

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Проект MVP проактивного управління командою аутсорс-компанії»

Студента 2 курсу, 4М групи,
спеціальності 121 «Інженерія
програмного забезпечення»
освітньої програми
«Управління проектами
програмних продуктів»

підпис студента

Руденка Вадима
Юрійовича

Науковий керівник
PhD,
доцент кафедри інженерії
програмного забезпечення та
кібербезпеки

підпис керівника

Десятко Альона
Миколаївна

Гарант освітньої програми
PhD,
доцент кафедри інженерії
програмного забезпечення та
кібербезпеки

підпис гаранта

Десятко Альона
Миколаївна

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення та кібербезпеки

Освітній ступінь магістр

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Управління проектами програмних продуктів»

Затверджую

Зав. кафедри інженерії програмного
забезпечення та кібербезпеки

Криворучко О. В.

«13» грудня 2023 р.

Завдання

на кваліфікаційну роботу студентів

Руденкові Вадиму Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Проект MVP проактивного управління командою аутсорс-компанії»

Затверджена наказом ректора від «27» листопада 2023 р. № 4149

2. Строк здачі студентом закінченої роботи 15 листопада 2024

3. Цільова установка та вихідні дані до роботи

Мета роботи: розробити концепцію та функціональний прототип MVP проактивного управління командою аутсорс-компанії.

Об'єкт дослідження: процеси управління командою аутсорс-компанії.

Предмет дослідження: розробка MVP проактивного управління командою аутсорс-компанії.

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ КОМАНДАМИ В АУТСОРС-КОМПАНІЯХ

1.1. ТРАДИЦІЙНІ МЕТОДИ УПРАВЛІННЯ КОМАНДАМИ

1.2. ПРОБЛЕМИ УПРАВЛІННЯ КОМАНДАМИ В АУТСОРС-КОМПАНІЯХ

1.3. ПОТРЕБА В ПРОАКТИВНОМУ УПРАВЛІННІ КОМАНДОЮ

РОЗДІЛ 2 КОНЦЕПЦІЯ MVP ПРОАКТИВНОГО УПРАВЛІННЯ КОМАНДОЮ АУТСОРС-КОМПАНІЇ

2.1 ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНОСТІ MVP

2.2 ТЕХНОЛОГІЧНИЙ СТЕК

2.3 АРХІТЕКТУРА MVP

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ MVP

3.1 РОЗРОБКА MVP

3.2 ТЕСТУВАННЯ MVP

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	<i>Вибір теми випускної кваліфікаційної роботи</i>	07.11.2023	07.11.2023
2.	<i>Розробка та затвердження завдання на роботу магістра (стац/заоч)</i>	13.12.2023	13.12.2023
3.	<i>Вступ та перелік літературних джерел</i>	22.02.2024	22.02.2024
4.	<i>Розробка технічного завдання</i>	14.03.2024	14.03.2024
5.	<i>Розділ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ КОМАНДАМИ В АУТСОРС-КОМПАНІЯХ</i>	10.04.2024	10.04.2024
6.	<i>Розділ 2. КОНЦЕПЦІЯ MVP ПРОАКТИВНОГО УПРАВЛІННЯ КОМАНДОЮ АУТСОРС-КОМПАНІЇ</i>	23.05.2024	23.05.2024
7.	<i>Розділ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ MVP</i>	05.09.2024	05.09.2024
8.	<i>Розробка програми та методики тестування</i>	27.09.2024	27.09.2024
9.	<i>Написання наукової статті</i>	16.04.2024	16.04.2024
10.	<i>Керівництво користувача</i>	11.10.2024	11.10.2024
11.	<i>Висновки та пропозиції</i>	16.10.2024	16.10.2024
12.	<i>Здача випускної кваліфікаційної роботи на кафедрі (перша перевірка)</i>	18.10.2024	18.10.2024
13.	<i>Підготовка реферату та презентації доповіді</i>	28.10.2024	28.10.2024
14.	<i>Попередній захист випускної кваліфікаційної роботи</i>	29.10.2024 – 31.10.2024	29.10.2024 – 31.10.2024
15.	<i>Здача зброшурованої випускної кваліфікаційної роботи</i>	15.11.2024	15.11.2024
16.	<i>Зовнішнє рецензування випускної кваліфікаційної роботи</i>	28.10.2024	28.10.2024
17.	<i>Підготовка до публічного захисту випускної кваліфікаційної роботи</i>	02.12.2024- 03.12.2024	02.12.2024- 03.12.2024

7. Дата видачі завдання _____ «13» грудня 2023 р.

8. Науковий керівник кваліфікаційної роботи _____
Десятко А.М.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми _____ Десятко А.М.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент _____ Руденко В. Ю.

(прізвище, ініціали, підпис)

АНОТАЦІЯ

В умовах сучасного бізнесу, де високі темпи розвитку та конкуренція стають нормою, аутсорсинг стає важливим інструментом для підвищення ефективності та гнучкості компаній. Однак управління командами в таких проєктах стикається з численними викликами через відстань, часові пояси, культурні та мовні бар'єри. Це дослідження спрямоване на розробку концепції та функціонального прототипу проактивного управління командою аутсорс-компанії. В роботі розглядаються ключові методи та інструменти для моніторингу та попередження ризиків, ефективного планування та покращення комунікації в розподілених командах.

Відповідно до мети дослідження робота присвячена розробці MVP проактивного управління командою аутсорс-компанії. Кваліфікаційна робота на тему «Проактивне управління командою аутсорс-компанії» містить 68 сторінок, 16 рисунків та 6 додатків. Перелік використаних джерел налічує 31 найменування.

Було проведено аналіз сучасних підходів до управління командами, визначено ключові фактори проактивного управління та розроблено функціональний прототип.

В результаті проведеного дослідження було створено систему проактивного управління, яка дозволяє підвищити ефективність командної роботи та мінімізувати ризики в аутсорс-проєктах.

Ключові слова: аутсорсинг, проактивне управління, розподілені команди, комунікація, управління ризиками, автоматизація, командна робота, ефективність, управління проєктами.

ABSTRACT

In the modern business environment, characterized by rapid development and intense competition, outsourcing has become a vital tool for enhancing the efficiency and flexibility of companies. However, managing teams in such projects faces numerous challenges due to distance, time zones, cultural, and language barriers. This study focuses on developing a concept and functional prototype for proactive management of an outsourcing company's team. The research explores key methods and tools for monitoring and preventing risks, effective planning, and improving communication in distributed teams.

In accordance with the research objective, the thesis is dedicated to the development of an MVP for proactive management of an outsourcing company's team. The qualification work on the topic "Proactive Management of an Outsourcing Company's Team" contains 68 pages, 16 figures, and 6 appendices. The list of references includes 31 sources.

An analysis of modern approaches to team management has been conducted, key factors of proactive management have been identified, and a functional prototype has been developed.

As a result of the research, a proactive management system was created, which allows for increased team efficiency and minimized risks in outsourcing projects.

Keywords: outsourcing, proactive management, distributed teams, communication, risk management, automation, teamwork, efficiency, project management.

Зміст

ВСТУП	9
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ КОМАНДАМИ В АУТСОРС-КОМПАНІЯХ	11
1.1. Традиційні методи управління командами.....	11
1.2. Проблеми управління командами в аутсорс-компаніях:	16
1.3. Потреба в проактивному управлінні командою	18
РОЗДІЛ 2 КОНЦЕПЦІЯ MVP ПРОАКТИВНОГО УПРАВЛІННЯ КОМАНДОЮ АУТСОРС-КОМПАНІЇ	23
2.1 Визначення функціональності MVP	23
2.2 Технологічний стек	27
2.3 Архітектура MVP	32
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ MVP	40
3.1 Розробка MVP	40
3.2 Тестування MVP	57
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63
ДОДАТКИ	66

ВСТУП

Сучасний світ бізнесу характеризується високою динамікою та конкурентним середовищем, що вимагає від компаній гнучкості та ефективності у своїй діяльності. Аутсорсинг, як стратегічний інструмент досягнення цих цілей, набуває все більшої популярності, особливо в сфері програмної інженерії. Однак, успіх аутсорсингового проекту напряду залежить від ефективності управління командами, що працюють над його реалізацією.

Традиційні методи управління командами, що були ефективними в умовах локальних команд, не завжди успішно справляються з викликами, що постають в аутсорсингових проектах. Відстань, різні часові пояси, мовні бар'єри та культурні відмінності – це лише деякі фактори, що ускладнюють комунікацію та координацію в розподілених командах. Як результат, виникають проблеми з ефективністю, якістю виконання завдань, затримками у поставках та зниженням задоволеності клієнтів.

Очевидно, що традиційні методи управління потребують переосмислення та доповнення проактивними підходами. Проактивне управління командою в аутсорс-компаніях передбачає постійне моніторинг і прогнозування можливих проблем та ризиків, своєчасне вжиття заходів для їх запобігання, а також активне випередження потреб та очікувань клієнтів. Це досягається шляхом:

Ефективної комунікації: розробки чітких комунікаційних каналів, використання інструментів для відеоконференцій та онлайн-співпраці, регулярного зворотнього зв'язку та моніторингу настрою в команді.

Прозорого планування: детального розкладу завдань, визначення відповідальності та чіткого моніторингу виконання поставлених цілей.

Використання сучасних технологій: інструментів автоматизації задач, моніторингу прогресу та оцінки ефективності роботи.

Фокусу на командній роботі: створення атмосфери взаємодопомоги та співпраці, стимулювання обміну знаннями та досвідом.

Важливо розуміти, що проактивне управління командою не тільки покращує ефективність виконання проектів, але й безпосередньо впливає на якість кінцевого продукту та задоволеність клієнтів. Збільшення продуктивності, мінімізація помилок, скорочення затрат на виправлення дефектів, дотримання термінів – все це позитивно впливає на усіх учасників процесу.

Вивчення та впровадження проактивних підходів до управління командами в аутсорс-компаніях стає надзвичайно актуальним завданням. Це дозволить підвищити ефективність бізнесу, збільшити конкурентоспроможність та забезпечити успішне виконання проектів в складних умовах сучасного ринку

Об'єкт дослідження – процеси управління командою аутсорс-компанії.

Предмет дослідження – розробка MVP проактивного управління командою аутсорс-компанії.

Мета роботи – розробити концепцію та функціональний прототип MVP проактивного управління командою аутсорс-компанії.

Відповідно до мети було поставлено такі завдання дослідження:

- провести аналіз сучасних методів управління командами в аутсорс-компанія;
- визначити ключові фактори проактивного управління командою;
- розробити концепцію MVP проактивного управління командою;
- створити функціональний прототип MVP;
- провести тестування та оцінку розробленого прототипу.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ КОМАНДАМИ В АУТСОРС-КОМПАНІЯХ

1.1. Традиційні методи управління командами

Водоспадний метод, також відомий як каскадний метод, є класичною моделлю управління проектами, яка базується на лінійному послідовному виконанні етапів. Кожен етап розробки проекту, від початкового аналізу до фінального впровадження, завершується повністю, перш ніж розпочинається наступний.

Ця модель отримала свою назву завдяки візуальному уявленню, де кожен етап представлений у вигляді «водоспаду», що перетікає з одного рівня в інший. Такий підхід забезпечує чітку структуру та чітке розмежування відповідальності між різними групами учасників проекту.



Рис 1.1 Модель водоспаду та Agile підхід

Ключові переваги водоспадного методу полягають у його простоті та легкості розуміння. Він забезпечує чіткий план дій, який легко контролювати та відстежувати. Також, завдяки чіткому розмежуванню етапів, водоспадний метод зменшує ризики перекриття функцій та непередбачуваних змін в процесі розробки.

Однак, традиційний водоспадний метод має ряд суттєвих недоліків. Його основним обмеженням є відсутність гнучкості та адаптації до змін. Оскільки кожен етап завершується повністю, внесення змін на пізніх етапах проекту може виявитися надзвичайно складним та затратним.

Також, водоспадний метод не передбачає швидкого отримання результатів та раннього виявлення помилок. Він зазвичай призводить до затримок в проєкті, оскільки виявлені недоліки на пізніх етапах вимагають значних зусиль для виправлення.

В сучасному світі, що характеризується високою динамікою та швидкими змінами, водоспадний метод все менш ефективний. Він не може швидко адаптуватися до змінних умов і не дозволяє збирати зворотний зв'язк від клієнтів на ранніх етапах проекту.

В результаті, водоспадний метод зазвичай застосовується для проєктів з чітко визначеними вимогами і низьким рівнем ризику, де зміни мало ймовірні, а вимоги до гнучкості не є критичними.

Agile-методи - це сучасний підхід до управління проєктами, що заснований на принципах гнучкості та адаптації до змін. На відміну від традиційних, лінійних моделей, таких як водоспадний метод, Agile-методи акцентують увагу на ітеративній та інкрементній розробці, де проєкт поділяється на невеликі, самодостатні цикли (спринти), що дозволяють швидко отримувати результат та вносити корективи на ранніх етапах.

Ключовим принципом Agile-методів є безперервне спілкування та взаємодія між командою розробки та замовником. Це дозволяє збирати зворотний зв'язк від клієнтів, впроваджувати зміни та коригувати напрямок розробки проєкту в реальному часі.

Agile-методи заохочують командну роботу та спільну відповідальність за результат. Команда працює в тісному співробітництві, збираючись на регулярних зустрічах, обговорюючи прогрес проєкту та вживаючи заходи для вирішення виникаючих проблем.

Однією з головних переваг Agile-методів є їхня гнучкість. Вони дозволяють легко вносити зміни в проєкт на будь-якому етапі розробки, що є особливо важливим в умовах швидких змін та невизначеності. Agile-методи також забезпечують швидке отримання результатів, що дозволяє збирати

зворотний зв'язк від клієнта на ранніх етапах і впроваджувати необхідні корективи для забезпечення успіху проекту.

Незважаючи на всі переваги, Agile-методи також мають деякі обмеження. Їх ефективність залежить від високого рівня компетентності та мотивації членів команди, а також від здатності до ефективної комунікації та взаємодії. Крім того, Agile-методи не завжди підходять для проектів з чітко визначеними вимогами та високим рівнем формалізації.

Незважаючи на ці обмеження, Agile-методи поступово стають все більш популярними в різних галузях діяльності, особливо в IT-індустрії. Їхня гнучкість, швидкість і фокус на задоволенні клієнтів роблять їх ефективним інструментом для управління проектами в сучасному світі.

PRINCE2 (Projects IN Controlled Environments) - це структурований метод управління проектами, що надає чіткий рамковий підхід до планування, контролю та звітності в проекті. Методика PRINCE2 орієнтована на управління проектами з високим рівнем складності та залученням різних зацікавлених сторін.

Метод PRINCE2 визначає чіткі ролі і відповідальність учасників проекту, зокрема, керівника проекту, керівника проектної офісу та команди проекту. Також він передбачає використання різних інструментів і технік для планування і контролю проекту, таких як:

- План проекту: визначає цілі, scope, час і бюджет проекту.
- Схема проекту: візуалізує етапи проекту і їх взаємозв'язки.
- Ризик-менеджмент: визначає потенційні ризики і вживає заходи для їх мінімізації.
- Контроль змін: забезпечує контроль за змінами в проекті і затвердження нових вимог.

Метод PRINCE2 застосовується на всіх етапах проекту, від початку до завершення. Він передбачає постійний моніторинг і контроль за виконанням проекту, що дозволяє вчасне виявити недоліки і вносити необхідні корективи.

Метод PRINCE2 застосовується у різних галузях діяльності, особливо в державних і комерційних організаціях. Він дозволяє звести до мінімуму ризику, забезпечити контроль над ресурсами і завершити проект в час і в межах бюджету.

Однак, PRINCE2 може бути складним у використанні для малих проектів і може не дозволити достатньо гнучко реагувати на зміни в проекті.

Незважаючи на це, PRINCE2 залишається популярним методом управління проектами, особливо для складних проектів з багатьма зацікавленими сторонами, які вимагають чіткого планування та контролю.

