

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

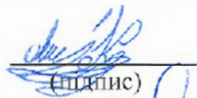
**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: комплексна система інтерактивного тайм - менеджменту

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІ-23-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»

Науковий керівник


(підпис)

Анікеєва Кирила Станіславовича
(прізвище, ім'я, по батькові, науковий ступінь)

Керівник освітньо-
професійної програми

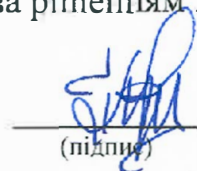

(підпис)

Філоненко М. М., к.ф.-м.н., доцент.
(прізвище, ім'я, по батькові, науковий ступінь)

Горбовий А. Ю., д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь)

Допущено до захисту за рішенням кафедри протокол від «10» січня 2025р № 11

Завідувач кафедри


(підпис)

Філоненко М. М., к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь)

Ірпінь – 2025

Реєстрація № 16

27.01.2025

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

27.01.2025

(дата)



(підпис)

Результат захисту

4 (добре)

(оцінка)

06.02.2025

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

(ініціали, прізвище)

Члени комісії:



(підпис)

А.Ю. Горбовий

(ініціали, прізвище)



(підпис)

М.М. Філоненко

(ініціали, прізвище)



(підпис)

І.Д. Погорєловська

(ініціали, прізвище)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	20		06.02.2025 Франчук В.М. 
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Горбовий А.Ю. 
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		Філоненко М.М. 
4. Рецензент	від 0 до 5	4		Погореловська І.Д. 
РАЗОМ БАЛІВ	від 0 до 50	34		Суховерхівка Л.П. 

Оцінювання кваліфікаційної роботи науковим керівником - 36 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
70	C	4 (добре)	06.02.2025	Франчук В.М.  Горбовий А.Ю.  Філоненко М.М.  Погореловська І.Д.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій

Кафедра комп'ютерних та інформаційних технологій і систем

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: «Інформаційні управляючі системи та технології в економіці»

Затверджую

 Завідувач кафедри
Філоненко М. М.
«23» грудня 2023р.

Завдання на кваліфікаційну роботу здобувачу вищої освіти

Анієсву Кирилу Станіславовичу

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної (магістерської) роботи

«комплексна система інтерактивного тайм - менеджменту»

Затверджена наказом ректора від «20» грудня 2023 р. № 1701

2. Строк здачі закінченої роботи: «07» січня 2025р.

3. Мета роботи: «Проектування комплексної системи інтерактивного тайм - менеджменту»

Об'єкт дослідження: Огляд існуючих стандартів систем тайм менеджменту.

Предмет дослідження: моделі тайм менеджменту.

4. Вихідні дані до роботи: Підбір та обґрунтування систем зберігання даних і менеджменту.

5. Перелік графічного матеріалу: Структурні схеми системи, діаграми потоків даних та макети інтерфейсу користувача.

6. Зміст випускної кваліфікаційної (дипломної) роботи (перелік питань за кожним розділом):

Розділ 1: АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Умова завдання

1.2. Опис предметної області

1.3. Аналіз вимог системи

1.4. Архітектура системи

1.5. Вхідні дані інформаційної системи

1.6. Вихідні дані інформаційної системи

Розділ 2: ПРОЄКТУВАННЯ БАЗИ ДАНИХ

2.1. Моделювання бази даних

2.2. Опис таблиць бази даних

Розділ 3: РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ

- 3.2. Вибір середовища розробки
3.3. Розробка інтерфейсу користувача

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури ДР	До 16.11.2023 р.	16.11.2023 р.
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.03. 2024 р.	26.03. 2024 р.
3	II розділ (аналіз експериментальної частини про і методики досліджень)	20. 06.2024 р.	20. 06.2024 р.
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	15.10.2024 р.	15.10.2024 р.
5	Висновки	20.11.2024 р.	20.11.2024 р.
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	20.01.2025 р.	20.01.2025 р.
7	Отримання відгуку керівника	25.01.2025 р.	25.01.2025 р.
8	Отримання зовнішньої рецензії	27.01.2025р.	27.01.2025р.

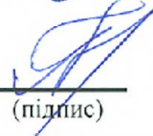
Дата видачі завдання «23» грудня 2023 р.

Керівник дипломної роботи


(підпис)

Філоненко М. М.

Керівник освітньо-професійної програми


(підпис)

Горбовий А. Ю.

Завдання до виконання отримав здобувач вищої освіти


(підпис)

Анікєєв К. С.

АНОТАЦІЯ

Анотація до дипломної роботи "Розробка комплексної системи інтерактивного тайм-менеджменту"

Мета роботи - створення комплексної системи для ефективного управління часом окремими користувачами та командами. Система має стати інструментом для організації роботи, планування завдань та автоматизації процесів, сприяючи підвищенню продуктивності та зниженню стресу.

Проведено аналіз існуючих систем тайм-менеджменту, виявлено їх переваги та недоліки. Сформульовано вимоги до нової системи, що включають підтримку різних типів завдань, інтеграцію з іншими сервісами та платформами, персоналізацію інтерфейсу. Розроблено концептуальну архітектуру системи, що включає базу даних, інтерфейс користувача та алгоритми оптимізації часу.

Створено базу даних та інтуїтивно зрозумілий інтерфейс користувача. Розглянуто питання безпеки та захисту персональних даних, реалізовано механізми шифрування та аутентифікації.

Розроблено інтерактивну систему, що забезпечує ефективне планування, контроль та аналіз виконання завдань. Система інтегрується з популярними календарями та сервісами, надає статистику продуктивності та рекомендації щодо її покращення.

Розроблена система може використовуватися для особистого та командного використання, оптимізуючи робочі процеси та підвищуючи ефективність.

Ключові слова: тайм-менеджмент, інтерактивна система, управління часом, планування, продуктивність, безпека даних.

КЛЮЧОВІ СЛОВА: Тайм-менеджмент, інтерактивна система, управління часом, планування, продуктивність, безпека даних, автоматизація, командна робота, персоналізація, календар.

ABSTRACT

The aim of the work is to create a comprehensive system for effective time management by individual users and teams. The system should serve as a tool for organizing work, planning tasks and automating processes, contributing to increased productivity and reduced stress.

An analysis of existing time management systems was conducted, and their advantages and disadvantages were identified. Requirements for the new system were formulated, including support for various types of tasks, integration with other services and platforms, and interface customization. A conceptual architecture of the system was developed, including a database, user interface, and time optimization algorithms.

A database and an intuitive user interface were created. Issues of security and personal data protection were considered, and encryption and authentication mechanisms were implemented.

An interactive system was developed to provide effective planning, control, and analysis of task execution. The system integrates with popular calendars and services, provides productivity statistics and recommendations for improvement.

The developed system can be used for personal and team use, optimizing workflows and increasing efficiency.

Keywords: time management, interactive system, time management, planning, productivity, data security, automation, teamwork, personalization, calendar.

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1: АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	5
1.1 Умова завдання	5
1.2 Опис предметної області.....	6
1.3 Аналіз вимог системи	8
1.4 Архітектура системи.....	10
1.5 Вхідні дані інформаційної системи	13
1.6 Вихідні дані інформаційної системи.....	14
РОЗДІЛ 2: ПРОЄКТУВАННЯ БАЗИ ДАНИХ	16
2.1 Моделювання бази даних	16
2.2 Опис таблиць бази даних.....	20
РОЗДІЛ 3: РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ.....	25
3.1 Вибір середовища розробки	25
3.2 Розробка архітектури WEB-базованої системи.....	29
3.3 Розробка інтерфейсу користувача	31
ВИСНОВКИ	41
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	43
ДОДАТОК КОД РОЗРОБЛЕНОГО ДОДАТКУ	46

ВСТУП

Сучасний світ характеризується стрімким зростанням інформаційного навантаження, що ставить перед людьми завдання ефективного управління власним часом. Тайм-менеджмент, як ключова навичка для досягнення особистісних і професійних цілей, стає невід'ємною складовою життя. Проте використання традиційних інструментів, таких як паперові планувальники або базові електронні календарі, не завжди забезпечує належний рівень інтерактивності та гнучкості, необхідних для адаптації до швидких змін у середовищі. У цьому контексті створення комплексної системи інтерактивного тайм-менеджменту є актуальним та своєчасним завданням.

Метою цієї магістерської роботи є розробка інтерактивної веб-базованої системи тайм-менеджменту, яка об'єднує сучасні технології для оптимізації управління часом і підвищення продуктивності. Основна увага приділяється створенню інтуїтивно зрозумілого інтерфейсу, зручної системи планування та автоматизації процесів.

Об'єктом дослідження є методи та засоби інтерактивного тайм-менеджменту, а предметом – програмні рішення, архітектурні підходи та алгоритми, що використовуються для реалізації веб-базованих систем управління часом.

У процесі дослідження поставлено такі завдання:

1. Провести аналіз предметної області, дослідити існуючі системи тайм-менеджменту, їх переваги та недоліки.
2. Сформулювати вимоги до системи, включаючи функціональні та нефункціональні аспекти.
3. Розробити архітектуру інтерактивної системи тайм-менеджменту, зокрема проєкт бази даних, яка забезпечує збереження, пошук та обробку інформації.
4. Реалізувати програмну частину, яка включає як серверну, так і клієнтську компоненти, з використанням сучасних технологій.

5. Створити інтерфейс користувача, орієнтований на зручність і доступність.

6. Провести тестування системи для перевірки її функціональності та відповідності вимогам.

Результатом роботи є комплексна інтерактивна система тайм-менеджменту, яка дозволяє користувачам ефективно організовувати свої задачі, планувати час та отримувати аналітику для підвищення продуктивності. Реалізована система може бути використана як індивідуальними користувачами, так і невеликими командами для координації спільної діяльності.

Актуальність роботи обумовлена потребою у високотехнологічних інструментах, здатних враховувати сучасні вимоги до мобільності, інтерактивності та інтеграції з іншими інформаційними системами. Отримані результати можуть бути використані як основа для подальших досліджень у сфері автоматизації тайм-менеджменту та розвитку програмного забезпечення.

РОЗДІЛ 1: АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Умова завдання

Умова завдання визначає основні вимоги до системи, її функціональні та нефункціональні характеристики, а також описує ключові аспекти предметної області, що є основою для її розробки. У сучасному світі зростаючий обсяг інформації, щільні робочі графіки та необхідність одночасного виконання кількох завдань обумовлюють потребу у високоефективних засобах управління часом. Тайм-менеджмент не лише сприяє раціональному використанню робочого часу, а й забезпечує підвищення продуктивності, зниження стресу та поліпшення загальної організації праці [1].

Існуючі рішення у сфері тайм-менеджменту, такі як електронні календарі, менеджери завдань або багатофункціональні платформи (наприклад, Trello, Notion, Microsoft To Do), мають значні переваги, проте часто не враховують індивідуальні потреби користувачів або не забезпечують необхідного рівня інтеграції з іншими системами. Крім того, вони нерідко фокусуються на певних аспектах планування, залишаючи поза увагою ключові функції, такі як автоматизація рутинних процесів, створення гнучких планів, аналітика продуктивності та інтерактивність. У цьому контексті виникає потреба у створенні нової системи, яка буде орієнтована на інтеграцію сучасних технологій, зокрема хмарних сервісів, адаптивного дизайну та алгоритмів штучного інтелекту [2].

Основною умовою розробки інтерактивної системи тайм-менеджменту є забезпечення її універсальності та зручності для широкого кола користувачів. Вона має підтримувати можливість персоналізації для задоволення потреб як індивідуальних користувачів, так і невеликих команд або підприємств.

Крім того, система має бути веб-базованою, що забезпечує доступність з будь-якого пристрою, підключеного до Інтернету. Це дозволить користувачам планувати свій час і відстежувати виконання завдань незалежно від їхнього місця перебування. Важливим аспектом є інтуїтивно зрозумілий інтер-

фейс, який спрощує взаємодію з системою навіть для користувачів, які не мають спеціальних технічних знань.

Ще однією умовою є забезпечення функціональної безпеки та конфіденційності даних користувачів. Система повинна використовувати сучасні методи шифрування, а також дотримуватися міжнародних стандартів захисту персональної інформації, таких як GDPR [3].

Таким чином, умовою завдання є створення комплексної інтерактивної системи тайм-менеджменту, яка має відповідати таким критеріям: підтримка персоналізації, інтеграція з іншими платформами, мобільність, зручність використання, безпека даних і висока продуктивність. Реалізація такого рішення не лише сприятиме покращенню управління часом, а й стане основою для подальшого вдосконалення інструментів організації праці та підвищення ефективності користувачів.

1.2 Опис предметної області

Тайм-менеджмент є багатогранною концепцією, яка включає планування, пріоритезацію завдань, моніторинг виконання та аналіз продуктивності. У сучасному світі це стає критично важливим як для індивідуального розвитку, так і для ефективної роботи команд та організацій.

