ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА ІНФОРМАТИКИ І КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ УПРАВЛІННЯ ПРОЦЕСОМ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Виконав: здобувач освіти 4 курсу, групи 4ІСТ
першого (бакалаврського) рівня вищої освіти
спеціальності 126 «Інформаційні системи та технології»
ОПП «Інформаційні системи та технологій в економіці, менеджменті та управлінні проектами»
Дзядевич Д.Д.
Керівник: Григорова А.А.
Рецензент: Захарченко Р.М.

Хмельницький – 2026 року

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)

Факультет _____Інфомаційних технологій та дизайну_____
Кафедра _Інформаціні системи та технології_____
Рівень вищої освіти __перший (бакалаврський)_____
Спеціальність ____126 «Інформаційні системи та технології»_____
Освітньо-професійна програма «Інформаційні системи та технології в економіці, менеджменті та управлінні проектами»_____

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою С.В. Моісеєнко
_____

«____» _____2026 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

_____Дзядевич Даніїл Дмитрович_____
(прізвище, ім'я, по батькові)
1. Тема роботи _Інформаційні технології управління процесом розробки програмного забезпечення _____
Керівник роботи _Григорова Анжела Анатоліївна, к.т.н., доцент_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «13»_01__2026 року №_46-с
2. Строки подання студентом роботи 10.06.2026_____
3. Вихідні дані до роботи публікації, статистичні данні, документація
_____
_____
_____
_____
4. Зміст роботи (перелік питань, які потрібно розробити) Теоретичні основи процесу розробки програмного забезпечення, життєвий цикл ПЗ та сучасні методології розробки. Аналіз інформаційних технологій та інструментів управління процесом розробки, зокрема систем контролю версій і технологій CI/CD. Оцінка ефективності використання та рекомендації щодо вдосконалення процесу управління розробкою програмного забезпечення._____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
___презентація_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділи 1-3	Григорова А.А.	20.01.2026	06.2026

7. Дата видачі завдання 20.01.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1.	Огляд літературних джерел з теми	24.01-19.02.2026р.	Виконано
2.	Складання і затвердження плану роботи	24.01-28.01.2026р.	Виконано
3.	Написання 1 розділу	29.01-29.02.2026р.	Виконано
4.	Написання 2 розділу	10.02-28.03.2026р.	Виконано
5.	Написання 3 розділу	29.03-20.04.2026р.	Виконано
6.	Формулювання висновків за темою дослідження	20.04-01.05.2026р.	Виконано
7.	Оформлення роботи	20.04-01.05.2026р.	Виконано
8.	Надання роботи керівнику для перевірки та написання подання	Травень 2026р.	Виконано
9.	Подання роботи на рецензування	Травень 2026р.	
10.	Подання роботи для перевірки у КСПНП	Травень 2026р.	
11.	Захист роботи в ЕК	Травень 2026р.	

Здобувач освіти _____ ____ Дзядевич Д.Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ ___Григорова А.А._
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить 84 сторінки, 7 таблиць, 5 рисунків, список використаних джерел з 42 найменувань.

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ УПРАВЛІННЯ ПРОЦЕСОМ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У першому розділі досліджено теоретичні основи управління процесом розробки програмного забезпечення, розглянуто життєвий цикл програмного забезпечення та сучасні методології розробки, зокрема Agile, Scrum і DevOps.

У другому розділі представлено аналіз сучасних інформаційних технологій управління розробкою програмного забезпечення. Розглянуто системи Jira, Trello та GitHub. Проведено порівняльний аналіз сучасних систем управління проєктами.

Третій розділ присвячено практичній реалізації інформаційної системи управління процесом розробки програмного забезпечення. Розроблено прототип веб-орієнтованої системи DevFlow Manager із використанням HTML, CSS та JavaScript. Описано архітектуру системи, структуру бази даних, алгоритм роботи програмного забезпечення. Наведено результати тестування системи.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, УПРАВЛІННЯ ПРОЄКТАМИ, AGILE, SCRUM, DEVOPS, WEB-ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій програмне забезпечення відіграє ключову роль у функціонуванні різноманітних сфер діяльності людини [1–5]. Практично жодна галузь економіки, науки чи освіти не може ефективно функціонувати без використання сучасних програмних продуктів. У зв'язку з цим особливої актуальності набуває питання ефективного управління процесом розробки програмного забезпечення.

Процес створення програмного забезпечення є складним, багатокомпонентним і багатоетапним. Він включає наступні етапи: аналіз вимог, проєктування, реалізацію, тестування та супровід. Кожен із цих етапів потребує чіткої організації, координації та контролю, що неможливо забезпечити без застосування сучасних інформаційних технологій управління.

Актуальність теми кваліфікаційної роботи полягає в необхідності підвищення ефективності управління процесом розробки програмного забезпечення за рахунок впровадження сучасних інформаційних технологій. Умови сучасного ринку вимагають скорочення термінів розробки, підвищення якості продукту та забезпечення гнучкості у реагуванні на зміну вимог.

Сучасні інформаційні технології управління дозволяють автоматизувати значну частину процесів, пов'язаних із плануванням, організацією та контролем виконання завдань [4–6]. Вони забезпечують прозорість роботи команди, покращують комунікацію між її учасниками та сприяють підвищенню продуктивності праці.

Метою даної кваліфікаційної роботи бакалавра є дослідження та розробка інформаційної технології управління процесом розробки програмного забезпечення.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

проаналізувати теоретичні основи управління процесом розробки програмного забезпечення;

дослідити сучасні інформаційні системи управління проєктами;

визначити архітектуру системи;

розробити модель інформаційної системи;

описати алгоритм роботи програмного забезпечення;

розробити прототип системи;

оцінити ефективність запропонованого рішення.

Об'єктом дослідження є процес управління розробкою програмного забезпечення.

Предметом дослідження є інформаційні технології, що використовуються для управління цим процесом.

Для вирішення поставлених завдань у дослідженні були використані наступні *методи*: порівняльний аналіз, системний аналіз, моделювання.

Практичне значення отриманих результатів полягає у можливості використання запропонованої системи для підвищення ефективності роботи IT-команд.

Інформаційну базу дослідження склали теоретична та практична інформація, статистичні дані, опубліковані у періодичних виданнях, мережі Інтернет.

За результатами дослідження опубліковано тези у матеріалах VIII Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні».

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПРОЦЕСОМ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Теоретичні основи та функції інформаційних технологій управління

У сучасному світі інформаційні технології є невід'ємною складовою діяльності будь-якого підприємства чи організації. Особливо важливу роль вони відіграють у сфері розробки програмного забезпечення, де ефективність управління процесами безпосередньо впливає на якість програмного продукту, строки його реалізації та продуктивність роботи команди. Зростання складності сучасних інформаційних систем потребує використання нових підходів до організації процесу розробки програмного забезпечення та впровадження сучасних інформаційних технологій управління [2, 4, 5, 10].

Інформаційні технології управління являють собою комплекс програмних, технічних та організаційних засобів, які забезпечують автоматизацію процесів планування, контролю, аналізу та координації роботи. Їх використання дозволяє значно підвищити ефективність діяльності IT-компаній та забезпечити стабільну реалізацію програмних проєктів [4, 5, 8].

Сучасний процес розробки програмного забезпечення включає велику кількість взаємопов'язаних етапів та потребує участі багатьох спеціалістів. У створенні програмного продукту беруть участь аналітики, програмісти, тестувальники, дизайнери, DevOps-інженери, системні адміністратори та менеджери проєкту. Координація роботи такої кількості працівників потребує використання спеціалізованих систем управління.

Основною метою інформаційних технологій управління є забезпечення ефективної організації процесу розробки програмного забезпечення. Використання сучасних цифрових платформ дозволяє оптимізувати роботу команди, забезпечити контроль виконання задач та підвищити якість програмного продукту[2, 4, 5, 10].

Однією з найважливіших функцій інформаційних технологій управління є планування. Планування передбачає визначення цілей проєкту, оцінювання ресурсів, формування структури задач та встановлення строків виконання робіт. Якісне планування дозволяє мінімізувати ризики реалізації проєкту та забезпечити ефективне використання ресурсів компанії [4, 5, 8].

У сучасних системах управління проєктами для планування використовуються календарі задач, діаграми Ганта та Kanban-дошки. Дані інструменти дозволяють візуалізувати процес виконання робіт та забезпечують зручний контроль реалізації проєкту.

Важливою функцією інформаційних технологій управління є організація роботи команди. Організація передбачає розподіл ролей між учасниками проєкту, визначення відповідальності та формування каналів взаємодії між працівниками [2, 4, 5, 10].

У процесі розробки програмного забезпечення особливе значення має швидкий обмін інформацією між членами команди. Відсутність належної комунікації може призвести до виникнення помилок, дублювання задач та затримок виконання робіт.

Сучасні системи управління дозволяють централізовано зберігати інформацію про проєкт та забезпечують швидкий доступ до необхідних даних. Це дозволяє підвищити ефективність взаємодії між учасниками команди та спростити процес організації роботи [2].

Ще однією важливою функцією інформаційних технологій управління є контроль. Контроль забезпечує можливість відстеження стану виконання задач та своєчасного виявлення проблем [2, 4, 5, 10].

Завдяки використанню цифрових платформ керівник проєкту може отримувати актуальну інформацію про виконання робіт у режимі реального часу. Це дозволяє швидко реагувати на зміни та коригувати процес реалізації проєкту.

Сучасні інформаційні системи управління також забезпечують можливість формування аналітичної звітності. На основі отриманих даних керівники можуть

оцінювати продуктивність працівників, швидкість виконання задач та ефективність використання ресурсів.

Аналітична функція інформаційних технологій управління має важливе значення для прийняття управлінських рішень. Використання аналітики дозволяє виявляти проблемні ділянки процесу розробки програмного забезпечення та оптимізувати роботу команди [2, 4, 5, 10].

Особливого значення у сучасних умовах набуває автоматизація процесів управління. Автоматизація дозволяє скоротити кількість рутинних операцій та зменшити навантаження на працівників.

Наприклад, автоматичне формування звітності або створення резервних копій даних дозволяє підвищити ефективність роботи команди та скоротити витрати часу на виконання допоміжних задач.

Сучасні IT-компанії активно використовують спеціалізовані системи управління проєктами. Серед найбільш популярних платформ можна виділити Jira, Trello, ClickUp, Asana та Microsoft Project [1, 3, 26, 27].

Jira є однією з найпоширеніших систем управління IT-проєктами. Вона підтримує Agile-підходи та дозволяє використовувати Scrum і Kanban-дошки. Система забезпечує можливість створення задач, формування звітності та контролю виконання робіт.

Trello орієнтована на візуальне управління задачами. Основним елементом системи є Kanban-дошка, яка складається зі списків та карток. Завдяки простому інтерфейсу система активно використовується невеликими командами та стартапами.

ClickUp поєднує функції планування, аналітики та організації командної роботи. Система дозволяє контролювати строки виконання задач, автоматизувати частину управлінських процесів та формувати звіти про продуктивність працівників.

Microsoft Project використовується переважно у великих компаніях та корпоративних проєктах. Система забезпечує детальне планування ресурсів, контроль бюджету та формування складної аналітичної звітності.

Важливим напрямом розвитку сучасних інформаційних технологій управління є використання хмарних сервісів. Хмарні технології дозволяють забезпечити доступ до інформації незалежно від місця перебування користувача [2, 4, 5, 10].

Використання хмарних платформ дозволяє організувати віддалену роботу команди та забезпечити взаємодію між працівниками, які знаходяться у різних містах або країнах.

Ще одним важливим напрямом розвитку інформаційних технологій управління є інтеграція систем управління проєктами із сервісами контролю версій. Найбільш популярною системою контролю версій є Git [2, 4, 5, 10].

Git дозволяє відстежувати зміни у програмному коді та координувати роботу кількох розробників одночасно. Використання GitHub або GitLab дозволяє організувати централізоване зберігання програмного коду та автоматизувати процес тестування програмного забезпечення.

Сучасні системи управління також підтримують інтеграцію із сервісами автоматизації тестування та розгортання програмного забезпечення. Для цього використовуються Jenkins, GitHub Actions, Docker та інші інструменти DevOps.

Особливого значення у сучасних умовах набуває питання управління ризиками. У процесі реалізації ІТ-проєктів можуть виникати проблеми, пов’язані зі зміною вимог замовника, перевищенням бюджету, затримкою виконання задач або виникненням технічних помилок.

Сучасні інформаційні системи управління дозволяють виявляти потенційні ризики ще на ранніх етапах реалізації проєкту та формувати заходи щодо їх усунення.

У сучасних ІТ-компаніях важливу роль відіграє інформаційна безпека. Системи управління повинні забезпечувати захист даних, контроль доступу користувачів та резервне копіювання інформації.

Для забезпечення безпеки використовуються механізми авторизації, шифрування даних та багаторівнева система контролю доступу. Це дозволяє

захистити інформацію від несанкціонованого доступу та зменшити ризик втрати даних.

Використання інформаційних технологій управління дозволяє значно підвищити ефективність роботи IT-команд. Автоматизація процесів управління сприяє скороченню часу виконання задач, покращенню взаємодії між учасниками команди та підвищенню якості програмного забезпечення [2, 4, 5, 10].

У сучасних умовах цифрової трансформації підприємств інформаційні технології управління стають необхідною складовою діяльності будь-якої IT-компанії. Їх використання дозволяє забезпечити стабільність процесу розробки програмного забезпечення та підвищити конкурентоспроможність організації.

Таким чином, інформаційні технології управління є важливою складовою процесу розробки програмного забезпечення. Вони забезпечують автоматизацію управлінських процесів, покращують взаємодію між учасниками команди та дозволяють підвищити ефективність реалізації IT-проєктів.

Таблиця 1.1 – Основні функції інформаційних технологій управління

Функція	Характеристика	Результат використання
Планування	Визначення цілей, строків та ресурсів	Контроль реалізації проєкту
Організація	Розподіл ролей між учасниками команди	Координація роботи працівників
Контроль	Моніторинг виконання задач	Своєчасне виявлення проблем
Аналіз	Формування звітності та оцінювання ефективності	Прийняття управлінських рішень
Автоматизація	Скорочення рутинних операцій	Підвищення продуктивності праці
Управління ризиками	Виявлення потенційних проблем	Мінімізація ризиків проєкту

1.2. Життєвий цикл програмного забезпечення

Життєвий цикл програмного забезпечення є однією з основних складових сучасної програмної інженерії. Він визначає послідовність етапів, які проходить програмний продукт від моменту виникнення ідеї до завершення його використання. У сучасних умовах розвитку інформаційних технологій правильна організація життєвого циклу програмного забезпечення має важливе значення для забезпечення якості програмного продукту, ефективного використання ресурсів та успішної реалізації ІТ-проєктів [11, 12, 14].

Сучасні програмні системи характеризуються високою складністю, великою кількістю функціональних модулів та необхідністю постійного оновлення. Саме тому процес розробки програмного забезпечення потребує чіткої організації та використання сучасних інформаційних технологій управління. Ефективне управління життєвим циклом дозволяє мінімізувати ризики, забезпечити контроль виконання робіт та підвищити продуктивність команди розробників [4, 5, 8].

Життєвий цикл програмного забезпечення складається з кількох основних етапів. Кожен етап виконує окрему функцію та має важливе значення для створення якісного програмного продукту. До основних етапів життєвого циклу належать аналіз вимог, проєктування, реалізація, тестування, впровадження та супровід програмного забезпечення [11, 12, 14].

Першим етапом життєвого циклу є аналіз вимог. На цьому етапі здійснюється вивчення потреб замовника та майбутніх користувачів системи. Основною метою аналізу вимог є визначення функціональних можливостей програмного продукту та формування технічного завдання [12, 14].

Під час аналізу вимог аналітики збирають інформацію про майбутню систему, визначають основні бізнес-процеси та формують перелік функціональних і нефункціональних вимог. Функціональні вимоги описують можливості системи, а нефункціональні визначають вимоги до продуктивності, безпеки, швидкодії та надійності програмного забезпечення.

Аналіз вимог є одним із найважливіших етапів життєвого циклу, оскільки помилки, допущені на цьому етапі, можуть негативно вплинути на весь процес

розробки. Неправильно сформовані вимоги часто призводять до необхідності переробки окремих модулів системи, збільшення витрат та затримки виконання проєкту [12, 14].

