для цього застосовуються методи UX-дослідження: анкетування, інтерв'ю, аналіз аналогів, побудова персон user personas.

2. Формування сценаріїв використання User Flow.

Для кожної категорії користувачів визначаються типові сценарії роботи із системою. Це дозволяє встановити, які функції є ключовими та як користувач переходить між ними. Сценарії формують основу логічної структури системи.

3. Створення інформаційної архітектури.

На основі user flow визначається структура веб-застосунку: головне меню, розділи, сторінки та взаємозв'язки між ними. Основною метою є побудова зрозумілої навігації, яка дозволяє користувачеві швидко знаходити потрібну інформацію.

4. Прототипування інтерфейсу.

Розробляються wireframes (каркасні макети) та інтерактивні прототипи, які дозволяють оцінити логіку інтерфейсу ще до реалізації програмного коду.

На цьому етапі формується розміщення елементів, кнопок, форм, таблиць і панелей.

5. Розробка UI-дизайну та дизайн-системи.

На основі прототипу формується візуальна частина інтерфейсу. Створюється дизайн-система, що включає:

- єдину кольорову палітру;
- типографіку;
- правила використання кнопок, форм, іконок;
- шаблони сторінок;
- принципи розміщення елементів.

Це забезпечує узгодженість інтерфейсу та підвищує якість сприйняття веб-застосунку.

6. Адаптивність і доступність інтерфейсу.

Веб-застосунок проєктується з урахуванням роботи на різних пристроях ПК, планшети, смартфони. Використовується адаптивний дизайн та принцип Mobile First. Також враховуються стандарти WCAG: достатній контраст кольорів, можливість навігації з клавіатури, підтримка екранних читачів.

7. Реалізація веб-застосунку та інтеграція UX/UI у процес програмування.

Після затвердження дизайну реалізується frontend частина системи з використанням сучасних веб-технологій (HTML5, CSS3, JavaScript/TypeScript, React або інший фреймворк). Backend частина забезпечує роботу з базою даних, авторизацію, обробку запитів і збереження інформації. UX/UI рішення інтегруються у програмний код через компонентний підхід.

8. Тестування та вдосконалення на основі зворотного зв’язку.

Після створення прототипу проводиться тестування usability, визначаються проблемні місця та виконується оптимізація. Таким чином, система вдосконалюється на основі реальних відгуків користувачів.

Запропонована система базується на таких принципах як зручність і простота — мінімальна кількість дій для виконання операцій. Логічність навігації — зрозуміле меню та структура сторінок. Інформативність — чіткі підказки, повідомлення про помилки та підтвердження дій. Уніфікація дизайну — використання єдиної дизайн-системи. Адаптивність — коректне відображення на різних пристроях. Доступність — відповідність стандартам WCAG. Безпека даних — захист доступу, контроль прав користувачів, захищене з'єднання.

Впровадження запропонованої концепції дозволяє значно підвищити ефективність роботи веб-застосунку. Користувач отримує інтуїтивно зрозумілий інтерфейс, швидкий доступ до функцій та зменшення кількості помилок під час роботи. У свою чергу, система стає більш конкурентоспроможною, оскільки відповідає сучасним вимогам щодо зручності, доступності та адаптивності.

Отже, концепція системи полягає у переході від традиційного підходу, орієнтованого лише на функціонал, до сучасного UX/UI-підходу, який передбачає проєктування веб-застосунку з урахуванням потреб користувача, тестування прототипів та формування цілісного інтерфейсу, що забезпечує високий рівень користувацького досвіду.

1.4 Математичне забезпечення розробки

Математичне забезпечення розроблення веб-застосунку на базі UX/UI-дизайну включає сукупність методів і моделей, що застосовуються для аналізу поведінки користувачів, оцінювання ефективності інтерфейсу та прийняття рішень щодо його вдосконалення. Основною метою використання математичних методів є забезпечення обґрунтованого проєктування інтерфейсу, зменшення кількості помилок користувачів та підвищення зручності роботи із системою.

У процесі розробки застосовуються методи статистичного аналізу, які дозволяють оцінити результати опитувань, тестування прототипів і поведінкових даних користувачів. Зокрема, використовуються показники

середнього значення, відсоткового співвідношення, дисперсії та аналіз частоти виконання операцій. Це дає можливість визначити найпоширеніші сценарії взаємодії користувачів із веб-застосунком.

Для оцінки зручності інтерфейсу використовується метрика usability, що включає аналіз часу виконання операцій, кількості помилок, кількості кроків для досягнення мети та загального рівня задоволеності користувачів. Також можуть застосовуватися відомі показники оцінювання UX, зокрема System Usability Scale, SUS, який дозволяє отримати інтегральну оцінку зручності інтерфейсу на основі анкетування.

У веб-застосунку також застосовується логічне моделювання процесів та алгоритмічні методи, що забезпечують коректну роботу функціоналу: обробку форм, перевірку введених даних, побудову сценаріїв переходів між сторінками та взаємодію клієнтської і серверної частин. Важливим елементом математичного забезпечення є застосування методів валідації даних, що базуються на правилах логіки та умовних операціях.

Якість інтерфейсу можна визначити як зважену суму основних критеріїв

$$Q_{UX} = w_1 U + w_2 A + w_3 E + w_4 S$$

де:

Q_{UX} — інтегральний показник якості інтерфейсу;

U — показник зручності використання Usability;

A — показник доступності Accessibility;

E — показник ефективності виконання завдань;

S — показник задоволеності користувачів;

w_i — вагові коефіцієнти критеріїв;

при цьому:

$$\sum_{i=1}^{4} w_i = 1$$

Функція часу виконання користувацького сценарію

Загальний час взаємодії користувача із системою визначається як сума часу виконання окремих операцій:

$$T = \sum_{i=1}^{n} t_i$$

де:

T — загальний час виконання сценарію;

t_i — час виконання i-ї дії;

n — кількість дій користувача.

Метою оптимізації UX/UI є мінімізація функції

$$T \to \min$$

Імовірність успішного завершення сценарію

Ефективність взаємодії користувача з веб-застосунком можна оцінити через ймовірність успішного виконання послідовності дій

$$P = \prod_{i=1}^{n} p_i$$

де:

P — загальна ймовірність успішного завершення сценарію;

p_i — ймовірність успішного виконання окремого кроку.

Коефіцієнт когнітивного навантаження

Для оцінювання складності інтерфейсу може використовуватись співвідношення кількості елементів управління до кількості необхідних функцій

$$C = \frac{N_e}{N_f}$$

де:

C — коефіцієнт когнітивного навантаження;

N_e — кількість елементів інтерфейсу;

N_f— кількість функцій системи.

Оптимальним є

$$C \to 1$$

Формалізована модель роботи веб-застосунку

Процес взаємодії користувача із системою можна описати функцією

$$Y = f(X, U, D)$$

де:

X — множина вхідних даних;

U — множина дій користувача;

D — множина дизайнерських рішень UX/UI;

Y — результат роботи веб-застосунку.

Таким чином, математичне забезпечення розробки веб-застосунку на основі UX/UI-дизайну включає статистичні методи аналізу, показники оцінювання usability та алгоритмічні підходи до моделювання процесів взаємодії користувача із системою, що дозволяє забезпечити ефективність і якість інтерфейсу.

1.5 Постановка задачі

Розроблення веб-застосунку на базі UX/UI-дизайну є актуальним завданням, оскільки сучасні інформаційні системи повинні не лише забезпечувати необхідний функціонал, а й бути зручними, інтуїтивно зрозумілими та доступними для користувачів. Практика показує, що значна

кількість веб-застосунків має недоліки, пов’язані з перевантаженням інтерфейсу, складною навігацією, відсутністю адаптивності та недостатньою орієнтацією на користувацький досвід. Це знижує ефективність взаємодії користувачів із системою, збільшує кількість помилок і негативно впливає на загальне сприйняття продукту.

У зв’язку з цим виникає необхідність розроблення веб-застосунку, в якому основний акцент зроблено на UX/UI-проєктуванні та застосуванні сучасних підходів до створення інтерфейсів, орієнтованих на потреби користувача.

Метою даної бакалаврської роботи є підвищення ефективності роботи системи за рахунок розроблення веб-застосунку з використанням принципів UX/UI-дизайну, який забезпечить підвищення зручності взаємодії користувача із системою, оптимізацію навігації та ефективне виконання основних функцій.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області веб-розробки з орієнтацією на UX/UI-дизайн.
2. Визначити основні недоліки існуючих веб-застосунків та обґрунтувати необхідність удосконалення користувацького інтерфейсу.
3. Сформувати концепцію системи та визначити її основні функціональні можливості.
4. Визначити методи математичного забезпечення для оцінювання зручності інтерфейсу та аналізу користувацьких сценаріїв.
5. Розробити структуру веб-застосунку та логіку взаємодії користувача із системою.
6. Створити прототипи wireframes та дизайн макети інтерфейсу.
7. Реалізувати веб-застосунок із застосуванням сучасних веб-технологій.

8. Забезпечити адаптивність інтерфейсу для різних пристроїв та підтримку принципів доступності.

9. Провести тестування usability та оцінити ефективність запропонованих UX/UI-рішень.

10. Сформувати висновки щодо результатів розробки та можливостей подальшого вдосконалення системи.

Об’єктом дослідження є процес розроблення веб-застосунків як програмних систем взаємодії користувача з інформаційним середовищем.

Предметом дослідження є методи та засоби UX/UI-дизайну, що застосовуються для проєктування і реалізації веб-інтерфейсу та забезпечення якісного користувацького досвіду.

У результаті виконання роботи очікується створення функціонального веб-застосунку з сучасним адаптивним інтерфейсом, побудованим на принципах UX/UI-дизайну, який може бути використаний як основа для подальшого розширення та впровадження у практичну діяльність.

Висновки до розділу 1

У першому розділі бакалаврської роботи було проведено аналіз предметної області розроблення веб-застосунків із використанням сучасних підходів UX/UI-дизайну. Встановлено, що в умовах активного розвитку цифрових технологій зручність та інтуїтивність інтерфейсу є одним із ключових факторів успішності веб-застосунку, оскільки саме користувацький досвід визначає ефективність взаємодії людини з інформаційною системою.

У межах підрозділу 1.1 розглянуто основні поняття UX та UI, визначено їх значення у процесі проєктування веб-інтерфейсів. З’ясовано, що UX/UI-дизайн охоплює не лише зовнішній вигляд веб-сторінок, але й логіку взаємодії, структуру навігації, доступність та ергономічність системи.

У підрозділі 1.2 проаналізовано типові недоліки сучасних веб-застосунків, серед яких виділено низький рівень usability, перевантаженість інтерфейсу, відсутність адаптивності для мобільних пристроїв, недостатню

доступність для користувачів з обмеженими можливостями, а також відсутність якісного зворотного зв'язку під час виконання операцій. Визначено, що усунення цих проблем можливе шляхом застосування підходів User-Centered Design, створення дизайн-систем, впровадження стандартів WCAG та використання сучасних методів тестування зручності.

У підрозділі 1.3 сформовано концепцію веб-застосунку, яка базується на переході від традиційного підходу розробки до орієнтованого на користувача проєктування. Запропоновано використання прототипування, моделювання сценаріїв взаємодії та поетапного вдосконалення інтерфейсу на основі результатів usability-тестування. Визначено ключові принципи системи: простота, логічність, адаптивність, доступність та узгодженість інтерфейсу.

