ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
«ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД»
МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ
ІНСТИТУТ КОМП'ЮТЕРНО-ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Кафедра комп'ютерних інформаційних систем та технологій

Труфанов Антон Володимирович
(прізвище, ім´я, по батькові здобувача вищої освіти в називному відмінку)

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня бакалавра
на тему**: «Проєктування та розроблення веборієнтованої інформаційної системи для обліку заявок користувачів»**

ТУчр-9-22-Б1КНТ (4.0д)
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітньо-професійна програма: «Комп'ютерні науки»
Рівень вищої освіти: перший (бакалаврський)

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

_____ Труфанов А.В.
(підпис) (прізвище та ініціали)

Науковий керівник
Старший викладач _____ Маклюк О.В.
(посада, науковий ступінь, вчене звання) (підпис) (прізвище та ініціали)

Київ – 2026 рік

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
«ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД»
МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ
ІНСТИТУТ КОМП'ЮТЕРНО-ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Кафедра комп'ютерних інформаційних систем та технологій

Рівень вищої освіти: перший (бакалаврський) рівень
Ступінь вищої освіти: бакалавр
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітньо-професійна програма: «Комп'ютерні науки»

Завдання
на кваліфікаційну роботу здобувачу вищої освіти

Труфанову Антону Володимировичу
(прізвище, ім´я, по батькові)

1.Тема кваліфікаційної роботи:
«Проєктування та розроблення веборієнтованої інформаційної системи для обліку заявок користувачів»
Керівник кваліфікаційної роботи
Старший викладач
(посада, науковий ступінь, вчене звання)
Маклюк Олег Володимирович
(прізвище, ім´я, по батькові)
2. Термін подання здобувачем кваліфікаційної роботи на попередній захист 15.05.2026 року
3. Вихідні дані до кваліфікаційної роботи (рекомендовані науковим керівником та відповідним методичним забезпеченням (окремі аспекти роботи, обсяги роботи):
Дослідити предметну область обліку, опрацювання та контролю заявок користувачів у веборієнтованому інформаційному середовищі. Проаналізувати призначення, функції та особливості інформаційних систем обліку заявок, визначити їх роль у цифровізації сервісних, адміністративних та організаційних процесів. Розглянути сучасні підходи до побудови веборієнтованих інформаційних систем, зокрема клієнт-серверну архітектуру, використання баз даних, розмежування ролей користувачів, механізми авторизації та засоби забезпечення зручності користувацького інтерфейсу.
Обґрунтувати вибір технологій і програмних засобів для розроблення веборієнтованої інформаційної системи обліку заявок користувачів. Спроєктувати архітектуру програмного продукту, визначити основні функціональні модулі системи, ролі користувачів, структуру бази даних, алгоритми створення, перегляду, редагування, фільтрації та зміни статусу заявок. Розробити програмну реалізацію основних модулів системи, користувацький інтерфейс і сценарії взаємодії користувачів із системою. Провести тестування розробленого програмного продукту, перевірити коректність виконання основних функцій, працездатність форм, логіку оброблення заявок, роботу з базою даних та відповідність системи визначеним функціональним і нефункціональним вимогам.

4. Необхідний графічний та аналітичний матеріал за темою дослідження:
рисунки, таблиці, графіки, формули, розрахунки.
5. Календарний план виконання кваліфікаційної роботи

№ п/п	Назва етапів виконання кваліфікаційної роботи	Строки виконання		Примітки
		Планові	Фактичні	
1	2	3	4	5
	Вступ	16.02.2026-02.03.2026		
	Розділ 1.	02.03.2026–16.03.2026		
	Розділ 2.	16.03.2026–06.04.2026		
	Розділ 3.	06.04.2026–22.04.2026		
	Висновки	До 30.04.2026		

6. Дата видачі завдання _____28.08.2025 р._____

Здобувач (ка) _____ ____Васерук В. О.____
(підпис) (прізвище та ініціали)

Науковий керівник _____ ____Маклюк О.В._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Труфанов А. В. «Проєктування та розроблення веборієнтованої інформаційної системи для обліку заявок користувачів» – Рукопис.

Кваліфікаційна робота на здобуття першого (бакалаврського) рівня вищої освіти – Приватне акціонерне товариство «Вищий навчальний заклад «Міжрегіональна Академія управління персоналом»», Інститут комп'ютерно-інформаційних технологій та дизайну – Київ, 2026.

Актуальність теми дослідження зумовлена потребою в автоматизації процесів реєстрації, опрацювання та контролю заявок користувачів у сучасних організаціях. Використання веборієнтованих інформаційних систем дозволяє впорядкувати роботу зі зверненнями, забезпечити централізоване збереження даних, підвищити прозорість виконання заявок, розмежувати права доступу користувачів і спростити контроль за станом виконання завдань.

Мета роботи полягає у проєктуванні та розробленні веборієнтованої інформаційної системи для обліку заявок користувачів, яка забезпечує створення, збереження, перегляд, редагування, фільтрацію, зміну статусів і контроль виконання заявок у цифровому середовищі.

У кваліфікаційній роботі досліджено предметну область систем обліку заявок користувачів, проаналізовано сучасні підходи до побудови веборієнтованих інформаційних систем, визначено функціональні та нефункціональні вимоги до програмного продукту. Обґрунтовано вибір технологій розроблення, спроєктовано архітектуру системи, функціональні модулі, ролі користувачів, структуру бази даних та алгоритми оброблення заявок. Реалізовано основні програмні модулі веборієнтованої системи, розроблено користувацький інтерфейс і проведено тестування працездатності програмного продукту.

Методи дослідження, використані у роботі: аналіз і узагальнення наукових та технічних джерел, порівняльний аналіз технологій розроблення, моделювання структури інформаційної системи, проєктування бази даних, алгоритмізація процесів оброблення заявок, методи програмної інженерії та тестування програмного забезпечення.

Ключові слова: веборієнтована інформаційна система, заявка користувача, база даних, програмне забезпечення, вебзастосунок, клієнт-серверна архітектура, інтерфейс користувача, тестування.

ANNOTATION

Trufanov A. V. "Design and Development of a Web-Oriented Information System for User Request Accounting" – Manuscript.

Qualification work for obtaining the first (bachelor's) level of higher education – Private Joint-Stock Company "Higher Educational Institution 'Interregional Academy of Personnel Management'", Institute of Computer Information Technologies and Design – Kyiv, 2026.

The relevance of the research topic is determined by the need to automate the processes of registering, processing, and controlling user requests in modern organizations. The use of web-oriented information systems makes it possible to organize request processing, ensure centralized data storage, increase the transparency of task execution, differentiate user access rights, and simplify monitoring of request statuses.

The aim of the qualification work is to design and develop a web-oriented information system for user request accounting that provides creation, storage, viewing, editing, filtering, status changing, and control of user requests in a digital environment.

The qualification work examines the subject area of user request accounting systems, analyzes modern approaches to the development of web-oriented information systems, and defines the functional and non-functional requirements for the software product. The choice of development technologies is substantiated, the system architecture, functional modules, user roles, database structure, and request processing algorithms are designed. The main software modules of the web-oriented system are implemented, the user interface is developed, and the functionality of the software product is tested.

The research methods used in the work include analysis and generalization of scientific and technical sources, comparative analysis of development technologies, modeling of the information system structure, database design, algorithmization of request processing procedures, software engineering methods, and software testing.

Keywords: web-oriented information system, user request, database, software, web application, client-server architecture, user interface, testing.

ЗМІСТ

ВСТУП

У сучасних умовах цифрової трансформації організацій дедалі більшого значення набувають інформаційні системи, які забезпечують автоматизацію повсякденних управлінських, сервісних та комунікаційних процесів. Одним із поширених напрямів такої автоматизації є створення програмних засобів для обліку, опрацювання та контролю заявок користувачів. Подібні системи використовуються в освітніх установах, IT-відділах, сервісних службах, адміністративних підрозділах, технічній підтримці, малому та середньому бізнесі, а також у внутрішніх корпоративних процесах, де необхідно швидко фіксувати звернення, призначати відповідальних осіб, відстежувати стан виконання завдань і накопичувати інформацію про виконані роботи.

Веборієнтована інформаційна система обліку заявок користувачів дає змогу перевести процес опрацювання звернень у структуровану цифрову форму. Її використання забезпечує реєстрацію заявки, збереження основної інформації про користувача, визначення категорії звернення, призначення статусу, контроль виконання, перегляд історії змін і формування підсумкових даних. Особливого значення набуває саме веборієнтований формат такої системи, оскільки він дозволяє працювати з програмним продуктом через браузер без потреби встановлення спеціального клієнтського програмного забезпечення на кожному робочому місці. Це підвищує доступність системи, спрощує її супровід і створює умови для використання різними групами користувачів.

Актуальність теми кваліфікаційної роботи визначається необхідністю розроблення простих, зрозумілих і функціональних веборієнтованих інформаційних систем, які можуть застосовуватися для автоматизації процесу обліку заявок користувачів. Така система має не лише зберігати інформацію про звернення, а й забезпечувати логіку їх опрацювання: створення заявки, перегляд її змісту, зміну статусу, призначення відповідального виконавця, фільтрацію, пошук, сортування та контроль завершення роботи. Саме тому розроблення подібного програмного продукту потребує поєднання аналізу предметної

області, вибору технологічних засобів, проєктування архітектури, створення структури бази даних, реалізації функціональних модулів і тестування результатів.

Метою кваліфікаційної роботи є проєктування та розроблення веборієнтованої інформаційної системи для обліку заявок користувачів, яка забезпечує реєстрацію, збереження, перегляд, опрацювання та контроль виконання заявок у цифровому середовищі.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

дослідити призначення, функції та особливості інформаційних систем обліку заявок користувачів;

проаналізувати сучасні підходи до побудови веборієнтованих інформаційних систем;

визначити основні функціональні та нефункціональні вимоги до системи обліку заявок;

обґрунтувати вибір технологій і програмних засобів для реалізації веборієнтованої системи;

спроєктувати архітектуру інформаційної системи, її функціональні модулі та ролі користувачів;

розробити структуру бази даних для збереження інформації про користувачів, заявки, статуси та історію змін;

описати алгоритми основних процесів роботи системи;

реалізувати основні функціональні модулі веборієнтованої інформаційної системи;

розробити користувацький інтерфейс і сценарії взаємодії користувачів із системою;

Об’єктом дослідження є процес обліку, опрацювання та контролю заявок користувачів у веборієнтованому інформаційному середовищі.

Предметом дослідження є методи, алгоритми, програмні засоби та технології проєктування і розроблення веборієнтованої інформаційної системи для обліку заявок користувачів.

У роботі використовуються такі **методи дослідження:** метод аналізу та узагальнення для вивчення предметної області й сучасних підходів до створення інформаційних систем; метод порівняння для вибору технологічних засобів розроблення; метод моделювання для побудови структури системи, бази даних і логіки взаємодії користувачів; метод алгоритмізації для опису основних процесів оброблення заявок; методи програмної інженерії для реалізації функціональних модулів; метод тестування для перевірки працездатності розробленого програмного продукту.

Практичне значення роботи полягає в тому, що запропонована веборієнтована інформаційна система може бути використана як прототип програмного рішення для обліку заявок користувачів у невеликій організації, навчальному закладі, сервісному підрозділі або внутрішній службі технічної підтримки. Розроблена система дозволяє впорядкувати процес реєстрації звернень, забезпечити зручний перегляд заявок, контролювати їх статуси та зберігати інформацію про виконання.

Науково-технічна новизна роботи полягає у формуванні цілісної моделі веборієнтованої інформаційної системи обліку заявок користувачів, яка поєднує структурований підхід до збереження даних, розмежування ролей користувачів, алгоритмічну логіку оброблення заявок і базові механізми контролю стану їх виконання. У межах роботи здійснюється не лише опис предметної області, а й розроблення програмної структури системи, що охоплює базу даних, функціональні модулі та користувацький інтерфейс.

Інформаційною основою дослідження є наукова, навчальна та технічна література з питань веброзробки, проєктування інформаційних систем, баз даних, програмної інженерії, моделювання програмних продуктів, тестування програмного забезпечення, а також офіційна документація до технологій, які можуть використовуватися під час реалізації системи.

Структура кваліфікаційної роботи відповідає логіці поставленої мети та завдань. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ ДО РОЗРОБЛЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ОБЛІКУ ЗАЯВОК

1.1 Поняття, призначення та функції інформаційних систем обліку заявок користувачів

Розвиток цифрових технологій істотно змінив підходи до організації роботи з інформацією в установах, підприємствах, сервісних службах та освітніх організаціях. Якщо раніше значна частина звернень користувачів могла фіксуватися у паперових журналах, електронних таблицях, електронній пошті або навіть у неформальних каналах комунікації, то сучасна практика потребує більш упорядкованих програмних рішень. Особливо це стосується тих процесів, де необхідно не просто отримати звернення, а забезпечити його реєстрацію, класифікацію, контроль виконання, збереження історії змін і подальший аналіз.

У цьому контексті важливе місце посідають інформаційні системи обліку заявок користувачів. Їх можна розглядати як спеціалізовані програмні системи, призначені для автоматизації процесу приймання, збереження, опрацювання та контролю виконання звернень, які надходять від користувачів. Такими заявками можуть бути повідомлення про технічну проблему, прохання надати консультацію, запит на виконання певної послуги, звернення щодо несправності, уточнення інформації або інше завдання, яке потребує реакції відповідальної особи.

У найбільш загальному розумінні інформаційна система є сукупністю програмних, інформаційних, технічних, організаційних і користувацьких компонентів, які забезпечують збирання, збереження, оброблення та передавання даних. У межах теми цієї роботи основна увага зосереджується саме на програмному аспекті інформаційної системи, тобто на створенні веборієнтованого програмного продукту, який забезпечує роботу із заявками

користувачів. Як зазначається у спеціальній літературі, «інформаційні системи мають не лише зберігати дані, а й підтримувати виконання певних процесів, пов’язаних із прийняттям рішень, контролем і взаємодією користувачів» [16, с.28].

Заявка користувача в такій системі виступає основною інформаційною одиницею. Вона містить дані про ініціатора звернення, тему або категорію проблеми, опис ситуації, дату створення, поточний статус, пріоритет, відповідального виконавця та результат опрацювання. Саме через заявку відбувається фіксація потреби користувача та її подальше перетворення на конкретне завдання для виконання. Тому якість організації обліку заявок безпосередньо впливає на швидкість реагування, прозорість виконання робіт і загальний рівень сервісу.

Особливістю інформаційної системи обліку заявок є те, що вона поєднує декілька різних функціональних напрямів. З одного боку, система виконує роль електронного журналу звернень, у якому фіксуються всі заявки. З іншого боку, вона є інструментом контролю виконання, оскільки дозволяє бачити, на якому етапі перебуває кожне звернення.

У практичному середовищі такі системи часто пов’язують із поняттями helpdesk, service desk, ticket system або система підтримки користувачів. Проте для бакалаврської роботи доцільно використовувати ширше формулювання – веборієнтована інформаційна система обліку заявок користувачів. Воно дозволяє не обмежувати розробку лише ІТ-підтримкою, а розглядати її як універсальний програмний засіб для фіксації та опрацювання звернень у різних організаційних умовах: навчальному закладі, сервісному відділі, адміністративній службі, внутрішній технічній підтримці або невеликій компанії.

Веборієнтований характер системи означає, що доступ до неї здійснюється через веббраузер. Це є важливою перевагою, оскільки користувачеві не потрібно встановлювати окреме програмне забезпечення на свій пристрій. Достатньо мати доступ до мережі та браузера. Такий підхід спрощує використання системи, полегшує її оновлення, централізує зберігання даних і дозволяє організувати

роботу різних категорій користувачів у межах єдиного інформаційного середовища.

Основне призначення системи обліку заявок полягає у впорядкуванні процесу роботи зі зверненнями. У разі відсутності спеціалізованого програмного рішення заявки можуть надходити через різні канали: телефонні дзвінки, повідомлення в месенджерах, електронні листи, усні звернення або неформальні домовленості. Така ситуація створює декілька проблем. По-перше, частина звернень може бути втрачена або залишена без відповіді. По-друге, складно визначити, хто саме відповідає за виконання конкретної заявки. По-третє, відсутня єдина історія змін, що ускладнює контроль і подальший аналіз. По-четверте, керівник або адміністратор не завжди може оцінити реальне навантаження на виконавців.

Використання інформаційної системи дозволяє подолати ці недоліки. Кожна заявка отримує фіксоване місце в базі даних, має свій статус, дату створення, опис і відповідального виконавця. Завдяки цьому процес опрацювання звернень стає більш прозорим і контрольованим. Користувач може бачити, що його заявка зареєстрована, виконавець отримує конкретне завдання, а адміністратор має змогу контролювати загальну картину роботи системи.

Для кращого розуміння ролі інформаційної системи обліку заявок користувачів доцільно узагальнити її основні функції (див. таблицю 1.1). Вони охоплюють не лише створення та збереження заявки, а й ширший комплекс дій, пов’язаних із її опрацюванням.

Як показано в таблиці 1.1, інформаційна система обліку заявок не обмежується лише збереженням звернень. Вона виконує ширшу роль – підтримує повний життєвий цикл заявки від моменту її створення до завершення опрацювання. Саме наявність такого життєвого циклу відрізняє повноцінну систему обліку заявок від звичайної електронної таблиці або журналу повідомлень. У таблиці також видно, що окремі функції системи мають не лише технічне, а й організаційне значення: вони допомагають розподіляти відповідальність, контролювати строки виконання та накопичувати дані для

подальшого аналізу.

Таблиця 1.1 – Основні функції інформаційної системи обліку заявок користувачів

Функція системи	**Зміст функції**	**Практичне значення**
Реєстрація заявки	Створення нової заявки із зазначенням теми, опису проблеми, користувача, дати та інших необхідних даних	Забезпечує первинну фіксацію звернення та запобігає втраті інформації
Класифікація заявки	Визначення категорії, типу звернення або напряму виконання	Дозволяє швидше спрямувати заявку до відповідного виконавця
Призначення статусу	Встановлення поточного стану заявки: нова, у роботі, виконана, закрита, відхилена	Дає змогу контролювати етап виконання кожного звернення
Призначення виконавця	Закріплення заявки за відповідальною особою або групою виконавців	Підвищує персональну відповідальність за виконання завдання
Перегляд і пошук заявок	Можливість переглядати список заявок, здійснювати пошук, фільтрацію та сортування	Спрощує роботу з великою кількістю звернень
Редагування даних	Оновлення опису, статусу, пріоритету, відповідального виконавця або інших параметрів заявки	Забезпечує актуальність інформації в системі
Збереження історії змін	Фіксація основних дій, пов’язаних із заявкою	Дозволяє відстежити процес її опрацювання
Розмежування ролей	Надання різних прав доступу користувачу, виконавцю та адміністратору	Підвищує керованість і безпечність роботи системи
Формування звітності	Отримання узагальненої інформації про кількість, стан і результати виконання заявок	Дає можливість аналізувати ефективність роботи системи

Важливою характеристикою системи обліку заявок є наявність різних ролей користувачів. У найпростішому варіанті можна виокремити три основні ролі: звичайний користувач, виконавець і адміністратор. Звичайний користувач створює заявку, переглядає її стан і за потреби уточнює інформацію. Виконавець опрацьовує заявку, змінює її статус, додає коментарі або результат виконання. Адміністратор керує користувачами, налаштовує категорії заявок, контролює загальний стан системи та має доступ до розширених функцій. Такий розподіл ролей дозволяє зробити систему більш структурованою та безпечною.

З погляду програмної реалізації, інформаційна система обліку заявок

повинна забезпечувати взаємодію між інтерфейсом користувача, серверною логікою та базою даних. «Інтерфейс відповідає за введення та відображення інформації, серверна частина обробляє запити й перевіряє логіку виконання операцій, а база даних зберігає інформацію про користувачів, заявки, статуси, ролі та історію змін. Така логіка відповідає загальному принципу побудови веборієнтованих інформаційних систем, у яких дані централізовано зберігаються на сервері, а користувач взаємодіє із системою через браузер» [19, с. 41].

Процес роботи із заявкою в узагальненому вигляді можна подати як послідовність взаємопов’язаних етапів. Спочатку користувач створює заявку, після чого вона реєструється в системі та отримує початковий статус. Далі заявка може бути переглянута адміністратором або виконавцем, класифікована, доповнена необхідною інформацією та передана в роботу. Після виконання відповідних дій статус заявки змінюється, а результат зберігається в системі. Завершальним етапом є закриття заявки та збереження її історії.

Для наочності цей процес подано на рисунку 1.1.

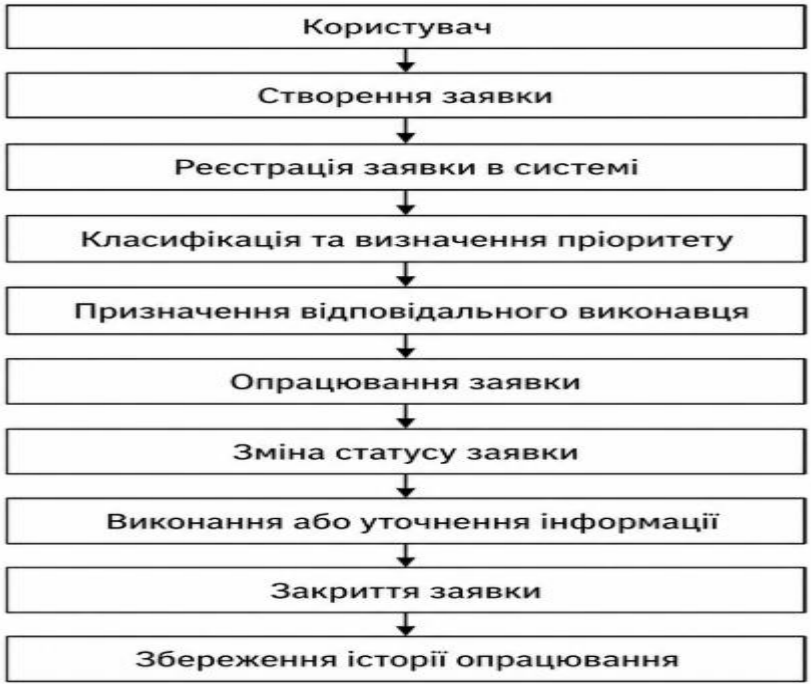

Рисунок 1.1 – Узагальнена схема процесу оброблення заявки користувача

На рисунку 1.1 відображено загальну логіку проходження заявки в інформаційній системі. Вона починається з ініціативи користувача, який формує

звернення, і завершується закриттям заявки після виконання необхідних дій. Важливо, що кожен етап цього процесу має бути відображений у програмній системі. Якщо хоча б один із ключових етапів не фіксується, втрачається прозорість роботи із заявкою. Наприклад, без етапу призначення виконавця складно визначити відповідального за виконання; без збереження історії змін неможливо відстежити, які дії виконувалися; без статусів не можна швидко зрозуміти поточний стан звернення.

З позиції користувача така система повинна бути простою і зрозумілою. Для створення заявки він має заповнити мінімально необхідну кількість полів: тему, опис проблеми, категорію або тип звернення. Надмірна складність форми може знизити зручність використання системи. Водночас занадто мала кількість полів може ускладнити подальше опрацювання заявки, оскільки виконавцю не вистачатиме інформації. Тому під час проєктування необхідно знайти баланс між простотою інтерфейсу та достатністю даних для виконання звернення.