Після завершення аналізу вимог формується технічне завдання. Даний документ містить опис структури системи, функціональних можливостей програмного забезпечення, вимог до інтерфейсу користувача та принципів роботи системи. Технічне завдання використовується як основний документ під час подальшої розробки програмного продукту [11, 12].

Наступним етапом життєвого циклу є проєктування системи. На цьому етапі формується архітектура програмного забезпечення, визначаються основні модулі системи та принципи їх взаємодії. Якісне проєктування дозволяє забезпечити масштабованість, стабільність та надійність програмного продукту [12, 14].

Проєктування включає створення структури бази даних, моделювання бізнес-процесів, формування алгоритмів роботи системи та розробку інтерфейсу користувача. Особливого значення набуває питання безпеки даних та захисту інформації.

У сучасних IT-проєктах під час проєктування широко використовуються UML-діаграми, які дозволяють візуалізувати структуру системи та взаємодію між її компонентами. Використання моделей дозволяє спростити процес розробки та підвищити зрозумілість архітектури програмного продукту [12, 14, 15].

Після завершення проєктування розпочинається етап реалізації програмного забезпечення. На цьому етапі програмісти виконують написання програмного коду відповідно до технічного завдання та архітектури системи.

Для створення сучасних програмних продуктів використовуються різні мови програмування та фреймворки. Вибір технологій залежить від складності проєкту, вимог замовника та особливостей майбутньої системи. Для веб-розробки часто використовуються JavaScript, React, Node.js та Python, а для створення серверних систем – Java, PHP або C# [9, 12, 14].

Сучасна розробка програмного забезпечення передбачає використання систем контролю версій. Найбільш популярною системою є Git, який дозволяє координувати роботу кількох розробників одночасно та відстежувати зміни у програмному коді [34, 35].

Git забезпечує можливість створення окремих гілок для реалізації нових функцій та виправлення помилок. Завдяки цьому декілька розробників можуть працювати над різними частинами системи незалежно один від одного [34, 35].

Одним із найважливіших етапів життєвого циклу є тестування програмного забезпечення. Основною метою тестування є виявлення помилок та перевірка відповідності системи встановленим вимогам [13, 15].

Функціональне тестування дозволяє перевірити правильність роботи окремих функцій системи. Під час тестування аналізується відповідність результатів роботи програмного забезпечення очікуваним значенням [13].

Інтеграційне тестування використовується для перевірки взаємодії між окремими модулями системи. Даний вид тестування дозволяє виявити проблеми, пов’язані з передачею даних між компонентами програмного забезпечення [13, 15].

Навантажувальне тестування застосовується для оцінювання стабільності роботи системи при великій кількості одночасних користувачів. Особливо важливим цей етап є для веб-сервісів та хмарних платформ [13, 15].

Автоматизоване тестування дозволяє скоротити час перевірки працездатності програмного забезпечення та забезпечити швидке виявлення помилок. Для автоматизації тестування використовуються спеціалізовані інструменти, серед яких Selenium, JUnit та Cypress [13, 39].

Після завершення тестування програмний продукт переходить на етап впровадження. На цьому етапі здійснюється встановлення системи у робочому середовищі та перевірка її працездатності в реальних умовах експлуатації [11, 14].

Впровадження програмного забезпечення також включає налаштування серверного середовища, перенесення даних та навчання користувачів. Для

ефективного використання системи необхідно забезпечити користувачів технічною документацією та інструкціями.

Важливим напрямом розвитку сучасних інформаційних технологій є використання DevOps. DevOps поєднує процеси розробки та експлуатації програмного забезпечення та забезпечує автоматизацію тестування, інтеграції та розгортання програмного продукту [22, 23, 24].

Використання DevOps дозволяє скоротити час доставки програмного забезпечення та забезпечити швидке реагування на помилки. Автоматизація процесів тестування та оновлення значно підвищує стабільність роботи системи [22, 23, 24].

Завершальним етапом життєвого циклу є супровід програмного забезпечення. Супровід передбачає оновлення системи, усунення помилок та адаптацію програмного продукту до нових вимог користувачів [11, 14].

У процесі експлуатації програмного забезпечення можуть виникати нові вимоги, пов’язані зі зміною бізнес-процесів або появою нових технологій. Саме тому підтримка та модернізація програмного продукту є необхідною складовою життєвого циклу.

Важливу роль у процесі супроводу відіграє документування. Документація містить опис структури системи, алгоритмів роботи та інструкції для користувачів. Наявність технічної документації дозволяє спростити процес підтримки та модернізації програмного забезпечення [11, 12].

Сучасні IT-компанії активно використовують хмарні технології для організації життєвого циклу програмного забезпечення. Хмарні сервіси дозволяють забезпечити швидкий доступ до даних, організувати віддалену роботу команди та спростити процес резервного копіювання інформації [5, 9].

Ще одним важливим напрямом розвитку сучасного програмного забезпечення є використання мікросервісної архітектури. Даний підхід передбачає поділ системи на окремі незалежні модулі, кожен із яких виконує власну функцію. Це дозволяє спростити модернізацію системи та забезпечити її масштабованість [9, 10].

Сучасні системи управління проєктами дозволяють автоматизувати процес контролю виконання задач та формування звітності. Такі платформи, як Jira, Trello та ClickUp, активно використовуються для організації роботи команди та контролю життєвого циклу програмного забезпечення [26, 27, 28, 31, 33].

Правильна організація життєвого циклу програмного забезпечення дозволяє підвищити ефективність реалізації IT-проєктів, покращити якість програмного продукту та мінімізувати ризики виникнення помилок [4, 8, 12].

Таким чином, життєвий цикл програмного забезпечення є складним багатокомпонентним процесом, який включає аналіз вимог, проєктування, реалізацію, тестування, впровадження та супровід системи. Використання сучасних інформаційних технологій управління дозволяє забезпечити ефективну організацію роботи команди, скоротити строки реалізації проєкту та підвищити якість програмного забезпечення [11, 12, 14].

Таблиця 1.2 – Основні етапи життєвого циклу програмного забезпечення

Етап життєвого циклу	Основний зміст етапу	Результат виконання
Аналіз вимог	Вивчення потреб користувачів та формування вимог до системи	Технічне завдання
Проєктування	Створення архітектури системи та структури бази даних	Модель системи
Реалізація	Написання програмного коду	Програмний продукт
Тестування	Перевірка працездатності системи та виявлення помилок	Виправлення недоліків
Впровадження	Встановлення та налаштування системи	Готова до використання система
Супровід	Оновлення та підтримка програмного забезпечення	Стабільна робота системи

1.3. Методології розробки програмного забезпечення, переваги та проблеми впровадження

У сучасних умовах стрімкого розвитку інформаційних технологій процес створення програмного забезпечення потребує використання ефективних методів організації роботи команди. Складність сучасних програмних систем, постійна зміна вимог користувачів та необхідність швидкого випуску програмного продукту змушують IT-компанії використовувати сучасні методології розробки програмного забезпечення. Методологія розробки визначає правила організації процесу створення програмного продукту, порядок виконання етапів робіт, способи взаємодії між учасниками команди та принципи контролю якості системи [14, 15, 16].

Вибір правильної методології безпосередньо впливає на успішність реалізації IT-проєкту. Використання сучасних підходів дозволяє скоротити строки розробки, підвищити ефективність командної роботи, покращити якість програмного забезпечення та забезпечити стабільність роботи системи. У сучасних IT-компаніях використовуються як класичні, так і гнучкі методології розробки програмного забезпечення [14, 16, 17].

Однією з найвідоміших класичних моделей є каскадна модель Waterfall. Дана методологія передбачає послідовне виконання етапів життєвого циклу програмного забезпечення. Кожен етап починається лише після завершення попереднього, що забезпечує чітку структуру процесу розробки [14, 15].

Каскадна модель включає етап аналізу вимог, проєктування, реалізації, тестування, впровадження та супроводу програмного забезпечення. Особливістю Waterfall є детальне документування усіх процесів розробки. Завдяки цьому керівник проєкту може контролювати виконання робіт та оцінювати поточний стан проєкту [14, 15].

Однією з основних переваг каскадної моделі є простота організації роботи. Чітка послідовність етапів дозволяє ефективно планувати строки реалізації проєкту та контролювати використання ресурсів. Waterfall часто використовується у великих державних системах, банківському програмному забезпеченні та корпоративних інформаційних системах [14, 15].

Ще однією перевагою Waterfall є можливість детального документування. Наявність повної документації спрощує процес підтримки та модернізації програмного забезпечення. Крім того, детальна документація дозволяє швидко вводити нових працівників у процес розробки [14].

Незважаючи на переваги, каскадна модель має і певні недоліки. Основною проблемою Waterfall є низька гнучкість процесу розробки. Якщо у процесі реалізації проєкту виникає необхідність змінити вимоги замовника, внесення таких змін може бути складним та затратним [15, 16].

Ще одним недоліком каскадної моделі є те, що замовник отримує готовий результат лише після завершення всіх етапів розробки. Це може призвести до ситуації, коли кінцевий програмний продукт не повністю відповідає очікуванням користувачів [15, 16].

У сучасних умовах більшої популярності набувають гнучкі методології розробки програмного забезпечення. Найбільш відомим гнучким підходом є Agile. Agile являє собою сукупність принципів організації процесу розробки, які орієнтовані на швидку адаптацію до змін та постійну взаємодію із замовником [16, 17, 18].

Основною особливістю Agile є поділ процесу розробки на короткі цикли – ітерації. Після завершення кожної ітерації команда отримує частково готовий функціональний результат. Це дозволяє швидко отримувати зворотний зв’язок від замовника та поступово вдосконалювати програмний продукт [16, 17].

Agile забезпечує високу гнучкість процесу розробки. Якщо вимоги замовника змінюються у процесі реалізації проєкту, команда може швидко адаптувати систему до нових умов. Саме тому Agile активно використовується у сучасних IT-компаніях [16, 17, 18].

Однією з переваг Agile є покращення взаємодії між учасниками команди. Розробники, тестувальники, аналітики та менеджери постійно обмінюються інформацією, що дозволяє швидше вирішувати проблеми та підвищувати ефективність роботи [17, 18].

Agile також сприяє скороченню ризиків реалізації проєкту. Оскільки програмний продукт створюється поступово, помилки можуть бути виявлені ще на ранніх етапах розробки. Це дозволяє уникнути значних витрат на виправлення критичних недоліків системи [16, 17].

Разом із перевагами Agile має певні недоліки. Для ефективного використання гнучких підходів необхідна постійна участь замовника у процесі розробки. Крім того, Agile потребує високого рівня самоорганізації команди [17, 18].

Однією з найпоширеніших реалізацій Agile є Scrum. Scrum базується на використанні коротких циклів розробки, які називаються спринтами. Тривалість одного спринту зазвичай становить від одного до чотирьох тижнів [19, 20].

На початку кожного спринту команда формує перелік задач, які необхідно виконати. Після завершення спринту відбувається демонстрація результатів роботи замовнику. Такий підхід дозволяє поступово вдосконалювати програмний продукт та швидко реагувати на зміни вимог [19, 20].

Важливу роль у Scrum відіграють щоденні зустрічі команди. Під час таких зустрічей учасники обговорюють виконані задачі, проблеми та плани на подальшу роботу. Це забезпечує прозорість процесу розробки та покращує взаємодію між працівниками [19, 20].

Scrum дозволяє значно підвищити швидкість реалізації проєктів. Завдяки коротким спринтам команда може регулярно оцінювати результати роботи та вчасно коригувати процес розробки [19, 20].

Однак впровадження Scrum потребує правильної організації роботи команди. Відсутність чіткої координації або недостатня дисципліна працівників можуть негативно вплинути на ефективність реалізації проєкту [19, 20].

Ще однією популярною методологією є Kanban. Основною особливістю Kanban є використання спеціальних дошок для візуалізації процесу виконання задач. Kanban-дошка дозволяє відображати поточний стан задач та контролювати навантаження команди [21].

Kanban-дошка зазвичай містить кілька стовпців, які відображають різні етапи виконання задач. Наприклад, задачі можуть знаходитися у категоріях «To Do», «In Progress» та «Done». Такий підхід дозволяє учасникам команди бачити поточний стан роботи та оцінювати прогрес реалізації проєкту [21].

Однією з переваг Kanban є простота впровадження. Для використання даної методології не потрібно повністю змінювати структуру роботи команди. Саме тому Kanban часто використовується у невеликих IT-компаніях та стартапах [21].

Kanban також дозволяє контролювати навантаження працівників. Якщо кількість задач у певному стовпці стає надто великою, команда може оперативно перерозподілити ресурси [21].

Разом із перевагами Kanban має певні недоліки. Основною проблемою є складність прогнозування строків виконання робіт. Крім того, відсутність чітко визначених спринтів може ускладнювати процес планування проєкту [21].

Сучасні IT-компанії також активно використовують підхід DevOps. DevOps поєднує процеси розробки та експлуатації програмного забезпечення. Основною метою DevOps є автоматизація тестування, інтеграції та розгортання програмного продукту [22, 23, 24].

DevOps дозволяє значно скоротити час доставки програмного забезпечення та забезпечити стабільність роботи системи. Автоматизація процесів тестування та оновлення дозволяє швидко виявляти помилки та усувати недоліки програмного продукту [22, 23, 24].

Однією з основних складових DevOps є CI/CD. Безперервна інтеграція забезпечує автоматичну перевірку програмного коду після внесення змін, а безперервна доставка дозволяє автоматично оновлювати систему [22, 23, 24, 39].

Для реалізації DevOps використовуються спеціалізовані інструменти, серед яких Jenkins, GitHub Actions, GitLab CI/CD та Docker. Використання таких платформ дозволяє автоматизувати процеси тестування та розгортання системи [36, 37, 38, 39].

Важливою перевагою DevOps є покращення взаємодії між розробниками та системними адміністраторами. Це дозволяє забезпечити стабільність роботи системи та прискорити процес оновлення програмного забезпечення [22, 23, 24].

Разом із перевагами впровадження DevOps потребує значних ресурсів та високого рівня підготовки працівників. Компанії повинні використовувати додаткові інструменти автоматизації та забезпечити навчання персоналу [22, 24].

У сучасних IT-проєктах широко використовуються спеціалізовані системи управління проєктами. Серед найбільш популярних платформ можна виділити Jira, Trello, ClickUp та Asana [26, 27, 28, 29].

Jira активно використовується для управління Agile-проєктами та підтримує Scrum і Kanban. Система дозволяє створювати задачі, формувати звіти, контролювати виконання робіт та аналізувати продуктивність працівників [31, 32].

Trello орієнтована на просте візуальне управління задачами. Основною перевагою Trello є зручний інтерфейс та швидкість налаштування системи [33].

ClickUp поєднує функції планування, аналітики та організації командної роботи. Система дозволяє автоматизувати частину управлінських процесів та контролювати строки виконання задач [28].

Asana використовується для координації роботи команди та управління великими IT-проєктами. Платформа забезпечує формування звітності та контроль виконання робіт у режимі реального часу [29].

Незважаючи на переваги сучасних методологій та систем управління, їх впровадження може супроводжуватись певними труднощами. Однією з основних проблем є необхідність навчання персоналу [17, 22, 29].

Перехід до нових підходів організації роботи потребує адаптації працівників до нових принципів управління. У деяких випадках працівники можуть негативно ставитись до змін, що ускладнює процес впровадження сучасних методологій [17, 18].

Ще однією проблемою є висока вартість впровадження сучасних систем управління проєктами. Компанії можуть витрачати значні кошти на придбання ліцензій, технічну підтримку та навчання персоналу [26, 29, 31].

Проблеми також можуть виникати під час інтеграції нових платформ із вже існуючими інформаційними системами підприємства. Для ефективного використання сучасних методологій необхідно забезпечити сумісність усіх компонентів інформаційної інфраструктури компанії [22, 24, 31].

