МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ З ВИКОРИСТАННЯМ МЕТОДОЛОГІЇ AGILE»**

Виконав: студент 4 курсу, групи 6.1221з
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
В.О. Кравченко
(ініціали та прізвище)
Керівник доцент, PhD, Калюжняк А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент професор кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кравченко Вадиму Олеговичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка інформаційної системи управління проєктами з використанням методології Agile

керівник роботи Калюжняк Анастасія Вікторівна, PhD, доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2002-с

2. Строк подання студентом роботи 22.05.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Назвати розділ у відповідності з темою.

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

7. Дата видачі завдання 24.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	24.12.2024	
2.	Збір вихідних даних.	10.01.2025	
3.	Обробка методичних та теоретичних	07.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	15.03.2025	
5.	Розробка третього розділу.	26.04.2025	
6.	Оформлення та нормоконтроль	16.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	29.05.2025	

Студент _____ В.О. Кравченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Калюжняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної си стеми управління проєктами з використанням методології Agile»: 59 с., 25 рис., 3 табл., 29 джерел, 1 додаток.

AGILE, ІНФОРМАЦІЙНА СИСТЕМА, МЕТОДОЛОГІЯ, ПРОЄКТ, УПРАВЛІННЯ.

Об’єкт дослідження – методології управління проєктами.

Мета роботи: створити прототип інформаційної системи для здійснення управління проєктами, використовуючи методологію Agile.

Метод дослідження – класифікація, систематизація, узагальнення, моделювання.

В представленій кваліфікаційній роботі було розглянуто особливості роботи методологій Agile, її структуру, основні переваги та недоліки.

Проаналізовано існуючі підходи до організації командної роботи, принципи гнучкого управління проєктами та особливості програмної реалізації таких систем. Особливу увагу приділено питанням побудови зручного інтерфейсу користувача, структурування даних та забезпечення ролей доступу.

У практичній частині розроблено програмний застосунок мовою Python із використанням бібліотеки Tkinter та системи керування базами даних SQLite. Застосунок забезпечує створення і керування проєктами, завданнями, призначення виконавців, формування звітності та графічне відображення робочих процесів. Передбачено можливість додавання коментарів до завдань, а також адміністрування користувачів.

Результати роботи демонструють ефективність обраної архітектури та потенціал запропонованого рішення для застосування в невеликих командах або навчальних цілях.

SUMMARY

Bachelor’s Qualifying Theses «Development of an information system for project management using Agile methodology»: 59 pages, 25 figures, 3 tables, 29 references, 1 supplements.

AGILE, INFORMATION SYSTEM, MANAGEMENT, METHODOLOGY, PROJECT.

Objective: to create a prototype of an information system for project management using the Agile methodology.

Research method-classification, systematization, generalization, modeling. In the presented qualification work, the features of Agile methodologies, its structure, main advantages and disadvantages were considered.

The article analyzes existing approaches to the organization of teamwork, principles of flexible project management, and features of software implementation of such systems. Special attention is paid to building a user-friendly interface, structuring data, and providing access roles. In the practical part, a Python software application was developed using the Tkinter library and the SQLite database management system.

The application provides creation and management of projects, tasks, assignment of performers, reporting, and graphical display of work processes. You can add comments to tasks, as well as administer users. The results demonstrate the effectiveness of the chosen architecture and the potential of the proposed solution for use in small teams or training purposes.

ЗМІСТ

Реферат 4
Summary 5
Вступ 7
1 Теоретичні Аспекти теорії управління інформаційними проєктами з використанням методології agile 9
1.1 Методологія гнучкого управління проєктами Agile 9
1.2 Класифікація програмних засобів для управління проєктами 14
2 Проєктування інформаційної системи для управління проєктами із використанням методології Agile 28
2.1 Опис поточної реалізації 28
2.2 Візуалізація архітектури системи 32
3 опис процесу реалізації інформаційної системи керування проєктом 39
3.1 Опис фрагментів розробки 39
3.2 Демонстрація результатів роботи інформаційної системи 44
Висновки 51
Перелік посилань 53
Додаток А Програмний код 56

ВСТУП

В останні 20 років традиційні підходи до проєктної діяльності не втрачають своєї актуальності – загалом з тих самих причин, що й 50-60 років тому. Багато хто бачать рішення щодо застосування гнучких підходів до проєктної діяльності. Найбільш популярним з них є Agile.

Agile як філософія сформувався порівняно недавно – у середині 90-х - на початку 2000-х років. І лише 2001 року з'явився Agile-маніфест, який ознаменував революцію у застосуванні гнучких методологій не лише у розробці, а й у багатьох інших сферах людської діяльності – від ІТ-сфери та виробництва до маркетингу та продажів, а зараз, і медицини та освіти.

Ідея гнучкої методології розробки Agile полягає в тому, щоб співробітники та керівники вміли дуже швидко знаходити нові рішення та за необхідності створювати нові продукти для збільшення прибутку компанії та її загальної конкурентоспроможності на ринку.

По суті, Agile-підхід будується на принципах проєктного управління, і більшою мірою властивий компаніям, які вибудовують управлінський процес у вигляді формування команд зі своїх співробітників.

З іншого боку, Agile є гнучким інструментом управління персоналом, який може змінювати свій механізм залежно від впливу факторів зовнішнього і внутрішнього середовища, що виникають в результаті будь-яких явищ.

Дана методологія дозволяє вибудовувати максимально ефективний робочий процес, а також змінювати мислення та культуру співробітників, не втрачаючи фокус на цільові орієнтири компанії, і отримуючи відповідний результат, виражений у зростанні продуктивності праці та результативності роботи співробітників.

У підході Agile використовують безліч методів, які дозволяють організовувати роботу персоналу, утворюючи самодостатні команди. У цьому роль управляючих властивостей грають безпосередньо регламенти, регулюючі процеси у створенні, спрямовані в розвитку цілеспрямованості та режиму

функціонування персоналу.

На відміну від традиційних методик Agile є більш гнучкою та м’якою, дозволяючи розвивати лояльність персоналу до керівництва, формуючи при цьому вищий рівень мислення та креативності у кадрів. Як правило, такі методики впроваджуються в наукомісткі та IT компанії, в яких працюють люди інтелектуальної праці.

Метою кваліфікаційної роботи є розробка інформаційної системи для ефективного управління проєктами на основі методології Agile, яка забезпечить автоматизацію планування, моніторингу задач та аналізу продуктивності команди.

Для досягнення поставленої мети потрібно виконати наступні завдання:

- проаналізувати принципи Agile та вимоги до сучасних систем управління проєктами;
- провести класифікацію програмних засобів для управління проєктами;
- визначити функціональні та нефункціональні вимоги до системи;
- спроєктувати архітектуру інформаційної системи з використанням UML-діаграм;
- реалізувати прототип системи з базовими функціями (беклог, дошка задач, звітність);
- провести тестування системи на реальному проєкті та оцінити її ефективність.

Об’єктом кваліфікаційної роботи є процес управління проєктами за методологією Agile та інструменти його підтримки.

Предметом кваліфікаційної роботи є інформаційна система для автоматизації Agile-управління проєктами, включаючи механізми планування, виконання задач та аналізу результатів.

Методами дослідження в кваліфікаційній роботі слугують аналіз літератури, класифікація, систематизація, узагальнення, порівняння, моделювання.

Структура кваліфікаційної роботи складається із вступу, трьох основних розділів роботи, висновків, списку використаних джерел та додатків.

1 ТЕОРЕТИЧНІ АСПЕКТИ ТЕОРІЇ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ ПРОЄКТАМИ З ВИКОРИСТАННЯМ МЕТОДОЛОГІЇ AGILE

1.1 Методологія гнучкого управління проєктами Agile

Проєктне управління є сучасним управлінським інструментом, що сприяє успішній реалізації нових ідей та починань, що розробляються як на рівні окремих підприємств, так і приватних осіб. В останні роки проєкти отримали визнання у багатьох сферах бізнесу, починаючи від великих промислових компаній, до соціально орієнтованих некомерційних систем. Підвищений інтерес до проєктів доводить і запровадження проєктних офісів у державних службах, що закріплено на законодавчому рівні. При цьому безліч автоматизованих рішень у галузі управління проєктами також свідчать про популярність цього підходу в управлінському середовищі. Однак для організації управління проєктами потрібна наявність спеціальних знань у галузі інформаційного забезпечення та розміщення бізнес-процесів всередині керованого середовища. Тому актуальними є численні навчально-практичні заходи, написані з урахуванням нових наукових шкіл, розробниками яких є різні вчені [1].

Загалом, управління проєктами це механізм реалізації конкретних дій лише на рівні окремих організаційних систем чи приватних осіб. При цьому пропонується процесний, системний, структурний та ризик-орієнтований підходи. Розглянемо докладніше кожен із них.

Так ряд вчених представляють проєктне управління на основі процесного підходу, бачачи його як поділ окремих дій по процесах, що виділяються в етапи проєктування конкретних кроків [2].

Інші вчені вважають, що проєктна діяльність може бути представлена як система, яка інтегрована в управлінське середовище організації, функціонує за

допомогою проєктного офісу, в якому всі процеси є невеликими окремими проєктами.

Також є інші підходи до розуміння управління проєктами. Відповідно до одного з них – це інформаційне середовище, що забезпечує реалізацію окремих кроків та етапів щодо впровадження конкретних продуктів, а також їх функціонування та контролю ефективності реалізації.

Останнє визначення більш підходить до банківського сектору, оскільки в банках проєктні офіси створюються далеко не завжди, проте в останні роки більша частина з них дедалі активніше реалізує свої цілі та завдання за допомогою проєктного управління при реалізації конкретних ідей [3].

Останніми роками технологія управління проєктами зміщується до дедалі більшого використання гнучких методів. Вони дозволяють перебудовувати етапи під час проєктування та більш гнучко приймати зміни внаслідок впливу зовнішніх та внутрішніх факторів. Розглянемо докладніше.

Тенденції розвитку гнучкого підходу розглядає Moloto M, який уявляє, що вони найбільш поширені в системах великих даних та хмарних обчислень. Автор розповідає про наявність безлічі типів гнучких розробок, в результаті чого виділяє їх переваги і недоліки, а оптимальної серед них не виділяє.

Він вважає, що кожен проєкт має свою специфіку, володіючи характеристиками, елементами, внаслідок чого найкращою гнучкою методологією виступає ретельно продумана власна індивідуальна методика, яка враховує всі ці змінні [4].