Метод критичного шляху (CPM) - це техніка управління проектами, що використовується для визначення найдовшого шляху виконання завдань, який впливає на загальну тривалість проекту. Цей шлях називається критичним шляхом і визначає мінімальний час, необхідний для завершення проекту.

CPM базується на створенні мережевої діаграми, яка відображає послідовність етапів проекту та їхні залежності. Кожен етап має певну тривалість, а між ними існують взаємозв'язки, що визначають порядок виконання. Метод CPM використовує аналіз мережі, щоб визначити найдовший шлях через мережу, який неможливо скоротити без впливу на загальну тривалість проекту.

Оскільки будь-яке запізнення на етапах критичного шляху призводить до запізнення всього проекту, CPM допомагає сфокусувати зусилля на оптимізації саме цих етапів. Це дозволяє ефективно використовувати ресурси та мінімізувати ризики затримок, ключові елементи системи показано на рисунку 1.2.

визначення завдань

- ідентифікація та опис всіх етапів проекту

визначення тривалості завдань

- оцінка часу, необхідного для виконання кожного етапу

визначення залежностей

- встановлення взаємозв'язків між етапами проекту (наприклад, один етап не може розпочатися, доки не завершиться інший)

створення мережевої діаграми

- візуалізація всіх етапів і їхніх залежностей у вигляді графічного представлення

визначення критичного шляху

- визнання найдовшого шляху через мережу, який визначає мінімальну тривалість проекту

аналіз критичного шляху

- оптимізація етапів на критичному шляху для скорочення загальної тривалості проекту

Рис. 1.2 Ключові елементи СРМ системи

Незважаючи на свою ефективність, СРМ має декілька обмежень. Він не враховує непередбачувані фактори та зміни в проекті, також його складно застосовувати для великих проектів з багатьма залежностями.

Незважаючи на це, СРМ залишається важливим інструментом управління проектами, що допомагає оптимізувати тривалість проекту та забезпечити його успішне завершення в час.

Окрім широко відомих методів, таких як водоспадний, Agile та PRINCE2, існує ряд інших традиційних підходів, що забезпечують структуру та інструменти для ефективного управління проектами. Ці методи часто доповнюють один одного, надаючи додаткові інструменти для планування, контролю та звітності.

Gantt chart – це графічне відображення плану проекту, що дозволяє візуалізувати етапи проекту, їх тривалість та взаємозв'язки. Горизонтальна вісь Gantt chart відображає час, тоді як вертикальна вісь представляє завдання проекту. Кожне завдання відображається у вигляді прямокутника, довжина якого відповідає тривалості завдання. Gantt chart дозволяє чітко визначити

терміни завершення кожного етапу проекту, а також визначити критичний шлях проекту і контролювати за виконанням завдань.

PERT-метод (Program Evaluation and Review Technique) - це метод аналізу мережі, що використовується для планування та контролю завдань у проекті. PERT-метод передбачає створення мережевої діаграми, що відображає всі завдання проекту і їхні взаємозв'язки. Кожному завданню присвоюється три оцінки тривалості: оптимістичну, песимістичну і найімовірнішу. PERT-метод дозволяє розрахувати очікувану тривалість кожного завдання і загальну тривалість проекту, а також визначити критичний шлях проекту і прогнозувати ймовірність завершення проекту в час.

Six Sigma - це система управління якістю, що фокусується на мінімізації помилок і недоліків в процесах проекту. Six Sigma використовує статистичні методи для визначення причин помилок і впровадження заходів для їх усунення. Six Sigma допомагає підвищити якість продукції і послуг, скоротити витрати і збільшити задоволеність клієнтів.

Ці методи надають додаткові інструменти для ефективного управління проектами, допомагаючи управлінцям проектів більш точно планувати, контролювати і звітувати про прогрес проекту, а також підвищити якість результатів.

Важливо зазначити, що вибір методу управління проектами залежить від характеристик проекту, таких як його розмір, складність, рівень ризику, і вимоги замовника. У деяких випадках може бути ефективним комбінувати різні методи, щоб максимізувати переваги кожного з них.

1.2. Проблеми управління командами в аутсорс-компаніях:

Глобалізація та розвиток інформаційних технологій призвели до широкого розповсюдження аутсорсингу, особливо в ІТ-індустрії. Аутсорсинг дозволяє компаніям скоротити витрати, отримати доступ до кваліфікованих спеціалістів та покращити гнучкість бізнесу. Однак, управління командами в

аутсорс-компаніях зустрічає ряд унікальних викликів, що вимагають спеціальних підходів та інструментів.

Одним з найбільших викликів є географічна розподіленість команд. Відстань, різні часові пояси та відсутність прямої комунікації можуть створити перешкоди для ефективної співпраці та координації роботи. Це може призвести до непорозумінь, затримок у виконанні завдань та зниження продуктивності.

Культурні різниці також можуть створювати проблеми в управлінні розподіленими командами. Різні способи мислення, стилі комунікації та культурні норми можуть призвести до конфліктів і непорозумінь. Важливо знати і враховувати ці різниці для ефективної комунікації і взаємодії в команді.

Мовні бар'єри також можуть стати серйозною перешкодою для успішного управління розподіленими командами. Непорозуміння у перекладі може призвести до помилок у виконанні завдань та зниження якості продукції.

Окрім цих зовнішніх факторів, існують також проблеми, що пов'язані з організацією і менеджментом роботи в аутсорс-компаніях. Відсутність безпосереднього нагляду і контролю над командою може призвести до зниження мотивації та дисципліни, а також ускладнити визначення та вирішення проблем.

Недостатня прозорість в процесах та комунікації також може стати серйозною перешкодою для ефективного управління командою. Не всі члени команди можуть мати доступ до необхідної інформації або зрозуміти цілі та завдання проекту, що може призвести до непорозумінь і зниження продуктивності.

Усі ці проблеми мають значний вплив на ефективність роботи розподілених команд і можуть призвести до зниження якості продукції, затримок у проектах та незадоволеності клієнтів.

Тому управління розподіленими командами в аутсорс-компаніях вимагає спеціальних підходів та інструментів для переборювання цих викликів. Важливо розробити чіткі процеси комунікації, використовувати

сучасні інструменти для співпраці та контролю, а також створювати атмосферу довіри і взаєморозуміння в команді.

1.3. Потреба в проактивному управлінні командою

Традиційні методи управління командами, такі як водоспадний метод та PRINCE2, хоч і є структурованими та ефективними в певних контекстах, не завжди здатні адекватно реагувати на виклики сучасного світу бізнесу. Швидкі зміни в технологіях, зростаюча конкуренція та потреба в гнучкості створюють новий контекст, де традиційні підходи можуть бути недостатньо ефективними.

Одним з ключових обмежень традиційних методів є їх лінійний характер. Вони передбачають послідовне виконання етапів проекту, що ускладнює внесення змін на пізніх етапах. У світі, де вимоги клієнтів можуть змінюватися в процесі розробки, такий підхід може призвести до затримок, перевитрат та зниження задоволеності клієнтів.

Традиційні методи також характеризуються відсутністю гнучкості і адаптивності до змінних умов проекту. Вони часто орієнтовані на чітко визначені плани і терміни, що не дозволяє швидко реагувати на непередбачувані обставини та впроваджувати необхідні корективи.

Ще одним важливим обмеженням є недостатня увага до комунікації і співпраці в команді. Традиційні методи часто фокусуються на індивідуальному виконанні завдань, що може призвести до непорозумінь, конфліктів і зниження ефективності.

Крім того, традиційні методи не завжди враховують особливості розподілених команд, які працюють над проектом в різних географічних локальних і часових поясах. Відстань, культурні різниці та мовні бар'єри можуть створювати значні перешкоди для ефективної комунікації і співпраці.

Нарешті, традиційні методи не завжди забезпечують достатнє заохочення іновацій і креативності в команді. Вони часто орієнтовані на виконання завдань за заздалегідь визначеними правилами, що може призвести до зниження мотивації та креативності членів команди.

Таким чином, традиційні методи управління командами, хоч і зберігають певну актуальність в деяких контекстах, не здатні повністю відповідати вимогам сучасного світу бізнесу. Вони потребують переосмислення та доповнення новими підходами, що враховують динаміку змін, значення гнучкості, співпраці і інновацій.

Проактивне управління командами – це сучасний підхід, що переосмислює традиційний спосіб управління проектами. Він переносить фокус від реакції на проблеми до їх передбачення і профілактики. Цей підхід має значні переваги як для команд, так і для замовників проектів.



Рис. 1.3 Схема процесу проактивного управління командою

Найважливішою перевагою проактивного управління є збільшення гнучкості та адаптивності команди. Воно дозволяє швидко реагувати на зміни в умовах проекту, включаючи зміни в вимогах клієнта, виявлення непередбачених обставин чи нові вимоги до продукту. Така гнучкість є ключовою в сучасному світі, де швидкі зміни стали нормою.

Проактивне управління також допомагає зменшити ризики затримок і недоліків в проекті. Завдяки постійному моніторингу і аналізу проекту, команда може визначити потенційні проблеми ще на ранніх етапах і вжити

заходів для їх усунення. Це значно зменшує ймовірність виникнення кризових ситуацій і допомагає зберегти час і ресурси.

Крім того, проактивне управління сприяє підвищенню ефективності і продуктивності команди. Завдяки постійному фокусу на завданнях та вирішенню проблем, члени команди можуть працювати більш ефективно, зменшуючи час на виправлення помилок і переробку роботи.

Важливим елементом проактивного управління є покращення комунікації в команді. Це досягається через регулярні зустрічі, відкритий обмін інформацією, а також через використання сучасних інструментів для співпраці і зв'язку. Ефективна комунікація допомагає зменшити ризики непорозумінь, покращити взаємодію між членами команди і забезпечити спільне розуміння цілей проекту.

Проактивне управління також сприяє підвищенню мотивації і задоволеності членів команди. Завдяки постійному зв'язку, відкритій комунікації та можливості впливати на процес проекту, члени команди відчують себе більш залученими і мотивованими до успішної реалізації проекту.

Нарешті, проактивне управління допомагає створити культуру постійного вдосконалення і інновацій. Завдяки постійному аналізу і оцінці процесів та результатів, команда може визначити слабкі місця і впроваджувати нові підходи і технології для покращення ефективності і якості роботи.

В цілому, проактивне управління командами є перспективним підходом для забезпечення успішної реалізації проектів в сучасному динамічному світі. Він допомагає підвищити гнучкість, ефективність, мотивацію та інноваційність команди, що є ключовими факторами успіху в конкурентному середовищі.

Проактивне управління командами – це не просто новий підхід, а цілий набір інструментів і принципів, які дозволяють змінити парадигму управління проектами. Для успішної реалізації цього підходу необхідно враховувати ключові аспекти, що забезпечують його ефективність.



Рис. 1.4 Етапи проактивного управління

Ефективна комунікація є основою проактивного управління. Це означає створення чітких каналів зв'язку між членами команди і замовником проекту. Необхідно використовувати різні інструменти для спілкування, включаючи відеоконференції, онлайн-платформи для співпраці та регулярні зустрічі. Важливо також створювати атмосферу відкритої комунікації, де члени команди можуть вільно висловлювати свої думки і запитувати з'ясування неясностей. Регулярний зворотний зв'язк від клієнта та відстеження настрою в команді також є необхідними елементами ефективної комунікації.

Прозоре планування передбачає розробку детального розкладу завдань з чітким визначенням відповідальності за їх виконання. Це дозволяє усім членам команди зрозуміти свої обов'язки і внести свій внесок в успішну реалізацію проекту. Важливим елементом прозорого планування є регулярний моніторинг виконання завдань і відстеження прогресу проекту. Це допомагає вчасне визначити відхилення від плану і вжити необхідні заходи для їх виправлення.

Використання сучасних технологій є необхідним для успішної реалізації проактивного управління. Сучасні інструменти автоматизації задач, моніторингу прогресу та оцінки ефективності роботи дозволяють звільнити

час команди для творчого розв'язання проблем і концентрації на ключових завданнях.

Фокус на командній роботі є ключовим аспектом проактивного управління. Створення атмосфери взаємодопомоги та співпраці допомагає зменшити ризики конфліктів і покращити ефективність команди. Стимулювання обміну знаннями та досвідом між членами команди також є важливим елементом успішної командної роботи.

Ці ключові аспекти проактивного управління не є окремими елементами, а скоріше взаємопов'язаними частинами цілісної системи. Їх спільне використання дозволяє створити ефективну і гнучку команду, здатну успішно реалізувати складні проекти в сучасному динамічному світі.

РОЗДІЛ 2 КОНЦЕПЦІЯ MVP ПРОАКТИВНОГО УПРАВЛІННЯ КОМАНДОЮ АУТСОРС-КОМПАНІЇ

2.1 Визначення функціональності MVP

Проект MVP проактивного управління командою в аутсорс-компаніях передбачає розробку комплексної системи, що складається з різних модулів, які забезпечують повний цикл управління проектами та командами. Кожен модуль виконує специфічні функції, що допомагають збирати і аналізувати дані, планувати і контролювати проект, зменшувати ризики та підвищувати ефективність роботи.

Модуль управління проектами є центральним елементом системи. Він надає інструменти для створення нових проектів, присвоєння завдань членам команди, встановлення термінів та пріоритетів, а також ведення дорожньої карти проекту. Цей модуль також дозволяє відстежувати прогрес виконання завдань, створювати звіти про хід проекту і виявляти потенційні проблеми. Важливим елементом цього модулю є функція управління ризиками, що дозволяє ідентифікувати потенційні загрози для проекту і розробити стратегії для їх мінімізації.

Модуль управління командою фокусується на організації та менеджменті команди проекту. Він дозволяє створювати профілі членів команди з їхньою контактною інформацією, навичками та досвідом. Цей модуль також надає інструменти для встановлення ролей і прав доступу для кожного члена команди, а також для моніторингу їхньої активності в проектах. Функція спілкування в цьому модулі допомагає зменшити перешкоди, пов'язані з розподіленими командами, забезпечуючи зручний спосіб спілкування між членами команди через чати і форуми.

Модуль аналізу та звітності використовується для збору та аналізу даних про ефективність роботи команди. Він надає інструменти для створення звітів про продуктивність, виконання завдань, затримки та ризики. Завдяки аналізу цих даних можна визначити проблемні місця в роботі команди, виявити потенційні ризики та вжити заходів для їх мінімізації.

Модуль моніторингу та прогнозування є ключовим елементом проактивного управління. Він передбачає створення системи раннього виявлення проблемних місць і ризиків. Цей модуль також дозволяє відстежувати настрій та мотивацію в команді і надавати інструменти для проактивного втручання та вирішення проблем. Завдяки цій функції команда може вживати заходів для виправлення ситуації ще до того, як вона стане кризовою.

Кожен з цих модулів виконує важливу роль в цілісній системі проактивного управління командами. Їх спільна робота дозволяє збирати і аналізувати інформацію, планувати і контролювати проекти, зменшувати ризики та підвищувати ефективність роботи розподілених команд.

Модуль управління проектами є центральним елементом системи проактивного управління командою, що надає інструменти для ефективного планування, контролю та координації роботи над проектами. Він забезпечує централізований пункт доступу до інформації про проект, дозволяючи керівникам проектів та членам команди отримувати актуальні дані про його хід.

Основною функцією модуля управління проектами є створення нових проектів. Цей процес включає введення основних даних про проект, таких як назва, опис, цілі, терміни і бюджет. Модуль також дозволяє створити структуру проекту, розділивши його на етапи і завдання, і призначити відповідальних за їх виконання.

Модуль також надає інструменти для управління завданнями: встановлення пріоритетів, призначення термінів, додавання описів і деталей. Керівники проектів можуть використовувати цьому модулі для відстеження прогресу виконання завдань, визначення затримок та виявлення потенційних проблем.

Одним із ключових елементів модуля управління проектами є моніторинг прогресу проекту. Він надає інформацію про виконання завдань, використання ресурсів і терміни завершення етапів. Завдяки цьому модулю

керівники проектів можуть вчасне визначити відхилення від плану і вжити заходів для їх виправлення.