Таблиця 1.3 – Порівняння сучасних методологій розробки програмного забезпечення

Методологія	Основні переваги	Основні недоліки	Особливості використання
Waterfall	Чітка структура, простота контролю	Низька гнучкість	Послідовне виконання етапів
Agile	Швидка адаптація до змін	Необхідність постійної участі замовника	Ітеративна розробка
Scrum	Висока прозорість процесу	Потребує дисципліни команди	Робота короткими спринтами
Kanban	Візуалізація задач	Складність прогнозування строків	Використання Kanban-дошок
DevOps	Автоматизація процесів	Високі вимоги до кваліфікації	CI/CD та автоматичне розгортання

Незважаючи на труднощі впровадження, сучасні методології розробки програмного забезпечення забезпечують значне підвищення ефективності роботи IT-команд. Вони дозволяють покращити організацію процесу розробки, скоротити строки реалізації проєктів та підвищити якість програмного продукту [16, 17, 22].

Отже, методології розробки програмного забезпечення є важливою складовою сучасної програмної інженерії. Використання Waterfall, Agile, Scrum, Kanban та DevOps дозволяє забезпечити ефективну організацію роботи команди, покращити контроль виконання задач та підвищити конкурентоспроможність IT-компаній [14, 16, 21, 22].

Висновки до розділу 1. У першому розділі кваліфікаційної роботи було досліджено теоретичні основи управління процесом розробки програмного забезпечення, розглянуто функції інформаційних технологій управління, етапи життєвого циклу програмного забезпечення та сучасні методології організації процесу розробки. Проведене дослідження показало, що ефективне управління IT-проєктами є необхідною умовою створення якісного програмного продукту та забезпечення стабільної роботи команди розробників.

У роботі було встановлено, що сучасні інформаційні технології управління дозволяють автоматизувати процеси планування, контролю та аналізу виконання задач, покращують взаємодію між учасниками команди та забезпечують централізоване зберігання інформації про проєкт. Досліджено основні функції систем управління, серед яких планування, організація роботи команди, контроль виконання задач, аналіз ефективності та управління ризиками.

Також у розділі було розглянуто основні етапи життєвого циклу програмного забезпечення, а саме аналіз вимог, проєктування, реалізацію, тестування, впровадження та супровід системи. Встановлено, що правильна організація життєвого циклу дозволяє підвищити якість програмного продукту, мінімізувати ризики реалізації проєкту та скоротити строки розробки програмного забезпечення.

Окрему увагу приділено сучасним методологіям розробки програмного забезпечення, серед яких Waterfall, Agile, Scrum, Kanban та DevOps. Проведений аналіз показав, що використання гнучких методологій дозволяє забезпечити швидку адаптацію до змін вимог замовника, покращити командну взаємодію та підвищити ефективність реалізації IT-проєктів.

Отже, результати дослідження підтверджують важливість використання сучасних інформаційних технологій управління та сучасних методологій розробки програмного забезпечення для забезпечення ефективної організації процесу створення програмних продуктів та підвищення конкурентоспроможності ІТ-компаній.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ РОЗРОБКОЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальна характеристика та класифікація сучасних систем управління проєктами

У сучасних умовах розвитку інформаційних технологій ефективне управління проєктами є важливою складовою діяльності ІТ-компаній. Процес розробки програмного забезпечення включає велику кількість взаємопов’язаних задач, що потребують координації роботи команди, контролю виконання робіт та оперативного реагування на зміни вимог замовника. Саме тому сучасні організації активно використовують системи управління проєктами, які дозволяють автоматизувати процеси планування, організації та контролю виконання задач [26, 27, 29].

Системи управління проєктами являють собою програмні засоби, призначені для координації роботи команди, контролю строків виконання робіт, розподілу ресурсів та формування аналітичної звітності. Використання таких систем дозволяє підвищити ефективність реалізації ІТ-проєктів, скоротити витрати часу на виконання рутинних операцій та забезпечити прозорість процесу розробки програмного забезпечення [26, 27, 30].

Сучасні системи управління проєктами використовуються у різних сферах діяльності, однак найбільшого поширення вони набули саме у сфері інформаційних технологій. Це пов’язано з тим, що процес створення програмного забезпечення характеризується високою складністю, великою кількістю учасників та необхідністю постійного контролю виконання задач [8, 26, 27].

Однією з основних функцій систем управління проєктами є планування. Планування передбачає визначення цілей проєкту, формування структури задач, встановлення строків виконання робіт та оцінювання необхідних ресурсів. Завдяки використанню сучасних цифрових платформ керівники можуть

створювати детальні плани реалізації проєкту та контролювати їх виконання [4, 8, 26].

Важливою функцією систем управління є організація роботи команди. Сучасні платформи дозволяють розподіляти задачі між працівниками, визначати рівень доступу користувачів та координувати взаємодію між учасниками проєкту [26, 27].

Ще однією важливою функцією є контроль виконання задач. Системи управління проєктами дозволяють відстежувати поточний стан виконання робіт у режимі реального часу. Це забезпечує можливість своєчасного виявлення проблем та коригування процесу реалізації проєкту [26, 27, 29].

Сучасні системи управління також підтримують формування аналітичної звітності. Керівники можуть отримувати інформацію про продуктивність працівників, швидкість виконання задач та ефективність використання ресурсів. Це дозволяє приймати обґрунтовані управлінські рішення та оптимізувати процес розробки програмного забезпечення [27, 29, 30].

Системи управління проєктами можна класифікувати за різними ознаками. Однією з основних ознак класифікації є функціональні можливості системи [26, 29].

За функціональністю системи управління проєктами поділяються на базові, розширені та комплексні. Базові системи призначені для простого управління задачами та організації командної роботи. До таких систем належить Trello [33].

Trello є популярною платформою для візуального управління задачами. Основним елементом системи є Kanban-дошка, яка складається зі списків та карток. Користувачі можуть створювати задачі, призначати відповідальних осіб та контролювати виконання робіт [33].

Основною перевагою Trello є простота використання та швидкість налаштування. Система активно використовується невеликими командами та стартапами. Разом із тим функціональні можливості Trello є обмеженими у порівнянні з більш складними системами управління [33].

До розширених систем управління належать Jira, ClickUp та Asana. Дані платформи забезпечують не лише управління задачами, але й підтримують аналітику, автоматизацію процесів та інтеграцію з іншими сервісами.

Jira є однією з найпопулярніших систем управління IT-проєктами. Платформа підтримує Agile-підходи та дозволяє використовувати Scrum і Kanban-дошки. Jira забезпечує можливість створення задач, формування звітності та контролю виконання робіт [31, 32].

Важливою перевагою Jira є гнучкість налаштування. Компанії можуть адаптувати систему відповідно до власних потреб та використовувати різні сценарії організації роботи команди [31, 32].

Jira також підтримує інтеграцію із сервісами GitHub, GitLab та Jenkins. Це дозволяє автоматизувати процес тестування та розгортання програмного забезпечення [31, 36, 37, 38].

Разом із перевагами Jira має певні недоліки. Система характеризується складним інтерфейсом та потребує часу для навчання працівників. Крім того, для великих компаній використання Jira може супроводжуватись значними фінансовими витратами [31, 32].

ClickUp є сучасною платформою для комплексного управління проєктами. Система поєднує функції планування, організації задач, аналітики та автоматизації процесів управління [28].

Однією з переваг ClickUp є можливість налаштування інтерфейсу відповідно до потреб користувача. Система підтримує календарі задач, діаграми Ганта, Kanban-дошки та інші інструменти візуалізації процесу виконання робіт[28].

Asana використовується для координації командної роботи та управління великими проєктами. Система забезпечує формування звітності, контроль строків виконання задач та підтримує інтеграцію з великою кількістю сторонніх сервісів [29].

До комплексних систем управління проєктами належить Microsoft Project. Дана система використовується переважно у великих компаніях та корпоративних проектах [30].

Microsoft Project забезпечує детальне планування ресурсів, контроль бюджету та формування складної аналітичної звітності. Система дозволяє створювати великі багаторівневі проєкти та контролювати виконання робіт на всіх етапах реалізації [30].

Ще однією ознакою класифікації систем управління є спосіб розгортання програмного забезпечення. За даною ознакою системи поділяються на локальні, хмарні та гібридні.

Локальні системи встановлюються безпосередньо на серверах компанії. Основною перевагою локальних рішень є високий рівень контролю над даними та можливість налаштування системи відповідно до потреб організації [5, 9].

Разом із тим локальні системи потребують значних витрат на технічне обслуговування та адміністрування серверного обладнання [5, 9].

Хмарні системи управління працюють через мережу Інтернет та не потребують встановлення програмного забезпечення на локальних серверах компанії. Користувачі отримують доступ до системи через веб-браузер або мобільний додаток [5, 9].

Основною перевагою хмарних платформ є можливість організації віддаленої роботи команди. Крім того, хмарні системи дозволяють швидко масштабувати ресурси та зменшити витрати на технічне обслуговування [5, 9].

Гібридні системи поєднують можливості локальних та хмарних рішень. Частина даних зберігається на локальних серверах компанії, а частина – у хмарному середовищі [5, 9].

Системи управління проєктами також можна класифікувати за сферою використання. Існують універсальні системи, які можуть використовуватись у різних галузях діяльності, та спеціалізовані системи, орієнтовані на сферу інформаційних технологій.

Спеціалізовані IT-системи підтримують Agile, Scrum, DevOps та інтеграцію із сервісами контролю версій. Це дозволяє ефективно організувати процес розробки програмного забезпечення та автоматизувати частину процесів тестування і розгортання системи [22, 23, 24, 31].

Сучасні системи управління проєктами активно використовують можливості автоматизації. Автоматизація дозволяє скоротити кількість рутинних операцій та зменшити навантаження на працівників [22, 28, 29].

Наприклад, системи можуть автоматично формувати звітність, надсилати повідомлення про зміни статусу задач або створювати резервні копії даних [22, 29].

Важливим напрямом розвитку сучасних систем управління є інтеграція із сервісами DevOps. Це дозволяє автоматизувати процеси тестування, інтеграції та розгортання програмного забезпечення [22, 23, 24].

У сучасних умовах особливого значення набуває питання інформаційної безпеки. Системи управління проєктами повинні забезпечувати захист даних, контроль доступу користувачів та резервне копіювання інформації [7, 10, 42].

Для забезпечення безпеки використовуються механізми авторизації, багаторівнева система доступу та шифрування даних. Це дозволяє захистити інформацію від несанкціонованого доступу та мінімізувати ризики втрати даних [7, 10, 42].

Таким чином, сучасні системи управління проєктами є важливою складовою процесу розробки програмного забезпечення. Вони забезпечують автоматизацію процесів планування, контролю та організації командної роботи, дозволяють підвищити ефективність реалізації IT-проєктів та забезпечити стабільну роботу команди розробників [26, 27, 31].

Таблиця 2.1 – Класифікація сучасних систем управління проєктами

Критерій класифікації	Тип системи	Приклади
За функціональністю	Базові	Trello
	Розширені	Jira, ClickUp, Asana
	Комплексні	Microsoft Project
За способом розгортання	Локальні	Microsoft Project Server
	Хмарні	Trello, Jira Cloud
	Гібридні	Jira Data Center
За сферою використання	Універсальні	Asana, ClickUp
	Спеціалізовані IT-системи	Jira, GitLab

2.2 Аналіз сучасних систем управління проєктами та платформ розробки

У сучасних умовах розвитку інформаційних технологій ефективне управління процесом розробки програмного забезпечення є одним із ключових факторів успішної реалізації IT-проєктів. Складність сучасних програмних систем, велика кількість учасників команди та необхідність швидкого реагування на зміни вимог замовника потребують використання спеціалізованих систем управління проєктами та платформ розробки [26, 27, 29].

Сучасні системи управління проєктами забезпечують автоматизацію процесів планування, контролю виконання задач, організації командної роботи та формування аналітичної звітності. Використання таких платформ дозволяє підвищити ефективність реалізації проєктів, скоротити строки розробки програмного забезпечення та покращити взаємодію між учасниками команди [26, 27, 30].

Однією з найпоширеніших систем управління проєктами є Jira. Дана платформа була розроблена компанією Atlassian і спочатку використовувалась як система відстеження помилок. Згодом Jira перетворилась на повноцінну систему управління IT-проєктами та стала однією з найпопулярніших платформ у сфері розробки програмного забезпечення [31, 32].

Jira орієнтована на підтримку гнучких методологій розробки, зокрема Agile, Scrum та Kanban. Система дозволяє організовувати роботу команди, формувати структуру задач, контролювати виконання робіт та аналізувати результати реалізації проєкту [31, 32].

Основним елементом Jira є задача. Користувачі можуть створювати задачі, визначати рівень їх пріоритетності, призначати відповідальних осіб та встановлювати строки виконання. Для деталізації процесу розробки використовуються підзадачі, що дозволяє розподіляти великі задачі на окремі етапи виконання [31, 32].

Однією з головних переваг Jira є висока гнучкість системи. Компанії можуть налаштовувати робочі процеси відповідно до власних потреб, створювати індивідуальні сценарії виконання задач та використовувати різні методи організації командної роботи [31, 32].

Jira підтримує використання Scrum-дошок та Kanban-дошок. Scrum-дошки дозволяють організовувати роботу короткими спринтами та контролювати виконання задач у межах певного циклу розробки. Kanban-дошки використовуються для візуалізації процесу виконання задач та контролю навантаження команди [19, 20, 21, 31].

Важливою перевагою Jira є можливість формування аналітичної звітності. Система дозволяє створювати діаграми, графіки та звіти, що відображають продуктивність команди, швидкість виконання задач та ефективність роботи проєкту [31, 32].

Jira також підтримує інтеграцію з великою кількістю сторонніх сервісів. Система може взаємодіяти з GitHub, GitLab, Jenkins, Slack та іншими платформами. Це дозволяє автоматизувати процес тестування та розгортання програмного забезпечення [31, 34, 36, 37, 38].

Разом із перевагами Jira має і певні недоліки. Основною проблемою системи є складність інтерфейсу. Для ефективного використання Jira працівникам необхідно пройти навчання та адаптацію до функціональних можливостей платформи [31, 32].

Ще одним недоліком є висока вартість використання Jira для великих компаній. При збільшенні кількості користувачів витрати на ліцензії можуть суттєво зростати [31, 32].

Ще однією популярною системою управління проєктами є Trello. Дана платформа орієнтована на візуальне управління задачами та характеризується простотою використання [33].

Основу Trello складають дошки, списки та картки. Дошка використовується для представлення проєкту, списки відображають окремі етапи виконання задач, а картки містять інформацію про конкретні задачі [33].

Користувачі можуть переміщувати картки між списками залежно від стану виконання роботи [33].

Наприклад, задачі можуть знаходитись у категоріях «To Do», «In Progress» та «Done».

Однією з головних переваг Trello є простий та інтуїтивно зрозумілий інтерфейс. Система не потребує складного налаштування та може використовуватись навіть невеликими командами без спеціальної підготовки [33].

Trello активно використовується стартапами та невеликими IT-компаніями. Платформа дозволяє швидко організувати командну роботу та забезпечити контроль виконання задач [33].

Система підтримує можливість додавання коментарів, вкладень, чек-листів та міток до карток. Це дозволяє централізовано зберігати інформацію про задачі та покращує взаємодію між учасниками команди [33].

Разом із тим функціональні можливості Trello є обмеженими у порівнянні з Jira. Система менш придатна для реалізації великих корпоративних проєктів, оскільки не підтримує складні механізми аналітики та управління ресурсами [33].

Важливе місце у сучасній розробці програмного забезпечення займає GitHub. GitHub є однією з найпопулярніших платформ для зберігання та спільної

розробки програмного забезпечення, яка базується на системі контролю версій Git [34, 35].

Основною функцією GitHub є контроль версій програмного коду. Система дозволяє відстежувати зміни у коді та координувати роботу кількох розробників одночасно [34, 35].

GitHub підтримує створення репозиторіїв, у яких зберігається програмний код проєкту. Користувачі можуть створювати окремі гілки для реалізації нових функцій або виправлення помилок [34, 35].

Однією з важливих можливостей GitHub є використання pull request. Pull request дозволяє перевіряти зміни у програмному коді перед їх об'єднанням з основною гілкою проєкту. Це сприяє покращенню якості програмного забезпечення та зменшує ризик виникнення помилок [34, 35].

GitHub також підтримує систему Issues, яка використовується для створення задач, відстеження помилок та організації командної роботи. Завдяки цьому GitHub може використовуватись не лише як система контролю версій, але і як інструмент управління проектами [34, 35].

Важливою перевагою GitHub є інтеграція з інструментами CI/CD. Платформа підтримує GitHub Actions, що дозволяє автоматизувати процес тестування, збірки та розгортання програмного забезпечення [37, 38].

