

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ УПРАВЛІННЯ**
ЗАДАЧАМИ ТА ПРОЄКТАМИ
(назва теми)

Виконала: студентка 4 курсу, групи КНТз-510
спеціальності

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Комп'ютерна інженерія»

ПІВЕНЬ А. С.

(ПРИЗВИЩЕ та ініціали)

Керівник КАСЬЯН К. М.

(ПРИЗВИЩЕ та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПІВЕНЬ Альони Сергіївни
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи управління задачами та проєктами
керівник проєкту (роботи) к. т. н., доцент, КАСЬЯН Костянтин Миколайович
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105
2. Строк подання студентом проєкту (роботи) 26 травня 2024 року
3. Вихідні дані до проєкту (роботи) наявні системи управління задачами та проєктами, організація та відстеження завдань та проєктів у команді, з можливістю призначення завдань та моніторингу їх виконання, враховуючи об'єктно-орієнтований підхід, мова програмування Python.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз технічного завдання, дослідження наявних методів управління проєктами та задачами;
2) Розробка структури та алгоритму роботи системи;
3) Реалізація та дослідження системи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Структурна схема реалізації системи;
Алгоритм управління задачами у системі;
Реляційна база даних.
Екранна форма системи
6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КАСЬЯН К. М., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 20 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз існуючих систем управління проєктами та обґрунтування необхідності розробки нової системи	20.03.2024 р.	
	проведення порівняльного аналізу систем, управління задачами та проєктами		
2	Розробка структури системи	03.04.2024 р.	
3	Розробка алгоритму роботи системи	15.04.2024 р.	
4	Реалізація системи управління задачами та проєктами	24.04.2024 р.	
5	Візуалізація розробленої системи	30.04.2023 р.	
6	Оформлення отриманих результатів у ПЗ	01.05.2023 р.	
7	Оформлення графічного матеріалу	01.05.2023 р.	
8	Оформлення допоміжного матеріалу	02.05.2023 р.	

Студент (ка)


(підпис)

Альона ШВЕНЬ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Костянтин КАСЬЯН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 72 с., 23 рис., 3 табл., 24 джерела.

КОМП'ЮТЕРНА СИСТЕМА, УПРАВЛІННЯ, ЗАВДАННЯ, ПРОЄКТИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ХАРАКТЕРИСТИКИ, ІНТЕРФЕЙС, ТЕХНОЛОГІЇ.

Об'єкт розробки – система управління задачами та проєктами.

Метою роботи є розробка системи управління задачами та проєктами для оптимізації робочого процесу, проведення аналізу методів управління проєктами.

Проєкт складається з чотирьох розділів.

Перший розділ присвячено аналізу систем управління задачами та проєктами. Проводиться порівняльний аналіз існуючих систем.

У другому розділі представлено обґрунтування потреби в новій системі.

У третьому розділі описано структуру системи, перелік використаних технологій та інструментів. А також наведено огляд коду та основних функціональних можливостей системи.

Четвертий розділ присвячено огляду структури бази даних, що використовується у системі. Також у розділі представлено моделювання системи, а також розглянуто елементи візуального дизайну. В кінці розділу надано висновки щодо візуалізації та рекомендації з покращення та модернізації системи на основі отриманих результатів.

В результаті виконання дипломної роботи проведено аналіз, моделювання та дослідження розроблюваної системи управління та розроблені рекомендації щодо її модернізації.

ЗМІСТ

Скорочення та умовні позначки	8
Вступ.....	9
1 Аналіз існуючих систем управління задачами та проєктами	11
1.1 Система управління задачами Jira від Atlassian.....	11
1.2 Система керування проєктами Microsoft Project.....	14
1.3 Онлайн-система керування проєктами Worksection.....	17
2 Обґрунтування потреби розроблення нової системи	19
3 Розробка системи управління задачами та проєктами	23
3.1 Огляд структури розроблюваної системи.....	23
3.2 Програмне забезпечення.....	36
3.3 Код програми та опис основних функціональних можливостей системи	40
4 Структура бази даних та інтерфейс системи.....	53
4.1 Структура бази даних	53
4.2 Розробка інтерфейсу системи	56
4.3 Тестування розробленої системи.....	61
Висновки	69
Перелік джерел посилання	70

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ППЗ	—	прикладне програмне забезпечення
РБД	—	реляційна база даних
СПЗ	—	системне програмне забезпечення
MIME	—	інтернет-стандарт, який розширює формат електронної пошти, Multipurpose Internet Mail Extensions
SMTP	—	комунікаційний протокол для пересилання електронної пошти, Simple Mail Transfer Protocol

ВСТУП

У сучасному світі щодня перед командами розробників і бізнесом постає велика кількість різних завдань та задач. Дуже часто це призводить до того, що деякі задачі так і залишаються не виконаними. Все це відбувається через відсутність чіткої пріоритезації, що в свою чергу веде до незапланованих затримок у виконанні проєктів. З огляду на цю проблему, управління проєктом є важливим аспектом успішного виконання завдання. Існує велика кількість різних систем з управління проєктами. І провідна роль цих систем полягає у групуванні задач та їхньому ефективному розподілі, аби досягати запланованих результатів. Розробка комп'ютерної системи управління задачами та проєктами сприяє підвищенню ефективності та якості роботи, що є ключовим для команд та організацій, особливо зараз у час швидких змін у ринковому середовищі.

Комп'ютерна система вирішує цілий ряд проблем, таких як недостатня організація та структурування задач, втрата даних під час виконання проєкту та інформаційний хаос, неспроможність відстежувати прогрес та виконання завдань. І не менш важливою перевагою використання системи є пошук неефективних підходів та процесів у проєктах команд. Бо саме дані, які збираються системою управління, дозволяють ідентифікувати недіючі підходи та процеси, і саме їх можна вдосконалювати та змінювати.

Підставами щодо необхідності розробки нової системи управління проєктами можна врахувати статистичні дані. Згідно з дослідженнями, лише 23 % організацій використовують програмне забезпечення для управління проєктами.

А сама недооцінка управління проєктами призводить до 50% провалів тих самих проєктів.

Саме тому розробка комп'ютерної системи управління задачами та проєктами є актуальним завданням, яке відповідає вимогам сучасності та розв'язує численні проблеми, які виникають під час роботи команд та організацій.

Сутність цієї системи полягає в створенні програмного забезпечення, яке дасть змогу ефективно керувати задачами та проєктами, оминаючи значних витрат ресурсів (часу, грошей тощо). Згідно з даними, компанії, які інвестують у перевірені практики управління проєктами, витрачають у 28 разів менше грошей.

Зважаючи на постійне зростання обсягів роботи та вимог щодо точності та швидкості виконання завдань, впровадження нової системи є важливим і необхідним.

Адже, недоліки традиційних методів управління стають більш очевидними, і необхідність вдосконалення системи управління більш актуальною.

Для розробки комп'ютерної системи управління проєктами необхідні вихідні дані, які містять в собі інформацію про поточні процеси управління, аналіз зібраних вимог потенційних користувачів системи, визначення технічних характеристик системи, таких як мова програмування, бази даних, архітектурні вимоги тощо.

Відтак впровадження комп'ютерної системи управління задачами та проєктами дасть нагоду ефективно планувати, координувати та контролювати виконання завдань і проєктів, знижуючи ризики та підвищуючи якість результатів. Такий крок є провідним для забезпечення конкурентоспроможності та успішного функціонування команд та організацій в існуючих умовах, де швидкість та якість є головними факторами успіху.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ УПРАВЛІННЯ ЗАДАЧАМИ ТА ПРОЄКТАМИ НА РИНКУ

1.1 Система управління задачами Jira від Atlassian

Для успішності проєкту необхідно обрати найбільш відповідний сервіс управління проєктами. Аби обрати найкращий варіант варто враховувати розмір команди, тривалість самого проєкту, складність завдань і тип методології. Адже все це надважливі складові успішних проєктів. Таким чином, можна знайти інструмент, який якнайкраще підійде для всього проєкту і буде задовольняти користувача. На сьогодні у світі існує велика кількість систем, які допомагають керувати проєктами та вирішувати задачі, які постають в процесі роботи над проєктами.

Згідно з дослідженням, які надає у відкритому доступі дослідницька та маркетингова компанія Capterra, на даний момент на ринку програмного забезпечення для управління проєктами лідирують системи, як-от Jira (37%), Microsoft Project (20%), Kanban (5,4%) і Smartsheet (5%). Вони займають перші місця серед користувачів.

Нині найпопулярнішою серед існуючих систем управління задачами та проєктами є система Atlassian JIRA (рис. 1.1). Ця система використовується командами розробників, великими організаціями та стартапами.

Jira — це система відстеження помилок, яка рекомендована для організації спілкування з користувачами та управління проєктами. Система була розроблена австралійською компанією з розробки програмного забезпечення для ІТ індустрії Atlassian ще у 2002 році.