З позиції виконавця система має забезпечувати швидкий доступ до списку заявок, можливість фільтрації за статусом, пріоритетом або датою, перегляд детальної інформації та зміну стану заявки. Для адміністратора важливими є функції управління користувачами, перегляду загальної кількості заявок, контролю виконання та аналізу навантаження. Таким чином, одна й та сама система повинна підтримувати різні сценарії роботи залежно від ролі користувача.

Окрему увагу необхідно звернути на питання статусів заявки. Статус є одним із ключових параметрів, оскільки саме він відображає поточний етап її опрацювання. У межах розроблюваної системи доцільно передбачити такі базові статуси: «нова», «у роботі», «очікує уточнення», «виконана», «закрита». За потреби система може бути розширена додатковими статусами, наприклад «відхилена», «передана іншому виконавцю», «призупинена». Проте на етапі бакалаврської роботи доцільно не перевантажувати систему надмірною кількістю станів, щоб логіка роботи залишалася зрозумілою.

Інформаційні системи обліку заявок також мають значення для

накопичення даних. Кожна заявка після завершення не зникає, а залишається в базі даних. У майбутньому це дозволяє аналізувати типові проблеми, частоту звернень, середній час виконання, навантаження на виконавців і повторюваність окремих категорій заявок. Навіть якщо в межах бакалаврської роботи не передбачається складний аналітичний модуль, сама структура системи має бути спроєктована так, щоб у подальшому її можна було розширити.

З технічної точки зору важливо забезпечити цілісність і коректність даних. Наприклад, заявка не повинна існувати без користувача, який її створив, а зміна статусу має відображатися у відповідному полі бази даних. Якщо система передбачає історію змін, то кожна дія повинна мати дату, автора та опис виконаної операції. Такі вимоги є важливими не лише для зручності роботи, а й для надійності програмного продукту.

Слід також ураховувати питання безпеки. Оскільки система працює з даними користувачів і заявками, необхідно передбачити авторизацію, розмежування прав доступу, перевірку введених даних і захист від некоректних дій. Звичайний користувач не повинен мати доступ до адміністративних функцій, виконавець не повинен змінювати системні налаштування, а адміністратор має контролювати структуру системи. На практиці це реалізується через механізм ролей і дозволів, який є одним із базових елементів веборієнтованих інформаційних систем [55].

Отже, інформаційна система обліку заявок користувачів є прикладним програмним рішенням, призначеним для автоматизації процесу роботи зі зверненнями. Вона забезпечує реєстрацію заявок, їх класифікацію, призначення виконавців, зміну статусів, збереження історії та контроль виконання. Її використання дозволяє підвищити прозорість роботи із заявками, зменшити ризик втрати інформації, упорядкувати взаємодію між користувачами й виконавцями та створити основу для подальшого аналізу сервісних процесів. Для теми цієї бакалаврської роботи така система є доцільним об’єктом проєктування, оскільки поєднує предметну область, базу даних, алгоритми, вебінтерфейс і програмну логіку в межах єдиного інформаційного продукту.

1.2 Аналіз сучасних підходів до побудови веборієнтованих інформаційних систем

Веборієнтовані інформаційні системи сьогодні є одним із найбільш поширених видів програмного забезпечення, оскільки вони забезпечують доступ користувачів до функцій системи через веббраузер і не потребують обов’язкового встановлення окремої клієнтської програми на кожному робочому місці. Такий підхід є особливо зручним для систем, які мають використовуватися різними групами користувачів, працювати з централізованими даними та забезпечувати оперативне оновлення інформації. Для системи обліку заявок користувачів це має принципове значення, оскільки заявки можуть створюватися, переглядатися й опрацьовуватися різними учасниками процесу: користувачами, виконавцями та адміністраторами.

У загальному вигляді веборієнтовану інформаційну систему можна розглядати як програмне рішення, у якому взаємодія користувача з даними відбувається через вебінтерфейс, а основна логіка оброблення запитів реалізується на серверній стороні. Користувач відкриває сторінку системи у браузері, вводить або переглядає інформацію, надсилає запит, після чого серверна частина обробляє цей запит, звертається до бази даних і повертає результат. Така модель дає змогу відокремити інтерфейс користувача від логіки оброблення даних і від їх фізичного збереження.

У спеціальній літературі зазначається, що «вебзастосунок є програмною системою, доступ до якої здійснюється через веббраузер, а оброблення основної частини даних виконується на сервері або у взаємодії клієнтської та серверної частин» [19, с. 52]. Це визначення добре відображає сутність системи, що розробляється в межах цієї бакалаврської роботи, оскільки майбутній програмний продукт має забезпечувати доступ користувачів до функцій створення, перегляду, редагування та контролю заявок саме через вебінтерфейс.

Сучасні підходи до побудови веборієнтованих інформаційних систем можуть відрізнятися залежно від складності системи, обсягу даних, кількості

користувачів, вимог до продуктивності, безпеки, масштабованості та зручності супроводу. Найчастіше розглядають монолітну архітектуру, клієнт-серверну архітектуру, багаторівневу архітектуру, архітектуру на основі REST API, сервісно орієнтований підхід і мікросервісну архітектуру. Для бакалаврської роботи важливо не просто перелічити ці підходи, а визначити, який із них є найбільш доцільним для системи обліку заявок користувачів.

Одним із традиційних підходів є монолітна архітектура. Вона передбачає, що основні компоненти системи – інтерфейс, бізнес-логіка та доступ до бази даних – реалізуються в межах одного програмного застосунку. Такий підхід є відносно простим для розроблення, тестування й початкового розгортання. Він добре підходить для невеликих систем, де кількість функцій обмежена, а навантаження на систему не є надмірним. Наприклад, проста система обліку заявок для невеликої організації може бути реалізована як єдиний вебзастосунок, у якому сторінки інтерфейсу, обробники запитів і операції з базою даних розміщені в межах одного проєкту.

Перевагою монолітної архітектури є її зрозумілість і відносна простота реалізації. Розробнику легше контролювати структуру системи, оскільки всі основні компоненти знаходяться в одному середовищі. Проте зі збільшенням кількості функцій монолітна система може ставати менш гнучкою. Будь-які зміни в одному компоненті можуть впливати на інші частини програми, ускладнюється масштабування окремих модулів, а підтримка коду потребує більшої уваги до структурування. Для бакалаврської роботи такий підхід можна використати, але необхідно показати внутрішній поділ системи на логічні модулі: авторизація, робота із заявками, адміністрування користувачів, оброблення статусів, формування списків і фільтрація.

Більш універсальним підходом є клієнт-серверна архітектура. Вона передбачає поділ системи на клієнтську частину, яка відповідає за взаємодію з користувачем, і серверну частину, яка забезпечує оброблення запитів, виконання бізнес-логіки та взаємодію з базою даних. У контексті веборієнтованої системи клієнтом найчастіше виступає браузер, а сервером – вебсервер і серверна

програма, що обробляє запити користувачів. Саме цей підхід є найбільш придатним для розроблення системи обліку заявок користувачів.

Клієнт-серверна модель має важливу перевагу: вона дозволяє централізовано зберігати дані та забезпечувати доступ до них із різних пристроїв. Для системи обліку заявок це означає, що користувач може створити заявку, виконавець може переглянути її зі свого робочого місця, а адміністратор – проконтролювати її стан у загальному списку. Усі учасники працюють із єдиною базою даних, тому знижується ризик дублювання інформації або її втрати. Крім того, оновлення серверної частини системи може виконуватися централізовано, без потреби змінювати програмне забезпечення на кожному клієнтському пристрої.

У працях із програмної інженерії підкреслюється, що «поділ програмної системи на клієнтську та серверну частини дає змогу розмежувати функції подання інформації, оброблення запитів і збереження даних» [25, с. 118]. Це положення є важливим для проєктування системи обліку заявок, адже така система повинна не лише відображати дані на екрані користувача, а й виконувати перевірку введеної інформації, змінювати статуси заявок, контролювати права доступу та підтримувати цілісність бази даних.

Наступним поширеним підходом є багаторівнева архітектура. У найпростішому варіанті вона містить три рівні: рівень подання, рівень бізнес-логіки та рівень даних. Рівень подання відповідає за інтерфейс користувача, рівень бізнес-логіки – за правила оброблення інформації, а рівень даних – за збереження і вибірку інформації з бази даних. Такий підхід є розвитком клієнт-серверної моделі та дозволяє зробити систему більш структурованою. Для бакалаврської роботи він є дуже доречним, оскільки дає змогу чітко описати логіку побудови системи.

У системі обліку заявок користувачів рівень подання може включати сторінки авторизації, форму створення заявки, список заявок, сторінку перегляду детальної інформації та адміністративну панель. Рівень бізнес-логіки відповідатиме за створення заявки, перевірку прав доступу, зміну статусу,

призначення виконавця, фільтрацію та оброблення запитів. Рівень даних міститиме таблиці користувачів, ролей, заявок, статусів, категорій та історії змін. Такий поділ робить систему зрозумілою для опису, реалізації й подальшого тестування.

Окреме місце серед сучасних підходів займає використання REST API. Такий підхід передбачає, що клієнтська частина системи взаємодіє із сервером через стандартизовані HTTP-запити. Наприклад, для отримання списку заявок може використовуватися запит GET, для створення нової заявки – POST, для оновлення статусу – PUT або PATCH, для видалення – DELETE. REST API є зручним тоді, коли потрібно відокремити клієнтську частину від серверної або передбачити можливість подальшого підключення мобільного застосунку чи зовнішніх сервісів.

Для цієї бакалаврської роботи REST API можна розглядати як перспективний підхід, але не обов’язково робити його надто складним. Якщо система реалізується у простішому варіанті, наприклад із використанням PHP та MySQL, взаємодія може бути побудована через серверні обробники форм і запитів. Якщо ж використовується сучасніший стек, наприклад Node.js, Express і клієнтська частина на JavaScript, тоді REST API буде природним рішенням для зв’язку між інтерфейсом і сервером. У будь-якому випадку в роботі доцільно показати, що система має логіку запитів і відповідей, а не просто набір статичних сторінок.

Ще одним підходом є сервісно орієнтована архітектура. Вона передбачає поділ системи на окремі сервіси, кожен із яких відповідає за певну функцію. Наприклад, окремо може працювати сервіс авторизації, сервіс оброблення заявок, сервіс сповіщень, сервіс звітності. Такий підхід добре підходить для великих корпоративних систем, де необхідно забезпечити масштабування, незалежне оновлення окремих компонентів і складну інтеграцію з іншими програмними продуктами. Проте для бакалаврської роботи така архітектура може бути надмірною, якщо система має обмежений набір функцій.

Близьким до сервісно орієнтованого підходу є мікросервісна архітектура.

Вона передбачає ще більш дрібний поділ системи на незалежні сервіси, які можуть розроблятися, розгортатися та масштабуватися окремо. Такий підхід є сучасним і широко використовується у великих інформаційних системах, але для розроблення прототипу системи обліку заявок він є занадто складним. Використання мікросервісів вимагало б додаткового опису контейнеризації, взаємодії сервісів, мережевих запитів, керування конфігураціями та моніторингу. Тому в межах цієї роботи доцільніше зосередитися на клієнт-серверній або багаторівневій архітектурі.

Для узагальнення основних підходів до побудови веборієнтованих інформаційних систем доцільно подати їх порівняльну характеристику в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика підходів до побудови веборієнтованих інформаційних систем

Підхід до побудови системи	**Сутність підходу**	**Переваги**	**Обмеження**	**Доцільність для системи обліку заявок**
Монолітна архітектура	Усі основні компоненти системи реалізуються в межах одного програмного застосунку	Простота розроблення, зручність початкового тестування, зрозуміла структура проєкту	Ускладнення підтримки при розширенні системи, обмежена гнучкість масштабування	Може бути використана для невеликого прототипу системи
Клієнт-серверна архітектура	Клієнтська частина відповідає за взаємодію з користувачем, серверна – за оброблення запитів і роботу з даними	Централізоване зберігання даних, доступ через браузер, зручність оновлення системи	Потребує налаштування серверної частини та захисту обміну даними	Є найбільш доцільною для розроблюваної системи
Багаторівнева архітектура	Система поділяється на рівень подання, рівень бізнес-логіки та рівень даних	Чіткий поділ відповідальності, зручність супроводу, можливість подальшого розвитку	Потребує якісного проєктування структури системи	Доцільна для опису архітектури системи в роботі
REST API	Взаємодія клієнта і	Гнучкість, можливість	Може бути складнішим для	Доцільний як перспективний

	сервера здійснюється через HTTP-запити до програмного інтерфейсу	інтеграції, відокремлення клієнтської та серверної частин	початкової реалізації	або частково використаний підхід
Сервісно орієнтована архітектура	Система складається з окремих сервісів, що виконують визначені функції	Масштабованість, можливість інтеграції з іншими системами	Надмірна складність для невеликої системи	Може розглядатися як напрям подальшого розвитку
Мікросервісна архітектура	Система поділяється на незалежні малі сервіси, кожен із яких має власну функціональну область	Висока масштабованість, незалежне розгортання компонентів	Складність розроблення, тестування та адміністрування	Недоцільна для базового прототипу бакалаврської роботи

Як видно з таблиці 1.2, для системи обліку заявок користувачів найбільш обґрунтованим є поєднання клієнт-серверного та багаторівневого підходів. Клієнт-серверна архітектура визначає загальну логіку взаємодії користувача із системою через браузер і сервер, а багаторівнева архітектура дозволяє внутрішньо структурувати програмний продукт на рівень інтерфейсу, рівень оброблення логіки та рівень даних. Це дає змогу створити систему, яка є достатньо простою для реалізації в межах бакалаврської роботи, але водночас має правильну технічну структуру.

Для інформаційної системи особливо важливим є рівень подання, оскільки саме через нього користувач взаємодіє із заявками. Інтерфейс має бути зрозумілим, логічним і не перевантаженим зайвими елементами. На сторінці створення заявки користувач повинен швидко ввести тему, опис, категорію та за потреби пріоритет. На сторінці перегляду заявок мають бути доступні фільтри за статусом, датою, категорією або виконавцем. Для виконавця важливо бачити список призначених заявок, а для адміністратора – загальну картину роботи системи.

Рівень бізнес-логіки є центральним елементом системи, оскільки саме він

визначає правила опрацювання заявки. Наприклад, після створення заявка автоматично отримує статус «нова». Після призначення виконавця вона може перейти до статусу «у роботі». Після виконання завдання статус змінюється на «виконана» або «закрита». Якщо інформації недостатньо, заявка може отримати статус «очікує уточнення». Такі переходи між статусами повинні бути не випадковими, а логічно визначеними. Саме тому ще на етапі проєктування необхідно описати алгоритм оброблення заявки.

Рівень даних забезпечує збереження всієї інформації, необхідної для роботи системи. До таких даних належать облікові записи користувачів, їхні ролі, заявки, категорії, статуси, коментарі, історія змін. База даних повинна бути спроєктована так, щоб уникати дублювання інформації, забезпечувати зв’язки між таблицями та підтримувати цілісність даних. Наприклад, кожна заявка має бути пов’язана з користувачем, який її створив, і за потреби з виконавцем, якому вона призначена. Також доцільно передбачити окрему таблицю для історії змін, щоб фіксувати дії із заявкою.

Значення має і вибір технологій для реалізації веборієнтованої системи. Для клієнтської частини можуть використовуватися HTML, CSS і JavaScript. HTML забезпечує структуру сторінок, CSS відповідає за оформлення, а JavaScript може використовуватися для динамічної взаємодії з користувачем, перевірки введених даних або оновлення окремих елементів сторінки. Серверна частина може бути реалізована за допомогою PHP, Node.js, Python, Java або інших технологій. Для бази даних можуть використовуватися MySQL, PostgreSQL, SQLite або інші системи керування базами даних.

У межах цієї бакалаврської роботи доцільно обрати технології, які дозволяють реалізувати систему без надмірного ускладнення. Якщо основна мета полягає в розробленні зрозумілого прототипу веборієнтованої інформаційної системи, доречним є використання HTML, CSS, JavaScript, PHP та MySQL. Такий стек є достатнім для створення форми авторизації, сторінки реєстрації заявки, списку заявок, механізму зміни статусів, бази даних і адміністративних функцій. Крім того, ці технології добре підходять для

пояснення в тексті бакалаврської роботи, оскільки їх функції легко розмежувати.

Водночас можна зазначити, що система в майбутньому може бути модернізована за допомогою сучасніших підходів. Наприклад, клієнтську частину можна реалізувати на основі React, Vue.js або Angular, серверну частину – на Node.js або Python, а обмін даними організувати через REST API. Проте для початкового варіанта системи важливо не стільки використати найскладніші технології, скільки забезпечити правильну архітектуру, коректну роботу з базою даних і зрозумілу логіку взаємодії користувача із системою.

Побудова веборієнтованої інформаційної системи також потребує врахування вимог до безпеки. Оскільки система передбачає роботу з користувачами, заявками та ролями, необхідно реалізувати механізм авторизації. Користувачі повинні входити до системи за допомогою логіна та пароля, а після входу отримувати доступ лише до тих функцій, які відповідають їхній ролі. Наприклад, звичайний користувач може створювати власні заявки й переглядати їхній стан, виконавець – працювати з призначеними заявками, адміністратор – керувати користувачами та контролювати всі заявки.

Крім авторизації, важливо передбачити перевірку введених даних. Якщо користувач створює заявку, система повинна перевірити, чи заповнені обов’язкові поля, чи не містить введена інформація некоректних символів, чи відповідає вона очікуваному формату. На практиці така перевірка може здійснюватися як на клієнтській стороні, так і на серверній. Клієнтська перевірка підвищує зручність користування, оскільки користувач одразу бачить помилки, але серверна перевірка є обов’язковою з погляду безпеки, оскільки дані не можна вважати надійними лише через те, що вони були перевірені у браузері.

Ще одним важливим аспектом є підтримка зручності супроводу системи. Навіть невелика веборієнтована інформаційна система повинна мати логічну структуру файлів, зрозуміле розміщення модулів і можливість подальшого розширення. Наприклад, сторінки роботи із заявками, файли підключення до бази даних, обробники авторизації та адміністративні функції доцільно відокремлювати один від одного. Це спрощує пошук помилок, тестування та

внесення змін.

Для наочності загальну логіку клієнт-серверної взаємодії у веборієнтованій інформаційній системі обліку заявок користувачів можна подати у вигляді схеми.

Рисунок 1.2 – Загальна логіка клієнт-серверної взаємодії у веборієнтованій інформаційній системі

На рисунку 1.2 показано, що робота веборієнтованої системи не зводиться лише до відображення сторінки у браузері. За кожною дією користувача стоїть послідовність технічних операцій: формування запиту, передавання його на сервер, оброблення логіки, звернення до бази даних, формування відповіді та відображення результату. Наприклад, коли користувач створює заявку, він заповнює форму у вебінтерфейсі. Після натискання кнопки дані передаються на сервер, перевіряються, записуються до бази даних, після чого користувач отримує повідомлення про успішне створення заявки. Такий процес має бути стабільним, захищеним і зрозумілим для подальшого тестування.

Клієнт-серверна взаємодія буде проявлятися в усіх основних операціях: авторизації користувача, створенні заявки, перегляді списку заявок, зміні статусу, призначенні виконавця, фільтрації та пошуку. Кожна з цих операцій передбачає обмін даними між користувацьким інтерфейсом, серверною частиною та базою даних. Саме тому архітектура системи повинна бути спроєктована таким чином, щоб ці процеси виконувалися послідовно та без втрати даних.

Окремо варто наголосити на значенні бази даних у веборієнтованій інформаційній системі. Якщо інтерфейс забезпечує зручність роботи користувача, а серверна логіка відповідає за правила оброблення запитів, то база даних є основою збереження інформації. Без правильно спроєктованої бази

даних система обліку заявок не зможе забезпечити надійне збереження звернень, статусів, ролей і результатів виконання. У літературі з проєктування баз даних зазначається, що «якість структури бази даних безпосередньо впливає на цілісність інформації, швидкість її пошуку та можливість подальшого розвитку інформаційної системи» [18, с. 74].

Для системи обліку заявок структура бази даних має відображати основні сутності предметної області. До них належать користувачі, ролі, заявки, категорії, статуси, коментарі та історія змін. Кожна сутність має власний набір атрибутів. Наприклад, заявка може містити ідентифікатор, тему, опис, дату створення, статус, пріоритет, ідентифікатор користувача та ідентифікатор виконавця. Користувач може мати ідентифікатор, ім’я, електронну пошту, пароль і роль. Такий підхід дозволяє надалі перейти від загального аналізу до конкретного проєктування бази даних у другому розділі.

Під час вибору підходу до побудови системи необхідно також ураховувати майбутнє тестування. Якщо система має зрозумілу архітектуру, її легше перевіряти. Наприклад, можна окремо протестувати авторизацію, створення заявки, зміну статусу, фільтрацію, обмеження доступу для різних ролей. Це важливо для третього розділу роботи, де потрібно буде показати, що розроблений програмний продукт не лише описаний теоретично, а й перевірений на працездатність за визначеними сценаріями.

Отже, аналіз сучасних підходів до побудови веборієнтованих інформаційних систем показав, що для розроблення системи обліку заявок користувачів найбільш доцільним є використання клієнт-серверної архітектури з елементами багаторівневої організації програмного продукту. Такий підхід дозволяє розмежувати інтерфейс користувача, бізнес-логіку та базу даних, забезпечити централізоване збереження інформації, підтримати роботу різних ролей користувачів і створити основу для подальшого розвитку системи. У межах цієї бакалаврської роботи саме така логіка побудови буде використана як базова для подальшого проєктування та реалізації веборієнтованої інформаційної системи обліку заявок користувачів.

1.3 Формування вимог до інформаційної системи обліку заявок користувачів

Проєктування будь-якої інформаційної системи повинно починатися не безпосередньо з написання програмного коду, а з визначення її призначення, користувачів, функцій, обмежень і вимог до майбутньої роботи. Для системи обліку заявок користувачів це має особливе значення, оскільки така система повинна забезпечувати не лише збереження інформації про звернення, а й підтримку повного процесу їх опрацювання: від створення заявки до її закриття та збереження історії виконання.

Вимоги до інформаційної системи можна розглядати як сукупність умов, функцій і характеристик, яким має відповідати програмний продукт для виконання поставлених завдань. Вони визначають, що саме повинна робити система, як вона має взаємодіяти з користувачем, які дані повинна зберігати, які обмеження має враховувати та яким критеріям якості повинна відповідати. У програмній інженерії вимоги є основою подальшого проєктування, адже саме вони визначають архітектуру, структуру бази даних, набір функціональних модулів, сценарії роботи користувачів і методи тестування.

У спеціальній літературі зазначається, що «вимоги до програмної системи описують сервіси, які система має надавати, а також обмеження, у межах яких вона повинна функціонувати» [71]. Це положення є важливим для теми цієї роботи, оскільки система обліку заявок повинна розглядатися не лише як набір вебсторінок, а як програмний продукт із визначеною логікою роботи, правилами доступу, структурою даних і очікуваними результатами взаємодії користувачів із системою.

Формування вимог до веборієнтованої інформаційної системи обліку заявок користувачів доцільно здійснювати з урахуванням кількох ключових аспектів. По-перше, необхідно визначити функції, які має виконувати система. По-друге, потрібно описати ролі користувачів і межі їх доступу. По-третє, слід визначити, які дані повинні зберігатися в системі. По-четверте, необхідно

врахувати вимоги до зручності інтерфейсу, безпеки, продуктивності та можливості подальшого розвитку програмного продукту.