У підрозділі 1.4 розглянуто математичне забезпечення розробки, що включає застосування статистичних методів аналізу, оцінювання часу виконання завдань, кількості помилок, рівня успішності та використання інтегральних показників usability, зокрема System Usability Scale SUS. Наведені формули дозволяють здійснювати кількісну оцінку ефективності інтерфейсу та обґрунтовано приймати рішення щодо його вдосконалення.

У підрозділі 1.5 сформовано постановку задачі, визначено мету, завдання, об'єкт та предмет дослідження. З'ясовано, що основною метою роботи є створення веб-застосунку з орієнтацією на UX/UI-дизайн, який забезпечить підвищення якості взаємодії користувача із системою та покращення загального користувацького досвіду.

Таким чином, у першому розділі було сформовано теоретичну та методичну основу для подальшого проєктування і реалізації веб-застосунку, визначено ключові проблеми предметної області та обґрунтовано необхідність використання сучасних UX/UI-підходів як головного інструменту підвищення ефективності веб-інтерфейсів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКУ НА БАЗІ UX/UI-ДИЗАЙНУ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

2.1 Загальносистемні рішення

У процесі розроблення веб-застосунку на базі UX/UI-дизайну важливим етапом є формування загальносистемних рішень, які визначають архітектуру програмного продукту, основні принципи його функціонування, структуру компонентів та підходи до реалізації інтерфейсу. Загальносистемні рішення забезпечують цілісність проєкту, узгодженість функціональних модулів та можливість подальшого масштабування системи.

Основною метою загальносистемних рішень є створення такого веб-застосунку, який буде зручним у використанні, надійним, швидким, адаптивним та відповідатиме сучасним вимогам до користувацького досвіду. При цьому необхідно враховувати, що веб-застосунок повинен забезпечувати не лише привабливий зовнішній вигляд, але й логічну структуру, доступність та ефективну взаємодію з користувачем.

Для реалізації системи доцільно використати клієнт-серверну архітектуру, яка є найбільш поширеним підходом у веб-розробці. Відповідно до цієї архітектури система складається з трьох основних рівнів:

- клієнтський рівень — відповідає за відображення інтерфейсу користувача, навігацію між сторінками, обробку подій та взаємодію з сервером;
- серверний рівень — реалізує бізнес-логіку системи, обробляє запити клієнта, здійснює авторизацію та забезпечує роботу з базою даних;
- рівень даних — зберігає інформацію, необхідну для функціонування веб-застосунку.

Використання такої структури дозволяє забезпечити гнучкість системи, спростити супровід та забезпечити можливість розширення функціоналу.

Принципи UX/UI у загальносистемних рішеннях

Оскільки система орієнтована на UX/UI-дизайн, загальносистемні рішення повинні передбачати реалізацію основних принципів проєктування інтерфейсу:

- логічна структура навігації — меню, сторінки та елементи інтерфейсу повинні бути розміщені зрозуміло і послідовно;
- мінімізація кількості кроків для виконання основних операцій;
- використання єдиного стилю оформлення сторінок, кнопок, форм та елементів;
- адаптивність інтерфейсу для різних розмірів екранів;
- доступність згідно зі стандартами WCAG;
- зворотний зв'язок сповіщення, підказки, повідомлення про помилки.

Таким чином, на системному рівні необхідно передбачити використання компонентного підходу та дизайн-системи, що забезпечить однаковий вигляд інтерфейсу на всіх сторінках.

Вибір технологій реалізації

Для створення веб-застосунку пропонується використання сучасних технологій:

- Frontend HTML5, CSS3, JavaScript/TypeScript з використанням фреймворку React або Vue;
- UI-бібліотека Bootstrap, Material UI або Tailwind CSS;
- Backend Node.js Express або Python Django/FastAPI;
- API REST API з обміном даними у форматі JSON;
- Database PostgreSQL або MySQL;
- Authentication JWT-токени для забезпечення безпечного входу в систему.

Такий набір технологій забезпечує високу швидкодію, гнучкість та підтримку сучасних підходів веб-розробки.

Загальносистемні рішення також передбачають визначення користувачів системи та їх прав доступу. У межах веб-застосунку можуть бути реалізовані такі ролі:

- Гість — перегляд загальної інформації без можливості редагування;
- Користувач — доступ до основного функціоналу, робота з даними;
- Адміністратор — повний доступ до управління користувачами та налаштування системи.

Рольова модель забезпечує безпеку системи та контроль за доступом до інформації.

Для забезпечення надійної роботи веб-застосунку необхідно врахувати основні заходи інформаційної безпеки:

- використання захищеного протоколу HTTPS;
- перевірка коректності введених даних валідація на клієнті та сервері;
- захист від SQL-ін’єкцій та XSS-атак;
- обмеження доступу до API через авторизацію;
- резервне копіювання бази даних.

Важливим рішенням є застосування механізму логування подій, який дозволяє відстежувати помилки та підозрілі дії користувачів.

У межах загальносистемних рішень, рисунок 2.1, веб-застосунок повинен містити основні функціональні модулі:

- модуль авторизації та реєстрації;
- головна панель користувача dashboard;
- модуль управління даними перегляд, додавання, редагування;
- модуль налаштувань профілю;
- модуль адміністрування за потреби;
- модуль повідомлень та зворотного зв’язку.

Кожен модуль реалізується як окремий компонент, що підвищує структурованість системи та спрощує її розширення.

Для підвищення ефективності UX/UI важливо забезпечити правильну структуру сторінок. Веб-застосунок повинен мати:

- головну сторінку з коротким описом функцій;
- розділ з основними функціями користувача;
- логічно побудоване меню;
- сторінки форм введення даних із підказками;
- можливість швидкого пошуку та фільтрації інформації;
- сторінку звітів або результатів.

Важливим системним рішенням є створення єдиної дизайн-системи, яка визначає стилі кнопок, полів, таблиць, шрифтів та кольорової палітри.

Таким чином, загальносистемні рішення для розроблення веб-застосунку на основі UX/UI-дизайну передбачають використання клієнт-серверної архітектури, впровадження сучасних веб-технологій, реалізацію рольової моделі доступу та забезпечення безпеки даних. Основний акцент робиться на проєктуванні системи, орієнтованої на користувача, що забезпечує логічну навігацію, адаптивність, доступність та зручність використання веб-застосунку. Запропоновані рішення створюють основу для подальшого проєктування архітектури системи та реалізації її функціональних модулів.

2.2 Функціональні можливості системи

Діаграма варіантів використання є одним із основних інструментів UML-моделювання, який застосовується для опису функціональних можливостей системи та взаємодії користувачів із веб-застосунком. Вона дозволяє визначити основні сценарії використання системи, перелік користувачів та їхні дії у межах функціоналу. Use Case Diagram є важливим етапом проєктування, оскільки надає загальне уявлення про структуру системи та вимоги до реалізації.

У межах бакалаврської роботи, присвяченої розробленню веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу, діаграма варіантів використання, рисунок 2.1, відображає ключові можливості системи та забезпечує орієнтацію проєкту на потреби кінцевого користувача.

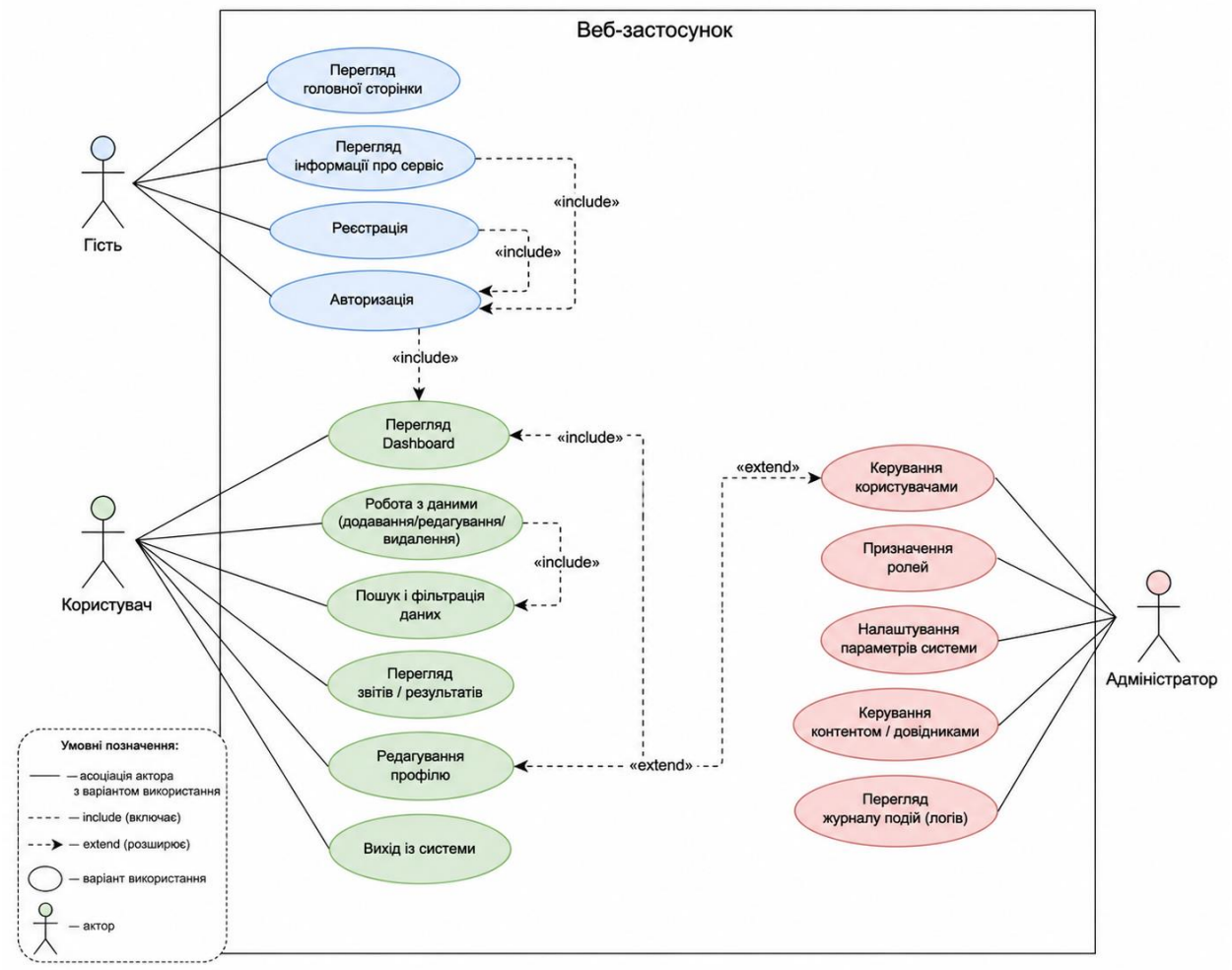

Рисунок 2.1 – Діаграма варіантів використання

У розроблюваному веб-застосунку передбачено декілька типів користувачів, кожен з яких має власні права доступу та функціональні можливості.

Основними акторами системи є:

- Гість Visitor — користувач, який не виконав авторизацію. Має доступ лише до перегляду загальної інформації та можливості реєстрації.
- Користувач User — авторизований користувач, який має доступ до основних функцій веб-застосунку.

- Адміністратор Admin — користувач з розширеними правами, який здійснює управління системою, контролює користувачів та налаштування.