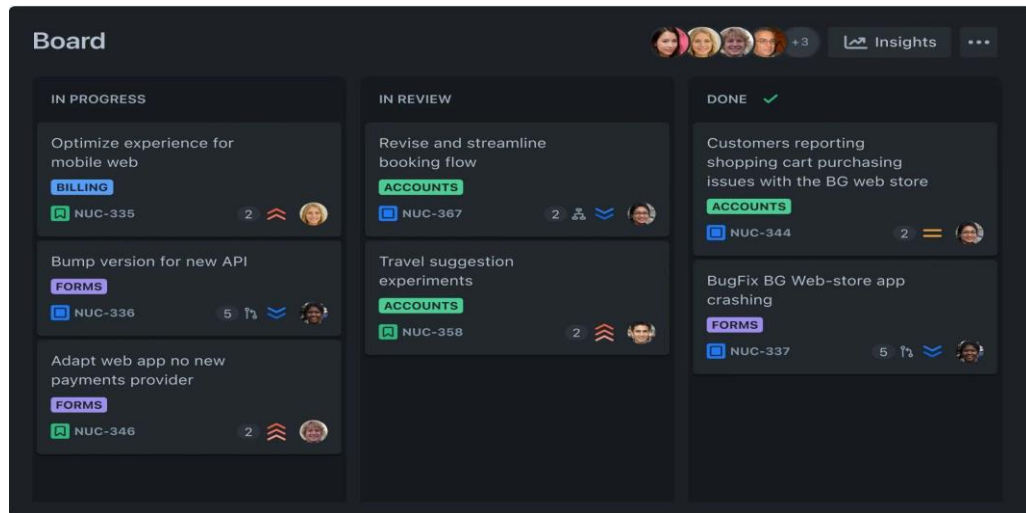


Рисунок 1.1 — Інтерфейс системи JIRA

Цей хмарний сервіс складається з трьох основних розділів: проекти (завдання, баги та запити), проблеми (списки багів) та робочий процес (послідовність кроків).

Основні функції цієї системи:

- планування проекту;
- встановлення етапів;
- організація робочих процесів;
- робота з Канбан-дошками;
- управління доступом та правами для користувачів;
- аналіз проекту та звітність.

Ще однією особливістю цієї платформи є розбиття складних задач та проектів на практичні етапи (рис. 1.2). Користувачі можуть забезпечити успішне втілення кожного етапу кампанії, забезпечуючи ефективний контроль і планування. При цьому система спрощує процес збору запитів та узгодження зусиль з різними функціональними командами, що дозволяє швидко реагувати на зміни та забезпечувати спільну спрямованість у реалізації маркетингових стратегій.

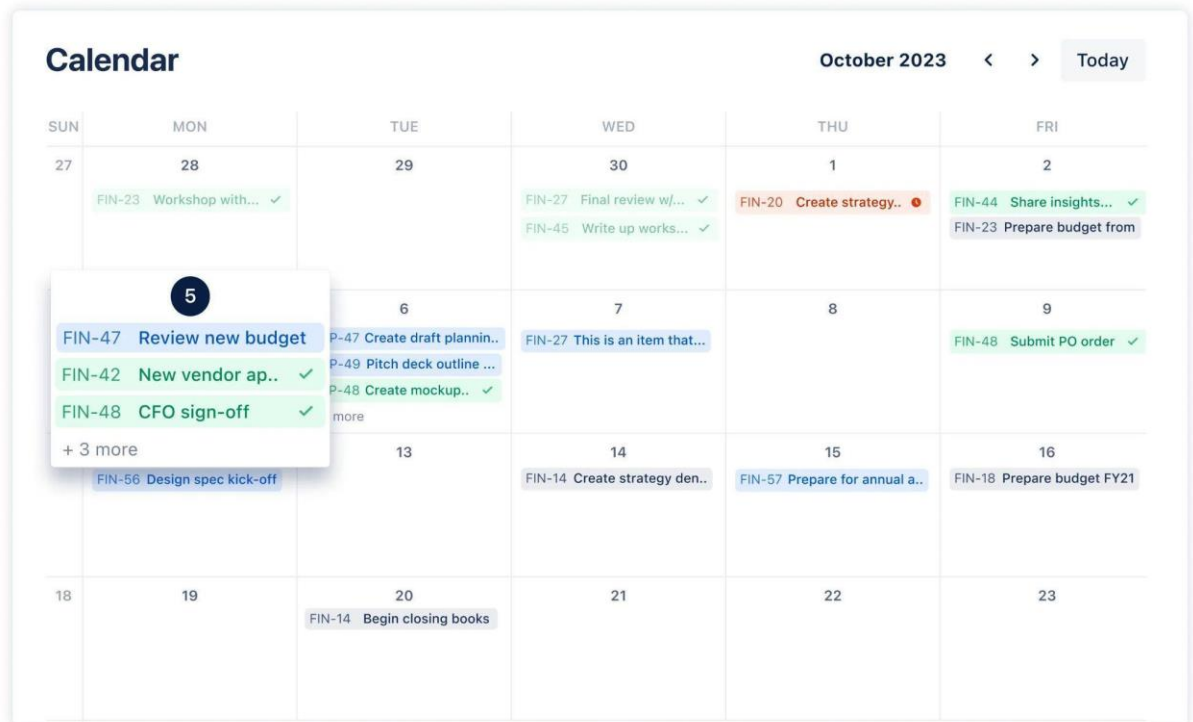


Рисунок 1.2 — Розбиття задач на компоненти

Всі ці функції роблять цей інструмент потужною системою управління проектами, яку обирають користувачі в усьому світі. А втім ця система має і свої недоліки, які зменшують можливості цього сервера. Інтерфейс Jira містить в собі безліч інструментів, які іноді буває важко знайти користувачам і це погіршує правильне використання окремих інструментів. Навіть ці інструменти можуть взагалі не використовуватися у роботі. Система управління Jira має також і мобільну версію, яка, на жаль, поступається браузерній версії. У мобільній версії користувачам складніше орієнтуватися в управлінні та виконувати завдання. Цей недолік зменшує кількість користувачів цього програмного забезпечення. Варто відзначити, що Jira також не надає змогу користувачам завантажувати зображення та графіки високої якості, що обмежує користувачів використовувати систему повноцінно в управлінні проектами різного напрямку.

1.2 Система керування проєктами Microsoft Project

Наступною системою за кількістю використання користувачами є Microsoft Project (рис. 1.3). Ця система управління задачами була розроблена корпорацією Microsoft. Сама система функціонує як частина пакету Microsoft Office.

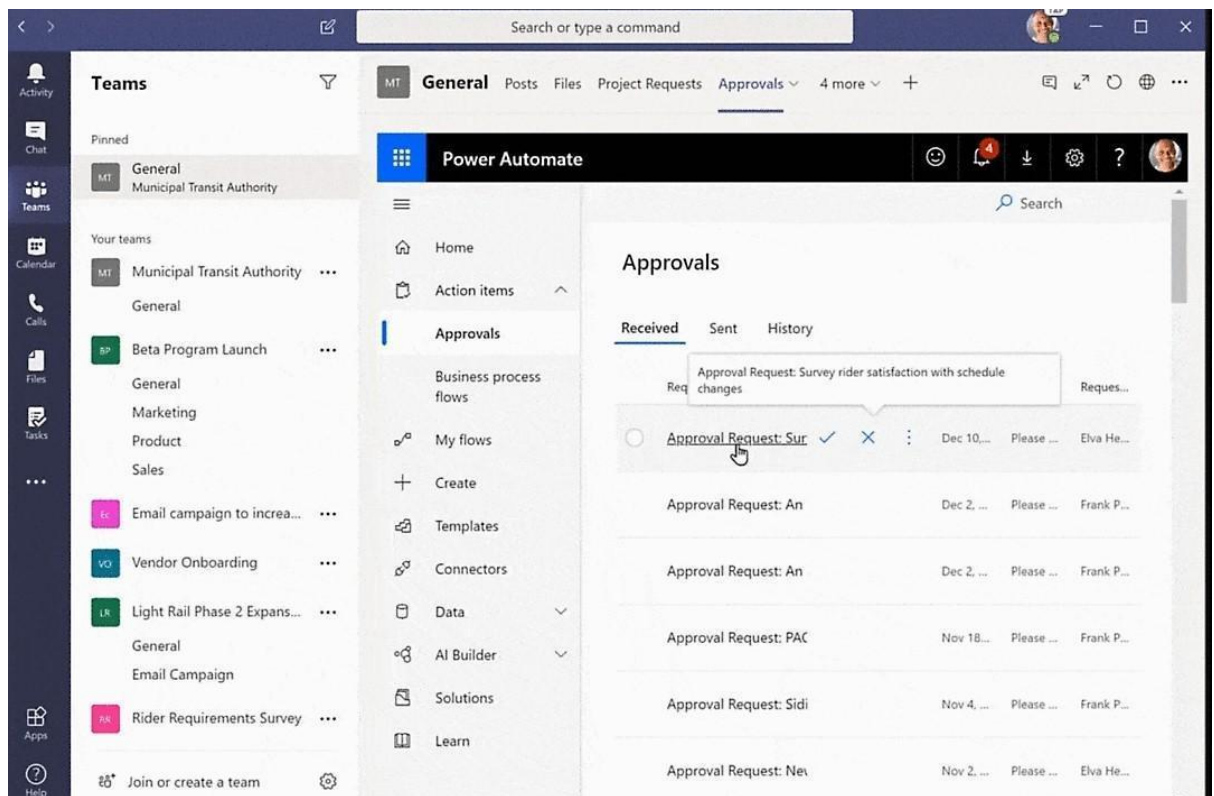


Рисунок 1.3 — Вікно вкладки «Approvals»

Microsoft Project використовується для планування, відстеження прогресу, а також для керування різними проєктами, незалежно від їхнього розміру чи складності. Ця система забезпечує розробку планів, розподіл ресурсів за задачами, відстеження прогресу й аналіз обсягів робіт, дає змогу генерувати розклади критичного шляху.