Модуль управління проектами також дозволяє створювати звіти про хід проекту. Ці звіти можуть містити інформацію про виконання завдань, використання ресурсів, затримки та ризику. Звіти можуть бути використані для інформування замовника про прогрес проекту, а також для аналізу ефективності роботи команди і вжиття необхідних заходів для підвищення продуктивності.

Важливим елементом модуля управління проектами є функція управління ризиками. Вона допомагає ідентифікувати потенційні загрози для проекту і розробити стратегії для їх мінімізації. Модуль надає інструменти для створення списку ризиків, оцінки їхньої ймовірності і наслідків, а також для розробки планів заходів для зменшення ризиків.

В цілому, модуль управління проектами є необхідним інструментом для успішного управління проектами в аутсорс-компаніях. Він надає інструменти для ефективного планування, контролю та координації роботи, допомагаючи зменшити ризики затримок та недоліків в проектах.

Модуль управління командою – це важливий елемент системи проактивного управління, який зосереджує увагу на організації та ефективному менеджменті розподілених команд проектів. Він надає інструменти для створення профілів членів команди, встановлення ролей та прав доступу, а також забезпечує платформу для зручного спілкування і співпраці.

Модуль починається зі створення профілів кожного члена команди. У профілі збирається ключова інформація: контактні дані, навичок, досвід, а також інформація про їхню роль в проекті. Ця інформація дозволяє керівникам проектів оцінити компетентність кожного члена команди і ефективно призначити завдання.

Важливою функцією модуля є встановлення ролей і прав доступу. Ця функція дозволяє керувати доступом до інформації та функціональними

можливостями системи. Наприклад, керівники проектів мають повний доступ до всіх даних і функцій, тоді як члени команди можуть мати доступ лише до інформації, що відноситься до їх завдань.

Модуль управління командою також надає інструменти для моніторингу активності членів команди. Він відстежує виконання завдань, час, що присвячений роботі над проектом, а також взаємодію з іншими членами команди. Ця інформація дозволяє оцінити продуктивність кожного члена команди і визначити потенційні проблеми.

Одним із ключових елементів цього модулю є функція спілкування. Вона допомагає зменшити перешкоди, пов'язані з розподіленими командами, забезпечуючи зручний спосіб спілкування між членами команди через чати і форуми. Модуль також може інтегруватися з іншими системами для обміну інформацією, наприклад, з платформами для командної роботи як Slack або Trello.

Модуль управління командою також дозволяє створювати звіти про продуктивність та настроїв в команді. Ці звіти містять інформацію про виконання завдань, час, що присвячений роботі, а також про рівень мотивації і задоволеності членів команди. Ця інформація допомагає керівникам проектів і менеджерам команд оцінити ефективність роботи команди і вжити заходів для її покращення.

В цілому, модуль управління командою є важливим інструментом для успішного управління розподіленими командами в аутсорс-компаніях. Він допомагає забезпечити ефективну взаємодію між членами команди, покращити комунікацію і співпрацю, а також оптимізувати процес управління проектами.

Модуль аналізу та звітності є ключовим елементом системи проактивного управління командою, що дозволяє збирати, аналізувати і візуалізувати дані про ефективність роботи команди і хід проектів. Цей модуль надає інструменти для виявлення проблемних місць та потенційних ризиків, а також для прогнозування майбутньої ефективності роботи команди.

Модуль збирає дані з різних джерел, включаючи модулі управління проектами та управління командою. Він відстежує виконання завдань, використання ресурсів, терміни завершення етапів, а також інформацію про активність членів команди, їхню продуктивність і настрої.

Цю інформацію модуль аналізує з використанням різних методів і інструментів, включаючи статистичні методи, візуалізацію даних і алгоритми машинного навчання. Це дозволяє визначити тренди в роботі команди, ідентифікувати фактори, що впливають на її ефективність, а також виявити потенційні проблеми ще до того, як вони виникнуть.

На основі проведеного аналізу модуль створює різні звіти, що допомагають керівникам проектів і менеджерам команд оцінити ефективність роботи команди і вжити необхідних заходів. Звіти можуть містити інформацію про продуктивність команди, виконання завдань, затримки, ризики, а також про настрої і мотивацію членів команди.

Модуль також дозволяє використовувати інформацію для прогнозування майбутньої ефективності роботи команди. На основі аналізу історичних даних модуль може передбачити ймовірність виконання завдань в час, ризики затримок і потенційні проблеми. Це дозволяє керівникам проектів вживати заходів для мінімізації ризиків і забезпечення успішного виконання проектів.

Модуль аналізу і звітності є важливим інструментом проактивного управління командами, що дозволяє перейти від реактивного підходу до проблем до передбачення і профілактики. Завдяки збору, аналізу і візуалізації даних можна виявити проблемні місця і вжити необхідних заходів для покращення ефективності роботи команди і забезпечення успішної реалізації проектів.

2.2 Технологічний стек

C# – це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft і що входить до екосистеми .NET. Вона відома своєю

гнучкістю, продуктивністю та підтримкою різноманітних платформ. С# широко використовується для розробки різних типів додатків, включаючи веб-додатки, настільні додатки, мобільні додатки і ігри.

Однією з ключових переваг С# є її зручний і зрозумілий синтаксис. Вона є мовою високого рівня, що зменшує кількість коду, необхідного для реалізації певних функцій, і збільшує швидкість розробки.

```
// Use a switch statement to do the math
switch (Console.ReadLine())
{
    case "a":
        Console.WriteLine($"Your result: {num1} + {num2} = " + (num1 + num2));
        break;
    case "s":
        Console.WriteLine($"Your result: {num1} - {num2} = " + (num1 - num2));
        break;
    case "m":
        Console.WriteLine($"Your result: {num1} * {num2} = " + (num1 * num2));
        break;
    case "d":
        // Ask the user to enter a non-zero divisor until they do so
        while (num2 == 0)
        {
            Console.WriteLine("Enter a non-zero divisor: ");
            num2 = Convert.ToInt32(Console.ReadLine());
        }
        Console.WriteLine($"Your result: {num1} / {num2} = " + (num1 / num2));
        break;
}

// Wait for the user to respond before closing
Console.WriteLine("Press any key to close the Calculator console app...");
Console.ReadKey();
```

Рис 2.1 Синтаксис с#

С# також відома своєю високою продуктивністю. Вона компілюється в машинний код, що дозволяє досягти високої швидкості виконання додатків. Це особливо важливо для розробки ігор і інших додатків, де продуктивність є ключовим фактором.

С# має широке співтовариство розробників, що надає багату бібліотеку компонентів і інструментів для розробки. Це зменшує час розробки і дозволяє концентрувати зусилля на реалізації специфічних функцій додатків.

С# підтримується .NET Framework і .NET Core, що надає гнучкість у розгортанні та розробці. .NET Framework є досить старою екосистемою, що підтримує Windows, тоді як .NET Core - це новіша екосистема, що підтримує різні платформи, включаючи Windows, Linux і macOS.

С# також має вбудовану підтримку об'єктно-орієнтованого програмування, що дозволяє розробляти складні додатки з використанням

класів, об'єктів і інтерфейсів. Це допомагає створювати модульний і легко обслуговуваний код.

В цілому, C# є могутньою і гнучкою мовою програмування, що відповідає вимогам розробки різноманітних додатків. Її зручний синтаксис, висока продуктивність, широке співтовариство і підтримка .NET Framework і .NET Core роблять її вільним вибором для розробки сучасних і ефективних додатків.



Рис 2.2 Синтаксис PostgreSQL

PostgreSQL - це потужна і надійна реляційна база даних з відкритим кодом, що широко використовується для розробки різноманітних додатків. Вона відома своєю стабільністю, гнучкістю і підтримкою SQL-стандартів, що робить її відмінним вибором для розробки складних і критично важливих додатків.

Однією з ключових переваг PostgreSQL є її відкритий код. Це означає, що її джерельний код доступний для всіх, що дозволяє розробникам внести зміни і доповнення до бази даних відповідно до їх потреб. Також це сприяє створенню великого і активного співтовариства розробників, що надає підтримку, документацію і нові функції для бази даних.

PostgreSQL відповідає SQL-стандартам, що збільшує її сумісність з іншими системами і програмами. Це дозволяє легко інтегрувати її з іншими додатками і використовувати стандартні SQL-запити для доступу до даних.

PostgreSQL надає широкий набір функцій і можливостей для розробки складних додатків. Вона підтримує різні типи даних, включаючи числа, текст, дату, час і бінарні дані. Також PostgreSQL надає можливість створювати складні запити з використанням з'єднань та субагрегатів, а також підтримує транзакції, що забезпечує цілісність даних в базі даних.

PostgreSQL відома своєю надійністю і стабільністю. Вона має систему відновлення після збою, що дозволяє відновити дані в разі поломки сервера. Також PostgreSQL має вбудовані функції безпеки, що допомагають захистити дані від несанкціонованого доступу.

PostgreSQL є гнучкою і масштабованою системою. Її можна використовувати як для маленьких додатків, так і для великих систем з великим обсягом даних. PostgreSQL також можна розгорнути на різних платформах, включаючи Windows, Linux і macOS.

PostgreSQL є відмінним вибором для розробки різноманітних додатків, що вимагають надійної, стабільної і гнучкої бази даних. Її відкритий код, підтримка SQL-стандартів, широкий набір функцій і можливостей, а також велике співтовариство розробників роблять її популярним вибором як для невеликих проектів, так і для великих корпоративних систем.

Інструменти розробки є необхідними для успішної реалізації будь-якого програмного проекту. Вони надають розробникам різні функції і можливості, які допомагають створювати якісне і ефективне програмне забезпечення. Вибір правильних інструментів розробки є ключовим фактором успіху проекту, оскільки вони впливають на продуктивність, якість коду і швидкість розробки.

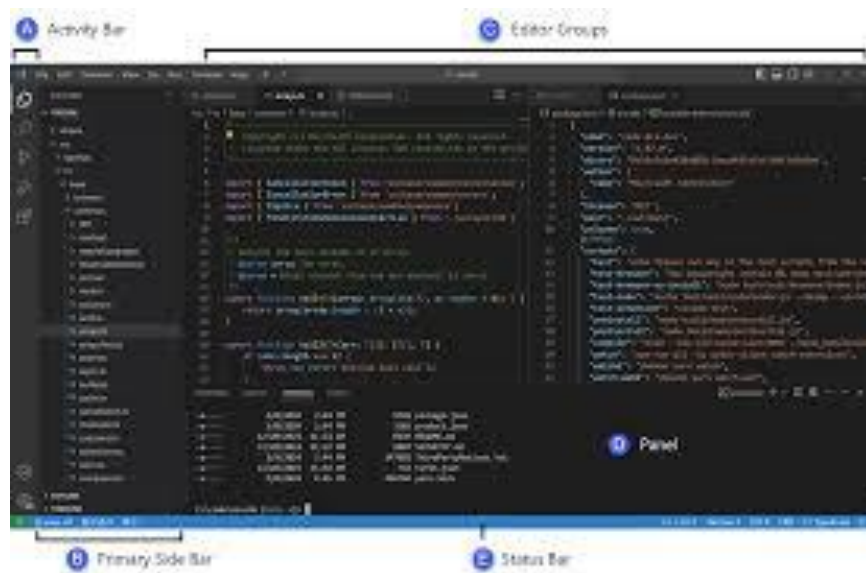


Рис 2.3 Інтерфейс Visual Studio

Visual Studio - це інтегроване середовище розробки (IDE), розроблене компанією Microsoft для розробки різних типів додатків, включаючи веб-додатки, настільні додатки, мобільні додатки і ігри. Visual Studio надає багатий набір функцій, які допомагають розробникам створювати, тестувати і розгортати програмне забезпечення. Він підтримує різні мови програмування, включаючи C#, Visual Basic, C++, Python і JavaScript.

Entity Framework Core - це об'єктно-реляційний мапер (ORM), що дозволяє розробникам працювати з базою даних за допомогою об'єктно-орієнтованих методів. ORM забезпечує абстракцію над базою даних, що зменшує кількість коду, необхідного для взаємодії з базою даних, і дозволяє розробникам зосередити свої зусилля на бізнес-логіці додатків.

NUnit - це фреймворк для тестування програмного забезпечення, що дозволяє створювати і виконувати автоматизовані тести для перевірки коректності і ефективності програмного забезпечення. Автоматизовані тести є ключовим елементом розробки якісного програмного забезпечення, оскільки вони допомагають виявити помилки на ранніх етапах розробки.

Docker - це платформа для контейнеризації додатків, що дозволяє створювати незалежні контейнери для додатків і їхніх залежностей. Це забезпечує незалежність від середовища розгортання і допомагає забезпечити стабільність і повторюваність розгортання додатків.

Git - це система контролю версій, що дозволяє відстежувати зміни в коді та забезпечувати спільну роботу в команді. Git дозволяє створювати гілки розробки, об'єднувати зміни, відстежувати історію змін і відновлювати попередні версії коду.

Вибір інструментів розробки залежить від специфіки проекту, мови програмування та вимог розробників. Важливо вибирати інструменти, що відповідають потребам проекту і допомагають створювати якісне і ефективне програмне забезпечення. Ефективне використання інструментів розробки є ключовим фактором успішної реалізації програмних проектів.

2.3 Архітектура MVP

Вибір архітектурного шаблону є ключовим етапом розробки будь-якого програмного проекту, оскільки він визначає структуру і організацію коду, впливаючи на його гнучкість, масштабованість, тестованість і підтримуваність. Для проекту MVP проактивного управління командою в аутсорс-компаніях потрібно зробити правильний вибір архітектурного шаблону, враховуючи специфічні вимоги та характеристики проекту.

Шаблон Model-View-Controller (MVC) є популярним архітектурним шаблоном для розробки веб-додатків. Він розділяє функціональність додатку на три основні частини: модель (Model), представлення (View) і контролер (Controller).

Модель представляє дані і бізнес-логіку додатку. Вона відповідає за зберігання, обробку і відновлення даних.

Представлення відображає дані користувачеві через інтерфейс додатку. Вона відповідає за візуалізацію даних і забезпечення зручності користування.

Контролер обробляє запити користувача, взаємодіє з моделлю для отримання даних і передає їх представленню для відображення.

MVC є зручним шаблоном для розробки веб-додатків з чітким розподілом функціональності між різними частинами додатку. Це дозволяє збільшити гнучкість і тестованість коду, а також спрощує роботу в команді.

Шаблон Clean Architecture - це більш складний і гнучкий архітектурний шаблон, що забезпечує краще розмежування відповідальності між шарами додатку. Він передбачає розділення додатку на декілька шарів: шари представлення, доменний шари і шари доступу до даних.

Шари представлення відображають дані користувачеві через інтерфейс додатку.

Доменний шари містить бізнес-логіку додатку і не залежить від шару представлення і шару доступу до даних.

Шари доступу до даних відповідає за взаємодію з базою даних.

Clean Architecture дозволяє збільшити тестованість коду, оскільки кожен частину додатку можна тестувати окремо. Також цей шаблон допомагає збільшити гнучкість і підтримуваність коду, оскільки зміни в одному шарі не впливають на інші шари.

Шаблон Microservices - це архітектурний шаблон, що передбачає розбиття додатку на незалежні служби, які можуть розгортатися і розвиватися окремо. Кожна служба виконує певну функцію і може бути розроблена і підтримувана окремою командою.