Основними користувачами системи є три групи: звичайний користувач, виконавець і адміністратор. Звичайний користувач створює заявку, описує проблему або потребу, переглядає стан її виконання та за потреби уточнює інформацію. Виконавець отримує заявку, переглядає її зміст, змінює статус, додає результат виконання або коментар. Адміністратор керує користувачами, контролює всі заявки, може змінювати статуси, призначати виконавців, переглядати загальну статистику та підтримувати працездатність системи.

Такий розподіл ролей є важливим, оскільки система не повинна надавати всім користувачам однаковий рівень доступу. Наприклад, звичайний користувач не повинен мати можливості змінювати чужі заявки, призначати виконавців або редагувати облікові записи інших користувачів. Виконавець не повинен мати повного адміністративного доступу до налаштувань системи. Адміністратор, навпаки, повинен мати розширені права для контролю загальної роботи системи. Отже, ще на етапі формування вимог необхідно передбачити механізм авторизації та розмежування доступу.

Функціональні вимоги визначають, які саме дії має виконувати система. Для інформаційної системи обліку заявок до таких вимог належать реєстрація користувачів, авторизація, створення заявки, перегляд списку заявок, фільтрація, пошук, зміна статусу, призначення відповідального виконавця, редагування окремих даних, збереження історії змін і формування базової звітності. Ці вимоги безпосередньо пов’язані з основним призначенням системи, тому вони мають бути враховані під час проєктування функціональних модулів.

Нефункціональні вимоги характеризують не стільки конкретні дії системи, скільки якість її роботи. До них належать зручність інтерфейсу, швидкість оброблення запитів, надійність збереження даних, безпечність доступу, сумісність із сучасними браузерами, простота супроводу та можливість подальшого розширення. У контексті бакалаврської роботи нефункціональні вимоги мають особливе значення, оскільки вони показують, що система

проєктується не лише як формальний прототип, а як програмний продукт, який повинен бути зрозумілим, стабільним і придатним до використання.

Для узагальнення основних вимог до інформаційної системи обліку заявок користувачів доцільно подати їх у таблиці 1.3.

Таблиця 1.3 – Функціональні та нефункціональні вимоги до інформаційної системи обліку заявок користувачів

Група вимог	**Вимога**	**Зміст вимоги**	**Очікуваний результат реалізації**
Функціональні вимоги	Авторизація користувачів	Система повинна забезпечувати вхід користувача за логіном і паролем	Доступ до системи отримують лише зареєстровані користувачі
Функціональні вимоги	Розмежування ролей	Система повинна підтримувати ролі користувача, виконавця та адміністратора	Кожна група користувачів отримує лише дозволені функції
Функціональні вимоги	Створення заявки	Користувач повинен мати можливість створити заявку із зазначенням теми, опису та категорії	Заявка зберігається в базі даних і отримує початковий статус
Функціональні вимоги	Перегляд заявок	Система повинна відображати список заявок відповідно до ролі користувача	Користувач бачить власні заявки, виконавець – призначені, адміністратор – усі
Функціональні вимоги	Зміна статусу заявки	Виконавець або адміністратор повинен мати можливість змінювати стан заявки	Процес виконання заявки стає контрольованим
Функціональні вимоги	Призначення виконавця	Адміністратор повинен мати можливість закріпити заявку за відповідальною особою	Забезпечується персональна відповідальність за виконання
Функціональні вимоги	Пошук і фільтрація	Система повинна підтримувати пошук і фільтрацію заявок за статусом, датою, категорією або виконавцем	Спрощується робота з великою кількістю звернень
Функціональні вимоги	Збереження історії змін	Система повинна фіксувати основні дії, пов’язані із заявкою	Можна відстежити послідовність опрацювання заявки
Нефункціональні вимоги	Зручність інтерфейсу	Інтерфейс має бути зрозумілим, логічним і	Користувач може швидко виконувати

		не перевантаженим зайвими елементами	основні дії
Нефункціональні вимоги	Швидкодія	Система повинна забезпечувати оперативне відкриття сторінок і оброблення типових запитів	Зменшується час очікування під час роботи із заявками
Нефункціональні вимоги	Безпека	Система повинна захищати доступ до даних за допомогою авторизації та перевірки прав	Знижується ризик несанкціонованого доступу
Нефункціональні вимоги	Цілісність даних	Дані про заявки, користувачів і статуси повинні зберігатися узгоджено	Зменшується ризик втрати або дублювання інформації
Нефункціональні вимоги	Масштабованість	Структура системи повинна дозволяти подальше додавання нових функцій	Система може бути розширена без повної переробки архітектури
Нефункціональні вимоги	Сумісність	Система повинна коректно працювати в сучасних веббраузерах	Забезпечується доступність програмного продукту для різних користувачів

Як показано в таблиці 1.3, вимоги до системи охоплюють не лише основні операції із заявками, а й характеристики якості програмного продукту. Такий підхід є важливим, оскільки навіть правильно реалізовані функції можуть бути недостатніми, якщо система буде незручною, повільною, небезпечною або складною для супроводу. Наприклад, можливість створення заявки є базовою функцією, але вона повинна супроводжуватися перевіркою введених даних, збереженням інформації в базі даних і відображенням зрозумілого повідомлення для користувача.

Окрему групу становлять інформаційні вимоги. Вони визначають, які саме дані повинні зберігатися та оброблятися в системі. Для системи обліку заявок основними інформаційними об’єктами є користувач, роль, заявка, категорія заявки, статус заявки, коментар і запис історії змін. Кожен із цих об’єктів має власне значення. Наприклад, користувач є ініціатором заявки або виконавцем; роль визначає рівень доступу; заявка містить основний опис звернення; статус відображає етап виконання; історія змін дозволяє відстежити дії, виконані із

заявкою.

Інформаційні вимоги мають бути узгоджені зі структурою майбутньої бази даних. Якщо на етапі формування вимог не визначити необхідні сутності, у подальшому можуть виникнути проблеми під час проєктування таблиць і зв’язків між ними. Наприклад, якщо в системі передбачено збереження історії змін, то доцільно створити окрему таблицю, у якій фіксуватимуться дата дії, користувач, тип зміни та ідентифікатор заявки. Якщо ж така вимога не буде закладена на початку, додавання історії змін пізніше може потребувати перероблення структури бази даних.

Інтерфейсні вимоги визначають, яким має бути зовнішній вигляд і логіка взаємодії користувача із системою. Для системи обліку заявок інтерфейс повинен бути максимально зрозумілим. Користувач не повинен витрачати багато часу на пошук кнопки створення заявки, перегляд статусу або відкриття детальної інформації. Основні дії мають бути розміщені логічно, а форми – містити лише необхідні поля. Надмірно складний інтерфейс може знизити ефективність використання системи навіть у тому випадку, якщо її функціональна частина реалізована правильно.

Для звичайного користувача ключовими елементами інтерфейсу є сторінка входу, форма створення заявки, список власних заявок і сторінка перегляду стану заявки. Для виконавця важливими є список призначених заявок, фільтри за статусами, сторінка зміни стану заявки та поле для внесення результату виконання. Для адміністратора необхідні додаткові елементи: перегляд усіх заявок, керування користувачами, призначення виконавців і контроль загального стану системи. Отже, інтерфейс має враховувати не лише загальну зручність, а й різні сценарії роботи користувачів.

Вимоги до безпеки є важливою частиною проєктування веборієнтованої інформаційної системи. Оскільки система працює з обліковими записами користувачів і заявками, необхідно передбачити захист від несанкціонованого доступу. Передусім це стосується авторизації, перевірки ролей, обмеження доступу до адміністративних функцій і контролю коректності введених даних. У

джерелах із веббезпеки підкреслюється, що «будь-які дані, отримані від користувача, повинні розглядатися як потенційно небезпечні до моменту їх перевірки на серверній стороні» [70]. Для розроблюваної системи це означає необхідність перевірки форм створення заявки, авторизації, зміни статусу та інших операцій, які передбачають введення або зміну даних.

Окремо слід ураховувати вимоги до збереження паролів. Паролі користувачів не повинні зберігатися у відкритому вигляді. Навіть у межах навчального прототипу доцільно зазначити, що пароль має зберігатися у хешованому вигляді, а доступ до облікового запису повинен здійснюватися через перевірку введеного пароля з його захищеним представленням у базі даних. Такий підхід демонструє розуміння базових принципів безпечної веброзробки.

Вимоги до тестування також бажано сформувати ще на етапі аналізу. Тестування системи обліку заявок має охоплювати перевірку основних функцій: авторизації, створення заявки, перегляду списку заявок, зміни статусу, призначення виконавця, пошуку, фільтрації та обмеження доступу для різних ролей. Окремо необхідно перевірити коректність роботи з некоректними або неповними даними, наприклад спробу створити заявку без теми або опису. У третьому розділі ці вимоги будуть використані для побудови тестових сценаріїв.

Для систематизації вимог до майбутньої інформаційної системи доцільно подати їх у вигляді узагальненої схеми (див. рис. 1.3). Вона дозволяє показати, що вимоги не є випадковим набором пунктів, а формують взаємопов’язану структуру, яка охоплює функціональність, дані, інтерфейс, безпеку та якість роботи системи.

На рисунку 1.3 відображено основні групи вимог, які мають бути враховані під час проєктування веборієнтованої інформаційної системи обліку заявок користувачів. Центральним елементом є загальні вимоги до системи, які деталізуються на функціональні, нефункціональні, інформаційні, інтерфейсні, безпекові та тестові. Такий поділ є зручним для подальшої роботи, оскільки кожна група вимог буде пов’язана з окремими елементами проєктування. Наприклад, функціональні вимоги визначатимуть модулі системи, інформаційні

– структуру бази даних, інтерфейсні – макети сторінок, а вимоги до тестування – перевірку працездатності системи в третьому розділі.

Рисунок 1.3 – Структура вимог до веборієнтованої інформаційної системи обліку заявок

Після визначення вимог можна перейти до формування загальної логіки майбутньої системи. Вона повинна передбачати, що користувач після авторизації отримує доступ до власного кабінету, створює заявку та може переглядати її стан. Виконавець бачить список призначених заявок і змінює їхній статус відповідно до результатів опрацювання. Адміністратор має ширші повноваження: він контролює всі заявки, керує користувачами, призначає виконавців і може переглядати загальну інформацію про роботу системи. Такий підхід дозволяє забезпечити зрозумілу логіку використання системи для кожної ролі.

Формування вимог також дає змогу уникнути надмірного розширення функціональності. Для бакалаврської роботи важливо не перевантажувати систему функціями, які складно реалізувати та пояснити. Наприклад, на першому етапі можна не включати складну аналітику, інтеграцію з електронною поштою, автоматичні сповіщення, мобільний застосунок або систему

оцінювання якості виконання заявок. Ці функції можна зазначити як перспективні напрями розвитку. Основна ж версія системи повинна стабільно виконувати базові операції: авторизація, створення заявки, перегляд, зміна статусу, призначення виконавця, фільтрація та збереження історії.

З погляду подальшого проєктування, сформовані вимоги будуть використані у другому розділі для обґрунтування вибору технологій, побудови архітектури, визначення функціональних модулів і розроблення структури бази даних. Зокрема, вимога щодо розмежування ролей вплине на структуру таблиці користувачів і механізм авторизації; вимога щодо історії змін – на створення окремої таблиці журналу дій; вимога щодо фільтрації – на реалізацію відповідних запитів до бази даних; вимога щодо зручності інтерфейсу – на побудову сторінок системи.

Таким чином, формування вимог є важливим етапом підготовки до проєктування веборієнтованої інформаційної системи обліку заявок користувачів. У межах цього підпункту визначено основні групи вимог, які повинна враховувати майбутня система: функціональні, нефункціональні, інформаційні, інтерфейсні, безпекові та тестові. Їх урахування дозволяє перейти від загального розуміння предметної області до конкретного проєктування програмного продукту. Саме на основі сформованих вимог у наступному розділі буде здійснено вибір технологій, опис архітектури, розроблення структури бази даних та алгоритмів роботи системи.

Висновки до розділу 1

У першому розділі бакалаврської кваліфікаційної роботи було досліджено предметну область, пов’язану з проєктуванням веборієнтованої інформаційної системи для обліку заявок користувачів. Визначено, що такі системи призначені для впорядкування процесу приймання, реєстрації, опрацювання та контролю виконання звернень, які надходять від користувачів. Їх використання дає змогу уникнути втрати інформації, забезпечити прозорість роботи із заявками,

розподілити відповідальність між виконавцями та зберігати історію виконаних дій.

У межах розділу встановлено, що основними функціями системи обліку заявок є створення заявки, її класифікація, призначення статусу, закріплення відповідального виконавця, перегляд і пошук заявок, редагування даних, збереження історії змін та розмежування ролей користувачів. Узагальнена схема процесу оброблення заявки показала, що робота системи має охоплювати повний життєвий цикл звернення – від його створення до закриття й архівування історії опрацювання.

Також було проаналізовано сучасні підходи до побудови веборієнтованих інформаційних систем. З’ясовано, що для розроблюваної системи найбільш доцільним є використання клієнт-серверної архітектури з елементами багаторівневої організації. Такий підхід дозволяє розмежувати вебінтерфейс, серверну бізнес-логіку та базу даних, що є важливим для стабільної роботи системи, її подальшого супроводу й розширення.

Окрему увагу приділено формуванню вимог до майбутнього програмного продукту. Визначено функціональні, нефункціональні, інформаційні, інтерфейсні, безпекові вимоги та вимоги до тестування. Сформовані вимоги створюють основу для подальшого проєктування системи, зокрема вибору технологій, побудови архітектури, розроблення структури бази даних, алгоритмів оброблення заявок і сценаріїв тестування.

Отже, результати першого розділу підтверджують актуальність розроблення веборієнтованої інформаційної системи обліку заявок користувачів. Проведений аналіз дав змогу перейти від загального розуміння предметної області до конкретних вимог, які будуть використані у другому розділі під час проєктування архітектури, функціональних модулів, бази даних та алгоритмів роботи системи.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ЗАЯВОК

2.1 Обґрунтування вибору технологій і засобів розроблення системи

Розроблення веборієнтованої інформаційної системи обліку заявок користувачів потребує попереднього визначення технологій і програмних засобів, за допомогою яких буде реалізовано основні функціональні можливості системи. Вибір технологічного стеку є важливим етапом проєктування, оскільки саме від нього залежить структура програмного продукту, спосіб взаємодії користувача із системою, організація серверної логіки, збереження даних, можливість подальшого супроводу та розширення системи.

У межах цієї бакалаврської кваліфікаційної роботи розроблювана система розглядається як веборієнтований програмний продукт, доступ до якого здійснюється через веббраузер. Такий підхід відповідає поставленій меті, оскільки система обліку заявок повинна бути доступною для різних груп користувачів: звичайних користувачів, виконавців і адміністратора. При цьому кожна група користувачів має працювати з єдиним інформаційним середовищем, але мати різні права доступу до функцій системи.

Вибір технологій для створення такої системи має ґрунтуватися на кількох практичних критеріях. По-перше, технології повинні забезпечувати можливість створення зручного вебінтерфейсу. По-друге, вони мають підтримувати оброблення запитів на серверній стороні, оскільки саме серверна частина відповідає за авторизацію, перевірку даних, зміну статусів заявок, призначення виконавців і взаємодію з базою даних. По-третє, обрані засоби повинні дозволяти працювати з реляційною базою даних, адже система обліку заявок передбачає збереження структурованої інформації про користувачів, ролі, заявки, статуси, категорії та історію змін.

Крім того, для бакалаврської роботи важливо, щоб обраний технологічний стек був достатньо зрозумілим для опису, реалізації й подальшого захисту. Надто складні технології можуть ускладнити пояснення логіки системи, особливо якщо вони потребують додаткової інфраструктури, контейнеризації, складних налаштувань або розподіленої архітектури. Тому в межах цієї роботи доцільно обрати технології, які дозволяють створити повноцінний прототип веборієнтованої системи, але не перевантажують розробку зайвою технічною складністю.

Для реалізації інформаційної системи обліку заявок користувачів розглянуто кілька поширених варіантів серверних технологій: PHP, Node.js, Python із використанням фреймворку Django або Flask, а також Java з використанням Spring. Кожен із цих варіантів може бути застосований для створення веборієнтованої системи, однак їх доцільність у межах конкретної бакалаврської роботи є різною. Для порівняння можливих засобів розроблення доцільно подати їх характеристику в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика технологій для розроблення веборієнтованої інформаційної системи

Технологія	**Загальна характеристика**	**Переваги**	**Обмеження**	**Доцільність використання в роботі**
PHP	Серверна мова програмування, яка широко використовується для створення вебзастосунків і взаємодії з базами даних	Простота розгортання, зрозуміла логіка роботи з формами, добра підтримка MySQL, придатність для навчальних і прикладних вебпроєктів	За відсутності правильної структури код може ставати складним для супроводу	Доцільна для реалізації прототипу системи обліку заявок
Node.js	Середовище виконання JavaScript на серверній стороні, яке дозволяє створювати вебсервери та API	Єдина мова для клієнтської та серверної частин, зручність створення REST API, сучасний підхід до веброзробки	Потребує кращого розуміння асинхронної моделі, налаштування залежностей і структури проєкту	Може бути використана, але для цієї роботи є дещо складнішою

Python / Django	Засоби веброзробки на основі Python із готовими інструментами для роботи з моделями, маршрутизацією та адміністративною частиною	Високий рівень структурованості, вбудовані механізми безпеки, зручність роботи з базами даних	Потребує вивчення фреймворку та специфічної структури проєкту	Доцільна для складніших систем, але не є обов'язковою для прототипу
Java / Spring	Потужний стек для розроблення корпоративних вебзастосунків	Надійність, масштабованість, придатність для великих інформаційних систем	Висока складність, значний обсяг налаштувань, надмірність для невеликого прототипу	Недоцільна для цієї роботи через надмірну складність
HTML, CSS, JavaScript	Базові технології клієнтської частини вебзастосунків	Забезпечують структуру сторінок, оформлення та динамічну взаємодію з користувачем	Не реалізують повноцінну серверну логіку без додаткових засобів	Обов'язкові для створення інтерфейсу системи
MySQL	Реляційна система керування базами даних	Зручність роботи з таблицями, зв'язками, запитами, підтримка PHP, достатня продуктивність для прототипу	Потребує правильного проєктування структури таблиць	Доцільна для збереження даних системи

Як показано в таблиці 2.1, для реалізації системи обліку заявок користувачів можуть бути використані різні технології. Проте з урахуванням мети бакалаврської роботи, очікуваного обсягу функцій і потреби в зрозумілій реалізації найбільш доцільним є використання поєднання HTML, CSS, JavaScript, PHP та MySQL. Такий технологічний стек дозволяє реалізувати всі основні компоненти системи: вебінтерфейс, серверну обробку запитів, авторизацію користувачів, роботу з базою даних, створення й редагування заявок, зміну статусів, пошук, фільтрацію та збереження історії змін.

HTML доцільно використовувати для формування структури сторінок системи. За його допомогою створюються форми авторизації, форма створення заявки, таблиці перегляду заявок, кнопки керування, поля введення, елементи навігації та інші компоненти вебінтерфейсу. У межах розроблюваної системи

HTML забезпечуватиме основу для взаємодії користувача з програмним продуктом.

CSS використовується для оформлення сторінок системи. Його застосування дозволяє зробити інтерфейс більш зручним, упорядкованим і зрозумілим. Для системи обліку заявок важливо, щоб користувач міг швидко знайти форму створення заявки, переглянути список звернень, визначити поточний статус і перейти до потрібної дії. Тому оформлення інтерфейсу має не лише естетичне, а й функціональне значення.

JavaScript може бути використаний для підвищення зручності взаємодії користувача із системою. Наприклад, за його допомогою можна реалізувати попередню перевірку заповнення форми, динамічне відображення повідомлень, підтвердження дій або оновлення окремих елементів сторінки без повного перезавантаження. У межах базової версії системи JavaScript не обов’язково має виконувати складну логіку, проте його використання дозволяє зробити роботу з інтерфейсом більш комфортною.

PHP обрано як основний засіб реалізації серверної логіки. Саме серверна частина має обробляти дані, які надходять від користувача, перевіряти коректність введеної інформації, виконувати авторизацію, звертатися до бази даних, створювати нові заявки, змінювати статуси, призначати виконавців і формувати відповіді для інтерфейсу. PHP добре підходить для такої задачі, оскільки має просту логіку роботи з вебформами, підтримує взаємодію з MySQL і дозволяє реалізувати повноцінний вебзастосунок без надмірно складної інфраструктури.

MySQL доцільно використати як систему керування базами даних. Система обліку заявок має працювати зі структурованими даними, між якими існують логічні зв’язки. Наприклад, кожна заявка пов’язана з користувачем, який її створив; за потреби – з виконавцем; кожна заявка має певний статус і категорію; зміни в заявці можуть фіксуватися в історії. Реляційна модель даних добре підходить для такого типу інформації, оскільки дозволяє створювати таблиці, визначати первинні та зовнішні ключі, забезпечувати цілісність даних і

виконувати запити для пошуку та фільтрації.

На основі проведеного порівняння можна сформувати обґрунтування вибору конкретних програмних засобів для розроблення системи. Це подано в таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору програмних засобів для реалізації системи обліку заявок користувачів

Засіб розроблення	Призначення в системі	Обґрунтування вибору
HTML	Створення структури вебсторінок	Дозволяє сформувати сторінки авторизації, створення заявки, перегляду списку заявок і адміністративних розділів
CSS	Оформлення інтерфейсу користувача	Забезпечує візуальну впорядкованість сторінок, зручність сприйняття інформації та логічне розміщення елементів
JavaScript	Реалізація клієнтської динаміки	Може використовуватися для перевірки форм, підтвердження дій і покращення взаємодії користувача із системою
PHP	Реалізація серверної логіки	Дає змогу обробляти запити, виконувати авторизацію, працювати з базою даних і реалізувати основні функції системи
MySQL	Збереження структурованих даних	Підходить для збереження користувачів, ролей, заявок, статусів, категорій, коментарів та історії змін
phpMyAdmin	Адміністрування бази даних	Дозволяє створювати таблиці, переглядати дані, виконувати SQL-запити й контролювати структуру бази даних
Visual Studio Code	Написання та редагування коду	Забезпечує зручне середовище для роботи з файлами HTML, CSS, JavaScript, PHP та SQL
Веббраузер	Перевірка роботи інтерфейсу	Використовується для тестування сторінок системи, перевірки форм і сценаріїв взаємодії користувача

Дані таблиці 2.2 показують, що обрані засоби розроблення охоплюють усі ключові компоненти майбутньої системи. HTML, CSS і JavaScript забезпечують клієнтську частину, PHP реалізує серверну логіку, MySQL відповідає за збереження даних, а допоміжні інструменти використовуються для написання коду, адміністрування бази даних і тестування вебінтерфейсу. Такий підхід є достатнім для створення функціонального прототипу інформаційної системи обліку заявок користувачів.

Важливо підкреслити, що вибір саме такого стеку не означає відмову від

сучасних підходів до веброзробки. Навпаки, система проєктується з урахуванням загальної клієнт-серверної логіки, поділу на інтерфейс, серверну обробку та базу даних. Тобто навіть за використання відносно простих технологій архітектура програмного продукту залишається зрозумілою, структурованою і придатною до подальшого розвитку.

У межах розроблюваної системи клієнтська частина відповідатиме за відображення форм і сторінок, через які користувач взаємодіє із заявками. Серверна частина відповідатиме за оброблення запитів, перевірку прав доступу, виконання операцій із заявками та звернення до бази даних. База даних забезпечуватиме збереження всієї інформації, необхідної для функціонування системи. Такий поділ дозволяє уникнути змішування різних рівнів логіки та спрощує подальше тестування програмного продукту.