Agile-проєкти в будь-якому випадку успішніший за традиційний підхід, що доводиться практичним застосуванням проєктного підходу та пошуком нових рішень у створенні інновацій. Далі наведено порівняльну таблицю класичного та Agile-підходу (табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця класичного та Agile-підходу

Характеристика	Класичний підхід	Agile
Постачання цінності (працюючого результату)	Відбувається наприкінці проєкту	Здійснюється протягом реалізації проєкту як діючих елементів товару. Використовується ітеративно-інкрементальний підхід.
Перевірка гіпотез	Як правило, виконується на передпроєктній стадії, до старту проєкту	Виконується командою під час проєкту поліпшення продукту. Частина гіпотез може бути визнана неспроможними
Планування	Детальне, до кінця проєкту. Для оцінки термінів використовується метод критичного шляху.	Емпіричне, на основі історичних даних про реалізовані елементи продукту
Стиль менеджменту, керівництва	Вертикаль управління: Керуючий комітет -> Куратор, Замовник -> Керівник проєкту	Самоорганізація всередині команди. Плоска команда без внутрішньої ієрархії.
Ставлення до змін	Як правило, має негативний характер – зміни як наслідки реалізації ризиків та настання проблем. Вимагає формального процесу з аналізу наслідків та перерахунку критичного шляху проєкту, аналізу альтернатив.	Зміни є частиною процесу розробки. Джерелом змін є зокрема. краще розуміння продукту на основі досвіду.
Тип мислення, стосунків	Як правило визначається культурою організації, найчастіше фіксований mindset	Необхідний гнучкий mindset для успішної роботи в середовищі з високою невизначеністю
Метрики проєкту	% реалізації, відхилення від плану, метод освоєного обсягу, прогнозна дата завершення проєкту	Діаграма згоряння завдань (Burn down chart), Накопичувальна діаграма реалізованих функцій (Burn Up Chart), Дата виходу на ринок (Time to market)
Наявність посібників, методик	Добре структуровані, детально описані (PMBoK, PRINCE2). Галузеві стандарти та практики.	Верхньорівневі фреймворки (наприклад, Scrum). Безліч окремих практик (щоденні збори, ретроспектива, спринт та ін.)
Область ефективного застосування	«Складні системи» за моделлю Кіневін (Cynefin) – багато робіт, агентів (стейкхолдерів). Продукт та вимоги до нього відомі, Склад робіт може бути описаний та зафіксований. Кордони проєкту фіксовані.	«Заплутані системи» за моделлю Кіневін (Cynefin) – не знаємо продукт та/або процес його створення. Склад робіт проєкту не визначено. Кордони проєкту розмиті.

Далі варто відзначити, що в даний час існує кілька різновидів методології Agile, які представлені на рис 1.1

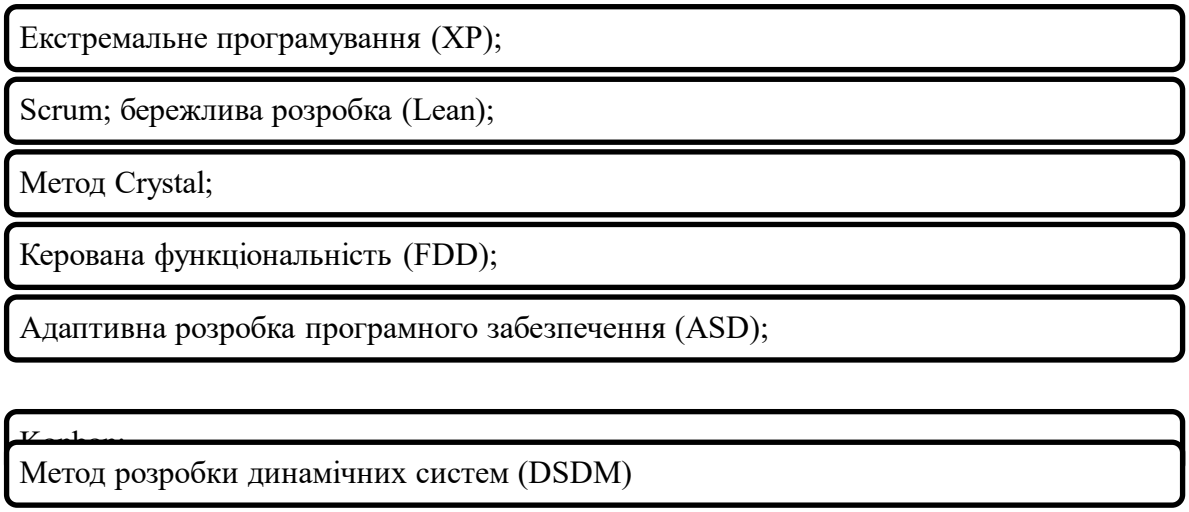


Рисунок 1.1 – Різновиди методології Agile

Всі ці методи націлені на реалізацію основних методик Agile-маніфесту, і більш конкретно описано у таблиці 1.2.

Таблиця 1.2 – Опис найпопулярніших гнучких методів

Найменування гнучкого методу	Опис
Scrum	Scrum передбачає спільну реалізацію проєкту кросфункціональною командою з чітким поділом за ролями, проведенням різних конференцій
Kanban	Особливість Kanban полягає у візуалізації завдань проєкту, що описуються на окремих картках, що додаються на дошку завдань (справжню або віртуальну).
Scrumban	Scrumban поєднує підходи Scrum і Kanban. У Scrumban командна робота організована у вигляді невеликих ітерацій (спринтів) і контролюється з використанням візуальної дошки. Члени команди самі обирають завдання, а командні ролі Scrumban більш спеціалізовані і менш крос-функціональні, ніж у Scrum командах
Екстремальне програмування	Екстремальне програмування (eXtreme Programming, XP) використовується виключно у розробці програмних продуктів.
Lean	Lean Soft ware Development (ощадлива розробка програмного забезпечення) – гнучка методологія, заснована на концепції ощадливого виробництва.

Як показала практика, найбільш використовуваним став метод Scrum технологій, а також scrumcanban, який виходив від методології «Kanban», як варіант управління Scrum полягає в тому, що він весь процес розділяє на спринти, що вимагають приблизно однакового часу.

Scrum прискорює час реалізації проєкту, знижує собівартість його управління, забезпечуючи узгоджену роботу з клієнтами та учасниками команди [5].

У той же час Kanban виконує завдання точно та вчасно. Він розподіляє навантаження між учасниками команди, демонструє, що в процесі роботи завдяки йому відбувається спільне розуміння між учасниками.

Симбіозом між Kanban і Scrum стала методологія scrumcanban, яка внесла зміни до стандартної практики і даний метод вважається кращим за інтегровані методики.

Ще одним поширеним методом є Lean-методика, яка включає поділ по процесах і управління кожним з них. По суті, «Lean» націлена не лише на скорочення споживання матеріальних ресурсів, а й на їх економію. До того ж основна мета цієї системи полягає в усуненні втрат та зменшенні кількості браку. За допомогою цього інструменту виявляються приховані втрати, а також причини, які спричиняють зниження продуктивності та мотивації праці співробітників. Тому саме «Lean» дає можливість виявляти втрати, розробляти заходи щодо їх усунення [7].

У той же час реалізація концепції Lean формує набір проблем, які наведені на рис. 1.2.



Рисунок 1.2 – Проблеми при впровадженні системи Lean

Так недостатня структуризація застосування цієї системи не дозволяє

ефективно інтегрувати її у проєктний процес підприємства. При цьому принципово важливим є пошук інструментів, які дозволять реалізувати систему Lean, переформатувавши управління проєктами.

Загалом концепція «Lean» прагне до мінімізації втрат та створення безперервного потоку процесів проєкту [6].

Представлена управлінська парадигма успішно інтегрується до управлінських систем багатьох підприємств світу, довівши тим самим свою дієвість і доцільність застосування.

1.2 Класифікація програмних засобів для управління проєктами

Інструменти управління проєктами спеціально розроблені, щоб допомогти людині чи команді ефективно керувати проєктами та завданнями. Термін «інструменти управління проєктами» зазвичай стосується програмного забезпечення для управління проєктами, яке можна придбати або використовувати безкоштовно в Інтернеті.

У сучасному бізнес-середовищі, що швидко розвивається, управління проєктами стало критично важливим компонентом для забезпечення успіху організацій. Ефективне управління проєктами передбачає планування, організацію та управління ресурсами для досягнення конкретних цілей у визначений термін. Однак керування проєктами вручну може бути складним, займати багато часу та призводити до помилок. Програмне забезпечення для управління проєктами з’явилося як рішення для цих проблем, надаючи широкий спектр функцій і можливостей [8].

Хоча на ринку є багато рішень, найкращий веб-інструмент керування проєктами залежатиме від кількох факторів:

– кількість користувачів визначає скільки людей використовує платформу управління проєктами і на основі цієї інформації, ідеальна платформа

запропонує відповідні варіанти як для починаючих стартапів, так і для підприємств, що розвиваються [9];

– структура ціноутворення, яка визначає, чи є можливість спробувати інструмент безкоштовно для оцінки наявних функцій;

– необхідні ключові функції, але потрібно переконатися, що вибрана платформа надає рішення для конкретних проблемних моментів [10];

– масштабованість, тобто, чи може платформа управління проєктами розвиватися разом з потребами компанії;

– простота використання, чи легко впровадити вибраний додаток для керування проєктами, чи може команда негайно ним скористатися, чи зможе команда легко адаптуватися до нових функцій.

Відповіді на ці запитання допоможуть звузити вибір, щоб знайти найкраще програмне забезпечення для керування проєктами.

Основними засобами для управління проєктами є:

Wrike – це відзначена нагородами платформа управління роботою, яку використовують понад 20 000 компаній і 2,4 мільйона людей у понад 140 країнах. З набором інструментів для всіх, від менеджерів проєктів до керівників маркетингу, Wrike є першим вибором для команд, відділів і організацій у будь-якій галузі та сфері [11].

Плюси і мінуси

Wrike має неперевершений набір функцій, включаючи вбудоване планування ресурсів, відстеження часу та функції бюджетування, які допомагають сучасним командам робити більше з меншими витратами. Він давно відкрив переваги автоматизації та штучного інтелекту, допомагаючи клієнтам економити час і ресурси, а також прогнозувати ризики проєкту, створювати розширені звіти та автоматично розставляти пріоритети [11].

Ключові особливості

Керування завданнями, дошки Kanban: доступ до інформаційної панелі керування завданнями в реальному часі для швидкої та легкої співпраці.

Налаштовувані діаграми Ганта: одна з найпопулярніших функцій Wrike, діаграма Ганта, допомагає візуалізувати терміни виконання та контролювати хід виконання проєкту в реальному часі [12].

Потужна аналітика Wrike надає цінну інформацію про продуктивність усієї компанії та видає звіти в реальному часі за лічені секунди.

Інтеграції

Wrike має понад 400 інтегрованих додатків, щоб клієнти могли легко переходити між платформою та іншими улюбленими додатками, такими як Google Drive, Google Calendar, Slack тощо. Він також пропонує вбудовану інтеграцію Adobe CC і DAM , тому користувачі можуть створювати, редагувати та публікувати безпосередньо з Wrike.

Цінові плани:

– безкоштовно – 0$;

– команда – 10$ за користувача/місяць (плата виставляється щорічно);

– бізнес – 24,80$ за користувача/місяць (рахунок виставляється щорічно);

– підприємство – спеціальні ціни для великих команд (рахунок виставляється щорічно);

– pinnacle – спеціальне ціноутворення для команд із складними потребами.

Asana – одна з найпопулярніших програмних платформ для управління проєктами. Створений для підприємств різного розміру, цей багатогранний інструмент поєднує в собі сховище файлів, дорожні карти проєктів, інформаційні панелі в одному привабливому інтерфейсі [13].

Плюси і мінуси

Asana – це комплексна платформа для управління проєктами, яка пропонує різні функції організації роботи, інструменти управління ресурсами та привабливий інтерфейс.

Однак йому не вистачає розширених можливостей перевірки, а також можливості легко налаштувати надійне керування обліковим записом.

Користувачі зі складними проєктами можуть виявити, що платформі важко задовольнити їхні потреби [13].

Ключові особливості

Функція Workflow Builder Asana – це візуальний інструмент, який дозволяє користувачам створювати власні робочі процеси та об’єднувати членів команди.