Метод критичного шляху (CPM) — це метод моделювання проєктів. Для того, щоб ефективно використовувати CPM, важливо побудувати модель проєкту,

яка включатиме конкретні елементи. Насамперед, це вичерпний перелік усіх робіт, необхідних для завершення проєкту, які, як правило, організовані в рамках структури розбиття робіт. Наступним важливим елементом має бути очікувана тривалість кожної роботи, а також час, необхідний для взаємозв'язків між роботами. Також необхідні логічні кінцеві точки, такі як етапи або готові продукти.

Розклади можуть бути складені з урахуванням використовуваних ресурсів. Ланцюжок візуалізується в діаграмі Ганта. Використання діаграми Ганта може бути ефективно поєднане зі створенням презентацій у Microsoft PowerPoint (див. рис. 1.4). Це дозволяє структурувати та візуалізувати розклад проєкту у форматі, зрозумілому для всіх учасників, а також представити його в зручній формі для презентацій та обговорень. Такий підхід сприяє зрозумілості та доступності інформації про проєкт для всієї команди, а також допомагає ефективно комунікувати ключові моменти та важливі дедлайни.

PowerPoint Gantt chart

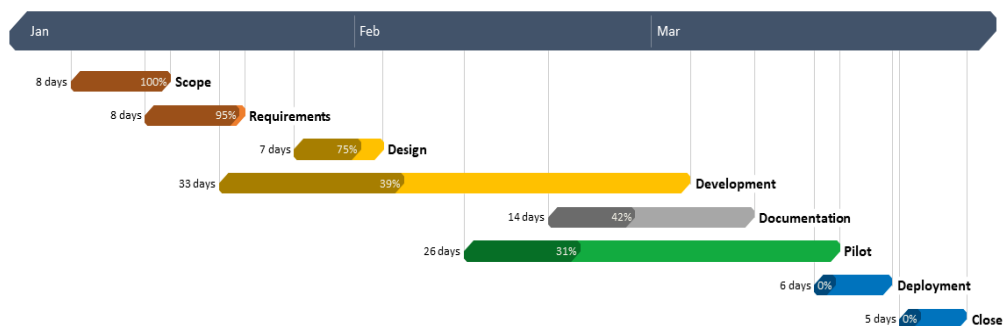


Рисунок 1.4 — Експортована діаграма Ганта у Power Point

Основними функціями системи є:

- створення детального плану проєкту, визначення таких важливих складових як, завдання, ресурси терміни, залежності;
- спільна робота учасників над проєктами;
- обмін даними та зворотнім зв'язком між усіма учасниками;
- аналізування задач та звітність;
- інтеграція з іншими програмами Microsoft.

І Microsoft Project надає підтримку користувачам на своєму власному сайті.

Утім варто ще наголосити, що це програмне забезпечення також має свої недоліки. По-перше, інтеграція з іншими програмами Microsoft, хоча і є перевагою, все-таки, може додавати труднощів команді при бажанні перейти на іншу платформу. Оскільки перехід на іншу платформу може стати складним через необхідність зміни використовуваних інструментів та переосмислення робочих процесів. Це все може призвести до перерв у роботі та зменшення продуктивності протягом періоду перепідготовки команди.

По-друге, недоліком системи є відсутність кросплатформеності. Microsoft Project призначений для операційної системи Windows, що обмежує функціонал застосунку на інших операційних системах. Варто зауважити, що Microsoft Project має версії і для macOS також.

По-третє, Microsoft Project є відносно дорогим програмним забезпеченням, особливо для невеликих компаній або некомерційних організацій. Це робить систему недоступною для користувачів, які мають обмежений бюджет для власних проєктів.

По-четверте, для недосвідчених користувачів ця система може бути складною у використанні через велику кількість функцій та можливостей. І повертаючих до третього недоліку, можна також сказати, що на навчання цією системою необхідні будуть додаткові витрати на навчання для команди.

1.3 Онлайн-система керування проєктами Worksection

На українському ринку також існують системи управління проєктами, які обираються користувачами для керування задачами. Це такі системи, як Wrike, Asana, Trello та Monday.com. Всі ці сервіси допомагають створювати проєкти та відстежувати їхнє виконання. Проте варто розглянути один із найпопулярніший і найвикористовуваніший із сервісів — це Worksection (рис. 1.5)



Рисунок 1.5 — Головне вікно системи Worksection

Цю систему можна інтегрувати із сервісами Google, будь-якою CRM-системою, Slack тощо. Worksection обирають понад 1000 компаній для керування своїми проєктами. Що й свідчить про популярність і довіру серед користувачів до програми. Worksection також можна використовувати і для різних типів бізнесу, і саме це робить систему нагальною для широкого кола користувачів.

Перевагою Worksection є вбудована діаграма Ганта, яка допомагає візуалізовувати робочі процеси та зв'язки між задачами (рис. 1.6). Діаграма Ганта надає чітке відображення графіка проєкту з урахуванням часових рамок для кожного завдання, дозволяючи команді точно розуміти, коли кожне завдання повинно бути виконане, тим самим надаючи чітке відстеження часу.

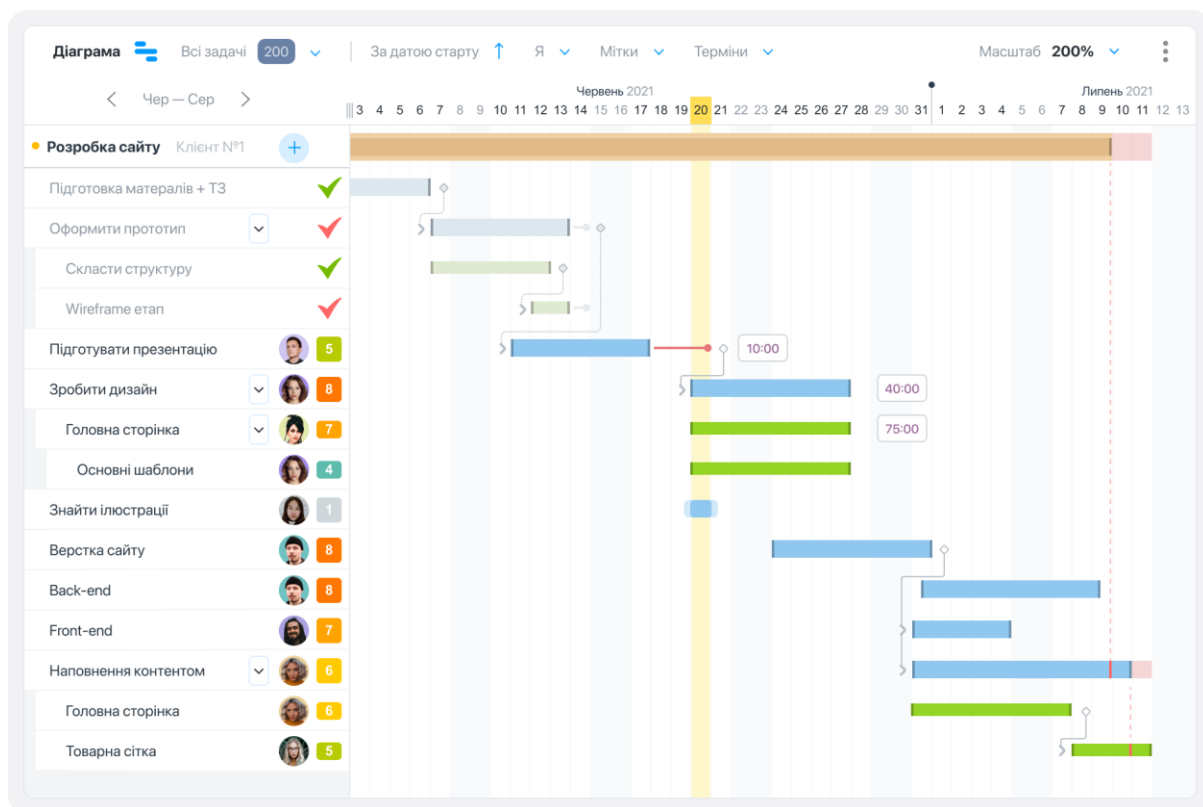


Рисунок 1.6 — Використання діаграми Ганта у системі

Ще однією перевагою Worksection є швидка та ефективна технічна підтримка, що надається користувачам сервісу. Це означає, що спеціалісти підтримки Worksection надають оперативну допомогу та консультують з будь-яких питань, пов'язаних з використанням сервісу. І технічна підтримка означає, що користувачі можуть швидко вирішувати будь-які технічні або функціональні проблеми і продовжувати працювати результативно та без перешкод. А ще система має інструмент візуального аналізу даних, який дозволяє стежити користувачам за розвитком проєкту в режимі реального часу. Також користуючись платформою

можна аналізувати залученість команди, швидко збирати статистику та контролювати бюджет проєкту. Що є великою перевагою, зберігаючи важливі ресурси, які закладаються командами та бізнесом на початкових етапах створення і реалізування всіх проєктів.