На основі функціональних вимог сформовано перелік ключових варіантів використання, які відображають основні сценарії взаємодії користувачів із веб-застосунком.

Варіанти використання для актора «Гість»

Гість може виконувати такі дії:

- перегляд головної сторінки веб-застосунку;
- перегляд інформації про можливості сервісу;
- реєстрація нового користувача;
- авторизація у системі.

Варіанти використання для актора «Користувач»

Після входу у систему користувач може:

- переглядати головну панель (Dashboard);
- працювати з основними розділами веб-застосунку;
- додавати або редагувати дані;
- виконувати пошук та фільтрацію інформації;
- переглядати результати та звіти;
- редагувати власний профіль;
- виходити із системи.

Варіанти використання для актора «Адміністратор»

Адміністратор має доступ до всіх функцій користувача, а також може виконувати додаткові операції:

- керування користувачами створення, редагування, блокування;
- призначення ролей;
- налаштування параметрів системи;
- перегляд журналу подій логів;
- керування контентом або довідниками.

У діаграмі варіантів використання можуть використовуватися зв’язки типу include та extend, які дозволяють деталізувати взаємодію між сценаріями.

- Авторизація є обов’язковою умовою для виконання більшості функцій, тому вона включається у сценарії доступу до функціоналу системи.
- Пошук та фільтрація можуть бути включені у перегляд даних.
- Редагування профілю може розширювати сценарій керування користувачем.
- Керування користувачами доступне лише адміністратору.

Узагальнена структура Use Case Diagram

Таким чином, діаграма варіантів використання містить:

Актори:

- Гість
- Користувач
- Адміністратор

Use Case варіанти використання:

- Реєстрація
- Авторизація
- Перегляд головної сторінки
- Перегляд Dashboard
- Робота з даними додавання/редагування/видалення
- Пошук і фільтрація
- Перегляд звітів/результатів
- Редагування профілю
- Вихід із системи
- Керування користувачами адміністратор
- Налаштування системи адміністратор
- Перегляд журналу подій адміністратор

Діаграма варіантів використання дозволяє визначити основних користувачів веб-застосунку та їх взаємодію з системою. Вона демонструє

основний функціонал, необхідний для реалізації, та відображає розподіл прав доступу між різними ролями. Use Case Diagram є основою для подальшого проєктування архітектури веб-застосунку, реалізації модулів системи та створення ефективного UX/UI-інтерфейсу.

2.3 Моделювання взаємодії між об’єктами

Діаграма послідовності є одним із типів UML-діаграм, що використовується для моделювання взаємодії між об’єктами або компонентами системи у часі, рисунок 2.2. Вона відображає порядок передачі повідомлень між учасниками процесу та дозволяє зрозуміти логіку виконання певного сценарію роботи веб-застосунку. Діаграма послідовності є важливою частиною проєктування, оскільки дає можливість детально описати процес обробки запитів, взаємодію клієнтської та серверної частин, а також роботу з базою даних.

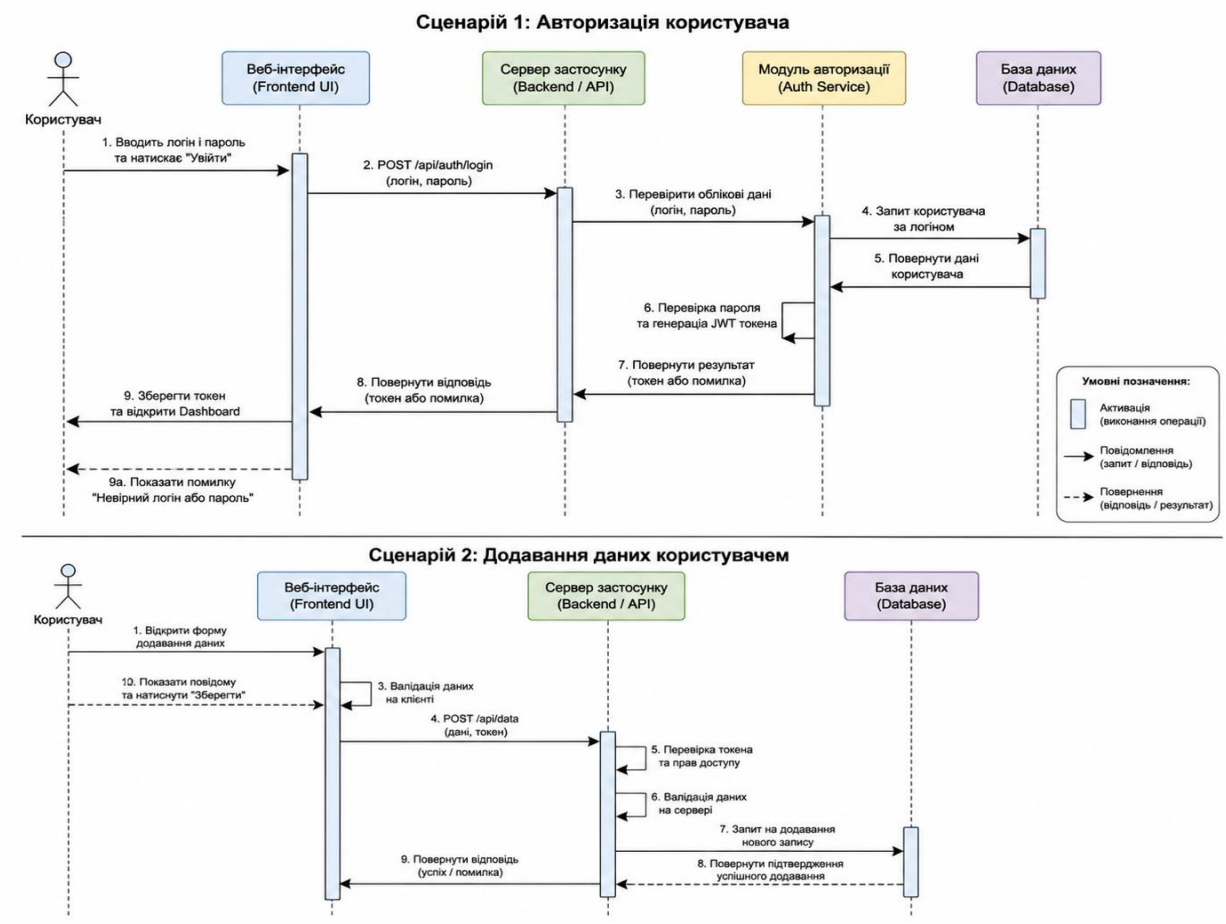

Рисунок 2.2 – Діаграма послідовності

У межах даної бакалаврської роботи, присвяченої розробці веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу, діаграма послідовності використовується для опису ключових сценаріїв функціонування системи. Одним із основних сценаріїв є процес авторизації користувача та подальший доступ до функціоналу веб-застосунку.

Основні учасники lifelines діаграми

У процесі взаємодії беруть участь такі компоненти системи:

- Користувач — ініціює виконання дій у веб-застосунку.
- Веб-інтерфейс — відображає сторінки та передає запити на сервер.
- Сервер застосунку — обробляє запити, виконує бізнес-логіку.
- База даних — зберігає інформацію про користувачів та інші дані.
- Модуль авторизації — забезпечує перевірку логіну і паролю та генерацію токена доступу.

Опис сценарію «Авторизація користувача»

Даний сценарій описує типовий процес входу користувача до веб-застосунку. Авторизація є обов’язковою умовою доступу до основного функціоналу системи.

Послідовність взаємодії:

1. Користувач відкриває сторінку входу та вводить логін і пароль.
2. Frontend формує запит авторизації та надсилає його на сервер.
3. Backend передає дані у модуль авторизації.
4. Модуль авторизації звертається до бази даних для перевірки користувача.
5. База даних повертає інформацію про користувача або повідомлення про відсутність запису.
6. Якщо користувач знайдений і пароль правильний, модуль авторизації генерує токен доступу JWT.
7. Backend повертає Frontend відповідь із токеном.
8. Frontend зберігає токен та відкриває головну сторінку Dashboard.

9. Користувач отримує доступ до функцій системи.

У випадку помилки неправильний пароль або відсутність користувача система повертає повідомлення про невдалу авторизацію та пропонує повторити спробу.

Опис сценарію «Додавання даних користувачем»

Ще одним важливим сценарієм є додавання нового запису наприклад, створення елемента у системі або введення інформації у форму. Даний сценарій демонструє роботу з інтерфейсом, сервером і базою даних.

Послідовність взаємодії:

1. Користувач відкриває форму додавання даних.
2. Вводить інформацію та натискає кнопку «Зберегти».
3. Frontend виконує первинну валідацію введених даних.
4. Frontend надсилає POST-запит на сервер із даними та токеном доступу.
5. Backend перевіряє токен та права доступу користувача.
6. Backend виконує серверну валідацію даних.
7. Backend надсилає запит до бази даних на додавання нового запису.
8. База даних зберігає інформацію та повертає підтвердження.
9. Backend повертає відповідь Frontend про успішне виконання операції.
10. Frontend відображає повідомлення «Дані успішно збережено».

У разі помилки система повертає повідомлення про некоректні дані або відсутність прав доступу.

Діаграма послідовності має важливе значення у розробці веб-застосунку з орієнтацією на UX/UI, оскільки дозволяє:

- визначити моменти, де потрібні повідомлення користувачу успіх/помилка;
- передбачити використання індикаторів завантаження loading;
- визначити логіку обробки помилок та підказок;
- забезпечити швидку та зрозумілу реакцію системи на дії користувача;
- оптимізувати кількість запитів до сервера.

Діаграма послідовності відображає логіку взаємодії користувача з основними компонентами веб-застосунку: інтерфейсом, сервером і базою даних. Вона демонструє порядок виконання операцій у часі та дозволяє деталізувати процеси авторизації, введення даних та отримання результатів. Використання Sequence Diagram забезпечує чітке розуміння роботи системи та сприяє створенню веб-застосунку, який відповідає принципам UX/UI-дизайну, є зручним, зрозумілим і ефективним для користувача.

2.4 Розробка бази даних системи

Розробка бази даних є одним із ключових етапів проєктування веб-застосунку, оскільки саме база даних забезпечує збереження, обробку та структурування інформації, необхідної для функціонування системи. Для веб-застосунку, створеного на основі UX/UI-дизайну користувацького інтерфейсу, база даних відіграє важливу роль у забезпеченні швидкого доступу до даних, коректного відображення інформації в інтерфейсі та підтримки основних функціональних можливостей системи.

У межах даної роботи база даних проєктується таким чином, щоб забезпечити зручність зберігання інформації про користувачів, їхні ролі, дані профілю, інформаційні записи, а також історію змін і взаємодій. Структура бази даних повинна бути масштабованою, підтримувати можливість розширення функціоналу системи та відповідати вимогам цілісності даних.

Для реалізації системи доцільно використовувати реляційну базу даних, оскільки вона забезпечує чітку структуру, підтримку зв’язків між таблицями та механізми забезпечення цілісності інформації. Як систему управління базами даних можна обрати PostgreSQL або MySQL, які є сучасними популярними СУБД, що підтримують роботу з великими обсягами даних, високий рівень безпеки та стабільну роботу в клієнт-серверних веб-системах.

Реляційна модель є оптимальною для веб-застосунку, оскільки дозволяє легко реалізувати основні операції CRUD та забезпечує можливість

виконання складних SQL-запитів для формування звітів і аналітики, див.рисунок 2.2.