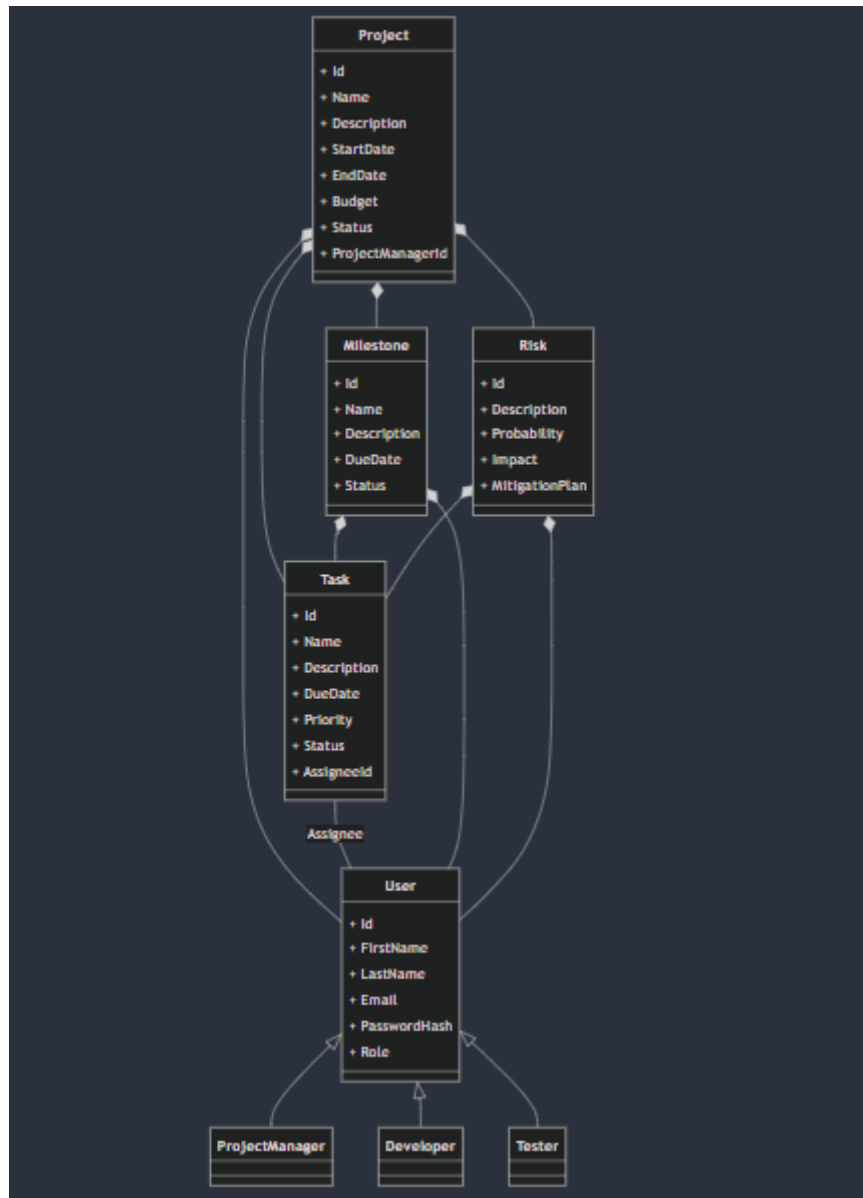


Рис 2.4 Контексна модель (діаграма класів)

Microservices дозволяють збільшити гнучкість і масштабованість додатків, оскільки кожна служба може бути розроблена і розгорнута незалежно від інших служб. Також цей шаблон допомагає зменшити ризики затримок в розробці, оскільки одна служба може бути розроблена і розгорнута без впливу на інші служби.

Вибір архітектурних принципів є важливим етапом розробки будь-якого програмного проекту. Ці принципи визначають основні правила і рекомендації, які керують розробкою архітектури системи. Вони допомагають створити зручний для розширення і підтримки код, зменшити ризики помилок і забезпечити довготривалу стабільність проекту.

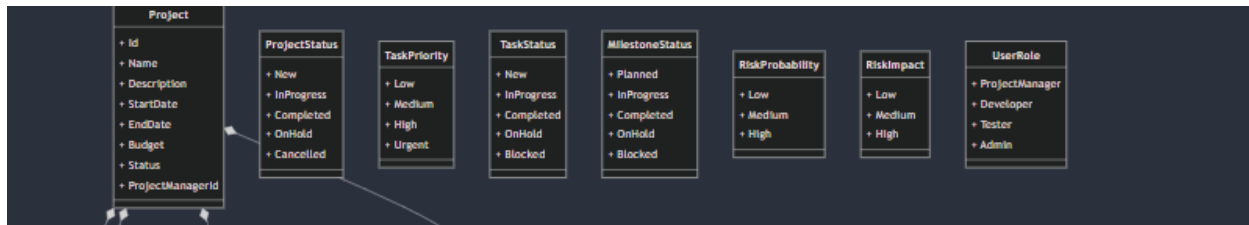


Рис 2.5 Enum елементи проекту

Принцип SOLID - це набір п'яти принципів, які допомагають створювати зручний для розширення і підтримки код.

Принцип відповідальності єдиного (SRP - Single Responsibility Principle): Кожен клас має мати лише одну відповідальність. Це дозволяє зменшити складність класів і зробити їх більш зрозумілими і тестованими.

Принцип відкритості/закритості (OCP - Open/Closed Principle): Класи мають бути відкритими для розширення, але закритими для зміни. Це дозволяє додавати нові функції до класів без внесення змін до існуючого коду.

Принцип підстановки Ліскова (LSP - Liskov Substitution Principle): Підкласи мають бути замінними на базові класи без зміни коректності програми. Це допомагає зберегти стабільність коду і зменшити ризики помилок.

Принцип сегрегації інтерфейсів (ISP - Interface Segregation Principle): Інтерфейси мають бути спеціалізованими і містити лише методи, що необхідні для певного використання. Це зменшує залежність між класами і спрощує їх тестування.

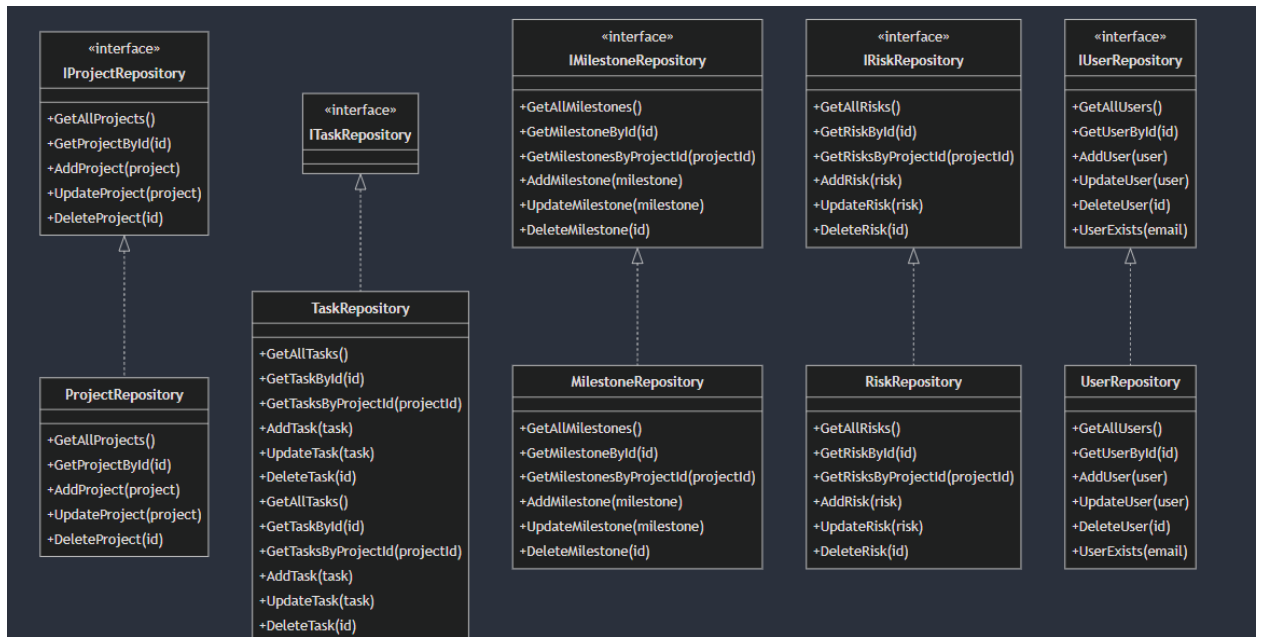


Рис 2.6 Розроблені для проекту інтерфейси

Принцип залежності інверсії (DIP - Dependency Inversion Principle): Класи мають залежати від абстракцій, а не від конкретних реалізацій. Це збільшує гнучкість коду і дозволяє легко замінювати окремі компоненти без впливу на інші частини системи.

Принцип DRY (Don't Repeat Yourself) - це принцип, що рекомендує зменшувати кількість дублювання коду в проекті. Це допомагає зменшити складність коду, збільшити ефективність розробки і зменшити ризики помилок.

Принцип KISS (Keep It Simple, Stupid) - рекомендує використовувати прості і зрозумілі рішення для розробки програмного забезпечення. Це допомагає зменшити складність проекту, збільшити ефективність розробки і покращити підтримуваність коду.

Вибір архітектурних принципів є важливим етапом розробки, що впливає на всю структуру і організацію проекту. Правильний вибір принципів дозволить створити зручний для розширення і підтримки код, що буде стабільним і ефективним протягом всього життя проекту.

Схема взаємодії модулів є ключовим елементом архітектури системи, що визначає спосіб взаємодії різних модулів між собою і забезпечує ефективну передачу інформації та даних між ними. Для проекту MVP проактивного

управління командами в аутсорс-компаніях схема взаємодії модулів має бути зрозумілою і ефективною, що забезпечить злагоджену роботу всієї системи.

Модуль управління проектами виступає як центральний модуль системи. Він збирає і обробляє інформацію про проекти, включаючи їхні цілі, терміни, бюджет і зв'язані з ними завдання. Цей модуль також відповідає за моніторинг прогресу виконання проекту, створення звітів про його хід і управління ризиками.

Модуль управління командою взаємодіє з модулем управління проектами для отримання інформації про проекти і їх членів. Він забезпечує функціональність для створення і управління профілями членів команди, визначення їхніх ролей та прав доступу, а також моніторингу їхньої активності в проектах.

Модуль аналізу та звітності одержує дані з модулів управління проектами та управління командою, обробляє їх і створює різні звіти про ефективність роботи команди і хід проектів. Він дозволяє ідентифікувати проблемні місця, виявити потенційні ризики та прогнозувати майбутню ефективність роботи команди.

Модуль моніторингу та прогнозування взаємодіє з іншими модулями для отримання інформації про проект і команду. Він використовує аналітичні інструменти для виявлення потенційних проблем і ризиків, що можуть виникнути в майбутньому. Цей модуль надає інструменти для проактивного втручання і вирішення проблем, що допомагає забезпечити успішне виконання проектів.

Схема взаємодії модулів має бути чіткою і ефективною, щоб забезпечити злагоджену роботу всієї системи. Вона має дозволити модулям обмінюватися інформацією і даними без конфліктів і затримок.

Правильне проектування схеми взаємодії модулів є ключовим фактором для створення надійної і ефективною системи проактивного управління командами в аутсорс-компаніях. Вона допомагає забезпечити ефективну

комунікацію між різними частинами системи, що є необхідним для успішної реалізації проектів і підвищення ефективності роботи команд.

Вибір технологій для реалізації MVP проактивного управління командами в аутсорс-компаніях є ключовим етапом проектування. Від правильного вибору залежить ефективність розробки, продуктивність системи, її гнучкість і масштабованість.

Веб-фреймворк визначає основу для розробки веб-додатку. ASP.NET Core є оптимальним вибором для цього проекту завдяки його гнучкості, високій продуктивності, підтримці різних платформ і широкому співтовариству розробників. ASP.NET Core надає різні функції і інструменти для розробки веб-додатків, включаючи MVC, Razor Pages, Web API та SignalR, що дозволяє створити зручний і функціональний інтерфейс для користувачів.

База даних є важливою частиною будь-якої системи управління проектами. PostgreSQL - це відкрита, надійна і гнучка реляційна база даних, що відповідає вимогам проекту з точки зору функціональності, безпеки і продуктивності. Вона підтримує різні типи даних і дозволяє створювати складні запити, що є необхідним для зберігання і обробки інформації про проекти, команди і їх активність.

Інструменти розробки допомагають розробникам створювати, тестувати і розгортати програмне забезпечення ефективно. Visual Studio - це інтегроване середовище розробки (IDE), що підтримує C# і ASP.NET Core і надає широкий набір інструментів для розробки веб-додатків. Entity Framework Core - це ORM, що дозволяє працювати з базою даних за допомогою об'єктно-орієнтованих методів, зменшуючи складність коду і збільшуючи швидкість розробки. NUnit - це фреймворк для тестування, що допомагає забезпечити якість коду і виявити помилки на ранніх етапах розробки. Docker - це платформа для контейнеризації додатків, що забезпечує незалежність від середовища розгортання і полегшує розгортання додатків на різних платформах. Git - це система контролю версій, що дозволяє відстежувати зміни в коді і забезпечує спільну роботу в команді.

Вибір технологій є важливим рішенням, що впливає на всю розробку проекту. Правильний вибір технологій дозволить створити ефективний, гнучкий і надійний MVP, що буде відповідати вимогам проекту і забезпечить його успішне впровадження.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ MVP

3.1 Розробка MVP

Модуль управління проектами є ключовою складовою системи проактивного управління командами, що забезпечує централізований пункт для планування, організації і контролю над проектами. Він надає інструменти для створення нових проектів, розподілу завдань, встановлення термінів і пріоритетів, відстеження прогресу і створення звітів про їх хід.



Рис 3.1 Карта продукту

Створення проектів: Модуль дозволяє керівникам проектів створювати нові проекти, вводячи основну інформацію про проект: назву, опис, цілі, терміни, бюджет і інші ключові деталі. Також він надає можливість структурувати проект, розділивши його на етапи і завдання, і призначити відповідальних за їх виконання.

Управління завданнями: Модуль допомагає керувати завданнями проекту: встановлювати пріоритети, призначати терміни, додавати детальні описи і документацію. Керівники проектів можуть використовувати модуль для відстеження прогресу виконання завдань, визначити затримки і виявити потенційні проблеми.

Моніторинг прогресу: Модуль надає різні інструменти для відстеження прогресу проекту. Він відображає стан виконання завдань, використання ресурсів, терміни завершення етапів. Завдяки цьому модулю керівники

проектів можуть вчасне визначити відхилення від плану і вжити необхідних заходів для їх виправлення.

Звітування: Модуль дозволяє створювати звіти про хід проекту, які містять інформацію про виконання завдань, використання ресурсів, затримки та ризики. Ці звіти допомагають інформувати замовника про прогрес проекту, а також аналізувати ефективність роботи команди і вживати необхідних заходів для її покращення.

Управління ризиками: Модуль надає інструменти для ідентифікації потенційних загроз для проекту і розробки стратегій для їх мінімізації. Він допомагає створювати список ризиків, оцінювати їх ймовірність і наслідки, а також розробляти плани заходів для зменшення ризиків.

Модуль управління проектами є необхідним інструментом для ефективного керування проектами в аутсорс-компаніях. Він допомагає зменшити ризики затримок і недоліків в проектах, покращує координацію роботи команди і забезпечує прозорість в процесі управління проектами.

Модуль управління командою - це важлива складова системи проактивного управління, що зосереджує увагу на організації та ефективному менеджменті розподілених команд проектів. Він надає інструменти для створення профілів членів команди, встановлення ролей та прав доступу, а також забезпечує платформу для зручного спілкування і співпраці.

Створення профілів команди: Модуль дозволяє створювати профілі для кожного члена команди, збираючи ключову інформацію про них: контактні дані, навичок, досвід, спеціалізації та інші релевантні деталі. Ця інформація допомагає керівникам проектів оцінити компетентність кожного члена команди і ефективно призначити завдання.

Управління ролями і правами доступу: Модуль дозволяє встановити різні ролі для членів команди, наприклад, керівник проекту, розробник, тестувальник. Він також забезпечує гнучкі настройки прав доступу до різних функцій системи, щоб кожен член команди мав доступ тільки до необхідної інформації та інструментів.

Моніторинг активності команди: Модуль відстежує активність членів команди в проектах, включаючи виконання завдань, час, що присвячений роботі, а також взаємодію з іншими членами команди. Ця інформація допомагає оцінити продуктивність кожного члена команди, визначити потенційні проблеми і забезпечити ефективне використання ресурсів.

Спілкування та співпраця: Модуль надає інструменти для зручного спілкування і співпраці між членами команди, що є особливо важливим для розподілених команд. Він може включати в себе чати, форуми і інтеграцію з іншими платформами для командної роботи, наприклад, Slack або Trello.

