

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ

PROJECT MANAGEMENT SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-132

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ТРОЦЕНКО Н.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ТРОЦЕНКО Наталії Анатоліївни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для керування проєктами.
Project Management Software

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 20 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Наталія ТРОЦЕНКО
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Андрій ОЛІЙНИК
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
94 с., 4 табл., 32 рис., 3 дод., 19 джерел.

C#, VISUAL STUDIO, КЕРУВАННЯ ПРОЄКТАМИ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для керування проєктами.

Предмет дослідження – програмні засоби для керування проєктами.

Мета роботи – розробка програмного забезпечення для керування проєктами та підвищення рівня організаційної ефективності.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio, система управління базами даних MongoDB.

Результати. Проаналізовано процес керування проєктами та відповідне програмне забезпечення. Виконано проєктування програмного забезпечення для керування проєктами. Створено програмне забезпечення для керування проєктами.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для керування проєктами за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – керування проєктами у різних організаціях.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 4 tables, 32 figures, 3 appendixes, 19 sources.

C#, VISUAL STUDIO, PROJECT MANAGEMENT, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing project management software.

The subject of the study is project management software.

The purpose of the work is to develop project management software.

Materials, methods and technical means: C# programming language, Visual Studio development environment, MongoDB database management system.

Results. The project management process and relevant software were analyzed. Project management software was designed. Project management software was created.

Conclusions. The purpose of the work was achieved. Project management software was developed using the C# programming language and the Visual Studio development environment.

Field of application: project management in various organizations.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для керування проєктами	11
1.2 Аналіз програмного забезпечення для керування проєктами.....	21
1.3 Висновки за розділом 1	32
2 Матеріали і методи	34
2.1 Вибір мови програмування	34
2.2 Вибір середовища розробки для створення програмного забезпечення для керування проєктами.....	37
2.3 Вибір системи управління базами даних для створення програмного забезпечення для керування проєктами.....	40
2.4 Висновки за розділом 2	42
3 Опис програми	44
3.1 Структура програмного забезпечення для керування проєктами	44
3.2 Функціонування програмного забезпечення для керування проєктами.	47
3.3 Розробка бази даних програмного забезпечення для керування проєктами.....	49
3.4 Проєктування інтерфейсу програмного забезпечення для керування проєктами.....	51
3.5 Висновки за розділом 3	54
4 Експлуатація, тестування та експериментальне дослідження програми.....	55
4.1 Призначення й умови застосування програми	55
4.2 Характеристики програми для керування проєктами.....	56
4.3 Інструкція по експлуатації програми.....	57
4.3.1 Звернення до програми.....	57
4.3.2 Вхідні й вихідні дані.....	57

4.3.3 Повідомлення	58
4.4 Виконання програмного забезпечення для керування проєктами	58
4.5 Тестування програмного забезпечення для керування проєктами.....	63
4.6 Висновки за розділом 4	64
Висновки	65
Перелік джерел посилання	68
Додаток А Технічне завдання	70
Додаток Б Фрагмент тексту програми	75
Додаток В Слайди презентації	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

DB – Data Base;

PM – Project Management;

БД – база даних;

КП – керування проектами;

ПЗ – програмне забезпечення.

ВСТУП

Програмні системи для підтримки управління проектами сьогодні відіграють ключову роль у діяльності багатьох компаній, оскільки темп бізнесових змін лише прискорюється. Зростання кількості віддалених команд, складніша структура проектів та високі вимоги до координації завдань потребують програмних інструментів, здатних поєднати в одному середовищі планування, моніторинг, комунікацію та роботу з інформацією. Такі програмні рішення створюють прозоре середовище взаємодії, де всі учасники бачать як власні зобов'язання, так і загальну картину виконання робіт, що мінімізує ризики непорозумінь [1]-[2].

Попит на подібне програмне забезпечення підтримується бажанням організацій точніше контролювати витрати часу й коштів, а також швидше реагувати на непередбачені обставини. Функції прогнозування, автоматичного оповіщення, аналітики та управління ризиками дають змогу виявляти проблеми ще до того, як вони вплинуть на кінцевий результат. Це сприяє раціональному використанню ресурсів і стабільнішому досягненню поставлених цілей [3]-[4].

Розвиток хмарних сервісів зробив такі системи доступними з будь-якого пристрою, дозволяючи командам ефективно працювати незалежно від місця перебування. Використання інтелектуальних алгоритмів у сучасних інструментах відкриває можливості автоматичного формування рекомендацій, адаптивного планування та глибокої аналітики, що зміцнює управлінські рішення та допомагає синхронізувати проектні завдання зі стратегічними планами бізнесу [5]-[6].

Отже, актуальність програмного забезпечення для керування проектами полягає у його здатності підтримувати конкурентоспроможність організацій, підвищувати якість реалізації будь-яких ініціатив та забезпечувати гнучкість у мінливому середовищі [1]-[6].

Попри численні переваги, впровадження таких програмних рішень може супроводжуватися труднощами, пов'язаними з необхідністю навчання працівників, налагодженням інтеграцій з існуючими внутрішніми програмними системами та забезпеченням належного рівня кібербезпеки. Фінансовий аспект також має вагоме значення. Витрати на розгортання, передплату, техпідтримку та навчання користувачів можуть бути відчутними, особливо для невеликих організацій або стартапів. Іноді економічний ефект проявляється не одразу, що створює сумніви в доцільності інвестицій [1]-[6].

Тому актуальною є розробка програмного забезпечення для керування проєктами. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для керування проєктами.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для керування проєктами;
- здійснити проєктування програмного забезпечення для керування проєктами;
- створити програмне забезпечення для керування проєктами;
- виконати тестування розробленого програмного забезпечення для керування проєктами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для керування проєктами

Управління програмними проєктами розглядається як комплекс дій, спрямованих на впорядкування процесу створення програмного забезпечення. Передбачається систематичне планування, координація робіт і контроль виконання, щоб кінцевий результат відповідав очікуванням, був реалізований у встановлені терміни та не виходив за межі виділеного бюджету. У процесі управління приділяється увага визначенню меж проєкту, оцінці необхідних ресурсів, формуванню завдань і перевірці фактичного стану виконання робіт. Основним орієнтиром виступає отримання програмного продукту належної якості, який задовольнятиме користувацькі потреби [1]-[2].

Актуальність такої діяльності пояснюється тим, що програмне забезпечення не має матеріальної форми, а галузь його розробки все ще формується та постійно розвивається. Понад те, створення ПЗ нерідко відбувається під індивідуальні запити замовника, що ускладнює можливість використання попереднього досвіду. Технології швидко змінюються, тому навіть успішні підходи, застосовані під час одного проєкту, можуть виявитися неефективними в іншому контексті. Унаслідок цього ризики значно зростають, що зумовлює потребу в належному управлінні для забезпечення стабільної якості, дотримання термінів і фінансових рамок [1]-[2].

Програмне забезпечення для керування проєктами допомагає організовувати, планувати, контролювати та координувати виконання проєктів. Такі програмні засоби надають можливість розподіляти завдання між учасниками, стежити за виконанням роботи в реальному часі, контролювати витрати та дотримання термінів, а також підтримувати комунікацію в команді. Основна мета подібних систем – забезпечити прозорість усіх процесів, підвищити ефективність роботи, зменшити ризики і допомогти завершити проєкт успішно, якісно і вчасно. Такі програмні засоби застосовуються в

різних сферах, зокрема в ІТ, будівництві, маркетингу, освіті та будь-яких інших галузях, де діяльність організована у вигляді проєктів [1]-[2].

Система управління програмними проєктами передбачає врахування вимог замовника, а також дотримання всіх обмежень з боку бюджету та часу. У її межах реалізується низка напрямів, що охоплюють основні організаційні та контрольні процеси [1]-[6].

Конфліктні ситуації в команді піддаються регулюванню, щоб мінімізувати негативні наслідки суперечностей і водночас використати їх як чинник розвитку колективу та підвищення продуктивності. Ризики виявляються та аналізуються завчасно, що дає змогу оптимально розподіляти ресурси для запобігання небажаним подіям або пом'якшення їх впливу [2]-[3].

Особливе значення приділяється роботі з вимогами: вони документуються, переглядаються, коригуються та постійно узгоджуються з усіма сторонами, зацікавленими в результаті. Усі зміни, що відбуваються в ході проєкту, підлягають системному управлінню, аби перехід до нових умов та процесів був максимально контрольованим і зрозумілим для учасників. Конфігурація програмного забезпечення зберігається під строгим контролем: фіксуються всі модифікації, відстежуються версії та встановлюються контрольні точки, які забезпечують стабільність розроблюваної системи. Робота над випусками кінцевого продукту передбачає чітке планування і впровадження нових функцій та покращень без порушення працездатності вже доступного функціоналу [2]-[3].

Таким чином, управління програмними проєктами виступає важливою складовою успішного створення ПЗ, забезпечуючи упорядкованість процесів, зниження ризиків та досягнення намічених технічних і бізнес-цілей [2]-[3].

Аспекти управління програмними проєктами охоплюють сукупність практик і процесів, що дозволяють контролювати весь життєвий цикл створення програмного забезпечення. Вони забезпечують відповідність робіт попередньо визначеній концепції, вимогам кінцевих користувачів і заданим межах часу та фінансів. За допомогою ретельно підготовленого плану

визначається майбутній напрям розробки, уточнюється зміст проєкту, формується система взаємодії між учасниками та встановлюються правила тестування й обслуговування програмного продукту [2]-[3].

Організація спільної діяльності виконавців передбачає координацію різнопрофільних спеціалістів, а також забезпечення ефективної комунікації всередині команди. Від керівної ролі залежить послідовність дій під час реалізації технічних завдань і злагоджене проходження міжетапних переходів. Контроль виконання допомагає своєчасно відстежувати прогрес, оцінювати рівень відповідності запланованим результатам і вносити коригування за необхідності [2]-[3].

Збереження темпу роботи є одним із критичних чинників, оскільки несподівані зміни в умовах розробки виникають доволі часто. У зв'язку з цим важливо завчасно враховувати можливі ризики та бути готовими оперативно реагувати на ускладнення, щоб запобігти зриву встановлених строків. Також значущим аспектом є раціональний розподіл фінансових ресурсів, що передбачає постійний моніторинг витрат і за потреби – перегляд кошторису. Тривала підтримка програмного рішення забезпечує стабільність і якість продукту, дозволяючи виявляти помилки на ранніх стадіях і адаптувати результат до оновлених вимог замовника. Постійний аналіз і вдосконалення створюють умови для успішного завершення проєкту та гарантують відповідність інтересам усіх сторін [2]-[3].

Разом із перевагами існує низка ризиків, пов'язаних із застосуванням управлінських методологій у галузі розробки ПЗ. Варто враховувати, що створення та підтримка необхідної інфраструктури може бути фінансово затратним процесом, особливо якщо впроваджуються спеціалізовані інструменти та зовнішня експертиза. Крім того, надмірне ускладнення процедур чи структур може спричинити втручання в робочий ритм команди й погіршити продуктивність [2]-[3].

Коли зростає кількість учасників, комунікація стає більш навантаженою, що іноді призводить до втрати чіткості в передаванні

інформації. Також надмірний тиск правил і стандартизованих підходів здатний обмежувати творчий пошук розробників та стримувати появу оригінальних ідей, корисних для розвитку продукту [2]-[3].

Таким чином, управління програмними проєктами потребує балансу між структурованістю та гнучкістю: ретельна організація процесу є необхідною, але не повинна перешкоджати інноваційності, швидкому реагуванню та командній динаміці [2]-[3].

Програмні рішення для керування проєктами створюються для того, щоб покращувати організацію та координацію роботи як для окремих виконавців, так і для команд. Завдяки таким платформам усі дані, пов'язані з проєктом, зосереджуються в єдиному середовищі, що забезпечує зручний доступ до матеріалів, бачення стану виконання та можливість взаємодії між учасниками. Ці системи легко адаптуються під різні масштаби діяльності та професійні сфери [3]-[4].

Застосування подібних інструментів допомагає уникати зайвої плутанини, зменшує ймовірність затримок і сприяє чіткому розподілу відповідальності. Робочі процеси стають зрозумілішими, а контроль за виконанням завдань ефективнішим, що зрештою дозволяє досягати запланованих результатів у встановлені строки та без зайвих витрат. Усі дії, пов'язані з документами чи комунікацією, виконуються в єдиному середовищі, без перемикання між різними застосунками [3]-[4].

Подібні системи стануть у пригоді будь-кому, хто працює над проєктами: від менеджерів і невеликих груп спеціалістів до великих організацій із багаторівневими процесами. Вони здатні підтримувати як прості щоденні завдання, так і комплексні ініціативи з багатьма учасниками та змінними [3]-[4].

Незважаючи на спільні принципи, платформи можуть орієнтуватися на різні методики керування, зокрема на підходи Agile, Scrum чи класичні каскадні моделі, або бути спеціалізованими для конкретних видів діяльності.

Деякі сервіси розроблені переважно для ІТ-середовища, інші залишаються універсальними та підходять для багатьох сфер [3]-[4].

Найпоширеніші можливості таких рішень пов'язані з постановкою та супроводом завдань, взаємодією в межах платформи, зберіганням і переданням файлів, формуванням звітності та візуалізацією даних щодо проєкту. Часто додаються засоби автоматизації та під'єднання зовнішніх сервісів, що робить робочий процес гнучкішим і зручнішим. Завдяки такому набору функцій програмне забезпечення для керування проєктами сприяє підвищенню ефективності роботи, прозорості процесів і стабільному досягненню поставлених цілей [3]-[4].

Програмні платформи, створені для організації проєктної діяльності, спрямовані на формування єдиного середовища, у якому всі дані щодо завдань, комунікації та етапів виконання залишаються упорядкованими й легко доступними. Завдяки такій централізації усувається хаотичність під час роботи та підсилюється узгодженість усіх учасників процесу [3]-[4].

Спільна взаємодія команди набуває нового формату: кожен учасник має можливість оперативно долучатися до обговорення, не враховуючи географічні бар'єри чи відмінності в графіках. Будь-які коментарі чи уточнення залишаються прив'язаними до конкретного завдання, що полегшує пошук потрібної інформації та сприяє прозорості спілкування [3]-[4].

Відстеження актуального стану роботи здійснюється без складнощів: усі зміни оновлюються автоматично та відображаються у різних візуальних форматах, що допомагає швидко оцінювати ситуацію та попереджувати можливі затримки. Наявність таких інструментів, як інтерактивні панелі керування, суттєво допомагає керівникам своєчасно виявляти проблемні аспекти [3]-[4].

Правильний розподіл навантаження стає більш передбачуваним: інформація щодо зайнятості та відповідальності кожного фахівця не втрачається й дозволяє уникати нерівномірного завантаження. Це також дає

можливість оптимізувати процес планування та підсилює раціональне використання потенціалу команди [3]-[4].

Підзвітність у такому середовищі підвищується природним чином: кожен бачить власні обов'язки, розуміє строки реалізації й отримує нагадування, які не дають відхилитися від наміченого плану. Відкритість даних зводить до мінімуму ризики непорозумінь і допомагає керівникам тримати під контролем динаміку виконання задач [3]-[4].

Завдяки широкому функціоналу такі програмні системи здатні заміщувати інші робочі інструменти, оскільки дозволяють налаштовувати процеси під конкретні вимоги, створювати шаблони та автоматизувати типові операції. Універсальність дає можливість інтегрувати численні аспекти діяльності в єдину платформу та зменшити кількість сторонніх додатків, необхідних у щоденній роботі [4]-[5].

Попри значні переваги, може виникати низка труднощів, пов'язаних із адаптацією, потребою в навчанні персоналу та необхідністю оптимального налаштування системи під конкретні умови використання. Тому обираючи платформу, важливо зважати на особливості власного середовища, комфортність інтерфейсу та придатність для розширення [4]-[5].

Ефективність упровадження залежить від правильного визначення того, які можливості дійсно потрібні у роботі. Під час вибору доцільно оцінювати зручність використання, сумісність з іншими інструментами, відповідність бюджету, а також можливість працювати віддалено. Успішний результат досягається тоді, коли програмне рішення органічно вписується в існуючі процеси та покращує їх без ускладнення [4]-[5].

Зрештою, управління проектами стає більш передбачуваним, структурованим і прозорим, якщо використовувати інструмент, який відповідає специфіці завдань, забезпечує візуальну наочність та гнучке налаштування під потреби команди. Саме правильний вибір платформи стає основою підвищення якості роботи й ефективного досягнення поставлених цілей [4]-[5].

Програмне забезпечення, призначене для проєктної діяльності, являє собою цифрове середовище, у якому створюються умови для впорядкування всіх процесів – від задуму до фактичного завершення роботи. У межах такого простору забезпечується координація дій команди, контроль ходу виконання кожного етапу, підтримка прозорих комунікацій і можливість раціонального застосування наявних ресурсів, що сприяє дотриманню встановлених рамок і цілей [4]-[5].

Сучасні системи управління проєктами вже давно перестали бути простими засобами обліку завдань. Вони перетворилися на універсальні комплекси, котрі поєднують у собі можливості планування, моніторингу показників, ведення аналітики, збереження й обміну документами, а також спрощують взаємодію між усіма учасниками. Завдяки цьому забезпечується повна візуалізація робочого процесу, а відповідність дій команди стратегічним намірам організації підтримується на кожному кроці [4]-[5].

Основний сенс використання таких платформ полягає у створенні впорядкованого робочого середовища, де не виникає плутанини через розрізнені повідомлення чи нечітко визначені обов'язки. За відсутності відповідних технологій зростають ризики зривів строків, втрачених даних та непорозумінь у відповідальності. Комплексні рішення усувають ці бар'єри, формуючи стабільну систему контролю, прозорості й відповідальності в межах усіх стадій виконання проєкту [4]-[5].

Таке програмне забезпечення придатне для широкого спектра сценаріїв – від гнучкого управління цифровими продуктами до масштабних операцій у технічних галузях. Воно може адаптуватися під різні методології організації роботи, включаючи поступове лінійне планування, ітеративний підхід і візуальне представлення робочих процесів. Подібні системи підходять і для компактних команд зі скромним набором задач, і для великих структур, у яких координуються численні учасники з різних підрозділів та регіонів [4]-[5].

Існують різновиди таких інструментів, кожен з яких зорієнтовано на власний формат діяльності. В одних перевага надається детальному

плануванню із застосуванням методів точного графічного моделювання, що підходить для галузей із жорсткими вимогами до документації. Інші системи створені для адаптивного управління, у якому швидка реакція на зміни середовища має вирішальне значення. Також є рішення, де основний наголос зроблено на візуальній організації робочого процесу, що дозволяє відразу помітити проблемні точки й оптимізувати послідовність роботи [4]-[5].

Правильний вибір програмного інструмента залежить від специфіки діяльності, стилю команди та очікуваних результатів. Саме відповідність характеристик системи реальним потребам визначає успіх її застосування та ефективність проєктного управління в цілому [4]-[5].