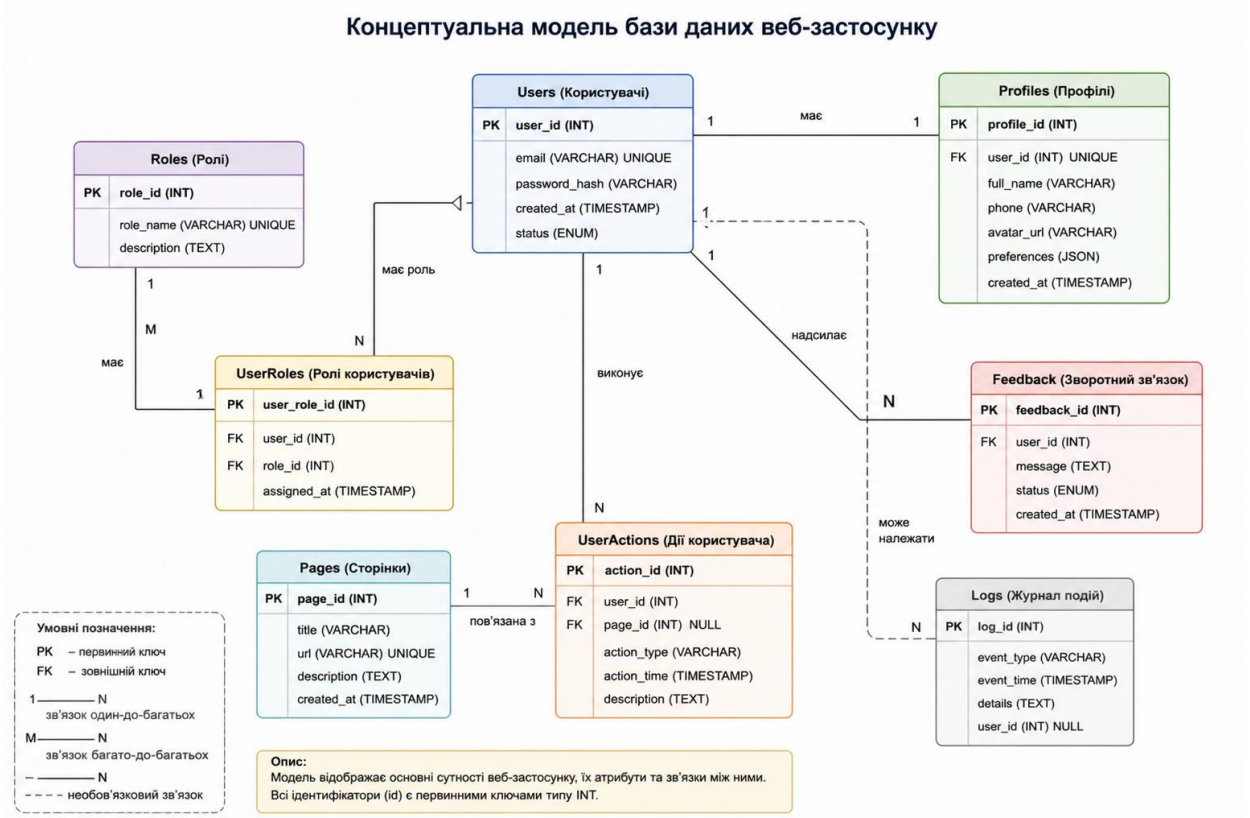

Рисунок 2.2 – Концептуальна модель

Під час розробки бази даних було визначено основні сутності таблиці, які необхідні для функціонування системи:

- Users — містить дані про зареєстрованих користувачів системи.
- Roles — визначає рольову модель доступу.
- UserRoles — таблиця зв’язку між користувачами та ролями.
- Profiles — додаткові дані користувача ім’я, контактні дані, налаштування.
- Pages — інформація про сторінки веб-застосунку для аналітики та UX.
- UserActions — збереження подій і взаємодій користувачів із системою.
- Feedback — звернення користувачів або повідомлення про помилки.
- Logs — журнал системних подій для адміністратора.

Кожна таблиця містить первинний ключ Primary Key, що забезпечує унікальність записів, а також зовнішні ключі Foreign Key для реалізації зв’язків між таблицями.

Таблиця Users призначена для зберігання облікових даних користувачів. Основними полями є:

- user_id — унікальний ідентифікатор користувача;
- email — логін користувача;
- password_hash — хеш пароля;
- created_at — дата реєстрації;
- status — активний/заблокований.

○ Таблиця Roles містить перелік можливих ролей:

- role_id — унікальний ідентифікатор ролі;
- role_name — назва ролі Admin, User, Guest.

Таблиця UserRoles реалізує зв’язок «багато до багатьох» між користувачами та ролями.

Таблиця Profiles використовується для збереження персональних даних:

- profile_id;
- user_id зв’язок з Users;
- full_name;
- phone;
- avatar_url;
- preferences налаштування користувача.

Таблиця UserActions призначена для збору інформації про взаємодію користувача із системою, що важливо для аналізу UX:

- action_id;
- user_id;
- action_type вхід, перегляд сторінки, натискання кнопки;
- action_time;

– description.

Таблиця Feedback використовується для збереження звернень:

– feedback_id;

– user_id;

– message;

– status нове, опрацьовано;

– created_at.

Таблиця Logs містить записи про системні події:

– log_id;

– event_type;

– event_time;

– details.

Проєктування бази даних здійснювалося із застосуванням принципів нормалізації, що дозволяє уникнути дублювання інформації та забезпечити цілісність даних. База даних приведена до третьої нормальної форми 3НФ, що означає кожне поле таблиці містить лише одне значення 1НФ. Усі неключові атрибути залежать від первинного ключа 2НФ. Усі неключові атрибути залежать лише від первинного ключа, без транзитивних залежностей 3НФ.

Нормалізація дозволяє забезпечити оптимальність структури та зручність підтримки бази даних у майбутньому.

Забезпечення цілісності та безпеки даних

Для забезпечення цілісності даних використовуються такі механізми:

– обмеження NOT NULL для обов’язкових полів;

– обмеження UNIQUE для email;

– використання FOREIGN KEY для реалізації зв’язків між таблицями;

– каскадне оновлення або видалення записів ON UPDATE, ON DELETE.

Безпека даних забезпечується збереженням паролів у вигляді хешу, а також розмежуванням доступу на рівні ролей користувачів.

База даних використовується серверною частиною веб-застосунку для реалізації основних функцій реєстрація та авторизація користувачів, збереження даних профілю, керування правами доступу, збереження даних про дії користувачів для UX-аналітики. Формування звітів і логування подій.

Для взаємодії із базою даних застосовується SQL або ORM-технології, Sequelize, Prisma, Django ORM, що дозволяє спростити роботу з таблицями та підвищити продуктивність розробки.

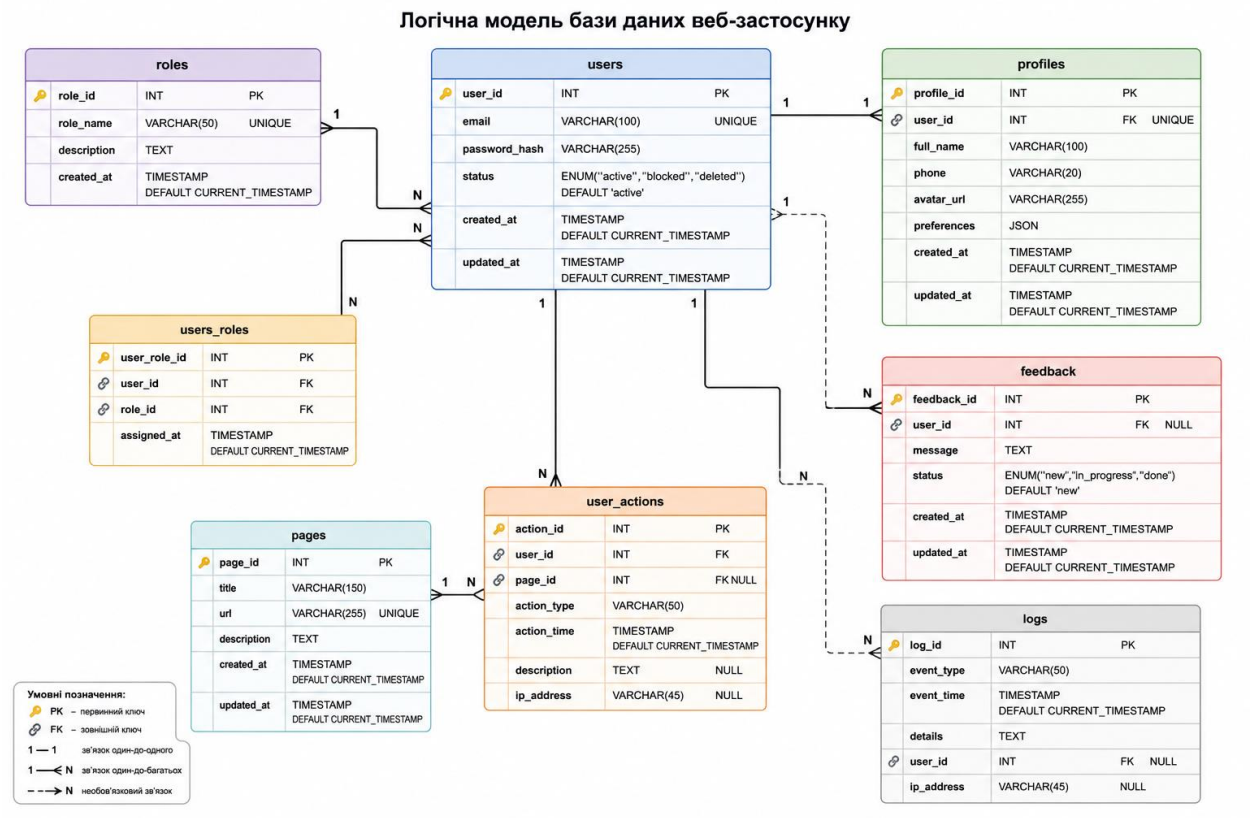

Рисунок 2.2 – Логічна модель БД

У процесі розроблення бази даних веб-застосунку було визначено основні сутності системи, сформовано структуру таблиць та їх взаємозв'язки. Запропонована база даних є реляційною, нормалізованою та відповідає вимогам цілісності й безпеки. Її використання забезпечує ефективне зберігання інформації, підтримку основних функцій веб-застосунку та можливість подальшого розширення системи відповідно до нових потреб користувачів.

Висновки до розділу 2

У другому розділі бакалаврської роботи виконано проєктування веб-застосунку на основі UX/UI-дизайну користувацького інтерфейсу. Розглянуто загальносистемні рішення, що визначають архітектуру системи, принципи її функціонування та основні підходи до реалізації програмного продукту. Обґрунтовано доцільність використання клієнт-серверної архітектури, яка забезпечує розподіл функцій між клієнтською частиною Frontend, серверною логікою Backend та базою даних.

У межах розділу визначено ключові модулі веб-застосунку, зокрема модуль авторизації та реєстрації, головну панель користувача, модуль роботи з даними, модуль зворотного зв'язку, а також модуль адміністрування. Встановлено, що впровадження принципів UX/UI-дизайну на етапі системного проєктування дозволяє забезпечити логічну структуру інтерфейсу, зручну навігацію та ефективне виконання користувачами основних операцій.