Звісно, кожна система має свої недоліки і обмеження. У Worksection — це відсутність комунікації щодо цілого проєкту між членами команди. У системі є тільки можливість спільного обговорення в межах кожної окремої задачі. Це може ускладнювати комунікацію та координацію робіт між учасниками команди, особливо на великих проєктах. Бо великий проєкт само собою містить у собі більше завдань, більшу кількість учасників та довший час виконання. Масштаб проєкту робить його складнішим для управління, координації та контролю. . Також як і у системі Jira, мобільна версія системи поступається веб-версії у плані зручності та зрозумілого інтерфейсу.

2 ОБГРУНТУВАННЯ ПОТРЕБИ РОЗРОБЛЕННЯ НОВОЇ СИСТЕМИ

Важливо відзначити, що не дивлячись на вже існуючі системи управління проєктами, ці системи мають свої недоліки, які не можуть повністю забезпечити всі необхідні запити користувачів. А найголовнішими запитами є ефективне використання ресурсів. Система управління має ефективно розподіляти ресурси, такі як людські, фінансові та матеріальні, для досягнення поставлених цілей проєкту. Тому, доречно розглядати створення нової комп'ютерної системи управління задачами та проєктами. Необхідність розроблення нової системи ґрунтується на її перевагах в порівнянні з вже існуючими сервісами з управління проєктами.

Насамперед нова система дозволяє спростити процес планування, виконання та відстеження задач. Це дозволяє команді зосереджуватися саме на досягненні поставлених цілей та виконувати всі проєкти дієво та у визначений термін.

Також перевагою є використання об'єктно-орієнтованого підходу. Саме використання цього підходу дає можливість моделювати реальні об'єкти системи з використанням класів і об'єктів. Об'єктно-орієнтований підхід порівнюється з традиційним процедурним підходом програмування. І використання саме цього підходу має свої вигоди.

Найголовнішим є те, що об'єктно-орієнтовані підходи, на відміну від традиційних методів, таких як модель водоспаду або V-подібна модель, пропонують більшу гнучкість у роботі з різними типами проєктів. Традиційні моделі часто є жорсткими і намагаються пристосовуватися до різноманітних вимог проєкту.

Також об'єктно-орієнтоване проєктування заохочує модуляризацію, тоді коли складні системи розбиваються на менші, придатні для повторного використання компоненти. Такий модульний підхід спрощує розробку, оскільки модулі можуть бути незалежно спроектовані, протестовані та підтримані. Крім того, багаторазове використання компонентів призводить до підвищення продуктивності та узгодженості між проєктами. Не можливо проігнорувати вагомість цих переваг, оскільки вони визначають успіх розробки системи управління.

У цій системі також використовується мова програмування Python, а також бази даних, все це відкриває безліч переваг, які сприяють продуктивності та забезпечують зручність використання користувачам системи управління проєктами.

Перш за все, вибір Python як основної мови програмування дозволяє забезпечити швидку розробку завдяки його простому та зрозумілому синтаксису.

Крім того, Python є однією з найпопулярніших та найбільш підтримуваних мов програмування у світі, що забезпечує наявність великої кількості ресурсів для підтримки та розвитку системи. Це означає, що з легкістю можна знайти необхідні модулі, а також користуватися великою кількістю бібліотек та фреймворків для розширення можливостей системи.

У процесі розробки системи використовується модуль Tkinter.

Застосування модулю Tkinter дозволяє будувати зручний інтерфейс системи для взаємодії з користувачами, що покращить зручність використання системи управління задачами. І це в свою чергу, забезпечує ефективну роботу з проєктами та задачами.

З використанням бази даних в системі управління проєктами можна зберігати великий обсяг даних з високою ефективністю. Це дозволяє ефективно керувати проєктами будь-якої складності, забезпечуючи доступ до потрібної інформації у будь-який час. Крім того, база даних дозволяє виконувати різноманітні аналітичні операції, генерувати звіти та статистику, що сприяє прийняттю обґрунтованих управлінських рішень.

Таким чином, використання мови програмування Python та бази даних у системі управління проєктами дозволяє забезпечити швидку розробку, дієве управління даними та розширення функціоналу, що робить цю систему потужним інструментом для успішного ведення проєктів будь-якої складності.

Не менш важливою перевагою комп'ютерної системи є її десктопна версія. Ця система призначена тільки для десктопної платформи, що є досить зручним рішенням для команди, яка працює лише на комп'ютерах і не потребує доступу до мобільних додатків. Так само перевагами додатку для роботи за стаціонарним робочим місцем є широкий функціонал системи, краще керування даними, більша продуктивність та стабільність. Адже на відміну від мобільних версій, саме комп'ютерні мають стабільніший робочий процес та більшу захищеність даних.

Ці переваги зумовлені наступними чинниками:

- тестування та валідація перед випуском, оскільки вони зазвичай мають довший цикл розробки. Це дозволяє виявляти і виправляти більше помилок та проблем перед випуском оновлень;
- присутність більшої кількості ресурсів (процесор, оперативна пам'ять, диск), ніж у мобільних пристроях. Це дозволяє десктопним програмам працювати стабільніше та дієвіше;

- на комп'ютерних операційних системах, таких як Windows або macOS, може бути більше можливостей для контролю за діями програм та обмеження їх доступу до системних ресурсів, що сприяє підвищенню безпеки;
 - можливість випускати оновлення безпеки швидше та ефективніше через більш розвинені механізми поставки оновлень на комп'ютери користувачів.
- Створення нової системи не може обмежуватися лише перевагами. Не менш важливим є визначення цілей у системі управління задачами, які потрібно досягти розробляючи комп'ютерну систему управління задачами та проєктами. Розроблювана система відповідає одній з найважливіших цілей — підвищення ефективності роботи команди. Цю ціль можна досягнути зменшивши час на планування та виконання завдань.

Наступною не менш важливою ціллю розробки нової системи управління проєктами є забезпечення її гнучкості. Система повинна бути гнучкою і легко адаптуватися до змін в потребах проєктів та команд. Це включає можливість швидко додавати нові завдання, редагувати та видаляти їх, змінювати терміни виконання та вносити інші зміни в процес управління проєктами.

І останньою, але дуже важливою ціллю створення комп'ютерної системи є підвищення користування системою та забезпечення зрозумілого інтерфейсу для її користувачів. Так як система повинна бути інтуїтивно зрозумілою і зручною. Це допоможе концентруватися на виконанні поставлених задач і не витратити час на розуміння процесу створення, планування чи зміни завдань.

Важливо також відзначити, що нова комп'ютерна система управління проєктами не лише спрощуватиме процес планування, виконання та контролю проєктів, але й сприятиме покращенню комунікації та співпраці в команді. Забезпечення доступу до актуальної інформації, можливість швидкої взаємодії та обміну даними дозволяють зменшити можливість недорозумінь та збільшують швидкість реагування на зміни та виклики проєкту. Такий інструмент стає важливою опорою для команди, допомагаючи їй досягати поставлених цілей та мати успіху в умовах постійної зміни та конкурентного середовища.

Таким чином, впровадження нової системи управління проєктами не лише підвищить продуктивність та результативність проєктної діяльності, але і буде сприяти створенню ефективної та конкурентоспроможної робочої середовища для команд, бізнесу та організацій.

Відтак, розглянуті переваги нової комп'ютерної системи управління проєктами, роблять її потужним інструментом для успішного ведення проєктів будь-якої складності. Крім того, визначені цілі доцільності розробки системи підкреслюють важливість підвищення ефективності роботи команди, забезпечення гнучкості та зрозумілого інтерфейсу для користувачів. Все це разом, підтверджує необхідність і сенс впровадження нової системи управління проєктами для підвищення результативності та успішності проєктної діяльності.

3 РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ЗАДАЧАМИ ТА ПРОЄКТАМИ

3.1 Огляд структури розроблюваної системи

Для розуміння системи управління задачами та проєктами необхідно використовувати архітектуру. Загальне визначення системної архітектури — це абстракція елементів часу виконання програмної системи під час певної фази її роботи. Система може складатися з багатьох рівнів абстракції та багатьох фаз роботи, кожна з яких має власну архітектуру програмного забезпечення.

Архітектура програмного забезпечення базується на принципі абстрагування: приховування деяких деталей системи шляхом інкапсуляції з метою кращого виявлення та збереження її властивостей. Складна система буде містити багато рівнів абстракції, кожен з яких має власну архітектуру. Архітектура — це абстракція поведінки системи на даному рівні, так що архітектурні елементи розмежовуються абстрактними інтерфейсами, які вони надають іншим елементам на цьому рівні. У середині кожного елемента можна знайти іншу архітектуру, яка визначає піделементи системи, що реалізують поведінку, представлену

абстрактним інтерфейсом батьківського елемента. Ця рекурсія архітектур триває аж до найпростіших елементів системи: тих, що не можуть бути розкладені на менш абстрактні елементи.

Розроблювальна комп'ютерна система управління задачами та проєктами має наступну архітектуру, яка складається з трьох основних компонентів та головного файлу `main.py`, який реалізує систему (рис.3.1).