GitHub широко використовується як у великих IT-компаніях, так і серед індивідуальних розробників. Платформа забезпечує можливість спільної роботи над проєктами та сприяє розвитку відкритого програмного забезпечення [34, 35].

Сучасна розробка програмного забезпечення неможлива без використання систем CI/CD. Дані системи відіграють важливу роль у процесі автоматизації створення програмного продукту та забезпечують швидке оновлення програмного забезпечення [22, 23, 24, 39].

CI/CD є скороченням від Continuous Integration та Continuous Delivery. Безперервна інтеграція передбачає автоматичне тестування програмного коду після внесення змін, а безперервна доставка забезпечує автоматичне розгортання програмного продукту [22, 23, 24, 39].

Використання CI/CD дозволяє значно скоротити час виходу нових версій програмного забезпечення та забезпечити швидке виявлення помилок [22, 23, 24, 39].

Однією з найпоширеніших платформ для реалізації CI/CD є Jenkins. Jenkins являє собою систему автоматизації, яка дозволяє створювати сценарії тестування, збірки та розгортання програмного забезпечення [36].

Основною перевагою Jenkins є висока гнучкість та можливість інтеграції з великою кількістю інструментів розробки. Система підтримує роботу з GitHub, GitLab, Docker та іншими платформами [36].

Разом із тим Jenkins потребує складного налаштування та технічної підготовки працівників. Для ефективного використання системи необхідно мати досвід роботи з серверним адмініструванням [36].

GitLab CI/CD є інтегрованою системою автоматизації, яка входить до складу платформи GitLab. Система дозволяє автоматизувати тестування, збірку та розгортання програмного забезпечення без використання сторонніх сервісів [37].

GitLab CI/CD характеризується зручністю використання та високим рівнем інтеграції з репозиторіями програмного коду. Це дозволяє спростити процес автоматизації та підвищити ефективність роботи команди [37].

GitHub Actions є сучасною платформою автоматизації, яка інтегрована безпосередньо у GitHub. Система дозволяє створювати сценарії автоматичного тестування та розгортання програмного забезпечення [38].

Однією з переваг GitHub Actions є простота налаштування та можливість використання готових шаблонів автоматизації. Це дозволяє швидко впроваджувати механізми CI/CD навіть у невеликих проектах [38].

Використання сучасних платформ розробки та систем управління проєктами дозволяє значно підвищити ефективність реалізації IT-проєктів. Автоматизація процесів управління, тестування та розгортання програмного забезпечення сприяє скороченню строків розробки та підвищенню якості програмного продукту [22, 24, 31, 34].

Разом із перевагами впровадження сучасних платформ може супроводжуватись певними труднощами. До основних проблем належать складність налаштування систем, необхідність навчання персоналу та фінансові витрати на використання комерційних сервісів [17, 22, 31].

Незважаючи на це, сучасні системи управління проєктами та платформи розробки є невід'ємною складовою діяльності IT-компаній. Їх використання дозволяє забезпечити ефективну організацію роботи команди, покращити контроль виконання задач та підвищити стабільність процесу розробки програмного забезпечення [26, 27, 31, 34].

Таблиця 2.2 – Порівняння сучасних систем управління проєктами та платформ розробки

Система / платформа	Основне призначення	Основні переваги	Основні недоліки
Jira	Управління Agile-проєктами	Гнучкість, аналітика, Scrum/Kanban	Складний інтерфейс
Trello	Візуальне управління задачами	Простота використання	Обмежений функціонал
GitHub	Контроль версій та спільна розробка	Pull request, CI/CD, Issues	Обмежені можливості управління проєктами
Jenkins	Автоматизація CI/CD	Висока гнучкість	Складність налаштування
GitLab CI/CD	Автоматизація тестування та розгортання	Інтеграція з GitLab	Потребує ресурсів сервера
GitHub Actions	Автоматизація процесів розробки	Простота використання	Залежність від Git

2.3 Порівняльний аналіз, проблеми та перспективи розвитку інформаційних технологій управління

У сучасних умовах розвитку інформаційних технологій процес створення програмного забезпечення потребує використання ефективних інструментів

управління проєктами та організації командної роботи. Розробка сучасних програмних продуктів характеризується високою складністю, великою кількістю учасників команди та необхідністю швидкого реагування на зміни вимог замовника. Саме тому сучасні ІТ-компанії активно використовують спеціалізовані системи управління проєктами та платформи автоматизації процесів розробки програмного забезпечення [26, 27, 29].

Інформаційні технології управління дозволяють автоматизувати процеси планування, контролю виконання задач, формування аналітичної звітності та координації роботи команди. Використання сучасних цифрових платформ забезпечує підвищення продуктивності праці, скорочення строків реалізації проєктів та покращення якості програмного забезпечення [4, 8, 26].

Для більш детального аналізу сучасних інформаційних технологій управління доцільно провести їх порівняння за основними критеріями ефективності. До таких критеріїв належать функціональність, зручність використання, масштабованість, інтеграція з іншими системами, підтримка Agile-методологій, аналітичні можливості та рівень автоматизації процесів [26, 29, 30].

Однією з найпопулярніших систем управління проєктами є Jira. Дана платформа активно використовується у сфері розробки програмного забезпечення та підтримує Agile, Scrum і Kanban. Jira забезпечує можливість створення задач, формування структури проєкту, контролю строків виконання робіт та аналізу ефективності роботи команди [31, 32].

Однією з головних переваг Jira є її висока функціональність. Система підтримує створення складних сценаріїв роботи, автоматизацію управлінських процесів та інтеграцію з великою кількістю сторонніх сервісів [31, 32].

Jira дозволяє використовувати Scrum-дошки та Kanban-дошки для організації процесу розробки програмного забезпечення. Scrum-дошки забезпечують можливість роботи короткими спринтами, а Kanban-дошки дозволяють візуалізувати процес виконання задач [19, 20, 21, 31].

Система також підтримує формування аналітичної звітності. Керівники можуть отримувати інформацію про продуктивність команди, швидкість виконання задач та поточний стан реалізації проєкту [31, 32].

Разом із перевагами Jira має і певні недоліки. Основною проблемою є складність інтерфейсу та необхідність навчання працівників. Для ефективного використання системи потрібен певний рівень технічної підготовки [31, 32].

Ще одним недоліком Jira є висока вартість використання для великих компаній. При збільшенні кількості користувачів витрати на ліцензії можуть суттєво зростати [31, 32].

Trello є іншою популярною системою управління проєктами, яка відзначається простотою використання та інтуїтивно зрозумілим інтерфейсом. Основу системи складають Kanban-дошки, списки та картки.

Однією з головних переваг Trello є швидкість освоєння системи. Навіть користувачі без технічної підготовки можуть швидко організувати роботу команди та розпочати використання платформи [33].

Trello активно використовується невеликими командами та стартапами. Система дозволяє створювати задачі, змінювати їх статус, додавати коментарі та вкладення [33].

Разом із тим функціональні можливості Trello є обмеженими у порівнянні з Jira. Система менш придатна для реалізації великих корпоративних проєктів та не підтримує складні механізми аналітики [33].

GitHub Projects є платформою, яка орієнтована переважно на процес розробки програмного забезпечення. Система інтегрована з GitHub та базується на використанні системи контролю версій Git [34, 35].

Основною перевагою GitHub Projects є інтеграція з репозиторіями програмного коду. Користувачі можуть пов’язувати задачі з pull request та контролювати процес внесення змін у програмний код [34, 35].

GitHub Projects також підтримує базові можливості управління задачами та дозволяє використовувати Kanban-дошки для організації процесу розробки [34, 35].

Платформа активно використовується командами розробників програмного забезпечення, які вже працюють із GitHub. Це дозволяє поєднати процес управління задачами та процес розробки програмного продукту в одному середовищі.

Разом із перевагами GitHub Projects має певні обмеження. Система менш функціональна у сфері управління проєктами порівняно з Jira або ClickUp.

ClickUp є універсальною системою управління проєктами, яка поєднує функції планування, аналітики, автоматизації та організації командної роботи.

Однією з переваг ClickUp є висока гнучкість налаштування. Система підтримує використання різних режимів представлення задач, серед яких Kanban-дошки, календарі, списки та діаграми Ганта.

ClickUp також підтримує автоматизацію процесів управління. Наприклад, система може автоматично змінювати статус задач або формувати повідомлення для користувачів.

Важливою перевагою ClickUp є сучасний інтерфейс та можливість інтеграції з великою кількістю сторонніх сервісів. Це дозволяє використовувати систему у проєктах різного рівня складності.

Важливим критерієм оцінювання сучасних систем управління є масштабованість. Масштабованість визначає здатність системи ефективно працювати з великими проєктами та значною кількістю користувачів.

Jira характеризується високою масштабованістю та використовується великими IT-компаніями для реалізації складних корпоративних проєктів.

Trello менш придатна для масштабних проєктів, оскільки функціональність системи орієнтована переважно на невеликі команди.

ClickUp та GitHub Projects займають проміжне положення та можуть використовуватись як у невеликих командах, так і у середніх IT-компаніях.

Ще одним важливим критерієм є інтеграція з іншими системами. Сучасні IT-компанії використовують велику кількість платформ для тестування, автоматизації та контролю версій, тому інтеграція між сервісами має важливе значення.

Jira підтримує інтеграцію з GitHub, GitLab, Jenkins, Slack та іншими платформами. Це дозволяє автоматизувати частину процесів управління та забезпечити централізовану організацію роботи команди.

GitHub Projects має перевагу у вигляді інтеграції з GitHub Actions та репозиторіями програмного коду. Це дозволяє автоматизувати процес тестування та розгортання програмного забезпечення [17, 18, 22].

ClickUp також підтримує інтеграцію з великою кількістю сервісів, серед яких Google Drive, Slack, Zoom та GitHub.

У сучасних умовах особливого значення набуває автоматизація процесів управління. Автоматизація дозволяє скоротити кількість рутинних операцій та підвищити ефективність роботи команди.

Сучасні системи управління підтримують автоматичне формування звітності, надсилання повідомлень та зміну статусу задач. Це дозволяє зменшити навантаження на працівників та покращити контроль виконання робіт.

Незважаючи на значний розвиток інформаційних технологій управління, їх впровадження супроводжується певними проблемами [17, 18].

Однією з основних проблем є складність впровадження нових систем управління. Великі компанії мають складну організаційну структуру, тому інтеграція нових платформ може вимагати значних технічних та фінансових ресурсів.

Ще однією проблемою є необхідність навчання персоналу. Для ефективного використання сучасних платформ працівники повинні освоїти нові принципи організації роботи та функціональні можливості систем.

У деяких випадках працівники можуть негативно ставитись до впровадження нових технологій. Опір змінам є поширеною проблемою у великих компаніях та може ускладнювати процес цифрової трансформації.

Важливою проблемою є інформаційна безпека. Системи управління проєктами містять конфіденційну інформацію про проєкти, програмний код та внутрішні процеси компанії [7, 10, 42].

Для забезпечення безпеки використовуються механізми авторизації, багаторівнева система доступу та шифрування даних. Проте навіть сучасні системи можуть бути вразливими до кібератак або витоку інформації.

Ще однією проблемою є висока вартість використання комерційних платформ. Великі IT-компанії можуть витрачати значні кошти на ліцензії, технічну підтримку та адміністрування систем управління.

Незважаючи на існуючі проблеми, сучасні інформаційні технології управління продовжують активно розвиватися.

Одним із перспективних напрямів розвитку є використання штучного інтелекту. AI-технології дозволяють автоматично аналізувати продуктивність команди, прогнозувати ризики реалізації проєкту та оптимізувати процес розподілу задач.

Перспективним напрямом є також використання машинного навчання для аналізу ефективності роботи команди та автоматичного формування рекомендацій щодо оптимізації процесу розробки програмного забезпечення.

Ще одним важливим напрямом розвитку є подальша автоматизація процесів тестування та розгортання програмного забезпечення. Використання DevOps-підходів дозволяє значно скоротити строки реалізації IT-проєктів та підвищити стабільність роботи системи.

Сучасні IT-компанії також активно використовують хмарні технології. Хмарні платформи дозволяють організувати віддалену роботу команди та забезпечити швидкий доступ до інформації незалежно від місця перебування користувача.

У перспективі інформаційні технології управління будуть ще більше орієнтовані на автоматизацію процесів, використання штучного інтелекту та інтеграцію з іншими цифровими платформами.

Таким чином, сучасні системи управління проєктами та платформи розробки відіграють важливу роль у процесі створення програмного забезпечення. Їх використання дозволяє підвищити ефективність реалізації IT-

проєктів, покращити взаємодію між учасниками команди та забезпечити стабільність процесу розробки програмного забезпечення.

Таблиця 2.3 – Порівняння сучасних систем управління проєктами

Критерій	Jira	Trello	GitHub Projects	ClickUp
Тип системи	Професійна PM-система	Kanban-дошка	Система для розробки	Універсальна PM-система
Складність освоєння	Висока	Низька	Середня	Середня
Гнучкість налаштувань	Дуже висока	Низька	Середня	Висока
Підтримка Agile	Повна	Часткова	Часткова	Повна
Управління задачами	Дуже розвинене	Базове	Середнє	Розвинене
Аналітика	Потужна	Мінімальна	Середня	Потужна
CI/CD інтеграція	Так	Ні	Так	Частково
Вартість	Висока	Безкоштовна / дешева	Безкоштовна / платна	Середня
Призначення	Великі проєкти	Малі проєкти	Розробка ПЗ	Універсальні проєкти

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЦЕСОМ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування та архітектура інформаційної системи

У межах даної кваліфікаційної роботи розглядається концепція інформаційної системи управління процесом розробки програмного забезпечення, яка призначена для автоматизації ключових процесів планування, організації, контролю та аналізу роботи команди розробників. У сучасних умовах розвитку інформаційних технологій ефективне управління процесом створення програмного забезпечення є важливою умовою успішної реалізації ІТ-проєктів [4, 8].

Сучасні програмні продукти характеризуються високою складністю, великою кількістю функціональних модулів та необхідністю постійної взаємодії між учасниками команди. Саме тому виникає потреба у створенні інформаційної системи, яка дозволить централізувати процес управління проєктами, автоматизувати рутинні операції та забезпечити швидкий доступ до інформації [26, 29].

Основною ідеєю створення системи є забезпечення єдиного інформаційного середовища, в якому всі учасники проєкту можуть взаємодіяти, обмінюватися даними та відстежувати стан виконання завдань у режимі реального часу [26, 29].

Запропонована система орієнтована на використання у сфері розробки програмного забезпечення та може застосовуватись як невеликими командами розробників, так і середніми ІТ-компаніями [12, 14].

Система створюється з метою:

- підвищення ефективності управління ІТ-проєктами;
- зменшення часу на планування та виконання завдань;
- покращення комунікації між учасниками команди;
- централізації даних;
- підвищення прозорості процесу розробки програмного забезпечення;
- автоматизації контролю виконання задач;
- забезпечення швидкого доступу до інформації про проєкт.

У сучасних ІТ-компаніях процес управління проєктами часто супроводжується значною кількістю рутинних операцій, пов’язаних із

плануванням задач, контролем строків виконання робіт та координацією діяльності працівників. Використання спеціалізованої інформаційної системи дозволяє автоматизувати значну частину цих процесів та підвищити продуктивність роботи команди [12, 14].

Важливим етапом створення інформаційної системи є аналіз вимог. Аналіз вимог дозволяє визначити основні функціональні можливості системи та сформувати перелік задач, які повинна вирішувати система [12, 14].

Коректно сформовані вимоги дозволяють уникнути помилок на наступних етапах розробки та забезпечують відповідність системи очікуванням користувачів.

У процесі аналізу вимог було визначено, що система повинна підтримувати роботу з кількома типами користувачів. Основними ролями у системі є адміністратор, менеджер проєкту та розробник.

Адміністратор відповідає за налаштування системи, управління користувачами та контроль безпеки. Менеджер проєкту організовує процес виконання задач, створює проєкти та контролює роботу команди. Розробник виконує поставлені задачі та взаємодіє з іншими учасниками проєкту.

Однією з ключових функцій системи є управління проєктами різного рівня складності. Користувачі повинні мати можливість створювати нові проєкти, редагувати інформацію про них та контролювати процес виконання задач [9, 10].