При виборі технологій також враховано характер самої системи. Інформаційна система обліку заявок не потребує складних математичних обчислень, оброблення великих масивів даних у реальному часі або використання штучного інтелекту. Її основна задача полягає в надійному збереженні структурованої інформації, організації доступу користувачів відповідно до ролей і забезпеченні зрозумілого процесу опрацювання заявок. Саме тому обраний технологічний стек є достатнім і виправданим.

Окремо слід зазначити, що обрана технологічна база дозволяє реалізувати подальше розширення системи. Наприклад, у майбутньому до неї можна додати модуль електронних повідомлень, формування звітів, експорт даних, панель аналітики, API для інтеграції з іншими сервісами або адаптивний інтерфейс для мобільних пристроїв. При цьому базова структура системи – користувачі, ролі, заявки, статуси, категорії та історія змін – залишатиметься актуальною.

З погляду безпеки обрані технології також дозволяють реалізувати необхідні базові механізми. На рівні PHP можна виконувати перевірку введених даних, обмеження доступу відповідно до ролі користувача, захищене збереження паролів у вигляді хешів і контроль сесій. На рівні MySQL можна забезпечити структуроване збереження інформації та зв’язки між таблицями. На рівні

інтерфейсу можна передбачити повідомлення про помилки, обов’язковість заповнення полів і зрозумілу навігацію.

Таким чином, для реалізації веборієнтованої інформаційної системи обліку заявок користувачів обґрунтовано вибір технологічного стеку HTML, CSS, JavaScript, PHP та MySQL. Обрані засоби дозволяють створити повноцінний прототип системи, що включає користувацький інтерфейс, серверну логіку, базу даних, авторизацію, розмежування ролей, створення й опрацювання заявок. Такий вибір є раціональним для бакалаврської кваліфікаційної роботи, оскільки поєднує практичну реалізованість, зрозумілість опису, технічну достатність і можливість подальшого розвитку програмного продукту.

2.2 Проєктування архітектури, функціональних модулів і ролей користувачів

Після визначення технологічної основи розроблення необхідно перейти до проєктування архітектури веборієнтованої інформаційної системи обліку заявок користувачів. Архітектура програмної системи визначає загальну організацію її компонентів, зв’язки між ними, розподіл функцій, порядок обміну даними та логіку взаємодії користувачів із програмним продуктом. Для системи обліку заявок архітектурне проєктування має особливе значення, оскільки вона повинна одночасно забезпечувати зручний інтерфейс, коректну серверну обробку, збереження даних, розмежування ролей і контроль життєвого циклу заявки.

Веборієнтована інформаційна система обліку заявок користувачів будується за клієнт-серверною логікою. Клієнтська частина забезпечує взаємодію користувача із системою через веббраузер. Серверна частина приймає запити, перевіряє дані, виконує бізнес-логіку, звертається до бази даних і формує відповідь. База даних зберігає інформацію про користувачів, ролі, заявки, статуси, категорії, коментарі та історію змін. Такий підхід дозволяє розмежувати рівень подання, рівень оброблення логіки та рівень збереження інформації.

Загальна архітектура системи передбачає наявність трьох основних груп

користувачів: звичайного користувача, виконавця та адміністратора. Кожна з цих ролей взаємодіє з вебінтерфейсом, однак отримує доступ лише до тих функцій, які відповідають її призначенню. Звичайний користувач створює заявки та переглядає стан їх виконання. Виконавець опрацьовує призначені заявки, змінює їх статуси та вносить результати виконання. Адміністратор контролює роботу системи, керує користувачами, призначає виконавців і має доступ до загального переліку заявок.

Архітектуру системи доцільно подати як сукупність взаємопов’язаних рівнів (див. рис. 2.1). На першому рівні розміщується користувацький інтерфейс, який забезпечує введення, перегляд і зміну інформації. На другому рівні перебуває серверна логіка, що реалізує правила роботи із заявками, перевірку прав доступу та оброблення запитів. На третьому рівні розташована база даних, у якій зберігається вся інформація, необхідна для функціонування системи.

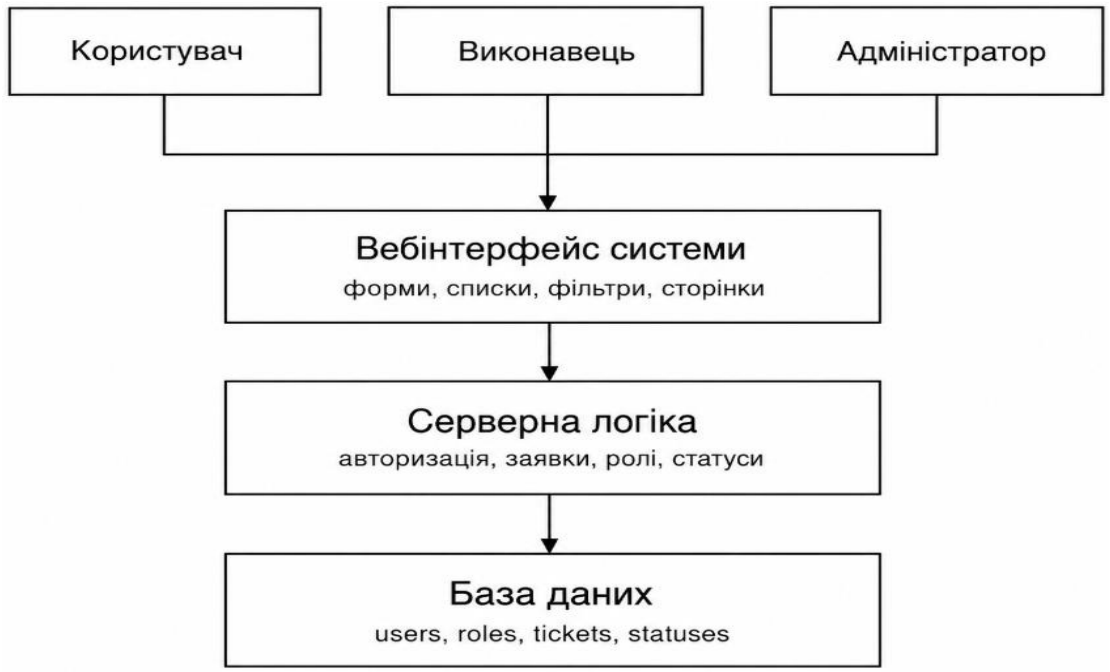

Рисунок 2.1 – Архітектура веборієнтованої інформаційної системи обліку заявок користувачів

На рисунку 2.1 показано загальну архітектуру веборієнтованої інформаційної системи обліку заявок користувачів. Її основою є взаємодія трьох категорій користувачів із єдиним вебінтерфейсом. Усі дії, які виконуються через

інтерфейс, передаються на серверну частину, де відбувається перевірка прав доступу, оброблення запиту та звернення до бази даних. Така архітектура забезпечує централізоване збереження інформації та дозволяє підтримувати єдину логіку роботи із заявками.

Першим важливим компонентом системи є модуль авторизації. Його призначення полягає в перевірці особи користувача та наданні доступу до системи відповідно до ролі. Користувач вводить логін або електронну пошту та пароль, після чого серверна частина перевіряє ці дані в базі. Якщо дані правильні, для користувача створюється сесія, а система визначає його роль. Саме роль впливає на те, які сторінки й функції будуть доступні після входу.

Модуль авторизації є базовим для всієї системи, оскільки без нього неможливо забезпечити розмежування доступу. Якщо користувач не пройшов авторизацію, він не повинен мати можливості створювати, переглядати або змінювати заявки. Якщо користувач має роль звичайного користувача, він не повинен отримувати доступ до адміністративних функцій. Якщо користувач має роль виконавця, йому мають бути доступні заявки, призначені для опрацювання. Для адміністратора відкриваються розширені функції контролю та управління.

Другим компонентом є модуль керування користувачами. Він використовується адміністратором для перегляду, додавання, редагування або деактивації облікових записів. У цьому модулі зберігається інформація про ім’я користувача, електронну пошту, пароль у захищеному вигляді, роль і статус облікового запису. Завдяки цьому адміністратор може підтримувати актуальність складу користувачів системи та контролювати, хто має доступ до функцій створення, виконання або адміністрування заявок.

Третім і центральним компонентом є модуль створення заявок. Через цей модуль звичайний користувач формує нову заявку, зазначаючи її тему, опис, категорію та за потреби пріоритет. Після надсилання форми серверна частина перевіряє заповнення обов’язкових полів, записує заявку до бази даних і присвоює їй початковий статус. Зазвичай таким статусом є «нова». З цього моменту заявка стає доступною для подальшого перегляду та опрацювання.

Модуль створення заявок повинен бути максимально простим для користувача. Надмірна кількість полів може ускладнити подання звернення, а недостатня кількість інформації може створити проблеми для виконавця. Тому форма створення заявки має містити лише необхідні реквізити: тему, опис, категорію, можливо – пріоритет або прикріплення додаткової інформації. У базовій структурі достатньо передбачити текстовий опис проблеми, оскільки саме він є основним джерелом інформації для подальшого опрацювання.

Наступним компонентом є модуль перегляду заявок. Його функція полягає у відображенні переліку заявок відповідно до ролі користувача. Звичайний користувач бачить власні заявки. Виконавець бачить заявки, які призначені йому або доступні для опрацювання. Адміністратор бачить повний перелік заявок у системі. Така логіка дозволяє зберегти цілісність інформації та водночас не перевантажувати користувача зайвими даними.

Модуль перегляду заявок має включати можливість фільтрації та сортування. Наприклад, заявки можуть відображатися за статусом, датою створення, категорією, пріоритетом або відповідальним виконавцем. Для адміністратора така функція особливо корисна, оскільки дозволяє швидко оцінити загальний стан заявок: скільки з них нових, скільки перебувають у роботі, скільки виконано або закрито. Для виконавця фільтрація допомагає зосередитися на актуальних завданнях.

Окремим компонентом є модуль зміни статусу заявки. Статус заявки відображає її поточний стан у процесі опрацювання. У системі доцільно передбачити такі основні статуси: «нова», «у роботі», «очікує уточнення», «виконана», «закрита». Зміна статусу повинна бути доступною лише тим користувачам, які мають відповідні права. Наприклад, виконавець може перевести заявку в стан «у роботі» або «виконана», а адміністратор може закрити заявку або змінити відповідального виконавця.

Статуси не повинні змінюватися випадково або без логіки. Система має підтримувати послідовність переходів. Наприклад, нова заявка спочатку може бути передана в роботу, потім виконана, а вже після цього закрита. Якщо даних

недостатньо, заявка може отримати статус «очікує уточнення». Такий підхід дозволяє уникнути хаотичного опрацювання звернень і створює зрозумілий життєвий цикл заявки.

Важливе місце займає модуль призначення виконавця. Він забезпечує закріплення конкретної заявки за відповідальною особою. Така функція насамперед потрібна адміністратору, який аналізує нові заявки та визначає, хто має їх опрацьовувати. Призначення виконавця дозволяє персоналізувати відповідальність і спростити контроль виконання. У базі даних це реалізується через поле, яке пов’язує заявку з користувачем, що має роль виконавця.

Ще одним компонентом є модуль коментарів або уточнень. Він може використовуватися для обміну додатковою інформацією між користувачем і виконавцем. Наприклад, якщо опис заявки є недостатньо точним, виконавець може поставити уточнювальне запитання, а користувач – надати відповідь. Така функція дозволяє не створювати нові заявки для кожного уточнення, а зберігати всю комунікацію в межах одного звернення.

Для забезпечення прозорості роботи системи необхідний модуль історії змін. Він фіксує основні дії, які виконуються із заявкою: створення, призначення виконавця, зміну статусу, додавання коментаря, закриття заявки. Історія змін є важливою не лише для контролю, а й для подальшого аналізу. Вона дозволяє зрозуміти, хто і коли виконував певні дії, як змінювався стан заявки та на якому етапі виникали затримки.

Загальна структура функціональних модулів системи може бути подана як взаємозв’язок окремих блоків, кожен із яких відповідає за певну частину роботи програмного продукту (див. таблицю 2.3). Такий поділ дає змогу уникнути змішування функцій і спростити подальшу реалізацію.

Як показано в таблиці 2.3, кожен функціональний модуль має власне призначення, але всі вони пов’язані між собою через загальну логіку роботи із заявками. Наприклад, модуль створення заявок формує новий запис у базі даних, модуль перегляду забезпечує доступ до цього запису, модуль зміни статусу оновлює його стан, а модуль історії змін фіксує виконані дії. Завдяки такому

поділу система стає більш зрозумілою для реалізації, тестування та подальшого супроводу.

Таблиця 2.3 – Функціональні модулі веборієнтованої інформаційної системи обліку заявок користувачів

Функціональний модуль	**Основне призначення**	**Основні операції**
Модуль авторизації	Забезпечення входу користувача до системи та визначення його ролі	Перевірка логіна і пароля, створення сесії, обмеження доступу
Модуль керування користувачами	Адміністрування облікових записів користувачів	Додавання, редагування, деактивація користувачів, призначення ролей
Модуль створення заявок	Формування нових заявок користувачами	Заповнення форми, перевірка даних, запис заявки до бази даних
Модуль перегляду заявок	Відображення списку заявок відповідно до ролі користувача	Перегляд, сортування, фільтрація, пошук заявок
Модуль зміни статусу	Керування поточним станом заявки	Переведення заявки між статусами, фіксація зміни в історії
Модуль призначення виконавця	Закріплення заявки за відповідальною особою	Вибір виконавця, оновлення даних заявки, повідомлення про зміну
Модуль коментарів	Уточнення інформації щодо заявки	Додавання коментарів, перегляд уточнень, збереження повідомлень
Модуль історії змін	Фіксація дій, виконаних із заявкою	Запис подій, збереження дати, користувача та опису дії
Адміністративний модуль	Загальний контроль роботи системи	Перегляд усіх заявок, керування користувачами, контроль статусів

Особливу увагу під час проєктування необхідно приділити ролям користувачів. Роль визначає не лише набір доступних функцій, а й загальну логіку взаємодії людини із системою. У системі обліку заявок передбачено три основні ролі: користувач, виконавець і адміністратор.

Користувач є ініціатором звернення. Його основна задача – створити заявку та надати необхідний опис проблеми або потреби. Після створення заявки користувач може переглядати її стан, бачити зміну статусу, отримувати уточнення або результат виконання. При цьому він не має права змінювати

статус заявки самостійно, призначати виконавця або переглядати заявки інших користувачів.

Виконавець є особою, яка безпосередньо опрацьовує заявку. Він отримує доступ до призначених йому заявок, аналізує їхній зміст, за потреби уточнює інформацію, змінює статус і додає результат виконання. Виконавець не повинен мати доступу до повного адміністрування системи, оскільки його роль пов’язана саме з виконанням заявок, а не з керуванням усією системою.

Адміністратор має найширші права. Він контролює роботу системи, переглядає всі заявки, призначає виконавців, керує користувачами, може змінювати статуси та аналізувати загальний стан звернень. Адміністраторська роль потрібна для підтримання впорядкованості роботи системи, особливо тоді, коли кількість заявок і користувачів зростає.

Розподіл функціональних можливостей між ролями користувачів подано в таблиці 2.4.

Таблиця 2.4 – Ролі користувачів та їх функціональні можливості в системі

Функціональна можливість	**Користувач**	**Виконавець**	**Адміністратор**
Авторизація в системі	+	+	+
Створення заявки	+	–	+
Перегляд власних заявок	+	+	+
Перегляд усіх заявок	–	–	+
Перегляд призначених заявок	–	+	+
Зміна статусу заявки	–	+	+
Призначення виконавця	–	–	+
Додавання коментаря до заявки	+	+	+
Перегляд історії змін	обмежено	+	+
Керування користувачами	–	–	+
Деактивація облікового запису	–	–	+
Формування загального переліку заявок	–	–	+

Дані таблиці 2.4 показують, що система має чітко розмежовувати права доступу. Такий розподіл захищає інформацію від некоректного редагування та забезпечує логічну організацію роботи. Наприклад, користувач не може самостійно змінювати статус заявки, оскільки це порушило б контроль виконання. Виконавець може працювати із заявками, але не керує обліковими

записами. Адміністратор виконує функції контролю та налаштування.

Для кращого розуміння взаємодії користувачів із системою доцільно використати UML-діаграму варіантів використання. Вона дозволяє показати, які актори взаємодіють із системою та які дії вони можуть виконувати. У цій діаграмі акторами є користувач, виконавець і адміністратор, а варіантами використання – основні функції системи.

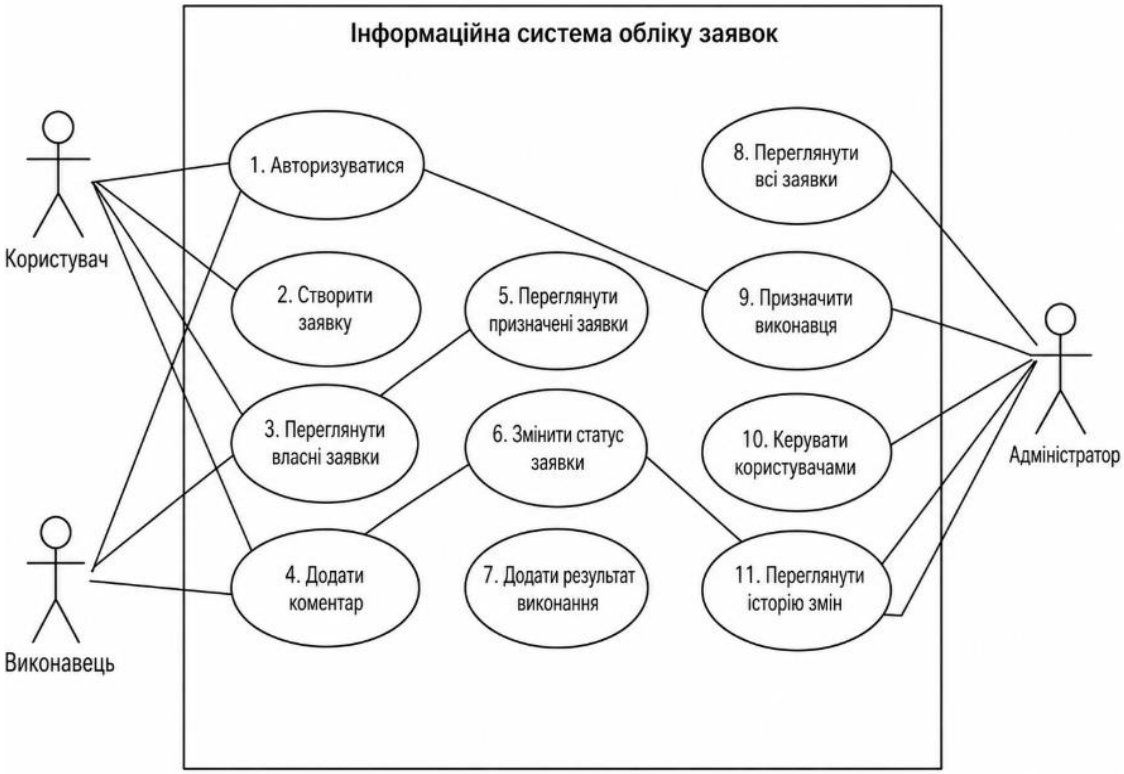

Рисунок 2.2 – UML-діаграма варіантів використання інформаційної системи обліку заявок

На рисунку 2.2 подано основні варіанти використання інформаційної системи обліку заявок. Діаграма показує, що всі ролі мають спільну дію – авторизацію, однак подальший набір можливостей залежить від статусу користувача в системі. Звичайний користувач працює насамперед зі своїми заявками, виконавець – із призначеними завданнями, а адміністратор – із загальним управлінням системою.

Проєктування варіантів використання дозволяє уточнити межі системи. Наприклад, система не передбачає вільного доступу до заявок без авторизації, не дозволяє звичайному користувачу змінювати статуси, не надає виконавцю

повноважень керувати обліковими записами. Такі обмеження є важливими для подальшої реалізації механізму доступу.

Загальна логіка роботи користувача із системою може бути описана через послідовність основних сценаріїв. Перший сценарій стосується створення заявки. Користувач входить до системи, відкриває форму створення заявки, заповнює необхідні поля, надсилає форму, після чого система перевіряє дані та створює запис у базі. Заявка отримує початковий статус і стає доступною для подальшого опрацювання.

Другий сценарій пов’язаний з опрацюванням заявки виконавцем. Виконавець авторизується, відкриває список призначених заявок, обирає потрібну заявку, аналізує її опис, змінює статус на «у роботі», виконує необхідні дії, додає результат і переводить заявку в статус «виконана». Якщо інформації недостатньо, заявка може бути переведена в статус «очікує уточнення».

Третій сценарій стосується адміністрування. Адміністратор входить до системи, переглядає всі заявки, визначає нові або проблемні звернення, призначає виконавців, контролює зміну статусів і за потреби керує обліковими записами користувачів. Такий сценарій забезпечує загальну керованість системи та дозволяє уникати ситуацій, коли заявки залишаються без відповідального виконавця.

Під час проєктування архітектури також необхідно врахувати структуру сторінок системи. До основних сторінок належать сторінка авторизації, головна сторінка після входу, сторінка створення заявки, сторінка списку заявок, сторінка детального перегляду заявки, сторінка зміни статусу, адміністративна сторінка керування користувачами та сторінка перегляду історії змін. Кожна сторінка повинна мати чітке призначення й бути пов’язана з певним функціональним модулем.

Сторінка авторизації забезпечує вхід до системи. Головна сторінка після входу може відображати коротку інформацію про доступні дії або список актуальних заявок. Сторінка створення заявки містить форму введення даних. Сторінка списку заявок відображає таблицю з основними параметрами: номер

заявки, тема, категорія, статус, дата створення, виконавець. Сторінка детального перегляду містить повний опис заявки, коментарі та історію змін. Адміністративна сторінка використовується для керування користувачами та контролю всієї системи.

Важливим принципом проєктування є узгодженість функціональних модулів зі структурою бази даних. Наприклад, модуль авторизації використовує таблиці користувачів і ролей; модуль створення заявок – таблицю заявок, категорій і статусів; модуль історії змін – окрему таблицю журналу дій; модуль коментарів – таблицю коментарів. Завдяки цьому кожен модуль має чітку інформаційну основу, а структура бази даних не є випадковою.

Також необхідно передбачити логіку повідомлень для користувача. Після створення заявки система повинна повідомити про успішне збереження. Якщо обов’язкові поля не заповнено, користувач має отримати зрозуміле повідомлення про помилку. Якщо доступ до певної функції заборонений, система повинна не просто блокувати дію, а пояснювати, що користувач не має відповідних прав. Такі повідомлення підвищують зручність роботи та зменшують кількість помилкових дій.

Архітектура системи має підтримувати подальше розширення. Наприклад, до неї можна додати модуль сповіщень, модуль звітності, експорт даних, панель статистики або інтеграцію з електронною поштою. Щоб це було можливим, уже на етапі проєктування потрібно уникати надмірного змішування логіки. Авторизація, заявки, користувачі, статуси, коментарі та історія змін мають бути представлені як окремі логічні частини системи.