За допомогою стратегічних цілей в Asana менеджери проєктів можуть надавати командам напрямки та налаштовувати показники для відстеження ефективності на цьому шляху.

Діаграми Ганта – це один із найпривабливіших способів перегляду ресурсів, термінів виконання та загального прогресу роботи.

Інтеграції

Asana має понад 200 інтегрованих додатків, за допомогою яких клієнти можуть продовжувати користуватися улюбленими додатками, такими як Google Drive, Google Calendar, Slack тощо.

Цінові плани:

– базовий: 0$;

– преміям: 13,49$ за користувача/місяць (плата виставляється щомісяця);

– бізнес – 30,49$ за користувача/місяць (плата виставляється щомісяця);

– enterprise: custom.

Monday.com

Monday.com – це популярне програмне забезпечення для керування проєктами, яке містить різноманітні шаблони та інструменти для оптимізації операцій і допомоги командам у досягненні вищих рівнів продуктивності. Програмне забезпечення відображає багато інформації для команд, але ключові дані можуть бути пропущені в довгих ланцюжках подій [17].

Плюси і мінуси

Monday.com є одним із найкращих варіантів для менеджерів проєктів, оскільки він має простий у використанні UX і повний набір функцій керування

проєктами, включаючи автоматизацію, налаштовувані інформаційні панелі та бібліотеку шаблонів.

Monday не має власних функцій перевірки правопису та аналітики, достатньо просунутої для багатьох складних організацій.

Ключові особливості

Спеціальні інформаційні панелі: Monday.com дозволяє користувачам створювати власні інформаційні панелі та переглядати інформацію про проєкт з різних точок зору. За допомогою цих інформаційних панелей менеджери проєктів і керівники груп можуть миттєво оцінити прогрес на основі даних у реальному часі з часових шкал і інформації про бюджет.

Автоматизація: Monday.com пропонує потужні функції автоматизації для оптимізації робочих процесів. Завдання можна налаштувати за допомогою автоматичних нагадувань, близьких до термінів виконання, а керівники команд можуть створювати сценарії «якщо це, то це», щоб пришвидшити прогрес [16].

Попередньо створені шаблони: користувачі можуть скористатися перевагами бібліотеки Monday.com із 100 різних шаблонів, щоб швидко розпочати проєкти, забезпечити одноманітність і високі стандарти в усій організації та створювати повторювані процеси для типових завдань.

Інтеграції

Хоча Monday.com має менше інтеграцій, ніж деякі інші конкуренти, він все одно синхронізується з найбільш затребуваними програмами, включаючи Google, Slack, Salesforce тощо.

Цінові плани:

– фізична особа: 0$;

– базовий: 30$ на місяць (плата виставляється щомісяця);

– стандартний: 36$ на місяць (плата виставляється щомісяця);

– pro: 60$ на місяць (плата виставляється щомісяця);

– enterprise: custom.

Adobe Workfront

Adobe Workfront зазвичай віддають перевагу маркетинговим командам як рішення для управління проєктами. Проте він все ще має широкий спектр функцій, що допомагає виправдати його відносно високу ціну. Особливо це стосується організацій розміру підприємства, яким потрібні складніші можливості [19].

Плюси і мінуси

Наповнений усіма основними функціями керування проєктами, які знадобляться звичайній команді, Adobe Workfront пропонує діаграми Ганта, функції звітності та розширені інформаційні панелі, а також потужні можливості перевірки правопису та керування ресурсами.

Однак деякі користувачі повідомляють, що найновіший UX не зовсім забезпечує простоту використання та розширену співпрацю. Це може призвести до обмеженого впровадження новими користувачами, які можуть за замовчуванням використовувати наявні інструменти. Крім того, їх функція DAM має помилки, що може спричинити затримку публікації.

Ключові особливості

Планувальник сценаріїв: планувальник сценаріїв Workfront є хорошим інструментом, який допоможе передбачити різні ситуації та підготуватися до них.

Стратегічне встановлення цілей: ця функція допомагає командам встановлювати чіткі, вимірювані цілі для виконання командами. Це допомагає забезпечити вирівнювання в ході роботи [19].

Інтеграції

Adobe Workfront є частиною дуже відомої та шанованої групи Adobe. Це означає, що він забезпечує бездоганну інтеграцію з більшістю дочірніх платформ, включаючи Creative Cloud. Однак він пропонує дуже обмежену кількість сторонніх інтеграцій.

Smartsheet. Заснований на традиційній моделі електронних таблиць, Smartsheet є однією з доступних платформ для візуального керування проєктами. Він спирається на хмарні електронні таблиці та автоматизовані

робочі процеси, щоб надати функції та переваги керування проєктами та завданнями [20].

Плюси і мінуси

Smartsheet має багату технічну функціональність, розширені інструменти та комплексні можливості, які досить добре обслуговуватимуть великі організації.

Недоліком є те, що для користувачів, які не мають глибоких знайомств або знань про електронні таблиці, може знадобитися процес навчання. Його додатки також можуть стати досить дорогими [20].

Ключові особливості

Автоматизовані робочі процеси – за допомогою Smartsheet можна автоматизувати свої робочі процеси за допомогою повторюваних креслень, щоб зробити доставку більш оптимізованою, точною та швидкою.

Керування вмістом – Smartsheet пропонує такі інструменти керування вмістом, як панелі прогресу, автоматичні сповіщення та інструменти керування ресурсами.

Детальні інформаційні панелі дають можливість візуалізувати всю інформацію про проєкт, робочий процес команди та організаційні цілі за допомогою інформаційних панелей, з даними [20].

Інтеграції

Smartsheet має понад 100 інтеграцій, а також додаткові параметри через Zapier і Automate.io.

Цінові плани:

– безкоштовно: 0$;

– pro: 9$ за користувача/місяць (плата виставляється щомісяця; максимум 10 користувачів);

– бізнес: 32$ за користувача/місяць (плата виставляється щомісяця; мінімум 3 користувачі);

– enterprise: custom.

Jira. Зараз Jira є частиною Atlassian Group, але вона виникла як рішення для розробки програмного забезпечення багато років тому. Відтоді він перетворився на загальну платформу, яка включає програмне забезпечення Jira, Jira Core та Jira Work Management, які разом пропонують допомогу в управлінні роботою для команд будь-якого розміру [21].

Плюси і мінуси

Його також можна налаштувати, що допомагає командам адаптувати свій робочий простір відповідно до їхніх бізнес-процесів.

Однак він не може підтримувати методологію Waterfall, лише Agile. Він також не має вбудованого планування ресурсів, перевірки правопису та звітності про проєкт, що може бути обмеженням для деяких команд. Його ціна також ставить його серед дорожчих доступних варіантів.

Ключові особливості

Перегляд інформаційної панелі – інформаційні панелі Jira допомагають користувачам планувати проєкти, вимірювати прогрес і відстежувати терміни з першого погляду. Ними також можна ділитися, щоб оновлювати колеги.

Унікальна функція пошуку – Jira використовує свою мову запитів і спеціальні фільтри, щоб полегшити та ефективніше шукати завдання та проєкти [16].

Форми запитів на роботу, які можна настроювати: адаптуючи форми запитів, команди можуть адаптувати свій прийом, щоб зробити його більш оптимізованим, з меншою кількістю запитань і швидшою доставкою.

Інтеграції

Завдяки численним дочірнім сайтам Jira дозволяє легко інтегруватись з іншими частинами Atlassian Group.

Цінові плани:

– безкоштовно: 0$ за користувача/місяць;

– стандартний: 7,75$ за користувача/місяць (плата виставляється щомісяця; у середньому);

– преміум: 15,25$ за користувача/місяць (рахунок виставляється щомісяця; у середньому);

– enterprise: custom.

ClickUp – це добре відоме програмне рішення для керування проєктами, яке добре працює для команд більшості розмірів, у тому числі тих, що працюють у віддалених середовищах. Він має всі стандартні функції керування, які підтримують планування, організацію та керування різними завданнями [34].

Плюси і мінуси

ClickUp – це комплексне рішення для управління проєктами з доступною ціною та всіма інструментами, які потрібні більшості команд, включаючи автоматизацію робочого процесу, діаграми Ганта та функції керування ресурсами [15].

Однак він також досить нестабільно працює, з відносно високим рівнем простою. Це може розчарувати команди, які працюють у стислі терміни. Його функції перевірки та затвердження також обмежені порівняно з іншими подібними платформами.

Ключові особливості

Центр документів – дає можливість централізувати всі внутрішні документи, вікі-ресурси та поширені запитання, щоб створити базу знань, яка допоможе прискорити та згладити процес адаптації.

Контрольні списки – можна розбивати завдання на невеликі дії в списку, щоб швидше розпочати проєкти, швидше визначити перешкоди та ефективніше контролювати прогрес [15].

Інтелектуальні карти – дають можливість планувати проєкти та завдання візуально, щоб кожен міг побачити основну стратегію та зрозуміти свою роль у досягненні мети.

Інтеграції

ClickUp є сильним інтегратором із зв’язками між усіма основними програмами сторонніх розробників, включаючи Slack, Календар Google, Outlook, HubSpot тощо.

Цінові плани:

– безкоштовно назавжди – 0$;

– необмежений: 9$ за користувача/місяць (плата виставляється щомісяця);

– бізнес: 19$ за користувача/місяць (плата виставляється щомісяця);

– business Plus: 29$ за користувача/місяць (плата виставляється щомісяця);

– enterprise: custom.

Microsoft Project є частиною бренду Microsoft, якому довіряють мільйони людей у всьому світі. Він був розроблений, щоб об’єднати в одній платформі багато функцій керування проєктами інших дочірніх брендів. Він добре відомий як ефективний для окремих проєктів, хоча не такий популярний, коли мова йде про кілька поточних проєктів [11].

Плюси і мінуси

Microsoft Project

Microsoft Project є частиною бренду Microsoft, якому довіряють мільйони людей у всьому світі. Він був розроблений, щоб об’єднати в одній платформі багато функцій керування проєктами інших дочірніх брендів. Він добре відомий як ефективний для окремих проєктів, хоча не такий популярний, коли мова йде про кілька поточних проєктів [22].

Плюси і мінуси

Microsoft Project пропонує надійний набір інструментів для сучасних команд, включаючи діаграми Ганта, сітки та дошки, а також різні функції керування ресурсами, які допомагають керувати завданнями та командами.

Microsoft Project найкраще працює разом з іншими продуктами MS, такими як Outlook, Teams і SharePoint, для належної співпраці та міжфункціональної роботи. Це може розчаровувати людей, які шукають єдине платформне рішення «все в одному».

Ключові особливості

Інтерактивні інформаційні панелі – для цього потрібна інтеграція з Power BI, але можливість швидко переглядати загальний стан проєкту та індивідуальний прогрес за допомогою інтерактивних інформаційних панелей є бонусом.

Перегляд часової шкали Ганта – можна сканувати майбутні терміни виконання, прогрес за допомогою постійно популярного перегляду Ганта.

Відстеження бюджету – хоча ця зручна функція оптимізована лише для окремих проєктів, дозволяє контролювати витрати, дотримуючись бюджету та вчасно [23].

Інтеграції

Як і слід було очікувати, Microsoft Project дуже сильний щодо інтеграції з іншими частинами екосистеми Microsoft, але меншою мірою з іншими програмами сторонніх розробників. Однак він дозволяє підключатися до Slack, Google Calendar, Outlook, HubSpot тощо.