Звітування про продуктивність і настрої: Модуль надає інструменти для створення звітів про продуктивність і настрої в команді. Ці звіти містять інформацію про виконання завдань, час, що присвячений роботі, а також про рівень мотивації і задоволеності членів команди. Ця інформація допомагає керівникам проектів і менеджерам команд оцінити ефективність роботи команди і вжити заходів для її покращення.

Модуль управління командою є необхідним інструментом для успішного управління розподіленими командами в аутсорс-компаніях. Він допомагає забезпечити ефективну взаємодію між членами команди, покращити комунікацію і співпрацю, а також оптимізувати процес управління проектами.

Модуль аналізу та звітності - це ключовий елемент системи проактивного управління командами, що дозволяє збирати, аналізувати і візуалізувати дані про ефективність роботи команди і хід проектів. Цей модуль надає інструменти для виявлення проблемних місць та потенційних ризиків, а також для прогнозування майбутньої ефективності роботи команди.

Модуль збирає дані з різних джерел системи, включаючи модулі управління проектами та управління командою. Він відстежує виконання завдань, використання ресурсів, терміни завершення етапів, а також інформацію про активність членів команди, їхню продуктивність і настрої.

Цю інформацію модуль аналізує з використанням різних методів і інструментів. Він використовує статистичні методи для обчислення середніх значень, дисперсій та інших статистичних показників, що дозволяє ідентифікувати тренди в роботі команди. Візуалізація даних допомагає представити інформацію в зрозумілому вигляді за допомогою графіків, діаграм та інших візуальних інструментів, що полегшує аналіз і прийняття рішень.

На основі проведеного аналізу модуль створює різні звіти, що допомагають керівникам проектів і менеджерам команд оцінити ефективність роботи команди і вжити необхідних заходів. Звіти можуть містити інформацію про продуктивність команди, виконання завдань, затримки, ризики, а також про настрій і мотивацію членів команди.

Модуль аналізу і звітності також дозволяє використовувати інформацію для прогнозування майбутньої ефективності роботи команди. На основі аналізу історичних даних модуль може передбачити ймовірність виконання завдань в час, ризики затримок і потенційні проблеми. Це дозволяє керівникам проектів вживати заходів для мінімізації ризиків і забезпечення успішного виконання проектів.

Модуль аналізу і звітності є важливим інструментом проактивного управління командами, що дозволяє перейти від реактивного підходу до проблем до передбачення і профілактики. Завдяки збору, аналізу і візуалізації даних можна виявити проблемні місця і вжити необхідних заходів для покращення ефективності роботи команди і забезпечення успішної реалізації проектів.

Модуль моніторингу та прогнозування є ключовою складовою системи проактивного управління командами, що надає інструменти для раннього виявлення потенційних проблем і ризиків, що можуть виникнути в проектах. Цей модуль дозволяє керівникам проектів і менеджерам команд вживати заходів для виправлення ситуації ще до того, як вона стане кризовою.

Модуль збирає дані з інших модулів системи, включаючи модулі управління проектами, управління командою і аналізу та звітності. Він аналізує дані про продуктивність команди, виконання завдань, терміни, ризики, а також інформацію про настрої і мотивацію членів команди.

Застосовуючи алгоритми машинного навчання і інші аналітичні методи, модуль моніторингу та прогнозування виявляє тренди і зв'язки в даних. Він може визначити фактори, що впливають на ефективність команди, і передбачити ймовірність виникнення проблем в майбутньому.

Модуль надає інструменти для проактивного втручання і вирішення проблем. Він може генерувати оповіщення для керівників проектів і менеджерів команд про потенційні ризики і проблеми, що можуть виникнути в майбутньому. Це дозволяє вчасно вжити заходів для мінімізації ризиків і забезпечення успішного виконання проектів.

Модуль також може надавати інструменти для вирішення проблем, що виникли. Наприклад, він може допомагати ідентифікувати членів команди, що потрібно додатково навчити, або визначити завдання, що вимагають додаткових ресурсів.

Модуль моніторингу та прогнозування є основою проактивного управління командами. Він допомагає перейти від реактивного підходу до проблем до проактивного втручання і передбачення. Завдяки використанню аналітичних інструментів і алгоритмів машинного навчання можна виявити потенційні ризики і вжити необхідних заходів для забезпечення успішного виконання проектів.

Впровадження функцій безпеки та захисту даних є необхідним етапом розробки будь-якої системи, особливо для систем управління проектами, що містять важливу конфіденційну інформацію про проекти, команди і їхню діяльність.

Аутентифікація і авторизація - це ключові елементи безпеки системи, що забезпечують доступ до системи тільки авторизованим користувачам. Аутентифікація перевіряє ідентичність користувача, наприклад, за допомогою

імені користувача та пароля, а авторизація визначає рівень доступу користувача до різних частин системи.

Шифрування даних - це процес перетворення даних в нечитабельний вид, що дозволяє захистити їх від несанкціонованого доступу. Шифрування використовує спеціальні алгоритми і ключі для перетворення даних і може бути застосовано до різних типів даних, включаючи паролі, конфіденційні файли та іншу чутливу інформацію.

Контроль доступу - це механізм, що дозволяє визначити рівень доступу користувачів до різних частин системи. Наприклад, керівники проектів можуть мати повний доступ до всіх даних і функцій системи, тоді як члени команди можуть мати доступ лише до інформації і функцій, що відповідають їх ролям.

Перевірка цілісності даних - це процес перевірки цілісності даних, що дозволяє виявити несанкціоновані зміни в даних. Система може використовувати хешування даних або інші методи для перевірки цілісності даних і забезпечення їхньої надійності.

Регулярне оновлення безпеки - це ключовий елемент захисту даних. Важливо регулярно оновлювати програмне забезпечення системи, щоб виправити виявлені вразливості і захистити систему від нових загроз.

Резервне копіювання даних - це процес створення резервних копій даних, що дозволяє відновити дані в разі поломки системи або випадкового видалення даних.

Впровадження функцій безпеки і захисту даних є необхідним етапом розробки системи проактивного управління командами в аутсорс-компаніях. Це допомагає забезпечити довірче середовище для зберігання і обробки критичної інформації, що є основою для ефективного і надійного функціонування системи.

ProjectManagementForm є ключовою формою в системі проактивного управління командами. Вона надає інтерфейс для керування проектами і

дозволяє користувачам виконувати різні операції з проектами, включаючи створення, редагування, видалення і перегляд деталей.

Головне вікно форми відображає список всіх проектів в табличному вигляді. Кожен рядок таблиці представляє один проект і містить основну інформацію про нього, наприклад, назву, опис, статус, термін виконання, бюджет, ім'я відповідального менеджера і іншу релевантну інформацію.

Додаткова функціональність форми включає:

Додавання нового проекту: користувач може додати новий проект, ввівши необхідні деталі у форму. Форма надає спеціальні поля для введення назви проекту, опису, дати початку і завершення, бюджету, статусу і іншої релевантної інформації.

Редагування проекту: користувач може змінити існуючий проект, вибравши його зі списку і відкривши форму редагування. Форма відображає існуючі значення властивостей проекту, які користувач може змінити.

Видалення проекту: користувач може видалити існуючий проект зі списку, вибравши його і підтвердивши видалення.

Перегляд деталей проекту: користувач може переглянути детальну інформацію про проект, вибравши його зі списку. Форма деталей відображає всю доступну інформацію про проект, включаючи список завдань, етапів, ризиків і інших деталей.

ProjectManagementForm є зручним інструментом для керівників проектів і менеджерів команд, що допомагає їм ефективно управляти проектами і відстежувати їх прогрес. Вона забезпечує централізований доступ до інформації про проекти і надає різні інструменти для роботи з проектами, що спрощує процес управління проектами і зменшує ризики затримок і помилок.

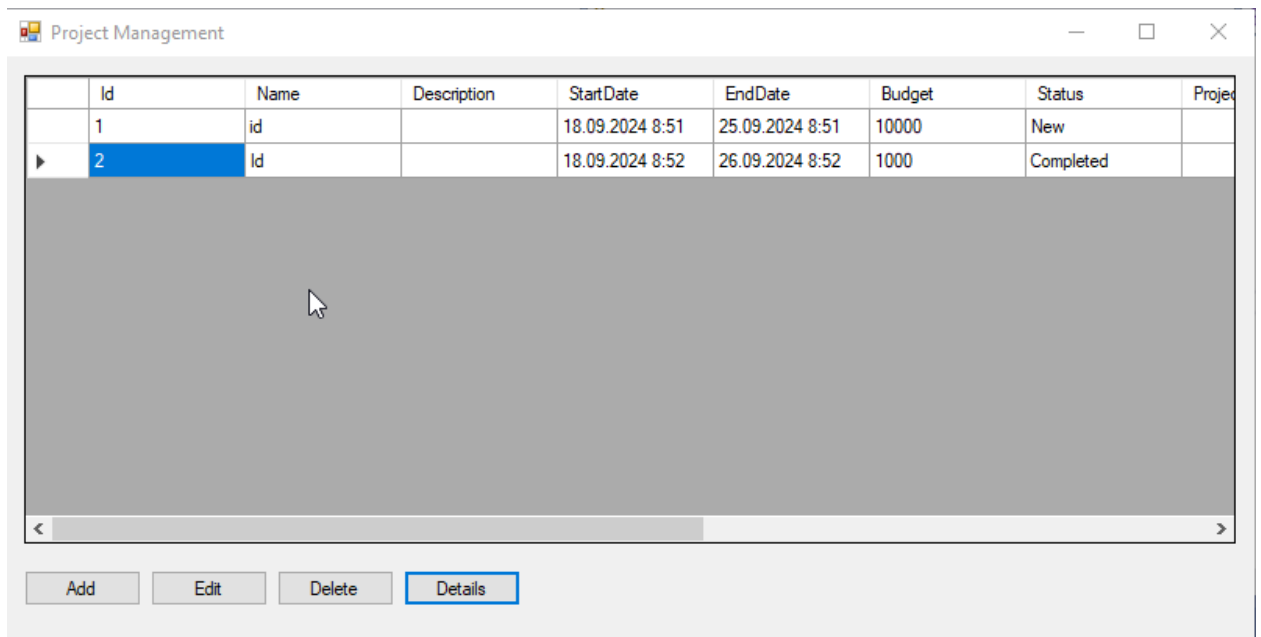


Рис 3.2 ProjectManagementForm

AddProjectForm є спеціалізованою формою в системі проактивного управління командами, що надає інтерфейс для створення нових проектів. Вона дозволяє користувачам ввести необхідні деталі і зберегти новий проект в базі даних.

Інтерфейс форми складається з різних елементів управління:

Поля введення тексту: форми містять поля для введення назви проекту, опису, бюджету, які забезпечують введення текстової інформації.

Елементи вибору даних: форма містить календарі для вибору дати початку і завершення проекту, а також спадний список для вибору статусу проекту.

Спадний список для вибору менеджера проекту: дозволяє вибрати відповідального менеджера проекту з списку доступних користувачів.

Функціональність форми включає:

Валідацію введених даних: перед збереженням проекту форма перевіряє коректність введених даних, наприклад, наявність назви проекту, правильний формат дати і бюджету. Якщо виявлені помилки, користувач отримує повідомлення з зазначенням необхідних виправлень.

Створення об'єкта проекту: після успішної валідації введених даних форма створює новий об'єкт класу Project, що представляє новий проект. Властивості об'єкта проекту заповнюються даними, введеними користувачем.

Збереження проекту: після створення об'єкта проекту форма зберігає його в базі даних за допомогою репозиторію проектів.

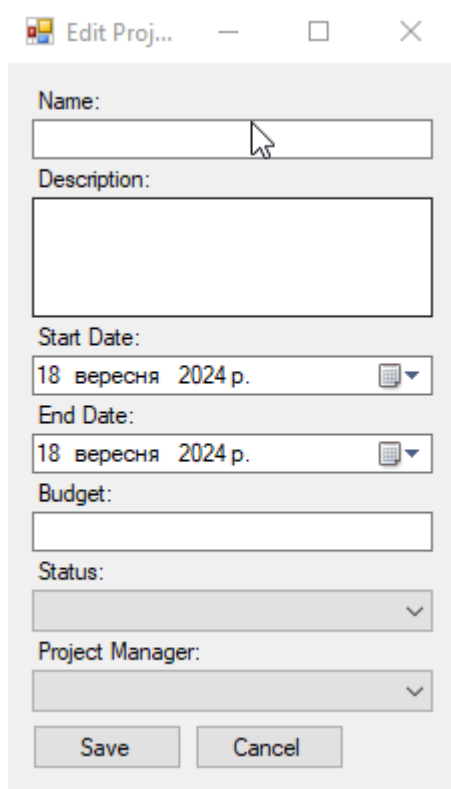


Рис 3.3 EditProjectForm

Закриття форми: після успішного збереження проекту форма закривається, і користувач повертається до головної форми ProjectManagementForm.

AddProjectForm є зручним інструментом для додавання нових проектів в систему управління проектами. Вона забезпечує інтуїтивно зрозумілий інтерфейс і виконує важливі функції для забезпечення коректності і надійності доданих проектів.

The image shows a standard Windows-style dialog box titled "Add Proj...". It features a title bar with minimize, maximize, and close buttons. The main area contains several labeled input fields: "Name:" with a text box, "Description:" with a larger text area, "Start Date:" and "End Date:" with date pickers showing "18 вересня 2024 р.", "Budget:" with a text box, "Status:" with a dropdown menu, and "Project Manager:" with a dropdown menu. At the bottom, there are two buttons: "Save" and "Cancel".

Рис 3.4 AddProjectForm

ProjectDetailsForm - це спеціалізована форма в системі проактивного управління командами, що надає детальну інформацію про конкретний проект. Вона відображає всі ключові властивості проекту, а також надає доступ до пов'язаних з ним даних, таких як список завдань, етапів, ризиків і інформації про членів команди.

Інтерфейс форми складається з різних елементів управління:

Етикетки з значеннями: форма відображає основні властивості проекту у вигляді етикеток, наприклад, назва проекту, опис, дата початку і завершення, бюджет, статус, ім'я менеджера проекту. Ці етикетки наповнені значеннями, які отримані з бази даних для конкретного проекту.

Списки: форма відображає список завдань, етапів і ризиків, пов'язаних з проектом. Кожен елемент списку містить коротку інформацію про об'єкт (наприклад, назву завдання або опис ризику).

Кнопка "Закрити": дозволяє користувачеві закрити форму і повернутися до головної форми ProjectManagementForm.

Функціональність форми включає:

Завантаження даних проекту: форма отримує інформацію про конкретний проект з бази даних за допомогою репозиторію проектів.

Відображення даних проекту: форма відображає отримані дані про проект у зрозумілому вигляді, використовуючи етикетки, списки і інші елементи інтерфейсу.

Навігація до деталей об'єктів: користувач може натиснути на елемент списку завдань, етапів або ризиків, щоб відкрити детальні форми для цих об'єктів.

ProjectDetailsForm забезпечує зручний і інтуїтивно зрозумілий спосіб перегляду детальної інформації про проект, що дозволяє керівникам проектів і менеджерам команд отримати повне уявлення про стан проекту і вжити необхідних заходів для його успішної реалізації.



Рис 3.5 ProjectDetailsForm

RoleManagementForm - це спеціалізована форма в системі проактивного управління командами, що надає інтерфейс для керування ролями в

конкретній команді. Вона дозволяє користувачам створювати, редагувати, видаляти ролі, а також керувати дозволами для кожної ролі.

Головне вікно форми відображає список всіх ролей, що призначені для конкретної команди. Кожен рядок таблиці представляє одну роль і містить основну інформацію про неї, наприклад, назву, опис.

Додаткова функціональність форми включає:

Додавання нової ролі: користувач може додати нову роль, ввівши необхідні деталі у форму. Форма надає спеціальні поля для введення назви ролі і опису.

Редагування ролі: користувач може змінити існуючу роль, вибравши її зі списку і відкривши форму редагування. Форма відображає існуючі значення властивостей ролі, які користувач може змінити.