Таким чином, архітектура веборієнтованої інформаційної системи обліку заявок користувачів базується на клієнт-серверній моделі з розмежуванням інтерфейсу, серверної логіки та бази даних. Система включає модулі авторизації, керування користувачами, створення заявок, перегляду, зміни статусу, призначення виконавця, коментарів, історії змін та адміністрування. Визначення ролей користувачів і їхніх функціональних можливостей забезпечує контроль доступу, упорядкованість роботи із заявками та підготовку до подальшого

проєктування структури бази даних і алгоритмів оброблення заявок.

2.3 Розроблення структури бази даних та алгоритмів оброблення заявок

Після визначення архітектури, функціональних модулів і ролей користувачів необхідно перейти до проєктування структури бази даних та алгоритмічної логіки оброблення заявок. Саме база даних забезпечує збереження інформації про користувачів, ролі, заявки, статуси, категорії, коментарі та історію змін. Алгоритми, своєю чергою, визначають порядок виконання основних операцій у системі: авторизації, створення заявки, перевірки введених даних, призначення виконавця, зміни статусу та фіксації виконаних дій.

Для веборієнтованої інформаційної системи обліку заявок користувачів доцільним є використання реляційної бази даних. Такий підхід пояснюється тим, що основні інформаційні об’єкти системи мають чітко визначену структуру та перебувають між собою у логічних зв’язках. Наприклад, користувач може створити одну або кілька заявок; кожна заявка має певний статус; заявка може належати до певної категорії; виконавець може бути закріплений за кількома заявками; історія змін пов’язується з конкретною заявкою та конкретним користувачем, який виконав певну дію.

У межах розроблюваної системи база даних має виконувати кілька основних завдань. По-перше, вона повинна забезпечувати надійне збереження облікових записів користувачів і їхніх ролей. По-друге, база даних має зберігати всі заявки, що створюються в системі, разом із їхніми основними реквізитами. По-третє, вона повинна підтримувати зв’язок заявки зі статусом, категорією, автором і виконавцем. По-четверте, необхідно передбачити збереження історії змін, оскільки для системи обліку заявок важливо знати не лише поточний стан звернення, а й послідовність його опрацювання.

Основними сутностями предметної області є користувач, роль, заявка, категорія заявки, статус заявки, коментар та історія змін. Кожна із цих сутностей

має власне призначення в системі. Сутність користувача відображає особу, яка працює із системою. Роль визначає доступні функції. Заявка є центральною сутністю, навколо якої побудовано основну логіку програмного продукту. Категорія допомагає класифікувати звернення. Статус відображає поточний етап опрацювання. Коментарі дозволяють уточнювати інформацію, а історія змін забезпечує прозорість роботи із заявкою.

Для реалізації зазначеної логіки пропонується така структура бази даних:

roles – таблиця ролей користувачів;

users – таблиця користувачів системи;

categories – таблиця категорій заявок;

statuses – таблиця статусів заявок;

tickets – таблиця заявок;

comments – таблиця коментарів до заявок;

ticket_history – таблиця історії змін заявки.

Така структура дозволяє розмежувати різні типи даних і уникнути дублювання інформації. Наприклад, назви статусів не потрібно зберігати безпосередньо в кожній заявці у текстовому вигляді. Достатньо зберігати посилання на відповідний запис у таблиці statuses. Аналогічно роль користувача зберігається через зв’язок із таблицею roles, а категорія заявки – через зв’язок із таблицею categories. Це робить базу даних більш структурованою та зручною для подальшого розвитку.

Для наочного подання структури бази даних доцільно використати схему, яка показує основні таблиці та зв’язки між ними.

На рисунку 2.3 подано узагальнену структуру бази даних системи. Центральною таблицею є tickets, оскільки саме вона зберігає інформацію про заявки користувачів. Таблиця tickets пов’язана з таблицями users, categories, statuses, comments і ticket_history. Завдяки цьому кожна заявка має автора, за потреби – виконавця, належить до певної категорії, перебуває в певному статусі та може мати коментарі й історію змін. Така структура створює основу для реалізації повного життєвого циклу заявки.

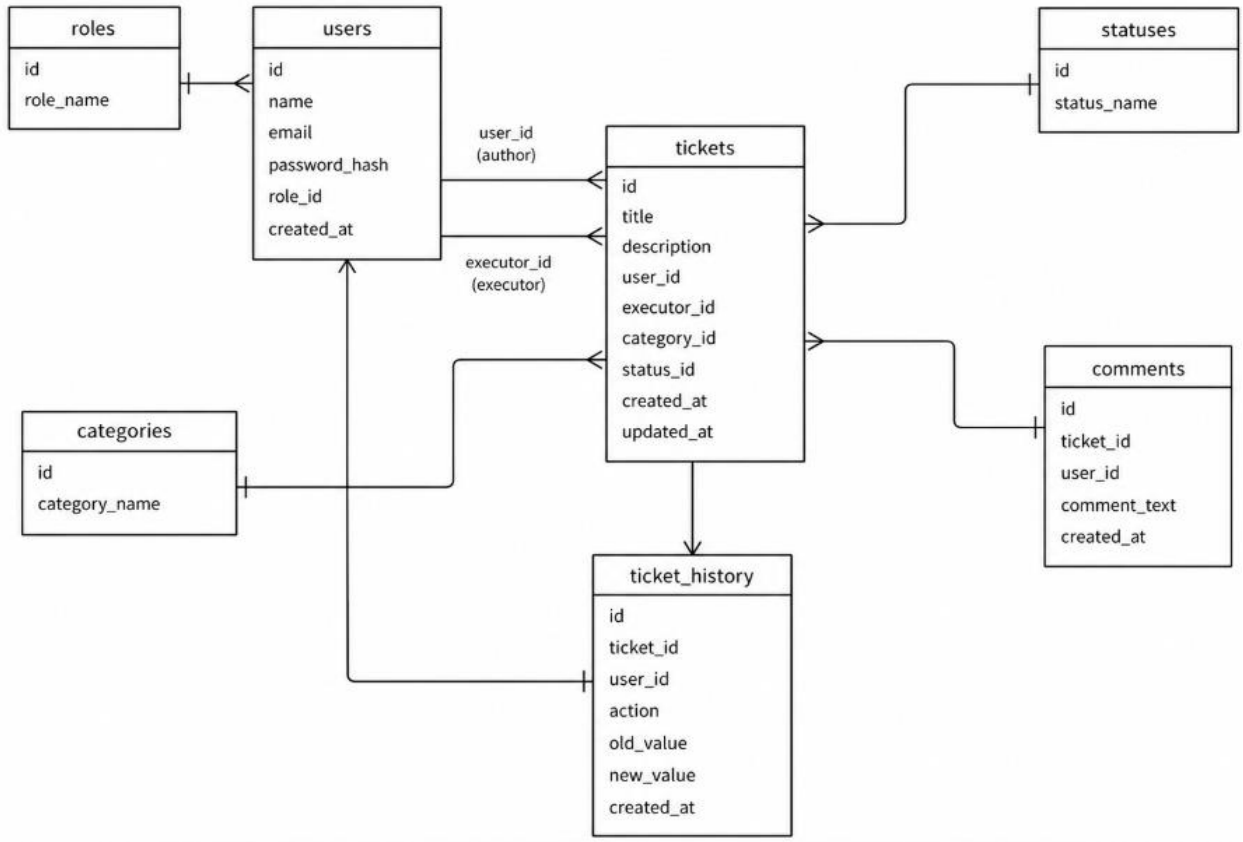

Рисунок 2.3 – Структура бази даних інформаційної системи обліку заявок користувачів

Для деталізації призначення таблиць і їх основних полів доцільно подати опис структури бази даних у табличній формі (див. таблицю 2.5).

Як видно з таблиці 2.5, структура бази даних відображає логіку функціонування системи. Таблиці не існують ізольовано, а утворюють єдину інформаційну модель. Наприклад, таблиця users використовується не лише для авторизації, а й для визначення автора заявки, виконавця, автора коментаря або користувача, який виконав зміну статусу. Таблиця tickets зберігає основну інформацію про заявку, але її повний контекст формується через зв’язки з іншими таблицями.

Особливу увагу слід приділити таблиці tickets, оскільки вона є центральною в системі. У ній зберігається тема заявки, опис, посилання на користувача, який її створив, посилання на виконавця, категорія, статус, дата створення та дата останнього оновлення. Поле user_id відображає автора заявки,

а поле executor_id – відповідального виконавця. Якщо заявка ще не призначена конкретному виконавцю, це поле може бути порожнім. Поле status_id визначає поточний стан заявки, а category_id дозволяє класифікувати звернення.

Таблиця 2.5 – Опис основних таблиць бази даних інформаційної системи обліку заявок

Таблиця	Призначення таблиці	Основні поля	Характер зв'язків
roles	Збереження ролей користувачів системи	id, role_name	Один запис ролі може бути пов'язаний із багатьма користувачами
users	Збереження облікових записів користувачів	id, name, email, password_hash, role_id, created_at	Користувач має одну роль, може створювати заявки, залишати коментарі й виконувати дії
categories	Збереження категорій заявок	id, category_name	Одна категорія може бути пов'язана з багатьма заявками
statuses	Збереження можливих статусів заявки	id, status_name	Один статус може бути пов'язаний із багатьма заявками
tickets	Збереження основної інформації про заявки	id, title, description, user_id, executor_id, category_id, status_id, created_at, updated_at	Центральна таблиця, пов'язана з користувачами, статусами, категоріями, коментарями та історією змін
comments	Збереження коментарів до заявок	id, ticket_id, user_id, comment_text, created_at	Коментар належить конкретній заявці та конкретному користувачу
ticket_history	Фіксація історії змін заявки	id, ticket_id, user_id, action, old_value, new_value, created_at	Запис історії пов'язаний із заявкою та користувачем, який виконав дію

Таблиця ticket_history потрібна для збереження подій, які відбуваються із заявкою. Наприклад, коли користувач створює заявку, у таблиці історії може з'явитися запис про створення. Коли адміністратор призначає виконавця, фіксується відповідна дія. Коли виконавець змінює статус, система зберігає попереднє та нове значення. Завдяки цьому в будь-який момент можна

відстежити, як змінювалася заявка від моменту створення до закриття.

Повний SQL-скрипт створення таблиць бази даних винесено в ДОДАТКИ. В основній частині достатньо подати логіку структури, призначення таблиць і зв’язки між ними. Такий підхід дозволяє зберегти зрозумілість викладу та водночас забезпечити повноту технічного матеріалу.

Крім структури бази даних, важливо описати алгоритмічну логіку оброблення заявки. Алгоритм визначає послідовність дій, які виконує система після того, як користувач створив нову заявку або коли виконавець починає її опрацьовувати. У найпростішому вигляді життєвий цикл заявки охоплює створення, перевірку введених даних, запис у базу, призначення статусу, перегляд адміністратором або виконавцем, зміну статусу, фіксацію історії та завершення опрацювання.

Основний алгоритм оброблення заявки можна описати так:

користувач проходить авторизацію в системі;

система визначає роль користувача;

користувач відкриває форму створення заявки;

користувач заповнює тему, опис і категорію заявки;

система перевіряє коректність введених даних;

якщо дані некоректні, система повертає повідомлення про помилку;

якщо дані коректні, система створює запис у таблиці tickets;

заявці присвоюється початковий статус «нова»;

у таблиці ticket_history фіксується подія створення заявки;

адміністратор переглядає нову заявку та призначає виконавця;

виконавець опрацьовує заявку;

статус заявки змінюється відповідно до результату роботи;

кожна зміна статусу або виконавця фіксується в історії;

після завершення роботи заявка отримує статус «закрита».

Для наочного відображення цього процесу доцільно подати блок-схему алгоритму.

Рисунок 2.4 – Алгоритм оброблення заявки в інформаційній системі

На рисунку 2.4 подано алгоритм оброблення заявки від моменту авторизації користувача до закриття звернення. Ключовим етапом є перевірка коректності введених даних. Якщо користувач не заповнив обов’язкові поля або ввів некоректну інформацію, система не повинна створювати заявку в базі даних, а має повернути повідомлення про помилку. Якщо дані заповнено правильно, заявка зберігається, отримує початковий статус і стає доступною для подальшого опрацювання.

З алгоритмічної точки зору важливо розмежувати дії, які виконує користувач, і дії, які виконує система. Користувач вводить інформацію та надсилає форму. Система перевіряє дані, створює запис, призначає початковий статус, фіксує історію та формує повідомлення про результат. Такий поділ дозволяє чітко визначити відповідальність кожного елемента системи й уникнути ситуації, коли частина логіки залишається невизначеною.

Окремо необхідно описати алгоритм зміни статусу заявки. Статус є одним із головних параметрів заявки, оскільки саме він показує, на якому етапі перебуває звернення. Зміна статусу повинна бути доступною лише користувачам із відповідними правами. Наприклад, звичайний користувач може бачити статус, але не повинен змінювати його самостійно. Виконавець може змінювати статус для заявок, які йому призначені. Адміністратор може змінювати статус будь-якої заявки.

Алгоритм зміни статусу можна подати у такій послідовності:

користувач із відповідною роллю відкриває заявку;

система перевіряє право доступу до зміни статусу;

якщо прав недостатньо, дія блокується;

якщо права підтверджено, користувач обирає новий статус;

система перевіряє допустимість переходу між статусами;

новий статус записується до таблиці tickets;

попередній і новий статуси фіксуються в таблиці ticket_history;

користувач отримує повідомлення про успішну зміну.

У цьому алгоритмі важливим є не лише сам факт оновлення поля status_id, а й фіксація події в історії змін. Без такого запису система зберігатиме тільки поточний стан заявки, але не дозволить відстежити, як саме цей стан змінювався. Для системи обліку заявок це небажано, оскільки контроль виконання значною мірою залежить від прозорості дій.

Ще одним важливим процесом є призначення виконавця. Коли адміністратор відкриває нову заявку, він може обрати відповідального виконавця зі списку користувачів, які мають роль виконавця. Після збереження такого

вибору в таблиці tickets оновлюється поле executor_id. Одночасно в таблицю ticket_history додається запис про те, хто і коли призначив виконавця. Це дозволяє надалі відстежити не лише факт виконання заявки, а й момент її передавання відповідальній особі.

Логіка коментарів також має бути пов’язана з основним життєвим циклом заявки. Коментар може бути доданий користувачем, виконавцем або адміністратором залежно від прав доступу. Кожен коментар пов’язується з конкретною заявкою та конкретним користувачем. Це дає змогу зберігати уточнення, пояснення або проміжні результати без зміни основного опису заявки. У структурі бази даних для цього передбачено таблицю comments.

Важливо, щоб усі основні алгоритми були узгоджені зі структурою бази даних. Наприклад, алгоритм авторизації використовує таблиці users і roles; алгоритм створення заявки – таблиці tickets, categories, statuses і ticket_history; алгоритм коментування – таблиці comments, users і tickets; алгоритм зміни статусу – таблиці tickets, statuses і ticket_history. Така узгодженість забезпечує цілісність проєктного рішення.

Для опису зв’язку між алгоритмами та таблицями бази даних доцільно подати додаткову таблицю.

Таблиця 2.6 – Зв’язок основних процесів системи з таблицями бази даних

Процес системи	**Таблиці, що використовуються**	**Результат виконання процесу**
Авторизація користувача	`users`, `roles`	Визначення особи користувача та його ролі
Створення заявки	`tickets`, `categories`, `statuses`, `ticket_history`	Формування нового запису заявки з початковим статусом
Перегляд списку заявок	`tickets`, `users`, `categories`, `statuses`	Відображення заявок відповідно до ролі користувача
Призначення виконавця	`tickets`, `users`, `ticket_history`	Закріплення заявки за відповідальною особою
Зміна статусу заявки	`tickets`, `statuses`, `ticket_history`	Оновлення стану заявки та фіксація зміни
Додавання коментаря	`comments`, `tickets`, `users`	Збереження уточнення або пояснення до заявки
Перегляд історії змін	`ticket_history`, `tickets`, `users`	Відображення послідовності дій, виконаних із заявкою

Таблиця 2.6 показує, що кожен процес системи має власну інформаційну основу. Це означає, що функціональні модулі, описані в попередньому підпункті, безпосередньо пов’язані з конкретними таблицями бази даних. Такий зв’язок є важливим для подальшої реалізації, оскільки дозволяє чітко визначити, які дані потрібні для кожної операції.

Крім основних процесів, необхідно врахувати правила перевірки даних. Перед створенням заявки система повинна перевіряти, чи заповнено тему, опис і категорію. Перед авторизацією необхідно перевірити наявність користувача з відповідною електронною поштою або логіном. Перед зміною статусу слід перевірити права користувача та існування заявки. Перед додаванням коментаря потрібно переконатися, що заявка існує, а користувач має доступ до неї. Такі перевірки дозволяють уникнути помилок і некоректних записів у базі даних.

Розширені фрагменти програмної реалізації цих перевірок доцільно подати в додатках разом із прикладами PHP-коду та SQL-запитів. В основному тексті достатньо описати логіку перевірки й показати, як вона пов’язана зі структурою системи. Це дозволяє не перевантажувати розділ великими фрагментами коду, але водночас зберегти технічну конкретність розробки.

З погляду подальшої реалізації структура бази даних повинна підтримувати цілісність інформації. Для цього доцільно використовувати первинні ключі, зовнішні ключі та обмеження на обов’язковість окремих полів. Наприклад, кожна заявка повинна мати автора, категорію, статус і дату створення. Коментар не може існувати без заявки, до якої він належить. Запис історії змін також має бути пов’язаний із конкретною заявкою. Така логіка забезпечує узгодженість даних і зменшує ризик появи “відірваних” записів.

Водночас структура бази даних має залишатися придатною до розширення. У майбутньому до системи можна додати таблицю вкладень, таблицю повідомлень, таблицю пріоритетів, таблицю оцінювання якості виконання заявки або таблицю налаштувань системи. Запропонована структура не перешкоджає такому розвитку, оскільки основні сутності вже розмежовані, а центральною залишається таблиця tickets.

Отже, у процесі проєктування структури бази даних визначено основні таблиці, їх призначення та зв’язки між ними. Центральним елементом бази даних є таблиця заявок, яка пов’язується з користувачами, ролями, категоріями, статусами, коментарями та історією змін. Розроблена алгоритмічна логіка передбачає послідовну обробку заявки: авторизацію користувача, створення звернення, перевірку даних, запис у базу, присвоєння статусу, призначення виконавця, зміну стану та фіксацію історії. Це створює технічну основу для подальшої реалізації основних функціональних модулів системи та її тестування.

Висновки до розділу 2

У другому розділі було здійснено проєктування веборієнтованої інформаційної системи обліку заявок користувачів. Обґрунтовано вибір технологічного стеку HTML, CSS, JavaScript, PHP та MySQL, який забезпечує створення клієнтської частини, серверної логіки та реляційної бази даних. Такий набір технологій є достатнім для реалізації основних функцій системи: авторизації користувачів, створення заявок, перегляду, фільтрації, зміни статусів, призначення виконавців і збереження історії змін.

У межах проєктування архітектури визначено основні компоненти системи: вебінтерфейс, серверну логіку та базу даних. Окремо розглянуто ролі користувачів – користувач, виконавець і адміністратор – та встановлено їх функціональні можливості. Такий поділ дозволяє забезпечити розмежування доступу, впорядкувати роботу із заявками та уникнути некоректного втручання користувачів у функції, які не відповідають їхнім повноваженням.

Також розроблено структуру бази даних, центральним елементом якої є таблиця заявок. Вона пов’язується з таблицями користувачів, ролей, категорій, статусів, коментарів та історії змін. Така структура дає змогу зберігати не лише поточний стан заявки, а й повну інформацію про її створення, призначення виконавця, зміну статусу та додаткові уточнення.

Окрему увагу приділено алгоритму оброблення заявки в інформаційній

системі. Він охоплює авторизацію користувача, введення даних, перевірку їх коректності, запис заявки до бази даних, присвоєння початкового статусу, призначення виконавця, опрацювання, зміну статусу та фіксацію дій в історії. Це створює логічну основу для подальшої програмної реалізації системи.

Отже, результати другого розділу сформували технічну основу майбутнього програмного продукту: визначено технології, архітектуру, модулі, ролі користувачів, структуру бази даних та алгоритми роботи із заявками. На цій основі в наступному розділі можна перейти до опису реалізації основних функціональних модулів, інтерфейсу користувача та тестування інформаційної системи.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ЗАЯВОК КОРИСТУВАЧІВ

3.1 Реалізація основних функціональних модулів веборієнтованої системи

Реалізація веборієнтованої інформаційної системи обліку заявок користувачів здійснюється на основі попередньо спроєктованої архітектури, структури бази даних і визначених функціональних модулів. Основна логіка системи полягає у забезпеченні повного циклу роботи із заявкою: від авторизації користувача та створення звернення до його опрацювання, зміни статусу, додавання коментарів, призначення виконавця й збереження історії змін.

Програмна реалізація системи передбачає взаємодію клієнтської частини, серверної логіки та бази даних. Клієнтська частина забезпечує відображення сторінок у веббраузері, приймання даних від користувача та передавання їх на сервер. Серверна частина обробляє запити, перевіряє права доступу, виконує основні операції із заявками та формує відповідь. База даних зберігає облікові записи користувачів, ролі, заявки, статуси, категорії, коментарі та історію змін.

Для реалізації системи використовується логічний поділ програмного продукту на окремі модулі. Такий поділ дозволяє уникнути змішування функцій, спрощує супровід коду та забезпечує зрозумілу структуру системи. Кожен модуль відповідає за окрему групу операцій, але всі вони взаємодіють між собою через спільну базу даних і механізм авторизації користувачів.

Першим базовим модулем є модуль підключення до бази даних. Його призначення полягає у встановленні з’єднання між серверною частиною системи та базою даних MySQL. У цьому модулі задаються параметри підключення: адреса сервера бази даних, ім’я бази, облікові дані користувача та кодування. Наявність окремого файлу підключення дає змогу не дублювати ці параметри в

різних частинах системи. Якщо в майбутньому змінюється назва бази даних або параметри доступу, достатньо змінити один конфігураційний файл.

Окремий модуль підключення до бази даних також підвищує впорядкованість програмної структури. Наприклад, сторінки авторизації, створення заявки, перегляду списку заявок або зміни статусу не повинні самостійно містити повний набір параметрів підключення. Вони звертаються до спільного конфігураційного файлу, після чого використовують підключення для виконання SQL-запитів. Це зменшує кількість повторів у коді та робить структуру проєкту більш зрозумілою.

Наступним важливим компонентом є модуль авторизації користувачів. Його основна функція полягає у перевірці облікових даних користувача та визначенні його ролі в системі. Користувач вводить електронну пошту або логін і пароль у формі входу. Після надсилання форми серверна частина перевіряє наявність такого користувача в таблиці users, порівнює введений пароль із захищеним значенням, що зберігається в базі, і в разі успішної перевірки створює сесію користувача.

Після авторизації система зберігає в сесії основні дані: ідентифікатор користувача, його ім’я та роль. Саме ці дані використовуються для подальшого розмежування доступу. Якщо користувач має роль звичайного користувача, йому доступні створення заявки та перегляд власних звернень. Якщо роль відповідає виконавцю, відкривається список призначених заявок. Якщо користувач є адміністратором, система надає доступ до загального переліку заявок, керування користувачами та призначення виконавців.

Реалізація авторизації має бути пов’язана з перевіркою доступу на кожній сторінці системи. Наприклад, якщо неавторизований користувач намагається відкрити список заявок, система повинна перенаправити його на сторінку входу. Якщо звичайний користувач намагається перейти до адміністративної сторінки, доступ має бути обмежений. Такий підхід забезпечує базовий рівень захисту та не дозволяє користувачам виконувати дії, які не відповідають їхній ролі.