Сучасні комплексні середовища для командної взаємодії орієнтовані на створення єдиного інформаційного простору, у якому поєднуються функції планування проєктів, засоби спілкування, обмін матеріалами й автоматизація процесів. Завдяки такому об'єднанню усувається необхідність користування різними несумісними сервісами, зменшується кількість відволікань та забезпечується безперервний рух інформації між учасниками [4]-[5].

Окрема група інструментів створюється спеціально для певних професійних сфер. Вони налаштовуються відповідно до нормативних вимог, характерних робочих процесів і специфіки діяльності тієї чи іншої галузі, будь то медицина, маркетинг, будівництво або креативні послуги. Такі системи зазвичай містять попередньо визначені шаблони, налаштовані механізми контролю дотримання стандартів, спеціалізовані звіти та інші можливості, орієнтовані на конкретні сценарії використання [4]-[5].

Для великих корпорацій, де здійснюється одночасний контроль численних ініціатив, використовуються рішення, здатні забезпечувати централізоване бачення всієї сукупності проєктів, а також керувати ресурсами, бюджетами й пріоритетами на більш високому рівні [4]-[5].

Гнучкі системи поєднують кілька підходів у межах однієї платформи, що робить їх придатними для організацій, які застосовують різні методики управління або перебувають у процесі трансформації своїх процесів. Завдяки

такій універсальності стає можливим узгодити роботу як у межах чітко структурованого планування, так і в умовах змінних вимог і динамічних циклів [4]-[5].

Кожен вид таких інструментів розрахований на свою сферу застосування, різні масштаби команд і рівень складності процесів, тому коректний вибір напряму впливає на успішність діяльності та здатність організації підтримувати ефективність у довгостроковій перспективі [4]-[5].

Під час оцінювання можливих варіантів ключову роль відіграє наявність широкого набору інструментів для впорядкування роботи над завданнями й забезпечення організованості всієї структури проекту. Здатність розділяти складні ініціативи на окремі етапи з чітким описом, залежностями й часовими межами допомагає уникати хаотичності та зберігати контроль над динамікою виконання. Додаткові можливості автоматизації рутинних операцій та використання готових моделей сприяють оптимізації робочого часу [5]-[6].

Не менш важливо, щоб система надавала різні способи представлення даних: від спискового до візуального, включно з календарями та часовими шкалами. Гнучке відображення інформації дає змогу враховувати різні підходи до організації роботи та сприйняття даних, а також забезпечує повноцінний огляд стану справ на будь-якому рівні деталізації [5]-[6].

Сучасне цифрове робоче середовище потребує зручних механізмів взаємодії, тому в засобах керування проектами особлива увага приділяється підтримці командної співпраці. Використання вбудованих каналів для обміну повідомленнями, гнучких інструментів для коментування матеріалів і механізмів позначення учасників у повідомленнях сприяє оперативному інформуванню та уникненню інформаційного перевантаження. Синхронне редагування, збереження змін і спостереження за перебігом роботи в один момент часу забезпечують узгодженість результатів. Крім того, можливість під'єднання зовнішніх служб комунікації дозволяє зручно поєднувати управління процесами з повсякденними каналами спілкування [5]-[6].

Програми управління проєктами пропонують інструменти, що допомагають розумно розподіляти навантаження між виконавцями та своєчасно реагувати на зміни в обсязі робіт. Функції обліку витраченого часу та аналізу ефективності сприяють коректному формуванню вартості робіт і дають можливість підтримувати продуктивність на належному рівні. Інформація про зайнятість команди допомагає запобігати перенавантаженості учасників і забезпечує збалансованість планів. Контроль витрат у співставленні з попередніми розрахунками створює умови для подальшого коригування бюджету, а прогнозування доступних ресурсів полегшує ухвалення рішень щодо майбутніх етапів [5]-[6].

Системи звітності й аналітики надають огляд загального стану робіт, відстежують наближення до поставлених цілей та висвітлюють чинники, що впливають на результативність. Гнучкі аналітичні панелі дають змогу адаптувати подання інформації до потреб конкретних учасників, а деталізація метрик допомагає виявляти проблемні місця до того, як вони почнуть впливати на кінцевий підсумок проєкту. Деякі рішення навіть передбачають прогнозування можливих труднощів, що дозволяє завчасно планувати дії. Можливість налагоджувати взаємодію з іншими програмами забезпечує безперервність робочих процесів і зменшує потребу в дублюванні даних. Синхронізація з поштовими сервісами, календарями, сховищами файлів та іншими корпоративними інструментами надає змогу створити цілісну систему управління діяльністю. Доступ до інтерфейсів програмування розширює потенціал для подальшої персоналізації [5]-[6].

Підтримка роботи на мобільних пристроях робить учасників процесу доступними для виконання доручень у будь-якій ситуації, навіть без можливості скористатися комп'ютером. Це особливо важливо для команд, що працюють у гнучких умовах або здійснюють частину завдань поза офісом. Системи захисту, що включають управління правами доступу, шифрування інформації та журналювання дій, сприяють збереженню конфіденційності даних. Для організацій з підвищеними вимогами до безпеки передбачаються

розширені механізми контролю доступу й альтернативні варіанти розміщення сервісів, що забезпечують повніше управління інформаційною безпекою [5]-[6].

Визначено, що програмне забезпечення для керування проєктами допомагає організовувати, планувати, контролювати та координувати виконання проєктів. Такі програмні засоби надають можливість розподіляти завдання між учасниками, стежити за виконанням роботи в реальному часі, контролювати витрати та дотримання термінів, а також підтримувати комунікацію в команді. Основна мета подібних систем – забезпечити прозорість усіх процесів, підвищити ефективність роботи, зменшити ризики і допомогти завершити проєкт успішно, якісно і вчасно. Такі програмні засоби застосовуються в різних сферах, зокрема в ІТ, будівництві, маркетингу, освіті та будь-яких інших галузях, де діяльність організована у вигляді проєктів.

1.2 Аналіз програмного забезпечення для керування проєктами

Проаналізуємо програмне забезпечення для керування проєктами [7]-[13].

Програмне забезпечення Wrike (рис. 1.1) є хмарним рішенням для керування проєктами, яке орієнтоване на підтримку класичних підходів до планування та контролю виконання робіт. У системі передбачене перегляд завдань у вигляді структурованих таблиць, календарного планування, списків і діаграми, що дозволяють контролювати часові залежності між етапами розробки. Для команд, що застосовують візуальне відображення робочого процесу, передбачені дошки типу Kanban, які спрощують розподіл завдань і забезпечують наочне відстеження їхнього стану. Водночас робота за методологією Scrum не належить до сильних сторін цього сервісу, оскільки відсутні ключові сутності та механізми, характерні для гнучкої розробки продуктів, тому командам, що працюють у такому форматі, зазвичай рекомендується звертатися до інших платформ [7], [8], [11].



Project management platform to streamline all workflows

Enter your business email

Try for free

or  Use Google account

 No credit card, no commitment, no downloads.

★★★★★ based on 13,000+ reviews from     

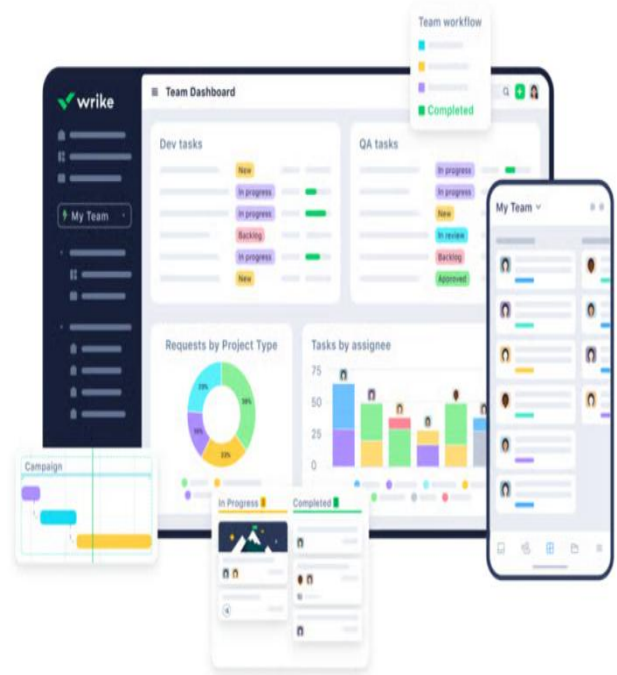


Рисунок 1.1 – Програмне забезпечення Wrike [11]

Архітектура програмного забезпечення Wrike ґрунтується на використанні трьох основних структурних елементів: папок, проєктів і завдань, які можуть бути впорядковані у будь-які вкладені структури. Це забезпечує гнучке формування ієрархій і дає змогу створювати логічні зони для колективної роботи. Кожен етап управління – від планування й визначення пріоритетів до перегляду виконання та коригування процесів – підтримується онлайн інструментами взаємодії, що забезпечують командну роботу в реальному часі [7], [9], [11].

Програмне забезпечення Wrike зберігає строгу та лаконічну стилістику інтерфейсу, орієнтуючись на функціональність. Взаємодія з ним не передбачає декоративних елементів чи ігрових механізмів, а зосереджена на структурованому відображенні даних. Особливо виділяється зручна система пошуку, яка одразу надає доступ до потрібних даних. Широкий набір можливостей спричиняє потребу в орієнтації та навчанні, оскільки опанування усіх інструментів може зайняти певний час [8], [10]-[11].

Модель користування заснована на підписці, де набір доступних функцій залежить від обраного рівня сервісу. Існує можливість ознайомлення з програмою без оплати, проте такий режим має обмежений функціонал і малий обсяг доступного сховища. Розширені можливості програмного забезпечення Wrike, підтримка управління ресурсами й часом, підсилені методами контролю доступу та безпеки, доступні лише в дорогих пакетах і тарифах найвищого рівня [7], [8], [11].

Wrike зберігає дані європейських користувачів на серверах, розташованих у межах Європейського економічного простору, що забезпечує формальну відповідність регуляторним вимогам щодо захисту персональної інформації. Однак специфіка законодавства країни походження постачальника створює певні суперечності щодо повної гарантії недоторканності даних. Підтримка користувачів містить доступ до довідкових матеріалів, інтерактивного чату та форми зворотного зв'язку, проте можливість швидкого спілкування з фахівцем у реальному часі не передбачена [7], [8], [11].

До функціональності Wrike входять засоби автоматизації на основі штучного інтелекту, спрямовані на полегшення рутинних операцій і підтримку рішення управлінських задач. Серед таких можливостей – створення текстових матеріалів прямо в системі, прогнозування ризиків залежно від поточного прогресу та формування завдань із нотаток, зроблених під час обговорення. У мобільному середовищі підтримуються голосові команди, а також автоматичне підсумовування робочого контенту. При цьому більшість інтелектуальних інструментів доступні лише в межах повнофункціональних тарифних пакетів [7], [8], [11].

Таким чином, Wrike виступає як комплексне рішення для управління проектами у компаніях, де домінує традиційний підхід до планування та контролю, з акцентом на чіткість інтерфейсу, розвинену співпрацю та розширювані можливості автоматизації, проте з помітними обмеженнями для повноцінного використання гнучких методологій [7], [8], [11].

Основними характеристиками програмного забезпечення Wrike є такі [7], [8], [11]:

- різні режими відображення завдань: програмне забезпечення Wrike підтримує таблиці, списки, календарі та діаграми для відстеження залежностей і термінів [7], [8], [11];

- Kanban-дошки: програмне забезпечення Wrike використовує візуальне управління станами завдань у вигляді карток на дошках [7], [8], [11];

- обмежена підтримка Scrum: відсутні повноцінні інструменти для гнучкої розробки у форматі спринтів [7], [8], [11];

- гнучка ієрархія об'єктів: папки, проекти та завдання можуть вкладатися одне в одного для логічної організації робіт [7], [8], [11];

- командна взаємодія онлайн: програмне забезпечення Wrike надає можливість узгодження, коментування та спільне редагування інформації в реальному часі [7], [8], [11];

- мінімалістичний інтерфейс: у програмному забезпеченні Wrike акцент робиться на читабельному й функціональному відображенні даних без зайвих декоративних елементів [7], [8], [11];

- зручний пошук: швидке знаходження завдань і проектів за ключовими параметрами та атрибутами [7], [8], [11];

- потреба у навчанні користувачів: велика кількість функцій програмного забезпечення Wrike вимагає часу на опанування інструментів системи [7], [8], [11];

- модель підписки: функціональні можливості та обсяг сховища програмного забезпечення Wrike залежать від обраного тарифу [7], [8], [11];

- доступ до базових можливостей безкоштовно: у програмному забезпеченні Wrike наявний ознайомлювальний режим з обмеженим функціоналом та ресурсами [7], [8], [11];

- підтримка безпеки даних: інформація користувачів розміщується у межах відповідно до нормативних вимог [7], [8], [11];

- автоматизація на базі штучного інтелекту: доступні інструменти для створення контенту, прогнозування ризиків та перетворення нотаток у завдання [7], [8], [11];
- мобільні можливості: програмне забезпечення Wrike підтримує голосові команди та автоматичне створення підсумків [7], [8], [11];
- обмежений доступ до інтелектуальних функцій: більшість інтелектуальних можливостей надається лише в повнофункціональних тарифах програмного забезпечення Wrike [7], [8], [11].

Серед недоліків програмного забезпечення Wrike можна відзначити підвищену складність для нових користувачів через значний набір інструментів і налаштувань, що ускладнює первинне освоєння. Крім того, повноцінне використання більшості можливостей вимагає підключення до розширених тарифних планів, оскільки базовий доступ суттєво обмежений. У командній роботі можуть виникати труднощі з навантаженням інтерфейсу, коли у системі одночасно працює багато користувачів або проєкт має надмірну кількість елементів. Деякі інтелектуальні функції, орієнтовані на автоматизацію, доступні лише для корпоративних користувачів, що створює нерівність у доступі до інструментів підвищення продуктивності. Загалом, незважаючи на широкі можливості, система вимагає часу на налаштування й оптимізацію під потреби конкретної команди [7], [8], [11].

Програмне забезпечення BigTime (рис. 1.2) являє собою комплексне рішення для організації професійних послуг, у якому зосереджено всі ключові компоненти для роботи з проєктами та обліком діяльності команд. Система забезпечує зручне зберігання і впорядкування даних, що дозволяє контролювати перебіг робіт, фінансові показники та ефективність використання ресурсів. Доступні інструменти програмного забезпечення BigTime дають змогу формувати бюджети, планувати строки, відстежувати витрати та аналізувати прибутковість, сприяючи прозорості процесів і стабільності результатів [8], [12].

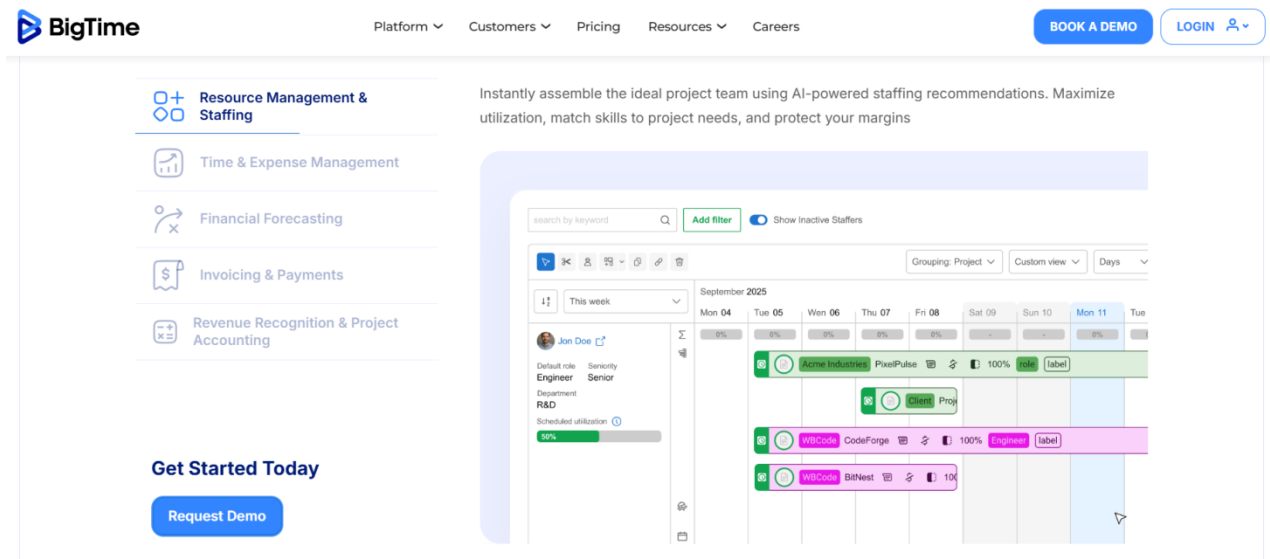


Рисунок 1.2 – Програмне забезпечення BigTime [12]

У програмі передбачено функціональність для швидкого внесення даних про робочі години та витрати, що спрощує нарахування оплати та взаємодію з бухгалтерськими системами. Інтеграція з популярними сервісами фінансового обліку допомагає уникнути повторного введення інформації та мінімізує ризик помилок. Завдяки функціям контролю доступу та відповідності встановленим вимогам програмне забезпечення може використовуватися у великих компаніях і організаціях, що реалізують державні контракти [8], [12].

Управління робочими процесами у програмному забезпеченні BigTime відбувається за допомогою інструментів планування, відображення етапів виконання і формування звітів, що дозволяє керівникам оперативно приймати рішення на основі аналітики. Візуалізація строків і навантаження допомагає краще розподіляти персонал і своєчасно виявляти можливі затримки. Разом із цим BigTime підтримує налаштовувані механізми перевірки дій та деталізовані журнали аудиту [8], [12].

Програмне забезпечення BigTime розрахована на організації, у яких значну роль відіграє облік часу та взаєморозрахунки з клієнтами. Таке програмне забезпечення активно впроваджується у сферах консалтингу, архітектурного та інженерного проектування, послуг в галузі інформаційних

технологій та бухгалтерського супроводу. Командам доступні інструменти, що допомагають орієнтуватися не лише у фінансових показниках, а й у продуктивності працівників, завдяки чому знижується рівень невизначеності у плануванні [8], [12].

Разом із перевагами існують і особливості, на які слід зважати: новим користувачам може знадобитися час на опанування інтерфейсу та функціональності, а повноцінні можливості управління фінансами передбачені лише у розширених тарифних пакунках. Водночас постійний розвиток, що включає впровадження інструментів штучного інтелекту для автоматизації аналізу та планування, підвищує привабливість рішення для компаній, які прагнуть оптимізувати бізнес-процеси [8], [12].