Система повинна підтримувати створення, редагування та видалення задач. Завдання є основною одиницею роботи в межах проєкту та містить інформацію про опис задачі, строки виконання, пріоритет та відповідального користувача [9, 10].

Для покращення організації процесу розробки система повинна підтримувати зміну статусу задач. Наприклад, задачі можуть перебувати у статусах «Нова», «У процесі виконання», «Перевірка» та «Завершено».

Важливою функцією системи є підтримка коментарів та обміну повідомленнями між учасниками команди. Це дозволяє покращити взаємодію між працівниками та забезпечити швидкий обмін інформацією.

Система також повинна забезпечувати можливість формування аналітичної звітності. Менеджери проєктів повинні мати доступ до інформації про продуктивність працівників, швидкість виконання задач та поточний стан реалізації проєкту.

Під час проєктування інформаційної системи особливу увагу було приділено архітектурі програмного забезпечення. Архітектура системи визначає принципи взаємодії між окремими компонентами програмного продукту та забезпечує стабільність роботи системи.

Запропонована інформаційна система базується на клієнт-серверному підході. Клієнт-серверна архітектура є одним із найбільш поширених підходів до створення сучасних веб-додатків.

Основною перевагою клієнт-серверної архітектури є розподіл функцій між окремими компонентами системи. Це дозволяє підвищити продуктивність програмного забезпечення та забезпечити масштабованість системи.

Frontend відповідає за взаємодію з користувачем та реалізує інтерфейс веб-додатку. Основним завданням Frontend є відображення інформації, обробка дій користувача та забезпечення зручності використання системи [40, 41].

Для створення інтерфейсу користувача можуть використовуватись сучасні веб-технології, серед яких HTML, CSS та JavaScript. Для покращення структури та швидкості розробки можуть застосовуватись бібліотеки та фреймворки, наприклад React [40, 41].

Інтерфейс системи повинен бути інтуїтивно зрозумілим та забезпечувати швидкий доступ до основних функцій. Користувачі повинні мати можливість переглядати список проєктів, створювати задачі та контролювати процес виконання робіт.

Backend відповідає за реалізацію бізнес-логіки системи, обробку запитів, взаємодію з базою даних та забезпечення безпеки [9, 10].

Основним завданням серверної частини є обробка даних та забезпечення взаємодії між користувачем і базою даних. Backend також відповідає за перевірку прав доступу користувачів та захист інформації.

Для реалізації серверної частини можуть використовуватись Node.js, Express або інші серверні технології. Використання JavaScript як для Frontend, так і для Backend дозволяє спростити процес розробки та забезпечити єдину структуру програмного коду [9, 10].

Важливою складовою інформаційної системи є база даних. База даних забезпечує зберігання інформації про користувачів, проєкти, задачі та результати роботи системи [10, 12].

Проєктування бази даних виконується з урахуванням принципів нормалізації, що дозволяє уникнути дублювання даних та забезпечити їх цілісність.

Основними сутностями бази даних є користувачі, проєкти, задачі та коментарі. Кожна сутність містить набір атрибутів, які описують відповідний об’єкт системи.

Наприклад, таблиця користувачів містить інформацію про ім’я користувача, адресу електронної пошти, пароль та роль у системі.

Таблиця проєктів містить назву проєкту, опис, дату створення та інформацію про відповідального менеджера.

Таблиця задач містить опис задачі, статус виконання, пріоритет та інформацію про користувача, який відповідає за виконання роботи.

Важливу роль у системі відіграє механізм авторизації та аутентифікації користувачів. Для забезпечення безпеки доступу до системи використовуються механізми перевірки логіна та пароля.

Паролі користувачів повинні зберігатись у зашифрованому вигляді. Для цього можуть використовуватись сучасні алгоритми хешування [10, 12].

Система також повинна підтримувати розмежування прав доступу. Наприклад, адміністратор має доступ до всіх функцій системи, тоді як розробник може працювати лише з призначеними йому задачами.

Особливого значення набуває питання масштабованості системи. Архітектура програмного забезпечення повинна забезпечувати можливість

подальшого розширення функціональних можливостей системи без необхідності повної зміни структури програмного продукту [10, 12].

Сучасні інформаційні системи повинні підтримувати інтеграцію з іншими платформами. Наприклад, система може взаємодіяти з GitHub, GitLab або сервісами CI/CD для автоматизації процесу розробки програмного забезпечення.

Важливим етапом проєктування є створення структури взаємодії між компонентами системи. Користувач взаємодіє з Frontend, який надсилає запити до Backend. Серверна частина обробляє запити та взаємодіє з базою даних для отримання або збереження інформації [40, 41].

Такий підхід дозволяє забезпечити чітке розмежування функцій між компонентами системи та спрощує процес підтримки програмного забезпечення.

Для забезпечення стабільної роботи системи необхідно реалізувати механізми резервного копіювання даних та обробки помилок. Це дозволяє мінімізувати ризики втрати інформації та забезпечити безперервність роботи системи [40, 41].

Важливою перевагою запропонованої архітектури є можливість подальшої модернізації системи. У майбутньому система може бути доповнена мобільним додатком, механізмами штучного інтелекту або розширеними інструментами аналітики.

Таким чином, проєктування та архітектура інформаційної системи є важливими етапами створення програмного забезпечення. Використання клієнт-серверного підходу, сучасних веб-технологій та принципів нормалізації бази даних дозволяє забезпечити ефективну організацію роботи системи та створити стабільний програмний продукт.

Таблиця 3.1 – Основні компоненти інформаційної системи

Компонент системи	Основне призначення	Основні функції
Frontend	Взаємодія з користувачем	Відображення інформації, обробка дій користувача

Компонент системи	Основне призначення	Основні функції
Backend	Реалізація бізнес-логіки	Обробка запитів, робота з базою даних
База даних	Зберігання інформації	Збереження користувачів, задач та проєктів
Система авторизації	Контроль доступу	Перевірка логіна та пароля
Аналітичний модуль	Формування звітності	Аналіз продуктивності та стану проєкту

3.2 Реалізація логіки роботи та інтерфейсу системи

Логіка роботи інформаційної системи реалізується на основі CRUD-операцій, які є базовими принципами взаємодії з даними у більшості сучасних інформаційних систем.

Операція Create використовується для додавання нових записів у систему. Вона застосовується при створенні нового користувача, додаванні проєкту, створенні задач та написанні коментарів.

Операція Read відповідає за отримання інформації з бази даних та її відображення користувачу.

Операція Update дозволяє змінювати вже існуючу інформацію в системі. Користувач може редагувати опис задачі, змінювати статус виконання та оновлювати дані проєкту.

Операція Delete використовується для видалення інформації із системи. Для запобігання випадковій втраті даних можуть використовуватись механізми soft delete.

Для демонстрації роботи системи використовується сценарій створення задачі. Користувач входить у систему, обирає проєкт, створює задачу, заповнює форму та зберігає інформацію.

Інтерфейс користувача є важливою складовою системи, оскільки саме через нього здійснюється взаємодія між користувачем і програмним забезпеченням [7,10].

Під час проєктування інтерфейсу основна увага приділялась простоті використання, зрозумілості навігації та адаптивності.

- інтерфейс;
- головну панель;
- список проєктів;
- Kanban-дошку
- сторінку задач;
- аналітичну панель.

Kanban-дошка використовується для візуального управління задачами. Задачі розподіляються між колонками відповідно до етапів виконання. Аналітична панель призначена для відображення статистики та аналізу ефективності роботи команди.

Обґрунтування вибору мови програмування. Для реалізації прототипу інформаційної системи управління процесом розробки програмного забезпечення було обрано мову програмування JavaScript, а також технології HTML та CSS.

Вибір JavaScript обумовлений тим, що дана мова є однією з основних технологій веб-розробки та підтримується всіма сучасними браузерами без необхідності встановлення додаткового програмного забезпечення. Це дозволяє запускати розроблений прототип безпосередньо у браузері користувача.

JavaScript забезпечує реалізацію інтерактивної логіки системи, зокрема:

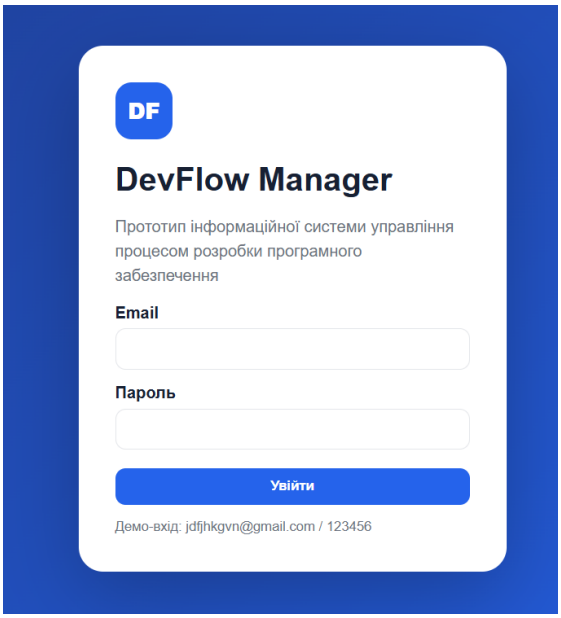

Рис. 3.1. Авторизація користувача

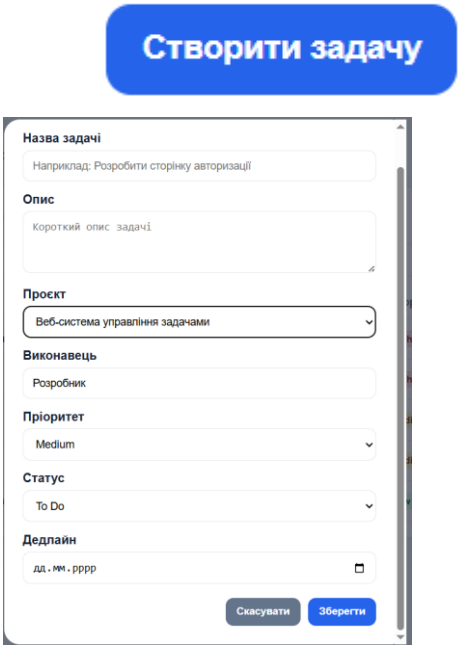

Рис. 3.2-створення нових задач

Назва	Проєкт	Виконавець	Пріоритет	Статус	Дедлайн	Дії
Розробити сторінку авторизації	Веб-система управління задачами	Розробник	High	To Do	2026-05-25	Статус Видалити
Створити Kanban-дошку	Веб-система управління задачами	Frontend Developer	High	In Progress	2026-05-28	Статус Видалити
Спроєктувати базу даних	Веб-система управління задачами	Backend Developer	Medium	In Progress	2026-05-30	Статус Видалити
Додати аналітичну панель	Модуль аналітики	Аналітик	Medium	Done	2026-06-01	Статус Видалити
Описати інструкцію користувача	UI/UX інтерфейс	Project Manager	Low	Testing	2026-06-03	Статус Видалити

Рис. 3.3. Зміна статусів задач

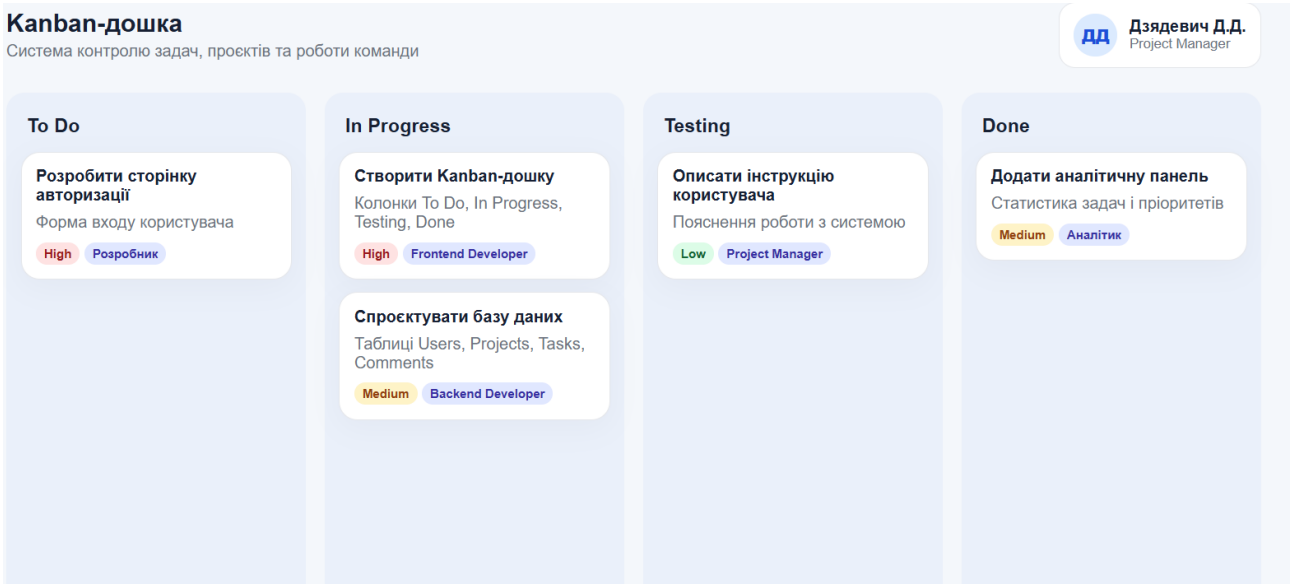

Рис. 3.4. Оновлення Kanban-дошки

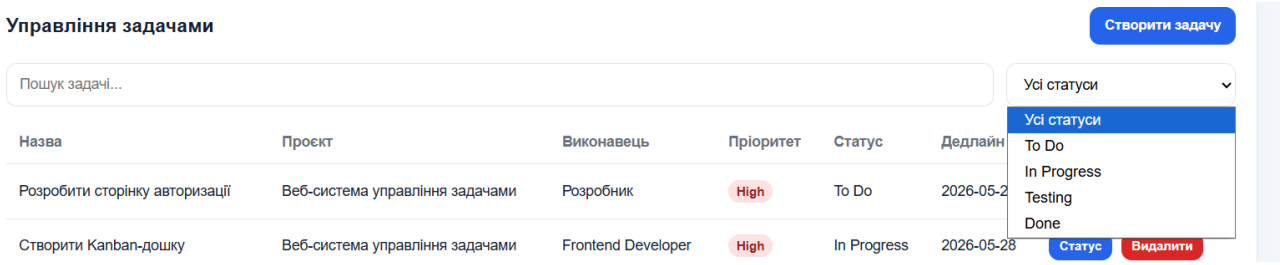

Рис. 3.5. Фільтрація та пошук задач

HTML було використано для побудови структури веб-сторінок, форм авторизації, таблиць, кнопок, блоків задач та елементів навігації. CSS застосовано для оформлення інтерфейсу, створення зручного дизайну, кольорової схеми, розташування елементів і адаптації зовнішнього вигляду системи.

Для забезпечення ефективної роботи інформаційної системи всі основні дії користувачів виконуються через централізовану систему обробки даних. Кожна операція проходить перевірку коректності введеної інформації та відповідності правам доступу користувача. Такий підхід дозволяє забезпечити цілісність даних та запобігти виникненню помилок під час роботи із системою.

Після успішної авторизації користувач отримує доступ до функціональних можливостей відповідно до своєї ролі. Адміністратор має повний доступ до всіх модулів системи та може керувати користувачами, проєктами й налаштуваннями.

Менеджер проєкту відповідає за створення проєктів, розподіл задач між виконавцями та контроль виконання робіт. Розробник працює із призначеними йому задачами та може змінювати їх статус залежно від етапу виконання.

Важливою складовою логіки роботи системи є механізм управління статусами задач. Кожна задача проходить декілька етапів життєвого циклу. Після створення задача отримує статус «Нова». Після початку роботи виконавця статус змінюється на «У процесі виконання». Після завершення програмної реалізації задача може бути переведена у статус «Перевірка», де виконується контроль результатів роботи. Якщо перевірка завершена успішно, задача отримує статус «Завершено».

Подібний підхід дозволяє керівнику проєкту контролювати поточний стан усіх робіт та швидко виявляти можливі затримки у виконанні завдань. Крім того, використання статусів спрощує процес формування аналітичної звітності та оцінювання продуктивності команди [34, 35, 36, 37, 38].