Цінові плани:

– план проєкту 1$ - 10$ за користувача на місяць;

– план проєкту 3$ - 30$ за користувача на місяць;

– план проєкту 5$ - 55$ за користувача на місяць;

– ціни на локальні рішення відрізняються.

Basecamp – це проста у використанні платформа, особливо корисна для невеликих команд, яким потрібні функції керування проєктами, які включають функції планування, співпраці та моніторингу.

Плюси і мінуси

Basecamp – це дуже простий варіант із ціною, яка базується на вимогах, а не на користувачах, що привабить менші команди чи компанії, що розвиваються. Через його простоту він також відносно простий у навігації та навчанні, тому він має короткий період навчання [23].

У масштабі Basecamp може не вистачати більш складних функцій, необхідних для міжфункціональної співпраці між зростаючими командами.

Незважаючи на те, що він має функцію «Line-Up», є обмежені способи перегляду роботи, на відміну від інших продуктів із налаштовуваними діаграмами Ганта, дошками Kanban, переглядами таблиць тощо.

Ключові особливості

Списки справ: швидко візуалізуються робочі навантаження та розділяються складні проєкти за допомогою списку справ для кожного проєкту, доповненого окремими примітками та вкладеннями.

Дошки оголошень: підтримка постійного зв'язку у спеціальних дошках, допомагаючи командам працювати разом, щоб просувати певний проєкт.

Планування проєкту: ця функція допомагає користувачам не пропустити ключові терміни [23].

Інтеграції

Незважаючи на те, що Basecamp не пропонує такого широкого діапазону, як деякі інші доступні програмні рішення, він допускає інтеграцію з багатьма програмами сторонніх розробників, які мають додаткові можливості розробки програмного забезпечення, управління проєктами та звітності.

Цінові плани:

– basecamp: 15$ за користувача/місяць;

– basecamp pro unlimited: 349$ на місяць (рахунок виставляється щомісяця; кількість користувачів необмежена).

Trello є одним із найперших інструментів на основі канбану. Він все ще дозволяє просту візуалізацію проєкту та ефективне керування завданнями за допомогою перетягування. Він знайомий багатьом людям, що дозволяє легко ділитися та співпрацювати навіть серед зовнішніх користувачів [24].

Плюси і мінуси

Trello робить керування завданнями цікавим завдяки привабливим дошкам і простому у використанні інтерфейсу, які роблять його чудовим для оптимізації продуктивності команди.

Однак він обмежений з точки зору загальноорганізаційних функцій управління проєктами та портфелем, тому, якщо потрібно розробляти стратегію

або планувати на високому рівні, можливо, краще підійде більш комплексне програмне забезпечення для керування проєктами.

Ключові особливості

Trello – один із найвідоміших інструментів дошки Kanban із привабливим інтерфейсом і простою функцією перетягування.

Користувальницькі картки – персоналізовані картки із завданнями, з датами виконання, коментарями, щоб усі члени команди могли швидко бачити, що потрібно.

Автоматизація робочого процесу – створення автоматизованих потоків між картками, які допомагають оптимізувати прийом і прискорити доставку.

Інтеграції

Trello має понад 200 готових інтеграцій із Google, Slack та іншими інструментами керування роботою, які найчастіше згадуються [24].

Цінові плани:

– безкоштовно: 0$;

– стандартний: 6$ за користувача/місяць (плата виставляється щомісяця);

– преміум: 12,50$ за користувача/місяць (плата виставляється щомісяця);

– підприємство: 17,50$ за користувача/місяць (рахунок виставляється щорічно).

Правильний інструмент управління проєктами може значно покращити централізацію, координацію та комунікацію. Це також може забезпечити реальну відчутну економію для організацій у той час, коли їм потрібно скоротити витрати та підвищити продуктивність.

На відміну від інших систем, які часто базуються на жорстких водоспадних моделях, розроблювана в роботі Agile-система сприятиме динамічному управлінню проєктами, зменшуючи ризик невідповідності кінцевого продукту очікуванням клієнта.

Також, в розроблюваній інформаційній системі буде зроблено акцент на співпраці та прозорості в команді. Розроблювана система інтегруватиме інструменти для ефективної комунікації, такі як дошки задач (наприклад,

Kanban чи Scrum), чати та системи відстеження прогресу, що дозволить усім учасникам проєкту бути в курсі поточного стану робіт. Це контрастує з іншими інформаційними системами, які можуть бути менш орієнтованими на командну взаємодію і часто ускладнюють швидкий обмін інформацією. Завдяки автоматизації процесів планування спринтів, оцінки завдань та відстеження метрик, система забезпечує чітке бачення прогресу, що сприяє підвищенню продуктивності та зниженню кількості помилок у процесі управління.

Розроблювана система на основі методології Agile є більш орієнтованою на користувача, оскільки вона дозволяє легко адаптувати функціонал до специфічних потреб команди чи організації. У порівнянні з іншими інформаційними системами, які часто пропонують універсальні рішення з обмеженою гнучкістю, Agile-система забезпечує персоналізований підхід, що дозволяє оптимізувати робочі процеси та підвищити задоволеність користувачів завдяки швидкому врахуванню їхнього зворотного зв’язку.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ УПРАВЛІННЯ ПРОЄКТАМИ ІЗ ВИКОРИСТАННЯМ МЕТОДОЛОГІЇ AGILE

2.1 Опис поточної реалізації

Поточна реалізація систем управління проєктами, заснованих на Agile-методології, хоч і надає ряд корисних інструментів, має кілька спільних рис, які вимагають додаткової уваги з позиції покращення комунікативних властивостей.

Традиційні діаграми та звіти у існуючих системах управління проєктами не завжди забезпечують достатню візуалізацію поточних процесів. Поліпшення візуалізації, включаючи більш наочні дашборди та діаграми, може суттєво покращити розуміння стану проєкту [25].

Багато систем не забезпечують достатньої підтримки реального часу для оперативного реагування на зміни у проєкті. Додавання функціоналу, що дозволяє відслідковувати зміни та приймати рішення в режимі реального часу, зробить комунікації динамічнішими.

Відсутність повноцінної інтеграції з сучасними засобами комунікації, такими як чати, відеоконференції та обмін документами, може обмежувати командну взаємодію. Інтеграція з популярними платформами обміну інформацією допоможе зробити комунікації більш гнучкими.

Деякі системи не завжди адаптовані до особливостей Agile-методології, що може ускладнювати відповідність гнучким процесам розробки та швидке реагування на зміни вимог.

Поліпшення відповідності Agile принципам може підвищити ефективність системи в контексті комунікативних процесів. Поліпшення комунікативні можливості систем управління проєктами, зробивши їх адаптованішими, динамічнішими і пристосованими до вимог сучасного Agile-проєктного управління [26].

Існуючі системи управління проєктами, незважаючи на численні переваги, стикаються з обмеженнями у візуалізації процесів, що може суттєво утруднити розуміння поточного стану проєкту. Одним із ключових аспектів, що потребують поліпшення, є обмежена наочність уявлення інформації про процеси розробки.

В сучасних умовах Agile, де процеси часто зазнають змін, ітерації відбуваються швидко, і команди стикаються з частими оновленнями вимог, необхідно забезпечити динамічніші та зрозуміліші засоби візуалізації. Існуючі діаграми Ганта і статичні звіти можуть виявитися недостатніми для відстеження динаміків проєкту, що швидко змінюються [27].

Тому, поліпшення візуалізації процесів має включати:

− динамічні дашборди, які можуть надавати миттєві огляди поточного стану проєкту;

− інтерактивні звіти, які дозволяють користувачам гнучкіше взаємодіяти з даними, мають можливість фільтрації, угруповання та додаткового аналізу даних у реальному часі може зробити процес моніторингу ефективнішим;

− графіки та діаграми прогресу, які прогрес виконання завдань, допомагає команді краще розуміти, які етапи завершено, а які вимагають додаткових зусиль, що важливо для забезпечення прозорості та прийняття обґрунтованих рішень;

− візуальне відображення залежностей між завданнями та проєктними елементами, що дозволяє команді чіткіше бачити взаємозв’язки та краще координувати зусилля;

− динамічна адаптація до змін, дозволяючи в реальному часі відстежувати та візуалізувати їх вплив на процеси.

Поліпшення візуалізації процесів у системах управління проєктами допоможе командам краще орієнтуватися у поточній ситуації, швидше реагувати на зміни та підвищувати рівень прозорості у рамках Agile-підходу [5].

Проблема неінтегрованих засобів комунікації в існуючих системах управління проєктами істотно впливає на ефективність обміну інформацією всередині команд. В сучасних умовах, де команди можуть перебувати в різних часових поясах або працювати віддалено, важливо, щоб засоби комунікації були вбудовані безпосередньо в інструменти управління проєктами» [28].

Ключові аспекти, які потребують поліпшення:

− інтеграція з чатами та відеоконференціями, що дозволяє команді обмінюватися миттєвими повідомленнями та проводити наради без необхідності перемикання між різними платформами;

− спільне редагування документів, дозволяє команді працювати над документацією в режимі реального часу, не вдаючись до використання додаткових інструментів;

− повідомлення та оповіщення, забезпечують своєчасне інформування команди, що важливо для оперативної реакції на зміни;

− інтеграція з електронною поштою, дозволяє автоматизувати надсилання повідомлень та оновлень за проєктом, що покращує доступність інформації для всіх учасників;

− управління коментарями та зворотним зв’язком, сприяє більш ефективному обміну ідеями та миттєвому внесенню правок [29];

− інтеграція із сховищами файлів, дозволяє легко обмінюватися та зберігати документацію та файли проєкту у централізованому місці.

Інтеграція засобів комунікації до системи управління проєктами є важливим кроком до створення єдиної точки доступу для всієї команди. Такий підхід зменшує розрізненість інструментів, які використовуються для спілкування та обміну інформацією, та підвищує ефективність взаємодії між учасниками проєкту. Реалізація таких удосконалень сприяє покращенню комунікативних можливостей проєктних груп у рамках Agile-підходу [12].

Неадекватна адаптація систем управління проєктами до принципів Agile створює перешкоди для ефективного застосування гнучких методологій розробки.

Ключові аспекти, що вимагають покращення з погляду відповідності Agile-принципам:

– гнучкість та мінливість до частих змін у вимогах та пріоритетах;

– ітеративність, що включає можливість створення та відстеження епіків, користувальницьких історій і завдань відповідно до Agile-принципів;

– колективна робота та комунікації, яка проявляється в інтеграції чатів, форумів та інших засобів обміну інформацією підтримує командну взаємодію [30].

– автономність команд, потрібна для прийняття рішень та організації роботи, тобто система управління проєктами повинна надавати гнучкі інструменти для організації завдань та планування, щоб команди могли ефективно самоорганізовуватись;

– зворотній зв’язок та ретроспективи потрібні для безперервного покращення процесів та комунікацій відповідно до Agile-принципів;

– візуалізація прогресу та метрик потрібна для візуалізації прогресу проєкту та ключових метрик, дозволяючи команді оцінювати ефективність та приймати обґрунтовані рішення;

– підтримка частого випуску товару, повинні сприяти частому випуску продукту, автоматизуючи процеси тестування, збирання та розгортання.

Удосконалення існуючих систем відповідно до цих принципів може значно підвищити їхню ефективність в Agile-проєктах. Висновки поточного етапу представлені у таблиці 2.1.