Центральним компонентом системи є модуль створення заявки. Він

забезпечує формування нового звернення користувача та його збереження в базі даних. Після входу до системи користувач відкриває форму створення заявки, де зазначає тему, опис проблеми, категорію та за потреби інші параметри. Після надсилання форми серверна частина перевіряє, чи заповнено обов’язкові поля, чи існує вибрана категорія, чи авторизований користувач має право створювати заявку.

Якщо дані введено коректно, система створює новий запис у таблиці tickets. До цього запису вносяться тема заявки, опис, ідентифікатор користувача, категорія, початковий статус і дата створення. Початковим статусом заявки є статус «нова», оскільки звернення ще не передано виконавцю й не перебуває в роботі. Після створення заявки система також може додати запис до таблиці ticket_history, фіксуючи факт створення звернення.

У разі помилки система не повинна створювати порожній або некоректний запис. Наприклад, якщо користувач не ввів тему заявки або залишив порожнім опис, система повертає повідомлення про необхідність заповнення відповідних полів. Така перевірка є важливою, оскільки база даних повинна зберігати тільки змістовні та придатні до опрацювання заявки. Перевірка може здійснюватися як на клієнтському рівні, так і на серверному, однак серверна перевірка є обов’язковою.

Після створення заявки користувач переходить до модуля перегляду заявок. Цей модуль відображає перелік звернень відповідно до ролі користувача. Для звичайного користувача формується список лише тих заявок, які були створені саме ним. Для виконавця відображаються заявки, призначені йому для опрацювання. Адміністратор отримує доступ до повного списку заявок у системі. Такий підхід забезпечує не лише зручність роботи, а й захист інформації від зайвого перегляду.

Список заявок подається у вигляді таблиці, у якій можуть відображатися номер заявки, тема, категорія, статус, дата створення, виконавець і доступні дії. Для зручності користувача доцільно реалізувати фільтрацію за статусом, категорією або датою створення. Наприклад, виконавець може швидко

переглянути лише заявки зі статусом «у роботі», а адміністратор – лише нові заявки, які ще не призначені відповідальному виконавцю.

Важливе місце займає модуль детального перегляду заявки. На відміну від загального списку, детальна сторінка містить повну інформацію про конкретну заявку: тему, опис, автора, дату створення, категорію, поточний статус, виконавця, коментарі та історію змін. Саме ця сторінка є основним робочим середовищем для виконавця й адміністратора, оскільки через неї здійснюється опрацювання звернення.

Для користувача детальна сторінка дозволяє перевірити, у якому стані перебуває його заявка, чи додано коментар або результат виконання. Для виконавця ця сторінка є джерелом інформації про проблему або запит. Для адміністратора вона дозволяє контролювати повний контекст заявки. Такий підхід робить роботу із заявками більш прозорою та зручною для всіх учасників процесу.

Окремим функціональним блоком є модуль зміни статусу заявки. Статус відображає поточний етап опрацювання звернення, тому його зміна повинна відбуватися за визначеними правилами. Наприклад, після створення заявка має статус «нова». Після призначення виконавця вона може отримати статус «у роботі». Якщо для виконання потрібні додаткові відомості, заявка може перейти до стану «очікує уточнення». Після виконання дій заявка отримує статус «виконана», а після остаточного завершення – «закрита».

Зміна статусу доступна не всім користувачам. Звичайний користувач може переглядати статус, але не повинен самостійно змінювати його. Виконавець може змінювати статус призначених йому заявок. Адміністратор має право змінювати статус будь-якої заявки. Після кожної зміни статусу система оновлює відповідне поле в таблиці tickets і додає запис до таблиці ticket_history. Завдяки цьому зберігається не лише поточний стан заявки, а й історія переходів між статусами.

Ще одним важливим компонентом є модуль призначення виконавця. Він використовується адміністратором для закріплення заявки за конкретною

відповідальною особою. Для цього на сторінці заявки або в адміністративному списку передбачається можливість обрати виконавця зі списку користувачів, які мають відповідну роль. Після збереження вибору система оновлює поле executor_id у таблиці tickets і фіксує цю дію в історії змін.

Призначення виконавця має значення для впорядкування роботи із заявками. Якщо заявка не має відповідального, складно визначити, хто саме повинен її виконувати. Після призначення виконавця система дозволяє відображати заявку в його робочому списку. Це забезпечує персоналізацію відповідальності та спрощує контроль виконання звернень.

Для уточнення інформації використовується модуль коментарів. Він дозволяє користувачам, виконавцям і адміністраторам додавати текстові повідомлення до заявки. Коментарі можуть містити додаткові пояснення, уточнення проблеми, проміжні результати виконання або відповідь виконавця. Усі коментарі зберігаються в таблиці comments і пов’язуються з конкретною заявкою та конкретним користувачем.

Модуль коментарів є корисним, оскільки дозволяє зберігати комунікацію в межах однієї заявки. Це зменшує потребу у використанні зовнішніх каналів, наприклад електронної пошти або месенджерів. Уся інформація, що стосується заявки, залишається в системі та доступна для перегляду відповідними користувачами. Якщо в подальшому потрібно проаналізувати процес виконання, коментарі допомагають зрозуміти, які уточнення або дії виконувалися.

Важливу роль відіграє модуль історії змін. Він забезпечує автоматичну фіксацію основних подій, пов’язаних із заявкою. До таких подій належать створення заявки, призначення виконавця, зміна статусу, додавання коментаря, закриття заявки. Кожен запис історії містить ідентифікатор заявки, ідентифікатор користувача, опис дії, попереднє й нове значення за потреби, а також дату виконання операції.

Історія змін дозволяє простежити повний шлях заявки від створення до завершення. Це особливо важливо для адміністратора, який контролює загальний процес роботи системи. Наприклад, якщо заявка довго перебуває у

статусі «нова», можна перевірити, чи був призначений виконавець. Якщо заявка кілька разів змінювала статус, історія дозволяє побачити послідовність цих змін. Такий механізм підвищує прозорість і керованість системи.

Останнім комплексним блоком є адміністративний модуль. Він об’єднує функції, необхідні для управління системою. До них належать перегляд усіх заявок, призначення виконавців, керування користувачами, зміна статусів, перегляд історії змін і контроль загального стану звернень. Адміністративний модуль доступний лише користувачам із відповідною роллю, що запобігає випадковому або несанкціонованому втручанню в роботу системи.

Для узагальнення реалізованих функціональних модулів доцільно подати їх у таблиці 3.1.

Таблиця 3.1 – Реалізація функціональних модулів інформаційної системи обліку заявок користувачів

Функціональний модуль	**Призначення модуля**	**Основна логіка реалізації**	**Результат роботи**
Модуль підключення до бази даних	Забезпечення зв’язку серверної частини з MySQL	Використання окремого конфігураційного файлу для параметрів підключення	Система отримує доступ до таблиць бази даних
Модуль авторизації	Перевірка користувача та визначення його ролі	Оброблення форми входу, перевірка облікових даних, створення сесії	Користувач отримує доступ до функцій відповідно до ролі
Модуль створення заявки	Формування нового звернення користувача	Перевірка полів форми, запис даних у таблицю `tickets`, присвоєння початкового статусу	У системі створюється нова заявка
Модуль перегляду заявок	Відображення списку заявок	Вибірка даних із бази відповідно до ролі користувача	Користувач бачить лише доступні йому заявки
Модуль детального перегляду	Відображення повної інформації про заявку	Отримання даних про заявку, автора, статус, виконавця, коментарі та історію	Користувач отримує повний контекст звернення
Модуль зміни статусу	Оновлення поточного стану заявки	Перевірка прав доступу, зміна `status_id`, запис події в історію	Статус заявки змінюється контрольовано

Модуль призначення виконавця	Закріплення заявки за відповідальною особою	Вибір виконавця зі списку, оновлення `executor_id`, фіксація дії	Заявка передається конкретному виконавцю
Модуль коментарів	Збереження уточнень і повідомлень до заявки	Додавання запису в таблицю `comments` із прив'язкою до заявки та користувача	Комунікація щодо заявки зберігається в системі
Модуль історії змін	Фіксація ключових дій із заявкою	Автоматичне створення записів у `ticket_history`	Забезпечується прозорість опрацювання заявки
Адміністративний модуль	Управління системою та контроль заявок	Перегляд усіх звернень, керування користувачами, призначення виконавців	Адміністратор контролює роботу системи

Як показано в таблиці 3.1, реалізація системи має модульний характер. Кожен модуль виконує окрему функцію, однак повноцінна робота програмного продукту забезпечується їх взаємодією. Наприклад, створення заявки неможливе без авторизації користувача, зміна статусу пов'язана з перевіркою ролі, а історія змін формується на основі дій, які виконуються в інших модулях. Така структура дозволяє підтримувати цілісну логіку роботи системи.

Загальна структура програмної реалізації може бути подана у вигляді взаємозв'язку основних частин проєкту. Вона включає конфігураційні файли, сторінки авторизації, сторінки роботи із заявками, адміністративні сторінки та файли оформлення інтерфейсу. Повну файлову структуру доцільно навести в додатку, а в основному тексті достатньо показати її узагальнену логіку.

На рисунку 3.1 відображено узагальнену структуру програмної реалізації системи. Вона показує, що програмний продукт не є набором окремих незв'язаних сторінок, а складається з логічних блоків, кожен із яких виконує визначену роль. Конфігураційний блок забезпечує підключення до бази даних, блок авторизації відповідає за вхід і перевірку ролей, блок роботи із заявками реалізує основні операції, адміністративний блок підтримує керування системою, а блок коментарів та історії забезпечує збереження додаткових дій.

Рисунок 3.1 – Структура програмної реалізації інформаційної системи обліку заявок

Фрагменти програмного коду, які реалізують зазначені модулі, подано у ДОДАТКАХ. До таких фрагментів можуть належати код авторизації, створення заявки, зміни статусу, додавання коментаря та перевірки ролі користувача. В основній частині достатньо розкрити логіку їх роботи, оскільки надмірне розміщення коду в тексті ускладнило б сприйняття структури системи.

Загальна реалізація модулів забезпечує повний цикл роботи з інформаційною системою. Користувач може увійти до системи, створити заявку, переглянути її стан і додати коментар. Виконавець може отримати список призначених звернень, опрацювати заявку, змінити її статус і додати результат виконання. Адміністратор може контролювати всі заявки, призначати виконавців і керувати користувачами. Усі дії зберігаються в базі даних, а ключові зміни фіксуються в історії.

Отже, реалізація основних функціональних модулів веборієнтованої системи забезпечує практичне втілення архітектурних і проєктних рішень,

сформованих на попередньому етапі. Модульна структура дозволяє підтримувати розмежування функцій, контроль доступу, збереження даних і прозорість оброблення заявок. Це створює основу для подальшого розгляду користувацького інтерфейсу, сценаріїв взаємодії з системою та тестування програмного продукту.

3.2 Розроблення користувацького інтерфейсу та сценаріїв взаємодії з системою

Користувацький інтерфейс є важливою складовою веборієнтованої інформаційної системи обліку заявок, оскільки саме через нього користувачі виконують основні дії: авторизуються, створюють заявки, переглядають їхній стан, додають коментарі, змінюють статуси або здійснюють адміністративне керування. Навіть за наявності правильно спроєктованої бази даних і серверної логіки система не буде зручною для практичного використання, якщо її інтерфейс є складним, перевантаженим або нелогічним.

Під час розроблення інтерфейсу враховується, що система використовується трьома основними категоріями користувачів: звичайним користувачем, виконавцем і адміністратором. Для кожної ролі має бути передбачено власний набір доступних сторінок і функцій. Водночас загальна стилістика інтерфейсу повинна залишатися єдиною, щоб користувач не сприймав різні частини системи як окремі неузгоджені компоненти.

Основними вимогами до інтерфейсу є простота, логічність, зрозуміла навігація, мінімальна кількість зайвих елементів і чітке розміщення основних дій. Користувач повинен швидко розуміти, де створити заявку, як переглянути її статус, де додати коментар або як повернутися до списку заявок. Для виконавця важливим є швидкий доступ до призначених заявок, а для адміністратора – можливість переглядати загальний стан системи та керувати користувачами.

Розроблення інтерфейсу доцільно розглядати не лише як оформлення сторінок, а як побудову сценаріїв взаємодії з системою. Кожна сторінка має

виконувати конкретну функцію та бути пов’язаною з певним етапом роботи користувача. Наприклад, сторінка авторизації забезпечує вхід до системи; головна сторінка після входу орієнтує користувача в доступних діях; сторінка створення заявки дозволяє сформувати нове звернення; сторінка списку заявок забезпечує перегляд і фільтрацію; сторінка детального перегляду містить повну інформацію про заявку.

Першою сторінкою системи є сторінка авторизації. Вона призначена для введення облікових даних користувача та перевірки його права доступу до системи. На цій сторінці розміщуються поля для введення електронної пошти або логіна, пароля, кнопка входу та повідомлення про помилки у випадку некоректного введення даних. Якщо користувач успішно проходить авторизацію, система визначає його роль і перенаправляє на відповідну головну сторінку.

Сторінка авторизації не повинна містити зайвих елементів, оскільки її основна функція є чіткою та обмеженою. Водночас вона має забезпечувати зрозумілий зворотний зв’язок. Якщо користувач ввів неправильний пароль або не заповнив обов’язкові поля, система повинна повідомити про це без технічних формулювань, які можуть бути незрозумілими. Наприклад, доцільно використовувати повідомлення на зразок: «Невірний логін або пароль» або «Заповніть обов’язкові поля».

Після входу користувач потрапляє на головну сторінку системи. Її зміст залежить від ролі. Для звичайного користувача головна сторінка може містити короткий перелік власних заявок, кнопку створення нової заявки та інформацію про кількість відкритих або виконаних звернень. Для виконавця головна сторінка повинна відображати список призначених заявок, які потребують опрацювання. Для адміністратора доцільно передбачити панель із загальною кількістю заявок за статусами, доступом до користувачів і переліком нових звернень.

Важливим елементом системи є форма створення заявки. Вона повинна бути простою, але достатньо інформативною. Основними полями форми є тема заявки, категорія та опис. Поле теми дозволяє коротко визначити суть звернення,

категорія допомагає класифікувати заявку, а опис містить детальну інформацію, необхідну для подальшого опрацювання. За потреби до форми може бути додано поле пріоритету, однак для базової реалізації достатньо основних реквізитів.

Форма створення заявки повинна містити візуальне позначення обов'язкових полів. Це дозволяє користувачу одразу зрозуміти, які дані необхідно ввести. Після натискання кнопки збереження система перевіряє заповнення полів. Якщо дані введено коректно, заявка зберігається в базі даних, отримує початковий статус і користувач бачить повідомлення про успішне створення. Якщо дані неповні, система повертає користувача до форми з поясненням помилки.

Окрему роль відіграє сторінка списку заявок. Вона є однією з найбільш використовуваних частин інтерфейсу, оскільки саме через неї користувачі переглядають поточний стан звернень. У таблиці списку заявок доцільно відображати номер заявки, тему, категорію, статус, дату створення та відповідального виконавця. Для різних ролей набір заявок у списку відрізняється: звичайний користувач бачить власні заявки, виконавець – призначені йому, адміністратор – усі заявки.

Список заявок має підтримувати фільтрацію та пошук. Наприклад, користувач може відфільтрувати заявки за статусом, щоб переглянути лише відкриті або виконані звернення. Виконавець може обрати заявки, які перебувають у роботі. Адміністратор може швидко знайти нові заявки, що ще не мають виконавця. Така функціональність підвищує зручність роботи, особливо коли кількість заявок збільшується.

Наступною важливою сторінкою є сторінка детального перегляду заявки. Вона містить повний опис звернення, дані автора, поточний статус, категорію, виконавця, дату створення, коментарі та історію змін. Для звичайного користувача ця сторінка дає змогу перевірити стан власної заявки та за потреби додати уточнення. Для виконавця вона є основним робочим простором, де можна ознайомитися з проблемою, додати коментар, змінити статус або внести результат виконання. Для адміністратора сторінка дозволяє контролювати

повний процес опрацювання заявки.

Інтерфейс детальної сторінки має бути структурованим. Основна інформація про заявку повинна розміщуватися у верхній частині сторінки, коментарі – нижче, а історія змін – окремим блоком. Дії, доступні користувачу, мають відображатися залежно від його ролі. Наприклад, звичайний користувач може бачити кнопку додавання коментаря, але не бачить кнопку зміни статусу. Виконавець може змінювати статус призначеної заявки, а адміністратор може додатково призначати або змінювати виконавця.

Для адміністратора передбачається адміністративна панель. Вона забезпечує доступ до функцій керування користувачами, перегляду всіх заявок, призначення виконавців і контролю статусів. Адміністративна панель повинна бути відокремлена від звичайних сторінок користувача, оскільки вона містить функції з підвищеним рівнем доступу. При цьому її інтерфейс має залишатися зрозумілим: основні розділи бажано подавати у вигляді меню або окремих вкладок.

Для узагальнення призначення основних сторінок інтерфейсу доцільно подати таблицю 3.2.

Таблиця 3.2 – Основні сторінки інтерфейсу інформаційної системи обліку заявок користувачів

Сторінка інтерфейсу	**Основне призначення**	**Доступні ролі користувачів**	**Основні елементи сторінки**
Сторінка авторизації	Забезпечення входу до системи	Користувач, виконавець, адміністратор	Поля логіна і пароля, кнопка входу, повідомлення про помилки
Головна сторінка	Відображення початкової інформації після входу	Користувач, виконавець, адміністратор	Короткий список заявок, навігаційне меню, основні дії відповідно до ролі
Сторінка створення заявки	Формування нового звернення	Користувач, адміністратор	Поле теми, категорія, опис, кнопка збереження
Сторінка списку заявок	Перегляд заявок відповідно до ролі	Користувач, виконавець, адміністратор	Таблиця заявок, статуси, фільтри, пошук, кнопка перегляду

Сторінка детального перегляду заявки	Відображення повної інформації про заявку	Користувач, виконавець, адміністратор	Опис заявки, статус, виконавець, коментарі, історія змін
Сторінка зміни статусу	Оновлення поточного стану заявки	Виконавець, адміністратор	Поле вибору статусу, коментар до зміни, кнопка збереження
Адміністративна панель	Керування системою	Адміністратор	Перелік усіх заявок, користувачі, призначення виконавців, контроль статусів
Сторінка керування користувачами	Адміністрування облікових записів	Адміністратор	Список користувачів, ролі, статус облікового запису, дії редагування

Як показано в таблиці 3.2, інтерфейс системи складається з кількох взаємопов’язаних сторінок, кожна з яких має чітке функціональне призначення. Такий підхід дозволяє не перевантажувати одну сторінку надмірною кількістю дій. Наприклад, створення заявки відокремлюється від перегляду списку, а адміністративні функції винесено в окрему панель. Це робить систему зрозумілішою та зручнішою для користувачів із різними ролями.

Важливим елементом інтерфейсу є навігація. Вона має забезпечувати швидкий перехід між основними сторінками системи. Для звичайного користувача достатньо передбачити пункти «Головна», «Створити заявку», «Мої заявки», «Вийти». Для виконавця доцільно передбачити пункти «Головна», «Призначені заявки», «Заявки в роботі», «Вийти». Для адміністратора меню може містити пункти «Усі заявки», «Користувачі», «Нові заявки», «Історія змін», «Вийти».

Сценарій взаємодії звичайного користувача із системою починається з авторизації. Після успішного входу користувач переходить на головну сторінку, де бачить доступні дії. Якщо потрібно створити нову заявку, він відкриває відповідну форму, заповнює поля та надсилає дані. Після збереження заявка з’являється у списку власних звернень. Надалі користувач може відкрити детальну сторінку заявки, переглянути її статус і додати коментар.

Сценарій взаємодії виконавця відрізняється тим, що його основна задача полягає не у створенні заявок, а в їх опрацюванні. Після входу виконавець бачить

список заявок, призначених йому адміністратором. Він відкриває конкретну заявку, аналізує її опис, за потреби додає коментар, змінює статус на «у роботі», а після виконання – на «виконана». Якщо інформації недостатньо, виконавець може залишити уточнення, після чого заявка може отримати статус «очікує уточнення».

Сценарій адміністратора є найбільш широким. Адміністратор після входу отримує доступ до всіх заявок і може контролювати їхній стан. Він переглядає нові заявки, призначає виконавців, змінює статуси, керує користувачами та перевіряє історію змін. Така логіка дозволяє адміністратору не лише виконувати окремі дії, а й підтримувати загальну впорядкованість системи.

Для наочного подання сценаріїв взаємодії доцільно сформувати схему навігації користувача в системі. Вона показує, які сторінки відкриваються після авторизації та які дії доступні різним ролям.

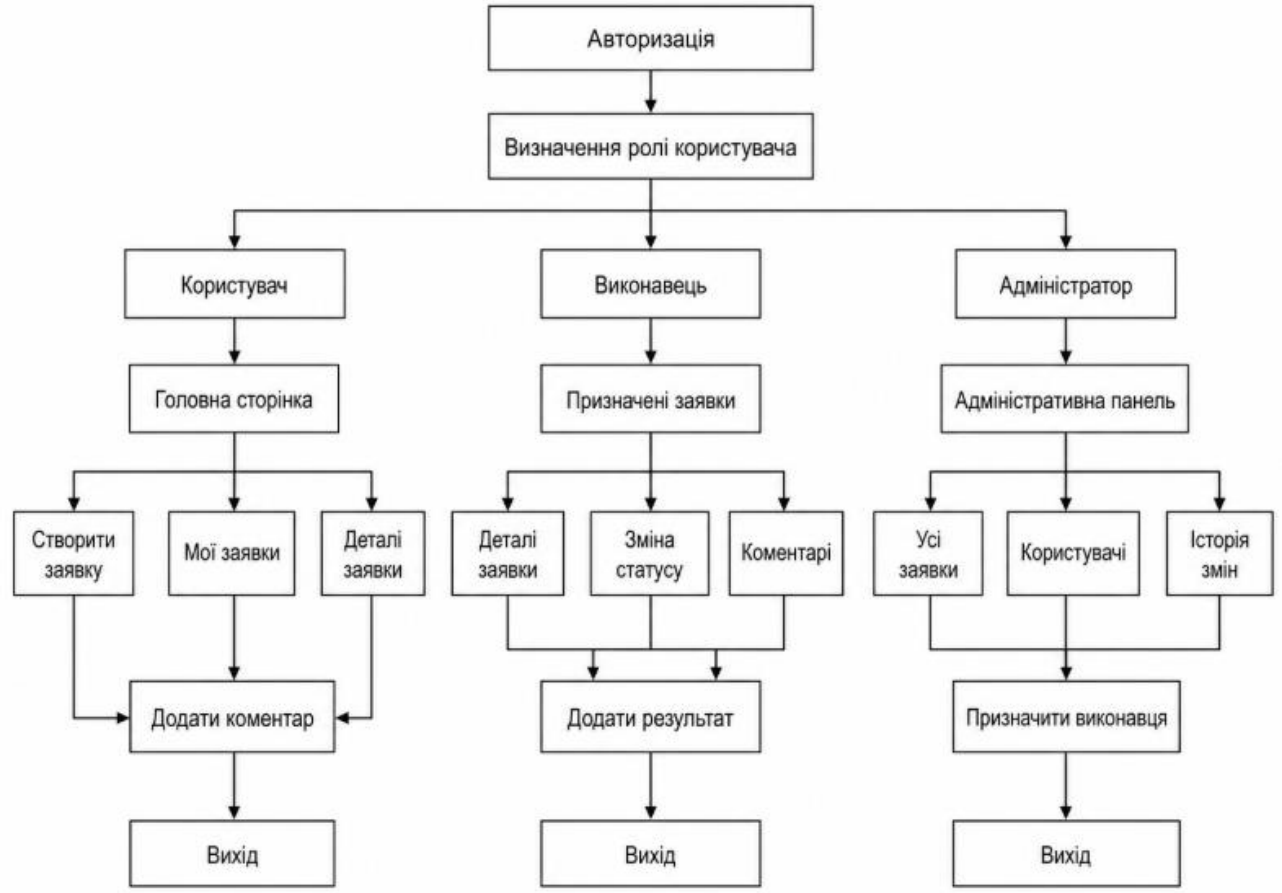

Рисунок 3.2 – Схема навігації користувача в інформаційній системі обліку заявок

На рисунку 3.2 подано узагальнену схему навігації користувача в інформаційній системі обліку заявок. Центральним етапом є авторизація та визначення ролі користувача, після чого система відкриває різні сценарії роботи. Звичайний користувач орієнтований на створення й перегляд власних заявок, виконавець – на опрацювання призначених звернень, адміністратор – на загальне керування системою. Такий підхід дозволяє уникнути перевантаження інтерфейсу та забезпечити доступ лише до тих функцій, які потрібні конкретній ролі.