Предметна область інтерактивного тайм-менеджменту охоплює широкий спектр завдань: від особистого планування до управління проектами. Зокрема, основними функціями є створення, редагування та пріоритезація завдань, їх відображення у вигляді календаря або списку, автоматичні нагадування, інтеграція з іншими системами (наприклад, хмарними сховищами або корпоративними комунікаційними платформами) та генерація звітів про виконану роботу [4].

Сучасні системи тайм-менеджменту діляться на кілька типів. Перший тип – базові електронні календарі, такі як Google Calendar або Microsoft Outlook. Вони забезпечують основні функції планування, зокрема створення подій та нагадувань. Другий тип – додатки для управління завданнями, наприклад, Todoist, Any.do чи Microsoft To Do. Ці інструменти дозволяють ко-

ристувачам створювати списки завдань, встановлювати дедлайни та отримувати сповіщення. Третій тип – багатофункціональні платформи, такі як Trello, Asana чи Notion, які орієнтовані на команди і пропонують можливості для спільної роботи, інтеграції з іншими інструментами та управління складними проектами.

Ключовими проблемами існуючих рішень є обмежена інтерактивність, недостатня гнучкість у налаштуванні, обмеження у функціональності або необхідність використання кількох інструментів одночасно. Наприклад, електронні календарі часто не надають можливості гнучкого управління завданнями або інтеграції з іншими платформами, тоді як платформи для управління проектами можуть бути надто складними для індивідуального використання.

Розробка інтерактивної системи тайм-менеджменту повинна враховувати ці недоліки, об'єднуючи найкращі рішення з різних типів платформ. Важливими компонентами є гнучка архітектура, яка дозволяє інтегрувати нові функції, адаптивний дизайн, що забезпечує комфортну роботу з будь-якого пристрою, а також підтримка багатокористувацького режиму для командної роботи [5,6].

Крім того, предметна область включає сучасні технології, такі як штучний інтелект і машинне навчання, які можуть використовуватися для автоматизації процесів, прогнозування дедлайнів або аналізу продуктивності. Наприклад, система може пропонувати оптимальні розклади на основі історії виконання завдань або визначати, які задачі потребують негайної уваги.

Важливим аспектом є забезпечення безпеки даних і конфіденційності. В умовах сучасного законодавства, такого як GDPR, системи тайм-менеджменту повинні забезпечувати високий рівень захисту персональної інформації, що зберігається та обробляється.

Опис предметної області інтерактивного тайм-менеджменту враховує як функціональні потреби користувачів, так і технічні можливості сучасних технологій. Це дозволяє створити систему, яка забезпечує ефективне управ-

ління часом, підтримує адаптивність до різних сценаріїв використання та пропонує нові інструменти для покращення продуктивності.

1.3 Аналіз вимог системи

Аналіз вимог системи є ключовим етапом у процесі розробки інтерактивної системи тайм-менеджменту. Цей процес включає визначення функціональних і нефункціональних вимог, які забезпечать відповідність системи очікуванням користувачів та стандартам сучасних технологій [7].

Функціональні вимоги визначають, що система повинна виконувати для досягнення поставлених цілей. Основні функціональні вимоги до інтерактивної системи тайм-менеджменту:

1. Реєстрація та аутентифікація користувачів:

- Система повинна підтримувати реєстрацію нових користувачів за допомогою електронної пошти або інтеграції з обліковими записами в соціальних мережах.

- Аутентифікація повинна включати підтримку багатофакторного входу для підвищення безпеки.

2. Створення та управління завданнями:

- Користувачі мають змогу створювати завдання з описом, пріоритетом, дедлайном і категоріями.

- Підтримка зміни статусу завдань (наприклад, "Виконується", "Завершено").

3. Аналітика продуктивності:

- Надання користувачам звітів про виконані завдання, використаний час та продуктивність.

- Графічне відображення статистичних даних (графіки, діаграми).

4. Підтримка командної роботи:

- Користувачі можуть створювати групи та спільно працювати над завданнями.

- Призначення завдань окремим членам команди та моніторинг їх виконання.

Нефункціональні вимоги забезпечують якість роботи системи, її надійність та масштабованість:

1. Продуктивність:

— Система повинна обробляти запити користувачів без затримок, навіть при великій кількості одночасних підключень.

— Час відгуку серверу не повинен перевищувати 200 мілісекунд.

2. Безпека:

— Використання шифрування даних на рівні передачі (TLS/SSL) і зберігання (AES).

— Захист від атак типу SQL-ін'єкції, XSS та CSRF.

3. Доступність:

— Система повинна бути доступною для користувачів з будь-яких пристроїв: настільних ПК, планшетів, смартфонів.

— Підтримка основних браузерів (Chrome, Firefox, Edge, Safari).

4. Масштабованість:

— Можливість обробки зростаючої кількості користувачів без втрати продуктивності завдяки масштабуванню серверної архітектури.

5. Надійність:

— Забезпечення доступності системи 99,9% часу (Service Level Agreement).

— Автоматичне резервне копіювання даних із можливістю їх відновлення.

6. Юзабіліті (зручність користування):

— Інтуїтивно зрозумілий інтерфейс із мінімальною кривою навчання для нових користувачів.

— Локалізація інтерфейсу кількома мовами (наприклад, українська, англійська).

7. Адаптивність:

— Дизайн інтерфейсу повинен адаптуватися до різних розмірів екранів і підтримувати навігацію на сенсорних пристроях.

Технічні вимоги

1. Архітектура системи:

- Використання клієнт-серверної архітектури з чітким розділенням між фронтенд і бекенд частинами.

- Бекенд базується на сучасному веб-фреймворку (наприклад, Node.js, Django).

2. Технології баз даних:

- Реляційна база даних (PostgreSQL, MySQL) із підтримкою ACID-транзакцій.

3. Інфраструктура:

- Розгортання системи у хмарному середовищі (AWS, Azure, Google Cloud) для забезпечення високої доступності.

Аналіз вимог системи охоплює широкий спектр аспектів, спрямованих на створення зручної, функціональної та надійної системи, яка задовольнятиме потреби користувачів та відповідатиме сучасним технологічним стандартам [8].

1.4 Архітектура системи

У цій темі та ряді наступних матеріалів ми розглянемо трирівневу архітектуру програми ASP.NET MVC та її можливу реалізацію.

Взагалі існує безліч різних видів та типів архітектур, які успішно застосовуються. Однією з найбільш використовуваних є класична трирівнева система, яка має на увазі поділ програми на три рівні.

Тут відразу треба сказати, що багаторівневою архітектурою часто позначають два не зовсім пов'язані поняття: n-layer та n-tier. I layer, i tier, зазвичай, позначаються словом "рівень", іноді стосовно "layer" ще вживається слово "шар". Однак у обох випадках рівні будуть різного порядку [9].

Tier представляє фізичний рівень. Тобто якщо ми говоримо про трирівневу архітектуру, то n-tier програма могла бути розділена на такі рівні: сервер бази даних, веб-додаток на веб-сервері та браузер користувача. Тобто кожен рівень представляв би особливий окремий фізичний процес, навіть якби сер-

вер баз даних, і веб-сервер, і браузер користувача знаходилися б на одному комп'ютері. Якби як клієнт альтернативно використовувався мобільний додаток, то це був би ще один фізичний рівень [10].

Layer представляє логічний рівень. Тобто, у нас може бути рівень доступу до даних, рівень бізнес-логіки, рівень уявлення, рівень сервісів тощо. У цьому логічні рівні не збігаються з фізичними. Так, зазвичай рівень надання у додатку ASP.NET містить і контролери, які обробляють введення, і уявлення, що відображаються у веб-браузері, тобто поділяється на два фізичні рівні [11].

В даному випадку ми говоритимемо саме про логічні рівні, тобто про n-layer архітектуру.

Класична трирівнева система складається з наступних рівнів (рис. 1.1):

Three-Tier (or Three-Layer) Architecture in Spring MVC

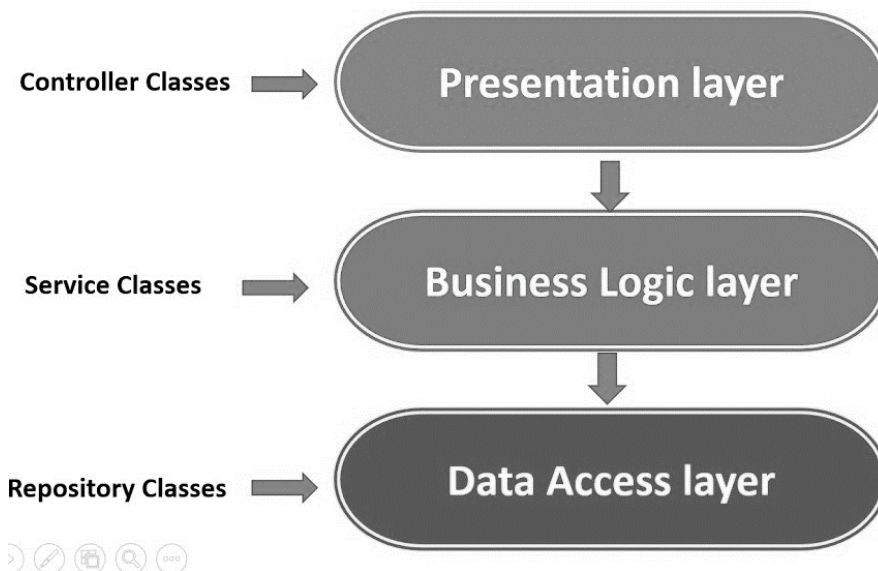


Рис. 1.1 Архітектура MVC

Presentation layer (рівень вистави): це той рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти інтерфейсу користувача, механізм отримання введення від користувача. Стосовно asp.net mvc на даному рівні розташовані уявлення і всі ті компоненти, які складають інтер-

фейс користувача (стилі, статичні сторінки html, javascript), а також моделі уявлень, контролери, об'єкти контексту запиту.

Business layer (рівень бізнес-логіки): містить набір компонентів, які відповідають за обробку отриманих від рівня уявлень даних, реалізує всю необхідну логіку програми, всі обчислення, взаємодіє з базою даних та передає рівню представлення результат обробки [12].

Data Access layer (рівень доступу до даних): зберігає моделі, що описують використовувані сутності, також розміщуються специфічні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут також зберігаються репозиторії, якими рівень бізнес-логіки взаємодіє з базою даних [13].

При цьому треба зазначити, що крайні рівні не можуть взаємодіяти між собою, тобто рівень уявлення (стосовно ASP.NET MVC, контролери) не можуть безпосередньо звертатися до бази даних і навіть рівня доступу до даних, а тільки через рівень бізнес-логіки .

Рівень доступу до даних залежить від інших рівнів, рівень бізнес-логіки залежить від рівня доступу до даних, а рівень уявлення - від рівня бізнес-логіки.

Компоненти, як правило, повинні бути слабко пов'язаними (loose coupling), тому невід'ємною ланкою багаторівневих додатків є використання залежностей [14].

При чому про ASP.NET MVC ми говоримо насамперед стосовно рівня уявлення, інші рівні можуть бути реалізовані незалежно і можуть використовуватися в додатках на інших технологіях, як Windows Forms, WPF і т.д. І, як правило, вся програма в цілому буде представляти рішення (solution) у Visual Studio, а окремі рівні – проекти. У той же час невірно вважати, що якщо рівень обов'язково має відповідати окремому проекту. При необхідності ми можемо роздробити один рівень на кілька проектів, головне, щоб його функціонал представляв єдину логічну ланку.

1.5 Вхідні дані інформаційної системи

Вхідні дані інформаційної системи відіграють важливу роль у забезпеченні її функціональності та відповідають за коректне виконання основних процесів. Для інтерактивної системи тайм-менеджменту вхідні дані поділяються на декілька категорій, кожна з яких забезпечує виконання різних функцій системи.

Основні типи вхідних даних:

1. Дані користувачів:

- Ім'я, прізвище, адреса електронної пошти (під час реєстрації).
- Пароль користувача (із подальшим хешуванням).
- Налаштування профілю (мовні уподобання, часовий пояс, методи сповіщень).

2. Дані завдань і подій:

- Назва завдання або події.
- Опис завдання (наприклад, деталі, важливі для виконання).
- Дата та час початку/завершення.
- Пріоритет завдання (низький, середній, високий).
- Категорія або мітка (наприклад, робота, особисте, навчання).
- Стан завдання (нове, виконується, завершено).

3. Дані календаря:

- Дати й часові інтервали для подій.
- Періодичність подій (одноразові або повторювані).
- Дані про синхронізовані події з зовнішніх календарів.