Програмне забезпечення BigTime має такі особливості [8], [12]:

- централізація даних про проєкти: програмне забезпечення BigTime надає єдине місце для керування інформацією про бюджети, строки, витрати й виконання робіт [8], [12];
- підтримка контролю часу: у програмному забезпеченні BigTime доступні інструменти для фіксації робочих годин і витрат з подальшим застосуванням у розрахунках з клієнтами [8], [12];
- інтеграція з бухгалтерськими системами: у програмному забезпеченні BigTime передбачено обмін даними з фінансовими сервісами для коректного виставлення рахунків і зниження помилок [8], [12];
- відповідність вимогам аудиту: у програмному забезпеченні BigTime підтримуються журнали змін і механізми контролю, корисні для компаній із підвищеними стандартами безпеки [8], [12];
- інструменти управління ресурсами: у програмному забезпеченні BigTime забезпечується розподіл працівників за завданнями і балансування навантаження на команду [8], [12];
- аналітика та звітність: у програмному забезпеченні BigTime доступні панелі моніторингу й формування звітів для аналізу ефективності та планування діяльності [8], [12];

- можливість налаштування робочих процесів: у програмному забезпеченні BigTime робочі сценарії адаптуються під специфіку компанії для підвищення гнучкості управління [8], [12];
- штучний інтелект у підтримку керування: у програмному забезпеченні BigTime використовується автоматичний аналіз, прогнозування та формування рекомендацій для оптимізації операцій [8], [12];
- зручність для професійних сервісних компаній: інтерфейс і функціональність програмного забезпечення BigTime орієнтовані на команди, які продають свій час і експертизу [8], [12];
- потреба у навчанні нових користувачів: повне освоєння програмного забезпечення BigTime можливе лише після певної адаптації користувачів до її функціоналу [8], [12].

Серед недоліків програмного забезпечення BigTime можна відзначити складність первинного налаштування та освоєння інтерфейсу, що може вимагати додаткового часу на навчання команди. До того ж доступ до розширених функцій управління фінансами та глибшої аналітики надається лише в дорожчих тарифних планах, що робить систему менш привабливою для невеликих компаній із обмеженим бюджетом. Також деяким користувачам бракує детальнішої довідкової документації, яка б спрощувала впровадження програмного забезпечення в робочі процеси [8], [12].

Програмне забезпечення Smartsheet (рис. 1.3) представляє собою універсальне хмарне рішення, яке забезпечує організацію, контроль і моніторинг проєктної діяльності в одному робочому просторі з можливістю спільного використання даних у режимі реального часу. Платформа поєднує можливості класичного та гнучкого управління проєктами, надаючи зручні інструменти для планування, виконання завдань і аналізу результатів. Завдяки інтерфейсу, що нагадує електронні таблиці, користувачі отримують інтуїтивний доступ до даних і можуть працювати зі структурованою інформацією без потреби у тривалому навчанні, швидко адаптуючись до нового інструменту [10], [13].

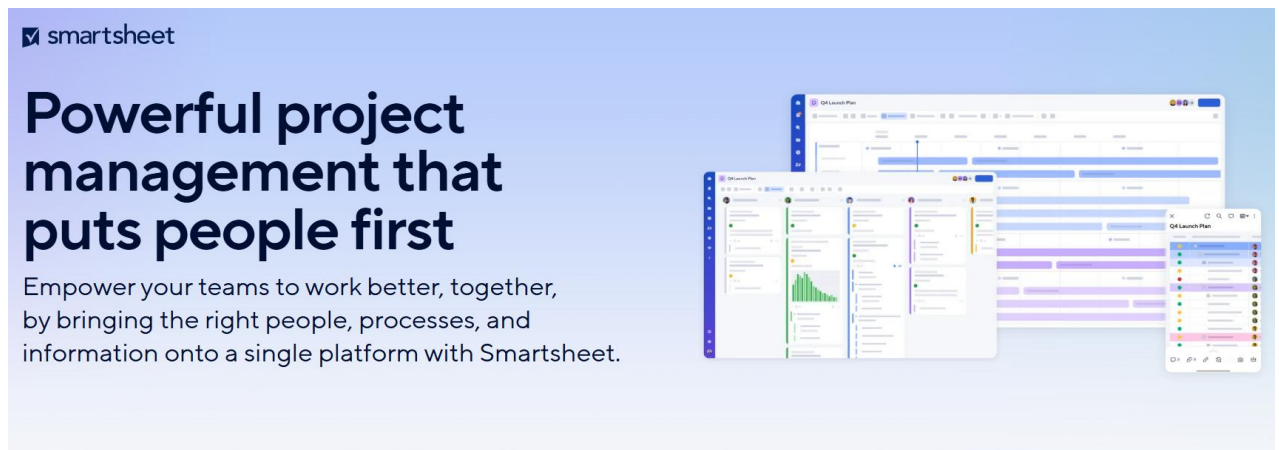


Рисунок 1.3 – Програмне забезпечення Smartsheet [13]

Серед ключових можливостей вирізняється підтримка різних форматів відображення проєкту, що дозволяє кожному учаснику обрати найбільш комфортний спосіб роботи, зокрема перегляд у вигляді календаря, карток, сітки або схеми. Розширені засоби співпраці забезпечують командну взаємодію без необхідності використання сторонніх каналів комунікації. Дані оновлюються синхронно для всіх учасників, а коментарі, статуси та зміни вносяться безпосередньо у робочі елементи [10], [13].

Велика увага у програмному забезпеченні Smartsheet приділена підтримці управління ресурсами: система надає інформацію про завантаженість співробітників та їхню доступність, дозволяючи ефективно розподіляти робоче навантаження. Платформа також включає функції створення дашбордів та аналітичних звітів для контролю загального стану проєктів і портфелів. Мобільний застосунок підтримує збір інформації без підключення до мережі, що робить систему зручною для використання поза офісом та у польових умовах [10], [13].

Програмне забезпечення Smartsheet пропонує широкий набір інтеграцій із сервісами для комунікації, бізнес-аналітики й керування розробкою, що робить його актуальним для організацій із різними сферами діяльності та багаторівневими робочими процесами. Це рішення спрямоване на підвищення прозорості виконання робіт, оптимізацію процесів і забезпечення гнучкої взаємодії між усіма учасниками проєкту [10], [13].

Програмне забезпечення Smartsheet має такі особливості [10], [13]:

- інтерфейс у вигляді таблиці: зовнішній вигляд програмного забезпечення Smartsheet нагадує знайомі електронні таблиці, що спрощує адаптацію користувачів і роботу з великими масивами даних [10], [13];
- різні режими перегляду: у програмному забезпеченні Smartsheet підтримуються календар, сітка, карткове подання та схема для зручного представлення задач залежно від стилю роботи [10], [13];
- робота в реальному часі: усі зміни у програмному забезпеченні Smartsheet синхронізуються миттєво, що забезпечує актуальність інформації та ефективну командну взаємодію [10], [13];
- управління ресурсами: інструменти програмного забезпечення Smartsheet дозволяють контролювати завантаженість членів команди та оптимізувати розподіл робочих навантажень [10], [13];
- мобільність використання: наявність застосунку з можливістю працювати офлайн і збирати дані поза робочим місцем [10], [13];
- аналітика та звітність: у програмному забезпеченні Smartsheet доступні дашборди, автоматичні звіти та розширені показники для відстеження стану проєктів і портфелів [10], [13];
- інструменти співробітництва: програмне забезпечення Smartsheet надає можливість створювати коментарі, а також забезпечує обмін файлами та спільне редагування, що доступні безпосередньо в робочому просторі виконуваного проєкту [10], [13];
- підтримка інтеграцій: з'єднання з іншими програмними сервісами для комунікації, бізнес-аналітики та розробки забезпечує комплексність управління [10], [13].

Серед недоліків програмного забезпечення Smartsheet можна відзначити відносно високу вартість ліцензій для команд із великою кількістю користувачів, що може бути суттєвим обмеженням для малих організацій; певну складність у початковому налаштуванні шаблонів і робочих просторів, особливо для користувачів без досвіду у системах керування проєктами;

обмежені можливості побудови складних діаграм Ганта в порівнянні з професійними інструментами проєктного менеджменту; залежність від стабільного Інтернет з'єднання, оскільки більшість функцій програмного забезпечення Smartsheet реалізована у хмарному середовищі; а також відсутність повноцінної підтримки локального розгортання для організацій із жорсткими вимогами до захисту даних [10], [13].

Порівняльну характеристику програмного забезпечення для керування проєктами наведено у таблиці 1.1. За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для керування проєктами. Проте впровадження таких програмних рішень може супроводжуватися труднощами, пов'язаними з необхідністю навчання працівників, налагодженням інтеграцій з існуючими внутрішніми програмними системами та забезпеченням належного рівня кібербезпеки. Фінансовий аспект також має вагоме значення.

Таблиця 1.1 – Порівняння програмного забезпечення для керування проєктами

Критерій порівняння	Wrike [11]	BigTime [12]	Smartsheet [13]
Зручність управління завданнями та проєктами	+	+—	+
Співпраця та командна комунікація	+	+—	+—
Інтеграції з іншими сервісами	+—	+	+—
Аналітика та звітність	+	+	+—
Невисока вартість	—	+—	—
Гнучкість налаштувань та автоматизація	+	+—	+

Витрати на розгортання, передплату, техпідтримку та навчання користувачів можуть бути відчутними, особливо для невеликих організацій або стартапів. Іноді економічний ефект проявляється не одразу, що створює сумніви в доцільності інвестицій. Тому актуальною є розробка програмного забезпечення для керування проєктами.

При розробці програмного забезпечення для керування проєктами необхідно забезпечити такі функціональні вимоги:

- підтримка можливості збереження та роботи з даними про співробітників, проєкти та замовників;
- підтримка можливості пошуку інформації про співробітників за заданими критеріями;
- можливість отримання інформації в узагальненому вигляді про проєкти компанії;
- можливість роботи з інформацією про виконавців та замовників проєкту;
- підтримка можливості зберігання та завантаження інформації про проєкти з текстових файлів.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для керування проєктами допомагає організовувати, планувати, контролювати та координувати виконання проєктів. Такі програмні засоби надають можливість розподіляти завдання між учасниками, стежити за виконанням роботи в реальному часі, контролювати витрати та дотримання термінів, а також підтримувати комунікацію в команді. Основна мета подібних систем – забезпечити прозорість усіх процесів, підвищити ефективність роботи, зменшити ризики і допомогти завершити проєкт успішно, якісно і вчасно. Такі програмні засоби застосовуються в різних сферах, зокрема в ІТ, будівництві, маркетингу, освіті та будь-яких інших галузях, де діяльність організована у вигляді проєктів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для керування проєктами. Проте впровадження таких програмних рішень може супроводжуватися труднощами, пов'язаними з необхідністю навчання працівників, налагодженням інтеграцій з існуючими внутрішніми програмними системами та забезпеченням належного рівня кібербезпеки. Фінансовий аспект також має вагоме значення. Витрати на розгортання, передплату, техпідтримку та навчання користувачів можуть бути відчутними, особливо для невеликих організацій або стартапів. Іноді економічний ефект проявляється не одразу, що створює сумніви в доцільності інвестицій. Тому актуальною є розробка програмного забезпечення для керування проєктами.

Сформульовано функціональні вимоги до програмного забезпечення для керування проєктами.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для керування проєктами було використано мову програмування C# [14]-[15].

Мова програмування C# вирізняється здатністю поєднувати різні підходи до програмування. Це не лише класична об'єктно-орієнтована мова, а й середовище для функціонального стилю: лямбда-вирази та асинхронні операції формують сучасний і лаконічний код, що особливо важливо в системах, де потрібно опрацьовувати великі обсяги даних або забезпечувати високу швидкодію при роботі з мережею [14]-[15].

Мова C# розвивається дуже динамічно. Кожна нова версія додає можливості, які зазвичай спочатку з'являються у мовах-новаторах, але тут отримують більш практичне, продумане втілення. Наприклад, записи (records), шаблонізація даних та покращена робота з асинхронністю роблять розробку не лише продуктивною, а й приємною з точки зору читабельності та підтримки коду [14]-[15].

Суттєвою особливістю є можливість роботи в екосистемі інструментів Microsoft. Середовище розробки Visual Studio пропонує розширені засоби налагодження, діагностики й автоматичного рефакторингу, що зменшує ймовірність помилок на ранніх етапах. Крім цього, .NET дозволяє запускати застосунки на Linux, Windows і macOS, а також розробляти ігри на основі рушія Unity, що робить C# універсальним вибором – від бізнес-додатків до ігрової індустрії [14]-[15].

Окремої уваги заслуговує культура спільноти: розробники C# активно поширюють відкритий код, створюють безліч бібліотек і фреймворків, які можна адаптувати практично під будь-яке завдання. Сам .NET став відкритим і доступним, що сприяло його глобальному поширенню [14]-[15].

Таким чином, C# – це мова, яка поєднує технологічну зрілість з інноваційністю. Вона постійно розширює спектр своїх можливостей і створює

середовище, у якому як великі команди, так і незалежні розробники можуть реалізовувати масштабні та гнучкі програмні рішення [14]-[15].

Для створення програмного забезпечення для керування проєктами доцільно обрати мову C#, оскільки вона поєднує потужні можливості корпоративної розробки, високу продуктивність та зручність у побудові масштабованих систем. Це сучасна, об'єктно-орієнтована мова, яка забезпечує чітку структуру коду, що є особливо важливим у комплексних застосунках із великою кількістю бізнес-логіки, взаємодій, користувацьких ролей і процесів [14]-[15].

Розробка на C# дає можливість використовувати платформу .NET, яка має широкий набір бібліотек і готових рішень, зокрема для роботи з базами даних, мережевими сервісами, хмарними технологіями, безпекою та інтеграцією з іншими системами. Це робить процес створення функціоналу більш швидким, а підтримку – простішою. Важливо й те, що .NET добре підходить для створення вебзастосунків, які найчастіше й використовуються для керування проєктами, а фреймворк ASP.NET дає змогу будувати продуктивні, надійні та масштабовані вебсервіси [14]-[15].

Додатковою перевагою є підтримка різних платформ. Завдяки .NET можна створювати не лише серверні рішення, а й мобільні або десктопні додатки, що дає змогу забезпечити доступ користувачів до системи управління проєктами з будь-якого пристрою. Інтеграція з хмарними сервісами Microsoft Azure робить можливим впровадження розподілених систем, які легко масштабуються відповідно до зростання кількості користувачів або обсягів даних [14]-[15].

C# також відзначається високим рівнем безпеки, що є важливим для програмного забезпечення, яке працює з конфіденційною інформацією компаній, наприклад бюджетами, звітами, даними про працівників і контрагентів. Інструменти для контролю доступу, шифрування та захисту від уразливостей дозволяють створювати надійні рішення [14]-[15].

Крім технічних переваг, економічні аспекти також грають важливу роль: велика спільнота розробників, хороша документація, наявність інструментів для автоматизації та тестування знижують витрати на розробку та підтримку продукту. Усе це робить С# одним із найдоцільніших виборів для створення систем управління проєктами, які мають бути зручними, швидкими, функціональними та захищеними [14]-[15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для керування проєктами наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для керування проєктами

Критерій порівняння мов програмування	Мова програмування		
	С#	Python	JavaScript
Продуктивність роботи застосунку	+	+—	+—
Підтримка складної бізнес-логіки	+	+	+—
Інструменти для розробки	+	+	+—
Зручність для створення ПЗ для керування проєктами	+	—	—
Швидкість розробки	+	+—	+
Підтримка асинхронності й високих навантажень	+	+	+—

Отже, для реалізації програмного забезпечення для керування проєктами обрано мову програмування С#, яка поєднує потужні можливості розробки, високу продуктивність та зручність у побудові масштабованих програмних систем. Крім того, ця об'єктно-орієнтована мова забезпечує чітку структуру коду, що є особливо важливим у застосунках із великою кількістю бізнес-логіки, взаємодій, користувацьких ролей і процесів. Іншою перевагою є

те, що інтеграція з хмарними сервісами Microsoft Azure робить можливим впровадження розподілених систем, які легко масштабуються відповідно до зростання кількості користувачів або обсягів даних.

2.2 Вибір середовища розробки для створення програмного забезпечення для керування проєктами

В якості середовища розробки для створення програмного забезпечення для керування проєктами було обрано середовище Visual Studio [16], [17].

Середовище розробки Visual Studio вирізняється не лише технічними можливостями, а й тим, що створює комфортне та продуктивне середовище для розробника. Воно фактично виступає в ролі цифрового «помічника», який адаптується до стилю роботи користувача. Завдяки інтелектуальній підтримці написання коду, гнучкому налаштуванню інтерфейсу та можливості підключати власні розширення, кожен може сформувати робочий простір під свої задачі – від швидкого прототипування до розробки складних корпоративних систем [16]-[17].

Важливою особливістю є те, що Visual Studio зосереджується на якості програмного продукту ще на етапі його створення. Вбудовані засоби для статичного аналізу, юніт-тестування, профілювання продуктивності та автоматизації перевірок сприяють тому, що більшість помилок виявляється до того, як продукт потрапить до користувачів. Це особливо цінно в системах, де стабільність і безпека відіграють ключову роль [16]-[17].

Visual Studio також підтримує сучасну культурну парадигму командної розробки. Інструменти інтеграції з Git та іншими системами контролю версій дозволяють стежити за змінами, переглядати історію правок, координувати роботу всередині команди й запобігати конфліктам у коді. Тісний зв'язок із GitHub і Azure DevOps забезпечує плавний процес CI/CD – від написання коду до автоматизованого розгортання [16]-[17].

Ще один аспект – універсальність. Visual Studio здатне працювати і як середовище для традиційних десктопних застосунків, і як платформа для створення мікросервісів, хмарних API, мобільних застосунків та навіть ігор. Це спрощує розвиток програмного продукту, коли проєкт розширюється на нові напрямки або потребує підтримки різних платформ [16]-[17].

Усе це робить Visual Studio не просто інструментом розробки, а екосистемою, що сприяє створенню якісних, конкурентоспроможних і довготривалих програмних рішень [16]-[17].

Visual Studio є доцільним вибором для створення програмного забезпечення для керування проєктами завдяки поєднанню продуктивних інструментів і глибокої інтеграції з технологічною екосистемою .NET. Це середовище надає розробникам можливість не лише писати код, а й повністю управляти життєвим циклом застосунку: від проєктування архітектури до тестування, розгортання та подальшої підтримки [16]-[17].

Однією з ключових переваг Visual Studio є інтелектуальні засоби аналізу коду: автоматичні підказки, пропозиції оптимізації, виявлення проблем ще до запуску програми. Такий підхід суттєво зменшує кількість помилок у складних системах, де важливо підтримувати високу надійність і безперервність роботи. Розширені можливості налагодження дозволяють не лише відстежувати поведінку програми у звичайному режимі, але й аналізувати виконання в реальному часі, відстежувати витoki пам'яті, перевіряти продуктивність окремих компонентів [16]-[17].

Visual Studio має тісну інтеграцію з хмарною інфраструктурою Microsoft Azure. Це відкриває шлях до створення розподілених систем управління проєктами, які можуть масштабуватися відповідно до потреб компанії, забезпечувати високу доступність і безпечне зберігання даних. Крім того, вбудовані засоби керування версіями, робочими задачами й автоматизованою збіркою на основі DevOps-підходів сприяють ефективній командній роботі – особливо важливо в проєктах, де кожна зміна має бути чітко контрольована [16]-[17].