У підрозділі 2.2 було розроблено діаграму варіантів використання, яка дозволила визначити основних акторів системи гість, користувач, адміністратор та описати ключові сценарії взаємодії з веб-застосунком. Використання даної діаграми сприяло формуванню функціональних вимог і уточненню прав доступу для різних категорій користувачів.

У підрозділі 2.3 побудовано діаграму послідовності, яка відобразила логіку взаємодії користувача з основними компонентами системи у часі. Діаграма дала можливість деталізувати процеси авторизації, обробки запитів, взаємодії з базою даних та формування відповідей системи. Це дозволило визначити необхідність реалізації механізмів зворотного зв'язку, обробки помилок та індикаторів виконання операцій, що є важливими елементами UX.

У підрозділі 2.4 розроблено базу даних веб-застосунку. Визначено основні сутності системи, їх атрибути та зв'язки між ними. Побудовано концептуальну та логічну моделі бази даних, що забезпечують структуроване

зберігання інформації про користувачів, ролі, профілі, дії користувача, сторінки, звернення та журнал подій. Обґрунтовано використання реляційної моделі даних та принципів нормалізації, що сприяє зменшенню дублювання інформації та підвищенню цілісності даних.

Таким чином, у другому розділі сформовано проєктну основу для реалізації веб-застосунку: визначено архітектурні рішення, змодельовано основні сценарії взаємодії користувача із системою та розроблено структуру бази даних. Отримані результати є базою для подальшого етапу реалізації програмного продукту, розроблення інтерфейсу та тестування веб-застосунку відповідно до принципів UX/UI-дизайну.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ НА БАЗІ UX/UI-ДИЗАЙНУ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

3.1 Вимоги до програмного забезпечення розробки

Для розроблення веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу необхідно визначити вимоги до програмного забезпечення, яке буде використано під час проєктування, реалізації, тестування та супроводу системи. Вибір програмного забезпечення повинен забезпечувати можливість ефективної розробки як клієнтської, так і серверної частини застосунку, а також підтримувати сучасні інструменти UX/UI-проєктування та засоби командної роботи.

Основні вимоги до програмного забезпечення розробки включають використання операційної системи, середовищ програмування, графічних редакторів для створення дизайну, засобів керування базами даних та систем контролю версій.

Для виконання розробки веб-застосунку може використовуватись одна з таких операційних систем Windows 10/11 Linux Ubuntu 20.04+ або аналогічні дистрибутиви, macOS версії 11.0 і вище.

Операційна система повинна підтримувати встановлення сучасних веб-інструментів, середовищ розробки та серверних компонентів.

Для створення програмного коду веб-застосунку необхідно використовувати середовище розробки або редактор коду, який забезпечує підтримку веб-технологій та зручність програмування. Рекомендовані інструменти Visual Studio Code основне середовище, WebStorm альтернативний варіант.

Редактор повинен підтримувати розширення для роботи з HTML, CSS, JavaScript/TypeScript, а також інструменти для автоматичного форматування коду.

Для реалізації користувацького інтерфейсу необхідно застосовувати:

- HTML5 — для створення структури веб-сторінок;
- CSS3 — для стилізації та адаптивного дизайну;

- JavaScript або TypeScript — для реалізації інтерактивної логіки;
- React.js / Vue.js / Angular — фреймворк для створення сучасного інтерфейсу;
- Bootstrap / Tailwind CSS / Material UI — бібліотеки компонентів для пришвидшення розробки UI.

Також необхідні засоби для керування залежностями Node.js npm або yarn.

Програмні засоби для Backend-розробки, серверна частина застосунку повинна забезпечувати обробку запитів, взаємодію з базою даних, авторизацію користувачів та реалізацію бізнес-логіки. Для цього можуть бути використані Node.js + Express або Python + Django/FastAPI.

Для тестування API можуть використовуватись Postman або Insomnia.

Система управління базами даних, для збереження даних веб-застосунку необхідно використовувати реляційну СУБД PostgreSQL рекомендовано або MySQL.

Для адміністрування бази даних можуть використовуватись pgAdmin для PostgreSQL. MySQL Workbench для MySQL.

Інструменти UX/UI-дизайну, оскільки розробка базується на UX/UI-проєктуванні, необхідно використовувати програмне забезпечення для створення прототипів і дизайну Figma рекомендовано, Adobe XD альтернатива, Sketch для macOS.

Ці інструменти дозволяють створювати wireframes, інтерактивні прототипи, дизайн-систему та макети сторінок.

Засоби тестування та перевірки якості, для перевірки функціональності, швидкодії та доступності веб-застосунку використовуються Google Lighthouse аналіз продуктивності та доступності. DevTools у браузері Chrome/Firefox. ESLint, Prettier перевірка якості коду. Jest або Mocha модульне тестування.

Також доцільно використовувати інструменти usability-тестування та аналізу поведінки користувачів, наприклад Hotjar аналіз кліків і теплові карти. Google Analytics аналітика використання.

Для забезпечення збереження змін у коді та організації командної роботи необхідно використовувати Git.

Для розміщення репозиторію можуть бути використані GitHub, GitLab, Bitbucket.

Засоби розгортання та серверного середовища, для тестового або демонстраційного запуску веб-застосунку необхідно використовувати веб-сервер Nginx або Apache. Контейнеризацію Docker за потреби. Хмарні сервіси для розміщення застосунку Heroku, Render, AWS, DigitalOcean.

Для розроблення веб-застосунку на основі UX/UI-дизайну необхідний комплекс програмних засобів, що включає операційну систему, середовище програмування, сучасні веб-фреймворки, систему керування базами даних, інструменти UX/UI-проєктування, засоби тестування та систему контролю версій. Використання зазначеного програмного забезпечення забезпечує ефективну реалізацію веб-застосунку, підвищує якість інтерфейсу та сприяє створенню конкурентоспроможного програмного продукту.

3.2 Вимоги до апаратного забезпечення розробки

Для ефективного виконання процесу розроблення веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу необхідно визначити вимоги до апаратного забезпечення. Апаратні ресурси впливають на швидкість створення програмного продукту, зручність роботи з інструментами проєктування, продуктивність середовищ розробки, а також можливість тестування застосунку на різних пристроях.

Оскільки процес розробки передбачає використання графічних редакторів, середовищ програмування Visual Studio Code, запуск локального серверу, бази даних та тестування в браузерах, комп’ютер повинен мати достатній рівень продуктивності та стабільності.

Робоча станція (персональний комп’ютер або ноутбук) повинна забезпечувати комфортну роботу з інструментами веб-розробки, UI-прототипування та тестування.

Мінімальні характеристики:

- процесор: Intel Core i3 8-го покоління або AMD Ryzen 3;
- оперативна пам’ять 8 ГБ RAM;
- накопичувач SSD 128 ГБ;
- відеоадаптер інтегрований (Intel UHD / AMD Vega);
- монітор 15.6" або більше з роздільною здатністю 1366×768;
- мережевий адаптер Wi-Fi або Ethernet.

Рекомендовані характеристики:

- процесор Intel Core i5 / i7 або AMD Ryzen 5 / Ryzen 7;
- оперативна пам’ять 16 ГБ RAM або більше;
- накопичувач SSD 256–512 ГБ;
- відеоадаптер інтегрований або дискретний для комфортної роботи з графікою;
- монітор 21–24" з роздільною здатністю Full HD (1920×1080) або вище;
- стабільне підключення до Інтернету не менше 20 Мбіт/с.

Збільшений обсяг оперативної пам’яті та швидкий SSD є важливими для одночасної роботи з IDE, браузером, сервером та графічними програмами.

Для розміщення серверної частини веб-застосунку та бази даних може використовуватись окремий сервер або віртуальне середовище.

Мінімальні характеристики серверу:

- процесор 2 ядра CPU;
- оперативна пам’ять 4 ГБ RAM;
- накопичувач SSD 50–100 ГБ;
- мережеве підключення доступ до Інтернету;
- операційна система Linux Ubuntu Server.

Рекомендовані характеристики серверу:

- процесор 4 ядра CPU і більше;
- оперативна пам’ять 8–16 ГБ RAM;
- накопичувач SSD 200 ГБ і більше;
- резервне копіювання даних;
- стабільний канал зв’язку.

Для бакалаврської роботи допускається використання локального серверу на ПК або використання хмарних платформ.

Оскільки веб-застосунок повинен бути адаптивним, важливо протестувати його роботу на мобільних пристроях.

Мінімальні вимоги:

- смартфон на Android 9.0+ або iOS 13+;
- обсяг оперативної пам’яті 3 ГБ;
- наявність сучасного браузера Chrome, Safari.

Для повноцінного тестування бажано використовувати декілька пристроїв з різною діагоналлю екрану.

Для підвищення ефективності роботи розробника рекомендується використання додаткового обладнання комп’ютерна миша та клавіатура для швидкого введення даних. Другий монітор для паралельної роботи з кодом та макетами UI. Навушники або гарнітура для онлайн-консультацій та відеоконференцій. Джерело безперебійного живлення для запобігання втраті даних. Зовнішній накопичувач для резервного копіювання.

Для розроблення веб-застосунку на базі UX/UI-дизайну необхідне сучасне апаратне забезпечення, що забезпечує стабільну роботу програмних засобів проєктування, програмування та тестування. Рекомендовано використовувати робочу станцію з процесором не нижче Intel Core i5 або AMD Ryzen 5, оперативною пам’яттю від 16 ГБ та SSD-накопичувачем. Для тестування адаптивності інтерфейсу необхідні мобільні пристрої з актуальними браузерами, а для розгортання застосунку може використовуватися локальний або хмарний сервер із достатніми ресурсами для роботи бази даних і серверної частини.

3.3 Відображення фізичної структури

Діаграма розгортання є однією з UML-діаграм, яка використовується для відображення фізичної структури програмної системи, рисунок 3.1. Вона демонструє, на яких апаратних вузлах сервер, клієнтський комп'ютер, мобільний пристрій розміщуються програмні компоненти веб-застосунку, а також як здійснюється взаємодія між ними через мережеві протоколи. Діаграма розгортання є важливим елементом проєктування, оскільки дозволяє оцінити архітектуру системи, забезпечити правильне розміщення модулів і визначити канали обміну даними.