4. Командні дані (для спільної роботи):

- Списки членів команди, пов'язаних із груповими завданнями.
- Ролі членів команди (відповідальний, виконавець, спостерігач).
- Завдання, призначені окремим користувачам.

5. Дані для аналітики:

- Інформація про виконані завдання (час завершення, витрачений час).

- Пріоритети й статуси завдань для побудови звітів.

Джерела вхідних даних:

- Ручне введення: Користувачі вводять інформацію через інтерфейс системи (наприклад, додають нове завдання чи подію).

- Синхронізація із зовнішніми джерелами: Імпорт подій із зовнішніх календарів чи завдань з інших систем управління проєктами.

- Автоматичне генерування: Система може автоматично створювати нагадування або події на основі заданих правил (наприклад, автоматичне повторення подій).

Вхідні дані є основою роботи інформаційної системи, забезпечуючи належний рівень інтерактивності, автоматизації та персоналізації для кожного користувача. Правильна обробка цих даних є ключовою для коректного функціонування системи й досягнення її основних цілей [15].

1.6 Вихідні дані інформаційної системи

Вихідні дані інформаційної системи є результатом її роботи та забезпечують користувачів інформацією, необхідною для управління часом, контролю виконання завдань і підвищення продуктивності. Ці дані генеруються на основі вхідних даних, оброблених відповідно до заданих алгоритмів, і представлені у зручному для користувача вигляді.

Основні типи вихідних даних:

1. Дані про завдання та події:

- Списки завдань із їхніми статусами (нове, виконується, завершено).

- Інформація про заплановані події з датами, часом і деталями.

- Нагадування про завдання, які наближаються до дедлайну.

2. Календарні дані:

- Візуалізація завдань

3. Аналітичні дані:

- Звіти про виконані завдання за певний період (день, тиждень, місяць).

- Графіки та діаграми, що відображають продуктивність користувача або команди.

- Статистика використання часу: частка завершених завдань, час, витрачений на виконання кожної категорії завдань.

4. Дані для командної роботи:

- Стан виконання групових завдань (хто відповідає, який статус).

- Звіти про внесок кожного члена команди у виконання проєкту.

- Сповіщення про нові чи змінені завдання для всієї команди.

Вихідні дані інтерактивної системи тайм-менеджменту охоплюють широкий спектр інформації, необхідної для ефективного управління часом та аналізу продуктивності. Вони формуються у різних форматах для максимального зручності та врахування потреб користувачів [16].

РОЗДІЛ 2: ПРОЄКТУВАННЯ БАЗИ ДАНИХ

2.1 Моделювання бази даних

Процес проєктування бази даних проходить в декілька етапів, аналіз предметної області та мовне описання інформаційної структури предметної області до формалізованого опису об'єкту предметної області [17].

- аналіз предметної області;
- концептуальне проєктування;
- логічне проєктування;
- фізичне проєктування;

Аналіз предметної області передбачає процес аналізу та мовного описання предметної області, визначення всіх зв'язків між об'єктами, визначення та опис атрибутів.

Атрибут – це інформаційне відображення властивостей об'єкта. При реалізації інформаційної моделі на будь-якому носії інформації, атрибут часто називають елементом даних, полем даних або просто полем.

Зв'язок – це з'єднання між двома або більше сутностями. Кожен зв'язок реалізується через значення атрибутів сутності. Зв'язки поділяються на три типи:

- один-до-одного;
- один-до-багатьох;
- багато-до-багатьох;

Зв'язок “один-до-одного” - при такому типові зв'язку одного запису в першій таблиці відповідає лише одна запис в іншій таблиці. В цьому випадку слід перевірити можливість розміщення всіх записів в одній таблиці. Проте у ряді випадків можна використовувати декілька простіших таблиць. Відповідність записів встановлюється по полю, яке є первинним ключем в першій таблиці, і полю, званим зовнішнім ключем іншої таблиці [18].

Зв'язок “один-до-багатьох” - в цьому випадку запис однієї таблиці може мати декілька погоджених з нею записів в іншій таблиці. При цьому кожен запис в другій таблиці узгоджується лише з одним записом в першій

таблиці. Наприклад, кожен покупець може купити декілька товарів, але кожен проданий товар має лише одного покупця. Поле, що містить первинний ключ нової таблиці, зв'язується із зовнішнім ключем старою. Значення в полі із зовнішнім ключем можуть повторюватися.

Зв'язок “багато-до-багатьох” - кожному запису з однієї таблиці може відповідати будь-яка кількість записів в іншій таблиці і навпаки. Наприклад, кожна людина може дзвонити з декількох телефонів. З іншого боку деякими телефонами можуть користуватися декілька чоловік. В цьому випадку поля, по яких встановлюється зв'язок, є зовнішніми ключами. Вони можуть містити значення, що повторюються [19].

Сутність – це унікальний об’єкт предметної області. Кожна сутність повинна містити унікальний ідентифікатор, володіє певною кількістю атрибутів, створених в етапі аналізу предметної області. Кількість зв’язків між сутностями може бути безліч [20].

Процес проєктування починається з виявлення елементів та групування в набори даних. При створенні сутностей першим етапом визначають атрибути, які характеризують дану сутність, та створення залежностей.

На рис. 2.1 відображена ER діаграма бази даних, на якій присутні наступні сутності:

- User – користувач системи, що зберігає в собі усі атрибути, що присутні клієнту чи працівнику;
- Role – роль чи спеціальність користувача в системі. Завдяки даній сутності є можливим відрізнити клієнта від працівника та чітко розподілити обов’язки між користувачами в системі;
- Project – проєкт програмного забезпечення, що буде розроблятися;
- ProjectType – тип проєкту для розробки;
- ProjectCommand – команда програмістів що розроблює проєкт;

- Task – окреме завдання для розробки програмного забезпечення, являється частиною проекту;
- TaskCommand – Команда що сформована на основі команди проекту, але для вирішення конкретного завдання при розробці.

Концептуальне проєктування процес створення концептуальної моделі, на якому визначаються об'єкти, зв'язки між об'єктами, атрибути, ключові атрибути.

Концептуальна модель бази даних – це модель, представлена безліччю понять і зв'язків між ними, що визначають смислову структуру розглянутої предметної області або її конкретного об'єкта [21].

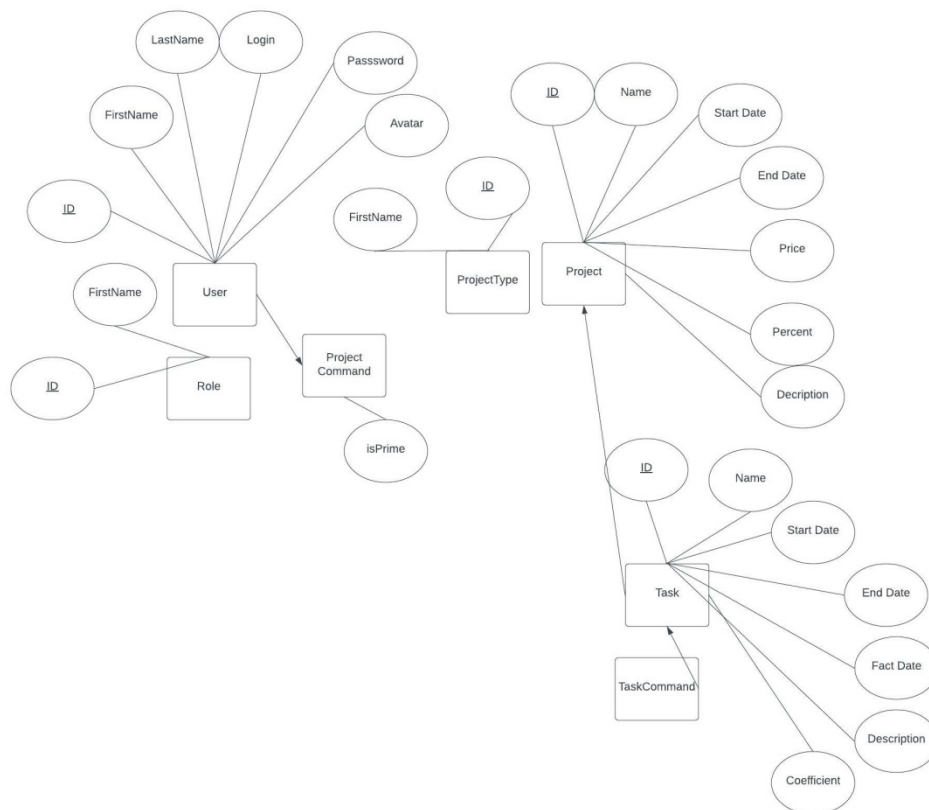


Рис. 2.1 ER діаграма бази даних

Логічне проєктування — створення схеми бази даних на основі конкретної моделі даних, наприклад, реляційної моделі БД. Для реляційної моделі даних — це набір схем відносин, зазвичай із зазначенням первинних ключів, а також «зв'язків» між відносинами, що представляють собою зовнішні ключі [23].

Перетворення концептуальної моделі в логічну модель, як правило, здійснюється за формальними правилами. Цей етап може бути в значній мірі автоматизований.

На етапі логічного проєктування враховується специфіка конкретної моделі даних, але може не враховуватися специфіка конкретної СУБД [22].

Логічна модель бази даних зображена на рис. 2.2.

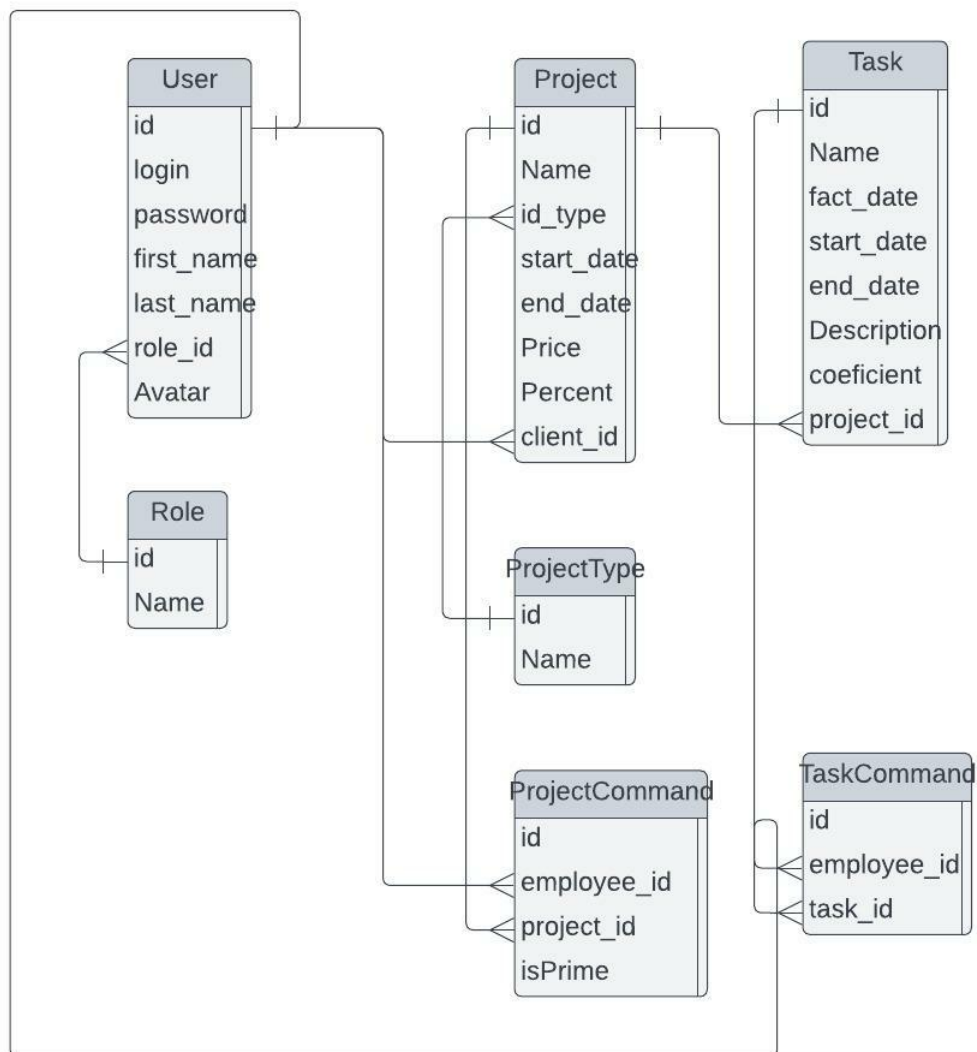


Рис. 2.2 Логічна модель бази даних

Фізичне проєктування — створення схеми бази даних для конкретної СУБД. Специфіка конкретної СУБД може включати в себе обмеження на іменування об'єктів бази даних, обмеження на підтримувані типи даних та інші. Крім того, специфіка конкретної СУБД при фізичному проєктуванні включає вибір рішень, пов'язаних з фізичним середовищем зберігання даних

(вибір методів управління дисковою пам'яттю, поділ БД по файлам і пристроям, методів доступу до даних), створення індексів та інші [24].