Таблиця 2.1 – Резюме аналізу рішень систем управління

Аспект	Висновки
Обмежена процесів	Поточні системи демонструють обмеження у наданні динамічних та наочних візуалізацій процесів розробки. Необхідні поліпшення підвищення розуміння поточного стану проєкту. Динамічні дашборди, інтерактивні звіти та візуальне відображення залежностей є ключовими елементами, які можуть значно покращити візуалізацію процесів.

Продовження таблиці 2.1

Аспект	Висновки
Чи не інтегровані засоби комунікації	Існуючі системи недостатньо інтегровані із засобами комунікації, що ускладнює ефективний обмін інформацією та узгодження робіт усередині команд. Інтеграція з чатами, відеоконференціями, спільне редагування документів та керування коментарями необхідні для створення єдиного середовища комунікацій усередині проєкту
Недостатня адаптація до Agile-принципів	Системи покращення в частині адаптації до Agile-принципів, щоб підтримувати гнучкі методології та стимулювати колективну роботу. управління проєктами потребують в Гнучкість Інкрементальність, а також підтримка автономності команд є критично важливими аспектами, які потрібно вдосконалити. і мінливість, ітеративність і Зворотний зв’язок, візуалізація прогресу та підтримка частого випуску продукту також повинні бути впроваджені для ефективної відповідності Agile принципам.

В ході аналізу існуючих систем управління проєктами було виявлено ключові проблеми, що впливають на ефективність комунікацій та відповідність Agile-принципам. Обмежена візуалізація процесів, не інтегровані засоби комунікації та недостатня адаптація до Agile ставлять під питання здатність поточних рішень задовольняти потреби сучасних команд. Незважаючи на виявлені недоліки, кожна система має свої сильні сторони.

2.2 Візуалізація архітектури системи

Важливим інструментом для опису будь-яких систем є діаграми використання. Діаграми використання (Use Case Diagrams) є одним із найпоширеніших і корисних інструментів у галузі розробки програмного

забезпечення. Вони використовуються для моделювання функціональності системи та ілюстрації варіантів її використання з боку акторів.

Основний принцип діаграм використання полягає в описі того, як актори взаємодіють з системою, і як система відповідає на ці взаємодії.

Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів (Use Case Diagram) є важливим інструментом для моделювання функціональних можливостей інформаційної системи управління проєктами на основі методології Agile. Вона визначає основних акторів системи – менеджера проєкту, членів команди та клієнта – та їхню взаємодію з ключовими функціями, такими як створення спринтів, планування задач, відстеження прогресу та надання зворотного зв’язку. Ця діаграма відображає принципи Agile, зокрема прозорість і динамічне управління задачами, забезпечуючи чітке розуміння ролей і взаємодій у процесі управління проєктами. Завдяки акценту на залученні всіх сторін до ітеративного процесу, діаграма прецедентів підкреслює орієнтованість системи на підтримку Agile-підходів, таких як Scrum або Kanban, і слугує основою для подальшого визначення вимог до системи. Діаграма прецедентів представлена на рис. 2.1.



Рисунок 2.1 – Діаграма прецедентів

Діаграма класів (Class Diagram) призначена для відображення внутрішньої структури програмного забезпечення системи управління проєктами. Вона моделює основні об’єкти (класи), які формують логіку додатку, а також їх атрибути, методи та зв’язки між собою.

У даній моделі виділено три основні класи: Project, Task і User. Клас Project відповідає за зберігання загальної інформації про проєкт, його часові межі, опис і статус виконання. Клас Task містить відомості про конкретні завдання, включаючи виконавця, терміни та поточний стан. Клас User представляє користувача системи з відповідними атрибутами: ім’ям, електронною адресою та роллю.

Зв’язки між класами вказують на те, що проєкт може містити багато завдань, а кожне завдання призначається конкретному користувачеві. Така модель забезпечує чітке розуміння логіки обробки даних та дозволяє ефективно реалізувати функціональність системи відповідно до принципів об’єктно-орієнтованого програмування. Діаграма класів розроблюваного проєкту представлена на рисунку 2.2.

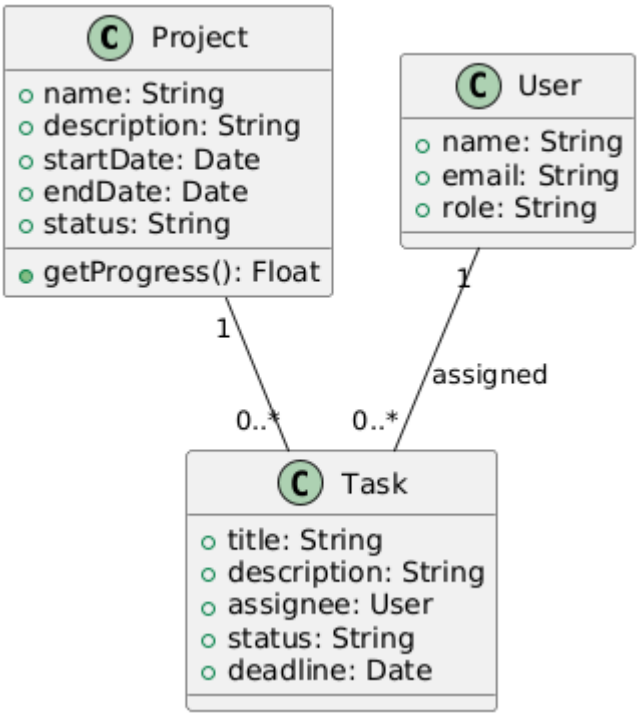


Рисунок 2.2 – Діаграма класів інформаційної систем

Діаграма активностей (Activity Diagram)

Діаграма активності ілюструє динамічну поведінку системи та послідовність дій, які виконуються користувачами в процесі роботи з нею.

У даній діаграмі відображено типовий життєвий цикл проєкту в системі управління: створення проєкту керівником, додавання завдань, розподіл обов'язків між членами команди, виконання завдань, оновлення статусу та генерація підсумкового звіту. Діаграма використовує блок-схему для демонстрації логічної послідовності дій і дозволяє чітко відстежити, які процеси відбуваються на кожному етапі.

Це особливо важливо в контексті методології Agile, де гнучкість, регулярне оновлення статусу та командна співпраця є ключовими аспектами. Діаграма активності забезпечує розробників візуальною моделлю поведінки системи, що сприяє якісному проєктуванню логіки роботи. Діаграма активностей зображена на рисунку 2.3.



Рисунок 2.3 – Діаграма активностей

Діаграма послідовностей (Sequence Diagram)

Діаграма послідовностей дозволяє відобразити сценарій взаємодії між компонентами системи в режимі реального часу. Вона є ключовим інструментом для моделювання часової послідовності обміну повідомленнями між об’єктами.

У даній діаграмі зображено типовий випадок – оновлення статусу завдання розробником. Взаємодія починається з того, що користувач через інтерфейс вибирає потрібне завдання. Запит передається на сервер, який, у свою чергу, звертається до бази даних для оновлення відповідного запису. Після завершення операції система повертає підтвердження і відображає оновлення для користувача.

Цей сценарій демонструє, як дані проходять через різні рівні архітектури – від клієнтської частини до серверної логіки й бази даних, – що є важливим для забезпечення коректної та надійної взаємодії в межах інформаційної системи. Діаграма послідовностей зображена на рисунку 2.4.

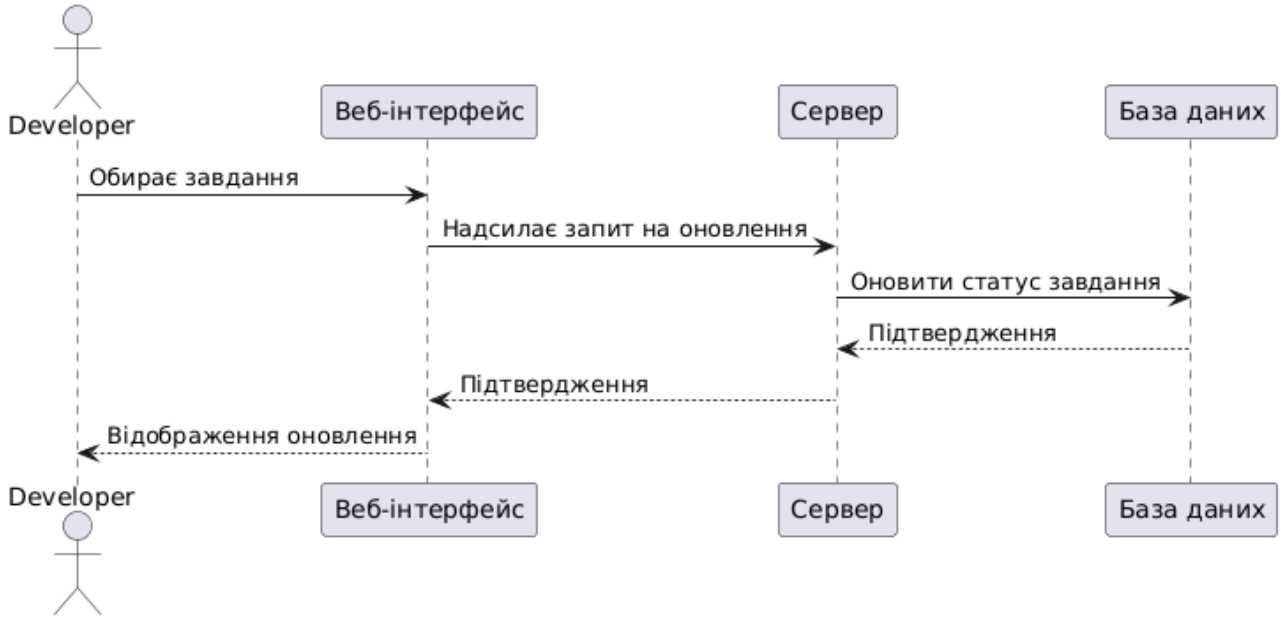


Рисунок 2.4 – Діаграма послідовностей

Діаграма компонентів (Component Diagram)

Діаграма компонентів використовується для відображення архітектурної структури системи на рівні програмних модулів. Вона показує, з яких логічних блоків складається система, а також які залежності існують між ними.

Серверна логіка виконує обробку запитів, працює з базою даних та передає дані для формування звітів.

Така структура забезпечує масштабованість системи, можливість багаторазового використання компонентів і спрощення супроводу. Діаграма компонентів є основою для реалізації мікросервісної або багаторівневої архітектури в рамках сучасних вимог до розробки інформаційних систем. Діаграма компонентів зображена на рисунку 2.5



Рисунок 2.5 – Діаграма компонентів

Діаграма розгортання (Deployment Diagram)

Діаграма розгортання демонструє фізичну структуру інформаційної системи – тобто, як її програмні компоненти розміщуються на конкретних

технічних засобах. Це дозволяє зрозуміти, як система буде функціонувати в реальному середовищі.

У даній діаграмі змодельовано три основні вузли: клієнтський пристрій (з веб-браузером), сервер застосунку (що містить backend та REST API), а також сервер бази даних (з СКБД PostgreSQL). Взаємозв'язки між вузлами ілюструють потоки даних між клієнтом, сервером і сховищем інформації.

Така модель дозволяє визначити вимоги до інфраструктури, спланувати розгортання системи в хмарному або локальному середовищі, а також забезпечити безперервність обслуговування користувачів у режимі 24/7. Діаграма розгортання зображена на рисунку 2.6.