Видалення ролі: користувач може видалити існуючу роль зі списку, вибравши її і підтвердивши видалення.

Перегляд деталей ролі: користувач може переглянути детальну інформацію про роль, вибравши її зі списку. Форма деталей відображає всю доступну інформацію про роль, включаючи список пов'язаних з нею дозволів.

Управління дозволами ролі: користувач може додати, редагувати або видалити дозволи, пов'язані з конкретною роллю. Для цього використовується окрема форма "PermissionManagementForm", що дозволяє керувати дозволами для кожної ролі окремо.

RoleManagementForm є ключовим інструментом для визначення і керування ролями в команді. Вона забезпечує централізований доступ до інформації про ролі і надає інструменти для визначення прав і дозволів для кожної ролі. Це дозволяє ефективно контролювати доступ членів команди до різних функцій і ресурсів в системі управління проектами.

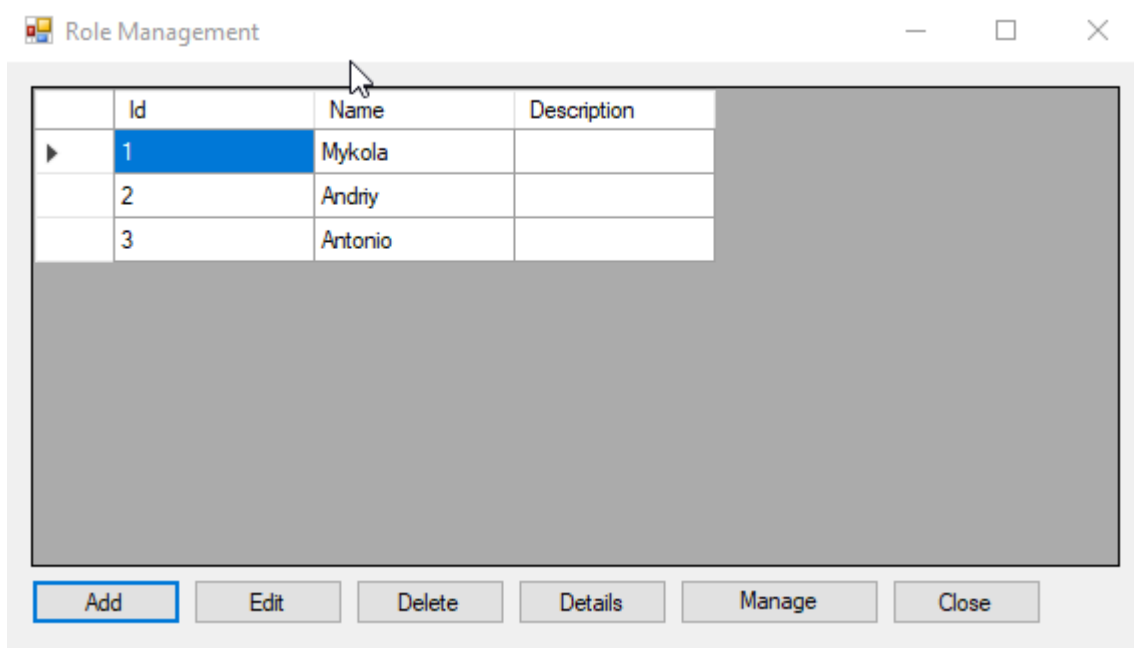


Рис 3.6 RoleManagementForm

EditRoleForm - це спеціалізована форма в системі проактивного управління командами, що надає інтерфейс для редагування властивостей існуючих ролей в конкретній команді. Вона дозволяє користувачам змінити назву і опис ролі, а також переглянути і можливо модифікувати пов'язані з нею дозволи.

Інтерфейс форми складається з різних елементів управління:

Поля введення тексту: форма містить поля для введення назви ролі і опису, що дозволяють користувачам змінити цю інформацію.

Кнопка "Зберегти": дозволяє користувачам зберегти зміни, що були внесені в властивості ролі.

Кнопка "Скасувати": дозволяє користувачам скасувати внесені зміни і закрити форму без збереження.

Функціональність форми включає:

Завантаження даних ролі: перед відкриттям форми з репозиторію завантажуються дані про обрану роль і відображаються в полях форми.

Редагування властивостей ролі: користувач може внести зміни в назву і опис ролі, вводячи нові значення в відповідні поля форми.

Збереження змін: після внесення змін користувач може натиснути на кнопку "Зберегти", що викликає оновлення даних ролі в базі даних.

Скасування змін: користувач може натиснути на кнопку "Скасувати", що закриває форму без збереження змін.

EditRoleForm є зручним інструментом для редагування існуючих ролей в команді. Вона дозволяє змінити основні властивості ролей, такі як назва і опис, що спрощує процес управління ролями і забезпечує гнучкість в визначенні прав і дозволів для членів команд.

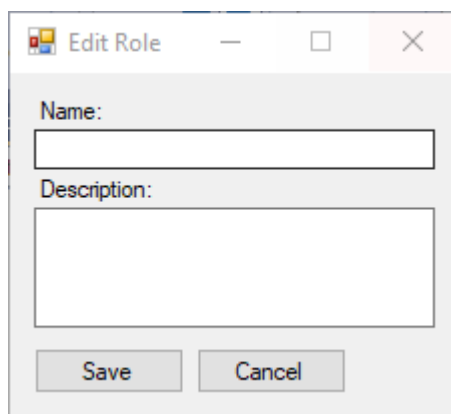
The image shows a screenshot of a software window titled "Edit Role". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are two text input fields. The first is labeled "Name:" and the second is labeled "Description:". Below these fields, there are two buttons: "Save" and "Cancel". The "Save" button is on the left and the "Cancel" button is on the right.

Рис 3.7 EditRoleForm

PermissionManagementForm - це спеціалізована форма в системі проактивного управління командами, що надає інтерфейс для керування дозволами конкретної ролі. Вона дозволяє користувачам створювати, редагувати і видаляти дозволи, пов'язані з цією роллю, тим самим визначаючи функціональність, що доступна користувачам з цією роллю.

Головне вікно форми відображає список всіх дозволів, що призначені для конкретної ролі. Кожен рядок таблиці представляє один дозвіл і містить основну інформацію про нього, наприклад, назву, опис.

Додаткова функціональність форми включає:

Додавання нового дозволу: користувач може додати новий дозвіл, ввівши необхідні деталі у форму. Форма надає спеціальні поля для введення назви дозволу і опису.

Редагування дозволу: користувач може змінити існуючий дозвіл, вибравши його зі списку і відкривши форму редагування. Форма відображає існуючі значення властивостей дозволу, які користувач може змінити.

Видалення дозволу: користувач може видалити існуючий дозвіл зі списку, вибравши його і підтвердивши видалення.

Закриття форми: користувач може закрити форму і повернутися до головної форми RoleManagementForm.

PermissionManagementForm є ключовим інструментом для визначення і керування дозволами в системі управління проектами. Вона забезпечує централізований доступ до інформації про дозволи і надає інструменти для визначення доступу користувачів до різних функцій і ресурсів. Це дозволяє ефективно контролювати доступ до критичної інформації і забезпечувати безпеку даних.

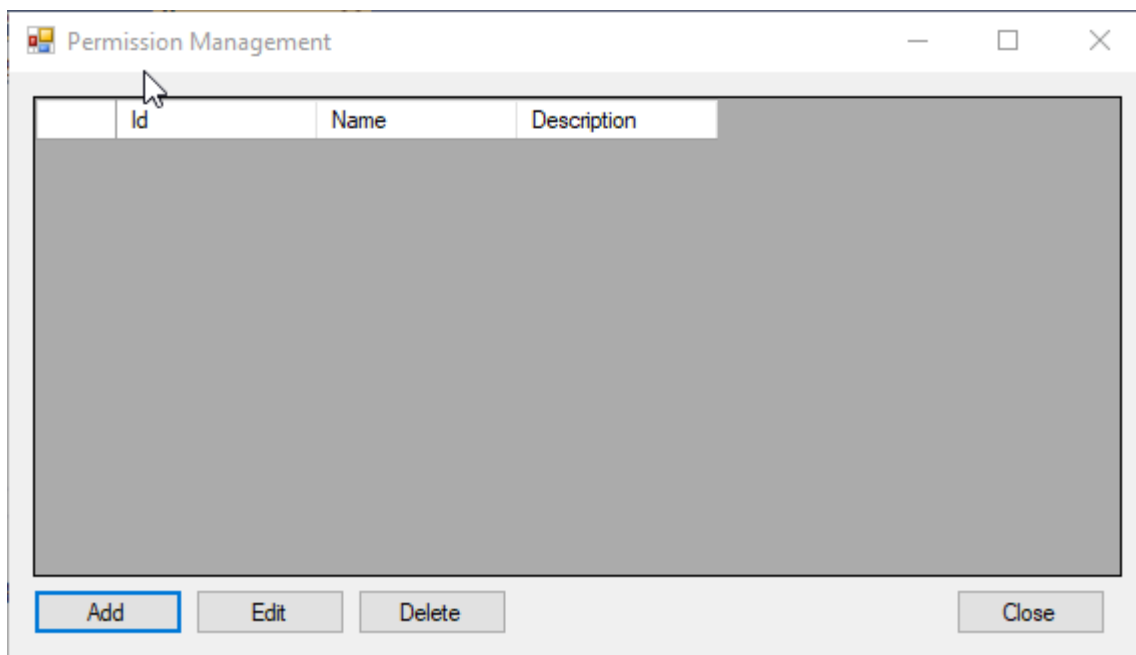


Рис 3.8 PermissionManagementForm

AddPermissionForm є спеціалізованою формою в системі проактивного управління командами, що надає інтерфейс для створення нових дозволів. Вона дозволяє користувачам ввести необхідні деталі і зберегти новий дозвіл в базі даних.

Інтерфейс форми складається з різних елементів управління:

Поля введення тексту: форми містять поля для введення назви дозволу і опису, які забезпечують введення текстової інформації.

Кнопка "Зберегти": дозволяє користувачам зберегти новий дозвіл в базі даних.

Кнопка "Скасувати": дозволяє користувачам скасувати внесені зміни і закрити форму без збереження.

Функціональність форми включає:

Валідацію введених даних: перед збереженням дозволу форма перевіряє коректність введених даних, наприклад, наявність назви дозволу. Якщо виявлені помилки, користувач отримує повідомлення з зазначенням необхідних виправлень.

Створення об'єкта дозволу: після успішної валідації введених даних форма створює новий об'єкт класу `Permission`, що представляє новий дозвіл. Властивості об'єкта дозволу заповнюються даними, введеними користувачем.

Збереження дозволу: після створення об'єкта дозволу форма зберігає його в базі даних за допомогою репозиторію дозволів.

Прив'язка дозволу до ролі: після збереження дозволу в базі даних форма додає цей дозвіл до списку дозволів ролі, до якої він має бути призначений.

Закриття форми: після успішного збереження дозволу форма закривається, і користувач повертається до головної форми `RoleManagementForm`.

`AddPermissionForm` є зручним інструментом для додавання нових дозволів в системі управління проектами. Вона забезпечує інтуїтивно зрозумілий інтерфейс і виконує важливі функції для забезпечення коректності і надійності доданих дозволів.

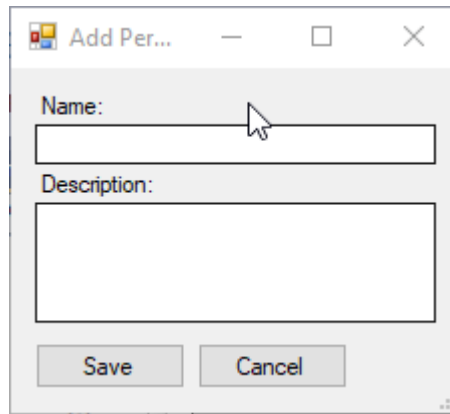


Рис 3.9 AddPermissionForm

TeamManagementForm - це ключова форма в системі проактивного управління командами, що надає централізований інтерфейс для керування розподіленими командами в проектах. Вона дозволяє користувачам створювати, редагувати, видаляти команди, а також керувати їхніми членами, ролями і дозволами.

Головне вікно форми відображає список всіх команд, пов'язаних з конкретним проектом. Кожен рядок таблиці представляє одну команду і містить основну інформацію про неї, наприклад, назву, опис, кількість членів.

Додаткова функціональність форми включає:

Додавання нової команди: користувач може додати нову команду, ввівши необхідні деталі у форму. Форма надає спеціальні поля для введення назви команди і опису, а також можливість вибору проекту, до якого належить команда.

Редагування команди: користувач може змінити існуючу команду, вибравши її зі списку і відкривши форму редагування. Форма відображає існуючі значення властивостей команди, які користувач може змінити.

Видалення команди: користувач може видалити існуючу команду зі списку, вибравши її і підтвердивши видалення.

Перегляд деталей команди: користувач може переглянути детальну інформацію про команду, вибравши її зі списку. Форма деталей відображає

всю доступну інформацію про команду, включаючи список членів, призначених ролей і дозволів.

Управління членами команди: користувач може додати, видалити або змінити членів команди за допомогою спеціальної форми.

Управління ролями в команді: користувач може додати, редагувати або видалити ролі, пов'язані з конкретною командою. Для цього використовується окрема форма "RoleManagementForm", що дозволяє керувати ролями в команді окремо.

TeamManagementForm є ключовим інструментом для організації і керування розподіленими командами в системі управління проектами. Вона забезпечує централізований доступ до інформації про команди і надає інструменти для керування їх складом, ролями і дозволами, що дозволяє ефективно організувати роботу команд і забезпечити успішне виконання проектів.

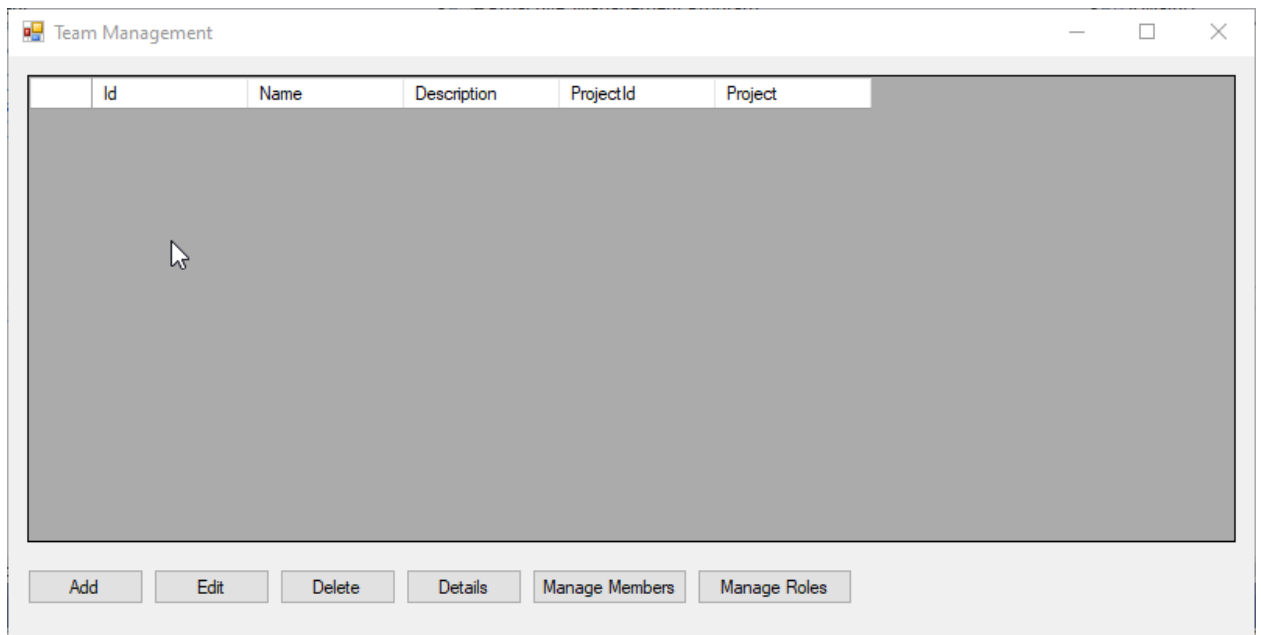


Рис 3.10 TeamManagementForm

3.2 Тестування MVP

Розробка тест-кейсів і сценаріїв тестування є ключовим етапом розробки будь-якого програмного забезпечення, що дозволяє перевірити його

функціональність, продуктивність, безпеку і зручність користування. Цей етап дозволяє виявити помилки і недоліки в програмі ще до того, як вона буде впроваджена в експлуатацію.