Фізична модель бази даних зображена на рис. 2.3.

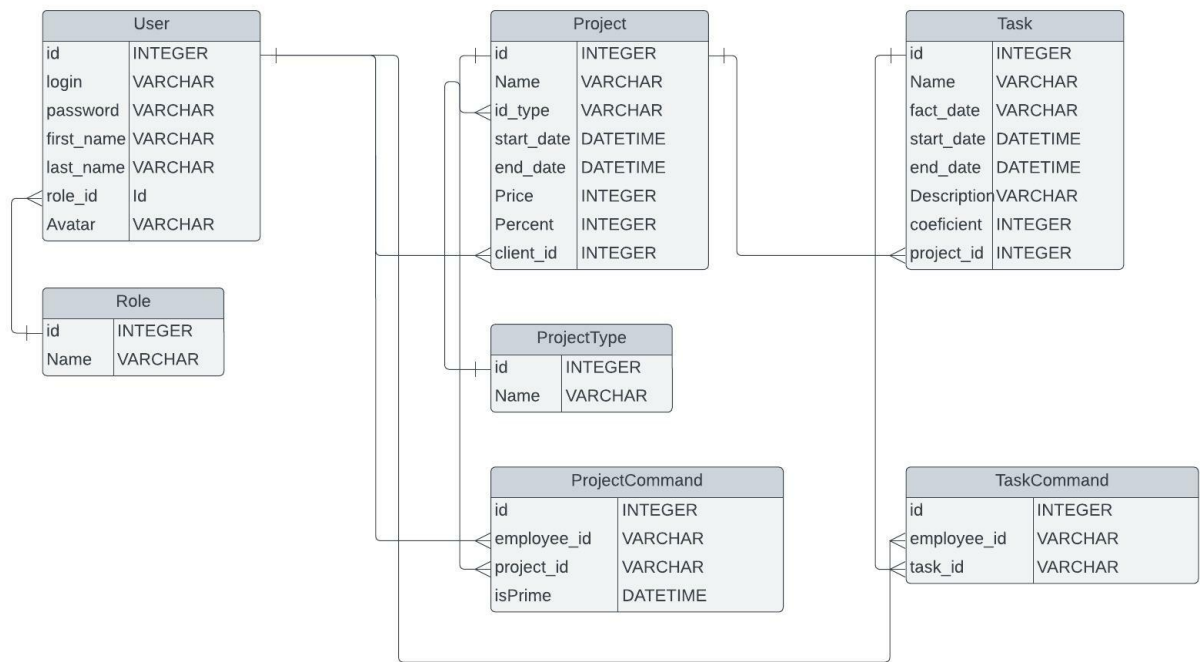


Рис. 2.3 Фізична модель бази даних

2.2 Опис таблиць бази даних

Таблиця «User» містить інформацію про зареєстрованих користувачів (клієнтів, працівників).

Таблиця 2.1

User

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифіка- тор користувача
login	VARCHAR	Логін користувача
password	VARCHAR	Пароль користувача
avatar	VARCHAR	Аватар користувача
first_name	VARCHAR	Ім'я користувача
last_name	VARCHAR	Прізвище користувача
role_id	INTEGER	Ідентифікатор ролі ко-

		ристувача
--	--	-----------

Таблиця «Role» містить інформацію про ролі користувача.

Таблиця 2.2

Role

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор ролі
name	VARCHAR	Назва ролі

Таблиця «ProjectType» містить інформацію про тип проекту.

Таблиця 2.3

ProjectType

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор статі
name	VARCHAR	Назва статі

Таблиця «Project» містить інформацію про проект.

Таблиця 2.4

Project

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор типу
name	VARCHAR	Назва типу
start_date	DATETIME	Дата початку виробництва

end_date	DATETIME	Кінцева дата виробництва
percent	INTEGER	Процент виконаних робіт
type_id	INTEGER	Тип проєкту
price	INTEGER	Ціна проєкту
client_id	INTEGER	Ідентифікатор замовника
description	VARCHAR	Опис проєкту

Таблиця «Task» містить інформацію про завдання проєкту.

Таблиця 2.5

Task

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор завдання
project_id	INTEGER	Ідентифікатор проєкту
employee_id	INTEGER	Ідентифікатор програміста
start_date	DATETIME	Дата початку завдання
end_date	DATETIME	Дата завершення завдання
fact_date	DATETIME	Фактична дата завершення завдання
coefficient	INTEGER	Коефіцієнт складності

		завдання
--	--	----------

Таблиця «ProjectCommand» містить інформацію про команду проєкту.

Таблиця 2.6

ProjectCommand

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор команди
project_id	INTEGER	Ідентифікатор проєкту
employee_id	INTEGER	Ідентифікатор працівника
isprime	DATETIME	Флаг, чи являється працівник відповідальним за проєкт

Таблиця «TaskCommand» містить інформацію про команду завдання.

Таблиця 2.7

TaskCommand

Поле таблиці	Тип даних	Опис поля
id	INTEGER	Унікальний ідентифікатор команди завдання
task_id	INTEGER	Ідентифікатор завдання

employee_id	INTEGER	Ідентифікатор працівника
-------------	---------	--------------------------

РОЗДІЛ 3: РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ

3.1 Вибір середовища розробки

Веб-додаток було розроблено в середовищі Visual Studio

Microsoft Visual Studio — один з продуктів компанії Майкрософт, який представляє з себе інтегровану середу розробки програмного забезпечення і ряд інших інструментальних засобів. Даний продукт дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в не керованому, так і керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight [25].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня.

Решта вбудовуваних інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду, додавання нових наборів інструментів, наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів циклу розробки програмного забезпечення. Загальний вигляд вікна, яке відображається при завантаженні Visual Studio, представлено на рис. 3.1[26]

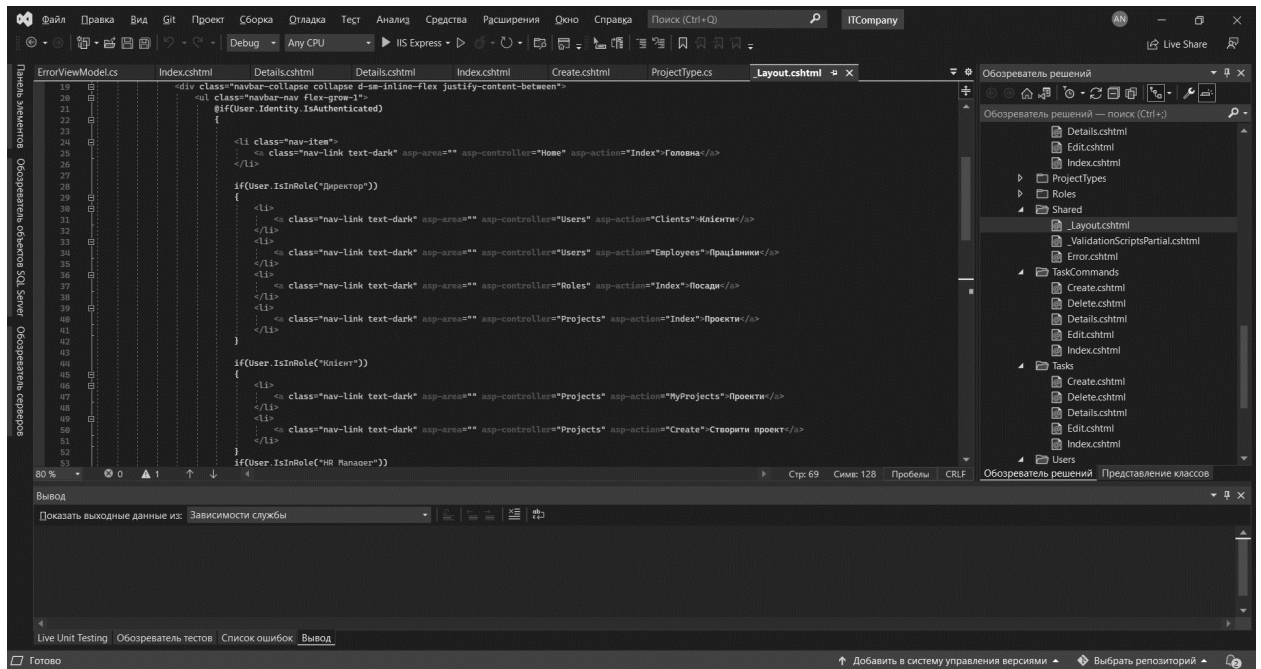


Рис. 3.1. Вікно з проектом у Visual Studio

Мова C#[6,7] — це чергова сходинка нескінченної еволюції мов програмування.

Її створення викликано процесом вдосконалення і адаптації, який визначав розробку комп'ютерних мов протягом останніх років. Подібно до всіх успішних мов, які побачили світ раніше, C# спирається на минулі досягнення мистецтва програмування, яке постійно розвивається.

У мові C# (створеному компанією Microsoft для підтримки середовища .NET Framework) перевірені часом засоби вдосконалені за допомогою найсучасніших технологій. C# надає дуже зручний і ефективний спосіб написання програм для сучасного безпечного середовища обчислювальної обробки даних, яка включає операційну систему Windows, Internet та інші компоненти. В процесі становлення мові C# "дісталася" багата спадщина. Вона — прямий нащадок двох найбільш успішних мов програмування (C і C++) [27].

Entity Framework являє собою спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. На відміну від засобів ADO.NET які дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і пра-

цювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами мови C# [28].

Центральною концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями. Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людину, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік, вага. Властивості необов'язково представляють прості дані типу int, а й можуть представляти більш комплексні структури даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативним зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бази даних, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД.

Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення.

На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних.

Рівень зіставлення служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних [29].

Microsoft SQL Server — це система управління базами даних, запропонована Microsoft. Мова, що використовується для написання SQL-запитів Transact-SQL, створений спільно Microsoft і Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO для структурованої мови запитів SQL з розширеннями. Він використовується як для малих, так і середніх баз даних, а також для великих корпоративних баз даних. Протягом багатьох років вона успішно конкурує з іншими системами управління базами даних.

Transact-SQL є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис збережених процедур і забезпечує підтримку транзакцій (взаємодія з базою даних з менеджером додатків). Microsoft SQL Server та Sybase ASE для мережових інтерфейсів використовують протокол прикладного рівня під назвою Табличний потік даних (TDS, табличний протокол передачі даних) [30].

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC), інтерфейс для взаємодії з додатками СУБД. SQL Server надає можливість підключення користувачів через сервіси веб-серверів, які використовують протокол SOAP. Це дозволяє клієнтським програмам, які не призначені для крос-платформних Windows, підключатися до SQL Server.

Корпорація Майкрософт також випустила сертифікований драйвер JDBC, що дозволяє підключати Java-програми (такі як BEA та IBM Websphere) до Microsoft SQL Server [31].

SQL Server підтримує дзеркальні та кластерні бази даних. Кластер SQL Server це набір ідентичних серверів; Ця схема допомагає розподілити навантаження між декількома серверами. Всі сервери мають одне віртуальне ім'я, а дані розподіляються відповідно до IP-адрес кластерних машин протягом ро-

бочого циклу. Крім того, у разі відмови або відмови на одному з кластерних серверів можливе автоматичне перенесення завантаження на інший сервер.

SQL Server має вбудовану підтримку .NET Framework. Завдяки цьому, збережені процедури бази даних можуть бути написані на будь-якій мові платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і створює елементи керування SQL Server без використання вбудованих інструментів Windows. Це покращує продуктивність за звичайних алгоритмів Windows, оскільки алгоритми розподілу ресурсів спеціально налаштовані для використання в структурах SQL Server

3.2 Розробка архітектури WEB-базованої системи

У ході розробки додатку було використано фреймворк Entity Framework який використовує принципи MVC [32]. Тому архітектура проєкту виглядає наступним чином (рис. 3.1):

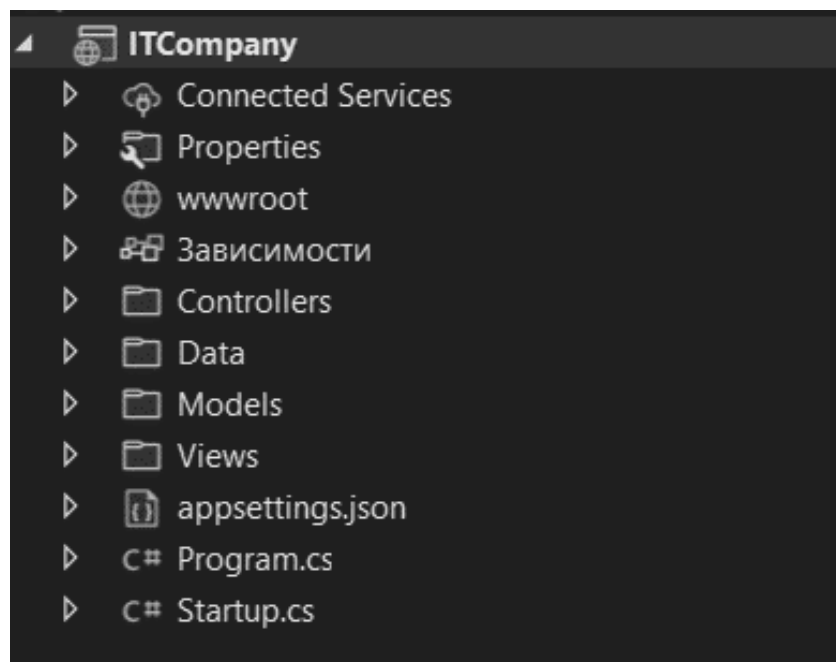


Рис. 3.1 Структура додатку

Папка wwwroot зберігає в собі файли зображень, веб бібліотек додатку (рис. 3.2).