Окрему увагу приділено механізму пошуку та фільтрації задач. У процесі роботи над великими проєктами кількість задач може становити декілька сотень або навіть тисяч записів. Для спрощення навігації система підтримує пошук за назвою задачі, її описом, виконавцем або поточним статусом. Також передбачено фільтрацію за пріоритетом та строками виконання.

Використання механізму пошуку значно спрощує роботу користувачів та дозволяє швидко знаходити необхідну інформацію. Це особливо важливо для менеджерів проєктів, які постійно працюють із великою кількістю даних.

Однією з ключових функцій системи є автоматичне оновлення інтерфейсу після внесення змін до даних. Після створення або редагування задачі інформація миттєво відображається на Kanban-дошці без необхідності перезавантаження сторінки. Такий підхід забезпечує комфортну взаємодію користувача із системою та підвищує швидкість роботи.

Під час проєктування інтерфейсу особлива увага приділялася принципам User Experience (UX) та User Interface (UI). Основною метою було створення

простого та зрозумілого інтерфейсу, який не потребує тривалого навчання користувачів.

Головна сторінка системи містить навігаційне меню, список доступних проєктів та панель швидкого доступу до основних функцій. Така структура дозволяє користувачам швидко переходити між різними розділами системи та отримувати доступ до необхідної інформації.

Сторінка проєкту відображає детальну інформацію про вибраний проєкт. Користувач може переглядати список задач, контролювати строки виконання робіт та аналізувати поточний стан реалізації проєкту. Для покращення сприйняття інформації використовуються кольорові індикатори статусів задач.

Kanban-дошка є одним із ключових елементів інтерфейсу системи. Вона дозволяє візуально відображати процес виконання робіт та контролювати поточне навантаження команди. Кожна колонка дошки відповідає певному етапу життєвого циклу задачі.

Використання Kanban-дошки забезпечує високу наочність процесу розробки програмного забезпечення та дозволяє швидко оцінювати поточний стан проєкту. Користувач може змінювати статус задачі шляхом її переміщення між колонками, що значно спрощує процес управління роботою команди.

Аналітична панель системи призначена для відображення статистичної інформації. На панелі можуть відображатися кількість активних задач, кількість завершених завдань, продуктивність виконавців та інші показники ефективності роботи проєкту.

Використання аналітичних інструментів дозволяє менеджеру проєкту оперативно оцінювати поточний стан виконання робіт та приймати обґрунтовані управлінські рішення. На основі отриманих даних можна визначати проблемні ділянки проєкту та своєчасно вживати заходів щодо їх усунення.

Для підвищення надійності роботи системи використовується механізм локального збереження даних за допомогою технології localStorage. Це дозволяє зберігати інформацію про задачі та налаштування користувача навіть після закриття браузера. Використання localStorage значно спрощує реалізацію

прототипу та дозволяє демонструвати основні можливості системи без необхідності підключення серверної бази даних.

Таким чином, реалізована логіка роботи та структура інтерфейсу забезпечують зручну взаємодію користувача із системою, ефективне управління задачами та наочне відображення процесу розробки програмного забезпечення.

Отже, використання HTML, CSS та JavaScript є доцільним для створення прототипу веб-орієнтованої інформаційної системи, оскільки ці технології дозволяють швидко реалізувати інтерфейс користувача, логіку роботи системи та демонстрацію основних функціональних можливостей програмного продукту[34, 35, 36, 37, 38].

3.3 Безпека, тестування та оцінка ефективності системи

Безпека інформаційної системи є одним із найважливіших аспектів її розробки та подальшої експлуатації. У сучасних умовах розвитку інформаційних технологій інформація виступає одним із найцінніших ресурсів будь-якої організації. Саме тому питання захисту даних, забезпечення конфіденційності інформації та контролю доступу до ресурсів системи набувають особливого значення. Інформаційна система управління процесом розробки програмного забезпечення працює з даними користувачів, проєктів, задач та результатів діяльності команди, тому повинна забезпечувати високий рівень безпеки та надійності.

Основними принципами інформаційної безпеки є конфіденційність, цілісність та доступність даних. Конфіденційність забезпечує захист інформації від несанкціонованого доступу. Цілісність гарантує правильність та незмінність даних під час їх зберігання і передачі. Доступність забезпечує можливість використання інформації авторизованими користувачами у необхідний момент часу [34, 35, 36, 37, 38].

Для доступу до функціоналу системи користувач повинен пройти процедуру авторизації. Кожен користувач має власний обліковий запис із

унікальними даними для входу. Авторизація дозволяє ідентифікувати користувача та визначити його права доступу до окремих функцій системи [34, 35, 36, 37, 38].

Процедура входу до системи є одним із найважливіших механізмів захисту. Після введення логіна та пароля виконується перевірка облікових даних. Якщо введена інформація є коректною, користувач отримує доступ до відповідного функціоналу. У разі помилки система повідомляє про невдалу спробу входу та не надає доступ до ресурсів.

Для підвищення рівня безпеки паролі користувачів не зберігаються у відкритому вигляді. Перед записом до бази даних вони проходять процедуру хешування. Хешування являє собою процес перетворення пароля у спеціальний набір символів фіксованої довжини. Отримане значення використовується для перевірки автентичності користувача під час авторизації.

Використання хешування значно підвищує рівень захисту системи та зменшує ризик компрометації облікових записів. Навіть у випадку отримання доступу до бази даних сторонні особи не зможуть визначити справжні паролі користувачів [34, 35, 36, 37, 38].

Важливим елементом забезпечення безпеки є система розмежування прав доступу. У межах розробленої інформаційної системи передбачено декілька ролей користувачів, кожна з яких має власний перелік доступних функцій.

Адміністратор системи має повний доступ до всіх можливостей програмного продукту. Він може створювати нові облікові записи, редагувати інформацію про користувачів, змінювати налаштування системи та здійснювати контроль за її роботою.

Менеджер проєкту відповідає за створення проєктів, постановку задач, контроль виконання робіт та формування звітності. Розробник працює лише із задачами, які були призначені йому для виконання, та не має доступу до адміністративних функцій системи.

Реалізація рольової моделі дозволяє підвищити рівень безпеки та запобігти несанкціонованому доступу до конфіденційної інформації. Крім того, такий підхід забезпечує більш ефективну організацію роботи користувачів.

Особлива увага приділяється забезпеченню цілісності даних. Для цього система перевіряє правильність введеної інформації та контролює виконання операцій над записами бази даних. Перед збереженням інформації здійснюється перевірка коректності заповнення полів та відповідності введених даних встановленим вимогам.

Важливим компонентом захисту інформації є резервне копіювання даних. Регулярне створення резервних копій дозволяє уникнути втрати інформації у випадку технічних несправностей або помилок користувачів. Резервне копіювання є обов’язковим елементом сучасних інформаційних систем та використовується для забезпечення безперервності роботи програмного забезпечення.

Для контролю діяльності користувачів можуть використовуватись механізми журналювання подій. Система фіксує інформацію про вхід користувачів, створення нових задач, редагування даних та інші важливі операції. Це дозволяє аналізувати роботу системи та швидко виявляти можливі порушення безпеки [34, 35, 36, 37, 38].

Після завершення розробки інформаційної системи важливим етапом є її тестування. Тестування дозволяє перевірити правильність роботи програмного забезпечення та виявити помилки до моменту початку його використання кінцевими користувачами.

Одним із основних видів тестування є функціональне тестування. Його метою є перевірка відповідності реалізованого функціоналу початковим вимогам до системи.

Під час функціонального тестування було перевірено процес авторизації користувачів, створення проєктів, додавання задач, зміну статусів, роботу Kanban-дошки та формування аналітичної інформації.

Окрему увагу було приділено перевірці CRUD-операцій. Було протестовано створення нових записів, отримання інформації з бази даних, редагування існуючих записів та їх видалення. Результати тестування підтвердили коректну роботу всіх реалізованих функцій.

Інтеграційне тестування використовується для перевірки взаємодії між окремими компонентами інформаційної системи. У процесі тестування перевіряється правильність обміну інформацією між інтерфейсом користувача, серверною частиною та базою даних.

Під час інтеграційного тестування було встановлено, що всі компоненти системи працюють узгоджено та забезпечують коректне виконання поставлених задач. Помилок у процесі взаємодії між окремими модулями виявлено не було.

Системне тестування передбачає перевірку роботи програмного продукту в цілому як єдиної системи. На цьому етапі оцінюється стабільність роботи, коректність взаємодії між компонентами та відповідність функціоналу вимогам користувачів.

У процесі системного тестування було виконано перевірку всіх основних сценаріїв роботи користувачів. Було встановлено, що система успішно виконує поставлені функції та забезпечує стабільну роботу при стандартному навантаженні.

Ще одним важливим видом перевірки є навантажувальне тестування. Його метою є оцінка продуктивності системи при одночасній роботі великої кількості користувачів.

Навантажувальне тестування дозволяє визначити максимальну кількість операцій, які система може виконувати без втрати продуктивності. Отримані результати використовуються для оцінювання можливостей масштабування програмного продукту.

Під час тестування аналізувались швидкість завантаження сторінок, час обробки запитів та стабільність роботи системи при виконанні великої кількості операцій. Результати перевірки показали достатній рівень продуктивності для використання системи у невеликих та середніх командах розробників.

Для оцінювання ефективності розробленої системи було проведено її порівняння з популярними платформами Jira та Trello. Дані системи широко використовуються у сфері управління проєктами та вважаються одними з найбільш відомих інструментів організації командної роботи.

Jira характеризується широкими функціональними можливостями та підтримкою Agile-методологій. Система забезпечує детальне планування проєктів, управління задачами та формування розширеної аналітики. Разом із тим Jira має складний інтерфейс та потребує додаткового часу на навчання користувачів [34, 35, 36, 37, 38]. Trello орієнтована на простоту використання та базується на використанні Kanban-дошок. Система має інтуїтивно зрозумілий інтерфейс та дозволяє швидко організувати роботу команди. Проте її функціональні можливості є обмеженими порівняно з Jira.

Розроблена інформаційна система займає проміжне положення між Jira та Trello. Вона поєднує простоту використання з достатнім набором функціональних можливостей для ефективного управління процесом розробки програмного забезпечення.

Основною перевагою системи є спрощений інтерфейс користувача. Під час проєктування особлива увага приділялася зручності використання та швидкому доступу до основних функцій.

Ще однією перевагою є менша складність налаштування. Для початку роботи користувачам не потрібно виконувати складну конфігурацію системи або проходити тривале навчання.

Система адаптована для використання невеликими командами розробників. Це дозволяє ефективно організувати процес управління проєктами без необхідності використання складних корпоративних платформ.

Важливою перевагою є швидкий доступ до функціональних можливостей. Користувач може оперативно створювати задачі, переглядати інформацію про проєкти та контролювати процес виконання робіт.

Використання Kanban-дошки забезпечує наочне представлення процесу розробки та дозволяє швидко оцінювати поточний стан задач. Це покращує взаємодію між учасниками команди та підвищує прозорість роботи над проєктом.

Аналітичний модуль дозволяє формувати статистичну інформацію про кількість задач, продуктивність користувачів та стан реалізації проєкту. Отримані дані можуть використовуватись для прийняття управлінських рішень та оцінювання ефективності роботи команди [34, 35, 36, 37, 38].

Загалом результати оцінювання показали, що розроблена система дозволяє автоматизувати значну частину рутинних процесів, пов’язаних із плануванням, організацією та контролем виконання задач. Використання системи сприяє підвищенню продуктивності праці та покращенню координації роботи команди.

У майбутньому система може бути розширена шляхом інтеграції з сервісами GitHub та GitLab. Це дозволить автоматизувати процес відстеження змін програмного коду та забезпечити більш тісний зв’язок між процесом розробки та управління проєктами.

Перспективним напрямом розвитку є впровадження інструментів штучного інтелекту. Використання AI-аналітики дозволить автоматично аналізувати продуктивність команди, прогнозувати ризики реалізації проєктів та формувати рекомендації щодо оптимізації роботи.

Також перспективним напрямом розвитку є створення мобільного додатку. Мобільна версія системи дозволить користувачам працювати з проєктами незалежно від місця перебування та оперативно отримувати інформацію про зміни у процесі розробки.

Таким чином, проведене тестування підтвердило працездатність розробленої інформаційної системи та її відповідність поставленим вимогам. Реалізовані механізми безпеки забезпечують захист інформації, а результати оцінювання ефективності свідчать про доцільність використання системи для організації процесу розробки програмного забезпечення у невеликих та середніх командах.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досліджено сучасні інформаційні технології управління процесом розробки програмного забезпечення, а також розглянуто основні підходи до організації ефективної роботи IT-команд. Актуальність теми обумовлена постійним розвитком інформаційних технологій, зростанням складності програмних продуктів та необхідністю підвищення ефективності управління процесами розробки програмного забезпечення.

У першому розділі роботи було проаналізовано теоретичні основи управління процесом розробки програмного забезпечення. Розглянуто сутність інформаційних технологій управління, їх роль у сучасній IT-індустрії, а також основні функції інформаційних систем управління. Проведено аналіз життєвого циклу програмного забезпечення та охарактеризовано основні етапи розробки ПЗ — від аналізу вимог до супроводу готового продукту. Окрему увагу приділено сучасним методологіям розробки програмного забезпечення, зокрема Waterfall, Agile, Scrum та DevOps. Було визначено переваги використання інформаційних технологій управління, а також проаналізовано основні проблеми їх впровадження.

У другому розділі здійснено аналіз сучасних систем та інструментів управління процесом розробки програмного забезпечення. Було розглянуто функціональні можливості таких платформ, як Jira, Trello, GitHub та систем CI/CD. Проведено їх порівняльний аналіз за критеріями функціональності, зручності використання, масштабованості, інтеграції та вартості. У результаті дослідження встановлено, що сучасні інформаційні системи значно підвищують ефективність управління проєктами, покращують взаємодію між учасниками команди та сприяють автоматизації ключових процесів розробки програмного забезпечення. Також було визначено основні проблеми впровадження таких систем і перспективи їх подальшого розвитку, пов’язані з автоматизацією, використанням штучного інтелекту та розвитком DevOps-підходів.

У третьому розділі виконано проєктування та розробку інформаційної системи управління процесом розробки програмного забезпечення. Було сформовано загальну концепцію системи, визначено функціональні та нефункціональні вимоги, а також розроблено клієнт-серверну архітектуру програмного продукту. У роботі виконано проєктування бази даних, описано логіку взаємодії між основними компонентами системи та реалізовано CRUD-операції для роботи з даними. Також було розроблено інтерфейс користувача, орієнтований на простоту використання та швидкий доступ до функціоналу. Значну увагу приділено питанням безпеки системи, тестуванню, оцінці ефективності та перспективам подальшого розвитку програмного продукту. Проведене тестування підтвердило стабільність роботи системи та коректність реалізації основних функцій.

У результаті виконання кваліфікаційної роботи поставлена мета була досягнута. Було досліджено сучасні інформаційні технології управління процесом розробки програмного забезпечення та розроблено концепцію інформаційної системи, яка може бути використана для автоматизації управління IT-проєктами. Запропонована система дозволяє підвищити ефективність роботи команди, покращити контроль виконання завдань, оптимізувати комунікацію між учасниками проєкту та зменшити кількість організаційних помилок.

Практичне значення отриманих результатів полягає у можливості використання розробленої системи в невеликих і середніх IT-командах, навчальних проєктах та стартапах. Подальший розвиток системи може включати інтеграцію зі сторонніми сервісами, впровадження AI-аналітики, автоматичне формування звітів та підтримку мобільних платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дубина, Максим Вікторович, and Олена Михайлівна Козлянчеко. "Концептуальні аспекти дослідження сутності діджиталізації та її ролі в роз витку сучасного суспільства." *Проблеми і перспективи економіки та управління* 3 (2019) http://journals.stu.cn.ua/index.php/2411-5215/article/view/195621.