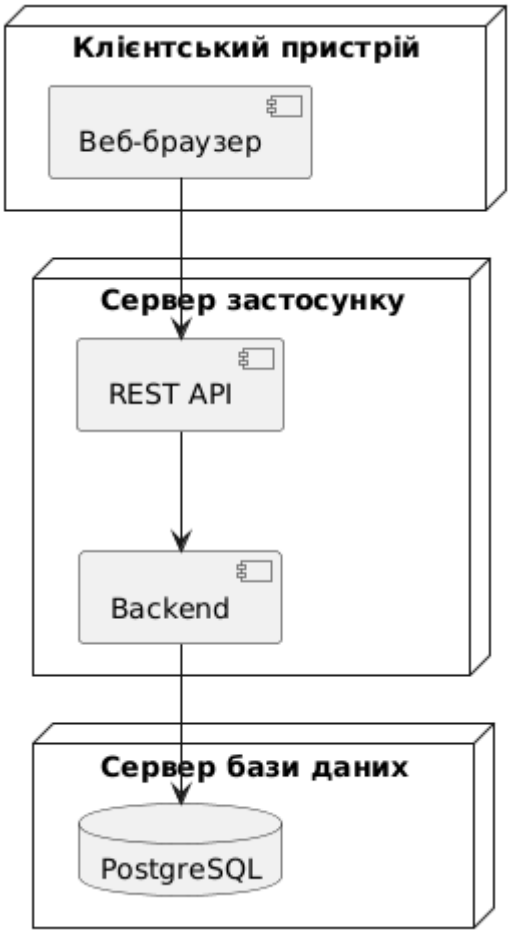


Рисунок 2.6 – Діаграма розгортання

3 ОПИС ПРОЦЕСУ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ КЕРУВАННЯ ПРОЄКТОМ

3.1 Опис фрагментів розробки

Для демонстрації процесу розробки інформаційної системи для управління проєктом із застосуванням методології Agile, доцільно привести фрагменти коду, який реалізує різні функції розроблюваної системи.

Налаштування бази даних (рис. 3.1).

```
import sqlite3

def init_db():
    conn = sqlite3.connect('project_manager.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS users (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 username TEXT NOT NULL UNIQUE,
                 password TEXT NOT NULL,
                 role TEXT NOT NULL)''')
    c.execute('''CREATE TABLE IF NOT EXISTS projects (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 name TEXT NOT NULL,
                 description TEXT)''')
    c.execute('''CREATE TABLE IF NOT EXISTS tasks (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 project_id INTEGER,
                 title TEXT,
                 status TEXT,
                 assigned_to TEXT,
                 comments TEXT,
                 FOREIGN KEY(project_id) REFERENCES projects(id))''')
    c.execute('''CREATE TABLE IF NOT EXISTS task_history (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 task_id INTEGER,
                 change_type TEXT,
                 old_value TEXT,
                 new_value TEXT,
                 change_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                 FOREIGN KEY(task_id) REFERENCES tasks(id))''')
    conn.commit()

    try:
        c.execute("INSERT INTO users (username, password, role) VALUES (?, ?,
?)", ("admin", "admin", "manager"))
        conn.commit()
    except sqlite3.IntegrityError:
        pass

        conn.close()
```

Рисунок 3.1 Лістинг налаштування бази даних

Цей фрагмент відповідає за ініціалізацію бази даних SQLite для інформаційної системи. Він створює чотири таблиці: users для зберігання даних користувачів, projects для проєктів, tasks для завдань із полем для коментарів і task_history для журналу змін завдань. Також додається початковий користувач-менеджер (admin) для доступу до системи. Цей код забезпечує основу для зберігання даних, що відповідає принципам Agile щодо прозорості та гнучкості управління.

Вікно входу до системи (рис. 3.2).

```
import tkinter as tk
from tkinter import messagebox

class LoginWindow:
    def __init__(self, master):
        self.master = master
        self.master.title("Вхід до системи")
        self.master.geometry("300x200")

        tk.Label(master, text="Логін").pack()
        self.username_entry = tk.Entry(master)
        self.username_entry.pack()

        tk.Label(master, text="Пароль").pack()
        self.password_entry = tk.Entry(master, show="*")
        self.password_entry.pack()

        tk.Button(master, text="Увійти", command=self.login).pack(pady=10)

    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()

        conn = sqlite3.connect('project_manager.db')
        c = conn.cursor()
        c.execute("SELECT * FROM users WHERE username=? AND password=?",
(username, password))
        user = c.fetchone()
        conn.close()

        if user:
            self.master.destroy()
            Dashboard(user)
        else:
                    messagebox.showerror("Помилка", "Невірний логін
або пароль")
```

Рисунок 3.2 – Вікно входу до системи

Цей фрагмент реалізує клас LoginWindow, який створює інтерфейс для автентифікації користувачів. Користувач вводить логін і пароль, які

перевіряються в базі даних. У разі успішного входу відкривається панель керування. Цей компонент забезпечує безпечний доступ до системи, що є важливим для розподілу ролей у методології Agile, де менеджери та члени команди мають різні права доступу.

Панель керування (див. рис. 3.3).

```
class Dashboard:
    def __init__(self, user):
        self.root = tk.Tk()
        self.root.title("Панель керування")
        self.root.geometry("400x300")

        self.user = user

        tk.Label(self.root, text=f"Вітаємо, {user[1]}
({user[3]})").pack(pady=10)
        tk.Button(self.root, text="Список проектів",
command=self.show_projects).pack(pady=5)
        tk.Button(self.root, text="Керування користувачами",
command=self.manage_users).pack(pady=5)
        tk.Button(self.root, text="Змінити пароль",
command=self.change_password).pack(pady=5)
        tk.Button(self.root, text="Вийти", command=self.root.quit).pack(pady=5)

        self.root.mainloop()

    def show_projects(self):
        ProjectWindow(self.user)

    def manage_users(self):
        if self.user[3] != "manager":
            messagebox.showerror("Доступ заборонено", "Лише менеджери можуть
керувати користувачами.")
            return
        UserManagementWindow()

    def change_password(self):
        new_password = simpledialog.askstring("Зміна пароля", "Введіть новий
пароль", show='*')
        if new_password:
            conn = sqlite3.connect('project_manager.db')
            c = conn.cursor()
            c.execute("UPDATE users SET password=? WHERE id=?", (new_password,
self.user[0]))
            conn.commit()
            conn.close()
                    messagebox.showinfo("Успіх", "Пароль успішно
змінено!")
```

Рисунок 3.3 – Панель керування

Клас Dashboard створює основний інтерфейс користувача після входу, відображаючи привітання та надаючи доступ до ключових функцій: перегляду проєктів, керування користувачами (доступно лише менеджерам), зміни пароля та виходу з системи. Цей компонент забезпечує централізований доступ до

функціоналу, що відповідає принципам Agile щодо прозорості та зручності взаємодії для всіх ролей.

Керування користувачами (рис. 3.4).

```
from tkinter import simpledialog

class UserManagementWindow:
    def __init__(self):
        self.window = tk.Toplevel()
        self.window.title("Користувачі")
        self.window.geometry("300x400")

        self.user_list = tk.Listbox(self.window)
        self.user_list.pack(fill=tk.BOTH, expand=True)

        tk.Button(self.window, text="Додати користувача",
command=self.add_user).pack(pady=5)
        tk.Button(self.window, text="Видалити користувача",
command=self.delete_user).pack(pady=5)
        self.load_users()

    def load_users(self):
        self.user_list.delete(0, tk.END)
        self.users = []
        conn = sqlite3.connect('project_manager.db')
        c = conn.cursor()
        c.execute("SELECT id, username, role FROM users")
        self.users = c.fetchall()
        for u in self.users:
            self.user_list.insert(tk.END, f"{u[1]} ({u[2]})")
        conn.close()

    def add_user(self):
        username = simpledialog.askstring("Користувач", "Введіть логін")
        password = simpledialog.askstring("Пароль", "Введіть пароль", show='*')
        role = simpledialog.askstring("Роль", "Введіть роль (manager/user)")
        if username and password and role:
            conn = sqlite3.connect('project_manager.db')
            c = conn.cursor()
            try:
                c.execute("INSERT INTO users (username, password, role) VALUES
(?, ?, ?)", (username, password, role))
                conn.commit()
```

Рисунок 3.4 – Реалізація керування користувачами

Клас UserManagementWindow реалізує інтерфейс для додавання та видалення користувачів, доступний лише для користувачів із роллю manager. Цей фрагмент дозволяє створювати нових членів команди та видаляти їх, що відповідає Agile-підходу до гнучкого управління командою та розподілу ролей у проєктах.

Керування проєктами (рис. 3.5).

```
class ProjectWindow:
    def __init__(self, user):
        self.user = user
        self.window = tk.Toplevel()
        self.window.title("Проєкти")
        self.window.geometry("400x400")

        self.project_list = tk.Listbox(self.window)
        self.project_list.pack(fill=tk.BOTH, expand=True)
        self.project_list.bind("<<ListboxSelect>>", self.show_project_details)

        tk.Button(self.window, text="Додати проєкт",
command=self.add_project).pack(pady=5)
        tk.Button(self.window, text="Звіт по проєкту",
command=self.generate_report).pack(pady=5)
        self.load_projects()

    def load_projects(self):
        self.project_list.delete(0, tk.END)
        conn = sqlite3.connect('project_manager.db')
        c = conn.cursor()
        c.execute("SELECT id, name FROM projects")
        self.projects = c.fetchall()
        for row in self.projects:
            self.project_list.insert(tk.END, row[1])
        conn.close()

    def add_project(self):
        top = tk.Toplevel()
        top.title("Новий проєкт")

        tk.Label(top, text="Назва проєкту").pack()
        name_entry = tk.Entry(top)
        name_entry.pack()

        tk.Label(top, text="Опис").pack()
        desc_entry = tk.Entry(top)
        desc_entry.pack()
```

Рисунок 3.5 – Реалізація керування проєктами

Клас ProjectWindow відповідає за відображення списку проєктів, додавання нових проєктів і генерацію звітів по проєктах. Користувач може переглядати деталі проєкту та переходити до завдань. Функція звітування показує статистику завдань за статусами, що підтримує принцип прозорості Agile. Цей фрагмент забезпечує управління проєктами в ітеративному стилі.

Керування завданнями (рис. 3.6).

Клас TaskWindow реалізує функціонал для додавання, редагування, фільтрації та перегляду історії змін завдань. Він підтримує коментарі, журналювання змін статусу та відповідального, а також обмеження доступу, де користувачі з роллю user можуть редагувати лише свої завдання. Цей фрагмент забезпечує ітеративне управління задачами, що відповідає принципам Agile, зокрема Kanban-подібному підходу та прозорості.

```
class TaskWindow:
    def __init__(self, project_id, user):
        self.project_id = project_id
        self.user = user

        self.window = tk.Toplevel()
        self.window.title("Завдання")
        self.window.geometry("500x500")

        filter_frame = tk.Frame(self.window)
        filter_frame.pack(pady=5)

        tk.Label(filter_frame, text="Фільтр за статусом:").grid(row=0, column=0)
        self.status_filter = tk.Entry(filter_frame)
        self.status_filter.grid(row=0, column=1)

        tk.Label(filter_frame, text="Фільтр за виконавцем:").grid(row=0,
column=2)
        self.assigned_filter = tk.Entry(filter_frame)
        self.assigned_filter.grid(row=0, column=3)

        tk.Button(filter_frame, text="Застосувати",
command=self.load_tasks).grid(row=0, column=4, padx=5)

        self.task_list = tk.Listbox(self.window)
        self.task_list.pack(fill=tk.BOTH, expand=True)
        self.task_list.bind("<<ListboxSelect>>", self.edit_task)

        tk.Button(self.window, text="Додати завдання",
command=self.add_task).pack(pady=5)
        tk.Button(self.window, text="Переглянути історію змін",
command=self.view_history).pack(pady=5)
        self.load_tasks()
```

Рисунок 3.6 – Керування завданнями

3.2 Демонстрація результатів роботи інформаційної системи

Для того, щоб переглянути роботу інформаційної системи, потрібно продемонструвати роботу кожного її елементу починаючи від самого початку і до самого кінця.