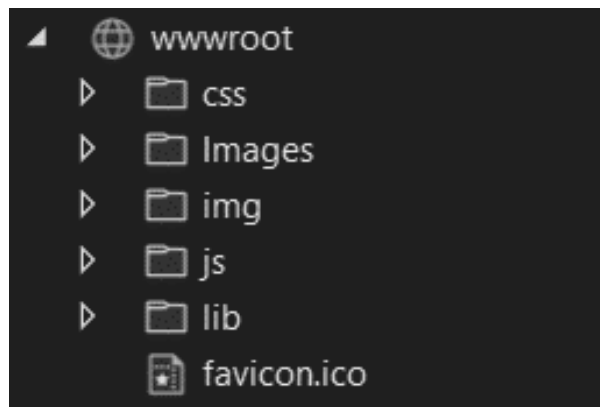


Рис. 3.2 Вміст папки wwwroot

Папка Controllers зберігає в собі контролери для доступу до даних з бази даних (рис. 3.3)

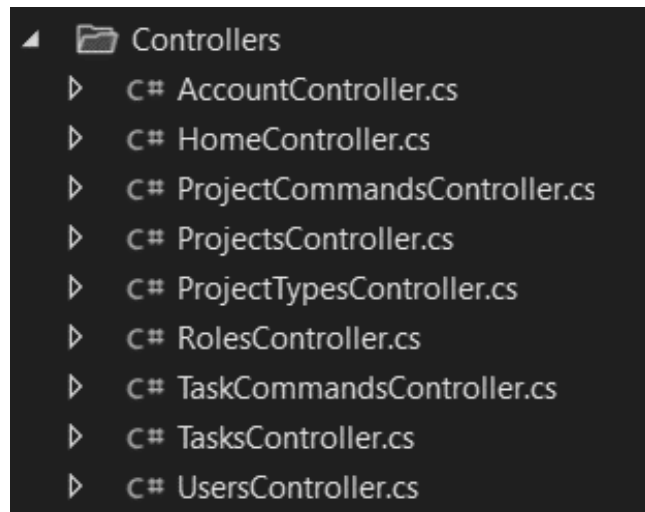


Рис. 3.3 Вміст папки Controllers

Папка Data – зберігає у собі файли ініціалізації бази даних (рис. 3.4).

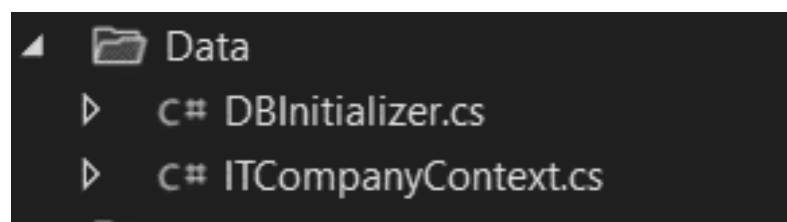


Рис. 3.4 Вміст папки Data

Models – зберігає класи моделей (сутностей) додатку (рис. 3.5).

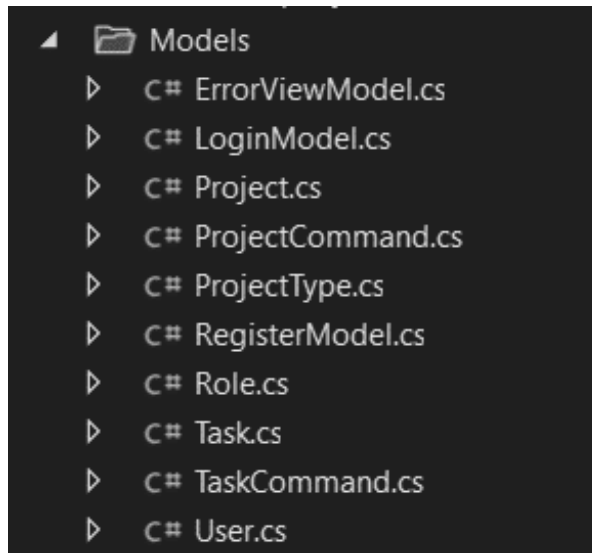


Рис. 3.5 Вміст папки Models

Views – зберігає у собі html сторінки інтерфейсу користувача (рис. 3.6).

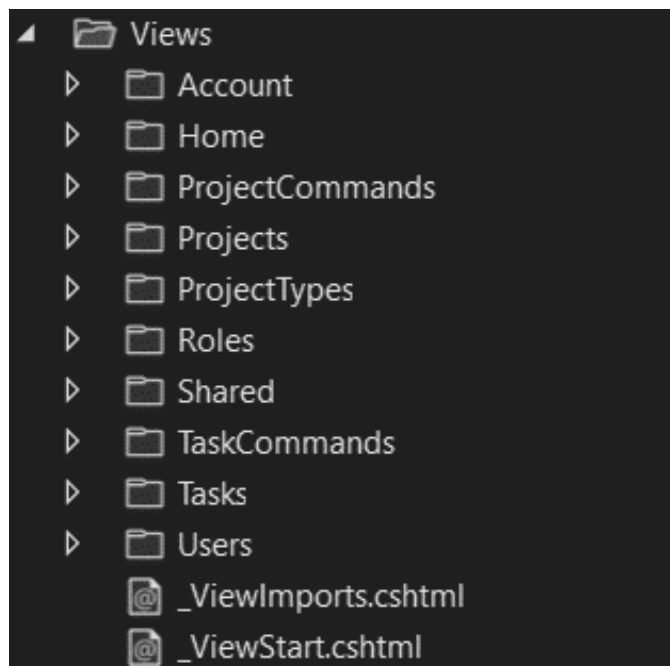


Рис. 3.6 Вміст папки Views

Клас Startup.cs – зберігає усі налаштування додатку, так як конфігурації доступу до бази даних, авторизацію, перевірку помилок на сервері тощо.

Клас Program.cs – основний клас, що запускає веб додаток.

Лістинг програмного коду представлено у додатку А.

3.3 Розробка інтерфейсу користувача

Робота користувача з додатком починається з головної сторінки сайту (рис. 3.7), в шапці сайту розташована навігація для переходу на різні сторін-

ки сайту. В основній частині головної сторінки знаходиться карусель з зображеннями сервісу

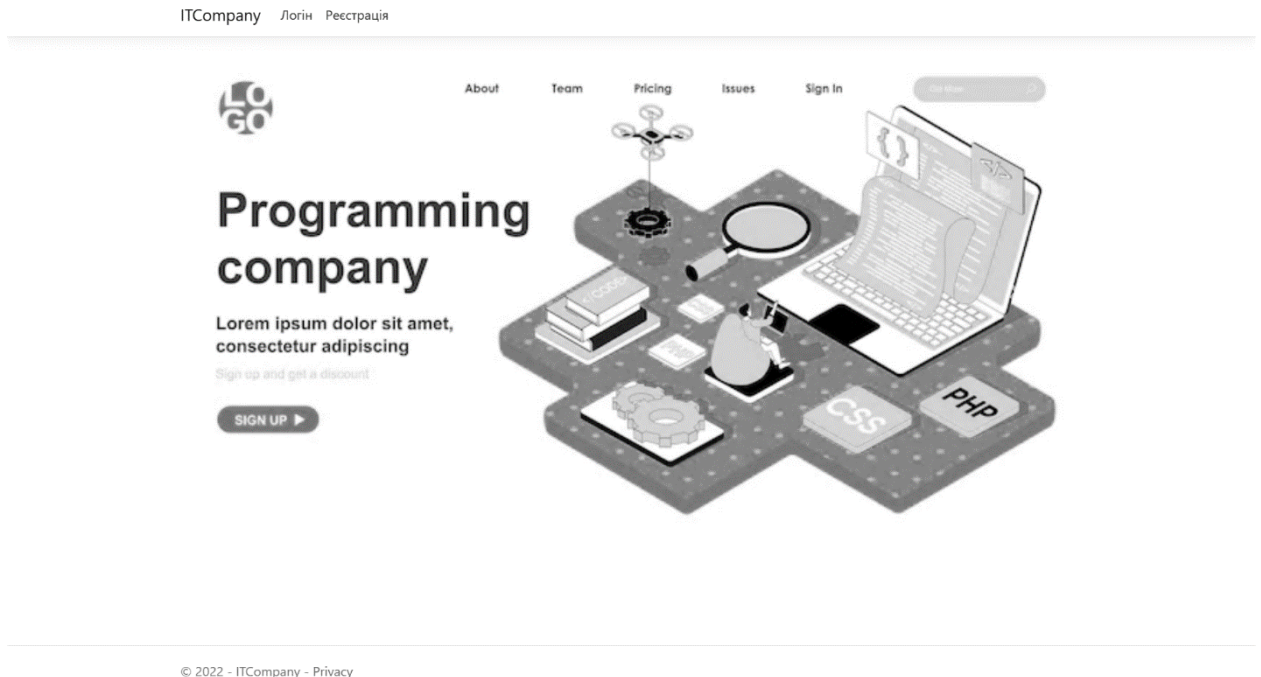


Рис. 3.7 Головна сторінка сайту

При натисканні кнопки під назвою «Логін», користувач потрапляє на сторінку авторизації, або при натисканні кнопки «Реєстрація» потрапляє на сторінку реєстрації. Сторінки авторизації та реєстрації відображені на рис. 3.8 та 3.9.

Вхід на сайт

Логін

Пароль

Війти

Рис. 3.8 Форма авторизації.

ITCompany Логін Реєстрація

Реєстрація

Ім'я

Прізвище

Стать

HR Manager ▾

Логін

Пароль

Повторіть пароль

Аватар

Обзор... Файл не выбран.

Реєстрація

Рис. 3.9 Форма реєстрації.

При натисканні на кнопку «Клієнти» - відображається список клієнтів компанії (рис. 3.10)

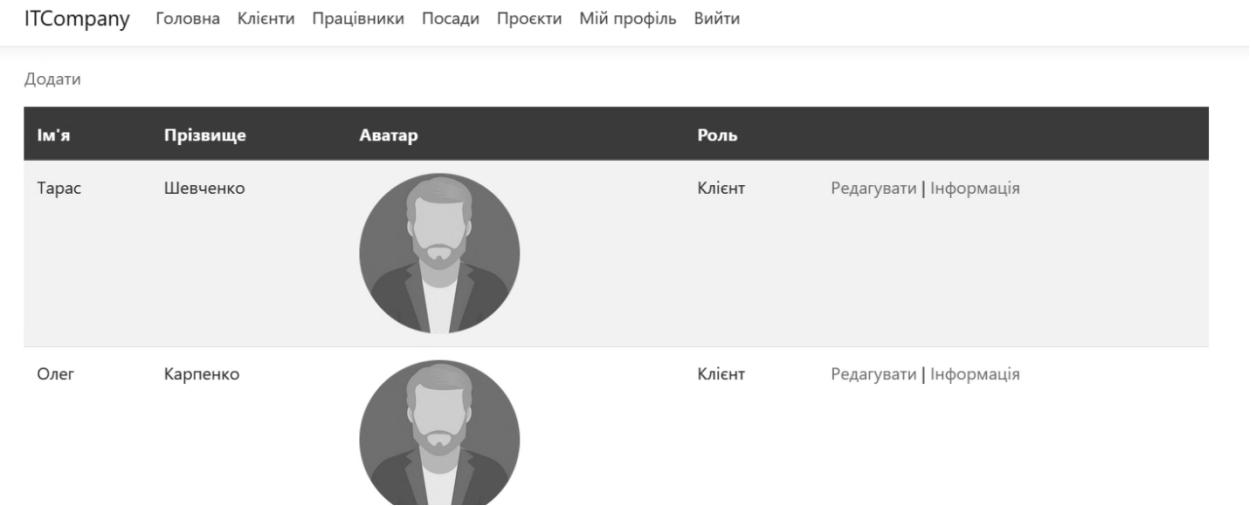


Рис. 3.10 Список клієнтів компанії

При натисканні на назву посилання «Інформація» - відривається сторінка повної інформації щодо клієнта (рис. 3.11) на якій відображається список усіх замовлень які робив даний клієнт.