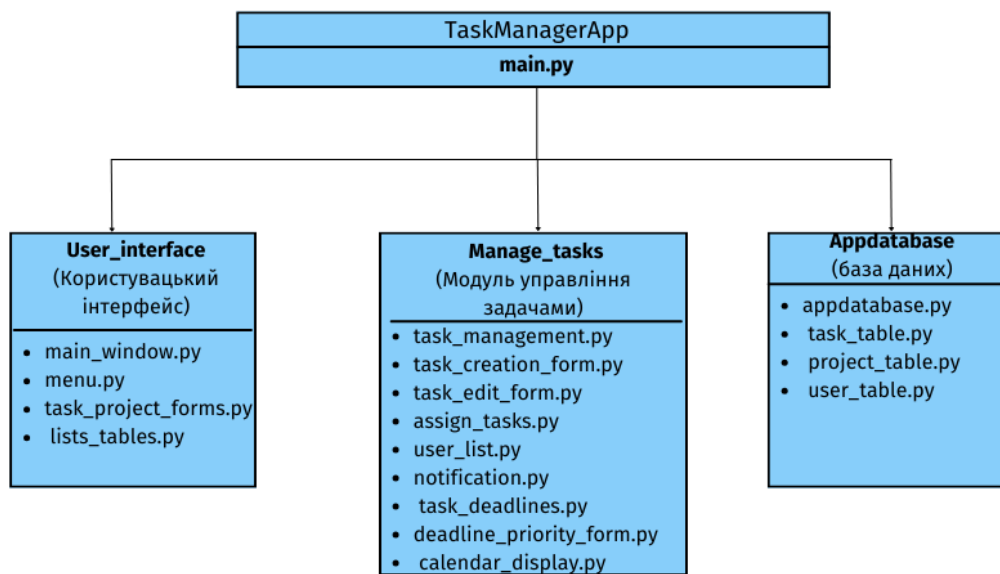


Рисунок. 3.1 — Структурна схема системи управління задачами та проєктами

Цими компонентами є Користувацький інтерфейс (User Interface), Модуль управління задачами (Module Tasks) та База даних (Appdatabase).

Користувацький інтерфейс містить у собі наступні файли:

- `main_window.py`;
- `menu.py`;
- `task_project_forms.py`;
- `lists_tables.py`.

`Menu.py` — це меню з різними функціями, які доступні користувачеві.

Цей модуль має основні складові, такі як Клас Menu (tk.Menu), який успадковує клас Menu з бібліотеки tkinter і розширює його функціональність.

Наступними складовими цього файлу є методи, які роблять ініціалізацію елементів системи. Метод `__init__` є конструктором класу, який ініціалізує об'єкт меню. Метод `init_ui` ініціалізує користувацький інтерфейс меню. Він створює три підменю: для задач, проєктів та довідки.

Першим підменю є Задачі (Tasks). Воно містить підпункти для створення нової задачі, редагування та видалення існуючих. Другим підменю є Проєкти (Projects). Який містить підпункти для створення нового проєкту, редагування та видалення існуючих. І третім підменю є Довідка (Help), що містить підпункт для отримання інформації про програму.

Останнім методом файлу `menu.py` є метод типу `_triggered`. Такий метод викликається при виборі відповідного пункту меню і відображають повідомлення з інформацією про виконання конкретної дії (створення, реагування, видалення тощо).

Файл `task_project_forms.py` містить у собі форми для створення, редагування та видалення задач і проєктів. Форми забезпечують практичний спосіб взаємодії користувачів з системою.

У цьому файлі як і у `menu.py` використовуються бібліотеки `tkinter`, `ttk` (для використання віджетів з покращеною стилізацією), а також `sqlite3` (для роботи з базою даних SQLite). Також у файлі міститься головний клас `TaskProjectForms`. Цей клас відповідає за створення форм для створення нових задач і проєктів. Окрім методів, які наявні у `menu.py`, таких як `__init__`, `init_ui`. `Task_project_forms.py` має методи `create_task` та `create_project`, які викликаються при натисканні відповідних кнопок для створення задач та проєктів. Вони отримують дані з віджетів, після їхньої перевірки на валідність, додають до бази даних SQLite.

Файл `lists_tables.py` є частиною користувацького інтерфейсу комп'ютерної системи управління задачами та проєктами. Його основна функція полягає у реалізації списків та таблиць для відображення інформації про завдання та проєкти.

Головним класом файлу є ListsTables. Клас ListsTables виконує функцію управління списками та таблицями для відображення інформації про завдання та проєкти в користувацькому інтерфейсі комп'ютерної системи управління задачами та проєктами. Він об'єднує всі необхідні функції та методи для створення, додавання, редагування та видалення завдань та проєктів у відповідних списках.

Основні функції та методи класу ListsTables відповідають за:

- ініціалізацію користувацького інтерфейсу;
- створення та налаштування списків (таблиць) для відображення завдань та проєктів;
- додавання нових задач та проєктів у відповідні списки;
- редагування тексту задач та проєктів;
- видалення вибраних задач та проєктів;
- відображення контекстного меню з опціями редагування та видалення елементів списків;
- обробку подвійного кліку мишею для редагування елементів списків;
- обробку правого кліку мишею для відображення контекстного меню.

Конструктор `__init__` ініціалізує об'єкт класу, встановлюючи заголовок та розміри вікна. Метод `init_ui` налаштовує користувацький інтерфейс, створюючи та налаштовуючи два списки для відображення задач та проєктів. Функції `add_task` та `add_project` додають нові елементи до відповідних списків. Методи `edit_task` та `edit_project` дозволяють редагувати вміст обраних елементів списків. Функція `show_context_menu` відображає контекстне меню з опціями редагування та видалення елементів списків. Метод `delete_item` видаляє обраний елемент зі списку. Крім того, клас обробляє подвійний та правий кліки мишею для редагування та відображення контекстного меню відповідно.

І головною відмінністю цього файлу від попередніх є використання віджетів `ttk.Treeview` для створення списків з деревоподібною структурою. Вони використовуються для відображення інформації про завдання та проєкти. `Main_window.py` — це головне вікно програми, яке надає користувачу доступ до

основного функціоналу. Цей файл є центральним елементом. Основною функцією цього файлу є створення графічного інтерфейсу за допомогою бібліотеки `tkinter`.

Клас `MainWindow` успадкований від `tk.Tk`, дозволяє йому створити вікно програми. У конструкторі класу виконується налаштування заголовку вікна за допомогою `self.title()` та розмірів вікна за допомогою `self.geometry()`.

Далі відбувається ініціалізація меню за допомогою імпортованого класу `Menu` з файлу `menu.py`. Меню додається до головного вікна програми за допомогою `self.config(menu=self.menu)`.

Після цього створюється `ttk.Notebook` — віджет, що дозволяє розміщувати вкладки з вмістом. Цей віджет використовується для організації різних частин програми.

Дві вкладки додаються до `ttk.Notebook`: одна для форм задач та проєктів, інша для списків та таблиць. Кожна вкладка має свій окремий `ttk.Frame`, на якій розміщується відповідний вміст. Вкладка з формами завдань та проєктів містить екземпляр класу `TaskProjectForms`, який відповідає за створення форм для введення даних про завдання та проєкти.

Вкладка зі списками та таблицями містить екземпляр класу `ListsTables`, який відповідає за створення списків та таблиць для відображення інформації про завдання та проєкти.

На завершальному етапі викликається метод `pack()` для `ttk.Notebook`, який розміщує його у вікні програми.

Умовний оператор `if __name__ == "__main__":` визначає, що код буде виконуватися тільки у разі прямого запуску файлу `mainwindow.py`, а не при імпорті. Він створює екземпляр класу `MainWindow` та запускає головний цикл програми за допомогою `mainloop()`.

Модуль управління задачами у своєму складі має 9 файлів, які забезпечують ефективне управління задачами у межах проєкту. Це найоб'ємніший компонент системи управління проєктами.

Файл `task_management.py` — це модуль, який містить клас для керування задачами в системі. Він містить у собі методи для створення, редагування та

видалення завдань. Він відповідає за основний функціонал управління задачами всієї системи.

`Task_creation_form.py` — форма, що дозволяє користувачам створювати нові задачі. Вона містить поля для введення даних про нову задачу, такі як назва, опис, термін виконання тощо. Цей файл імпортує бібліотеку Tkinter під псевдонімом `tk`, клас з бібліотеки Tkinter для роботи з віджетами. А також імпортує клас `Database` з модуля `appdatabase`.

У файлі створюється екземпляр бази даних, потім оголошується клас `TaskCreationForm`. Цей клас є підкласом `tk.Toplevel` з імпортованої бібліотеки Tkinter.

Метод `__init__` класу `TaskCreationForm` — це конструктор класу, який викликається при створенні нового екземпляру `TaskCreationForm`.

Цей метод створює вікно з назвою "Create New Task". А також він ініціалізує елементи введення (`ttk.Entry`) та кнопки (`ttk.Button`), й розміщує елементи на вікні за допомогою методу `grid`.

А метод `create_task` класу `TaskCreationForm` отримує значення введених даних з полів введення, додає нову задачу до бази даних викликом методу `add_task` об'єкта `db.task_table`. А вже після створення завдання закриває вікно форми.

Третім файлом Модулю управління задачами є `task_edit_form.py`. У цьому файлі також імпортується бібліотека Tkinter, класи з бібліотеки Tkinter для роботи з віджетами. У файлі `task_edit_form.py` всі компоненти взаємодіють для створення форми редагування параметрів існуючих задач. Компонентами програми є мітки `ttk.Label` для відображення тексту перед полями вводу, поля вводу для введення нових значень параметрів задач.