Тест-кейси - це документи, що описують набір умов і дій, які необхідно виконати для перевірки певного функціоналу системи. Кожен тест-кейс містить опис вхідних даних, очікуваних результатів і критеріїв успіху.

Сценарії тестування - це набори тест-кейсів, що відображають реальні сценарії використання системи. Вони допомагають перевірити, як система функціонує в різних умовах і за різних сценаріїв.

Розробка тест-кейсів і сценаріїв тестування - це складна і відповідальна задача. Необхідно враховувати різні аспекти системи, включаючи функціональність, продуктивність, безпеку і зручність користування.

Тестування функціональності передбачає перевірку того, чи система виконує всі необхідні функції відповідно до заданих вимог. Тест-кейси для тестування функціональності включають перевірку введення і виведення даних, обробку запитів, виконання операцій і інших ключових функцій.

Тестування продуктивності перевіряє, як система працює під навантаженням і з великим об'ємом даних. Тест-кейси для тестування продуктивності включають перевірку часу відгуку системи, пропускну здатності і надійності під навантаженням.

Тестування безпеки перевіряє, чи система захищена від несанкціонованого доступу і злочинних дій. Тест-кейси для тестування безпеки включають перевірку аутентифікації і авторизації, шифрування даних і інших механізмів захисту даних.

Тестування зручності користування перевіряє, чи система зручна в користуванні і інтуїтивна для користувачів. Тест-кейси для тестування зручності користування включають перевірку інтерфейсу користувача, навігації по системі і загальної ергономіки.

Правильно розроблені тест-кейси і сценарії тестування є ключовим фактором для створення якісного і надійного програмного забезпечення. Вони

допомагають виявити помилки і недоліки в програмі ще до того, як вона буде впроваджена в експлуатацію, що зменшує ризики помилок і забезпечує стабільність і надійність системи.

Для оцінки ефективності MVP проактивного управління командою аутсорс-компанії, необхідно аналізувати низку ключових метриків. До них належать кількість користувачів, проектів, завдань, етапів і ризиків, а також їхній статус. Окрім того, важливо враховувати час виконання завдань, кількість відкритих і заблокованих завдань, кількість коментарів до завдань, а також час реакції користувачів на повідомлення.

Кількість користувачів – за оптимістичним сценарієм, кількість активних користувачів може сягнути 100 осіб протягом місяця. Це може призвести до реєстрації близько 10 нових користувачів щомісяця. В системі можуть використовуватися чотири основні ролі: Project Manager, Developer, Tester і Admin.

Кількість проектів – система може зберігати до 50 проектів, з яких в будь-який момент часу близько 10 проектів знаходяться в статусі "InProgress". Кількість завершених проектів може сягнути 20. Кожен Project Manager може керувати до 5-ти проектів.

Кількість завдань – в системі може зберігатися до 200 завдань, з яких приблизно 50 завдань знаходяться в статусі "InProgress". Кількість завершених завдань може сягнути 100. Кожен проект може мати до 20 завдань, причому більшість завдань мають середній рівень пріоритету. Кожен Developer або Tester може виконувати до 10 завдань.

Кількість етапів – в системі може зберігатися до 100 етапів, з яких до 20 етапів знаходяться в статусі "InProgress". Кожен проект може мати до 5 етапів.

Кількість ризиків: в системі може зберігатися до 50 ризиків, з яких до 10 ризиків мають високу імовірність виникнення і до 5 ризиків мають високий вплив на проект. Кожен проект може мати до 5 ризиків.

Інші метрики – середній час виконання завдань може складати 2-3 дні. Кількість відкритих завдань може сягнути 20. Кількість заблокованих завдань

може складати до 5. Кількість коментарів до завдань може сягнути 50 за місяць. Середній час реакції користувачів на повідомлення може складати до 1 години.

Інтуїтивність інтерфейсу – час навчання користувачів може сягнути до 2 годин. Кількість помилок, що робляться користувачами під час використання системи, має бути мінімальною (наприклад, до 5 помилок за місяць). Рівень задоволеності користувачів має бути високим і може бути оцінений за результатами анкетування або збору зворотного зв'язку.

ВИСНОВКИ

Дослідження успішно визначило ключові аспекти проактивного управління командами в аутсорс-компаніях, що дозволило розробити концепцію і функціональний прототип MVP системи.

Аналіз сучасних методів управління командами в аутсорс-компаніях показав, що традиційні методи, хоч і мають певну актуальність, не завжди ефективні в умовах розподілених команд. Відстань, різні часові пояси, культурні різниці і мовні бар'єри створюють значні перешкоди для ефективної комунікації і співпраці.

Визначення ключових факторів проактивного управління командою визначило необхідність зміни фокусу з реактивного підходу до проактивного. Це передбачає постійний моніторинг, аналіз і передбачення можливих проблем, а також впровадження механізмів раннього виявлення і вирішення проблем.

Розробка концепції MVP проактивного управління командою забезпечила чітке розуміння функціональності і цілей системи. MVP було зосереджено на ключових функціях, що допомагають зменшити ризики затримок і недоліків в проектах, підвищити продуктивність і мотивацію команди, покращити комунікацію і співпрацю в команді.

Створення функціонального прототипу MVP дозволило перевірити практичну реалізацію концепції і забезпечити функціональність системи. Прототип був розроблений з використанням C# і ASP.NET Core та інтегрований з базою даних PostgreSQL.

Тестування та оцінка розробленого прототипу показали позитивні результати. Прототип ефективно виконував ключові функції, з високою продуктивністю і зручністю користування. Тестування також визначило незначні недоліки, що можуть бути виправлені на наступних етапах розробки.

В цілому, дослідження успішно продемонструвало значення проактивного управління командами в аутсорс-компаніях. Розроблений прототип MVP дозволяє створити ефективну систему управління проектами,

що допомагає підвищити продуктивність команд, зменшити ризики і забезпечити успішне виконання проєктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AKHTAR, Asma; BAKHTAWAR, Birra; AKHTAR, Samia. Extreme programming vs scrum: A comparison of agile models. *International Journal of Technology, Innovation and Management (IJTIM)*, 2022, 2.2: 80-96.
2. ALAMI, Adam; KRANCHER, Oliver; PAASIVAARA, Maria. The journey to technical excellence in agile software development. *Information and Software Technology*, 2022, 150: 106959.
3. AL-SAQQA, Samar; SAWALHA, Samer; ABDELNABI, Hiba. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 2020, 14.11.
4. ARSHAD, Beenish; HASSAN, Hamid; AZAM, Akbar. How does managerial coaching influence knowledge sharing in the workplace? A perspective of proactive motivation model. *Management Research Review*, 2024.
5. BAAH, Charles, et al. Understanding the influence of environmental production practices on firm performance: a proactive versus reactive approach. *Journal of Manufacturing Technology Management*, 2021, 32.2: 266-289.
6. BAKKER, Arnold B., et al. Proactive vitality management, work engagement, and creativity: The role of goal orientation. *Applied Psychology*, 2020, 69.2: 351-378.
7. BREGE, Harald; KINDSTRÖM, Daniel. Exploring proactive market strategies. *Industrial Marketing Management*, 2020, 84: 75-88.
8. CASTELLI, Vittorio, et al. Proactive management of software aging. *IBM Journal of Research and Development*, 2001, 45.2: 311-332.
9. DARAOJIMBA, Emmanuel Chibuike, et al. Comprehensive review of agile methodologies in project management. *Computer Science & IT Research Journal*, 2024, 5.1: 190-218.
10. EDISON, Henry; WANG, Xiaofeng; CONBOY, Kieran. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, 2021, 48.8: 2709-2731.
11. HINDERKS, Andreas, et al. Approaches to manage the user experience process in Agile software development: A systematic literature review. *Information and Software Technology*, 2022, 150: 106957.
12. KASAULI, Rashidah, et al. Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software*, 2021, 172: 110851.
13. KOLOMVATSOS, Kostas. Proactive tasks management for pervasive computing applications. *Journal of Network and Computer Applications*, 2021, 176: 102948.

14. KOLOMVATSOS, Kostas; ANAGNOSTOPOULOS, Christos. A deep learning model for demand-driven, proactive tasks management in pervasive computing. *IoT*, 2020, 1.2: 15.
15. KOLOMVATSOS, Kostas; ANAGNOSTOPOULOS, Christos. A proactive statistical model supporting services and tasks management in pervasive applications. *IEEE Transactions on Network and Service Management*, 2022, 19.3: 3020-3031.
16. KOLOMVATSOS, Kostas; ANAGNOTOPOULOS, Christos. Proactive Tasks Management based on a Deep Learning Model. *arXiv preprint arXiv:2007.12857*, 2020.
17. KOLOMVATSOS, Kostas; ANAGNOTOPOULOS, Christos. Proactive Tasks Management based on a Deep Learning Model. *arXiv preprint arXiv:2007.12857*, 2020.
18. KUHRMANN, Marco, et al. What makes agile software development agile?. *IEEE transactions on software engineering*, 2021, 48.9: 3523-3539.
19. LIU, Hu-Chen, et al. Failure mode and effects analysis for proactive healthcare risk evaluation: a systematic literature review. *Journal of evaluation in clinical practice*, 2020, 26.4: 1320-1337.
20. MALIK, Mohsin; SARWAR, Shagufta; ORR, Stuart. Agile practices and performance: Examining the role of psychological empowerment. *International Journal of Project Management*, 2021, 39.1: 10-20.
21. MARMIER, François; VARNIER, Christophe; ZERHOUNI, Nourredine. Proactive, dynamic and multi-criteria scheduling of maintenance activities. *International Journal of Production Research*, 2009, 47.8: 2185-2201.
22. MERGEL, Ines; GANAPATI, Sukumar; WHITFORD, Andrew B. Agile: A new way of governing. *Public Administration Review*, 2021, 81.1: 161-165.
23. MISHRA, Alok, et al. Organizational issues in embracing Agile methods: an empirical assessment. *International Journal of System Assurance Engineering and Management*, 2021, 12.6: 1420-1433.
24. POPOOLA, Oladapo Adeboye, et al. Conceptualizing agile development in digital transformations: Theoretical foundations and practical applications. *Engineering Science & Technology Journal*, 2024, 5.4: 1524-1541.
25. SHORE, James; WARDEN, Shane. *The art of agile development*. "O'Reilly Media, Inc.", 2021.
26. SMITH, Preston G.; MERRITT, Guy M. *Proactive risk management: Controlling uncertainty in product development*. productivity press, 2020.
27. SMITH, Preston G.; MERRITT, Guy M. *Proactive risk management: Controlling uncertainty in product development*. productivity press, 2020.

28. STRODE, Diane; DINGSØYR, Torgeir; LINDSJORN, Yngve. A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, 2022, 27.2: 56.

29. TYLER, Beverly B., et al. Environmental practice adoption in SMEs: The effects of firm proactive orientation and regulatory pressure. *Journal of Small Business Management*, 2024, 62.5: 2211-2246.

30. YORKE-SMITH, Neil, et al. The design of a proactive personal agent for task management. *International Journal on Artificial Intelligence Tools*, 2012, 21.01: 1250004.

31. ZASA, Federico P.; PATRUCCO, Andrea; PELLIZZONI, Elena. Managing the hybrid organization: How can agile and traditional project management coexist?. *Research-Technology Management*, 2020, 64.1: 54-63.

ДОДАТКИ

Додаток А

```
public class Team
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }

    // Список членів команди
    public List<User> Members { get; set; } = new List<User>();
    public List<(int UserId, int RoleId)> MembersRoles { get; set; } = new List<(int
UserId, int RoleId)>();

    // Проект, до якого належить команда
    public int ProjectId { get; set; }
    public Project Project { get; set; }

    // Список ролей в команді
    public List<Role> Roles { get; set; } = new List<Role>();
}
```

Додаток Б

```
public class User
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Email { get; set; }
    public string PasswordHash { get; set; }

    public UserRole Role { get; set; }

    // Список проектів, в яких користувач бере участь
    public ICollection<Project> Projects { get; set; } = new List<Project>();

    // Список завдань, які призначені користувачеві
    public ICollection<Task> AssignedTasks { get; set; } = new List<Task>();

}

// Перелічення для ролі користувача
public enum UserRole
{
    ProjectManager,
    Developer,
    Tester,
    Admin
}
```

Додаток В

```
public interface IUserRepository
{
    // Отримання всіх користувачів
    IEnumerable<User> GetAllUsers();

    // Отримання користувача за Id
    User GetUserById(int id);

    // Додавання нового користувача
    void AddUser(User user);

    // Оновлення користувача
    void UpdateUser(User user);

    // Видалення користувача
    void DeleteUser(int id);

    // Перевірка існування користувача за електронною поштою
    bool UserExists(string email);

    // ... Додаткові методи, якщо потрібно (наприклад, пошук користувачів за
    критеріями, аутентифікація, авторзація)
}
```

Додаток Г

```
public partial class MainForm : Form
{
    private IProjectRepository _projectRepository;
    private ITeamRepository _teamRepository;
    private IRoleRepository _roleRepository;
    private IPermissionRepository _permissionRepository;

    public MainForm(IProjectRepository projectRepository, ITeamRepository
teamRepository,
        IRoleRepository roleRepository, IPermissionRepository
permissionRepository)
    {
        InitializeComponent();
        _projectRepository = projectRepository;
        _teamRepository = teamRepository;
        _roleRepository = roleRepository;
        _permissionRepository = permissionRepository;
        CreateComponent();
    }

    private void CreateComponent()
    {
        this.projectManagementButton = new System.Windows.Forms.Button();
        this.teamManagementButton = new System.Windows.Forms.Button();
        this.SuspendLayout();
        //
        // projectManagementButton
        //
        this.projectManagementButton.Location = new System.Drawing.Point(12,
12);
        this.projectManagementButton.Name = "projectManagementButton";
        this.projectManagementButton.Size = new System.Drawing.Size(150, 23);
        this.projectManagementButton.TabIndex = 0;
        this.projectManagementButton.Text = "Project Management";
        this.projectManagementButton.UseVisualStyleBackColor = true;
        this.projectManagementButton.Click += new
System.EventHandler(this.projectManagementButton_Click);
        //
        // teamManagementButton
        //
        this.teamManagementButton.Location = new System.Drawing.Point(12, 41);
        this.teamManagementButton.Name = "teamManagementButton";
        this.teamManagementButton.Size = new System.Drawing.Size(150, 23);
```

```

        this.teamManagementButton.TabIndex = 1;
        this.teamManagementButton.Text = "Team Management";
        this.teamManagementButton.UseVisualStyleBackColor = true;
        this.teamManagementButton.Click += new
System.EventHandler(this.teamManagementButton_Click);
        //
        // MainForm
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(284, 81);
        this.Controls.Add(this.teamManagementButton);
        this.Controls.Add(this.projectManagementButton);
        this.Name = "MainForm";
        this.Text = "Main Form";
        this.ResumeLayout(false);

    }

    private void projectManagementButton_Click(object sender, EventArgs e)
    {
        var projectManagementForm = new
ProjectManagementForm(_projectRepository);
        projectManagementForm.ShowDialog();
    }

    private void teamManagementButton_Click(object sender, EventArgs e)
    {
        var teamManagementForm = new TeamManagementForm(_teamRepository,
_roleRepository, _permissionRepository); // Передайте _permissionRepository
        teamManagementForm.ShowDialog();
    }

    private System.Windows.Forms.Button projectManagementButton;
    private System.Windows.Forms.Button teamManagementButton;
}