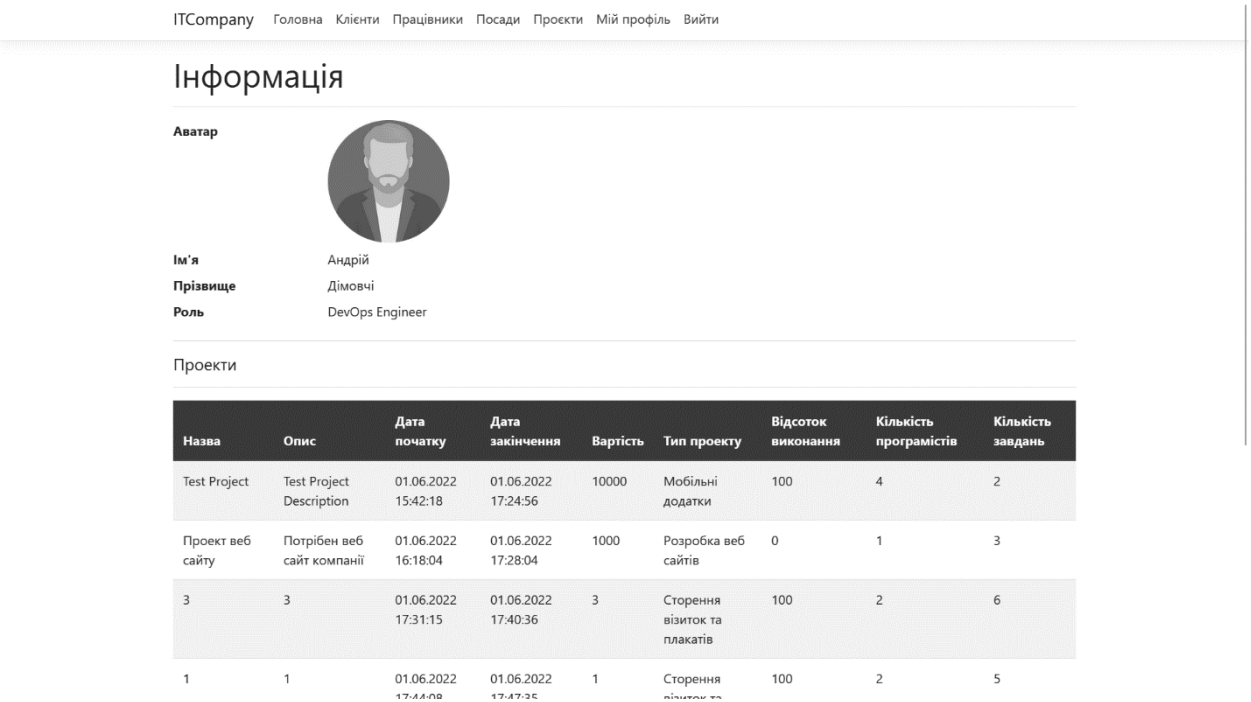


Рис. 3.11 Сторінка інформації про комп'ютер

На даній сторінці для працівника також відображатимуться завдання та проекти у яких він приймав участь.

Для того, щоб створити замовлення на розробку програмного забезпечення, необхідно авторизуватись у системі під роллю «Клієнт» та натиснути на кнопку «Створити проект» і заповнити необхідну форму (рис. 3.12)

Додати

Назва

Проект для відображення у звіті

Опис

Новий проект який треба прийняти!

Ціна

1850

Тип проекту

Web програмування

Створити

Рис. 3.11 Форма створення нового замовлення на розробку програмного забезпечення

Після успішного створення замовлення, директор побачить активне замовлення у списку проектів (рис. 3.12)

Назва	Опис	Дата початку	Дата закінчення	Вартість	Тип проекту	Відсоток виконання	Кількість програмістів	Кількість завдань	
Test Project	Test Project Description	01.06.2022 15:42:18	01.06.2022 17:24:56	10000	Мобільні додатки	100	4	2	Інформація
Проект веб сайту	Потрібен веб сайт компанії	01.06.2022 16:18:04	01.06.2022 17:28:04	1000	Розробка веб сайтів	0	1	3	Інформація
3	3	01.06.2022 17:31:15	01.06.2022 17:40:36	3	Сторення візиток та плакатів	100	2	6	Інформація
1	1	01.06.2022 17:44:08	01.06.2022 17:47:35	1	Сторення візиток та плакатів	100	2	5	Інформація
MiXzo PROJECT 2M	Test Project Description	01.06.2022 17:54:19	-	10000	Сторення візиток та плакатів	28	0	7	Інформація
Test Project For Employee	Test Project For Employee	01.06.2022 18:08:38	-	100	Мобільні додатки	66	2	3	Інформація
Новий проект для відображення у звіті	Новий проект який необхідно прийняти!	-	-	1850	Web програмування	0	0	0	Інформація

Рис. 3.12 – Список проектів

Після чого, директор переглядає інформацію про проєкт (рис. 3.13) і може його прийняти.

Інформація

Назва	Новий проєкт для відображення у звіті!
Опис	Новий проєкт який необхідно прийняти!
Дата початку	-
Дата закінчення	-
Вартість	1850
Тип проєкту	Web програмування

Прийняти

Рис. 3.13 – Інформація про заказ

Після натискання на кнопку «Прийняти» дата початку проєкту стає з поточним значенням, та додаються дві кнопки: «Додати члена команди» та «Додати задачу» (рис. 3.14) яка переводить нас на сторінку додавання програмістів у команду розробників програми (рис. 3.15), чи додавання нового завдання до програми (рис. 3.16)

ITCompanу Головна Клієнти Працівники Посади Проєкти Мій профіль Вийти

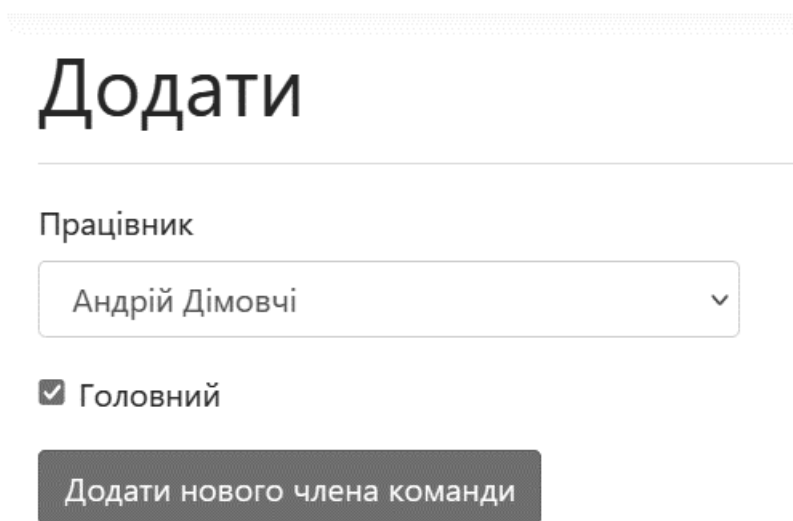
Інформація

Назва	Новий проєкт для відображення у звіті!
Опис	Новий проєкт який необхідно прийняти!
Дата початку	01.06.2022 20:26:16
Дата закінчення	-
Вартість	1850
Тип проєкту	Web програмування

Додати члена команди

Додати задачу

Рис. 3.14 – Сторінка інформації про активний проект.



Додати

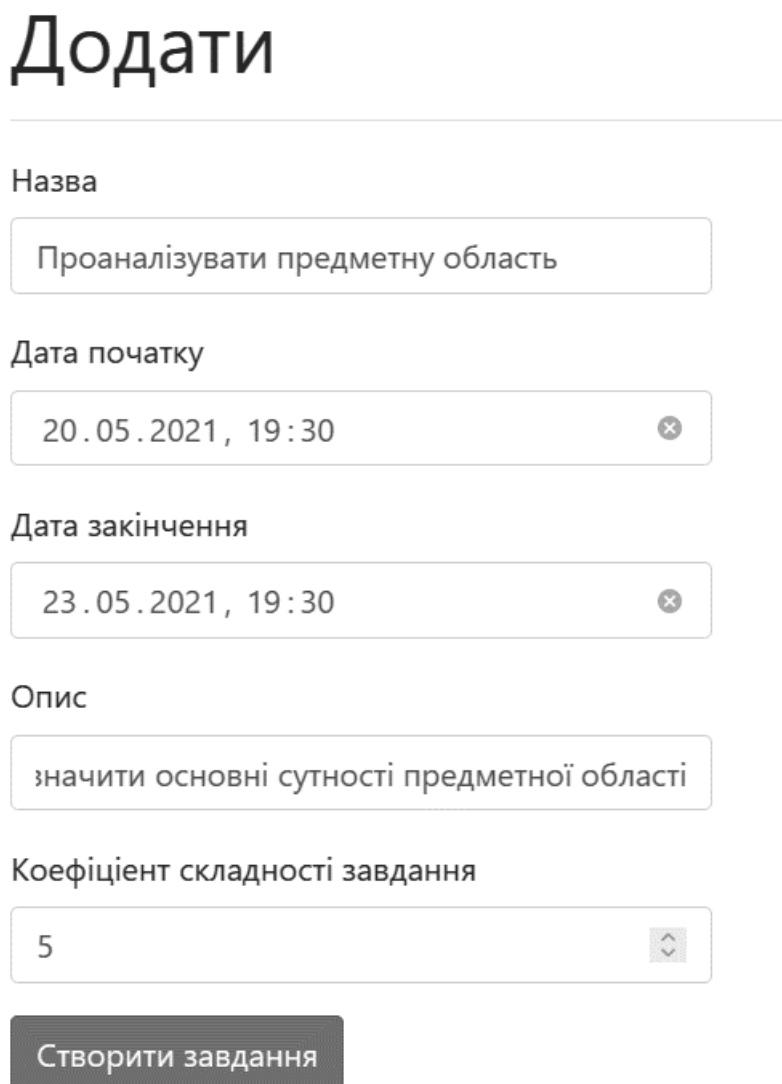
Працівник

Андрій Дімовчі

☒ Головний

Додати нового члена команди

Рис. 3.15 – Сторінка додавання програміста у команду розробника.



Додати

Назва

Проаналізувати предметну область

Дата початку

20.05.2021, 19:30

Дата закінчення

23.05.2021, 19:30

Опис

значити основні сутності предметної області

Коефіцієнт складності завдання

5

Створити завдання

Рис. 3.16 – Сторінка додавання нового завдання у процес розробки проекту.

Після заповнення усіх даних проекту. Сторінка інформації про проект виглядає наступним чином (рис. 3.17). На даній сторінці відображається команда розробників та список завдань для виготовлення проекту.

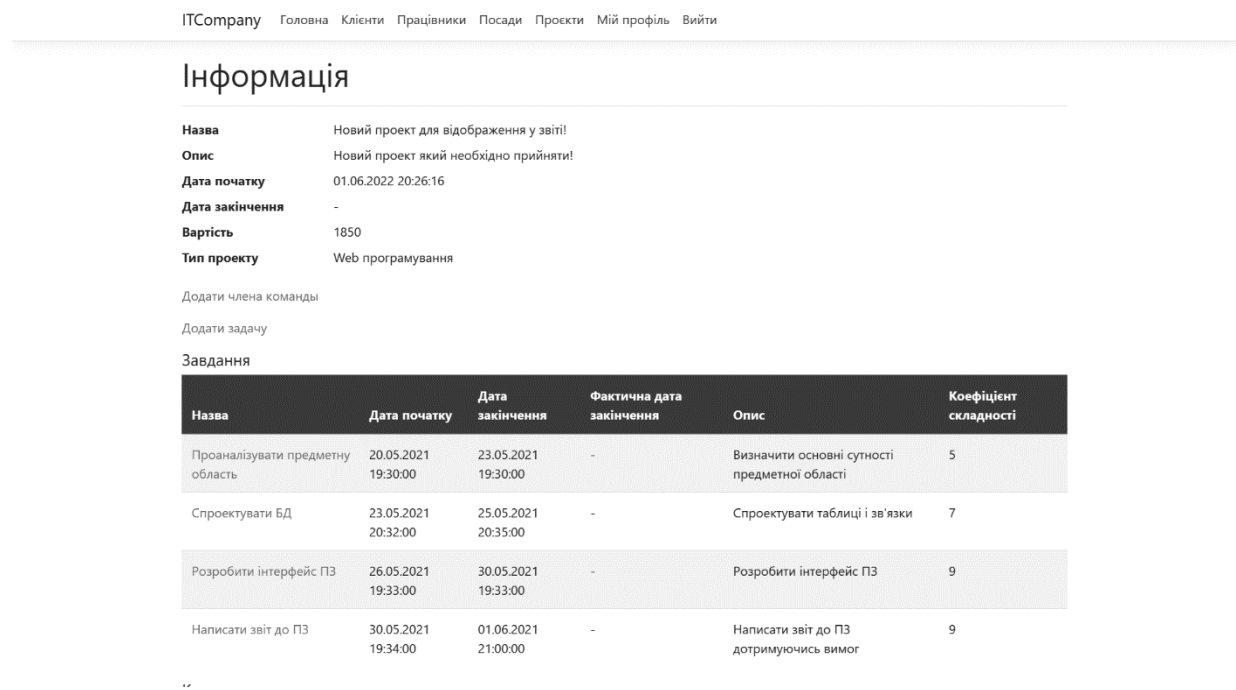


Рис. 3.17 Сторінка інформації про проект.