Для забезпечення зручності роботи в інтерфейсі необхідно передбачити систему повідомлень. Повідомлення мають інформувати користувача про результат виконання дії: успішне створення заявки, збереження коментаря, зміну статусу, помилку введення даних або відсутність прав доступу. Такі повідомлення не повинні бути надто технічними. Їхнє завдання – швидко пояснити користувачу, що сталося та які дії потрібно виконати далі.

Наприклад, після створення заявки доцільно виводити повідомлення «Заявку успішно створено». Якщо користувач не заповнив обов’язкове поле, повідомлення може мати вигляд «Заповніть тему та опис заявки». Якщо користувач намагається відкрити сторінку, до якої він не має доступу, система може повідомити: «Недостатньо прав для виконання цієї дії». Така логіка підвищує зрозумілість інтерфейсу та зменшує кількість помилкових дій.

Окремо потрібно враховувати оформлення статусів заявок. Статус є одним із найважливіших елементів, який користувач бачить у списку або на детальній сторінці. Тому статус має бути відображений зрозуміло й помітно. Наприклад, у таблиці заявок можна передбачити окрему колонку «Статус», де відображатимуться значення «нова», «у роботі», «очікує уточнення», «виконана», «закрита». Це дозволяє швидко оцінити стан звернення без відкриття кожної заявки окремо.

Інтерфейс також повинен враховувати принцип мінімізації зайвих дій. Наприклад, якщо користувач переглядає власну заявку, йому не потрібно бачити кнопки адміністративного керування. Якщо виконавець працює із заявкою, він

має одразу бачити доступні дії: змінити статус, додати коментар або результат виконання. Якщо адміністратор переглядає список нових заявок, поруч із кожною заявкою доцільно розмістити дію призначення виконавця.

Приклади програмної реалізації окремих сторінок інтерфейсу, зокрема форми створення заявки, списку заявок і сторінки детального перегляду, доцільно винести в додатки. В основному тексті подається логіка побудови інтерфейсу та сценарії взаємодії, а в додатках можуть бути наведені фрагменти HTML, CSS і JavaScript-коду, які демонструють практичну реалізацію відповідних сторінок.

Загалом розроблений інтерфейс має забезпечувати безперервний і зрозумілий перехід між діями користувача. Авторизація веде до головної сторінки, головна сторінка – до списку або створення заявки, список – до детального перегляду, а детальна сторінка – до коментарів, зміни статусу або повернення назад. Така логіка відповідає природному сценарію роботи із заявками та забезпечує цілісність взаємодії з системою.

Отже, користувацький інтерфейс інформаційної системи обліку заявок побудовано з урахуванням ролей користувачів, основних сценаріїв роботи та потреби в простій навігації. Визначено основні сторінки системи, їх призначення, доступність для різних ролей і ключові елементи. Запропонована схема навігації дозволяє організувати роботу користувача, виконавця й адміністратора без перевантаження інтерфейсу зайвими функціями. Це створює основу для подальшого тестування програмного продукту та оцінювання відповідності системи сформованим вимогам.

3.3 Тестування програмного продукту та оцінювання результатів розроблення

Після реалізації основних функціональних модулів і користувацького інтерфейсу необхідно виконати тестування інформаційної системи обліку заявок користувачів. Тестування дає змогу перевірити, чи правильно працюють основні

функції системи, чи відповідає її поведінка сформованим вимогам, чи коректно обробляються дані, чи забезпечується розмежування доступу між різними ролями користувачів.

Тестування програмного продукту спрямоване не лише на пошук технічних помилок, а й на перевірку логіки роботи системи загалом. Для системи обліку заявок важливо переконатися, що користувач може створити заявку, виконавець може її опрацювати, адміністратор може призначити відповідального, а всі ключові зміни коректно зберігаються в базі даних. Окрему увагу потрібно приділити перевірці помилкових або неповних дій, оскільки саме вони часто виявляють недоліки у програмній логіці.

Тестування системи доцільно проводити за кількома основними напрямами. Перший напрям пов’язаний із перевіркою авторизації користувачів. Другий – із перевіркою створення, перегляду та редагування заявок. Третій напрям стосується зміни статусів і призначення виконавців. Четвертий – перевірки розмежування ролей. П’ятий – роботи з коментарями та історією змін. Такий підхід дозволяє охопити не окремі випадкові дії, а повний цикл функціонування системи.

Перед початком тестування визначаються тестові сценарії. Кожен сценарій має містити опис дії, очікуваний результат і фактичний результат після виконання перевірки. Якщо фактичний результат відповідає очікуваному, сценарій вважається успішним. Якщо результат відрізняється, це свідчить про потребу в доопрацюванні відповідного модуля.

Основні тестові сценарії перевірки інформаційної системи обліку заявок користувачів подано в таблиці 3.3.

Як показано в таблиці 3.3, тестування охоплює основні сценарії використання системи. Перевірено як стандартні дії користувачів, так і ситуації, пов’язані з помилками або обмеженням доступу. Це дає змогу оцінити не лише працездатність окремих модулів, а й узгодженість системи загалом. Наприклад, створення заявки перевіряється не окремо від бази даних, а разом із присвоєнням початкового статусу та подальшим відображенням у списку заявок.

Таблиця 3.3 – Тестові сценарії перевірки інформаційної системи обліку заявок користувачів

Тестовий сценарій	**Дія користувача**	**Очікуваний результат**	**Фактичний результат**
Авторизація з правильними даними	Користувач вводить коректний логін і пароль	Система відкриває головну сторінку відповідно до ролі користувача	Виконано
Авторизація з неправильним паролем	Користувач вводить неправильний пароль	Система не виконує вхід і показує повідомлення про помилку	Виконано
Спроба доступу без авторизації	Користувач відкриває сторінку заявок без входу	Система перенаправляє на сторінку авторизації	Виконано
Створення заявки з коректними даними	Користувач заповнює тему, категорію й опис заявки	Заявка зберігається в базі даних і отримує статус «нова»	Виконано
Створення заявки без обов’язкових полів	Користувач залишає порожнім поле теми або опису	Система не створює заявку й показує повідомлення про помилку	Виконано
Перегляд власних заявок	Користувач відкриває розділ «Мої заявки»	Відображаються лише заявки, створені цим користувачем	Виконано
Перегляд призначених заявок	Виконавець відкриває список призначених заявок	Відображаються заявки, закріплені за цим виконавцем	Виконано
Перегляд усіх заявок адміністратором	Адміністратор відкриває адміністративну панель	Відображається повний перелік заявок системи	Виконано
Призначення виконавця	Адміністратор обирає виконавця для нової заявки	У заявці оновлюється поле виконавця, дія фіксується в історії	Виконано
Зміна статусу заявки виконавцем	Виконавець змінює статус призначеної заявки	Статус заявки оновлюється, зміна зберігається в історії	Виконано
Спроба зміни статусу звичайним користувачем	Користувач намагається змінити статус заявки	Система блокує дію через недостатній рівень доступу	Виконано
Додавання коментаря до заявки	Користувач або виконавець додає коментар	Коментар зберігається й відображається на сторінці заявки	Виконано
Перегляд історії змін	Адміністратор відкриває історію заявки	Відображається послідовність основних дій із заявкою	Виконано
Фільтрація заявок за статусом	Користувач обирає певний статус у фільтрі	Система відображає лише заявки з вибраним статусом	Виконано
Вихід із системи	Користувач натискає кнопку виходу	Сесія завершується, користувач повертається на сторінку входу	Виконано

Окреме значення має перевірка авторизації та розмежування ролей. Якщо система неправильно визначає роль користувача, це може призвести до некоректного доступу до функцій. Наприклад, звичайний користувач не повинен мати можливості переглядати всі заявки або змінювати їхні статуси. Виконавець не повинен керувати користувачами. Адміністратор, навпаки, повинен мати доступ до розширених функцій. Проведена перевірка показала, що доступ до сторінок і дій залежить від ролі користувача, визначеної під час авторизації.

Під час тестування створення заявки перевірялася коректність роботи форми. Якщо всі обов’язкові поля заповнено правильно, заявка створюється та зберігається в таблиці tickets. Якщо користувач залишає порожнім важливе поле, система не повинна записувати неповні дані в базу. Така перевірка підтверджує, що механізм валідації даних працює правильно та запобігає появі некоректних заявок.

Перевірка зміни статусу заявки показує, чи правильно працює життєвий цикл звернення. Заявка не повинна залишатися статичною після створення. Вона має переходити між станами: «нова», «у роботі», «очікує уточнення», «виконана», «закрита». Кожна зміна статусу повинна не лише оновлювати поточне значення в таблиці заявок, а й фіксуватися в історії змін. Це забезпечує прозорість процесу опрацювання.

Важливим етапом є перевірка роботи коментарів. Коментарі дозволяють зберігати уточнення та проміжні повідомлення без зміни основного опису заявки. Під час тестування перевіряється, чи додається коментар до правильної заявки, чи зберігається автор коментаря, дата створення та текст повідомлення. Коректна робота цього модуля важлива для підтримки комунікації між користувачем, виконавцем і адміністратором.

Також перевіряється робота історії змін. Цей модуль має автоматично фіксувати ключові дії із заявкою: створення, призначення виконавця, зміну статусу, додавання коментаря або закриття заявки. Якщо історія змін працює коректно, адміністратор може відстежити, хто саме виконував дію, коли вона була виконана та які зміни відбулися. Це підвищує контрольованість системи.

Розширений перелік тестових даних, прикладів запитів і фрагментів перевірки окремих функцій можна подати в додатках. В основному тексті достатньо навести узагальнені тестові сценарії та результати їх виконання, щоб не перевантажувати розділ технічними деталями.

Окрім перевірки окремих сценаріїв, доцільно оцінити відповідність реалізованої системи вимогам, сформованим у першому розділі. Це дозволяє показати, наскільки програмний продукт відповідає початковим функціональним, нефункціональним, інформаційним, інтерфейсним і безпековим вимогам.

Таблиця 3.4 – Оцінка відповідності реалізованої системи сформованим вимогам

Група вимог	**Вимога**	**Спосіб реалізації**	**Рівень виконання**
Функціональні вимоги	Авторизація користувачів	Реалізовано форму входу, перевірку облікових даних і створення сесії	Виконано
Функціональні вимоги	Створення заявки	Реалізовано форму створення заявки та запис даних у таблицю `tickets`	Виконано
Функціональні вимоги	Перегляд заявок	Реалізовано списки заявок відповідно до ролі користувача	Виконано
Функціональні вимоги	Зміна статусу заявки	Реалізовано оновлення статусу та запис зміни в історію	Виконано
Функціональні вимоги	Призначення виконавця	Реалізовано вибір виконавця адміністратором і оновлення заявки	Виконано
Функціональні вимоги	Коментарі до заявки	Реалізовано додавання та перегляд коментарів	Виконано
Інформаційні вимоги	Збереження даних про користувачів	Передбачено таблиці `users` і `roles`	Виконано
Інформаційні вимоги	Збереження даних про заявки	Передбачено таблицю `tickets` зі зв’язками з іншими таблицями	Виконано
Інформаційні вимоги	Збереження історії змін	Реалізовано таблицю `ticket_history`	Виконано
Інтерфейсні вимоги	Зрозуміла навігація	Передбачено окремі сторінки для входу, заявок, деталей і адміністрування	Виконано
Інтерфейсні вимоги	Простота форм	Форми містять основні необхідні поля без зайвого перевантаження	Виконано
Безпекові вимоги	Розмежування	Доступ до сторінок і дій	Виконано

	ролей	залежить від ролі користувача	
Безпекові вимоги	Перевірка введених даних	Передбачено перевірку обов'язкових полів під час створення заявки	Виконано
Нефункціональні вимоги	Простота супроводу	Система поділена на логічні модулі та окремі сторінки	Виконано
Нефункціональні вимоги	Можливість подальшого розвитку	Структура бази даних і модульна логіка дозволяють додавати нові функції	Виконано частково

Як видно з таблиці 3.4, основні вимоги до інформаційної системи були реалізовані. Система забезпечує авторизацію, створення й перегляд заявок, зміну статусів, призначення виконавців, додавання коментарів і збереження історії змін. Також реалізовано розмежування ролей, що дозволяє організувати різні сценарії роботи для користувача, виконавця та адміністратора.

Водночас окремі можливості можуть бути розглянуті як напрями подальшого розвитку. Наприклад, система може бути доповнена модулем автоматичних сповіщень, розширеною аналітикою, експортом даних, прикріпленням файлів до заявки, формуванням звітів за період або інтеграцією з електронною поштою. Такі функції не є критичними для базової роботи системи, але можуть підвищити її практичну цінність у разі подальшого використання.

Оцінювання результатів розроблення показує, що створена система відповідає поставленому призначенню. Вона дозволяє організувати процес роботи із заявками у структурованому цифровому середовищі. Замість неформального обміну повідомленнями або ведення окремих таблиць користувачі отримують єдину систему, у якій кожна заявка має автора, опис, категорію, статус, виконавця, коментарі та історію змін.

Практичне значення реалізованої системи полягає в тому, що вона може бути використана як прототип для організації внутрішнього обліку заявок у невеликій установі, навчальному підрозділі, сервісній службі або IT-відділі. Система не потребує складної інфраструктури, оскільки доступ до неї здійснюється через веббраузер, а дані зберігаються централізовано в базі даних. Це робить її зручною для розгортання та подальшого вдосконалення.

До переваг реалізованого рішення можна віднести зрозумілу структуру,

поділ користувачів за ролями, наявність основного життєвого циклу заявки, збереження історії змін і можливість подальшого розширення. Важливо й те, що система побудована на поширених технологіях, які не потребують складних умов експлуатації. Це спрощує її підтримку та модифікацію.

Разом із тим система має певні обмеження. У базовій версії вона не містить розширеної аналітики, автоматичних повідомлень, інтеграції з поштовими сервісами або мобільного застосунку. Такі можливості можуть бути додані на наступних етапах розвитку. Наприклад, модуль сповіщень може автоматично інформувати виконавця про нову заявку, а користувача – про зміну статусу. Модуль аналітики може показувати кількість заявок за період, середній час виконання та навантаження на виконавців.

Загальна оцінка результатів розроблення свідчить, що інформаційна система виконує основні функції, передбачені проєктними рішеннями. Вона забезпечує створення, збереження, перегляд та опрацювання заявок користувачів, підтримує розмежування ролей і дозволяє фіксувати зміни протягом усього життєвого циклу заявки. Результати тестування підтверджують працездатність основних модулів і відповідність системи сформованим вимогам.

Отже, тестування програмного продукту показало, що реалізована веборієнтована інформаційна система обліку заявок користувачів є функціонально придатною для виконання основних завдань. Вона забезпечує структуроване опрацювання звернень, контроль статусів, призначення виконавців, роботу з коментарями та збереження історії змін. Отримані результати створюють підстави для використання системи як робочого прототипу та її подальшого розвитку шляхом додавання нових функціональних можливостей.

Висновки до розділу 3

У третьому розділі було розглянуто реалізацію та тестування веборієнтованої інформаційної системи обліку заявок користувачів. Описано

основні функціональні модулі системи, зокрема модуль підключення до бази даних, авторизації, створення заявок, перегляду звернень, зміни статусу, призначення виконавця, додавання коментарів, збереження історії змін та адміністрування.

Визначено, що реалізація системи має модульну структуру, у якій кожен блок виконує окреме функціональне завдання, але всі компоненти пов’язані між собою через спільну базу даних і механізм авторизації. Такий підхід забезпечує цілісність роботи системи, спрощує її супровід і створює умови для подальшого розширення функціональності. Фрагменти програмного коду, SQL-скрипти та детальну файлову структуру проєкту доцільно подати в додатках.

У розділі також охарактеризовано користувацький інтерфейс системи та основні сценарії взаємодії з нею. Встановлено, що для користувача, виконавця й адміністратора передбачаються різні навігаційні маршрути та функціональні можливості. Це дає змогу уникнути перевантаження інтерфейсу, забезпечити логічність роботи із заявками та надати кожній ролі лише ті функції, які відповідають її призначенню.

Проведене тестування підтвердило працездатність основних модулів системи. Перевірено авторизацію користувачів, створення заявок, перегляд списків, зміну статусів, призначення виконавців, додавання коментарів, збереження історії змін і розмежування доступу між ролями. Результати тестових сценаріїв показали, що система коректно виконує основні операції та відповідає сформованим функціональним, інформаційним, інтерфейсним і безпековим вимогам.

Отже, у третьому розділі підтверджено практичну придатність розробленої інформаційної системи. Вона забезпечує структуроване опрацювання заявок користувачів, контроль їх статусів, взаємодію між учасниками процесу та збереження історії виконаних дій. Подальше вдосконалення системи може бути пов’язане з додаванням автоматичних сповіщень, розширеної аналітики, експорту даних, прикріплення файлів до заявок та інтеграції з поштою.

ВИСНОВКИ

У кваліфікаційній роботі було вирішено завдання проєктування та розроблення веборієнтованої інформаційної системи для обліку заявок користувачів. Необхідність створення такої системи зумовлена потребою в упорядкуванні процесу приймання, реєстрації, опрацювання та контролю виконання звернень, які надходять від користувачів. Використання спеціалізованої інформаційної системи дозволяє замінити неструктуровану комунікацію через усні звернення, електронну пошту або месенджери єдиним цифровим середовищем, у якому кожна заявка має визначений статус, автора, виконавця, історію змін і результат опрацювання.

У першому розділі було досліджено предметну область і визначено призначення інформаційних систем обліку заявок користувачів. Встановлено, що основними функціями таких систем є створення заявки, її класифікація, призначення статусу, закріплення відповідального виконавця, перегляд і пошук звернень, додавання коментарів, збереження історії змін та розмежування ролей користувачів. Також було проаналізовано сучасні підходи до побудови веборієнтованих інформаційних систем і визначено, що для розроблюваного програмного продукту найбільш доцільною є клієнт-серверна архітектура з поділом на вебінтерфейс, серверну логіку та базу даних.

У межах першого розділу також сформовано основні вимоги до майбутньої системи. Визначено функціональні, нефункціональні, інформаційні, інтерфейсні, безпекові вимоги та вимоги до тестування. Це дало змогу перейти від загального опису предметної області до конкретного проєктного бачення системи, у якому передбачено авторизацію користувачів, створення й перегляд заявок, зміну статусів, призначення виконавців, збереження коментарів та історії змін.

У другому розділі було здійснено проєктування веборієнтованої інформаційної системи обліку заявок користувачів. Обґрунтовано вибір технологічного стеку, що включає HTML, CSS, JavaScript, PHP та MySQL.

Зазначені технології забезпечують створення клієнтської частини, реалізацію серверної логіки та збереження структурованих даних у реляційній базі даних. Такий підхід дозволяє реалізувати систему без надмірного ускладнення, зберігаючи при цьому її функціональність, зрозумілу архітектуру та можливість подальшого розвитку.

У процесі проєктування було визначено основні функціональні модулі системи: модуль авторизації, керування користувачами, створення заявок, перегляду звернень, зміни статусу, призначення виконавця, коментарів, історії змін та адміністрування. Окремо розглянуто ролі користувачів – користувача, виконавця та адміністратора – і встановлено межі їхніх функціональних можливостей. Такий розподіл дозволяє забезпечити контроль доступу й уникнути некоректного втручання користувачів у функції, які не відповідають їхній ролі.

Також у другому розділі розроблено структуру бази даних системи. Центральною таблицею визначено таблицю заявок, яка пов’язується з таблицями користувачів, ролей, категорій, статусів, коментарів та історії змін. Така структура дає змогу зберігати не лише поточну інформацію про заявку, а й повний контекст її опрацювання. Розроблений алгоритм оброблення заявки охоплює авторизацію користувача, введення даних, перевірку їх коректності, запис заявки до бази даних, присвоєння початкового статусу, призначення виконавця, зміну статусу, фіксацію дій в історії та закриття заявки.

У третьому розділі було описано реалізацію основних функціональних модулів інформаційної системи. Показано, що програмний продукт має модульну структуру, у якій окремі блоки відповідають за підключення до бази даних, авторизацію, роботу із заявками, адміністративне керування, коментарі, історію змін та інтерфейс. Повний програмний код, SQL-скрипти створення таблиць бази даних і деталізована структура файлів проєкту доцільно подаються в додатках, а в основному тексті розкрито логіку їх функціонування та взаємодії.

Окрему увагу приділено користувацькому інтерфейсу та сценаріям взаємодії з системою. Визначено основні сторінки інтерфейсу: сторінку

авторизації, головну сторінку, сторінку створення заявки, список заявок, сторінку детального перегляду, сторінку зміни статусу, адміністративну панель і сторінку керування користувачами. Для кожної ролі передбачено власний навігаційний сценарій, що забезпечує зручність роботи й доступ лише до необхідних функцій.

Проведене тестування підтвердило працездатність основних модулів системи. Було перевірено авторизацію користувачів, створення заявок, оброблення неповних даних, перегляд власних і призначених заявок, зміну статусів, призначення виконавців, додавання коментарів, перегляд історії змін, фільтрацію заявок і вихід із системи. Результати тестових сценаріїв показали, що система коректно виконує основні операції та відповідає сформованим вимогам.

Практичне значення розробленої системи полягає в тому, що вона може бути використана як прототип для організації обліку заявок у невеликій установі, навчальному підрозділі, сервісній службі або IT-відділі. Система забезпечує структуроване збереження звернень, контроль їх виконання, розмежування ролей, фіксацію історії змін і зручну взаємодію між користувачами, виконавцями та адміністратором.

Отже, мету бакалаврської кваліфікаційної роботи досягнуто. У процесі виконання роботи було досліджено предметну область, сформовано вимоги до системи, обґрунтовано вибір технологій, спроєктовано архітектуру, функціональні модулі, структуру бази даних та алгоритм оброблення заявок, описано реалізацію програмного продукту й проведено його тестування. Подальший розвиток системи може бути пов’язаний із додаванням автоматичних сповіщень, розширеної аналітики, прикріплення файлів до заявок, експорту звітів, адаптації для мобільних пристроїв та інтеграції з електронною поштою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Київ: ДП «УкрНДНЦ», 2016. 16 с. URL: https://nv-oneu.com.ua/downloads/dstu-8302-2015.pdf (дата звернення: 12.01.2026).