2. Григор'єва, Тетяна, Денис Розенвассер, and Юлія Дишкантюк. "Інформаційні технології в бізнесі." *Економіка та суспільство* 72 (2025) https://economyandsociety.in.ua/index.php/journal/article/download/5636/5576

3. Войтенко, А. Б., et al. "Інформаційні технології у комунікації та взаємодії влади, суспільства і бізнесу у публічному управлінні, місцевому самоврядуванні та розвитку підприємництва." (2021) https://www.researchgate.net/profile/Mariia-Plotnikova-2/publication/349651146_INFORMATION_TECHNOLOGIES_IN_COMMUNICATION_AND_INTERACTION_OF_GOVERNMENT_SOCIETY_AND_BUSINESS_IN_PUBLIC_ADMINISTRATION_LOCAL_SELF-GOVERNMENT_AND_ENTREPRENEURSHIP_DEVELOPMENT/links/61e87be29a753545e2e1067e/INFORMATION-TECHNOLOGIES-IN-COMMUNICATION-AND-INTERACTION-OF-GOVERNMENT-SOCIETY-AND-BUSINESS-IN-PUBLIC-ADMINISTRATION-LOCAL-SELF-GOVERNMENT-AND-ENTREPRENEURSHIP-DEVELOPMENT.pdf.

4. Ткач, Степан. "Інформаційні технології як інструмент управління бізнес-структурою." *Академічні візії* 36 (2024) https://www.academy-vision.org/index.php/av/article/view/1509.

5. Климчук, О. В. "Сучасні аспекти використання інформаційних систем і технологій в управлінні." *Бізнес, інновації, менеджмент: проблеми та перспективи* (2021) https://confmanagement-proc.kpi.ua/article/view/230931.

6. Ушакова, І. О., et al. "Інформаційні системи та технології на підприємстві: конспект лекцій." (2009) https://repository.hneu.edu.ua/handle/123456789/3112.

7. Лужний, Роман Ігорович. "Інформаційна система тестування програмного забезпечення." (2024) https://ena.lpnu.ua/items/77a2c704-895f-4a23-9d6c-0e81a92ad779.

8. Якимів, А. І. "Інформаційні системи управління проектами." *Наука й економіка* 3 (2013) http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Nie_2013_3_19.pdf.

9. Сікора, О. В., Т. Я. Вдовичин, and У. П. Когут. "Технології програмування інформаційних систем." *Таврійський науковий вісник. Серія: Технічні науки* 2 (2022) http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/xnvtn_2022_2_4.pdf.

10. Ситнік, Борис Тимофійович. "Основи інформаційних систем і технологій." (2019) http://files.znu.edu.ua/files/Bibliobooks/Inshi78/0058425.pdf.

11. Колдовской, Вячеслав Васильевич. *Управління інноваціями на етапах життєвого циклу програмного забезпечення*. Diss. Українська академія банківської справи, 2005 https://essuir.sumdu.edu.ua/items/201f09b0-9a73-4ad9-b32a-1f902d144429.

12. Карпенко, М. Ю., Н. О. Манакова, and І. О. Гавриленко. "Технології створення програмних продуктів та інформаційних систем: навчальний посібник." (2017) https://duikt.edu.ua/uploads/l_417_56149946.pdf.

13. Плотницький, Ярослав Валерійович. "Методи тестування програмних систем." (2015) https://eprints.zu.edu.ua/18023/.

14. Цибульник, Сергій Олексійович, and Катерина Сергіївна Барандич. "Технології розроблення програмного забезпечення. Частина 1. Життєвий цикл програмного забезпечення. Підручник." (2022) http://files.znu.edu.ua/files/Bibliobooks/Inshi72/0053481.pdf.

15. Костюк, Дар. "Порівняльний аналіз сучасних стратегій розробки вимог та контролю якості програмного забезпечення." (2023) https://ir.nmu.org.ua/bitstreams/65d491bf-079c-4c3a-a143-b4ecbfe1959d/download.

16. Лаба, Оксана Василівна, Любов Володимирівна Гринів, and Вікторія Станіславівна Артюшок. "Agile-методології у менеджменті проектів: виклики впровадження та шляхи їх подолання." (2024) http://repository.ukd.edu.ua/handle/123456789/1820.

17. Городова, К. І. "Впровадження Agile-підходів до управління HR-процесами в IT-компанії." (2024) https://er.ucu.edu.ua/items/49554db7-28fa-4833-862d-bddc305bc214.

18. Арсеніна, К. О. "Сучасні методи управління економічними та бізнес-рішеннями." (2025) https://essuir.sumdu.edu.ua/handle/123456789/100566.

19. Гулько, О. С., І. Л. Дибач, and А. О. Калиновський. "Порівняльна ефективність традиційного та Agile-менеджменту в розробці програмного забезпечення." (2025) https://repository.hneu.edu.ua/handle/123456789/37413.

20. Стельмах, Анастасії Андріївні. "Управління проєктом розробки сайту тревел-агенства “Daily Encourage” http://eprints.kname.edu.ua/72132/1/%D0%A3%D0%9F%28%D0%BA%D0%BD%29%202021-1_%D0%9F%D0%97_%D0%A1%D1%82%D0%B5%D0%BB%D1%8C%D0%BC%D0%B0%D1%85%20%D0%90%D0%90.pdf.

21. Граб, Дмитро Віталійович. *Розробка модуля інформаційної системи для управління IT-проєктами*. MS thesis. Тернопільський

національний технічний університет імені Івана Пулюя, 2024 https://elartu.tntu.edu.ua/handle/lib/47055.

22. Maslianko, Pavlo, and Ivan Savchuk. "DevOps–концепт і структурне представлення." *KPI Science News* 4 (2021) http://scinews.kpi.ua/article/download/261938/261876.

23. Олійник, Сергій Сергійович. "Автоматизація процесів розроблення та впровадження програмного забезпечення за допомогою DevOps." (2023) https://test.knutd.edu.ua/handle/123456789/27538.

24. Колиняк, Владислав Михайлович. *Аналіз та розробка ефективних методів DevOps для оптимізації процесу розробки програмного забезпечення*. BS thesis. Тернопільський національний технічний університет імені Івана Пулюя, 2024 https://elartu.tntu.edu.ua/handle/lib/45520.

25. Червяков, Костянтин Віталійович. "Моделі гнучкого управління IT-проєктами в умовах високої частоти релізів." *Управління розвитком складних систем* 64 (2025) http://mdcs.knuba.edu.ua/article/view/351649.

26. Чупріна, М. О., and І. А. Шеховцова. "Використання IT-іструментів для оптимізації управління бізнес-процесами підприємств України." *Економічний вісник Національного технічного університету України «Київський політехнічний інститут»* 13 (2016) http://ev.fmm.kpi.ua/article/view/80346.

27. Денисенко, Микола Павлович, and Ірина Василівна Колос. "Інформаційне забезпечення ефективного управління підприємством." (2006) https://dspace.nuft.edu.ua/items/ede9f9f9-6122-49a8-ade4-5fcad73a34ac.

28. Оксамитна, Любов, and Роман Пряха. "Особливості сучасних ERP-систем управління бізнес-процесами підприємства." *Управління розвитком складних систем* 51 (2022) http://mdcs.knuba.edu.ua/article/view/273713.

29. Шведа, Наталія Михайлівна, and Наталія Євгенівна Юрик. "Система управління проектами в Україні." *Збірник тез доповідей IV Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій "* 2 (2015) https://elartu.tntu.edu.ua/bitstream/123456789/11180/2/ConfATMT_2015v2_Shveda_N_M-Project_management_in_246-247.pdf.

30. Башинська, Ірина Олександрівна, and Аліна Валеріївна Хрістова. "Використання сучасних інформаційних технологій управління проектами." *Економічний журнал Одеського політехнічного університету* 1 (2017) http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/ejopu_2017_1_6.pdf.

31. П’ятничук, Ірина. "Інформаційні системи в управлінні проєктами: Онлайн-платформи і сервіси." *Економіка та суспільство* 42 (2022) https://economyandsociety.in.ua/index.php/journal/article/view/1618.

32. Крук, Роман, and Наталія Жуковська. "Застосунок-посередник для трансформації вимог до ПЗ Atlassian Jira п’ятикроковим семантичним аналізом генеративної великої мовної моделі." *Математичне та комп'ютерне моделювання. Серія: Технічні науки* (2025) http://mcm-tech.kpnu.edu.ua/article/view/332139.

33. Лимар, Вікторія Романівна. *Інформаційна панель згадок для відстеження та керування комунікаціями програмного середовища Jira*. Diss. Тернопіль, ЗУНУ, 2025 https://dspace.wunu.edu.ua/items/9f0a432c-1879-4750-8c25-56fb1d8773a1.

34. Жеплінський, Віталій Васильович. "Інформаційна система для організації IT проектів у вигляді веб-застосунку." (2023) https://ena.lpnu.ua/items/89674c9b-da2d-4ed2-aa55-1f1a900ff08c.

35. Онисько, Ераст, and Ігор Фармага. "Огляд та аналіз систем управління проєктами." (2024) https://ena.lpnu.ua/items/12100c50-430d-4407-9222-e9601129bad9.

36. Кордунов, Сергій Юрійович. "Автоматизація управління IT-проєктами за допомогою сучасних інструментів (Jira, Trello, Monday)." *Здобутки економіки: перспективи та інновації* 20 (2025) http://econp.com.ua/index.php/journal/article/view/549.

37. Коцовський, Владислав. "Аналіз вразливостей у Github Actions: дослідження безпеки ланцюгів постачання." (2025) https://dspace.uzhnu.edu.ua/bitstreams/7be28cc4-c4b1-4d30-b5e3-d30495b75dbe/download.

38. Семенько, Дмитрий Сергеевич. "Обзор исследований в области сетевого анализа сообществ на платформе GitHub." *Социология науки и технологий* 16.1 (2025) https://cyberleninka.ru/article/n/obzor-issledovaniy-v-oblasti-setevogo-analiza-soobschestv-na-platforme-github.

39. ЧАНКВЕТАДЗЕ, ДР, and ЛІ ФЕШАНИЧ. "МЕТОДОЛОГІЧНІ АСПЕКТИ ІНТЕГРАЦІЇ ФОРМАЛЬНОЇ ВЕРИФІКАЦІЇ В КОНВЕЄРИ CI/CD: АНАЛІЗ ВИКЛИКІВ І ПЕРСПЕКТИВ." *Вісник Херсонського національного технічного університету* 2.1 (92) (2025) https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/911.

40. Кравчук, Ольга. "Модель впровадження CI/CD для оптимізації управління IT-проєктами." (2023) https://elar.khmnu.edu.ua/items/00ef1f5a-c680-4daf-bb8b-32806fc43edf.

41. Васюта, Василь Васильович, and Кирило Олегович Харченко. "Аналіз систем безперервної інтеграції." (2021) http://reposit.nupp.edu.ua/handle/PoltNTU/10334.

42. Шинкарук, Олексій Олександрович. "Метод та інформаційна веб-система для ефективного розподілу аудиторій з урахуванням особливостей освітніх компонентів та параметрів доступності." (2025) https://elar.khmnu.edu.ua/items/59b68393-b2de-4904-9854-d7c802a40cb6

ДОДАТОК А

ТЕКСТ ПРОГРАМИ

Index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>DevFlow Manager — прототип системи управління розробкою ПЗ</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <div id="loginScreen" class="login-screen">
    <div class="login-card">
      <div class="logo">DF</div>
      <h1>DevFlow Manager</h1>
      <p>Прототип інформаційної системи управління процесом розробки програмного забезпечення</p>
      <label>Email</label>
      <input id="emailInput" type="email" value="" />
      <label>Пароль</label>
      <input id="passwordInput" type="password" value="" />
      <button onclick="login()">Увійти</button>
      <small>Демо-вхід: jdfjhkgvn@gmail.com / 123456</small>
    </div>
  </div>

  <div id="app" class="app hidden">
    <aside class="sidebar">
      <div class="brand">
        <div class="logo small">DF</div>
        <div>
          <strong>DevFlow</strong>
          <span>Project System</span>
        </div>
      </div>
      <nav>
        <button class="nav-btn active" onclick="showPage('dashboard', this)">Головна панель</button>
        <button class="nav-btn" onclick="showPage('projects', this)">Проєкти</button>
        <button class="nav-btn" onclick="showPage('kanban', this)">Kanban-дошка</button>
        <button class="nav-btn" onclick="showPage('tasks', this)">Задачі</button>
        <button class="nav-btn" onclick="showPage('analytics', this)">Аналітика</button>
        <button class="nav-btn" onclick="showPage('users', this)">Користувачі</button>
      </nav>
      <button class="logout" onclick="logout()">Вийти</button>
    </aside>

    <main class="content">
      <header class="topbar">
        <div>
```

```
            <h2 id="pageTitle">Головна панель</h2>
            <p>Система контролю задач, проєктів та роботи команди</p>
          </div>
          <div class="profile">
            <span class="avatar">ДД</span>
            <div>
              <strong>Дзядевич Д.Д.</strong>
              <small>Project Manager</small>
            </div>
          </div>
        </header>

        <section id="dashboard" class="page">
          <div class="stats-grid">
            <div class="stat-card"><span>Проєкти</span><strong id="projectCount">0</strong></div>
            <div class="stat-card"><span>Задачі</span><strong id="taskCount">0</strong></div>
            <div class="stat-card"><span>Виконано</span><strong id="doneCount">0</strong></div>
            <div class="stat-card"><span>У роботі</span><strong id="progressCount">0</strong></div>
          </div>

          <div class="panel">
            <div class="panel-head">
              <h3>Останні задачі</h3>
              <button onclick="showPageById('tasks')">Перейти до задач</button>
            </div>
            <div id="recentTasks" class="task-list"></div>
          </div>
        </section>

        <section id="projects" class="page hidden">
          <div class="panel">
            <div class="panel-head">
              <h3>Список проєктів</h3>
              <button onclick="addProject()">Додати проєкт</button>
            </div>
            <div id="projectList" class="cards-grid"></div>
          </div>
        </section>

        <section id="kanban" class="page hidden">
          <div class="kanban">
            <div class="column">
              <h3>To Do</h3>
              <div id="todoCol"></div>
            </div>
            <div class="column">
              <h3>In Progress</h3>
              <div id="progressCol"></div>
            </div>
```

```
    <div class="column">
      <h3>Testing</h3>
      <div id="testingCol"></div>
    </div>
    <div class="column">
      <h3>Done</h3>
      <div id="doneCol"></div>
    </div>
  </div>
</section>

<section id="tasks" class="page hidden">
  <div class="panel">
    <div class="panel-head">
      <h3>Управління задачами</h3>
      <button onclick="openTaskModal()">Створити задачу</button>
    </div>
    <div class="filters">
      <input id="searchInput" placeholder="Пошук задачі..." oninput="render()" />
      <select id="statusFilter" onchange="render()">
        <option value="">Усі статуси</option>
        <option value="To Do">To Do</option>
        <option value="In Progress">In Progress</option>
        <option value="Testing">Testing</option>
        <option value="Done">Done</option>
      </select>
    </div>
    <table>
      <thead>
        <tr>
          <th>Назва</th>
          <th>Проєкт</th>
          <th>Виконавець</th>
          <th>Пріоритет</th>
          <th>Статус</th>
          <th>Дедлайн</th>
          <th>Дії</th>
        </tr>
      </thead>
      <tbody id="taskTable"></tbody>
    </table>
  </div>
</section>

<section id="analytics" class="page hidden">
  <div class="analytics-grid">
    <div class="panel">
      <h3>Статуси задач</h3>
      <div id="statusBars" class="bars"></div>
    </div>
    <div class="panel">
      <h3>Пріоритети</h3>
      <div id="priorityBars" class="bars"></div>
```

```
          </div>
        </div>
      </section>

      <section id="users" class="page hidden">
        <div class="panel">
          <h3>Ролі користувачів</h3>
          <table>
            <thead>
              <tr><th>Користувач</th><th>Email</th><th>Роль</th><th>Права</th></tr>
            </thead>
            <tbody>
              <tr><td>Дзядевич
Д.Д.</td><td>jdfjhkgvn@gmail.com</td><td>Manager</td><td>Проєкти, задачі,
аналітика</td></tr>

<tr><td>Розробник</td><td>frghfhh@gmail.com</td><td>Developer</td><td>Задачі,
коментарі</td></tr>

<tr><td>Адміністратор</td><td>fawfaw3fa@gmail.com</td><td>Admin</td><td>Повний
доступ</td></tr>
            </tbody>
          </table>
        </div>
      </section>
    </main>
  </div>