Для отримання інформації про задачі наявний параметр `task_id`, який передається у конструктор класу `TaskEditForm`, використовується для отримання інформації про задачі з бази даних.

А метод `get_task_by_id(task_id)` об'єкта бази даних використовується для отримання даних про задачу за його ідентифікатором. У внутрішніх реалізаціях класу `Database`, використовується запит до бази даних. В результаті виконання

запиту об'єкт бази даних повертатиме дані про задачу з відповідним ідентифікатором, які потім використовуються для заповнення полів введення на формі редагування завдання.

Також у файлі `task_edit_form.py` міститься логіка редагування та збереження завдання.

Процес редагування виконується наступним чином:

- метод `save_task()` визначає нові значення для назви та опису завдання, отримані з полів вводу;
- оновлені дані зберігаються у базі даних за допомогою методу `update_task(task_id, new_name, new_description)`.

Файл `assign_tasks.py` реалізує ефективний розподіл задач між користувачами системи управління задачами, спрощуючи процес призначення та відстеження виконання задач у проєкті. Файл у собі містить клас `AssignTasksDialog`, який відповідає за призначення та розподіл завдань між учасниками проєкту. При створенні об'єкта класу `AssignTasksDialog` передаються параметри `parent`, `users` та `database`.

Параметр `parent` вказує на батьківське вікно, зазвичай це основне вікно програми. Другий параметр `users` є списком доступних користувачів, яких можна призначити для завдання. І параметр `database` представляє об'єкт бази даних, в якій зберігається інформація про завдання та їх відповідальних користувачів. У `assign_tasks.py` наявні методи `init_ui` та `assign_task`. У методі `init_ui` створюються всі необхідні елементи інтерфейсу, такі як мітки, випадаючі списки. Також створюються кнопки "Assign Task" (Призначити завдання) і "Cancel" (Скасувати).

Метод `assign_task` викликається при натисканні кнопки "Assign Task". Він отримує дані про обрану задачу і обраного користувача з відповідних випадаючих списків. Потім відбувається логіка для призначення обраної задачі обраному користувачу. Отримані дані передаються у базу даних для оновлення інформації про завдання.

Файл `user_list.py` призначений для вибору відповідального користувача для кожного завдання. Код файлу містить клас `UserListDialog`, який використовується

для вибору відповідального за завдання користувача зі списку доступних користувачів. При створенні об'єкта класу `UserListDialog` передаються параметри `parent`, `users` та `db`.

Параметр `parent` вказує на батьківське вікно, зазвичай це основне вікно програми. Параметр `users` є списком доступних користувачів. Параметр `db` представляє об'єкт бази даних, до якої буде додано вибраного користувача.

У програмі `user_list.py` є метод `init_ui`, який ініціалізує компоненти інтерфейсу користувача. А також є метод `select_user`, який викликається при натисканні кнопки "Select". Він отримує індекс вибраного користувача зі списку із віджета `Listbox`. Якщо користувач вибрав користувача, отримані дані передаються до бази даних для подальшої обробки. Після чого вікно закривається.

Компоненти файлу `user_list.py` залежать один від одного наступним чином:

- клас `UserListDialog` використовується для створення вікна вибору користувача з списку;
- у методі `init_ui` створюються всі необхідні елементи інтерфейсу, такі як мітка, список та кнопки;
- метод `select_user` відповідає за обробку вибору користувача і передачу даних до бази даних;
- файл може використовуватися в системі управління задачами

Файл `notification.py` відповідає за механізм надсилання сповіщень користувачам про призначення нових завдань. Цей файл містить функцію `send_notification`, яка використовується для відправлення електронних повідомлень про нові завдання.

Основними компонентами файлу є функція `send_notification`. Ця функція призначена для відправлення електронних повідомлень про нові завдання.

Вона отримує наступні параметри:

- `receiver_email`: електронна адреса одержувача повідомлення;
- `subject`: тема повідомлення;
- `message`: текст повідомлення.

Для відправлення електронних повідомлень функція `send_notification` використовує бібліотеку `smtplib`, яка надає інтерфейс для роботи з SMTP-серверами.

Повідомлення створюється як об'єкт `MIMEMultipart`, до якого додаються тема та текст повідомлення. Потім відправляється електронне повідомлення за допомогою методу `sendmail` об'єкта `SMTP`. У функції вказується електронна адреса та пароль від облікового запису, з якого буде відправлено повідомлення за допомогою змінних `sender_email` та `password`.

Також у коді передбачено обробку винятків для виявлення помилок під час відправлення повідомлень. У випадку помилки виводиться відповідне повідомлення.

Компоненти файлу `notification.py` взаємодіють одна з одною шляхом передачі даних від функції `send_notification` до SMTP-сервера через бібліотеку `smtplib`. Після виклику функції з відповідними параметрами електронне повідомлення надсилається на вказану адресу одержувача з вказаною темою та текстом повідомлення.

У програмі `task_deadlines.py` створюється клас `TaskDeadline`. Цей клас відповідає за створення вікна для встановлення терміну виконання та пріоритету завдання. Клас ініціалізується як підклас `tk.Toplevel`, щоб створити окреме вікно для налаштувань. Має метод `init_ui()`, який створює всі компоненти інтерфейсу користувача та розміщує їх на формі. А також метод `save_deadline_priority()`, який отримує дані з компонентів та виводить їх на екран. Ці дані можуть бути використані для подальшого збереження у базі даних, передачі на сервер або обробки в інших частинах системи.

Файл `deadline_priority_form.py` забезпечує можливість користувачу встановлювати термін виконання та пріоритет завдання за допомогою відповідних елементів введення на формі, і після натискання кнопки "Save" ці дані можуть бути оброблені або збережені.

Спочатку у файлі `deadline_priority_form.py` створюється клас `DeadlinePriorityForm`, який відповідає за відображення форми для встановлення

терміну виконання та пріоритетів завдань. Цей клас успадковується від класу `tk.Toplevel`, що означає, що він є окремим вікном, яке може відображатися поверх основного вікна програми. Він містить методи для створення та управління елементами введення форми.

У програмі є головні елементи, які дозволяють користувачеві встановлювати термін виконання завдання. Користувач також може обирати дату за допомогою календаря. Головними елементами є мітка `deadline_label` та календар `calendar`.

Також є мітка `priority_label` та комбінований список `priority_combobox`. Ці елементи дозволяють користувачеві вибрати пріоритет завдання з варіантів "Low", "Medium" та "High".

І центральним методом є `save_deadline_priority()`. Цей метод викликається при натисканні на кнопку "Save". Він отримує значення терміну виконання з календаря та обраного пріоритету з комбінованого списку.

У файлі `calendar_display.py` створюється клас `CalendarDisplay`, який відповідає за відображення календаря та вибраної дати.

Цей клас є основою програми. Він успадковується від класу `tk.Frame` і містить методи для ініціалізації користувацького інтерфейсу та відображення вибраної дати. Потім створюється об'єкт класу `Calendar` з бібліотеки `tkcalendar`. Цей календар дозволяє користувачеві вибирати дати. Також присутня мітка `selected_date_label`, яка відображає обрану користувачем дату. По замовчуванню мітка містить текст "Selected Date: ".

Також у файлі є методи `init_ui()` та `show_selected_date()`. Метод `init_ui()` ініціалізує користувацький інтерфейс. Він створює календар та мітку, і підключає обробник події для вибору дати. А метод `show_selected_date()` викликається при виборі користувачем дати на календарі. Він отримує вибрану дату з календаря та оновлює текст мітки, відображаючи обрану дату.

Увесь функціонал програми зосереджений на взаємодії з користувачем через календар та відображенням обраної дати.

Третім компонентом є База даних (Appatabase). Важливість використання цього компоненту є забезпечення необхідною інформацією модулів

Користувачького інтерфейсу та Модулю управління задачами для виконання їхніх функцій і взаємодії з користувачем.

База даних містить у собі головний файл `appdatabase.py`, який відповідає за зберігання та управління даними, що стосуються системи управління задачами. Цей головний файл імпортує такі файли як, `task_table.py`, `project_table.py`, `user_table.py`.

Файл `task_table.py` містить опис таблиці бази даних, у якій зберігається інформація про задачі. Код цього файлу має методи для створення таблиці, додавання задач, отримання всіх задач та інші операції з базою даних.

Методи `create_table`, `add_task`, `get_all_tasks` створюють таблицю в базі даних, додає нове завдання до таблиці та повертає всі завдання з таблиці відповідно.

Файл `project_table.py` представляє собою таблицю бази даних, у якій зберігається інформація про проекти, які створюються користувачем.

він містить у собі клас `TaskTable`. Цей клас відповідає за управління таблицею даних про завдання.

Також файл містить наступні методи:

- `__init__`, який ініціалізує об'єкт класу `TaskTable`, підключається до бази даних та викликає метод `create_table`;
- `create_table` — створює таблицю в базі даних, якщо вона ще не існує;
- `add_task` — додає нове завдання до таблиці;
- `get_all_tasks` — повертає всі завдання з таблиці.