2. Стандарт вищої освіти України. Перший (бакалаврський) рівень, галузь знань 12 «Інформаційні технології», спеціальність 122 «Комп'ютерні науки» : наказ Міністерства освіти і науки України від 10.07.2019 № 962. Київ : Міністерство освіти і науки України, 2019. URL: https://mon.gov.ua/static-objects/mon/sites/1/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyut.nauk.bakalavr-1.pdf (дата звернення: 15.01.2026).

3. ISO/IEC/IEEE 12207:2017. Systems and software engineering. Software life cycle processes. Geneva : ISO, 2017. URL: https://www.iso.org/standard/63712.html (дата звернення: 20.01.2026).

4. ISO/IEC/IEEE 29148:2018. Systems and software engineering. Life cycle processes. Requirements engineering. Geneva : ISO, 2018. URL: https://www.iso.org/obp/ui/ (дата звернення: 23.01.2026).

5. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. Geneva : ISO, 2011. URL: https://www.iso.org/standard/35733.html (дата звернення: 27.01.2026).

6. ISO 9241-11:2018. Ergonomics of human-system interaction. Part 11: Usability: Definitions and concepts. Geneva : ISO, 2018. URL: https://www.iso.org/ics/13.180.html (дата звернення: 31.01.2026).

7. ISO 9241-210:2019. Ergonomics of human-system interaction. Part 210: Human-centred design for interactive systems. Geneva : ISO, 2019. URL: https://www.iso.org/standard/77520.html (дата звернення: 04.02.2026).

8. ISO/IEC/IEEE 29119-1:2022. Software and systems engineering. Software testing. Part 1: General concepts. Geneva : ISO, 2022. URL:

https://committee.iso.org/committee/45086.html?angularParam=p%2F0%2Fu%2F1%2Fw%2F0%2Fd%2F0&view=catalogue (дата звернення: 08.02.2026).

9. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection. Information security management systems. Requirements. Geneva : ISO, 2022. URL: https://www.iso.org/home.html (дата звернення: 11.02.2026).

10. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection. Information security controls. Geneva: ISO, 2022. URL: https://www.iso.org/standard/75652.html (дата звернення: 14.02.2026).

11. ISO/IEC 20000-1:2018. Information technology. Service management. Part 1: Service management system requirements. Geneva : ISO, 2018. URL: https://www.iso.org/publication/PUB200200.html (дата звернення: 17.02.2026).

12. Голуб Б. Л. Методичні вказівки до розробки бакалаврської кваліфікаційної роботи для студентів спеціальності 122 «Комп'ютерні науки» освітнього ступеня «Бакалавр». Київ : НУБіП України, 2023. 4 друк. арк. URL: https://nubip.edu.ua/naukova-diyalnist-4 (дата звернення: 21.02.2026).

13. Шебаніна О. В., Тищенко С. І., Полторак А. С., Пархоменко О. Ю., Кучмійова Т. С., Табацкова Г. В., Мальченко П. О., Жебко О. О. Методичні рекомендації для підготовки і публічного захисту кваліфікаційних робіт здобувачами першого (бакалаврського) рівня вищої освіти денної форми навчання. Галузь знань 12 «Інформаційні технології», спеціальність 122 «Комп'ютерні науки». Миколаїв : Миколаївський національний аграрний університет, 2023. 65 с. URL: https://dspace.mnau.edu.ua/jspui/bitstream/123456789/17460/3/kvalifikacijna_robota_122.pdf (дата звернення: 24.02.2026).

14. Бородкіна І. Л., Бородкін Г. О. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів: навч. посіб. Київ: Ліра-К, 2020. 212 с.

15. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення : посібник для студентів вищих навчальних закладів. Київ: Центр учбової літератури, 2020. 204 с.

16. Вишня Б. В., Гайдай І. І., Рогоза М. Є. Інформаційні системи і технології : підручник. Київ : Каравела, 2021. 472 с.

17. Лавріщева К. М. Програмна інженерія: підручник. Київ: Академперіодика, 2008. 319 с.

18. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ : Видавнича група BHV, 2006. 384 с.

19. Пасічник О. В., Пасічник В. В., Угрин Д. І. Веб-технології: підручник. Львів: Магнолія 2006, 2018. 336 с.

20. Трофименко О. Г., Логінова Н. І., Задерейко О. В., Плачінда Т. С. Веб-технології та веб-дизайн : навч. посіб. Одеса : Фенікс, 2019.

21. Anvik J., Hiew L., Murphy G. C. Who Should Fix This Bug? Proceedings of the 28th International Conference on Software Engineering (ICSE ’06). New York : ACM, 2006. P. 361–370. DOI: 10.1145/1134285.1134336.

22. Authentication Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 01.03.2026).

23. AXELOS. ITIL Foundation: ITIL 4 Edition. London: The Stationery Office, 2019. 212 p.

24. Bangor A., Kortum P. T., Miller J. T. An Empirical Evaluation of the System Usability Scale. International Journal of Human-Computer Interaction. 2008. Vol. 24, no. 6. P. 574–594. DOI: 10.1080/10447310802205776.

25. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley Professional, 2021. 464 p.

26. Beck K. et al. Manifesto for Agile Software Development. 2001. URL: https://agilemanifesto.org/ (дата звернення: 04.03.2026).

27. Bertram D., Voida A., Greenberg S., Walker R. Communication, Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams. Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW ’10). New York : ACM, 2010. P. 291–300. DOI: 10.1145/1718918.1718972.

28. Boehm B. W. A Spiral Model of Software Development and Enhancement. Computer. 1988. Vol. 21, no. 5. P. 61–72. DOI: 10.1109/2.59.

29. Brooke J. SUS: A ‘Quick and Dirty’ Usability Scale. Usability Evaluation in Industry / ed. by P. W. Jordan, B. Thomas, I. L. McClelland, B. Weerdmeester. London : Taylor & Francis, 1996. P. 189–194.

30. Brooks F. P. No Silver Bullet: Essence and Accidents of Software Engineering. Computer. 1987. Vol. 20, no. 4. P. 10–19. DOI: 10.1109/MC.1987.1663532.

31. Chacon S., Straub B. Pro Git. 2nd ed. New York : Apress, 2014. DOI: 10.1007/978-1-4842-0076-6.

32. Chen P. P.-S. The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems. 1976. Vol. 1, no. 1. P. 9–36. DOI: 10.1145/320434.320440.

33. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970. Vol. 13, no. 6. P. 377–387. DOI: 10.1145/362384.362685.

34. Composer Documentation. Composer, 2026. URL: https://getcomposer.org/doc/ (дата звернення: 07.03.2026).

35. Cooper A., Reimann R., Cronin D., Noessel C. About Face: The Essentials of Interaction Design. 4th ed. Indianapolis, IN : Wiley, 2014. 690 p.

36. CREATE TABLE Statement. MySQL 8.4 Reference Manual. Oracle. URL: https://dev.mysql.com/doc/en/create-table.html (дата звернення: 10.03.2026).

37. Crispin L., Gregory J. Agile Testing: A Practical Guide for Testers and Agile Teams. Boston : Pearson Education, 2008. 576 p.

38. CSS Snapshot 2024. W3C, 2025. URL: https://www.w3.org/TR/css-2024/ (дата звернення: 13.03.2026).

39. Davis F. D. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly. 1989. Vol. 13, no. 3. P. 319–340. DOI: 10.2307/249008.

40. DeLone W. H., McLean E. R. The DeLone and McLean Model of

Information Systems Success: A Ten-Year Update. Journal of Management Information Systems. 2003. Vol. 19, no. 4. P. 9–30. DOI: 10.1080/07421222.2003.11045748.

41. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Hoboken, NJ : Pearson, 2015.

42. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Irvine : University of California, Irvine, 2000. Dissertation. URL: https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf (дата звернення: 16.03.2026).

43. Flanagan D. JavaScript: The Definitive Guide. Master the World's Most-Used Programming Language. 7th ed. Sebastopol, CA : O'Reilly Media, 2020.

44. FOREIGN KEY Constraints. MySQL 8.4 Reference Manual. Oracle. URL: https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html (дата звернення: 19.03.2026).

45. Garrett J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond. 2nd ed. Berkeley, CA : Pearson Education, 2010. 192 p.

46. HTML Living Standard. WHATWG. 2026. URL: https://html.spec.whatwg.org/ (дата звернення: 22.03.2026).

47. JavaScript Guide. MDN Web Docs. Mozilla. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 25.03.2026).

48. Ko A. J., DeLine R., Venolia G. Information Needs in Collocated Software Development Teams. Proceedings of the 29th International Conference on Software Engineering (ICSE '07). Washington, DC : IEEE Computer Society, 2007. P. 344–353. DOI: 10.1109/ICSE.2007.45.

49. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley, CA : New Riders, 2013. 216 p.

50. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken, NJ : Wiley, 2011.

51. MySQL 8.4 Reference Manual. Oracle. URL:

https://docs.oracle.com/cd/E17952_01/mysql-8.4-en/ (дата звернення: 28.03.2026).

52. Nielsen J., Molich R. Heuristic Evaluation of User Interfaces. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Seattle, 1990. P. 249–256. DOI: 10.1145/97243.97281.

53. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites. 6th ed. Sebastopol, CA : O'Reilly Media, 2021.

54. Norman D. The Design of Everyday Things: Revised and Expanded Edition. New York : Basic Books, 2013. 368 p.

55. OWASP Application Security Verification Standard 4.0.3. OWASP, 2021. URL: https://raw.githubusercontent.com/OWASP/ASVS/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf (дата звернення: 01.04.2026).

56. OWASP Top 10:2021. OWASP, 2021. URL: https://owasp.org/Top10/2021/A00_2021_Introduction/ (дата звернення: 04.04.2026).

57. password_hash. Manual. PHP Manual. URL: https://www.php.net/manual/en/function.password-hash.php (дата звернення: 07.04.2026).

58. Patton J., Economy P. User Story Mapping: Discover the Whole Story, Build the Right Product. Sebastopol, CA : O'Reilly, 2014. 276 p.

59. PHP: PDO. Manual. PHP Manual. URL: https://www.php.net/manual/en/book.pdo.php (дата звернення: 10.04.2026).

60. PHPUnit Manual. Edition for PHPUnit 12.5. Sebastian Bergmann, 2026. URL: https://docs.phpunit.de/en/12.5/index.html (дата звернення: 13.04.2026).

61. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2020.

62. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7th ed. Newtown Square, PA : Project Management Institute, 2021.

63. RFC 9110. HTTP Semantics. IETF, 2022. URL: https://datatracker.ietf.org/doc/html/rfc9110 (дата звернення: 16.04.2026).

64. Saltzer J. H., Reed D. P., Clark D. D. End-to-End Arguments in System Design. ACM Transactions on Computer Systems. 1984. Vol. 2, no. 4. P. 277–288. DOI: 10.1145/357401.357402.

65. Schwaber K., Sutherland J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (дата звернення: 19.04.2026).

66. Session Management Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html (дата звернення: 22.04.2026).

67. Shneiderman B., Plaisant C., Cohen M. S., Jacobs S., Elmqvist N. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed. Boston : Pearson, 2016. 622 p.

68. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill, 2020.

69. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 796 p.

70. SQL Injection Prevention Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 25.04.2026).

71. Wiegers K. E., Beatty J. Software Requirements. 3rd ed. Redmond, WA : Microsoft Press, 2013.

72. Zimmermann T., Premraj R., Bettenburg N., Just S., Schröter A., Weiss C. What Makes a Good Bug Report? IEEE Transactions on Software Engineering. 2010. Vol. 36, no. 5. P. 618–643. DOI: 10.1109/TSE.2010.63.

Фрагменти програмного коду інформаційної системи обліку заявок користувачів

```
<?php
// config/db.php
$host = 'localhost';
$dbname = 'request_system';
$username = 'root';
$password = '';
try {
    $pdo = new PDO(
        "mysql:host=$host;dbname=$dbname;charset=utf8mb4",
        $username,
        $password
    );
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    die('Помилка підключення до бази даних: ' . $e->getMessage());
}
?>
```

А.1. Фрагмент підключення до бази даних

```
<?php
// public/login.php
session_start();
require_once '../config/db.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = trim($_POST['email'] ?? '');
    $password = $_POST['password'] ?? '';
    if ($email === '' || $password === '') {
        $error = 'Заповніть усі обов’язкові поля';
    } else {
        $stmt = $pdo->prepare("
            SELECT users.id, users.name, users.email, users.password_hash, roles.role_name
            FROM users
            INNER JOIN roles ON users.role_id = roles.id
            WHERE users.email = :email
            LIMIT 1
        ");
        $stmt->execute(['email' => $email]);
        $user = $stmt->fetch();
        if ($user && password_verify($password, $user['password_hash'])) {
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['user_name'] = $user['name'];
            $_SESSION['role'] = $user['role_name'];
            header('Location: dashboard.php');
            exit;
        } else {
            $error = 'Невірний логін або пароль';
        }
    }
}
?>
```

А.2. Фрагмент авторизації користувача

```php
<?php
// tickets/create_ticket.php
require_once '../config/db.php';
session_start();
if (!isset($_SESSION['user_id'])) {
    header('Location: ../public/login.php');
    exit;
}
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $title = trim($_POST['title'] ?? '');
    $description = trim($_POST['description'] ?? '');
    $categoryId = (int)($_POST['category_id'] ?? 0);
    $userId = (int)$_SESSION['user_id'];
    if ($title === '' || $description === '' || $categoryId === 0) {
        $error = 'Заповніть тему, опис і категорію заявки';
    } else {
        $stmt = $pdo->prepare("
            INSERT INTO tickets
                (title, description, user_id, category_id, status_id, created_at, updated_at)
            VALUES
                (:title, :description, :user_id, :category_id,
                (SELECT id FROM statuses WHERE status_name = 'нова' LIMIT 1),
                NOW(), NOW())
        ");
        $stmt->execute([
            'title' => $title,
            'description' => $description,
            'user_id' => $userId,
            'category_id' => $categoryId
        ]);
        $ticketId = (int)$pdo->lastInsertId();
        $historyStmt = $pdo->prepare("
            INSERT INTO ticket_history (ticket_id, user_id, action, created_at)
            VALUES (:ticket_id, :user_id, :action, NOW())
        ");
        $historyStmt->execute([
            'ticket_id' => $ticketId,
            'user_id' => $userId,
            'action' => 'Створено нову заявку'
        ]);
        header('Location: tickets_list.php');
        exit;
    }
}
?>
```

А.3. Фрагмент створення нової заявки

```php
<?php
// tickets/update_status.php
require_once '../config/db.php';
session_start();
if (!isset($_SESSION['user_id']) || !in_array($_SESSION['role'], ['executor', 'admin'])) {
    die('Недостатньо прав для виконання цієї дії');
}
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $ticketId = (int)($_POST['ticket_id'] ?? 0);
    $newStatusId = (int)($_POST['status_id'] ?? 0);
    $userId = (int)$_SESSION['user_id'];
    if ($ticketId === 0 || $newStatusId === 0) {
        die('Некоректні дані для зміни статусу');
    }
    $oldStmt = $pdo->prepare("
        SELECT status_id
        FROM tickets
        WHERE id = :ticket_id
    ");
    $oldStmt->execute(['ticket_id' => $ticketId]);
    $oldStatus = $oldStmt->fetchColumn();
    $updateStmt = $pdo->prepare("
        UPDATE tickets
        SET status_id = :status_id, updated_at = NOW()
        WHERE id = :ticket_id
    ");
    $updateStmt->execute([
        'status_id' => $newStatusId,
        'ticket_id' => $ticketId
    ]);
    $historyStmt = $pdo->prepare("
        INSERT INTO ticket_history
            (ticket_id, user_id, action, old_value, new_value, created_at)
        VALUES
            (:ticket_id, :user_id, :action, :old_value, :new_value, NOW())
    ");
    $historyStmt->execute([
        'ticket_id' => $ticketId,
        'user_id' => $userId,
        'action' => 'Змінено статус заявки',
        'old_value' => $oldStatus,
        'new_value' => $newStatusId
    ]);
    header('Location: ticket_view.php?id=' . $ticketId);
    exit;
}
?>
```

А.4. Фрагмент зміни статусу заявки

```php
<?php
// tickets/add_comment.php

require_once '../config/db.php';
session_start();

if (!isset($_SESSION['user_id'])) {
    header('Location: ../public/login.php');
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $ticketId = (int)($_POST['ticket_id'] ?? 0);
    $commentText = trim($_POST['comment_text'] ?? '');
    $userId = (int)$_SESSION['user_id'];

    if ($ticketId === 0 || $commentText === '') {
        die('Коментар не може бути порожнім');
    }

    $stmt = $pdo->prepare("
        INSERT INTO comments (ticket_id, user_id, comment_text, created_at)
        VALUES (:ticket_id, :user_id, :comment_text, NOW())
    ");

    $stmt->execute([
        'ticket_id' => $ticketId,
        'user_id' => $userId,
        'comment_text' => $commentText
    ]);

    $historyStmt = $pdo->prepare("
        INSERT INTO ticket_history (ticket_id, user_id, action, created_at)
        VALUES (:ticket_id, :user_id, :action, NOW())
    ");

    $historyStmt->execute([
        'ticket_id' => $ticketId,
        'user_id' => $userId,
        'action' => 'Додано коментар до заявки'
    ]);

    header('Location: ticket_view.php?id=' . $ticketId);
    exit;
}
?>
```

А.5. Фрагмент додавання коментаря до заявки

ДОДАТОК Б

SQL-структура таблиць бази даних інформаційної системи обліку заявок користувачів

```
CREATE DATABASE IF NOT EXISTS request_system
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
USE request_system;

CREATE TABLE roles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role_id INT NOT NULL,
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_users_roles
        FOREIGN KEY (role_id)
        REFERENCES roles(id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE categories (
    id INT AUTO_INCREMENT PRIMARY KEY,
    category_name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE statuses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    status_name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE tickets (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    user_id INT NOT NULL,
    executor_id INT NULL,
    category_id INT NOT NULL,
    status_id INT NOT NULL,
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME NULL DEFAULT NULL,
    CONSTRAINT fk_tickets_user
        FOREIGN KEY (user_id)
        REFERENCES users(id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
```

```
  CONSTRAINT fk_tickets_executor
    FOREIGN KEY (executor_id)
    REFERENCES users(id)
    ON UPDATE CASCADE
    ON DELETE SET NULL,
  CONSTRAINT fk_tickets_category
    FOREIGN KEY (category_id)
    REFERENCES categories(id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  CONSTRAINT fk_tickets_status
    FOREIGN KEY (status_id)
    REFERENCES statuses(id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);
CREATE TABLE comments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  ticket_id INT NOT NULL,
  user_id INT NOT NULL,
  comment_text TEXT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_comments_ticket
    FOREIGN KEY (ticket_id)
    REFERENCES tickets(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT fk_comments_user
    FOREIGN KEY (user_id)
    REFERENCES users(id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);
CREATE TABLE ticket_history (
  id INT AUTO_INCREMENT PRIMARY KEY,
  ticket_id INT NOT NULL,
  user_id INT NOT NULL,
  action VARCHAR(255) NOT NULL,
  old_value VARCHAR(255) NULL,
  new_value VARCHAR(255) NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_history_ticket
    FOREIGN KEY (ticket_id)
    REFERENCES tickets(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT fk_history_user
    FOREIGN KEY (user_id)
    REFERENCES users(id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);
```

ДОДАТОК В

Структура файлів програмного проєкту інформаційної системи обліку заявок користувачів

```
request-system/
│
├── config/
│   └── db.php
│
├── includes/
│   ├── header.php
│   └── footer.php
│
├── public/
│   ├── index.php
│   ├── login.php
│   ├── dashboard.php
│   └── logout.php
│
├── tickets/
│   ├── create_ticket.php
│   ├── tickets_list.php
│   ├── ticket_view.php
│   ├── update_status.php
│   └── add_comment.php
│
├── admin/
│   ├── users.php
│   ├── assign_executor.php
│   └── all_tickets.php
│
├── assets/
│   ├── css/
│   │   └── style.css
│   └── js/
│       └── main.js
│
└── database/
    └── request_system.sql
```

ДОДАТОК Г

Приклади інтерфейсних сторінок інформаційної системи обліку заявок користувачів

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Вхід до системи</title>
    <link rel="stylesheet" href="../assets/css/style.css">
</head>
<body>

<div class="login-container">
    <h1>Інформаційна система обліку заявок</h1>
    <h2>Авторизація користувача</h2>

    <form method="post" action="login.php" class="login-form">
        <label for="email">Електронна пошта</label>
        <input
            type="email"
            id="email"
            name="email"
            placeholder="Введіть електронну пошту"
            required
        >

        <label for="password">Пароль</label>
        <input
            type="password"
            id="password"
            name="password"
            placeholder="Введіть пароль"
            required
        >

        <button type="submit">Увійти</button>
    </form>
</div>

</body>
</html>
```

Г.1. Приклад сторінки авторизації користувача

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Створення заявки</title>
  <link rel="stylesheet" href="../assets/css/style.css">
</head>
<body>
<header class="page-header">
  <h1>Створення нової заявки</h1>
  <nav>
    <a href="../public/dashboard.php">Головна</a>
    <a href="tickets_list.php">Мої заявки</a>
    <a href="../public/logout.php">Вийти</a>
  </nav>
</header>
<main class="content">
  <form method="post" action="create_ticket.php" class="ticket-form">
    <label for="title">Тема заявки</label>
    <input
      type="text"
      id="title"
      name="title"
      placeholder="Коротко опишіть суть звернення"
      required
    >
    <label for="category_id">Категорія заявки</label>
    <select id="category_id" name="category_id" required>
      <option value="">Оберіть категорію</option>
      <option value="1">Технічна проблема</option>
      <option value="2">Консультація</option>
      <option value="3">Доступ до системи</option>
      <option value="4">Помилка в роботі сервісу</option>
      <option value="5">Інше</option>
    </select>
    <label for="description">Опис заявки</label>
    <textarea
      id="description"
      name="description"
      rows="6"
      placeholder="Детально опишіть проблему або запит"
      required
    ></textarea>
    <button type="submit">Зберегти заявку</button>
  </form>
</main>
</body>
</html>
```

Г.2. Приклад форми створення нової заявки

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Список заявок</title>
    <link rel="stylesheet" href="../assets/css/style.css">
</head>
<body>
<header class="page-header">
    <h1>Список заявок</h1>
    <nav>
        <a href="../public/dashboard.php">Головна</a>
        <a href="create_ticket.php">Створити заявку</a>
        <a href="../public/logout.php">Вийти</a>
    </nav>
</header>
<main class="content">
    <table class="tickets-table">
        <thead>
            <tr>
                <th>№</th>
                <th>Тема</th>
                <th>Категорія</th>
                <th>Статус</th>
                <th>Дата створення</th>
                <th>Дія</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>Проблема з доступом до особистого кабінету</td>
                <td>Доступ до системи</td>
                <td>нова</td>
                <td>12.05.2026</td>
                <td>
                    <a href="ticket_view.php?id=1">Переглянути</a>
                </td>
            </tr>
            <tr>
                <td>2</td>
                <td>Помилка під час відправлення форми</td>
                <td>Помилка в роботі сервісу</td>
                <td>у роботі</td>
                <td>13.05.2026</td>
                <td>
                    <a href="ticket_view.php?id=2">Переглянути</a>
                </td>
            </tr>
        </tbody>
    </table>
</main>
</body>
</html>
```

Г.3. Приклад сторінки списку заявок