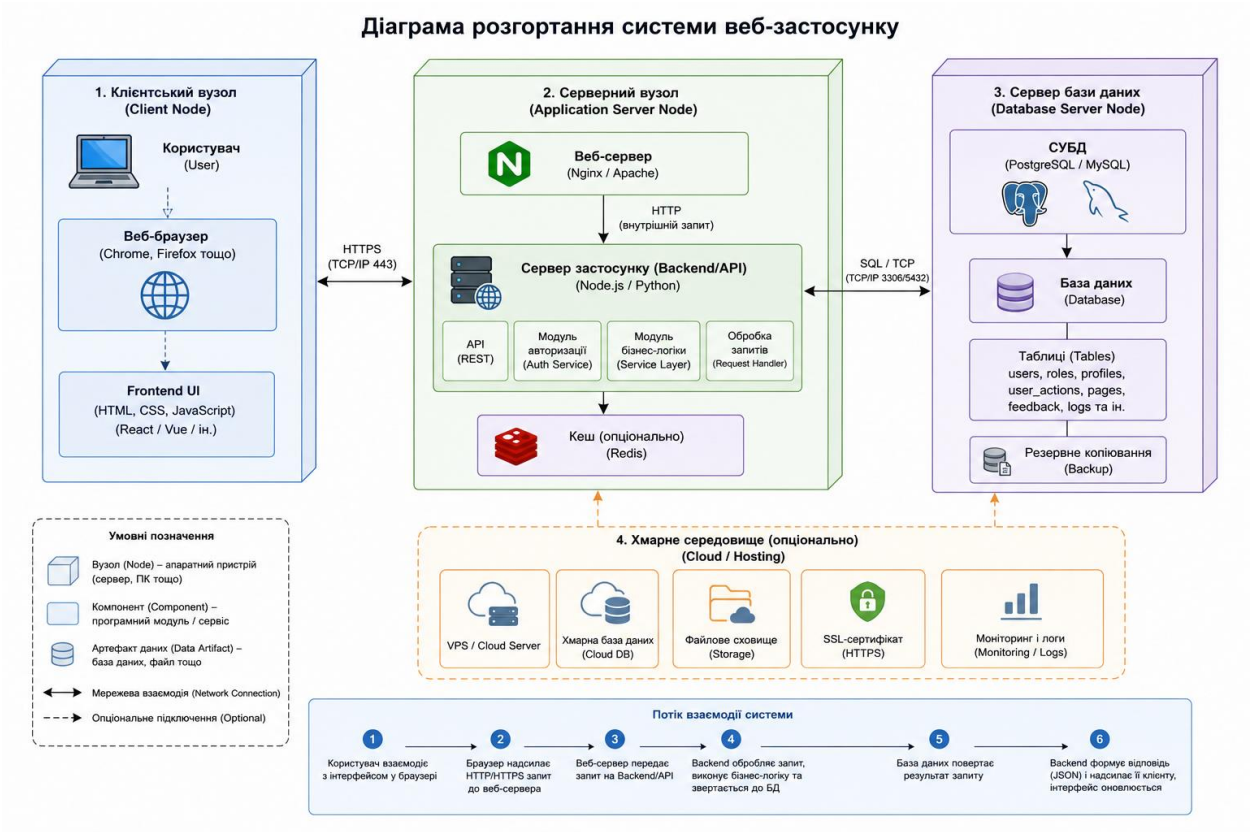

Рисунок 3.1 – Діаграма розгортання системи

У даній бакалаврській роботі веб-застосунок розробляється з використанням UX/UI-дизайну, тому система повинна забезпечувати стабільну роботу користувацького інтерфейсу, швидке завантаження сторінок та надійний доступ до даних. Для цього проєктується клієнт-серверна архітектура з використанням веб-сервера, серверної частини застосунку та бази даних.

У системі передбачено такі основні вузли:

1. Клієнтський вузол

Клієнтський вузол представлений персональним комп'ютером або ноутбуком користувача, який має встановлений веб-браузер.

На цьому вузлі розміщуються:

- веб-браузер Google Chrome, Mozilla Firefox;
- клієнтська частина застосунку Frontend UI, що виконується у браузері.

Функції клієнтського вузла:

- відображення веб-сторінок;
- взаємодія з користувачем через UI;
- надсилання HTTP/HTTPS запитів на сервер;
- отримання відповідей у форматі JSON через API.

2. Серверний вузол Application Server Node

Серверний вузол є центральним компонентом системи та забезпечує виконання бізнес-логіки веб-застосунку.

На сервері розміщуються:

- веб-сервер Nginx або Apache;
- сервер застосунку Backend/API;
- модуль авторизації;
- модуль обробки запитів;
- REST API для взаємодії з клієнтською частиною.

Функції серверного вузла:

- прийом запитів від клієнта;
- перевірка авторизації та прав доступу;
- виконання операцій обробки даних;
- взаємодія з базою даних;
- повернення результатів у вигляді відповідей API.

3. Сервер бази даних

Сервер бази даних забезпечує збереження та обробку інформації веб-застосунку.

На цьому вузлі розміщується СУБД PostgreSQL або MySQL. Таблиці бази даних. Механізми резервного копіювання.

Функції вузла зберігання даних користувачів, ролей, профілів. Збереження журналу дій користувачів. Обробка SQL-запитів. Забезпечення цілісності даних.

4. Хмарний вузол опційно

У разі розгортання системи у хмарному середовищі можуть бути використані хостингові платформи або сервіси типу AWS, Azure, Google Cloud, DigitalOcean.

Хмарний вузол може включати:

- віртуальний сервер VPS;
- хмарну базу даних;
- систему резервного копіювання;
- SSL-сертифікати для HTTPS.

Взаємодія між вузлами відбувається за таким принципом:

- Клієнтський браузер звертається до веб-сервера через протокол HTTPS.
- Веб-сервер передає запити на Backend/API.
- Backend обробляє запит і надсилає SQL-запит до сервера бази даних.
- База даних повертає відповідь Backend.
- Backend формує відповідь у форматі JSON і надсилає її клієнту.
- Frontend відображає результат користувачу.

Для забезпечення безпеки всі запити повинні виконуватись через захищене з’єднання HTTPS із використанням SSL-сертифікатів.

На діаграмі розгортання відображаються основні програмні компоненти:

- Frontend UI React/Vue/HTML+CSS+JS;
- Backend/API Node.js або Python;
- Auth Service JWT авторизація;

- Database PostgreSQL/MySQL;
- Web Server Nginx/Apache.

Таке розділення компонентів забезпечує модульність системи, підвищує продуктивність та спрощує супровід веб-застосунку.

Діаграма розгортання системи демонструє фізичну структуру веб-застосунку та розміщення його компонентів на різних вузлах. Визначено, що система функціонує за клієнт-серверною архітектурою, де клієнтська частина виконується у веб-браузері, серверна логіка реалізується на сервері застосунку, а дані зберігаються на сервері бази даних. Така структура забезпечує масштабованість, надійність та можливість подальшого розширення функціоналу веб-застосунку.

Висновки до розділу 3

У третьому розділі бакалаврської роботи було визначено та обґрунтовано вимоги до програмного й апаратного забезпечення, необхідного для розроблення веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу. Проведений аналіз показав, що для реалізації сучасного веб-застосунку доцільно використовувати комплекс програмних засобів, які забезпечують ефективне проєктування, програмування, тестування та розгортання системи.

У підрозділі 3.1 було розглянуто вимоги до програмного забезпечення розробки. Встановлено, що для створення веб-застосунку необхідні сучасні інструменти frontend- та backend-розробки, система управління базами даних, засоби UX/UI-проєктування, тестування якості та система контролю версій. Обґрунтовано використання таких технологій, як HTML, CSS, JavaScript/TypeScript, сучасні фреймворки React/Vue, серверні платформи (Node.js або Python), а також СУБД PostgreSQL чи MySQL.

У підрозділі 3.2 визначено вимоги до апаратного забезпечення, яке повинно забезпечувати стабільну та продуктивну роботу середовищ розробки, браузерів, графічних редакторів і серверних компонентів. З’ясовано, що для комфортної реалізації веб-застосунку рекомендовано

використовувати робочу станцію з процесором рівня Intel Core i5 або AMD Ryzen 5, оперативною пам’яттю не менше 16 ГБ та SSD-накопичувачем, що забезпечує швидкість роботи програмного забезпечення та зменшує час виконання операцій.

У підрозділі 3.3 було розглянуто діаграму розгортання системи, яка відобразила фізичну структуру веб-застосунку та розміщення його компонентів на різних вузлах. Визначено, що система функціонує за клієнт-серверною архітектурою, де клієнтська частина працює у браузері користувача, серверна логіка розміщується на сервері застосунку, а база даних функціонує на окремому сервері або у хмарному середовищі. Така структура забезпечує масштабованість, надійність і можливість подальшого розширення функціоналу.

Отже, результати третього розділу підтвердили, що запропоновані вимоги до програмного та апаратного забезпечення є достатніми для реалізації веб-застосунку з використанням сучасних UX/UI-підходів. Сформовані рішення створюють технічну основу для практичної реалізації системи та її подальшого впровадження.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1 Загальні положення

Охорона праці є важливою складовою процесу розроблення програмного забезпечення, оскільки робота програміста та UX/UI-дизайнера пов’язана з тривалим перебуванням за комп’ютером, високим зоровим навантаженням, статичним положенням тіла та психоемоційною напругою. Під час виконання бакалаврської роботи за темою «Розробка веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу» розробник працює з комп’ютерною технікою, мережевим обладнанням, програмними засобами та електронною документацією, що потребує дотримання правил безпеки та санітарно-гігієнічних норм.

Метою даного розділу є аналіз умов праці під час виконання робіт з проєктування та програмування веб-застосунку, визначення можливих небезпечних і шкідливих факторів, а також розроблення заходів щодо забезпечення безпечних умов праці.

4.2 Характеристика робочого місця розробника

Робоче місце розробника веб-застосунку являє собою робочу зону з персональним комп’ютером або ноутбуком, монітором, клавіатурою, мишею, мережевим підключенням та додатковими периферійними пристроями. Основні види робіт включають розроблення програмного коду, створення UX/UI-макетів та прототипів, тестування веб-застосунку, роботу з документацією та базами даних, підготовку звітної документації.

Робота виконується переважно в сидячому положенні та має розумовий характер. Робоче місце повинно відповідати ергономічним вимогам, забезпечувати зручність виконання операцій і мінімізувати негативний вплив на здоров’я працівника.

4.3 Аналіз шкідливих та небезпечних виробничих факторів

Під час роботи за комп’ютером на розробника можуть впливати такі шкідливі та небезпечні фактори:

Фізичні фактори електричний струм при пошкодженні ізоляції проводів або несправності обладнання. Електромагнітне випромінювання від монітора та системного блоку. Недостатнє або надмірне освітлення робочого місця. Підвищений рівень шуму від комп'ютерної техніки та вентиляторів. Несприятливий мікроклімат температура, вологість, вентиляція.

Психофізіологічні фактори зорове перенапруження внаслідок тривалої роботи з екраном, статичне навантаження на хребет та м'язи через тривале сидіння, перевтома кистей рук і розвиток тунельного синдрому, психоемоційне навантаження через дедлайни, складність завдань, інтенсивну розумову діяльність.

Організаційні фактори неправильна організація робочого часу, відсутність перерв, незручне розташування монітора, клавіатури, миші.

Таким чином, основні ризики пов'язані не з механічними небезпеками, а з тривалим впливом на організм людини при роботі за комп'ютером.

4.4 Вимоги до освітлення та мікроклімату

Однією з важливих умов безпечної роботи є правильне освітлення робочого місця. Робоче місце програміста повинно бути забезпечене природним та штучним освітленням. Освітленість повинна бути достатньою для роботи з текстовою інформацією та графічними елементами.

Рекомендовано розташовувати робочий стіл таким чином, щоб світло падало зліва, для правші, та не створювало відблисків на екрані монітора. Для штучного освітлення використовуються світильники з розсіяним світлом.

Оптимальні параметри мікроклімату в робочому приміщенні температура повітря 20–24 °C, відносна вологість 40–60%, швидкість руху повітря до 0,1–0,2 м/с.

Дотримання зазначених параметрів сприяє підвищенню працездатності та зменшенню втоми.