У файлі `user_table.py` відповідно є таблиця бази даних про користувачів. Головним класом є Клас `UserTable`, який відповідає за управління таблицею даних про користувачів. А також у файлі є методи, які подібні до методів з файлів `project_table.py` та `task_table.py`, але відносяться до користувача. Це `create_table`, `add_user` й `get_all_users`.

Файли `appdatabase.py`, `task_table.py`, `project_table.py` та `user_table.py` разом створюють ієрархію для зберігання та обробки даних, а їх методи дозволяють реалізувати роботу з базою даних.

Всі компоненти розроблюваної комп'ютерної системи взаємодіють, аби забезпечити функціональність програми. Таким чином, Користувацький інтерфейс взаємодіє з Модулем управління задачами та базою даних, відображаючи інформацію про задачі та проєкти, а також дозволяє користувачам створювати, редагувати та видаляти їх. Модуль управління задачами взаємодіє з Користувацьким інтерфейсом та базою даних, обробляючи дії користувача щодо створення, редагування та видалення завдань. А також відповідає за призначення та розподіл завдань між учасниками проєкту. База даних забезпечує зберігання інформації про задачі, проєкти, користувачів та іншу важливу інформацію, яка необхідна для функціонування системи управління. А взаємодія з іншими компонентами системи необхідна для зчитування та зберігання даних.

Зв'язки між компонентами забезпечують користувачам зручний інструмент для управління їхніми проєктами, дозволяючи організовувати роботу та спрощувати процес управління проєктами.

Архітектура системи управління задачами та проєктами представлена простим і зручним способом візуалізації у формі ASCII-графіки (рис. 3.2). Цей підхід дозволяє з легкістю передати структуру системи та взаємозв'язки між її компонентами, забезпечуючи зрозуміле уявлення про архітектурну концепцію проєкту.

Така архітектура системи дозволяє дієво організувати роботу з задачами та проєктами, забезпечуючи ефективне управління системою та взаємодію користувачів з системою. Розроблена компонентизація допомагає зберігати код чистим та організованим, що полегшить підтримку та розвиток системи у майбутньому.

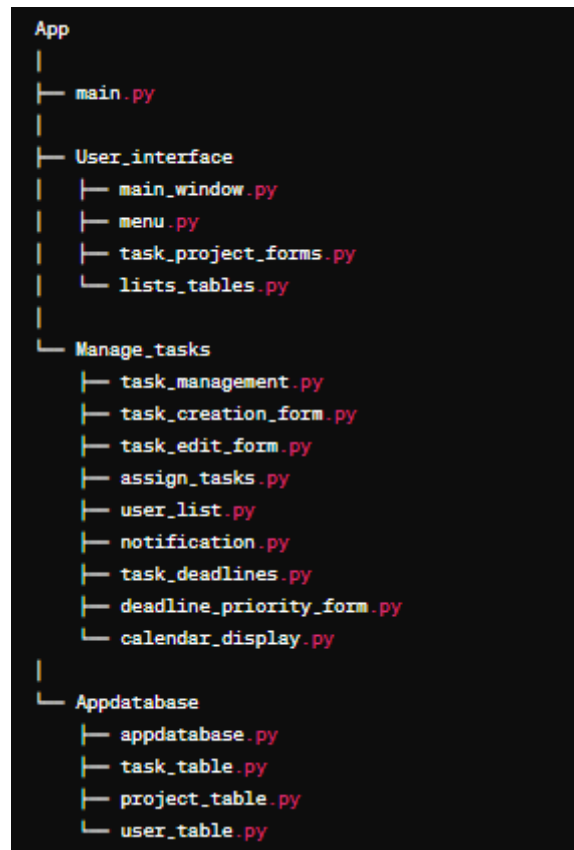


Рисунок 3.2 — Архітектура системи управління за допомогою ASCII-графіки

Створення ефективної структури системи управління було ключовим етапом у процесі створення інструменту, який організовує роботу користувачів та допомагає досягати поставлених цілей.

Чітке розуміння внутрішньої логіки системи, її компонентів та взаємозв'язків дозволяє дієво працювати над покращенням та розвитком продукту в майбутньому. Ретельне проектування архітектури забезпечує не лише потужний функціонал, але й забезпечує зручність та ергономічність для кінцевого користувача, що є критичним фактором для успіху будь-якої системи.

3.2 Програмне забезпечення

Ключову роль у розробці комп'ютерних систем відіграє програмне забезпечення. Воно забезпечує необхідними інструментами та середовищем системи. Застосування програмного забезпечення дозволяє автоматизувати багато аспектів розробки, що сприяє прискоренню процесу створення програм та зменшенню кількості помилок. Більше того, програмне забезпечення забезпечує зручний механізм підтримки стандартів розробки, що сприяє сумісності та безпеці програм та систем.

Програмне забезпечення — це набір інструкцій або програм, які вказують комп'ютеру, що робити. Воно не залежить від апаратного забезпечення і робить комп'ютери програмованими.

Існує три основні типи програмного забезпечення:

- системне програмне забезпечення;
- програмне забезпечення для програмування;
- прикладне програмне забезпечення.

Розроблювана комп'ютерна система управління задачами та проєктами є типовим прикладом прикладного програмного забезпечення (ППЗ). ППЗ — це програми або додатки, які допомагають користувачам виконувати певні завдання або вирішувати конкретні проблеми. Вони розроблені з урахуванням конкретних потреб користувачів та забезпечують широкий спектр функціональності для підтримки їхньої діяльності.

Прикладне програмне забезпечення для управління задачами та проєктами може включати в себе такі функції, як створення, організація та відстеження завдань, розподіл ресурсів та призначення відповідальностей, планування та встановлення дедлайнів, а також звітність та аналітику щодо прогресу проєктів. Вони дозволяють ефективно керувати робочим процесом, зменшуючи час та зусилля, необхідні для досягнення поставлених цілей.

Таким чином, прикладне програмне забезпечення для управління задачами та проєктами стає невід'ємною складовою сучасного робочого середовища та допомагає організаціям та користувачам досягати своїх цілей ефективно та продуктивно.

Варто також відзначити переваги використання прикладного програмного забезпечення.

Насамперед ППЗ допомагає підвищувати продуктивність праці. автоматизація робочого процесу дозволяє звести до мінімуму ручну роботу.

Наступною перевагою є покращення використання даних. Так як це є важливим аспектом у будь-якій комп'ютерній системі управління проєктами. Навіть при наявності великого обсягу даних у системі, без належного їх використання може виникнути проблема з їх аналізом та використанням для прийняття рішень. Саме тому прикладне програмне забезпечення виявляється надзвичайно дієвим. Завдяки спеціально розробленим функціям, які враховують конкретні потреби та цілі, комп'ютерна система управління дозволить ефективно використовувати наявні дані. Це зменшить ризики та сприятиме підвищенню рентабельності інвестицій в проєкти та задачі.

У розроблюваній системі управління задачами та проєктами використовуються технології та інструменти для забезпечення ефективності та зручності користування. Основні компоненти програмного забезпечення включають мову програмування Python, базу даних SQLite, бібліотеку Tkinter для створення графічного інтерфейсу користувача (GUI) та tkcalendar для роботи з календарем.

Python використовується як основна мова програмування для реалізації всіх компонентів системи. Доцільність використовувати саме цю мову полягає у її простоті, гнучкості та широким вибором бібліотек.

У системі використовується вбудована реляційна база даних SQLite для зберігання інформації про завдання, проєкти та користувачів. SQLite дозволяє зручно та ефективно керувати даними, забезпечуючи високу швидкість та надійність.

Обираючи базу даних для розробки комп'ютерної системи управління задачами та проєктами, було вирішено використовувати SQLite через існуючі переваги.

По-перше, SQLite пропонує атомарні, стійкі до збоїв та долі (ACID) транзакції, що робить його надійним рішенням навіть після аварій системи чи втрати живлення. Крім того, він не вимагає конфігурації або адміністрування, що полегшує процес розгортання та підтримки.

Далі, SQLite має повнофункціональну реалізацію SQL з передовими можливостями, такими як часткові індекси, індекси на вирази, робота з JSON, спільні таблиці та функції вікна, що дозволяє ефективно керувати даними та виконувати складні запити.

Крім того, SQLite зберігає всю базу даних в одному файлі, що робить його ідеальним для використання як формат файлу додатка та підвищує переносимість та доступність даних.

Ще однією перевагою SQLite є його мала розмір коду та простий API, що робить його легким у використанні та інтеграції в проєкти. Крім того, він швидкий, а в деяких випадках навіть швидший за пряму роботу з файловою системою. Таким чином, обираючи SQLite для нашої системи управління задачами та проєктами, ми можемо бути впевнені в його надійності, продуктивності та простоті використання.

У процесі створення програмного коду для системи управління задачами та проєктами імпортувався пакет tkinter —стандартний інтерфейс Python для набору інструментів Tcl/Tk GUI. Пакет tkinter є тонким об'єктно-орієнтованим шаром поверх Tcl/Tk. Головні переваги tkinter полягають у тому, що він швидкий і постачається у комплекті з Python.