Для того щоб закрити проект, необхідно виконати усі завдання. Для цього переходимо на сторінку інформації про завдання (рис. 3.18) та натискаємо кнопку «Завершити завдання»

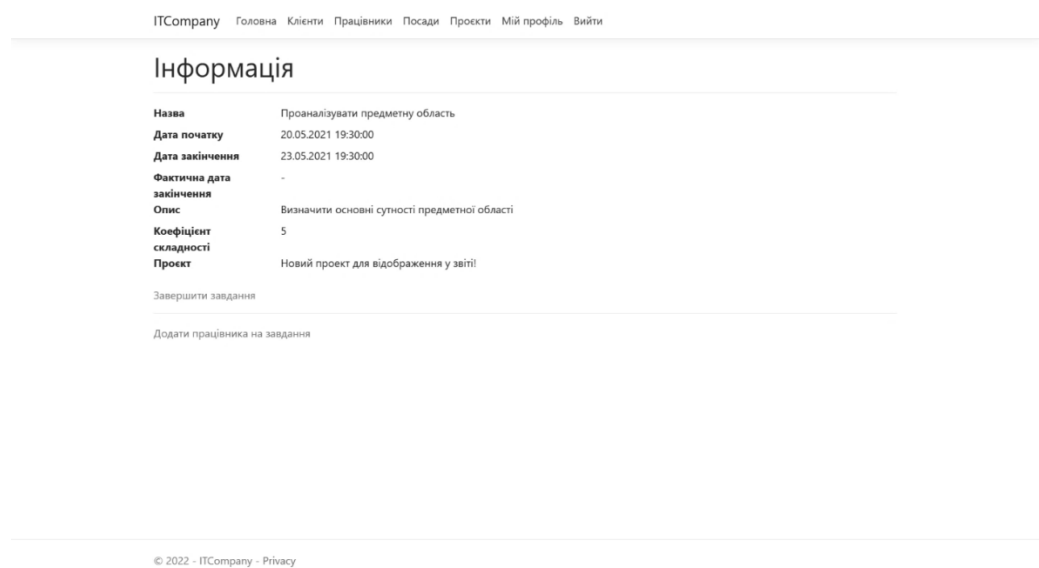


Рис. 3.17 Сторінка інформації про завдання.

Після чого відсоток виконання проекту буде змінено в залежності від кількості виконаних завдань (рис. 3.18)

Назва	Опис	Дата початку	Дата закінчення	Вартість	Тип проекту	Відсоток виконання	Кількість програмістів	Кількість завдань	Інформація
Test Project	Test Project Description	01.06.2022 15:42:18	01.06.2022 17:24:56	10000	Мобільні додатки	100	4	2	Інформація
Проект веб сайту	Потрібен веб сайт компанії	01.06.2022 16:18:04	01.06.2022 17:28:04	1000	Розробка веб сайтів	0	1	3	Інформація
3	3	01.06.2022 17:31:15	01.06.2022 17:40:36	3	Сторення візиток та плакатів	100	2	6	Інформація
1	1	01.06.2022 17:44:08	01.06.2022 17:47:35	1	Сторення візиток та плакатів	100	2	5	Інформація
Mixzo PROJECT 2M	Test Project Description	01.06.2022 17:54:19	-	10000	Сторення візиток та плакатів	28	0	7	Інформація
Test Project For Employee	Test Project For Employee	01.06.2022 18:08:38	-	100	Мобільні додатки	66	2	3	Інформація
Новий проект для відображення у звіті!	Новий проект який необхідно прийняти!	01.06.2022 20:26:16	-	1850	Web програмування	25	4	4	Інформація

© 2022 - ITCompany - Privacy

Рис. 3.18 Сторінка інформації про відсоток виконання проекту.

Кожен користувач може переглянути та відредагувати власну інформація в особистому кабінеті (рисунок 3.19). Для цього необхідно натиснути кнопку «Мій профіль»

ITCompany Головна Мої проекти Мій профіль Вийти

Інформація

Аватар



Логін

worker12

Пароль

worker12

Ім'я

Андрій

Прізвище

Дімовчі

Роль

DevOps Engineer

Редагувати

Проекти

Назва	Опис	Дата початку	Дата закінчення	Вартість	Тип проекту	Відсоток виконання	Кількість програмістів	Кількість завдань
Test Project	Test Project Description	01.06.2022 15:42:18	01.06.2022 17:24:56	10000	Мобільні додатки	100	4	2
Проект веб сайту	Потрібен веб сайт компанії	01.06.2022 16:18:04	01.06.2022 17:28:04	1000	Розробка веб сайтів	0	1	3
3	3	01.06.2022	01.06.2022	3	Сторення	100	2	6

Рис. 3.18 Особистий профіль користувача

На даній сторінці також відображаються списки усіх проектів та завдань у яких приймав участь даний користувач.

ВИСНОВКИ

У результаті виконання магістерської роботи було розроблено комплексну систему інтерактивного тайм-менеджменту, орієнтовану на використання в межах ІТ компанії. Основною метою роботи було створення ефективного інструменту для управління часом, завданнями та проектами, що дозволяє підвищити продуктивність роботи як окремих працівників, так і команд у динамічному та швидко змінному середовищі, характерному для ІТ-сектору.

Під час роботи над проектом було здійснено детальний аналіз вимог та проблем, з якими стикаються фахівці ІТ-компаній під час планування та виконання завдань. Особлива увага була приділена специфіці управління часом у командній роботі, де важливу роль відіграє синхронізація дій різних учасників проєктів, швидка адаптація до змін, ефективна комунікація та контроль над виконанням завдань у межах жорстких дедлайнів.

Розроблена система тайм-менеджменту забезпечує можливість інтеграції з іншими інструментами, які вже активно використовуються в ІТ-компаніях, такими як системи управління проектами, комунікаційні платформи та календарі. Це дозволяє користувачам не тільки планувати й організовувати завдання, але й ефективно взаємодіяти з іншими членами команди, що є важливим аспектом для забезпечення продуктивності та виконання завдань у командному середовищі.

Одним із важливих аспектів є інтерактивність та гнучкість системи, що дозволяє налаштовувати інтерфейс відповідно до потреб користувачів. Врахування потреб різних ролей в команді, таких як розробники, тестувальники, менеджери проєктів та інші, забезпечує індивідуальний підхід до організації робочого процесу, дозволяючи кожному учаснику зосередитися на своїх завданнях, не перевантажуючи систему надмірною інформацією.

Завдяки розробленій базі даних та програмній частині, система здатна швидко обробляти великі обсяги інформації та надавати точні звіти про виконання завдань, що важливо для оцінки ефективності роботи команди. Оскільки ІТ-компанії часто працюють з кількома проектами одночасно, сис-

тема дозволяє легко відстежувати прогрес по кожному з них, управляти пріоритетами завдань і коригувати розподіл ресурсів в режимі реального часу.

Розробка інтерфейсу була орієнтована на зручність і доступність, оскільки для ІТ-команд важливо мати можливість швидко орієнтуватися в системі, щоб не витрачати час на навчання. Інтерфейс має інтуїтивно зрозумілу навігацію та підтримує адаптивний дизайн, що дозволяє зручно працювати як на десктопних комп'ютерах, так і на мобільних пристроях. Ці фактори є важливими, оскільки в ІТ-сфері часто використовується мобільний доступ до робочих інструментів.

Система також включає механізми безпеки, зокрема захист персональних даних користувачів, аутентифікацію та авторизацію, що є важливими для ІТ-компаній, де конфіденційність і безпека даних мають високий пріоритет. Крім того, можливість інтеграції з іншими платформами для управління проектами та комунікації дозволяє забезпечити безперебійну роботу між різними інструментами, що використовуються в компанії.

Завдяки усім цим аспектам, система має значний практичний потенціал для використання в ІТ-компаніях, дозволяючи не тільки організовувати роботу команди, а й підвищувати ефективність використання часу, знижувати рівень стресу серед співробітників та забезпечувати досягнення проектних цілей у визначені терміни. Система інтерактивного тайм-менеджменту, що була розроблена в межах цієї роботи, є потужним інструментом для підвищення продуктивності та ефективності управління часом у сучасному ІТ-середовищі, що допомагає досягти кращих результатів при мінімальних витратах часу і ресурсів.

Отже, виконана робота дозволяє зробити висновок, що розроблена система є ефективним і перспективним інструментом для управління часом у рамках ІТ-компанії, що сприяє підвищенню продуктивності, поліпшенню комунікації в команді та досягненню цілей проектів у визначені терміни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Куліш В.І. Організація часу як складова ефективного менеджменту / В.І. Куліш. – К.: Видавництво "Наукова думка", 2018. – 256 с.
2. Васильєв В.Ю. Тайм-менеджмент для керівників: теорія та практика / В.Ю. Васильєв. – Харків: ХІМ, 2020. – 312 с.
3. Жолудь О.Л. Інформаційні системи для управління проектами / О.Л. Жолудь. – Одеса: ОНУ, 2019. – 340 с.
4. Лях В.М. Інтерактивні технології в освіті: тайм-менеджмент як засіб підвищення ефективності навчання / В.М. Лях. – Львів: ЛДУ, 2021. – 210 с.
5. Савченко Т.В. Основи менеджменту: стратегічне планування та управління часом / Т.В. Савченко. – Київ: Видавничий дім "Освітянська думка", 2017. – 184 с.
6. Попович Н.П. Тайм-менеджмент у роботі сучасного менеджера / Н.П. Попович. – К.: Центр навчальної літератури, 2016. – 150 с.
7. Андрєєва М.С. Практичні аспекти управління часом у команді / М.С. Андрєєва. – Харків: Вид-во "Юрінком Інтер", 2018. – 220 с.
8. Шевченко О.І. Управління проектами та планування часу в організаціях / О.І. Шевченко. – Одеса: ОНПУ, 2020. – 270 с.
9. Романенко В.А. Тайм-менеджмент у сфері бізнесу: сучасні підходи та інструменти / В.А. Романенко. – К.: Видавництво "Наукова думка", 2021. – 310 с.
10. Ломако О.Г. Інформаційні технології для управління часом: розробка інтерактивних систем / О.Г. Ломако. – Чернівці: ЧНУ, 2022. – 185 с.
11. Гончарук І.В. Автоматизовані системи управління проектами / І.В. Гончарук. – К.: КПІ, 2019. – 198 с.
12. Мельник С.А. Ефективне планування часу у бізнесі / С.А. Мельник. – Харків: ХНЕУ, 2017. – 245 с.
13. Дроздов П.І. Тайм-менеджмент у цифровому середовищі / П.І. Дроздов. – Львів: ЛНУ, 2021. – 190 с.