Спочатку, потрібно здійснити вхід до системи. Після запуску застосунку, користувач побачить вікно входу в систему, яке має вигляд (рис. 3.7).

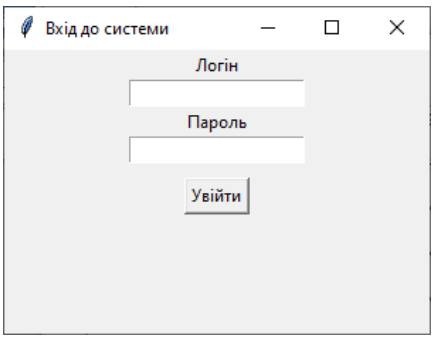


Рисунок 3.7 – Вікно входу до системи

Здійснивши вхід, користувач перейде до панелі керування (рис. 3.8).

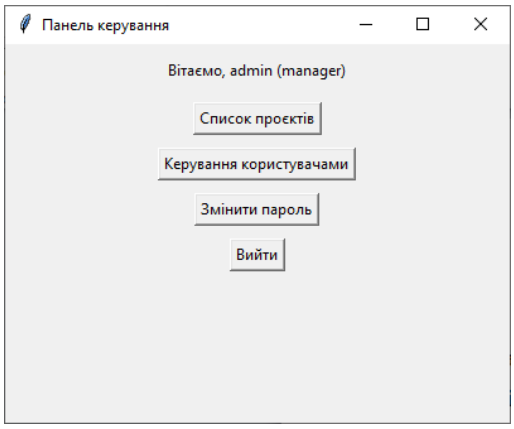


Рисунок 3.8 – Панель керування проєктами

На панелі керування, додамо трьох користувачів (рис. 3.9).

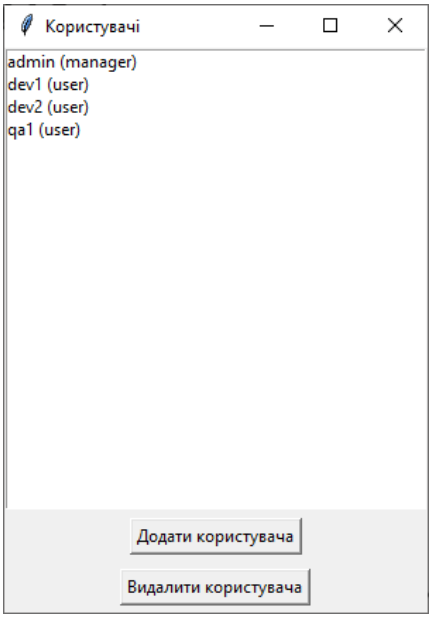


Рисунок 3.9 – Додавання нових користувачів

Як результат, в базі даних створено користувачів dev1 (розробник), dev2 (розробник), і qa1 (тестувальник) із роллю user.

Наступним кроком є створення проєкту. Для цього, потрібно на панелі керування натиснути «Список проєктів» і у вікні «Проєкти» натиснути кнопку «Додати проєкт». В результаті, користувач отримає наступне (рис. 3.10).

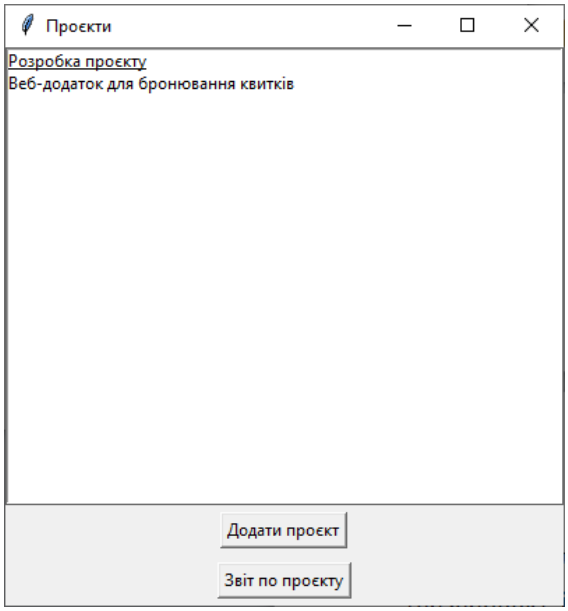


Рисунок 3.10 – Додавання проєкту «Веб-додаток для бронювання квитків»

Далі, потрібно обрати створений проєкт «Веб-додаток для бронювання квитків» і натиснути «Перейти до завдань», (рис. 3.11).

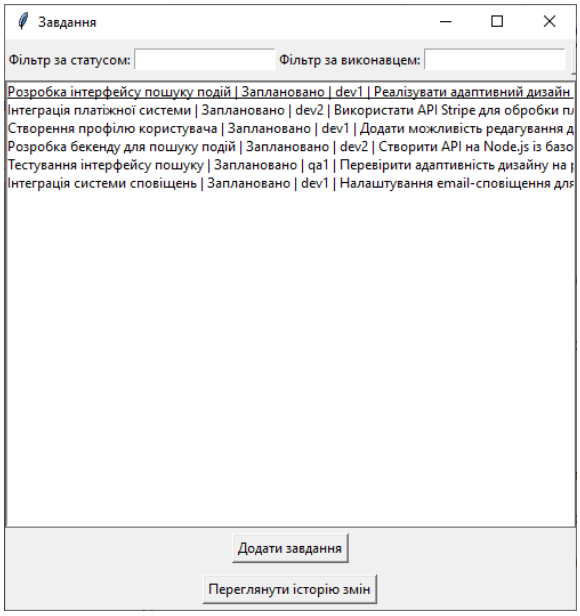


Рисунок 3.11 – Додані завдання

Далі проведемо редагування завдань і створення історії змін. Для цього виконаємо вхід як розробник dev1 та відкриємо довільне завдання і відредагуємо його. Після цього у вікні «Завдання» відбудуться наступні зміни (рис. 3.12).

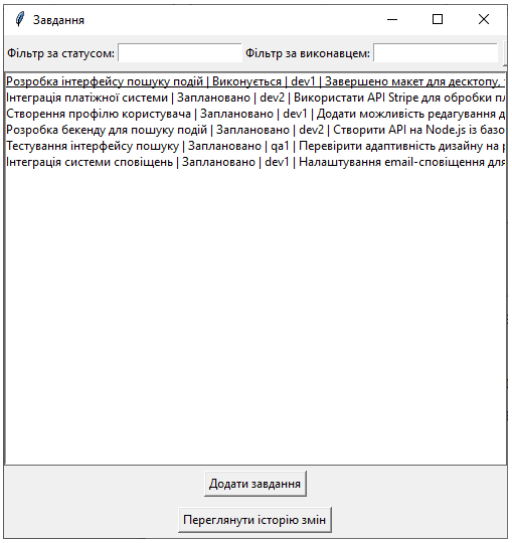


Рисунок 3.12 – Редагування завдання користувачем dev1

Тепер проведемо зміну відповідального за виконання завдання, для цього потрібно зайти як admin та внести відповідні зміни (рис. 3.13).

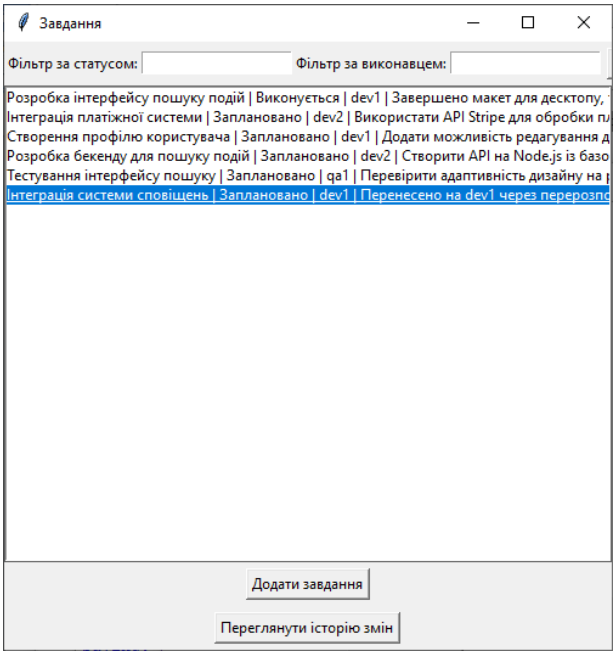


Рисунок 3.13 – Відображення змін у завданнях

Розглянемо перегляд історії змін. Для цього потрібно обрати завдання «Розробка інтерфейсу пошуку подій» і натиснути «Переглянути історію змін». Отримаємо наступний результат (рис. 3.14).

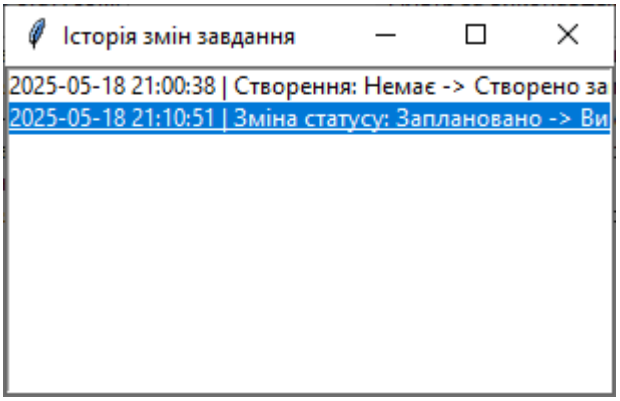


Рисунок 3.14 – Історія змін в завданні

Генерація звіту. Потріно натиснути «Звіт по проєкту». В результаті відкриється вікно із статистикою (рис. 3.15).