```

Додаток Д

```
this.projectEndDateDataGridViewTextBoxColumn,
this.projectBudgetDataGridViewTextBoxColumn,
this.projectStatusDataGridViewTextBoxColumn,
this.projectManagerIdDataGridViewTextBoxColumn});
this.projectsDataGridView.Location = new System.Drawing.Point(12, 12);
this.projectsDataGridView.Name = "projectsDataGridView";
this.projectsDataGridView.ReadOnly = true;
this.projectsDataGridView.Size = new System.Drawing.Size(776, 300);
this.projectsDataGridView.TabIndex = 0;
//
// projectIdDataGridViewTextBoxColumn
//
this.projectIdDataGridViewTextBoxColumn.DataPropertyName = "Id";
this.projectIdDataGridViewTextBoxColumn.HeaderText = "Id";
this.projectIdDataGridViewTextBoxColumn.Name =
"projectIdDataGridViewTextBoxColumn";
this.projectIdDataGridViewTextBoxColumn.ReadOnly = true;
//
// projectNameDataGridViewTextBoxColumn
//
this.projectNameDataGridViewTextBoxColumn.DataPropertyName =
"Name";
this.projectNameDataGridViewTextBoxColumn.HeaderText = "Name";
this.projectNameDataGridViewTextBoxColumn.Name =
"projectNameDataGridViewTextBoxColumn";
this.projectNameDataGridViewTextBoxColumn.ReadOnly = true;
//
// projectDescriptionDataGridViewTextBoxColumn
//
this.projectDescriptionDataGridViewTextBoxColumn.DataPropertyName
= "Description";
this.projectDescriptionDataGridViewTextBoxColumn.HeaderText =
"Description";
this.projectDescriptionDataGridViewTextBoxColumn.Name =
"projectDescriptionDataGridViewTextBoxColumn";
this.projectDescriptionDataGridViewTextBoxColumn.ReadOnly = true;
//
// projectStartDateDataGridViewTextBoxColumn
//
this.projectStartDateDataGridViewTextBoxColumn.DataPropertyName =
"StartDate";
```

```

        this.projectStartDateDataGridViewTextBoxColumn.HeaderText =
"StartDate";
        this.projectStartDateDataGridViewTextBoxColumn.Name =
"projectStartDateDataGridViewTextBoxColumn";
        this.projectStartDateDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // projectEndDateDataGridViewTextBoxColumn
        //
        this.projectEndDateDataGridViewTextBoxColumn.DataPropertyName =
"EndDate";
        this.projectEndDateDataGridViewTextBoxColumn.HeaderText =
"EndDate";
        this.projectEndDateDataGridViewTextBoxColumn.Name =
"projectEndDateDataGridViewTextBoxColumn";
        this.projectEndDateDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // projectBudgetDataGridViewTextBoxColumn
        //
        this.projectBudgetDataGridViewTextBoxColumn.DataPropertyName =
"Budget";
        this.projectBudgetDataGridViewTextBoxColumn.HeaderText = "Budget";
        this.projectBudgetDataGridViewTextBoxColumn.Name =
"projectBudgetDataGridViewTextBoxColumn";
        this.projectBudgetDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // projectStatusDataGridViewTextBoxColumn
        //
        this.projectStatusDataGridViewTextBoxColumn.DataPropertyName =
"Status";
        this.projectStatusDataGridViewTextBoxColumn.HeaderText = "Status";
        this.projectStatusDataGridViewTextBoxColumn.Name =
"projectStatusDataGridViewTextBoxColumn";
        this.projectStatusDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // projectManagerIdDataGridViewTextBoxColumn
        //
        this.projectManagerIdDataGridViewTextBoxColumn.DataPropertyName =
"ProjectManagerId";
        this.projectManagerIdDataGridViewTextBoxColumn.HeaderText =
"ProjectManagerId";
        this.projectManagerIdDataGridViewTextBoxColumn.Name =
"projectManagerIdDataGridViewTextBoxColumn";
        this.projectManagerIdDataGridViewTextBoxColumn.ReadOnly = true;
        //

```

```

// addButton
//
this.addButton.Location = new System.Drawing.Point(12, 329);
this.addButton.Name = "addButton";
this.addButton.Size = new System.Drawing.Size(75, 23);
this.addButton.TabIndex = 1;
this.addButton.Text = "Add";
this.addButton.UseVisualStyleBackColor = true;
this.addButton.Click += new System.EventHandler(this.addButton_Click);
//
// editButton
//
this.editButton.Location = new System.Drawing.Point(93, 329);
this.editButton.Name = "editButton";
this.editButton.Size = new System.Drawing.Size(75, 23);
this.editButton.TabIndex = 2;
this.editButton.Text = "Edit";
this.editButton.UseVisualStyleBackColor = true;
this.editButton.Click += new System.EventHandler(this.editButton_Click);
//
// deleteButton
//
this.deleteButton.Location = new System.Drawing.Point(174, 329);
this.deleteButton.Name = "deleteButton";
this.deleteButton.Size = new System.Drawing.Size(75, 23);
this.deleteButton.TabIndex = 3;
this.deleteButton.Text = "Delete";
this.deleteButton.UseVisualStyleBackColor = true;
this.deleteButton.Click += new
System.EventHandler(this.deleteButton_Click);
//
// detailsButton
//
this.detailsButton.Location = new System.Drawing.Point(255, 329);
this.detailsButton.Name = "detailsButton";
this.detailsButton.Size = new System.Drawing.Size(75, 23);
this.detailsButton.TabIndex = 4;
this.detailsButton.Text = "Details";
this.detailsButton.UseVisualStyleBackColor = true;
this.detailsButton.Click += new
System.EventHandler(this.detailsButton_Click);
//
// ProjectManagementForm
//

```

```

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(800, 375);
this.Controls.Add(this.detailsButton);
this.Controls.Add(this.deleteButton);
this.Controls.Add(this.editButton);
this.Controls.Add(this.addButton);
this.Controls.Add(this.projectsDataGridView);
this.Name = "ProjectManagementForm";
this.Text = "Project Management";

((System.ComponentModel.ISupportInitialize)(this.projectsDataGridView)).EndInit();
this.ResumeLayout(false);

}

private void LoadProjects()
{
    var projects = _projectRepository.GetAllProjects().ToList();
    projectsDataGridView.DataSource = projects;
}

private void addButton_Click(object sender, EventArgs e)
{
    // Відкрийте форму для додавання нового проекту
    var addProjectForm = new AddProjectForm(_projectRepository);
    if (addProjectForm.ShowDialog() == DialogResult.OK)
    {
        LoadProjects(); // Оновіть таблицю проектів
    }
}

private void editButton_Click(object sender, EventArgs e)
{
    if (projectsDataGridView.SelectedRows.Count > 0)
    {
        // Отримайте Id обраного проекту
        int projectId =
Convert.ToInt32(projectsDataGridView.SelectedRows[0].Cells["Id"].Value);

        // Відкрийте форму для редагування проекту
        var editProjectForm = new EditProjectForm(_projectRepository,
projectId);

```

```

        if (editProjectForm.ShowDialog() == DialogResult.OK)
        {
            LoadProjects(); // Оновіть таблицю проектів
        }
    }

private void deleteButton_Click(object sender, EventArgs e)
{
    if (projectsDataGridView.SelectedRows.Count > 0)
    {
        // Отримайте Id обраного проекту
        int projectId =
Convert.ToInt32(projectsDataGridView.SelectedRows[0].Cells["Id"].Value);

        if (MessageBox.Show("Ви впевнені, що хочете видалити цей
проект?", "Підтвердження видалення", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            _projectRepository.DeleteProject(projectId);
            LoadProjects(); // Оновіть таблицю проектів
        }
    }
}

private void detailsButton_Click(object sender, EventArgs e)
{
    if (projectsDataGridView.SelectedRows.Count > 0)
    {
        // Отримайте Id обраного проекту
        int projectId =
Convert.ToInt32(projectsDataGridView.SelectedRows[0].Cells["Id"].Value);

        // Відкрийте форму для детальної інформації про проект
        var projectDetailsForm = new ProjectDetailsForm(_projectRepository,
projectId);
        projectDetailsForm.ShowDialog();
    }
}

private System.Windows.Forms.DataGridView projectsDataGridView;
private System.Windows.Forms.DataGridViewTextBoxColumn
projectIdDataGridViewTextBoxColumn;

```

```
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectNameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectDescriptionDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectStartDateDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectEndDateDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectBudgetDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectStatusDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
projectManagerIdDataGridViewTextBoxColumn;
        private System.Windows.Forms.Button addButton;
        private System.Windows.Forms.Button editButton;
        private System.Windows.Forms.Button deleteButton;
        private System.Windows.Forms.Button detailsButton;
    }
```

Додаток Е

```
public partial class PermissionManagementForm : Form
{
    private IPermissionRepository _permissionRepository;
    private int _roleId; // Ідентифікатор ролі, до якої належать дозволи

    public PermissionManagementForm(IPermissionRepository
permissionRepository, int roleId)
    {
        InitializeComponent();
        CreateComponent();
        _permissionRepository = permissionRepository;
        _roleId = roleId;

        LoadPermissions();
    }

    private void CreateComponent()
    {
        this.permissionsDataGridView = new
System.Windows.Forms.DataGridView();
        this.permissionIdDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.permissionNameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.permissionDescriptionDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.addButton = new System.Windows.Forms.Button();
        this.editButton = new System.Windows.Forms.Button();
        this.deleteButton = new System.Windows.Forms.Button();
        this.closeButton = new System.Windows.Forms.Button();

        ((System.ComponentModel.ISupportInitialize)(this.permissionsDataGridView)).B
eginInit();
        this.SuspendLayout();
        //
        // permissionsDataGridView
        //
        this.permissionsDataGridView.AllowUserToAddRows = false;
        this.permissionsDataGridView.AllowUserToDeleteRows = false;
        this.permissionsDataGridView.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
;
    }
}
```

```

        this.permissionsDataGridView.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
        this.permissionIdDataGridViewTextBoxColumn,
        this.permissionNameDataGridViewTextBoxColumn,
        this.permissionDescriptionDataGridViewTextBoxColumn });
        this.permissionsDataGridView.Location = new System.Drawing.Point(12,
12);
        this.permissionsDataGridView.Name = "permissionsDataGridView";
        this.permissionsDataGridView.ReadOnly = true;
        this.permissionsDataGridView.Size = new System.Drawing.Size(544,
240);
        this.permissionsDataGridView.TabIndex = 0;
        //
        // permissionIdDataGridViewTextBoxColumn
        //
        this.permissionIdDataGridViewTextBoxColumn.DataPropertyName =
"Id";
        this.permissionIdDataGridViewTextBoxColumn.HeaderText = "Id";
        this.permissionIdDataGridViewTextBoxColumn.Name =
"permissionIdDataGridViewTextBoxColumn";
        this.permissionIdDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // permissionNameDataGridViewTextBoxColumn
        //
        this.permissionNameDataGridViewTextBoxColumn.DataPropertyName =
"Name";
        this.permissionNameDataGridViewTextBoxColumn.HeaderText =
"Name";
        this.permissionNameDataGridViewTextBoxColumn.Name =
"permissionNameDataGridViewTextBoxColumn";
        this.permissionNameDataGridViewTextBoxColumn.ReadOnly = true;
        //
        // permissionDescriptionDataGridViewTextBoxColumn
        //
        this.permissionDescriptionDataGridViewTextBoxColumn.DataPropertyName =
"Description";
        this.permissionDescriptionDataGridViewTextBoxColumn.HeaderText =
"Description";
        this.permissionDescriptionDataGridViewTextBoxColumn.Name =
"permissionDescriptionDataGridViewTextBoxColumn";
        this.permissionDescriptionDataGridViewTextBoxColumn.ReadOnly =
true;
        //

```

```

// addButton
//
this.addButton.Location = new System.Drawing.Point(12, 258);
this.addButton.Name = "addButton";
this.addButton.Size = new System.Drawing.Size(75, 23);
this.addButton.TabIndex = 1;
this.addButton.Text = "Add";
this.addButton.UseVisualStyleBackColor = true;
this.addButton.Click += new System.EventHandler(this.addButton_Click);
//
// editButton
//
this.editButton.Location = new System.Drawing.Point(93, 258);
this.editButton.Name = "editButton";
this.editButton.Size = new System.Drawing.Size(75, 23);
this.editButton.TabIndex = 2;
this.editButton.Text = "Edit";
this.editButton.UseVisualStyleBackColor = true;
this.editButton.Click += new System.EventHandler(this.editButton_Click);
//
// deleteButton
//
this.deleteButton.Location = new System.Drawing.Point(174, 258);
this.deleteButton.Name = "deleteButton";
this.deleteButton.Size = new System.Drawing.Size(75, 23);
this.deleteButton.TabIndex = 3;
this.deleteButton.Text = "Delete";
this.deleteButton.UseVisualStyleBackColor = true;
this.deleteButton.Click += new
System.EventHandler(this.deleteButton_Click);
//
// closeButton
//
this.closeButton.Location = new System.Drawing.Point(473, 258);
this.closeButton.Name = "closeButton";
this.closeButton.Size = new System.Drawing.Size(75, 23);
this.closeButton.TabIndex = 4;
this.closeButton.Text = "Close";
this.closeButton.UseVisualStyleBackColor = true;
this.closeButton.Click += new
System.EventHandler(this.closeButton_Click);
//
// PermissionManagementForm
//

```

```

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(568, 293);
this.Controls.Add(this.closeButton);
this.Controls.Add(this.deleteButton);
this.Controls.Add(this.editButton);
this.Controls.Add(this.addButton);
this.Controls.Add(this.permissionsDataGridView);
this.Name = "PermissionManagementForm";
this.Text = "Permission Management";

((System.ComponentModel.ISupportInitialize)(this.permissionsDataGridView)).EndInit();
    this.ResumeLayout(false);

}

private void LoadPermissions()
{
    // Отримати список дозволів з репозиторію
    var permissions = _permissionRepository.GetAllPermissions().ToList();
    permissionsDataGridView.DataSource = permissions;
}

private void addButton_Click(object sender, EventArgs e)
{
    // Відкрийте форму для додавання нового дозволу
    var addPermissionForm = new
AddPermissionForm(_permissionRepository, _roleId); // Додайте
AddPermissionForm
    if (addPermissionForm.ShowDialog() == DialogResult.OK)
    {
        LoadPermissions(); // Оновіть таблицю дозволів
    }
}

private void editButton_Click(object sender, EventArgs e)
{
    if (permissionsDataGridView.SelectedRows.Count > 0)
    {
        // Отримайте Id обраного дозволу
        int permissionId =
Convert.ToInt32(permissionsDataGridView.SelectedRows[0].Cells["Id"].Value);

```

```

        // Відкрийте форму для редагування дозволу
        var editPermissionForm = new
EditPermissionForm(_permissionRepository, permissionId); // Додайте
EditPermissionForm
        if (editPermissionForm.ShowDialog() == DialogResult.OK)
        {
            LoadPermissions(); // Оновіть таблицю дозволів
        }
    }
}

private void deleteButton_Click(object sender, EventArgs e)
{
    if (permissionsDataGridView.SelectedRows.Count > 0)
    {
        // Отримайте Id обраного дозволу
        int permissionId =
Convert.ToInt32(permissionsDataGridView.SelectedRows[0].Cells["Id"].Value);

        if (MessageBox.Show("Ви впевнені, що хочете видалити цей
дозвіл?", "Підтвердження видалення", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            _permissionRepository.DeletePermission(permissionId);
            LoadPermissions(); // Оновіть таблицю дозволів
        }
    }
}

private void closeButton_Click(object sender, EventArgs e)
{
    this.Close();
}

private System.Windows.Forms.DataGridView permissionsDataGridView;
private System.Windows.Forms.DataGridViewTextBoxColumn
permissionIdDataGridViewTextBoxColumn;
private System.Windows.Forms.DataGridViewTextBoxColumn
permissionNameDataGridViewTextBoxColumn;
private System.Windows.Forms.DataGridViewTextBoxColumn
permissionDescriptionDataGridViewTextBoxColumn;
private System.Windows.Forms.Button addButton;
private System.Windows.Forms.Button editButton;
private System.Windows.Forms.Button deleteButton;
private System.Windows.Forms.Button closeButton; }

```