14. Сидоренко О.В. Інноваційні методи управління часом в ІТ-компаніях / О.В. Сидоренко. – Одеса: ОДАХ, 2020. – 215 с.
15. Іваненко Ю.М. Управління персоналом і ефективний тайм-менеджмент / Ю.М. Іваненко. – Київ: Видавничий центр "Академія", 2018. – 275 с.
16. Козлов В.С. Планування та управління проєктами в ІТ-секторі / В.С. Козлов. – К.: НТУУ "КПІ", 2019. – 220 с.
17. Петрова Л.І. Використання цифрових технологій у тайм-менеджменті / Л.І. Петрова. – Харків: ХНУРЕ, 2020. – 180 с.
18. Корнієнко Т.М. Управління проєктами: сучасні інструменти та підходи / Т.М. Корнієнко. – К.: Видавництво "Освіта", 2022. – 310 с.
19. Бондаренко Ю.В. Методи організації часу в умовах високої динаміки роботи / Ю.В. Бондаренко. – Одеса: ОНУ, 2018. – 200 с.
20. Чорний П.Є. Тайм-менеджмент у корпоративному середовищі / П.Є. Чорний. – Львів: ЛНУ, 2021. – 230 с.
21. Морозов Д.А. Використання SCRUM і Kanban у тайм-менеджменті / Д.А. Морозов. – Київ: Видавництво "Наука", 2020. – 260 с.
22. Ковальчук Н.М. Гнучкі методи управління часом у командних проєктах / Н.М. Ковальчук. – Харків: ХНЕУ, 2019. – 210 с.
23. Рибак О.О. Управління завданнями та тайм-менеджмент у великих організаціях / О.О. Рибак. – Одеса: ОНПУ, 2021. – 275 с.
24. Мартиненко І.П. Автоматизація тайм-менеджменту в ІТ-компаніях / І.П. Мартиненко. – Львів: ЛДУ, 2022. – 185 с.
25. Ткаченко С.В. Практичні аспекти управління часом для менеджерів / С.В. Ткаченко. – К.: Видавництво "Фенікс", 2020. – 195 с.
26. Ігнатенко А.Г. Тайм-менеджмент у стартапах та молодих командах / А.Г. Ігнатенко. – Київ: ВЦ "Освіта", 2021. – 180 с.
27. Юрченко О.М. Системи управління завданнями та часом у бізнесі / О.М. Юрченко. – Харків: ХІМ, 2019. – 205 с.
28. Демченко В.Л. Тайм-менеджмент та ефективність персоналу / В.Л. Демченко. – Львів: ЛНУ, 2018. – 230 с.

- 29.Макаренко П.О. Управління робочим навантаженням в ІТ-командах / П.О. Макаренко. – Одеса: ОДАХ, 2022. – 175 с.
- 30.Степаненко Ю.І. Тайм-менеджмент у цифрових стартапах / Ю.І. Степаненко. – К.: Видавництво "Альфа", 2019. – 190 с.
- 31.Гаврилюк В.М. Ефективне управління часом у великих корпораціях / В.М. Гаврилюк. – Харків: ХНЕУ, 2021. – 280 с.
- 32.Мішин Д.В. Використання CRM-систем для управління часом / Д.В. Мішин. – Одеса: ОНУ, 2020. – 210 с.
- 33.Білик О.П. Моделювання часу в інформаційних системах / О.П. Білик. – Львів: ЛНУ, 2022. – 195 с.
- 34.Савчук О.А. Планування часу за методологією GTD / О.А. Савчук. – Київ: ВЦ "Освіта", 2019. – 205 с.
- 35.Фролов І.Ю. Управління змінами та тайм-менеджмент / І.Ю. Фролов. – Харків: ХНУРЕ, 2021. – 215 с.
- 36.Пархоменко Л.М. Роль штучного інтелекту в управлінні часом / Л.М. Пархоменко. – К.: Видавництво "Техніка", 2020. – 230 с.
- 37.Коваленко О.Є. Інтерактивні системи планування часу / О.Є. Коваленко. – Одеса: ОНПУ, 2019. – 220 с.
- 38.Іванов В.А. Сучасні підходи до управління проєктами / В.А. Іванов. – Київ: ВЦ "Наука", 2021. – 250 с.
- 39.Гречко Ю.М. Мобільні додатки для тайм-менеджменту / Ю.М. Гречко. – Харків: ХІМ, 2020. – 190 с.
- 40.Карпенко П.В. Управління часом у віддалених командах / П.В. Карпенко. – Львів: ЛДУ, 2022. – 185 с.

ДОДАТОК КОД РОЗРОБЛЕНОГО ДОДАТКУ

Приклад контролеру

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ITCompany.Data;
using ITCompany.Models;
namespace ITCompany.Controllers
{
    public class ProjectsController : Controller
    {
        private readonly ITCompanyContext _context;

        public ProjectsController(ITCompanyContext context)
        {
            _context = context;
        }

        // GET: Projects
        public async Task<ActionResult> Index()
        {
            var iTCompanyContext = _context.Projects.Include(p =>
p.ProjectType).Include(x=>x.ProjectCommands).Include(x => x.Tasks);
            return View(await iTCompanyContext.ToListAsync());
        }

        public async Task<ActionResult> MyProjects()
        {
            var currentUser = _context.Users.Include(x=>x.Role).First(x =>
x.Login.Equals(HttpContext.User.Identity.Name));
            if (currentUser.Role.Name.Equals("Клієнт"))
            {
                var iTCompanyContext = _context.Projects.Include(p => p.ProjectType).Include(x =>
x.ProjectCommands).Include(x => x.Tasks).Where(x => x.ClientID == currentUser.ID);
                return View("Index", await iTCompanyContext.ToListAsync());
            }
            else
            {
                var iTCompanyContext = _context.Projects.Include(x=>x.ProjectCommands).Include(p
=> p.ProjectType).Include(x => x.ProjectCommands).Include(x => x.Tasks).Where(x =>
x.ProjectCommands.Select(x=>x.EmployeeID).Contains(currentUser.ID));
                return View("Index", await iTCompanyContext.ToListAsync());
            }
        }
    }
}

```

```

    }

    // GET: Projects/Details/5
    public async Task<ActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }
        var project = await _context.Projects
            .Include(p => p.ProjectType)
            .Include(p => p.ProjectCommands)
            .Include(p => p.Tasks)
            .FirstOrDefaultAsync(m => m.ID == id);

        var currentUser = _context.Users.First(x =>
            x.Login.Equals(HttpContext.User.Identity.Name));

        ViewBag.isPrime = false;

        foreach (ProjectCommand pc in project.ProjectCommands)
        {
            pc.Employee = _context.Users.Include(x=>x.Role).First(x=>x.ID == pc.EmployeeID);
            if (pc.EmployeeID == currentUser.ID && pc.isPrime)
                ViewBag.isPrime = true ;
        }

        if (project == null)
        {
            return NotFound();
        }

        return View(project);
    }

    public async Task<ActionResult> Accept(int? id)
    {
        var project = await _context.Projects
            .Include(p => p.ProjectType)
            .FirstOrDefaultAsync(m => m.ID == id);

        project.StartDate = DateTime.Now;

        _context.Projects.Update(project);
        _context.SaveChanges();
        return Redirect("/Projects/Details/"+project.ID);
    }

```

```

// GET: Projects/Create
public IActionResult Create()
{
    ViewData["ProjectTypeID"] = new SelectList(_context.ProjectTypes, "ID", "Name");
    return View();
}
// POST: Projects/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("ID,Name,Description,Price,ProjectTypeID")]
Project project)
{
    if (ModelState.IsValid)
    {
        var currentUser = _context.Users.First(x =>
x.Login.Equals(HttpContext.User.Identity.Name));
        project.ClientID = currentUser.ID;
        _context.Add(project);
        await _context.SaveChangesAsync();
        return Redirect("/Projects/MyProjects");
    }
    ViewData["ProjectTypeID"] = new SelectList(_context.ProjectTypes, "ID", "ID",
project.ProjectTypeID);
    return View(project);
}

// GET: Projects/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var project = await _context.Projects.FindAsync(id);
    if (project == null)
    {
        return NotFound();
    }
    ViewData["ProjectTypeID"] = new SelectList(_context.ProjectTypes, "ID", "ID",
project.ProjectTypeID);
    return View(project);
}

// POST: Projects/Edit/5

```

```

// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Edit(int id,
[Bind("ID,Name,Description,StartDate,EndDate,Price,ProjectTypeID")] Project project)
{
    if (id != project.ID)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(project);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ProjectExists(project.ID))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["ProjectTypeID"] = new SelectList(_context.ProjectTypes, "ID", "ID",
project.ProjectTypeID);
    return View(project);
}

// GET: Projects/Delete/5
public async Task<ActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var project = await _context.Projects
        .Include(p => p.ProjectType)
        .FirstOrDefaultAsync(m => m.ID == id);

```

```

        if (project == null)
        {
            return NotFound();
        }

        return View(project);
    }

    // POST: Projects/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task DeleteConfirmed(int id)
    {
        var project = await _context.Projects.FindAsync(id);
        _context.Projects.Remove(project);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool ProjectExists(int id)
    {
        return _context.Projects.Any(e => e.ID == id);
    }
}

```

Моделі

```

using System;
using System.Collections.Generic;

namespace ITCompany.Models
{
    public class Project
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime StartDate { get; set; }
        public DateTime EndDate { get; set; }
        public int Price { get; set; }
        public int ProjectTypeID { get; set; }
        public int Percent { get; set; }
        public int ClientID { get; set; }
        public User Client { get; set; }
        public ProjectType ProjectType { get; set; }
        public ICollection<Task> Tasks { get; set; }
        public ICollection<ProjectCommand> ProjectCommands { get; set; }
    }
}

```

```

    }
}
Інтерфейсу
@model ITCompany.Models.Project
@{
    ViewData["Title"] = "Details";
}

<h1>Інформація</h1>
<div>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            Назва
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Name)
        </dd>
        <dt class="col-sm-2">
            Опис
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt class="col-sm-2">
            Дата початку
        </dt>
        @if(Model.StartDate.ToString() != "01.01.0001 0:00:00"){
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.StartDate)
            </dd>
        }
        else{
            <dd class="col-sm-10">
                @Html.Raw("-")
            </dd>
        }
        <dt class="col-sm-2">
            Дата закінчення
        </dt>
        @if(Model.EndDate.ToString() != "01.01.0001 0:00:00"){
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.EndDate)
            </dd>
        }
        else{
            <dd class="col-sm-10">
                @Html.Raw("-")
            </dd>
        }
    </dl>
</div>

```

```

        </dd>
    }
    <dt class = "col-sm-2">
        Вартість
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Price)
    </dd>
    <dt class = "col-sm-2">
        Тип проекту
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.ProjectType.Name)
    </dd>
</dl>
</div>
<div>
    @if(Model.StartDate.ToString() == "01.01.0001 0:00:00") {
        <p>
            <a asp-controller="Projects" asp-action="Accept" asp-route-
id="@Model.ID">Прийняти</a>
        </p>
    }
    @*todo add functional for isPrimeUser*@
    else if(!User.IsInRole("Клієнт") && (Model.EndDate.ToString() == "01.01.0001 0:00:00") ||
ViewBag.isPrime){
        <p>
            <a asp-controller="ProjectCommands" asp-action="Create" asp-route-
id="@Model.ID">Додати члена команди</a>
        </p>
        <p>
            <a asp-controller="Tasks" asp-action="Create" asp-route-id="@Model.ID">Додати зада-
чу</a>
        </p>
    }
</div>

<div>
    @if(Model.Tasks.Count() > 0)
    {
        <h5>Завдання</h5>
        <table class="table table-striped">
            <thead class="thead-dark">
                <tr>
                    <th>
                        Назва
                    </th>
                    <th>

```

```

        Дата початку
    </th>
    <th>
        Дата закінчення
    </th>
    <th>
        Фактична дата закінчення
    </th>
    <th>
        Опис
    </th>
    <th>
        Коефіцієнт складності
    </th>
</tr>
</thead>
<tbody>
@foreach (var item in Model.Tasks) {
    <tr>
        <td>
            <a asp-controller="Tasks" asp-action="Details" asp-route-
id="@item.ID">@Html.DisplayFor(modelItem => item.Name)</a>
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.StartDate)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EndDate)
        </td>
        <td>
            @if(item.FactDate.ToString() != "01.01.0001 0:00:00")
            {
                <td>
                    @Html.DisplayFor(modelItem => item.FactDate)
                </td>
            }
            else
            {
                <td>
                    @Html.Raw("-")
                </td>
            }
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Description)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Coeffecient)
        </td>
    </tr>
}

```

```

    }
    </tbody>
  </table>
}
</div>
<div>
  @if(Model.ProjectCommands.Count() > 0)
  {
    <h5>Команда</h5>
    <table class="table table-striped">
    <thead class="thead-dark">
      <tr>
        <th>
        </th>
        <th>
          Працівник
        </th>
        <th>
        </th>
        <th>
          Головний?
        </th>
      </tr>
    </thead>
    <tbody>
    @foreach (var item in Model.ProjectCommands) {
      <tr>
        <td>
          <a asp-controller="Users" asp-action="Details" asp-route-
id="@item.Employee.ID"></a>
        </td>
        <td>
          <a asp-controller="Users" asp-action="Details" asp-route-
id="@item.Employee.ID">@Html.DisplayFor(modelItem => item.Employee.FullName)</a>
        </td>
        <td>
          @Html.DisplayFor(modelItem => item.Employee.Role.Name)
        </td>
        <td>
          @Html.DisplayFor(modelItem => item.isPrime)
        </td>
      </tr>
    }
    </tbody>
  </table>
  }
</div>

```