4.5 Ергономічні вимоги до робочого місця

Для запобігання професійним захворюванням важливо забезпечити ергономічну організацію робочого місця. Основні вимоги висота робочого столу повинна забезпечувати правильне положення рук. Стілець має бути регульованим по висоті та мати спинку. Монітор повинен знаходитися на відстані 50–70 см від очей. Верхній край екрану має бути на рівні очей або трохи нижче. Клавіатура розташовується на відстані 10–30 см від краю столу. Необхідно використовувати підставку для ніг у разі потреби.

Правильне розташування обладнання знижує навантаження на зір, хребет і суглоби.

4.6 Режим праці та відпочинку

Для зменшення негативного впливу комп'ютерної роботи необхідно дотримуватися оптимального режиму праці та відпочинку. Рекомендовано робити перерву 10–15 хвилин кожні 1–2 години роботи. Виконувати вправи для очей переведення погляду на віддалені предмети. Робити коротку фізичну розминку для шиї, спини та рук. Чергувати види діяльності програмування, тестування, робота з документацією.

Дотримання режиму дозволяє уникнути перенапруження, підвищити концентрацію та продуктивність праці.

4.7 Електробезпека та пожежна безпека

Робота з комп'ютерною технікою передбачає дотримання вимог електробезпеки. Основні правила використання справного електрообладнання та кабелів. Заборона роботи з пошкодженими розетками або проводами. Недопущення перевантаження електромережі. Вимкнення обладнання після завершення роботи. Використання джерел безперебійного живлення UPS для захисту техніки.

Для забезпечення пожежної безпеки необхідно не використовувати несправні подовжувачі, не розміщувати легкозаймисті матеріали біля електрообладнання, забезпечити наявність вогнегасника у приміщенні,

дотримуватися правил евакуації та знати місцезнаходження аварійних виходів.

4.8 Заходи щодо зниження ризиків та забезпечення безпеки праці

Для мінімізації шкідливих факторів під час розроблення веб-застосунку необхідно впровадити такі заходи забезпечення ергономічного робочого місця регульований стілець, правильне розміщення монітора. Використання сучасного монітора з високою роздільною здатністю та захистом від мерехтіння. Контроль освітлення та уникнення відблисків на екрані. Підтримка оптимального мікроклімату вентиляція, кондиціонування. Дотримання режиму праці та відпочинку, проведення регулярного технічного обслуговування комп'ютерної техніки. Використання антивірусного програмного забезпечення та захисту інформації, організація резервного копіювання даних.

Висновки до розділу 4

У даному розділі розглянуто основні аспекти охорони праці під час виконання робіт із розроблення веб-застосунку на основі UX/UI-дизайну. Визначено, що головними шкідливими факторами є зорове перенапруження, статичне навантаження на опорно-руховий апарат, психоемоційна втома, а також потенційні ризики, пов'язані з електробезпекою та пожежною безпекою. Запропоновані заходи щодо організації робочого місця, дотримання режиму праці та забезпечення безпечних умов дозволяють знизити ризики для здоров'я та забезпечити ефективну роботу розробника.

ВИСНОВКИ

У ході виконання бакалаврської роботи на тему «Розробка веб-застосунку на базі UX/UI-дизайну користувацького інтерфейсу» було досліджено та проаналізовано сучасні підходи до створення веб-застосунків, орієнтованих на потреби користувача. Встановлено, що якість UX/UI-дизайну є одним із ключових чинників ефективності програмного продукту, оскільки саме зручність інтерфейсу та логіка взаємодії визначають швидкість виконання операцій, кількість помилок та загальний рівень задоволеності користувачів.

У першому розділі роботи проведено аналіз предметної області, визначено основні проблеми сучасних веб-застосунків, серед яких виділено складну навігацію, перевантаженість інтерфейсу, відсутність адаптивності та недостатню доступність для користувачів. Також сформовано концепцію системи, яка передбачає застосування принципів User-Centered Design, прототипування та тестування зручності. Окремо розглянуто математичне забезпечення, яке дозволяє оцінювати ефективність інтерфейсу за допомогою статистичних показників, метрик usability та інтегрального показника SUS.

У другому розділі виконано проєктування системи, визначено загальносистемні рішення та архітектуру веб-застосунку. Розроблено UML-діаграми варіантів використання та послідовності, що дозволило формалізувати функціональні можливості системи та логіку взаємодії користувача з основними компонентами. Також було спроєктовано базу даних, визначено основні сутності та їх зв’язки, побудовано концептуальну і логічну моделі. Запропонована структура бази даних забезпечує надійне зберігання інформації, підтримку рольового доступу та можливість подальшого розширення функціоналу системи.

У третьому розділі сформовано вимоги до програмного та апаратного забезпечення, необхідного для реалізації веб-застосунку. Визначено перелік сучасних інструментів та технологій, що забезпечують ефективний процес розробки та тестування. Побудовано діаграму розгортання системи, яка

підтвердила доцільність використання клієнт-серверної архітектури з окремим сервером застосунку та базою даних.

У четвертому розділі розглянуто питання охорони праці, визначено основні небезпечні та шкідливі фактори, що виникають під час роботи розробника за комп’ютером. Запропоновано заходи щодо забезпечення безпечних умов праці, зокрема вимоги до ергономіки робочого місця, освітлення, режиму праці та відпочинку, а також правила електро- та пожежної безпеки.

Таким чином, поставлену мету бакалаврської роботи було досягнуто: розроблено та спроєктовано веб-застосунок із використанням сучасних підходів UX/UI-дизайну, що забезпечує зручність взаємодії користувача з системою, логічну навігацію, адаптивність інтерфейсу та можливість подальшого вдосконалення. Результати роботи можуть бути використані як основа для створення реального веб-продукту або подальших досліджень у сфері веб-розробки та UX/UI-проєктування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Garrett, J. J. (2019). *The elements of user experience: User-centered design for the web and beyond* (2nd ed.). New Riders.
2. Norman, D. A. (2020). *The design of everyday things* (Revised and expanded edition). Basic Books.
3. Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability* (3rd ed.). New Riders.
4. W3C Web Accessibility Initiative. (2023). *Web Content Accessibility Guidelines (WCAG) 2.2*. World Wide Web Consortium. https://www.w3.org/TR/WCAG22/
5. Nielsen, J. (2020). *10 usability heuristics for user interface design*. Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/
6. Brooke, J. (1996). SUS: A "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189–194). Taylor & Francis.
7. Google. (2024). *Material Design 3 guidelines*. Google Developers. https://m3.material.io/
8. Mozilla Developer Network. (2024). *MDN Web Docs: HTML, CSS, and JavaScript documentation*. Mozilla. https://developer.mozilla.org/
9. Figma. (2024). *Figma design documentation and best practices*. Figma, Inc. https://www.figma.com/resources/
10. Bass, L., Clements, P., & Kazman, R. (2021). *Software architecture in practice* (4th ed.). Addison-Wesley.

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

1. Загальні відомості

1.1 Назва програмного продукту «Веб-застосунок на базі UX/UI-дизайну користувацького інтерфейсу».

1.2 Підстава для розробки «Технічне завдання» розроблено відповідно до теми бакалаврської роботи та необхідності створення сучасного веб-застосунку з інтуїтивно зрозумілим інтерфейсом, адаптивним дизайном та зручним функціоналом для користувачів.

1.3 Розробник здобувач спеціальності 126 «Інформаційно управляючі системи та технології», Саєнко Д.

1.4 Замовник Кафедра інформаційно управляючих систем та технологій.

1.5 Термін виконання травень 2026 рік.

2. Мета та призначення системи

2.1 Мета розробки

Метою розробки є створення веб-застосунку, орієнтованого на користувача, із застосуванням принципів UX/UI-дизайну, що забезпечить зручність навігації, логічність структури, доступність та підвищення ефективності взаємодії користувача з системою.

2.2 Призначення системи

Веб-застосунок призначений для демонстрації реалізації сучасного інтерфейсу з підтримкою реєстрації користувачів, авторизації, управління даними, перегляду інформації та взаємодії через веб-браузер.

3. Характеристика об’єкта автоматизації

Об’єктом автоматизації є процес взаємодії користувача з веб-системою через інтерфейс, що включає виконання типових операцій: реєстрація, авторизація, перегляд даних, пошук, додавання та редагування інформації, формування повідомлень та управління профілем.

4. Вимоги до функціональних характеристик

4.1 Основні функції системи

Система повинна забезпечувати реалізацію наступних функцій:

- реєстрація нового користувача;
- авторизація та вихід із системи;
- відновлення доступу за потреби;
- перегляд головної сторінки та інформаційних розділів;
- перегляд особистого кабінету користувача Dashboard;
- створення, редагування та видалення записів;
- пошук та фільтрація інформації;
- перегляд статистики або результатів за наявності;
- робота з профілем користувача;
- можливість надсилання звернень через форму зворотного зв’язку;
- адміністрування користувачів для ролі адміністратора.

4.2 Вимоги до ролей користувачів

У системі повинні бути реалізовані такі ролі:

- Гість – перегляд загальної інформації, доступ до реєстрації та авторизації;
- Користувач – доступ до основного функціоналу, робота з даними та профілем;
- Адміністратор – повний доступ до управління користувачами та налаштування системи.

4.3 Вимоги до введення та обробки даних

Система повинна:

- забезпечувати введення даних через форми;
- перевіряти правильність введених даних (валідація);
- повідомляти користувача про помилки введення;
- забезпечувати збереження даних у базі даних;
- забезпечувати можливість редагування та видалення даних.

5. Вимоги до інтерфейсу користувача (UX/UI)

Інтерфейс веб-застосунку повинен відповідати сучасним вимогам UX/UI:

- адаптивність для мобільних пристроїв;
- логічна структура сторінок;
- мінімальна кількість кроків для виконання дій;
- використання зрозумілих кнопок та елементів керування;
- повідомлення про результат виконання операцій;
- підтримка темної/світлої теми за можливості;
- відповідність принципам доступності WCAG;
- застосування єдиної дизайн-системи кольори, шрифти, стилі.

6. Вимоги до надійності

Система повинна:

- працювати стабільно без збоїв;
- забезпечувати коректну обробку помилок;
- запобігати втраті даних при неправильних діях користувача;
- забезпечувати резервне копіювання бази даних за потреби;
- вести журнал подій логування.

7. Вимоги до інформаційного забезпечення

Інформаційне забезпечення включає базу даних, що повинна містити такі основні таблиці:

- користувачі;
- ролі користувачів;
- профілі;
- дані/записи системи;
- звернення користувачів;
- журнал подій;
- історія дій користувача.

База даних повинна забезпечувати цілісність, нормалізацію та підтримку зв’язків між таблицями.

8. Вимоги до програмного забезпечення

Для розробки та функціонування системи необхідне програмне забезпечення:

- ОС Windows 10/11 або Linux;
- Visual Studio Code;
- Node.js / Python (для серверної частини);
- React або Vue (frontend);
- PostgreSQL або MySQL;
- Git (контроль версій);
- Figma (проєктування UX/UI);
- Postman (тестування API);
- Web browser (Chrome/Firefox).

9. Вимоги до апаратного забезпечення

Мінімальні вимоги до комп'ютера розробника:

- процесор Intel Core i3 або AMD Ryzen 3;
- оперативна пам'ять 8 ГБ;
- SSD 128 ГБ;
- монітор від 15.6", роздільна здатність не менше 1366×768.

Рекомендовані вимоги:

- процесор Intel Core i5 / AMD Ryzen 5;
- оперативна пам'ять 16 ГБ;
- SSD 256–512 ГБ;
- монітор Full HD (1920×1080).