Середовище підтримує розробку різних типів клієнтів: веб-інтерфейсів, десктопних і мобільних застосунків, що дозволяє створювати комплексні й універсальні рішення для управління проєктами з єдиним стеком технологій. Розширюваність Visual Studio через плагіни та інструменти сторонніх розробників забезпечує гнучкість і можливість налаштування під потреби конкретної команди [16]-[17].

Таким чином, Visual Studio – це не просто редактор коду, а потужна платформа для побудови надійних, масштабованих та інтелектуальних систем управління проєктами з повним циклом супроводу продукту [16]-[17].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для керування проєктами наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для керування проєктами

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Підтримка .NET та екосистеми Microsoft	+	+	+—
Інструменти для налагодження складних застосунків	+	+	—
Вбудована підтримка UI-фреймворків (WPF, WinForms, MAUI)	+	+—	+—
Продуктивність при роботі з великими рішеннями	+	+—	—
Можливість гнучкого налаштування середовища	+	+	+—

Таким чином, для створення програмного забезпечення для керування проєктами обрано середовище розробки Visual Studio завдяки поєднанню продуктивних інструментів і глибокої інтеграції з технологічною екосистемою .NET. Це середовище надає розробникам можливість не лише писати код, а й повністю управляти життєвим циклом застосунку: від проєктування архітектури до тестування, розгортання та подальшої підтримки.

2.3 Вибір системи управління базами даних для створення програмного забезпечення для керування проєктами

В якості системи управління базами даних для створення програмного забезпечення для керування проєктами було обрано MongoDB [18]-[19].

MongoDB – це сучасна документоорієнтована система управління базами даних, яка поєднує в собі гнучкість NoSQL-підходу та інструменти для побудови високопродуктивних і масштабованих рішень. Її ключова сила полягає у здатності працювати з даними, які не мають жорстко фіксованої структури, тому модель зберігання може еволюціонувати разом із застосунком, без необхідності складної міграції або перепроєктування схеми. Завдяки використанню формату BSON дані зберігаються у вигляді, максимально наближеному до об'єктів у коді, що зменшує кількість перетворень та пришвидшує обробку запитів [18]-[19].

MongoDB відкриває можливості для побудови розподілених систем, адже підтримує горизонтальне масштабування, автоматичне розподілення навантаження між вузлами та відмовостійкість завдяки реплікації. Вбудовані механізми повнотекстового пошуку, широкі можливості індексування й потужний конвеєр агрегації роблять цю СУБД не лише сховищем даних, а й інструментом для оперативного аналізу інформації. Хмарні сервіси на кшталт MongoDB Atlas дозволяють розробникам зосередитися на функціоналі застосунку, оскільки питання масштабування, безпеки та моніторингу беруть на себе керовані платформи [18]-[19].

Екосистема MongoDB включає драйвери для різних мов програмування, засоби для візуального керування, моделювання та міграцій, що забезпечує комфортний процес розробки. Завдяки своїй адаптивності й здатності ефективно працювати з великими обсягами різномірної інформації MongoDB стала однією з найпопулярніших технологій зберігання даних у сучасних веб-та мобільних застосунках, де швидкість розвитку та зміна вимог є нормою [18]-[19].

MongoDB є доцільним вибором для створення програмного забезпечення з управління проєктами, оскільки вона органічно відповідає динаміці змін, властивій таким системам. Дані, що пов'язані з проєктною діяльністю, часто мають гнучку структуру: задачі можуть мати різні атрибути, команди можуть доповнювати або змінювати поля, з'являються нові типи інформації – і все це без жорсткої та заздалегідь визначеної схеми. Документоорієнтована модель MongoDB дозволяє розширювати структуру даних без складних міграцій, що пришвидшує розвиток функціоналу продукту [18]-[19].

Ще одним важливим аргументом є здатність MongoDB ефективно працювати з великими обсягами інформації та підтримувати горизонтальне масштабування. У системах керування проєктами з великою кількістю користувачів, файлами, коментарями, журналами активності та звітами навантаження постійно зростає. MongoDB забезпечує стабільну продуктивність навіть за умов різкого збільшення даних або кількості одночасних запитів [18]-[19].

Крім того, MongoDB має високорозвинені механізми реплікації та відмовостійкості, що дозволяє підтримувати безперервну роботу сервісу – важливу рису для платформ, на які покладається вся команда. Доступність даних у хмарі та зручність інтеграції з вебзастосунками також підсилюють її переваги, особливо в умовах, коли проєктні команди працюють віддалено та потребують швидкого доступу до актуальної інформації [18]-[19].

Обґрунтування вибору системи управління базами даних для розробки програмного забезпечення для керування проєктами наведено у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору системи управління базами даних для розробки програмного забезпечення для керування проєктами

Критерій порівняння систем управління базами даних	Системи управління базами даних		
	MongoDB	MySQL	SQLite
Гнучкість структури даних	+	+—	+—
Масштабованість	+	+—	—
Відмовостійкість та реплікація	+	+—	—
Простота інтеграції з сучасними вебтехнологіями	+	+	+—
Швидкість розробки	+	+	+—
Продуктивність у хмарних рішеннях	+	+—	+—

Таким чином, для створення програмного забезпечення для керування проєктами обрано систему управління базами даних MongoDB, яка поєднує гнучкість структури даних, можливості масштабування та сучасні засоби забезпечення надійності, що робить її ефективною основою для розробки адаптивних і високопродуктивних систем управління проєктами.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для керування проєктами обрано мову програмування C#, яка поєднує потужні можливості розробки, високу продуктивність та зручність у побудові масштабованих програмних систем. Крім того, ця об'єктно-орієнтована мова забезпечує чітку структуру

коду, що є особливо важливим у застосунках із великою кількістю бізнес-логіки, взаємодій, користувацьких ролей і процесів.

Для створення програмного забезпечення для керування проєктами обрано середовище розробки Visual Studio завдяки поєднанню продуктивних інструментів і глибокої інтеграції з технологічною екосистемою .NET. Це середовище надає розробникам можливість не лише писати код, а й повністю управляти життєвим циклом застосунку: від проєктування архітектури до тестування, розгортання та подальшої підтримки.

Для створення програмного забезпечення для керування проєктами обрано систему управління базами даних MongoDB, яка поєднує гнучкість структури даних, можливості масштабування та сучасні засоби забезпечення надійності, що робить її ефективною основою для розробки адаптивних і високопродуктивних систем управління проєктами.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для керування проєктами

Структуру програмного забезпечення для керування проєктами наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного забезпечення для керування проєктами

Структура програмного забезпечення для керування проєктами організована у логічно взаємопов'язані модулі (модуль управління користувачами, модуль управління проєктами, модуль роботи з базою даних,

модуль інтерфейсу користувача, модуль аналітики й моніторингу), кожен з яких виконує окремий набір функцій і забезпечує цілісність системи.

Центральне місце займає модуль управління користувачами, відповідальний за автентифікацію, розмежування доступу та персоналізацію середовища взаємодії, щоб кожен учасник проєкту працював лише з потрібними даними.

Модуль адміністративного управління проєктами забезпечує створення, налаштування та архівування проєктних сутностей, що включають назву, опис, часові рамки й статус виконання. Взаємодія з локальними файлами також зосереджена у цьому модулі, що дає можливість створювати, редагувати та відновлювати дані проєктів у текстовому форматі у випадках офлайн-доступу чи резервування.

Ключову функцію виконує модуль роботи з базою даних, який забезпечує підключення до MongoDB, виконання CRUD-операцій і синхронізацію інформації між сервером і клієнтом.

За відображення інформації відповідає модуль інтерфейсу користувача, який перетворює дані на зручні таблиці та візуальні компоненти, забезпечуючи швидке сприйняття стану завдань, ресурсів та ролей учасників.

Модуль аналітики й моніторингу, що відстежує прогрес, формує статистику, дозволяє оцінювати ефективність виконання проєкту та приймати рішучі управлінські рішення на основі актуальних показників. Сукупність цих шести модулів формує гнучку систему, здатну підтримувати повний цикл керування проєктами від початку планування до завершення робіт.

У програмному забезпеченні для керування проєктами визначено кілька ключових сутностей, що відповідають за моделювання учасників і процесів у системі. Для зберігання інформації про замовників застосовується окремий клас, який містить властивості для імені, прізвища, країни та пов'язаних проєктів. Ці поля прив'язуються до відповідних документів у MongoDB, а ідентифікація елементів виконується за ObjectId. Завдяки наявності конструктора без параметрів забезпечується можливість створення

об'єктів під час отримання даних із бази даних програмного забезпечення для керування проєктами.

Інтерфейс взаємодії з даними замовників у програмному забезпеченні для керування проєктами реалізовано у формі, що приймає під час створення активний екземпляр MongoClient. У прихованих елементах зберігається підключення та методи, які формують запити до бази залежно від користувацьких параметрів. Після обробки інформація може бути відображена на окремому екрані із таблицею, куди дані передаються через конструктор відповідної форми.

Аналогічна структура застосована для співробітників: існує клас співробітника із властивостями для імені, прізвища, посадової ролі, зайнятості та назви діючого проєкту. Він також синхронізується із документами з MongoDB. Для навігації в інтерфейсі програмного забезпечення для керування проєктами передбачені форми, які запитують перелік працівників, відображають їх і дозволяють переглядати детальнішу інформацію за певним проєктом.

Для керування зв'язком між працівниками та їх завданнями передбачена форма програмного забезпечення для керування проєктами, що отримує одночасно і локальну колекцію, і зовнішнє посилання на колекцію з БД. Там також враховується вибраний користувачем елемент у таблиці для подальших операцій. Якщо необхідно, може відкриватися додаткова сторінка, яка відзначає участь співробітників у конкретному проєкті.

Головна форма програмного забезпечення для керування проєктами виконує функції навігаційного контейнера, підтримує єдине підключення до бази, стежить за станом відкритих вкладок та забезпечує виклик сторінок, що відповідають за замовників, співробітників і проєкти. Під час запуску програмного забезпечення для керування проєктами відбувається початкове налаштування інтерфейсу та службових об'єктів.

Окремі форми програмного забезпечення для керування проєктами надають допоміжний функціонал: сторінка довідки відображає підказки для

користувача, сторінка з деталями про замовника демонструє його дані у структурованому вигляді.

Система також оперує інформацією про самі проєкти. Для цього визначено клас, що описує назву, представника-замовника, чисельність команди розробників, дату початку та кінцевий термін виконання. Форма керування проєктами отримує всі проєкти з БД та забезпечує пошук і перегляд залучених ресурсів. Під час взаємодії з елементами таблиці користувач може переходити до перегляду відповідного замовника або співробітників, що беруть участь у роботі.

У випадках, коли інформація зберігається локально у файлах, застосовується спеціальний модуль, що зчитує вміст текстового документа та відбудовує дані у табличному представленні. Після редагування можливе повне оновлення файлу.

Додатково існує форма для створення нового проєкту у локальному представленні: вона дає змогу визначити шлях до майбутнього файлу, а кнопкові події відповідають за збереження введених даних.

Такий підхід до побудови структури програмного забезпечення для керування проєктами дозволяє відокремити логіку доступу до бази, взаємодію з користувачем і зберігання даних у моделях, що забезпечує підтримуваність і розширюваність системи керування проєктами.

3.2 Функціонування програмного забезпечення для керування проєктами

Функціонування програмного забезпечення для керування проєктами подамо за допомогою схеми, зображеної на рис. 3.2.

Коли користувачу потрібно розпочати роботу з новим проєктом без підключення до БД або зберегти певну конфігурацію локально, система створює новий текстовий документ. Такий файл може містити базові дані про проєкт і використовуватися як автономне джерело інформації.

Під час відкриття локального проєкту застосунок читає вміст відповідного файлу за допомогою потоку введення. Зчитана інформація аналізується і відображається на екрані в структурованому вигляді, тому навіть локальні файли інтегруються у єдину логіку роботи програми.

Редагування проєктних даних також автоматизоване: після внесення змін у таблицю система вилучає вміст попередньої версії файлу та формує новий запис із оновленою інформацією. Таким чином забезпечується актуальність локального сховища й уникнення дублювання або накопичення застарілих даних.

У підсумку застосунок функціонує як засіб централізованої роботи з проєктними даними: отримує їх із хмарного сховища, відображає для користувача, дозволяє створювати, редагувати та синхронізувати інформацію, а також підтримує локальне зберігання, що робить роботу з проєктами гнучкою та зручною навіть за відсутності постійного доступу до мережі.

3.3 Розробка бази даних програмного забезпечення для керування проєктами

База даних програмного забезпечення для керування проєктами формує централізоване сховище, у якому зосереджена ключова інформація про співробітників, замовників та самі проєкти. Використовується MongoDB як документоорієнтована СУБД, оскільки вона забезпечує швидке збереження структурованих даних без складної конфігурації та природно поєднується з об'єктною моделлю у програмному коді. Уся необхідна інформація у сховищі організована в межах бази даних під назвою Work, що містить три незалежні

тематичні колекції: дані щодо співробітників, відомості про клієнтів та описи активних проєктів. Їхній розмір невеликий, що прискорює доступ і оновлення.

Схему бази даних програмного забезпечення для керування проєктами наведено на рис. 3.3.

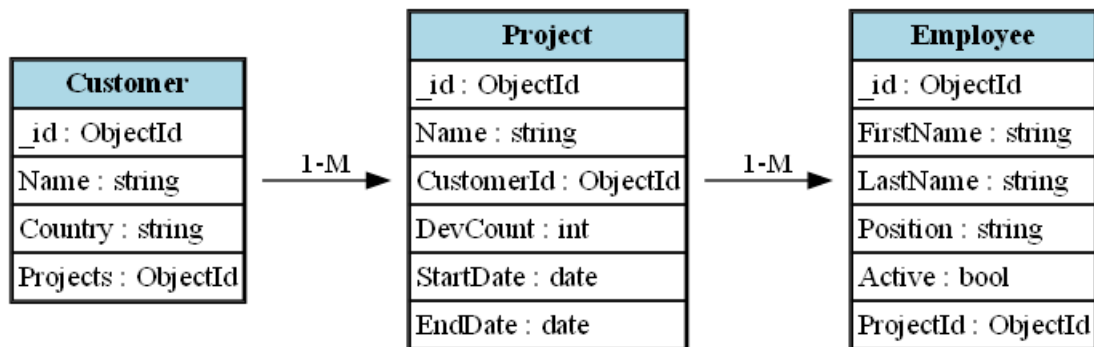


Рисунок 3.3 – Схема бази даних програмного забезпечення для керування проєктами

У колекції Employee зберігаються характеристики учасників команди: персональний ідентифікатор, ім'я та прізвище, робоча позиція, статус зайнятості та назва проєкту, до якого вони прикріплені. Колекція Customer містить набір даних про замовників, включно з країною походження та переліком їхніх проєктів. Колекція Projects формує модель кожного окремого проєкту: назву, клієнта-ініціатора, кількість залучених фахівців, дату запуску та граничний термін завершення. Для кожного запису застосовується автоматично сформований ідентифікатор ObjectId, що гарантує унікальність кожного документа.

Для взаємодії між застосунком і сховищем було створено класи-репрезентатори (карти сутностей), які точно повторюють структури даних у MongoDB та дозволяють зчитувати чи оновлювати інформацію у форматі об'єктних моделей. Клас EmployeeClass інкапсулює дані співробітника та отримує властивості безпосередньо через позначення елементів BSON. Аналогічно реалізовано ProjectClass і CustomerClass, що абстрагують програму

від прямої роботи з документами та спрощують подальшу еволюцію структури даних.

Така організація бази дозволяє динамічно керувати всіма учасниками процесу розробки проєктів, вивантажувати інформацію у зручному форматі та оперативно синхронізувати зміни між інтерфейсом користувача та даними, що фізично зберігаються в MongoDB. Завдяки використанню колекцій без жорстких зв'язків забезпечується адаптивність структури, можливість розширення та гнучкість у роботі з новими атрибутами, не порушуючи поточну архітектуру програмного забезпечення

3.4 Проєктування інтерфейсу програмного забезпечення для керування проєктами

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування проєктами виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для керування проєктами враховані функціональні вимоги до програми:

- підтримка можливості збереження та роботи з даними про співробітників, проєкти та замовників;
- підтримка можливості пошуку інформації про співробітників за заданими критеріями;
- можливість отримання інформації в узагальненому вигляді про проєкти компанії;
- можливість роботи з інформацією про виконавців та замовників проєкту;
- підтримка можливості зберігання та завантаження інформації про проєкти з текстових файлів.

Інтерфейс головної форми програмного забезпечення для керування проєктами наведено на рис. 3.4.

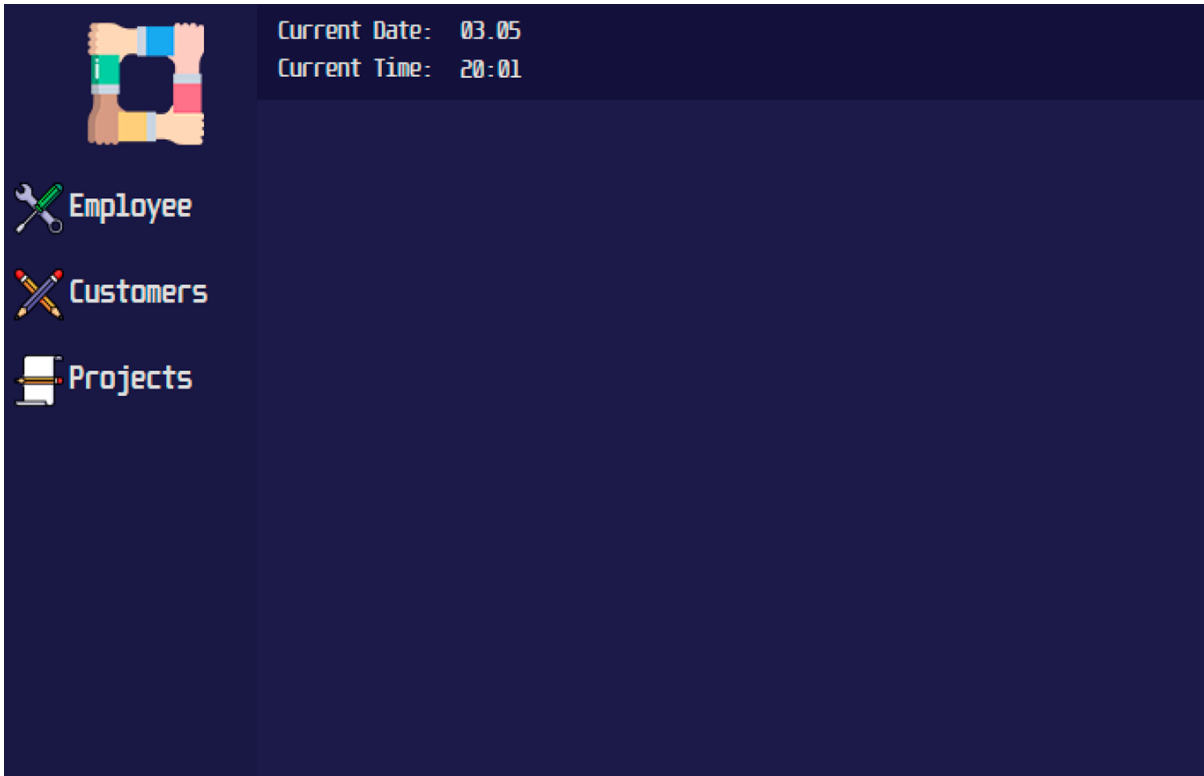


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для керування проєктами

Інтерфейс форми з інформацією про проєкти компанії наведено на рис. 3.5.