  <div id="taskModal" class="modal hidden">
    <div class="modal-card">
      <h3>Створення задачі</h3>
      <label>Назва задачі</label>
      <input id="taskTitle" placeholder="Наприклад: Розробити сторінку авторизації" />
      <label>Опис</label>
      <textarea id="taskDescription" placeholder="Короткий опис задачі"></textarea>
      <label>Проєкт</label>
      <select id="taskProject"></select>
      <label>Виконавець</label>
      <input id="taskAssignee" value="Розробник" />
      <label>Пріоритет</label>
      <select id="taskPriority">
        <option>Low</option>
        <option selected>Medium</option>
        <option>High</option>
      </select>
      <label>Статус</label>
      <select id="taskStatus">
        <option>To Do</option>
        <option>In Progress</option>
        <option>Testing</option>
        <option>Done</option>
      </select>
      <label>Дедлайн</label>
```

```
        <input id="taskDeadline" type="date" />
        <div class="modal-actions">
          <button class="secondary" onclick="closeTaskModal()">Скасувати</button>
          <button onclick="saveTask()">Зберегти</button>
        </div>
      </div>
    </div>

    <script src="script.js"></script>
  </body>
  </html>

  Script.js
  const defaultData = {
    projects: [
      { id: 1, title: "Веб-система управління задачами", description: "Прототип системи для керування процесом розробки ПЗ" },
      { id: 2, title: "Модуль аналітики", description: "Формування статистики виконання задач" },
      { id: 3, title: "UI/UX інтерфейс", description: "Створення зручного інтерфейсу користувача" }
    ],
    tasks: [
      { id: 1, title: "Розробити сторінку авторизації", description: "Форма входу користувача", projectId: 1, assignee: "Розробник", priority: "High", status: "Done", deadline: "2026-05-25" },
      { id: 2, title: "Створити Kanban-дошку", description: "Колонки To Do, In Progress, Testing, Done", projectId: 1, assignee: "Frontend Developer", priority: "High", status: "In Progress", deadline: "2026-05-28" },
      { id: 3, title: "Спроєктувати базу даних", description: "Таблиці Users, Projects, Tasks, Comments", projectId: 1, assignee: "Backend Developer", priority: "Medium", status: "Testing", deadline: "2026-05-30" },
      { id: 4, title: "Додати аналітичну панель", description: "Статистика задач і пріоритетів", projectId: 2, assignee: "Аналітик", priority: "Medium", status: "To Do", deadline: "2026-06-01" },
      { id: 5, title: "Описати інструкцію користувача", description: "Пояснення роботи з системою", projectId: 3, assignee: "Project Manager", priority: "Low", status: "To Do", deadline: "2026-06-03" }
    ]
  };

  let data = JSON.parse(localStorage.getItem("devflowData")) || defaultData;

  function saveData() {
    localStorage.setItem("devflowData", JSON.stringify(data));
  }

  function login() {
    const email = document.getElementById("emailInput").value;
    const password = document.getElementById("passwordInput").value;

    const validEmail = "jdfjhkgvn@gmail.com";
    const validPassword = "123456";

    if (email === validEmail && password === validPassword) {
```

```
        document.getElementById("loginScreen").classList.add("hidden");
        document.getElementById("app").classList.remove("hidden");

        render();

      } else {

        alert("Невірний логін або пароль");

      }
    }

    function logout() {
      document.getElementById("loginScreen").classList.remove("hidden");
      document.getElementById("app").classList.add("hidden");
    }

    function showPage(pageId, btn) {
      document.querySelectorAll(".page").forEach(p => p.classList.add("hidden"));
      document.getElementById(pageId).classList.remove("hidden");
      document.querySelectorAll(".nav-btn").forEach(b => b.classList.remove("active"));
      if (btn) btn.classList.add("active");
      const titles = {
        dashboard: "Головна панель",
        projects: "Проєкти",
        kanban: "Kanban-дошка",
        tasks: "Задачі",
        analytics: "Аналітика",
        users: "Користувачі"
      };
      document.getElementById("pageTitle").innerText = titles[pageId];
      render();
    }

    function showPageById(pageId) {
      const btns = Array.from(document.querySelectorAll(".nav-btn"));
      const btn = btns.find(b => b.textContent.toLowerCase().includes(pageId === "tasks" ?
"задач" : pageId));
      showPage(pageId, btn);
    }

    function getProjectTitle(id) {
      const project = data.projects.find(p => p.id === Number(id));
      return project ? project.title : "Без проєкту";
    }

    function render() {
      renderStats();
      renderProjects();
      renderTasks();
      renderKanban();
      renderAnalytics();
```

```
  fillProjectSelect();
 }

 function renderStats() {
  document.getElementById("projectCount").innerText = data.projects.length;
  document.getElementById("taskCount").innerText = data.tasks.length;
  document.getElementById("doneCount").innerText = data.tasks.filter(t => t.status === "Done").length;
  document.getElementById("progressCount").innerText = data.tasks.filter(t => t.status === "In Progress").length;

  const recent = document.getElementById("recentTasks");
  recent.innerHTML = data.tasks.slice(-4).reverse().map(t => `
   <div class="task-card">
    <h4>${t.title}</h4>
    <p>${getProjectTitle(t.projectId)}</p>
    <span class="badge">${t.status}</span>
    <span class="badge priority-${t.priority}">${t.priority}</span>
   </div>
  `).join("");
 }

 function renderProjects() {
  const list = document.getElementById("projectList");
  list.innerHTML = data.projects.map(p => `
   <div class="project-card">
    <h4>${p.title}</h4>
    <p>${p.description}</p>
    <span class="badge">${data.tasks.filter(t => t.projectId === p.id).length} задач</span>
   </div>
  `).join("");
 }

 function renderTasks() {
  const search = document.getElementById("searchInput")?.value?.toLowerCase() || "";
  const status = document.getElementById("statusFilter")?.value || "";
  const tbody = document.getElementById("taskTable");
  let tasks = data.tasks.filter(t =>
   t.title.toLowerCase().includes(search) &&
   (!status || t.status === status)
  );

  tbody.innerHTML = tasks.map(t => `
   <tr>
    <td>${t.title}</td>
    <td>${getProjectTitle(t.projectId)}</td>
    <td>${t.assignee}</td>
    <td><span class="badge priority-${t.priority}">${t.priority}</span></td>
    <td>${t.status}</td>
    <td>${t.deadline}</td>
    <td>
     <button class="action-btn" onclick="nextStatus(${t.id})">Статус</button>
     <button class="action-btn delete" onclick="deleteTask(${t.id})">Видалити</button>
```

```
        </td>
      </tr>
    `).join("");
  }

  function renderKanban() {
    const map = {
      "To Do": "todoCol",
      "In Progress": "progressCol",
      "Testing": "testingCol",
      "Done": "doneCol"
    };

    Object.values(map).forEach(id => document.getElementById(id).innerHTML = "");

    data.tasks.forEach(t => {
      const col = document.getElementById(map[t.status]);
      if (!col) return;
      col.innerHTML += `
        <div class="task-card">
          <h4>${t.title}</h4>
          <p>${t.description}</p>
          <span class="badge priority-${t.priority}">${t.priority}</span>
          <span class="badge">${t.assignee}</span>
        </div>
      `;
    });
  }

  function renderAnalytics() {
    renderBars("statusBars", ["To Do", "In Progress", "Testing", "Done"], item => data.tasks.filter(t => t.status === item).length);
    renderBars("priorityBars", ["High", "Medium", "Low"], item => data.tasks.filter(t => t.priority === item).length);
  }

  function renderBars(elementId, labels, countFn) {
    const max = Math.max(...labels.map(countFn), 1);
    document.getElementById(elementId).innerHTML = labels.map(label => {
      const count = countFn(label);
      const width = (count / max) * 100;
      return `
        <div class="bar-row">
          <div class="bar-label"><span>${label}</span><strong>${count}</strong></div>
          <div class="bar-track"><div class="bar-fill" style="width:${width}%"></div></div>
        </div>
      `;
    }).join("");
  }

  function fillProjectSelect() {
    const select = document.getElementById("taskProject");
    if (!select) return;
```

```
    select.innerHTML = data.projects.map(p => `<option value="${p.id}">${p.title}</option>`).join("");
  }

  function addProject() {
    const title = prompt("Введіть назву проєкту:");
    if (!title) return;
    data.projects.push({
      id: Date.now(),
      title,
      description: "Новий проєкт, створений користувачем"
    });
    saveData();
    render();
  }

  function openTaskModal() {
    document.getElementById("taskModal").classList.remove("hidden");
    fillProjectSelect();
  }

  function closeTaskModal() {
    document.getElementById("taskModal").classList.add("hidden");
  }

  function saveTask() {
    const title = document.getElementById("taskTitle").value;
    if (!title.trim()) {
      alert("Введіть назву задачі");
      return;
    }
    data.tasks.push({
      id: Date.now(),
      title,
      description: document.getElementById("taskDescription").value,
      projectId: Number(document.getElementById("taskProject").value),
      assignee: document.getElementById("taskAssignee").value,
      priority: document.getElementById("taskPriority").value,
      status: document.getElementById("taskStatus").value,
      deadline: document.getElementById("taskDeadline").value || "2026-06-10"
    });
    saveData();
    closeTaskModal();
    render();
    document.getElementById("taskTitle").value = "";
    document.getElementById("taskDescription").value = "";
  }

  function deleteTask(id) {
    if (!confirm("Видалити задачу?")) return;
    data.tasks = data.tasks.filter(t => t.id !== id);
    saveData();
    render();
```

```
}

function nextStatus(id) {
  const order = ["To Do", "In Progress", "Testing", "Done"];
  const task = data.tasks.find(t => t.id === id);
  const current = order.indexOf(task.status);
  task.status = order[(current + 1) % order.length];
  saveData();
  render();
}

render();
```

style.css

```
:root {
  --bg: #f4f7fb;
  --panel: #ffffff;
  --primary: #2563eb;
  --primary-dark: #1d4ed8;
  --text: #172033;
  --muted: #6b7280;
  --border: #e5e7eb;
  --danger: #dc2626;
  --success: #16a34a;
  --warning: #d97706;
}

* { box-sizing: border-box; }

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: var(--bg);
  color: var(--text);
}

button {
  border: none;
  background: var(--primary);
  color: white;
  padding: 10px 14px;
  border-radius: 10px;
  cursor: pointer;
  font-weight: 600;
}

button:hover { background: var(--primary-dark); }

.hidden { display: none !important; }

.login-screen {
  min-height: 100vh;
  display: grid;
```

```
  place-items: center;
  background: linear-gradient(135deg, #1e3a8a, #2563eb);
}

.login-card {
  width: 420px;
  background: white;
  padding: 36px;
  border-radius: 24px;
  box-shadow: 0 24px 80px rgba(0,0,0,.25);
}

.login-card h1 { margin-bottom: 8px; }
.login-card p { color: var(--muted); line-height: 1.5; }

.logo {
  width: 56px;
  height: 56px;
  background: var(--primary);
  color: white;
  display: grid;
  place-items: center;
  border-radius: 16px;
  font-weight: 800;
  font-size: 22px;
}

.logo.small {
  width: 42px;
  height: 42px;
  border-radius: 12px;
  font-size: 16px;
}

label {
  display: block;
  margin-top: 14px;
  margin-bottom: 6px;
  font-weight: 600;
}

input, select, textarea {
  width: 100%;
  border: 1px solid var(--border);
  border-radius: 10px;
  padding: 11px 12px;
  font-size: 14px;
  background: white;
}

textarea { min-height: 80px; resize: vertical; }

.login-card button { width: 100%; margin-top: 18px; }
```

```
.login-card small { display: block; margin-top: 14px; color: var(--muted); }

.app {
  min-height: 100vh;
  display: flex;
}

.sidebar {
  width: 260px;
  background: #0f172a;
  color: white;
  padding: 24px;
  display: flex;
  flex-direction: column;
}

.brand {
  display: flex;
  gap: 12px;
  align-items: center;
  margin-bottom: 28px;
}

.brand span {
  display: block;
  color: #94a3b8;
  font-size: 12px;
}

nav {
  display: flex;
  flex-direction: column;
  gap: 8px;
}

.nav-btn {
  background: transparent;
  text-align: left;
  color: #cbd5e1;
}

.nav-btn:hover, .nav-btn.active {
  background: #1e293b;
  color: white;
}

.logout {
  margin-top: auto;
  background: #334155;
}

.content {
  flex: 1;
```

```
  padding: 28px;
  overflow: auto;
}

.topbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 24px;
}

.topbar h2 { margin: 0; }
.topbar p { margin: 4px 0 0; color: var(--muted); }

.profile {
  display: flex;
  align-items: center;
  gap: 12px;
  background: white;
  padding: 10px 14px;
  border-radius: 16px;
  border: 1px solid var(--border);
}

.profile small { display: block; color: var(--muted); }

.avatar {
  width: 42px;
  height: 42px;
  border-radius: 50%;
  background: #dbeafe;
  color: #1d4ed8;
  display: grid;
  place-items: center;
  font-weight: 800;
}

.stats-grid, .cards-grid, .analytics-grid {
  display: grid;
  gap: 18px;
}

.stats-grid {
  grid-template-columns: repeat(4, 1fr);
  margin-bottom: 18px;
}

.stat-card, .panel, .project-card, .task-card {
  background: var(--panel);
  border: 1px solid var(--border);
  border-radius: 18px;
  padding: 20px;
  box-shadow: 0 8px 30px rgba(15, 23, 42, .04);
```

```
}

.stat-card span { color: var(--muted); }
.stat-card strong {
  display: block;
  font-size: 36px;
  margin-top: 8px;
}

.panel-head {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 18px;
}

.panel-head h3 { margin: 0; }

.cards-grid { grid-template-columns: repeat(3, 1fr); }

.project-card h4 { margin: 0 0 8px; }
.project-card p { color: var(--muted); }

.kanban {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 18px;
}

.column {
  background: #eaf0fa;
  border-radius: 18px;
  padding: 14px;
  min-height: 520px;
}

.column h3 { margin: 8px 6px 14px; }

.task-card {
  margin-bottom: 12px;
  padding: 14px;
}

.task-card h4 { margin: 0 0 8px; }
.task-card p { color: var(--muted); margin: 0 0 10px; }

.badge {
  display: inline-block;
  padding: 4px 8px;
  border-radius: 999px;
  font-size: 12px;
  font-weight: 700;
  background: #e0e7ff;
```

```
  color: #3730a3;
}

.priority-High { background: #fee2e2; color: #991b1b; }
.priority-Medium { background: #fef3c7; color: #92400e; }
.priority-Low { background: #dcfce7; color: #166534; }

.filters {
  display: grid;
  grid-template-columns: 1fr 220px;
  gap: 12px;
  margin-bottom: 14px;
}

table {
  width: 100%;
  border-collapse: collapse;
  background: white;
}

th, td {
  padding: 12px;
  border-bottom: 1px solid var(--border);
  text-align: left;
  font-size: 14px;
}

th {
  color: var(--muted);
  font-weight: 700;
}

.action-btn {
  padding: 7px 10px;
  font-size: 12px;
  margin-right: 5px;
}

.delete { background: var(--danger); }

.analytics-grid {
  grid-template-columns: 1fr 1fr;
}

.bar-row {
  margin: 14px 0;
}

.bar-label {
  display: flex;
  justify-content: space-between;
  font-size: 14px;
  margin-bottom: 6px;
```

```
}

.bar-track {
  height: 12px;
  background: #e5e7eb;
  border-radius: 999px;
  overflow: hidden;
}

.bar-fill {
  height: 100%;
  background: var(--primary);
}

.modal {
  position: fixed;
  inset: 0;
  background: rgba(15, 23, 42, .6);
  display: grid;
  place-items: center;
  padding: 20px;
}

.modal-card {
  width: 520px;
  max-height: 92vh;
  overflow: auto;
  background: white;
  border-radius: 20px;
  padding: 24px;
}

.modal-card h3 { margin-top: 0; }

.modal-actions {
  display: flex;
  justify-content: flex-end;
  gap: 10px;
  margin-top: 18px;
}

.secondary {
  background: #64748b;
}

@media (max-width: 900px) {
  .sidebar { width: 210px; }
  .stats-grid, .kanban, .cards-grid, .analytics-grid {
    grid-template-columns: 1fr;
  }
}
```