10. Вимоги до безпеки

Система повинна забезпечувати:

- захищений доступ через HTTPS;
- зберігання паролів у вигляді хешу;
- авторизацію користувачів за JWT або сесіями;
- захист від SQL-ін'єкцій та XSS-атак;
- обмеження доступу до функцій згідно ролей;
- контроль доступу до API.

11. Вимоги до тестування

Під час тестування повинні бути виконані:

- тестування функціональних модулів;
- перевірка коректності роботи форм;
- тестування API-запитів;
- перевірка адаптивності інтерфейсу;
- usability-тестування (зручність користування);
- перевірка доступності через Lighthouse.

12. Вимоги до документації

У результаті розробки повинна бути створена така документація:

- пояснювальна записка до бакалаврської роботи;
- технічне завдання;
- інструкція користувача;
- опис архітектури системи;
- опис структури бази даних;
- UML-діаграми (Use Case, Sequence, Deployment);
- приклади екранних форм інтерфейсу.

13. Очікувані результати

Результатом виконання технічного завдання є розроблений веб-застосунок, який:

- має сучасний UX/UI-дизайн;
- підтримує авторизацію та роботу з даними;
- має зручну навігацію та адаптивний інтерфейс;
- забезпечує безпечне збереження інформації;
- може бути використаний як демонстраційний або навчальний проєкт.

14. Порядок приймання роботи

Приймання веб-застосунку здійснюється шляхом:

- демонстрації роботи системи на локальному або тестовому сервері;

- перевірки функціональності всіх модулів;
- перевірки коректності роботи бази даних;
- оцінки зручності інтерфейсу;
- надання повного комплекту документації.

ДОДАТОК Б - ІНСТРУКЦІЯ КОРИСТУВАЧА

1. Загальні відомості

Даний веб-застосунок розроблено з метою забезпечення зручної та інтуїтивно зрозумілої взаємодії користувача з інформаційною системою через веб-інтерфейс. Застосунок реалізований із використанням принципів UX/UI-дизайну та адаптований для роботи на персональних комп'ютерах і мобільних пристроях.

Інструкція користувача призначена для ознайомлення з основними можливостями веб-застосунку, правилами роботи в системі та порядком виконання основних операцій.

2. Системні вимоги

Для коректної роботи веб-застосунку користувачу необхідно мати персональний комп'ютер, ноутбук, планшет або смартфон. Підключення до мережі Інтернет. Сучасний веб-браузер, рекомендовано, Google Chrome, а також Mozilla Firefox, Microsoft Edge, Safari.

Рекомендована роздільна здатність екрану не менше 1366×768, для ПК.

3. Запуск веб-застосунку

Для початку роботи користувачеві необхідно:

1. Запустити веб-браузер.
2. У адресному рядку ввести адресу веб-застосунку URL.
3. Після завантаження сторінки відкриється головна сторінка системи.

4. Головна сторінка

Головна сторінка містить назву та опис веб-застосунку, навігаційне меню, кнопки швидкого доступу до функцій входу та реєстрації, інформаційні блоки з описом можливостей системи.

Для переходу до функціоналу користувач повинен виконати реєстрацію або авторизацію.

5. Реєстрація користувача

Для створення нового облікового запису необхідно:

1. Натиснути кнопку «Реєстрація».

2. Заповнити реєстраційну форму:
   - ім’я за потреби,
   - email,
   - пароль,
   - підтвердження пароля.
3. Натиснути кнопку «Зареєструватися».
4. Після успішної реєстрації користувач отримає повідомлення про створення облікового запису.

У разі неправильного введення даних система повідомляє користувача про помилку та пропонує виправити введену інформацію.

6. Авторизація користувача вхід у систему

Для входу в систему необхідно:

1. Натиснути кнопку «Вхід».
2. Увести email та пароль у форму авторизації.
3. Натиснути кнопку «Увійти».

Після успішного входу користувач перенаправляється на головну панель (Dashboard).

Якщо введені дані некоректні, система відображає повідомлення про помилку.

7. Головна панель користувача Dashboard

Після входу користувач отримує доступ до основних функцій системи через Dashboard. На панелі відображаються:

- основні розділи веб-застосунку;
- кнопки для виконання операцій;
- інформаційні блоки (статистика, останні записи, повідомлення);
- меню профілю користувача.

Dashboard забезпечує швидкий доступ до ключового функціоналу та спрощує навігацію.

8. Робота з даними створення, перегляд, редагування.

У системі передбачено можливість виконання основних операцій з інформаційними записами.

8.1 Перегляд даних

Для перегляду записів необхідно:

1. Відкрити відповідний розділ у меню.
2. Переглянути список доступних записів.
3. Для детального перегляду натиснути на потрібний запис.

8.2 Додавання нового запису

Для створення нового запису необхідно:

1. Натиснути кнопку «Додати» або «Створити».
2. Заповнити форму введення даних.
3. Натиснути кнопку «Зберегти».

Після збереження система відобразить повідомлення про успішне додавання.

8.3 Редагування запису

Для редагування необхідно:

1. Обрати потрібний запис зі списку.
2. Натиснути кнопку «Редагувати».
3. Внести зміни у форму.
4. Натиснути кнопку «Зберегти».

8.4 Видалення запису

Для видалення запису необхідно:

1. Обрати потрібний запис.
2. Натиснути кнопку «Видалити».
3. Підтвердити дію у вікні підтвердження.

9. Пошук і фільтрація даних

Для швидкого знаходження інформації у веб-застосунку передбачено функції пошуку та фільтрації.

Користувач може:

- вводити ключові слова у поле пошуку;

- використовувати фільтри за категоріями або датою;
- сортувати записи за певними параметрами.

Результати пошуку відображаються автоматично після введення запиту або натискання кнопки «Пошук».

10. Робота з профілем користувача

У системі реалізовано можливість перегляду та редагування профілю.

Для переходу до профілю необхідно:

1. Натиснути на іконку користувача у верхній частині сторінки.
2. Обрати пункт «Профіль».

У профілі можна:

- переглядати персональні дані;
- змінювати ім’я, номер телефону або інші параметри;
- змінювати пароль за наявності функції;
- налаштовувати параметри інтерфейсу наприклад, тема.

11. Форма зворотного зв’язку

Для надсилання повідомлення адміністратору або звернення до служби підтримки необхідно:

1. Відкрити розділ «Зворотний зв’язок».
2. Заповнити форму:
   - тема повідомлення;
   - текст звернення.
3. Натиснути кнопку «Надіслати».

Після надсилання користувач отримує повідомлення про успішну відправку звернення.

12. Функції адміністратора

Користувач з роллю Адміністратор має доступ до додаткових функцій:

- перегляд списку користувачів;
- блокування або видалення користувачів;
- призначення ролей;
- перегляд журналу подій системи;

- керування налаштуваннями веб-застосунку.

Для доступу до адміністративного меню необхідно перейти у розділ «Адміністрування».

13. Вихід із системи

Для завершення роботи необхідно:

1. Натиснути на меню користувача.
2. Обрати пункт «Вийти».

Після виходу система повертає користувача на головну сторінку.

14. Повідомлення про помилки та рекомендації

У процесі роботи можуть виникати такі ситуації:

- Невірний логін або пароль — необхідно повторити введення даних.
- Некоректне заповнення форми — система підсвічує поля з помилками.
- Відсутність доступу до функції — означає, що користувач не має відповідної ролі.
- Втрата Інтернет-з’єднання — потрібно перевірити мережу та перезавантажити сторінку.

15. Рекомендації з безпеки

Для безпечного використання веб-застосунку рекомендується:

- не передавати пароль третім особам;
- використовувати складні паролі;
- виходити із системи після завершення роботи;
- не зберігати пароль у загальнодоступних місцях;
- не використовувати застосунок на незахищених пристроях.

Дана інструкція користувача описує основні можливості веб-застосунку та порядок виконання основних операцій. Використання системи не потребує спеціальних технічних знань, оскільки інтерфейс розроблено з урахуванням принципів UX/UI-дизайну, що забезпечує простоту, доступність та зручність роботи для користувача.

ДОДАТОК В. ЕКРАННІ ФОРМИ РОЗРОБКИ

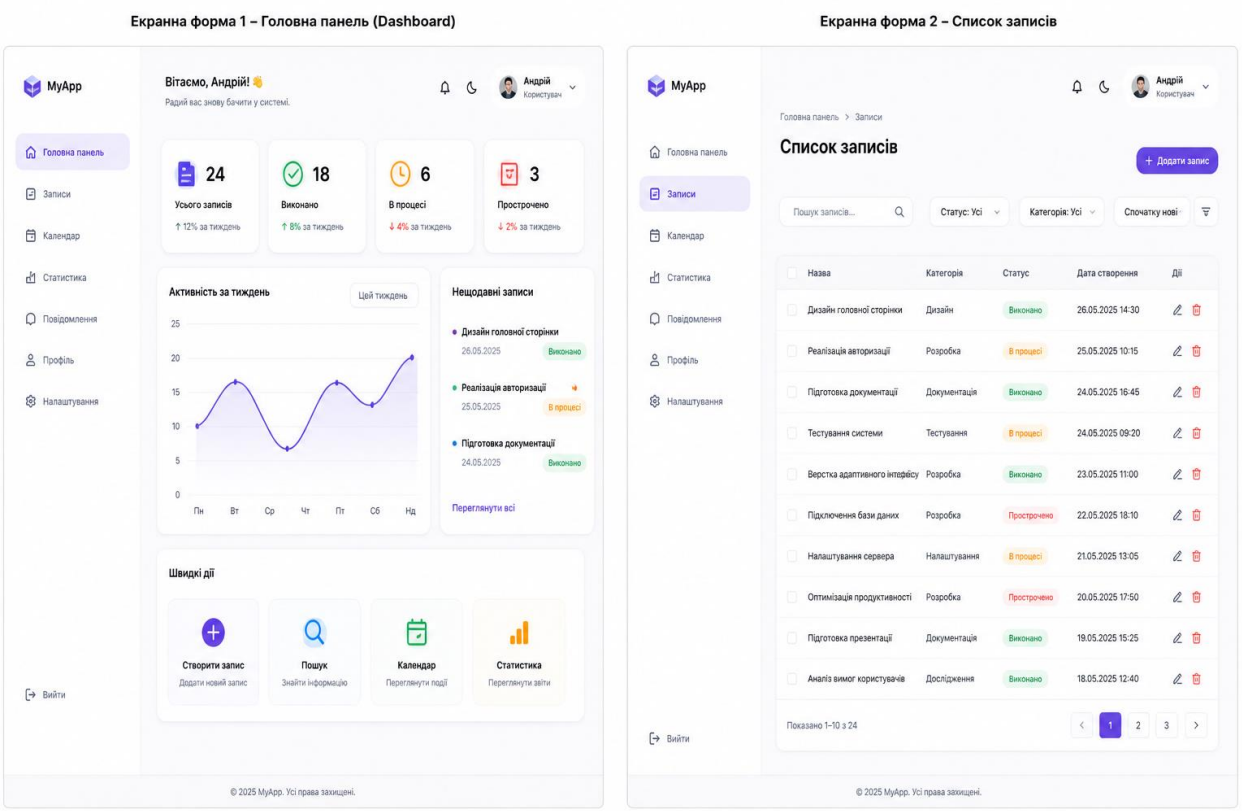

Рисунок В.1 – Екранні форми розробки