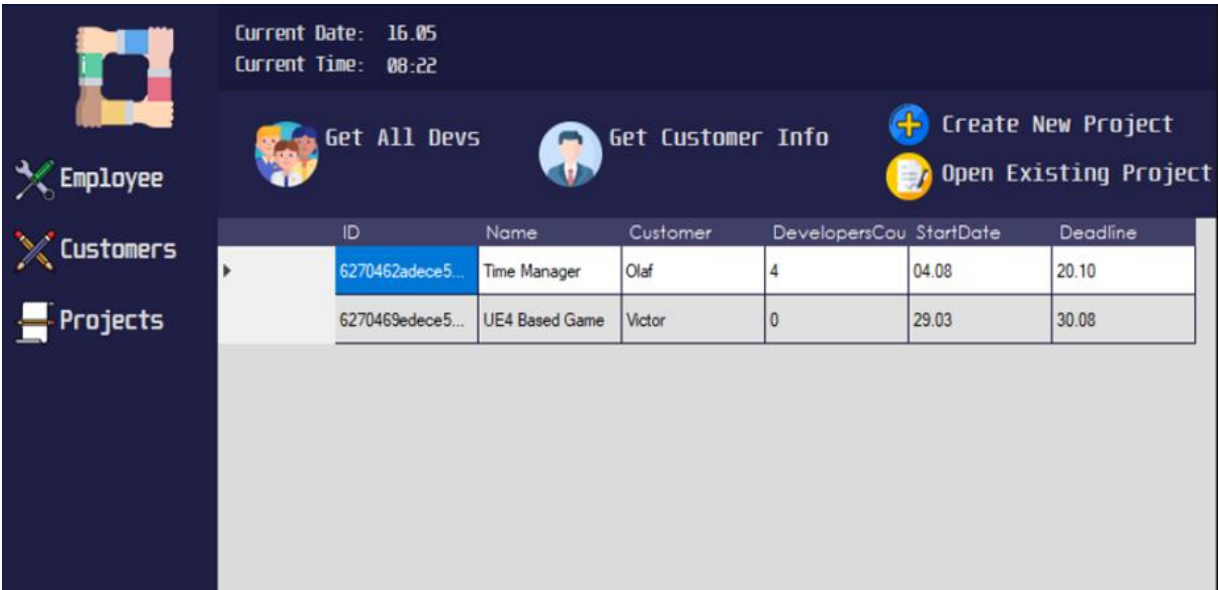
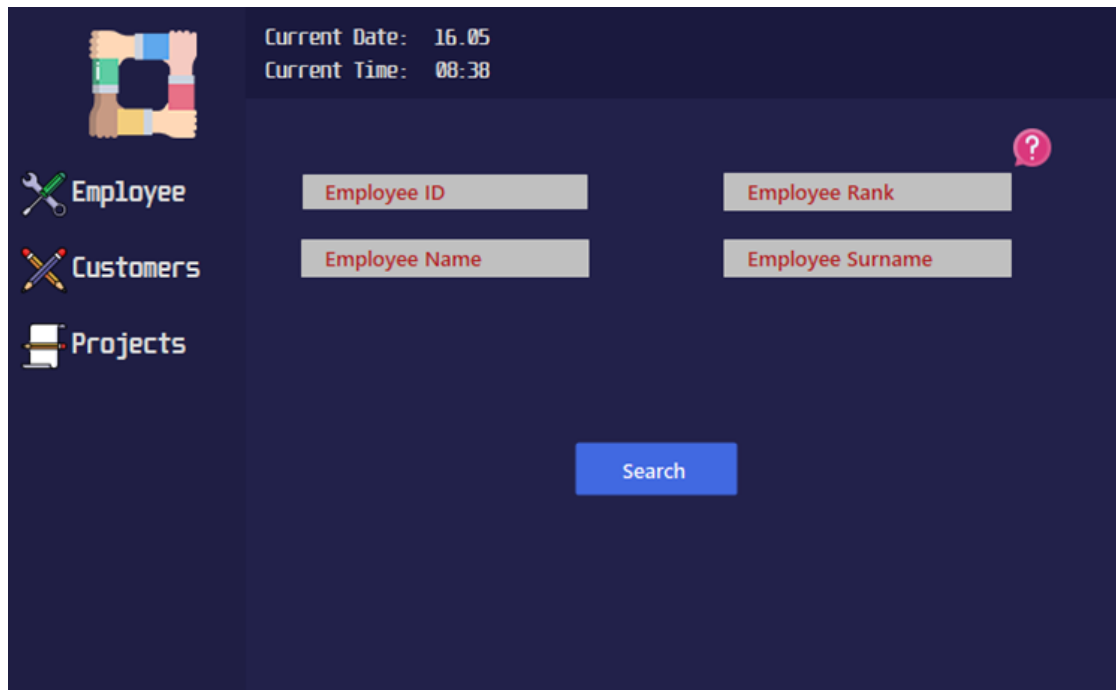


Рисунок 3.5 – Інтерфейс форми з інформацією про проєкти компанії

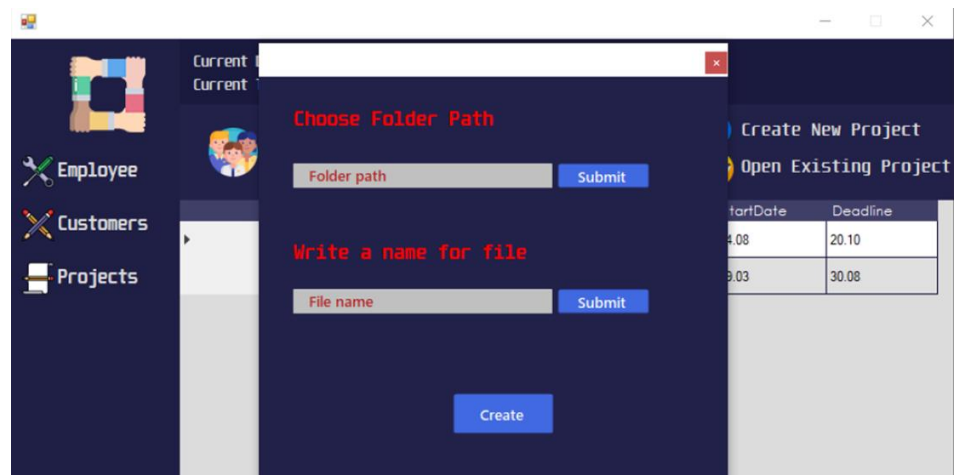
Форма пошуку співробітників наведено на рис. 3.6.



The screenshot shows a web application interface for searching employees. On the left is a sidebar with a logo of four hands and three menu items: 'Employee' (with a wrench icon), 'Customers' (with a pencil icon), and 'Projects' (with a document icon). The main area has a dark blue header with 'Current Date: 16.05' and 'Current Time: 08:38'. Below the header, there are four input fields arranged in a 2x2 grid: 'Employee ID', 'Employee Rank', 'Employee Name', and 'Employee Surname'. A blue 'Search' button is centered at the bottom. A small pink question mark icon is in the top right corner.

Рисунок 3.6 – Форма пошуку співробітників

Форму створення нового проєкту наведено на рис. 3.7.



The screenshot shows a web application interface for creating a new project. A modal dialog box is open in the center with a red title 'Choose Folder Path'. It contains two sections: 'Choose Folder Path' with a 'Folder path' input field and a 'Submit' button, and 'Write a name for file' with a 'File name' input field and a 'Submit' button. A blue 'Create' button is at the bottom of the modal. In the background, the application interface is visible, including the sidebar with 'Employee', 'Customers', and 'Projects' menu items. The main area shows a 'Create New Project' button, an 'Open Existing Project' button, and a table with two columns: 'StartDate' and 'Deadline'. The table has two rows of data: the first row has '14.08' and '20.10', and the second row has '19.03' and '30.08'.

Рисунок 3.7 – Форма створення нового проєкту

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для керування проєктами.

Запропоновано структуру програмного забезпечення для керування проєктами. Визначено, що структура програмного забезпечення для керування проєктами організована у логічно взаємопов'язані модулі (модуль управління користувачами, модуль управління проєктами, модуль роботи з базою даних, модуль інтерфейсу користувача, модуль аналітики й моніторингу), кожен з яких виконує окремий набір функцій і забезпечує цілісність системи.

Описано функціонування програмного забезпечення для керування проєктами. Функціонування програмного забезпечення для керування проєктами побудоване таким чином, щоб забезпечити зручний обмін інформацією між інтерфейсом користувача, локальними файлами та базою даних.

Розроблено базу даних програмного забезпечення для керування проєктами, яка зберігає інформацію про співробітників, замовників та самі проєкти. Організація бази даних програмного забезпечення для керування проєктами дозволяє динамічно керувати всіма учасниками процесу розробки проєктів, вивантажувати інформацію у зручному форматі та оперативно синхронізувати зміни між інтерфейсом користувача та даними, що фізично зберігаються в MongoDB.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування проєктами.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для керування проєктами. Програма написана на мові C#, яка поєднує потужні можливості розробки, високу продуктивність та зручність у побудові масштабованих програмних систем. В якості середовища розробки для створення програмного забезпечення для керування проєктами обрано Visual Studio завдяки поєднанню продуктивних інструментів і глибокої інтеграції з технологічною екосистемою .NET.

Програмне забезпечення для керування проєктами забезпечує виконання таких функцій:

- підтримка можливості збереження та роботи з даними про співробітників, проєкти та замовників;
- підтримка можливості пошуку інформації про співробітників за заданими критеріями;
- можливість отримання інформації в узагальненому вигляді про проєкти компанії;
- можливість роботи з інформацією про виконавців та замовників проєкту;
- підтримка можливості зберігання та завантаження інформації про проєкти з текстових файлів.

Для роботи програмного забезпечення керування проєктами потрібні такі технічні та програмні ресурси: комп'ютерна система з процесором, що забезпечує стабільну обробку даних та взаємодію з базою даних, наприклад, центральний процесор із тактовою частотою не нижче 1.8 ГГц. Обсяг оперативної пам'яті має бути не меншим за 4 ГБ, що гарантує плавне виконання обчислень та візуалізацію інформації в інтерфейсі користувача без затримок. Для встановлення та зберігання локальних файлів рекомендується наявність вільного місця на диску обсягом від 500 МБ.

З боку програмного забезпечення потрібна операційна система сімейства Windows, починаючи з версії Windows 10 або новіших, зі встановленим пакетом .NET Framework не нижче версії 4.7.2, що забезпечує підтримку технологічних компонентів програми. Також необхідний драйвер або клієнт для взаємодії з MongoDB, що дозволяє здійснювати доступ до даних та їх обробку. Для мережевої взаємодії з базою даних у разі віддаленого розміщення потрібно стабільне підключення до Інтернету зі швидкістю не нижче 2 Мбіт/с для швидкого обміну інформацією.

Такі вимоги гарантують коректне функціонування програмного забезпечення, стабільний доступ до бази даних і комфортну роботу користувачів із системою під час керування інформацією про проекти, замовників і співробітників.

Функціональні характеристики розробленого програмного забезпечення для керування проектами наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для керування проектами наведено у додатку Б.

4.2 Характеристики програми для керування проектами

Програмне забезпечення для керування проектами характеризується можливістю централізованої організації даних, що дозволяє в єдиному середовищі опрацьовувати інформацію про проекти, замовників і співробітників. Забезпечується швидкий доступ до оновлених відомостей та синхронізація відображення інформації після будь-яких змін. Застосунок вирізняється інтуїтивним інтерфейсом, який сприяє зручному взаємодіянню користувача з елементами системи без потреби у тривалому навчанні.

Система підтримує кероване редагування та збереження даних, що знижує ризик втрати важливої інформації й забезпечує прогнозовану логіку змін. Використання документно-орієнтованого підходу в роботі з базою даних дозволяє легко адаптувати структуру до змін вимог і швидко масштабувати

систему без суттєвих переробок. Застосовується механізм валідації для запобігання помилкам введення, що підвищує точність обробки інформації.

Функціонування програмного забезпечення для керування проєктами передбачає мінімальні залежності від зовнішніх компонентів, завдяки чому система залишається стабільною та здатною працювати автономно при необхідності. Швидкість обробки даних гарантує оперативне оновлення відображуваної інформації та комфортну взаємодію користувачів із інструментами керування. Передбачена підтримка базових можливостей захисту, що дає можливість контролювати доступ до конфіденційних відомостей.

Загалом програмне забезпечення для керування проєктами орієнтоване на забезпечення ефективного управління проєктами шляхом оптимізації процесів планування, моніторингу та розподілу навантаження між учасниками розробки, одночасно зберігаючи простоту експлуатації та гнучкість удосконалення.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для керування проєктами необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для керування проєктами є персональні дані співробітників, інформація про їхню посаду, рівень завантаженості та участь у поточних проєктах. Також у систему надходять дані про замовників, їхні контакти, країну діяльності й пов'язані з ними проєкти. Окремим видом вхідної інформації є атрибути проєктів, такі як їх назва, відповідальні особи, терміни виконання та кількість залучених

розробників, а також зміни до цих параметрів, які менеджер може вносити в процесі планування або реалізації.

Вихідними даними програмного забезпечення для керування проєктами є результати обробки та відображення введеної інформації у вигляді зручних для аналізу таблиць. Система надає статус виконання завдань, відображає зміни у ролях і зайнятості працівників, а також формує інформаційне представлення про терміни, що дозволяє контролювати організацію розробки та своєчасно реагувати на ризики.

4.3.3 Повідомлення

Користувач програмного забезпечення для керування проєктами може отримати такі повідомлення: підтвердження успішних дій, наприклад, під час оновлення статусу співробітника або зміни термінів розробки; інформаційні повідомлення про виконання пошуку та результат його відсутності, коли система не знаходить співробітника або замовника відповідно до зазначених критеріїв; попередження про можливі помилки введення або відсутність підключення до бази даних під час спроби редагування.

4.4 Виконання програмного забезпечення для керування проєктами

Після запуску програмного забезпечення для керування проєктами користувачу відображається головна форма (рис. 4.1). За допомогою основного меню, яке знаходиться зліва, користувач може отримати інформацію про співробітників, замовників та проєкти.

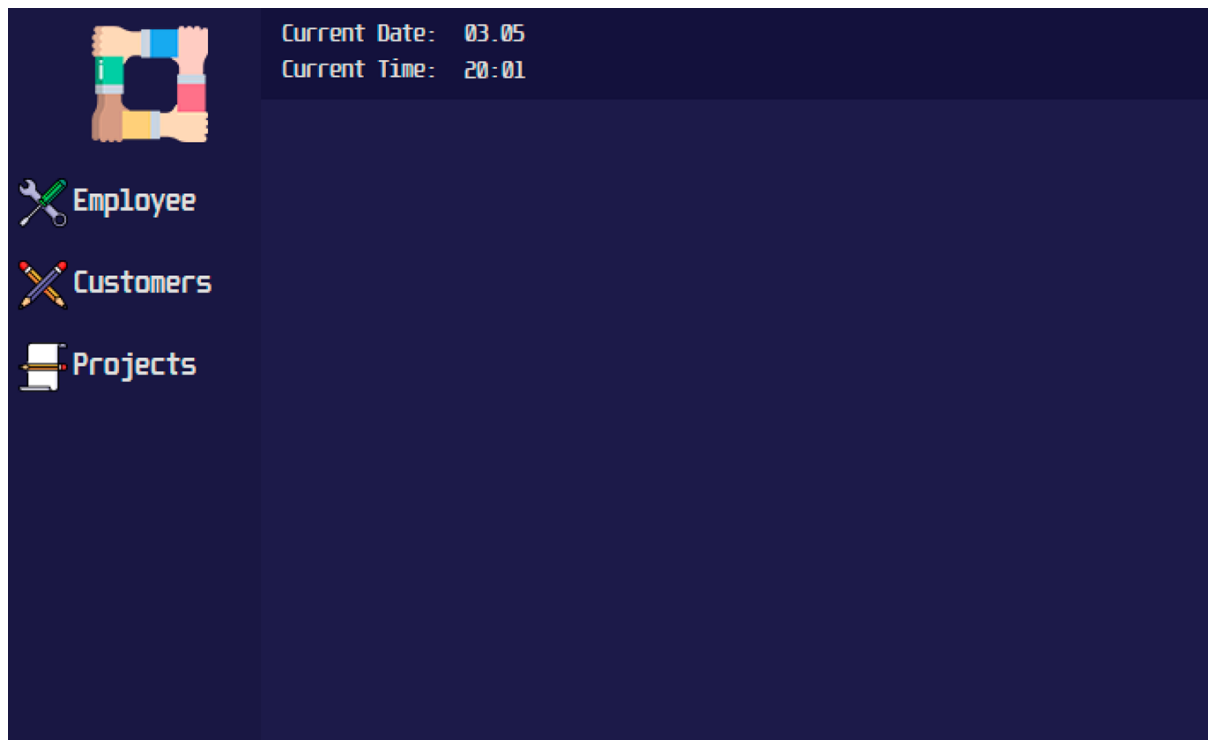


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для керування проєктами

Для пошуку співробітників (рис. 4.2) необхідно натиснути кнопку Employee та обрати критерій пошуку (індивідуальний номер, прізвище та ін.).