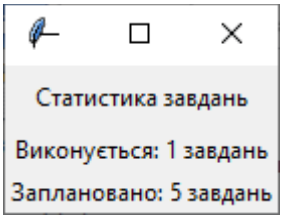


Рисунок 3.15 – Статистика завдань

Наведемо приклади, як виглядають дані в базі даних. Таблиця завдання (рис. 3.16).

id	project_id	title	status	assigned_to	comments
1	1	Розробка інтерфейсу пошуку подій	Виконується	dev1	Завершено макет для десктопу, тестується мобільна...
2	1	Інтеграція платіжної системи	Заплановано	dev1	Перенесено на dev1 через перерозподіл завдань.
3	1	Створення профілю користувача	Заплановано	dev1	Додати можливість редагування даних користувача...
4	1	Розробка бекенду для пошуку	Заплановано	dev2	Створити API на Node.js із базою даних PostgreSQL...
5	1	Тестування інтерфейсу	Заплановано	qa1	Перевірити адаптивність дизайну

Рисунок 3.16 – Дані бази даних «Завдання»

Таблиця «Історія завдань» (рис. 3.17).

id	task_id	change_type	old_value	new_value	change_time
1	1	Створення	Немає	Створено завдання: Розробка інтерфейсу...	2025-05-18 14:46:00
2	1	Зміна статусу	Заплановано	Виконується	2025-05-18 14:50:00
3	2	Створення	Немає	Створено завдання: Інтеграція платіжної...	2025-05-18 14:47:00
4	2	Зміна відповідального	dev2	dev1	2025-05-18 14:51:00

Рисунок 3.17 – Дані бази даних «Історія завдань»

Наведений приклад демонструє принципи Agile:

– ітеративність: Завдання розбиті на невеликі частини (наприклад, розробка інтерфейсу, бекенду, тестування), що відповідає спринтам у Scrum або потоку в Kanban.

– прозорість: Історія змін і звіти забезпечують видимість прогресу для команди та менеджера.

– гнучкість: Коментарі та можливість змінювати відповідальних дозволяють адаптувати завдання до змін у вимогах.

– ролі та доступ: Менеджери керують проєктами та користувачами, а користувачі працюють лише зі своїми завданнями, що відповідає розподілу ролей в Agile.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно досягнуто поставленої мети – розроблено прототип інформаційної системи управління проєктами на основі принципів методології Agile.

На етапі аналізу принципів Agile встановлено, що гнучкість, ітеративність, прозорість та орієнтація на командну взаємодію є ключовими факторами ефективного управління проєктами. Проведена класифікація програмних засобів показала, що сучасні системи, такі як Jira, Trello та Asana і ін., що пропонують широкий спектр інструментів, однак потребують адаптації до специфічних потреб користувачів, що стало основою для розробки власного прототипу. Визначено функціональні вимоги (управління беклогом, дошкою задач, звітування) та нефункціональні вимоги (зручність інтерфейсу, швидкодія, масштабованість), які лягли в основу проєктування системи.

Архітектура системи була спроєктована з використанням UML-діаграм, що дозволило чітко структурувати компоненти та їх взаємодію, забезпечивши модульність і можливість подальшого розширення. Реалізований прототип системи включає базові функції: створення та управління проєктами, завданнями, коментарями, а також журналювання змін і формування звітів. Використання Python, бібліотеки Tkinter та бази даних SQLite забезпечило простоту реалізації та доступність системи для користувачів.

Тестування прототипу на реальному проєкті продемонструвало його працездатність і відповідність принципам Agile, зокрема щодо прозорості процесів та зручності управління задачами. Оцінка ефективності системи підтвердила її здатність оптимізувати процеси планування та виконання проєктів у невеликих командах, хоча для масштабних проєктів необхідне подальше вдосконалення, зокрема розширення функціоналу звітності та інтеграція з іншими інструментами.

Розроблена система має потенціал для практичного застосування в умовах гнучкого управління проєктами. Подальший розвиток може включати

впровадження веб-інтерфейсу, підтримку розподілених команд та автоматизацію аналітики проєктів, що сприятиме підвищенню її конкурентоспроможності на ринку програмних засобів для управління проєктами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alex Campbell. Agile and Scrum : Complete Guide. What is Agile and What is Scrum? Alex Published, 2020. 300 p.

2. Alex Campbell. Agile project management with Scrum : how to become a great project manager? Alex Published, 2022. 149 p.

3. Alex Cowan. Hypothesis-Driven Development. Cooke & McDouglas, 2023. 438 p.

4. Andrew Sammons. Agile project management with Scrum + Kanban 2 In 1 : The Last 2 Approaches You'll Need To Become More Productive And Meet Your Project Goals. M & M Limitless Online Inc., 2021. 322 p.

5. David Kadavy. Mind management, Not Time Management : Productivity When Creativity Matters. Kadavy, Inc., 2020. 249 p.

6. Dombrowski M., Sachenko A, Sachenko O., Dombrowski Z. Proactive Project Management as a Discrete Event System. *2020 IEEE European Technology and Engineering Management Summit (E-TEMS)*. Dortmund, Germany, 2020. Pp. 1–4. Doi : 10.1109/E-TEMS46250.2020.9111777.

7. Dr. Festus Elleh. Project Management Waterfall-Agile-It-Data Science. Authorhouse, 2023. 402 p.

8. Dr. Sam Swapn Sinha. Best PMP Exam Prep Guide 2023–2024 : Get PMP Certified in 2 weeks- study 2 hours a day before-after work. Strategism, 2023. 202 p.

9. Gustavo Razzetti. Remote Not Distant : Design a Company Culture That Will Help You Thrive in a Hybrid Workplace. Liberationist Press, 2022. 328 p.

10. Hassan Osman. Hybrid work management : how to manage a hybrid team in the new workplace? (A super-short book about how to analyze, plan, manage, and evaluate your team's hybrid work arrangement). 2021. 64 p.

11. James Shore, Shane Warden. The art of agile development. Ascent Audio, 2021. 537 p.

12. James Stockwell. Agile and Scrum : Streamlining Project Management for Efficient Team Collaboration. Anthony Pierce, 2023. 123 p.

13. Mayank Mittal. The art of building great products : Combine your intuition with the best proven methodologies to build digital products everyone will love. Mayank Mittal, 2022. 350 p.

14. Блага Н. В. Управління проєктами : навч. посібник. Львів : Львівський державний університет внутрішніх справ, 2021. 152 с.

15. Большая О., Іщенко І. Інноваційний розвиток проєктного менеджменту в Україні. *Via Economica*. 2023. С. 21–26.

16. Борисов О. В., Данченко О. Б. Технологія вибору ефективної методології управління ІТ-проєктом. *Вісник Національного технічного університету «ХПІ»*. 2022. С. 7–13.

17. Бушуєв С. Д. Гібридизація методологій управління інфраструктурними проєктами та програмами. *Вісник Одеського національного морського університету*. 2020. С. 187–207.

18. Вакалюк Т. А. Організаційні структури в ІТ компаніях. *Тези ІІ Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення»*. 2019. С. 148–149.

19. Василевська А. Управління проєктами підприємства із використанням інформаційних технологій. *Вісник Київського національного торговельно-економічного університету*. 2012. № 1. С. 99–105.

20. Вербицький Н. І. Життєвий цикл ІТ-проєктів із штучним інтелектом. *Міжнародна наукова інтернет-конференція Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення*. Вип. 63. Тернопіль. 2021. С. 17–18.

21. Гавловська Н. І., Рудніченко Є. М. Управління проєктами : навч. посібник. Хмельницький : ХНУ, 2022. 258 с.

22. Головач О. В., Шевченко Г. І. Agile-технології в управлінні інноваційними проєктами : особливості та переваги. *Економічний простір*. 2019. №144. С. 119–128.

23. Горбаченко С. А., Чепурна О. Є. Адаптація проєктного підходу до управління стартапами. *Трансформаційна економіка*. 2023. С. 24–28.

24. Гулага Я. С., Мнушка О. В. Критерії оцінки якості в проєктах, що використовують AGILE. *Збірник наукових праць за матеріалами міжнародної науково-практичної конференції «Комп'ютерні технології і мехатроніка»*. 2019. С. 82–85.

25. Гусєв Є. Agile та Scrum у управлінні проєктами : практичні рекомендації. Харків : Фактор. 2021.

26. Дьяченко А. В. Сучасні підходи до організації управління проєктами в умовах поширення Agile-технологій. *Вісник Київського національного торговельно-економічного університету*. 2021. №2. С. 59–67.

27. Зачко О. Б., Івануса А. І., Кобилкін Д. С. Управління проєктами : теорія, практика, інформаційні технології. Львів : ЛДУ БЖД, 2019. 173 с.

28. Іванов О. О. Використання Scrum-методології в управлінні проєктами. *Вісник Національного університету "Львівська політехніка": Серія "Менеджмент та підприємництво в Україні: етапи становлення та проблеми розвитку"*. 2021. №974. С. 142–150.

29. Кіндрат І. Управління проєктами на основі Agile : методології, інструменти та практичні кейси. Київ : Академія. 2021

ДОДАТОК А

Програмний код

```
import sqlite3
import tkinter as tk
from tkinter import messagebox, simpledialog

def init_db():
    conn = sqlite3.connect('project_manager.db')
    c = conn.cursor()

    # Створення таблиці users
    c.execute('''CREATE TABLE IF NOT EXISTS users (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 username TEXT NOT NULL UNIQUE,
                 password TEXT NOT NULL,
                 role TEXT NOT NULL)''')

    # Створення таблиці projects
    c.execute('''CREATE TABLE IF NOT EXISTS projects (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 name TEXT NOT NULL,
                 description TEXT)''')

    # Створення таблиці tasks
    c.execute('''CREATE TABLE IF NOT EXISTS tasks (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 project_id INTEGER,
```

```
                title TEXT,
                status TEXT,
                assigned_to TEXT,
                FOREIGN KEY(project_id) REFERENCES
projects(id))’’’)

        # Створення таблиці task_history
        c.execute(‘‘‘CREATE TABLE IF NOT EXISTS task_history (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                task_id INTEGER,
                change_type TEXT,
                old_value TEXT,
                new_value TEXT,
                change_time TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
                FOREIGN KEY(task_id) REFERENCES tasks(id))’’’)

        conn.commit()

        # Додавання початкового користувача
        try:
            c.execute("INSERT INTO users (username, password, role)
VALUES (?, ?, ?)", ("admin", "admin", "manager"))
            conn.commit()
        except sqlite3.IntegrityError:
            pass

        conn.close()

    class LoginWindow:
```

```
        def __init__(self, master):
            self.master = master
            self.master.title("Вхід до системи")
            self.master.geometry("300x200")

            tk.Label(master, text="Логін").pack()
            self.username_entry = tk.Entry(master)
            self.username_entry.pack()

            tk.Label(master, text="Пароль").pack()
            self.password_entry = tk.Entry(master, show="*")
            self.password_entry.pack()

            tk.Button(master, text="Увійти",
command=self.login).pack(pady=10)

        def login(self):
            username = self.username_entry.get()
            password = self.password_entry.get()

            conn = sqlite3.connect('project_manager.db')
            c = conn.cursor()
            c.execute("SELECT * FROM users WHERE username=? AND
password=?", (username, password))
            user = c.fetchone()
            conn.close()

            if user:
                self.master.destroy()
                Dashboard(user)
```

```
            else:
                messagebox.showerror("Помилка", "Невірний логін або пароль")

        class Dashboard:
            def __init__(self, user):
                self.root = tk.Tk()
                self.root.title("Панель керування")
                self.root.geometry("400x300")

                self.user = user

                tk.Label(self.root, text=f"Вітаємо, {user[1]} ({user[3]})").pack(pady=10)
                tk.Button(self.root, text="Список проєктів", command=self.show_projects).pack(pady=5)
                tk.Button(self.root, text="Керування користувачами", command=self.manage_users).pack(pady=5)
                tk.Button(self.root, text="Змінити пароль", command=self.change_password).pack(pady=5)
                tk.Button(self.root, text="Вийти", command=self.root.quit).pack(pady=5)

                self.root.mainloop()

            def show_projects(self):
                ProjectWindow(self.user)

            def manage_users(self):
                if self.user[3] != "manager":
```