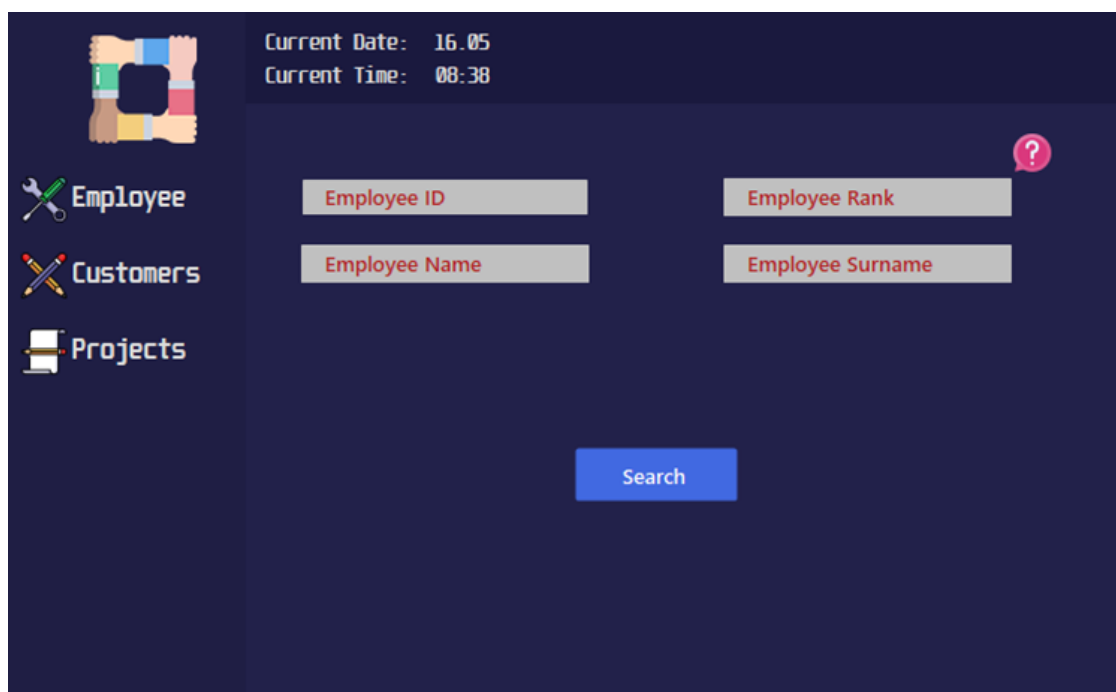
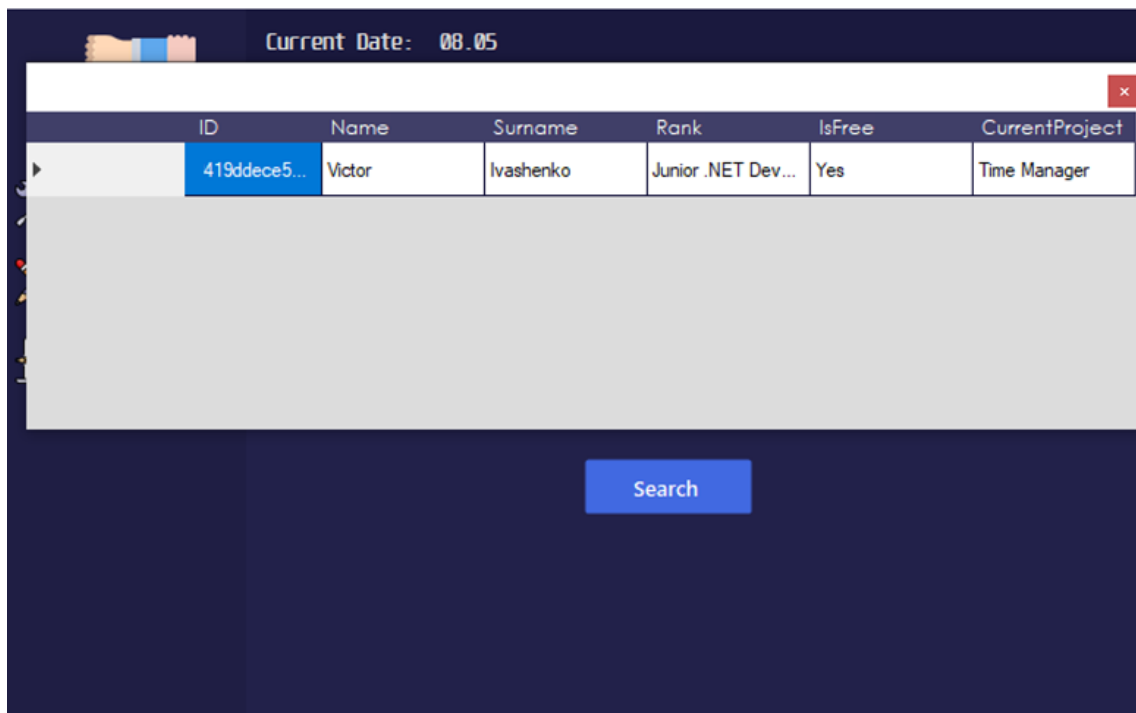


Рисунок 4.2 – Форма пошуку співробітників

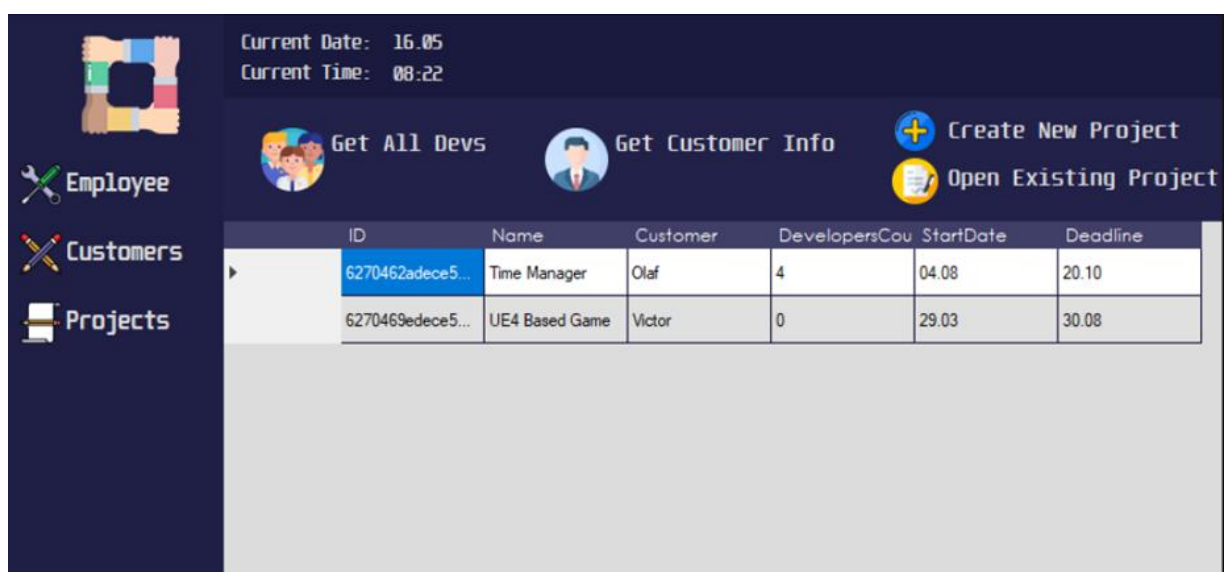
Після введення даних для пошуку виводиться таблиця (рис. 4.3) співробітників, які відповідають заданим пошуковим критеріям.



ID	Name	Surname	Rank	IsFree	CurrentProject
419ddece5...	Victor	Ivashenko	Junior .NET Dev...	Yes	Time Manager

Рисунок 4.3 – Результати пошуку співробітника

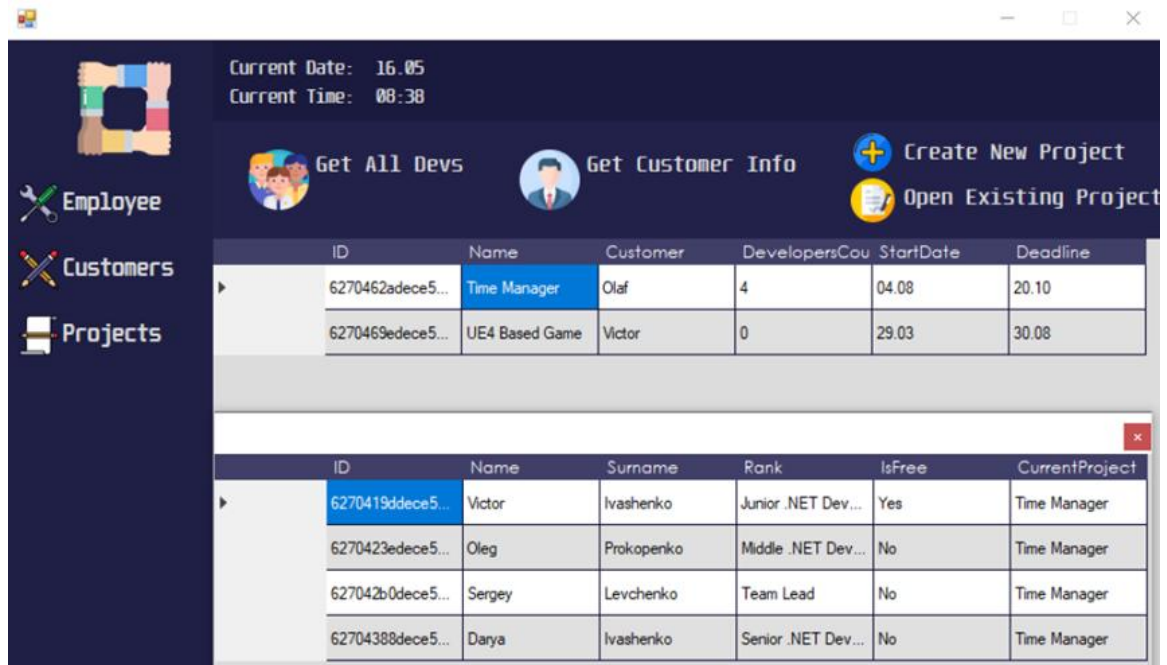
Для отримання інформації про проєкти компанії (рис. 4.4) необхідно натиснути кнопку Projects, внаслідок чого відобразиться таблиця, що подає узагальнену інформацію про проєкти (назва, замовник, дати виконання та ін.).



ID	Name	Customer	DevelopersCou	StartDate	Deadline
6270462adece5...	Time Manager	Olaf	4	04.08	20.10
6270469edece5...	UE4 Based Game	Victor	0	29.03	30.08

Рисунок 4.4 – Форма з інформацією про проєкти компанії

При навігації по проєкту у відповідній таблиці відображається більш детальна інформація про нього, зокрема про всіх співробітників, що задіяні у його виконанні (рис. 4.5).



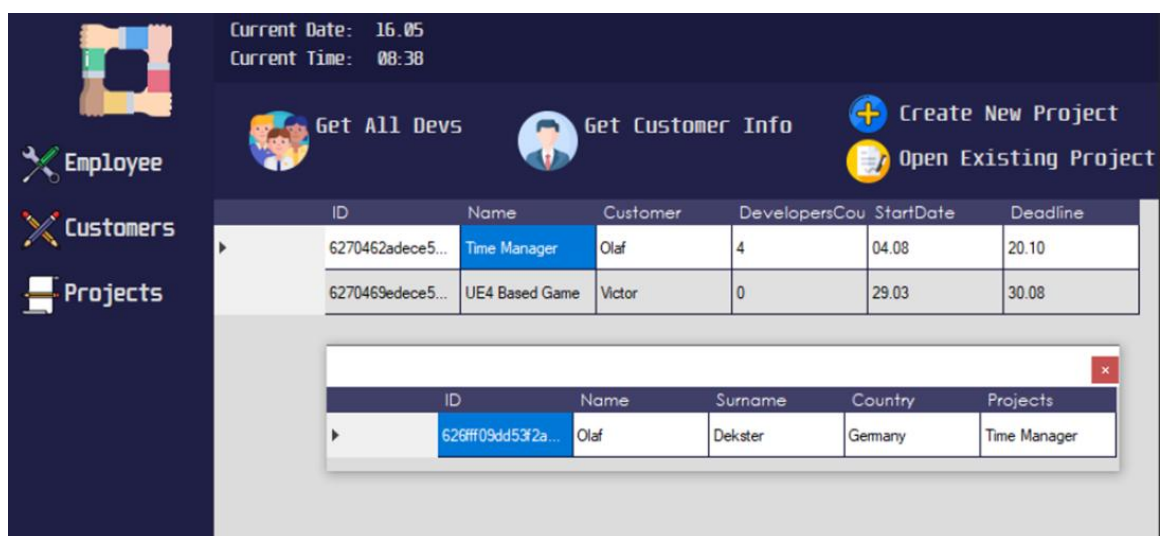
The screenshot shows a software interface with a dark blue sidebar on the left containing icons for 'Employee', 'Customers', and 'Projects'. The main area has a top header with 'Current Date: 16.05' and 'Current Time: 08:38'. Below this are two buttons: 'Get All Devs' and 'Get Customer Info'. To the right are two more buttons: 'Create New Project' and 'Open Existing Project'. The main content area displays two tables. The first table lists projects with columns: ID, Name, Customer, DevelopersCou, StartDate, and Deadline. The second table lists employees with columns: ID, Name, Surname, Rank, IsFree, and CurrentProject.

ID	Name	Customer	DevelopersCou	StartDate	Deadline
6270462adece5...	Time Manager	Olaf	4	04.08	20.10
6270469edece5...	UE4 Based Game	Victor	0	29.03	30.08

ID	Name	Surname	Rank	IsFree	CurrentProject
6270419ddece5...	Victor	Ivashenko	Junior .NET Dev...	Yes	Time Manager
6270423edece5...	Oleg	Prokopenko	Middle .NET Dev...	No	Time Manager
627042b0dece5...	Sergey	Levchenko	Team Lead	No	Time Manager
62704388dece5...	Darya	Ivashenko	Senior .NET Dev...	No	Time Manager

Рисунок 4.5 – Форма з інформацією про виконавців проєкту

Аналогічно можна отримати інформацію про замовника проєкту (рис. 4.6).



The screenshot shows the same software interface as Figure 4.5, but with a different table displayed. The table lists customer information with columns: ID, Name, Surname, Country, and Projects.

ID	Name	Surname	Country	Projects
626ff09dd53f2a...	Olaf	Dekster	Germany	Time Manager

Рисунок 4.6 – Форма з інформацією про замовника проєкту

Для створення нового проєкту необхідно натиснути кнопку Create New Project та ввести відповідну інформацію (рис. 4.7), зокрема обрати папку проєкту та назву.

StartDate	Deadline
14.08	20.10
19.03	30.08

Рисунок 4.7 – Форма створення нового проєкту

При натисканні на кнопку Open Existing Project можна відкрити файл з проєктами, зокрема, з тими, інформацію про які було збережено раніше.

Для пошуку інформації про замовників необхідно натиснути кнопку Customers та у відповідній формі (рис. 4.8) обрати критерій пошуку (індивідуальний номер, прізвище та ін.).

Рисунок 4.8 – Форма пошуку інформації про замовників

Після введення пошукового запиту виводиться таблиця (рис. 4.9) з інформацією про замовників, які відповідають заданим пошуковим критеріям.

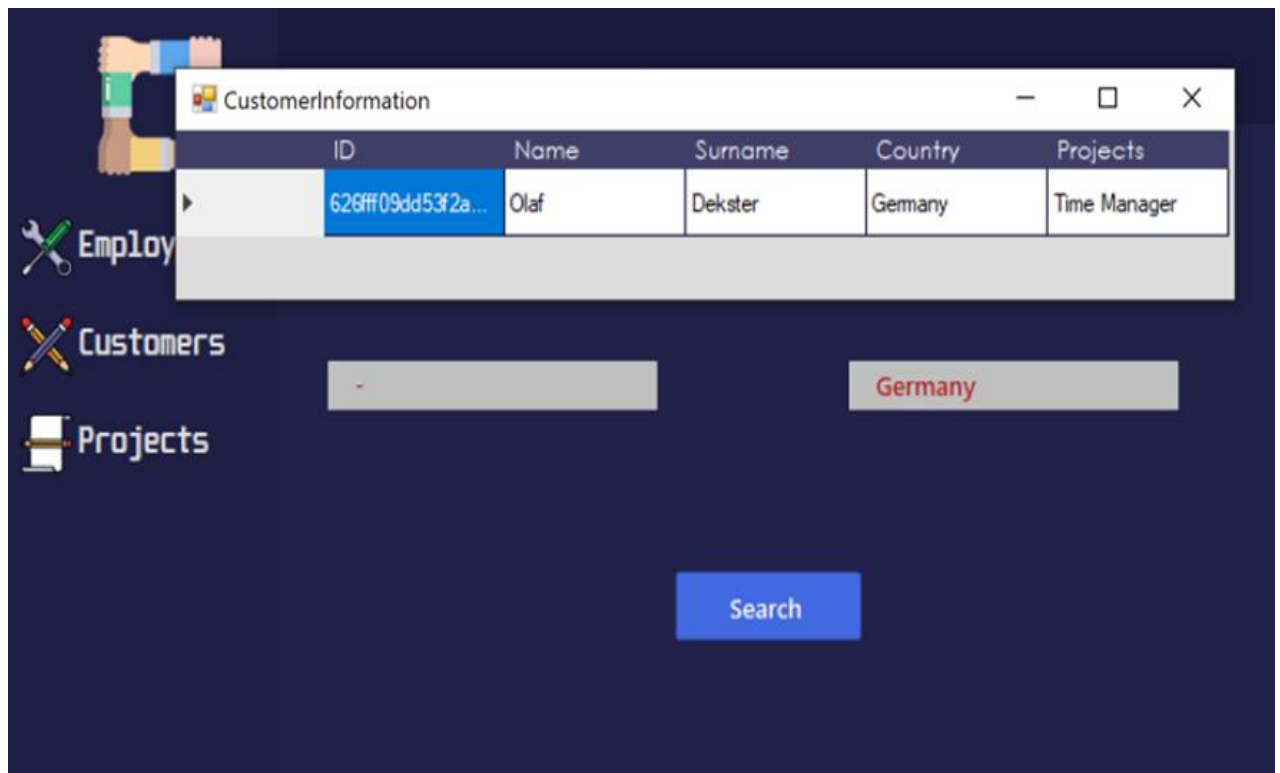


Рисунок 4.9 – Результати пошуку інформації про замовників

4.5 Тестування програмного забезпечення для керування проєктами

Виконано тестування програмного забезпечення для керування проєктами.

Виявлено, що програма для керування проєктами функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу управління проєктами.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу керування проєктами.

4.6 Висновки за розділом 4

Описано програмне забезпечення для керування проєктами. Виконано тестування розробленого програмного забезпечення для керування проєктами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу керування проєктами.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для керування проєктами.

Визначено, що програмне забезпечення для керування проєктами допомагає організовувати, планувати, контролювати та координувати виконання проєктів. Такі програмні засоби надають можливість розподіляти завдання між учасниками, стежити за виконанням роботи в реальному часі, контролювати витрати та дотримання термінів, а також підтримувати комунікацію в команді. Основна мета подібних систем – забезпечити прозорість усіх процесів, підвищити ефективність роботи, зменшити ризики і допомогти завершити проєкт успішно, якісно і вчасно. Такі програмні засоби застосовуються в різних сферах, зокрема в ІТ, будівництві, маркетингу, освіті та будь-яких інших галузях, де діяльність організована у вигляді проєктів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для керування проєктами. Проте впровадження таких програмних рішень може супроводжуватися труднощами, пов'язаними з необхідністю навчання працівників, налагодженням інтеграцій з існуючими внутрішніми програмними системами та забезпеченням належного рівня кібербезпеки. Фінансовий аспект також має вагоме значення. Витрати на розгортання, передплату, техпідтримку та навчання користувачів можуть бути відчутними, особливо для невеликих організацій або стартапів. Іноді економічний ефект проявляється не одразу, що створює сумніви в доцільності інвестицій. Тому актуальною є розробка програмного забезпечення для керування проєктами.

Сформульовано функціональні вимоги до програмного забезпечення для керування проєктами.

Для реалізації програмного забезпечення для керування проєктами обрано мову програмування C#, яка поєднує потужні можливості розробки,

високу продуктивність та зручність у побудові масштабованих програмних систем. Крім того, ця об'єктно-орієнтована мова забезпечує чітку структуру коду, що є особливо важливим у застосунках із великою кількістю бізнес-логіки, взаємодій, користувацьких ролей і процесів.

Для створення програмного забезпечення для керування проєктами обрано середовище розробки Visual Studio завдяки поєднанню продуктивних інструментів і глибокої інтеграції з технологічною екосистемою .NET. Це середовище надає розробникам можливість не лише писати код, а й повністю управляти життєвим циклом застосунку: від проєктування архітектури до тестування, розгортання та подальшої підтримки.

Для створення програмного забезпечення для керування проєктами обрано систему управління базами даних MongoDB, яка поєднує гнучкість структури даних, можливості масштабування та сучасні засоби забезпечення надійності, що робить її ефективною основою для розробки адаптивних і високопродуктивних систем управління проєктами.

Запропоновано структуру програмного забезпечення для керування проєктами. Визначено, що структура програмного забезпечення для керування проєктами організована у логічно взаємопов'язані модулі (модуль управління користувачами, модуль управління проєктами, модуль роботи з базою даних, модуль інтерфейсу користувача, модуль аналітики й моніторингу), кожен з яких виконує окремий набір функцій і забезпечує цілісність системи.

Описано функціонування програмного забезпечення для керування проєктами. Функціонування програмного забезпечення для керування проєктами побудоване таким чином, щоб забезпечити зручний обмін інформацією між інтерфейсом користувача, локальними файлами та базою даних.

Розроблено базу даних програмного забезпечення для керування проєктами, яка зберігає інформацію про співробітників, замовників та самі проєкти. Організація бази даних програмного забезпечення для керування проєктами дозволяє динамічно керувати всіма учасниками процесу розробки

проектів, вивантажувати інформацію у зручному форматі та оперативно синхронізувати зміни між інтерфейсом користувача та даними, що фізично зберігаються в MongoDB.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для керування проєктами.

Виконано тестування розробленого програмного забезпечення для керування проєктами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу керування проєктами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Software Project Management. URL: <https://www.geeksforgeeks.org/software-engineering/software-engineering-software-project-management-spm/> (date of access: 20.04.2026).
2. What Is Project Management Software & What Does It Do. URL: <https://www.cloudwards.net/what-is-project-management-software/> (date of access: 20.04.2026).
3. Compare Project Management Tools & Software Examples. URL: <https://morningmate.com/blog/top-project-management-tools/> (date of access: 20.04.2026).
4. What is Project Management Software and What are its Benefits. URL: <https://www.softlist.io/overview-of-project-management-software/> (date of access: 20.04.2026).
5. All You Need to Know About the 30 Types of Project Management Software. URL: <https://desktrack.timentask.com/blog/types-of-project-management-software/> (date of access: 20.04.2026).
6. What Is Project Management Software: All You Need to Know. URL: <https://productive.io/blog/what-is-project-management-software/> (date of access: 20.04.2026).
7. Project management software comparison. URL: <https://www.projektron.de/en/blog/details/project-management-software-comparison-3669/> (date of access: 20.04.2026).
8. Best Project Management Software. URL: <https://softwareconnect.com/roundups/best-project-management-software/> (date of access: 20.04.2026).
9. Best Project Management Software by Experts. URL: <https://project-management.com/top-10-project-management-software/> (date of access: 20.04.2026).

10. Best Free Project Management Tools to Boost Team Efficiency. URL: <https://project-management.com/free-project-management-software-tools/> (date of access: 20.04.2026).

11. Wrike. URL: <https://www.wrike.com/es/> (date of access: 20.04.2026).

12. BigTime. URL: <https://www.bigtime.net/> (date of access: 20.04.2026).

13. Smartsheet. URL: <https://www.smartsheet.com/s/project-management> (date of access: 20.04.2026).

14. C# Environment. URL: https://www.tutorialspoint.com/csharp/csharp_environment_setup.htm (date of access: 20.04.2026).

15. Methods in C#. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/methods> (date of access: 20.04.2026).

16. Introduction to Visual Studio. URL: <https://www.geeksforgeeks.org/c-sharp/introduction-to-visual-studio/> (date of access: 20.04.2026).

17. Visual Studio Features. URL: <https://riptutorial.com/visual-studio/learn/100004/visual-studio-features> (date of access: 20.04.2026).

18. Introduction to MongoDB. URL: <https://www.mongodb.com/docs/manual/introduction/> (date of access: 20.04.2026).

19. MongoDB Data Modelling. URL: https://www.tutorialspoint.com/mongodb/mongodb_data_modeling.htm (date of access: 20.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для керування проєктами.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для керування проєктами», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для керування проєктами.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для керування проєктами забезпечує виконання таких функцій:

- підтримка можливості збереження та роботи з даними про співробітників, проєкти та замовників;
- підтримка можливості пошуку інформації про співробітників за заданими критеріями;
- можливість отримання інформації в узагальненому вигляді про проєкти компанії;
- можливість роботи з інформацією про виконавців та замовників проєкту;

– підтримка можливості зберігання та завантаження інформації про проєкти з текстових файлів.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для керування проєктами повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для керування проєктами повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для керування проєктами необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення керування проєктами потрібні такі технічні та програмні ресурси: комп'ютерна система з процесором, що забезпечує стабільну обробку даних та взаємодію з базою даних, наприклад, центральний процесор із тактовою частотою не нижче 1.8 ГГц. Обсяг оперативної пам'яті має бути не меншим за 4 ГБ, що гарантує плавне виконання обчислень та візуалізацію інформації в інтерфейсі користувача без

затримок. Для встановлення та зберігання локальних файлів рекомендується наявність вільного місця на диску обсягом від 500 МБ.

З боку програмного забезпечення потрібна операційна система сімейства Windows, починаючи з версії Windows 10 або новіших, зі встановленим пакетом .NET Framework не нижче версії 4.7.2, що забезпечує підтримку технологічних компонентів програми. Також необхідний драйвер або клієнт для взаємодії з MongoDB, що дозволяє здійснювати доступ до даних та їх обробку. Для мережевої взаємодії з базою даних у разі віддаленого розміщення потрібно стабільне підключення до Інтернету зі швидкістю не нижче 2 Мбіт/с для швидкого обміну інформацією.

Такі вимоги гарантують коректне функціонування програмного забезпечення, стабільний доступ до бази даних і комфортну роботу користувачів із системою під час керування інформацією про проекти, замовників і співробітників.

A.3.6 Вимоги до маркування і пакування

Програма для керування проектами може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для керування проектами».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для керування проектами є персональні дані співробітників, інформація про їхню посаду, рівень завантаженості та участь у поточних проектах. Також у систему надходять дані про замовників, їхні контакти, країну діяльності й пов'язані з ними проекти. Окремим видом вхідної інформації є атрибути проектів, такі як їх назва, відповідальні особи, терміни виконання та кількість залучених

розробників, а також зміни до цих параметрів, які менеджер може вносити в процесі планування або реалізації.

Вихідними даними програмного забезпечення для керування проєктами є результати обробки та відображення введеної інформації у вигляді зручних для аналізу таблиць. Система надає статус виконання завдань, відображає зміни у ролях і зайнятості працівників, а також формує інформаційне представлення про терміни, що дозволяє контролювати організацію розробки та своєчасно реагувати на ризики.

ДОДАТОК Б
Фрагмент тексту програми

```

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    private MongoKlnr bdKl = new
MongoKlnr("mngdb+srv://PrsnlData:PrsnlData@cluster0.qzkrc.mngdb.n
et/Work?retryWrites=true&w=majority");
    private Form krntFrm = null;
    private DateTime dt = DateTime.UtcNow.Date;

    private void OpnChldForm(Form _form)
    {
        if (this.krntFrm != null)
            this.krntFrm.Close();
        this.krntFrm = _form;
        Mnknt.Controls.Add(_form);
        Mnknt.Tag = _form;
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        this.Tmlbl.Text = this.dt.ToString("dd/MM/yyyy");
        this.Lbltm.Text = DateTime.Now.ToString("HH:mm");
    }

    private void SpivrbtnkButton_Click(object sender,
EventArgs e)
    {
        Spivrbtnk _ctmp = new Spivrbtnk(ref this.bdKl);
        OpnChldForm(_ctmp);
    }

    private void KstmrsButton_Click(object sender, EventArgs
e)
    {
        Kstmr _ctmp = new Kstmr(ref this.bdKl);
        OpnChldForm(_ctmp);
    }

    private void PrjctsButton_Click(object sender, EventArgs
e)
    {
        PrjctsForm _ctmp = new PrjctsForm(ref this.bdKl);
        OpnChldForm(_ctmp);
    }
}

```



```

        PrizvshTextBox.BorderColorIdle = Color.Red;
        PrizvshTextBox.Text = "Fill it!";
    }
    if(KstmrCountry.Text == String.Empty)
    {
        KstmrCountry.BorderThickness = 1;
        KstmrCountry.BorderColorIdle = Color.Red;
        KstmrCountry.Text = "Fill it!";
    }
    if(NameKstmr.Text == String.Empty)
    {
        NameKstmr.BorderThickness = 1;
        NameKstmr.BorderColorIdle = Color.Red;
        NameKstmr.Text = "Fill it!";
    }
}
IEnumerable<KstmrClass> klktn = null;
String _ctmpS = SearchType();
klktn = KstmrsBrowse(ref _ctmpS);
KstmrInformation _ctmp = new KstmrInformation(ref
klktn);
    _ctmp.Show();
}
}
}

```

KstmrClass.cs

```

using Mngdb.Bson;
using Mngdb.Bson.Serialization.Attributes;
using Mngdb.Driver;

namespace Mnprjct
{
    public partial class KstmrInformation : Form
    {
        public KstmrInformation(ref IEnumerable<KstmrClass>
_list)
        {
            InitializeComponent();
            this.List = _list;
        }
        private IEnumerable<KstmrClass> List;

        private void KstmrInformation_Load(object sender,
EventArgs e)
        {
            DataGridView1.DataSource = this.List;
        }
    }
}

```

```

    }
}

Spivrbtnk.cs
using Mngddb.Driver;

namespace Mnprjct
{
    public partial class Spivrbtnk : Form
    {
        public Spivrbtnk(ref MongoKlnr _klnr)
        {
            InitializeComponent();
            this.Klnr = _klnr;
        }

        private MongoKlnr Klnr = null;
        private IMongoKlktn<SpivrbtnkClass> outsideMain = null;

        private IEnumerable<SpivrbtnkClass>
        SpivrbtnkBrowse(String _st)
        {
            var dbs = this.Klnr.GetDbs("Work");
            IMongoKlktn<SpivrbtnkClass> klktn =
            dbs.GetKlktn<SpivrbtnkClass>("Spivrbtnk");
            this.outsideMain = klktn;
            if (_st.Equals("Rank"))
                return klktn.Find(s =>
            s.Rank.Equals(SpivrbtnkRank.Text)).ToList();
            return klktn.Find(spivrbtnk =>
            spivrbtnk.Name.Equals(NameSpivrbtnk.Text) &&
            spivrbtnk.Prizvsh.Equals(PrizvshTextBox.Text)).ToList();
        }

        private void bifButton1_Click(object sender, EventArgs e)
        {
            if(NameSpivrbtnk.Text == String.Empty ||
            IDTxtBox.Text == String.Empty || PrizvshTextBox.Text ==
            String.Empty || SpivrbtnkRank.Text == String.Empty)
            {
                if(NameSpivrbtnk.Text == String.Empty)
                {
                    NameSpivrbtnk.BorderThickness = 1;
                    NameSpivrbtnk.BorderColorIdle = Color.Red;
                    NameSpivrbtnk.Text = "Fill it!";
                }
                if(IDTxtBox.Text == String.Empty)

```

```

        {
            IDTxtBox.BorderThickness = 1;
            IDTxtBox.BorderColorIdle = Color.Red;
            IDTxtBox.Text = "Fill it!";
        }
        if(PrizvshTextBox.Text == String.Empty)
        {
            PrizvshTextBox.BorderThickness = 1;
            PrizvshTextBox.BorderColorIdle = Color.Red;
            PrizvshTextBox.Text = "Fill it!";
        }
        if(SpivrbtnkRank.Text == String.Empty)
        {
            SpivrbtnkRank.BorderColorIdle = Color.Red;
            SpivrbtnkRank.Text = "Fill it!";
        }
    }
    String _st = SearchType();
    IEnumerable<SpivrbtnkClass> info =
SpivrbtnkBrowse(_st);
    SpivrbtnkGridPage _egp = new SpivrbtnkGridPage(ref
info, ref outsideMain, ref this.Klnr);
    _egp.Show();
}
private void IDTxtBox_Click(object sender, EventArgs e)
{
    IDTxtBox.Text = String.Empty;
}

private void SpivrbtnkRank_Click(object sender, EventArgs
e)
{
    SpivrbtnkRank.Text = String.Empty;
}

private void NameSpivrbtnk_Click(object sender, EventArgs
e)
{
    NameSpivrbtnk.Text = String.Empty;
}

private void PrizvshTextBox_Click(object sender,
EventArgs e)
{
    PrizvshTextBox.Text = String.Empty;
}
private String SearchType()
{

```

```

        String rslt = String.Empty;
        if (this.NameSpivrbtnk.Text.Equals("-"))
            if (this.PrizvshTextBox.Text.Equals("-"))
                rslt = "Rank";
        if (this.SpivrbtnkRank.Text.Equals("-"))
            rslt = "InitialName";
        return rslt;
    }
}

}

SpivrbtnkClass.cs
using System.Klktns.Generic;
using Mngdb.Driver;
using Mngdb.Bson;
using Mngdb.Bson.Serialization.Attributes;

namespace Mnprjct
{
    public partial class SpivrbtnkGridPage : Form
    {
        public SpivrbtnkGridPage(ref IEnumerable<SpivrbtnkClass>
        _ec, ref IMongoKlktn<SpivrbtnkClass> _st,
        ref MongoKlnr _klnr)
        {
            InitializeComponent();
            this.Klktn = _ec;
            this.KlktnOutside = _st;
            this.Klnr = _klnr;
        }

        private MongoKlnr Klnr = null;
        private IEnumerable<SpivrbtnkClass> Klktn = null;
        private IMongoKlktn<SpivrbtnkClass> KlktnOutside = null;
        private String OutsideData = String.Empty;

        private void SpivrbtnkGridPage_Load(object sender,
        EventArgs e)
        {
            {
                DataGridView.DataSource = this.Klktn;
            }
        }
    }
}

```

```

SpivrbtnkGridPrjct.cs
using Mngdb.Driver;
using Mngdb.Bson;
using Mngdb.Bson.Serialization.Attributes;

```

```

namespace Mnprjct
{
    public partial class SpivrbtnkGridPrjct : Form
    {
        public SpivrbtnkGridPrjct(ref IEnumerable<SpivrbtnkClass>
_ctmp)
        {
            InitializeComponent();
            this.Klktn = _ctmp;
        }
        private IEnumerable<SpivrbtnkClass> Klktn;

        private void SpivrbtnkGridPrjct_Load(object sender,
EventArgs e)
        {
            DataGridView.DataSource = this.Klktn;
        }
    }

    public struct Data
    {
        public string ID;
        public string Name;
        public string Kstmr;
        public string Dvlprs;
        public string Strttm;
        public string Ddln;
        public Data(string _one, string _two, string _three,
string _four, string _five, string _six)
        {
            this.ID = _one;
            this.Name = _two;
            this.Kstmr = _three;
            this.Dvlprs = _four;
            this.Strttm = _five;
            this.Ddln = _six;
        }
    }
    private List<Data> datas = new List<Data>();
    private void ReadFile()
    {
        try
        {
            var fileStream = new FileStream(this.Path,
FileMode.Open, FileAccess.Read);
            using (var reader = new StreamReader(fileStream))
            {

```

```

        string _ctmp = String.Empty;
        while ((_ctmp = reader.ReadLine()) != null)
        {
            this.file.Add(_ctmp);
        }
        this.datas.Add(new Data(this.file.ElmntAt(0),
this.file.ElmntAt(1), this.file.ElmntAt(2),
            this.file.ElmntAt(3), this.file.ElmntAt(4),
this.file.ElmntAt(5)));
    }catch (Exception ex) { };
}

private void bifImageButton1_Click(object sender,
EventArgs e)
{
    using (StreamWriter writetext = new
StreamWriter(Path))
    {
        writetext.WriteLine(this.file.ElmntAt(0));
        writetext.WriteLine(this.file.ElmntAt(1));
        writetext.WriteLine(this.file.ElmntAt(2));
        writetext.WriteLine(this.file.ElmntAt(3));
        writetext.WriteLine(this.file.ElmntAt(4));
        writetext.WriteLine(this.file.ElmntAt(5));
    }
}

private void FilesLocal_Load(object sender, EventArgs e)
{
    ReadFile();
    try
    {
        DataTable dt = new DataTable();
        dt.Columns.Add("Kstmr", typeof(string));
        dt.Columns.Add("Dvlprs", typeof(string));
        dt.Columns.Add("Ddln", typeof(string));
        dt.Rows.Add(this.datas[0].ID, this.datas[0].Name,
this.datas[0].Kstmr, this.datas[0].Dvlprs,
            this.datas[0].Strttm, this.datas[0].Ddln);
        DataGrid1.DataSource = dt;
    }catch (Exception ex) { };
}

}
}

```

```

using System.Klktns.Generic;

namespace Mnprjct
{
    public partial class PrjctCreatecs : Form
    {
        public PrjctCreatecs()
        {
            InitializeComponent();
        }
        private string FlPth = String.Empty;

        private void bifButton1_Click(object sender, EventArgs e)
        {
            this.FlPth = TxtBxPth.Text;
            TxtBxPth.Text = String.Empty;
        }

        private void bifButton2_Click(object sender, EventArgs e)
        {
            this.FlPth = FlPth + Flnm.Text;
            Flnm.Text = String.Empty;
        }

        public partial class PrjctKstmrInformationGrid : Form
        {
            public PrjctKstmrInformationGrid(ref
IEnumerable<KstmrClass> _cc)
            {
                InitializeComponent();
                this.KstmrClass = _cc;
            }
            private IEnumerable<KstmrClass> KstmrClass;

            private void PrjctKstmrInformationGrid_Load(object
sender, EventArgs e)
            {
                DataGridView.DataSource = this.KstmrClass;
            }
        }
    }

    private IEnumerable<KstmrClass> KstmrsBrowse()
    {
        var dbs = Klnr.GetDbs("Work");
        var klktn = dbs.GetKlktn<KstmrClass>("Kstmr");
    }
}

```

```

        return klktn.Find(kstmr =>
kstmr.Prjcts.Equals(this.OutsideData)).ToList();
    }

    private IEnumerable<SpivrbtnkClass> SpivrbtnkBrowse()
    {
        var dbs = Klnr.GetDbs("Work");
        var klktn =
dbs.GetKlktn<SpivrbtnkClass>("Spivrbtnk");
        return klktn.Find(spivrbtnk =>
spivrbtnk.CurrentPrjct.Equals(this.OutsideData)).ToList();
    }
    #endregion

    #region UI FUNCTIONS
    private void bifImageButton1_Click(object sender,
EventArgs e)
    {
        try { DataGridView.SelectedRows.Clear();
        }catch(Exception ex) { };
    }

    private void PrjctsForm_Load(object sender, EventArgs e)
    {
        IEnumerable<PrjctsClass> info = PrjctsBrowse();
        DataGridView.DataSource = info;
    }

    private void ManageDevs_Click(object sender, EventArgs e)
    {
        IEnumerable<SpivrbtnkClass> _ctmp =
SpivrbtnkBrowse();
        SpivrbtnkGridPrjct _egp = new SpivrbtnkGridPrjct(ref
        _ctmp);
        _egp.Show();
    }

    private void bifImageButton1_Click_1(object sender,
EventArgs e)
    {
        IEnumerable<KstmrClass> _ctmp = KstmrsBrowse();
        PrjctKstmrInformationGrid _pcif = new
PrjctKstmrInformationGrid(ref _ctmp);
        _pcif.Show();
    }

```

```

        private void DataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            string _ctmp =
DataGridView.Rows[e.RowIndex].Cells[1].Value.ToString();
            this.OutsideData = _ctmp;
        }
    #endregion

    private void AddNewPrjct_Click(object sender, EventArgs
e)
    {
        PrjctCreatecs _ctmp = new PrjctCreatecs();
        _ctmp.Show();
    }

    private void bifImageButton2_Click(object sender,
EventArgs e)
    {
        string _ctmp = String.Empty;
        OpenFileDialog dlg = new OpenFileDialog();
        dlg.Filter = "txt files (*.txt)|*.txt";
        if(dlg.ShowDialog() == DialogResult.OK)
            _ctmp = dlg.FileName;
        FilesLocal _ctmp1 = new FilesLocal(ref _ctmp);
        _ctmp1.Show();
    }
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для керування проєктами

Виконала
ст. гр. КНТ-132

Наталія ТРОЦЕНКО

Керівник
д.т.н., професор

Андрій ОЛІЙНИК

Рисунок В.1 – Слайд 1

ОБ’ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

Об’єктом дослідження є процес розробки програмного забезпечення для керування проєктами.

Предмет дослідження – програмні засоби для керування проєктами.

Мета роботи – розробка програмного забезпечення для керування проєктами та підвищення рівня організаційної ефективності.

Рисунок В.2 – Слайд 2

Задачі роботи:

- виконати аналіз предметної області та програмних засобів для керування проєктами;
- здійснити проєктування програмного забезпечення для керування проєктами;
- створити програмне забезпечення для керування проєктами;
- виконати тестування розробленого програмного забезпечення для керування проєктами.

3

Рисунок В.3 – Слайд 3

Аналіз програмного забезпечення для керування проєктами

Критерій порівняння	Wrike	BigTime	Smartsheet
Зручність управління завданнями та проєктами	+	+–	+
Співпраця та командна комунікація	+	+–	+–
Інтеграції з іншими сервісами	+–	+	+–
Аналітика та звітність	+	+	+–
Невисока вартість	–	+–	–
Гнучкість налаштувань та автоматизація	+	+–	+

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	JavaScript
Продуктивність роботи застосунку	+	+-	+-
Підтримка складної бізнес-логіки	+	+	+-
Інструменти для розробки	+	+	+-
Зручність для створення ПЗ для керування проектами	+	-	-
Швидкість розробки	+	+-	+
Підтримка асинхронності й високих навантажень	+	+	+-

5

Рисунок В.5 – Слайд 5

Обґрунтування вибору середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Підтримка .NET та екосистеми Microsoft	+	+	+-
Інструменти для налагодження складних застосунків	+	+	-
Вбудована підтримка UI-фреймворків (WPF, WinForms, MAUI)	+	+-	+-
Продуктивність при роботі з великими рішеннями	+	+-	-
Можливість гнучкого налаштування середовища	+	+	+-

6

Рисунок В.6 – Слайд 6

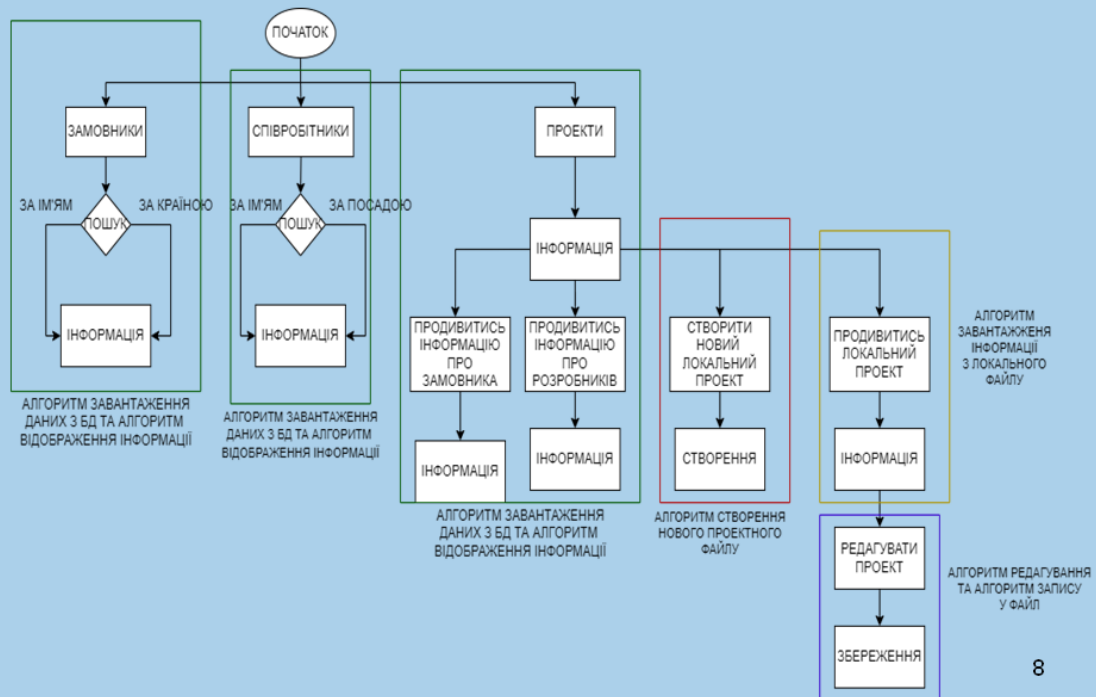
Структура ПЗ для керування проєктами



7

Рисунок В.7 – Слайд 7

Схема функціонування програмного забезпечення для керування проєктами



8

Рисунок В.8 – Слайд 8

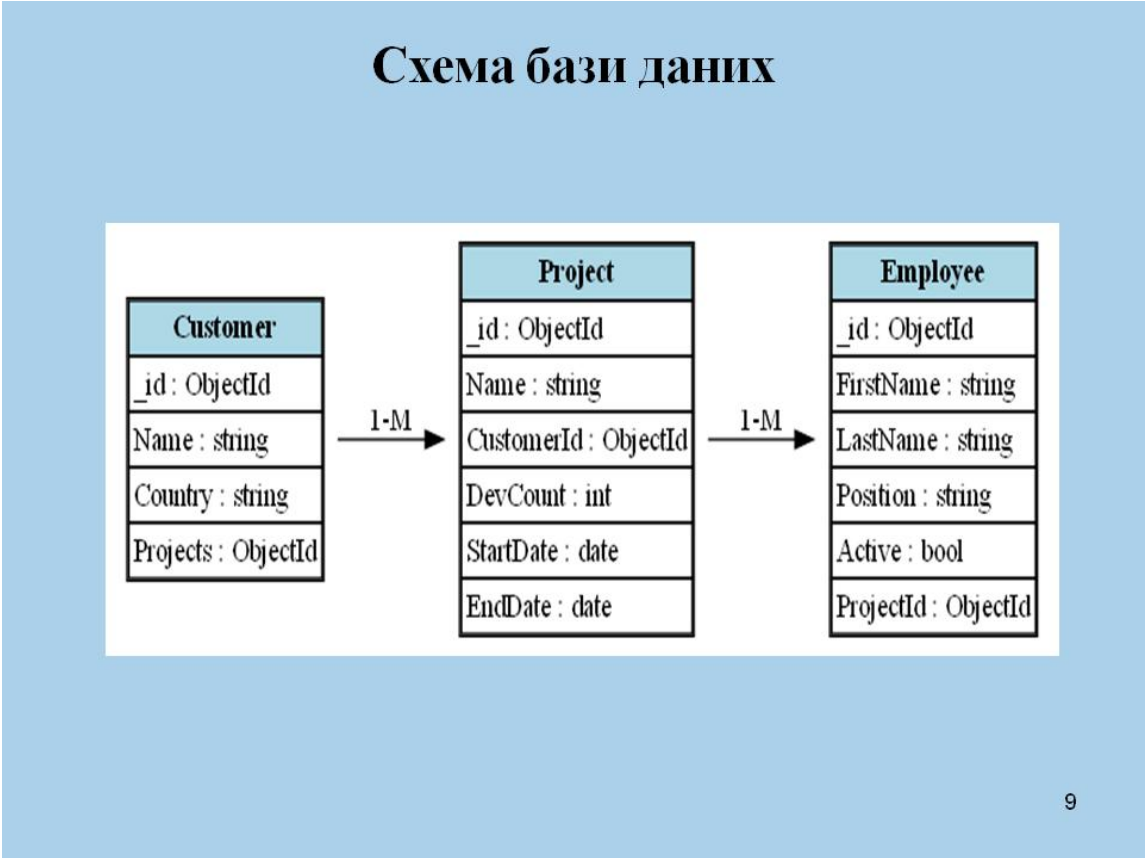


Рисунок В.9 – Слайд 9

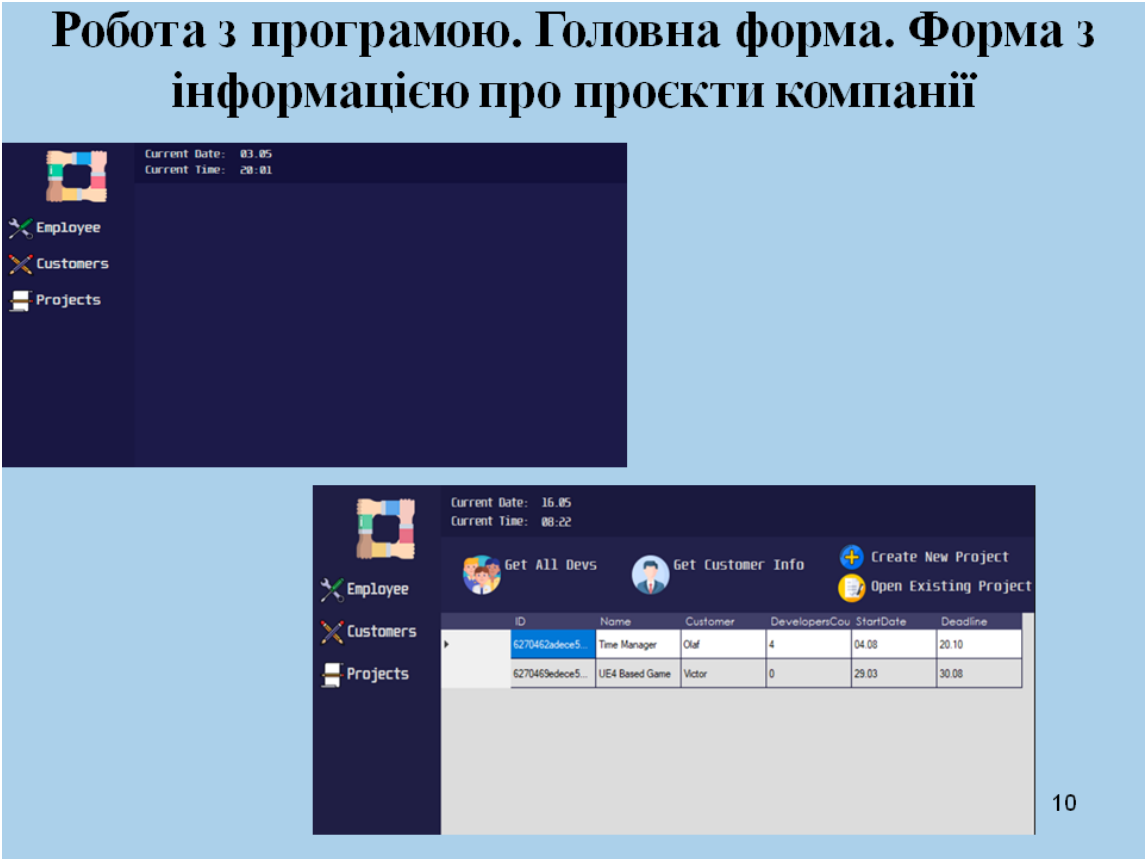


Рисунок В.10 – Слайд 10

Робота з програмою. Інформація про виконавців та замовників проєктів

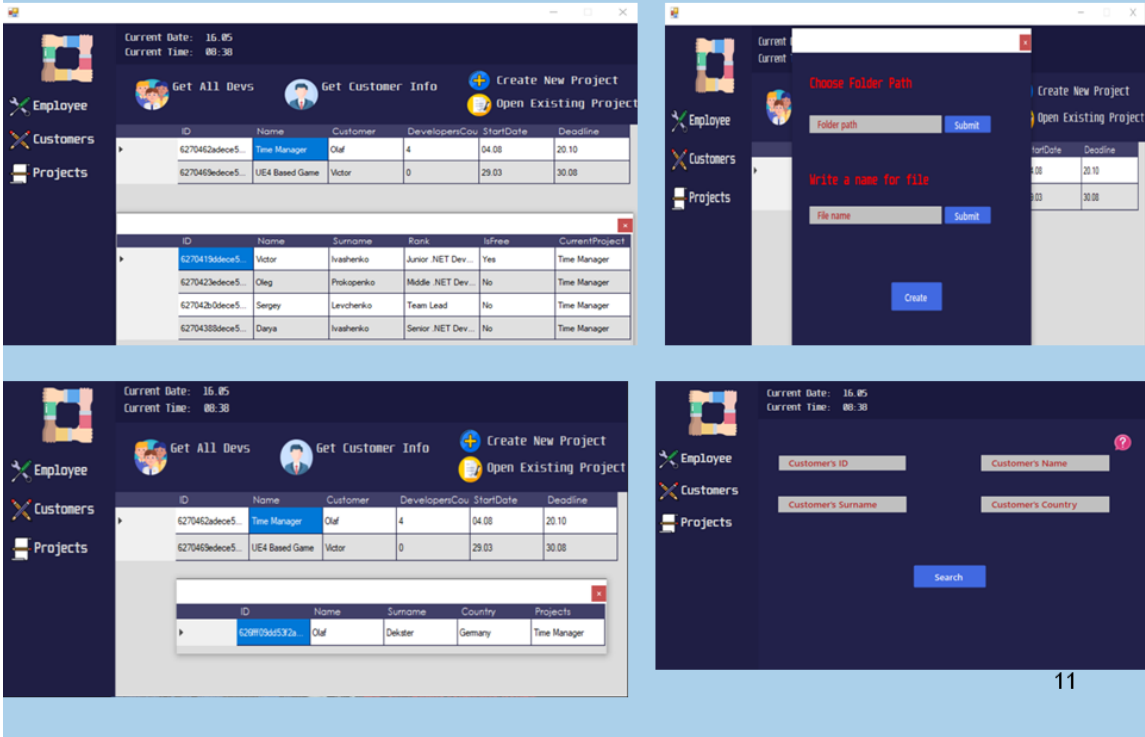


Рисунок В.11 – Слайд 11

Робота з програмою. Інформація про співробітників

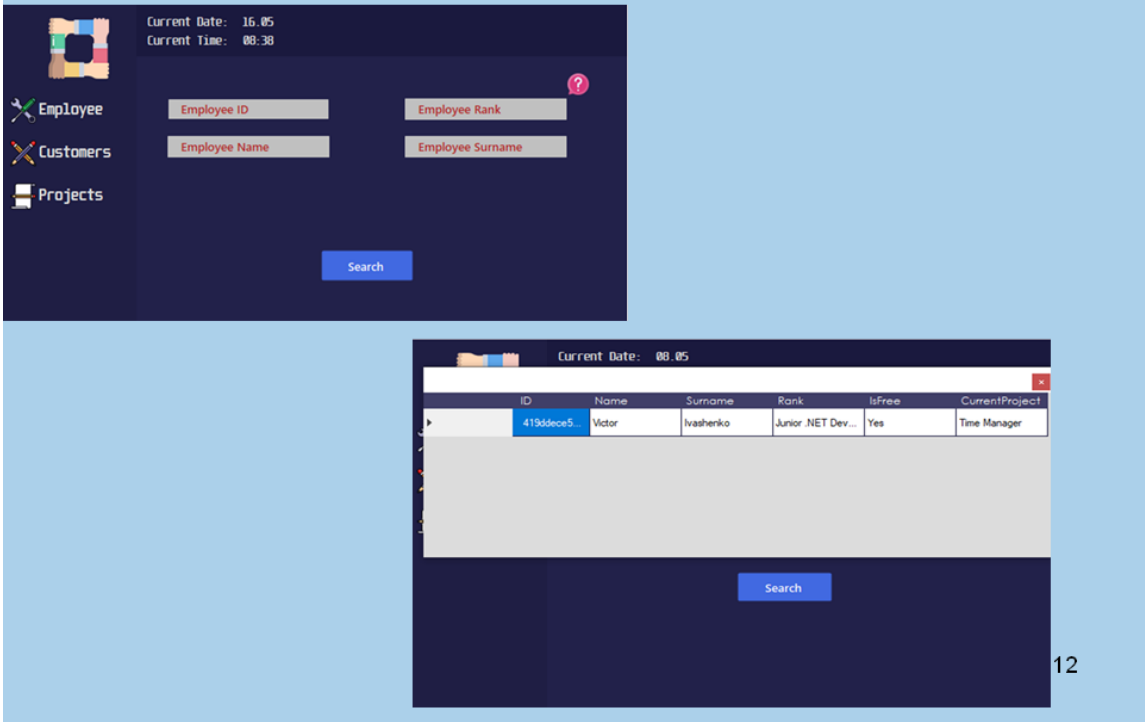


Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи було проаналізовано та досліджено процес розробки програмного забезпечення для керування проектами.

Розроблено програмне забезпечення для керування проектами. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для керування проектами.

Результати тестування програмного забезпечення для керування проектами показали, що розроблена програма може використовуватися за призначенням.

13

Рисунок В.13 – Слайд 13