Для роботи з календарем у графічному інтерфейсі використовувався модуль tkcalendar. tkcalendar - модуль python, який надає віджети Calendar і DateEntry для Tkinter. Використання цього модулю було доречним, так як події у календарі можна відображати за допомогою спеціальних кольорів, а підказка показує список подій на даний день. tkcalendar сумісний як з Python 2, так і з Python 3. Він підтримує багато налаштувань локалей (наприклад, 'fr_FR', 'en_US', ...), а кольори

можна налаштовувати. Використання цього модулю надає можливість вибору дати та відображення календарних подій, що дозволяє користувачам зручно спланувати та відстежити терміни виконання задач та проєктів.

Модуль управління задачами містить в собі файл `notification.py`, який відповідає за механізм надсилання сповіщень користувачам про призначення нових завдань.

Для реалізації програми відправлення електронних повідомлень було використано SMTP-протокол та стандарт MIME, завдяки імпортуванню бібліотеки `smtplib`.

`Smtplib` — клієнт протоколу SMTP. Модуль `smtplib` визначає об'єкт сеансу клієнта SMTP, який можна використовувати для надсилання пошти на будь-яку машину в Інтернеті з демоном прослуховування SMTP або ESMTP.

SMTP — це протокол прикладного рівня, який дозволяє передавати електронні листи між різними серверами та комп'ютерними мережами. Він робить це, визначаючи правила комунікації.

Протокол електронної пошти виконує функцію ретранслятора, тобто передавача. У кожного є унікальна адреса, що дозволяє знаходити отримувача при пересиланні.

SMTP протокол супроводжує виконання ряду команд у вигляді послідовностей запитів і відповідей.

Основні параметри, які передають SMTP команди:

- MAIL FROM — адреса відправника;
- RCPT TO — отримувача;
- DATA — заголовок і тіло.

Програма передає: адреси, тему і контент. SMTP приймає дані та шукає відповідь на стороні отримувача. Коли SMTP знаходить, на якому поштовому клієнті зареєстрований отримувач, запитує IP, встановлює з'єднання за 25 (587 або 465) портом та здійснює обмін. У випадку, якщо сервер отримувача не відповідає, то здійснюються ще кілька спроб зв'язатися з ним. Якщо не вдається встановити

зв'язок у кілька спроб, то повертається помилка. На цьому його завдання завершується, оскільки за доставку відповідають інші протоколи (POP та IMAP).

MIME є стандартом, який дозволяє включати різноманітні типи медіа у повідомлення електронної пошти. У програмі використовуються класи MIMEText та MIMEMultipart, які дозволяють створювати та формувати текстові та багаточастинкові повідомлення. MIME розширює можливості SMTP, дозволяючи включати в повідомлення такі елементи, як зображення, аудіо або відео, що робить їх більш інформативними та зрозумілими для отримувача.

Отже, розглядаючи ключові аспекти програмного забезпечення розроблюваної системи управління задачами та проєктами, варто підкреслити, що необхідний комплексний підхід до створення програмного продукту, який відповідає вимогам сучасного ринку та задовольняє потреби користувачів.

Описані ключові елементи реалізації програми дають чітке уявлення про технічні аспекти системи. Такий підхід дозволяє не тільки розробити функціональну та ефективну систему, але й забезпечити її подальший розвиток та сумісність з майбутніми технологічними інноваціями. Результатом цього аналізу є впевненість у тому, що прикладне програмне забезпечення, розроблене для цієї системи, є надійною основою для впровадження в реальне середовище управління проєктами та задачами.

3.3 Код програми та опис основних функціональних можливостей системи

Для реалізації функціонування системи управління задачами та проєктами необхідні ключові компоненти програми. Система управління проєктами містить у собі 4 складових. Це головний файл main.py та три компоненти, які відповідають за окремі функції і задачі при розробці системи. Першим компонентом є User_interface, який є користувацьким інтерфейсом. Тобто відповідає за спосіб

взаємодії користувача з системою. Він представляє собою ту частину програми, яка відображається на екрані користувача і дає йому можливість взаємодіяти з програмою шляхом введення даних, вибору опцій, натискання кнопок тощо.

Кожен файл цього компоненту імпортує модуль `tkinter`, який надає інтерфейс до бібліотеки Tk GUI (лістинг 3.1).

Лістинг 3.1 — Імпорт модуля `tkinter`

```
import tkinter as tk
```

`User_interface` містить у собі головне вікно програми — `main_window.py`. Основними компонентами коду є клас `MainWindow`, який відповідає за створення та управління головним вікном програми. Клас `MainWindow` наведено у лістингу 3.2.

Лістинг 3.2 — Клас головного вікна програми

```
class MainWindow(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Task Manager App") # Встановлення заголовка вікна
        self.geometry("800x600") # Встановлення розмірів вікна
        self.init_ui() # Ініціалізація графічного інтерфейсу
```

Функція `init_ui()`, яка створює головний інтерфейс вікна представлена у лістингу 3.3. Вона створює рамку для розміщення віджетів, додає мітку з текстом "Main Window" та кнопку з текстом "Create New Task". Крім того, вона створює головне меню за допомогою класу `AppMenu`, а також встановлює це меню для вікна.

Коли користувач натискає кнопку "Create New Task", викликається функція `open_task_form()`, яка створює нове вікно форми завдань та проєктів за допомогою класу `TaskProjectForms` та запускає його головний цикл подій.

Лістинг 3.3 — Функція ініціалізації графічного інтерфейсу

```
def init_ui(self):
    layout = tk.Frame(self) # Створення рамки для розміщення віджетів
    layout.pack(expand=True, fill="both") # Розміщення рамки у вікні
```

```

self.label = tk.Label(layout, text="Main Window") # Створення мітки
self.label.pack(pady=10) # Розміщення мітки у вікні з відступом

self.new_task_button = tk.Button(layout, text="Create New Task",
command=self.open_task_form) # Створення кнопки
self.new_task_button.pack(pady=10) # Розміщення кнопки у вікні з відступом

self.menu = AppMenu(self) # Створення головного меню
self.config(menu=self.menu) # Встановлення головного меню для вікна

def open_task_form(self):
    task_form = TaskProjectForms() # Створення вікна для форми завдань та проєктів
    task_form.mainloop() # Запуск головного циклу подій для вікна

```

Запуск головного вікна програми реалізується спеціальною конструкцією мови Python, яка використовується для визначення точки входу в програму (лістинг 3.4). Ця конструкція перевіряє, чи файл був запущений як самостійна програма, а не імпортований в інший модуль. Якщо це головний файл програми, тобто якщо спеціальна змінна `__name__` має значення `"__main__"`, то виконується код всередині цього блоку.

Лістинг 3.4 — Запуск головного вікна програми

```

if __name__ == "__main__":
    window = MainWindow()
    window.mainloop()

```

Наступним файлом модулю є `menu.py`. основною частиною цієї програми є клас `AppMenu`. Головний конструктор класу представлено у лістингу 3.5.

Лістинг 3.5 — Конструктор класу `AppMenu` з параметрами

```

def __init__(self, parent=None, task_callback=None,
project_callback=None):
    super().__init__(parent)
    self.init_ui()
    self.create_project_callback = project_callback
    self.create_task_callback = task_callback

```

Цей конструктор приймає три параметри: `parent`, `task_callback`, та `project_callback`. Він спочатку викликає конструктор батьківського класу `Menu`, потім ініціалізує графічний інтерфейс за допомогою методу `init_ui()`, і нарешті встановлює дві змінні `create_project_callback` та `create_task_callback` для обробки подій відповідно до створення нових завдань та проєктів.

У файлі `task_project_forms.py`, який містить у собі форми створення, редагування та видалення завдань та проєктів можна визначити основні частини. Ними є клас `TaskProjectForms`, методи `create_task`, `create_project` та зв'язок з базою даних.

Клас `TaskProjectForms` відповідає за створення головного вікна програми та управління ним. Без цього класу програма не матиме основного графічного інтерфейсу та функціоналу для роботи з задачами та проєктами (представлено у лістингу 3.6)

Лістинг 3.6 — Ініціалізація класу `TaskProjectForms` та підключення до бази даних

```
# Клас TaskProjectForms, який успадковує від класу tk.Tk
class TaskProjectForms(tk.Tk):
    # Конструктор класу
    def __init__(self):
        # Виклик конструктора батьківського класу tk.Tk()
        super().__init__()
        # Встановлення заголовка вікна
        self.title('Task and Project Forms')
        # Встановлення розмірів вікна
        self.geometry('600x400')
        # Ініціалізація користувацького інтерфейсу
        self.init_ui()
        # Підключення до бази даних та виклик методу для створення
        # необхідних таблиць
        self.create_database_connection()
        # Завантаження списку завдань з бази даних
        self.load_tasks()
        # Завантаження списку проєктів з бази даних
        self.load_projects()
```

Методи `create_task` та `create_project` відповідають за створення нових завдань та проєктів в базі даних та їх відображення у графічному інтерфейсі. Код методів наведено у лістингу 3.7.

Лістинг 3.7 — Методи створення задач та проєктів користувачем

```
# Метод create_task
def create_task(self):
    task_name = self.task_name_entry.get().strip()
    task_desc = self.task_desc_entry.get().strip()

    # Перевірка, чи введені дані не є порожніми
    if task_name and task_desc:
        # Виконання вставки даних в таблицю tasks бази даних
        self.cursor.execute('INSERT INTO tasks (name, description) VALUES
        (?, ?)', (task_name, task_desc))
        self.conn.commit()
        # Інформаційне повідомлення про успішне створення завдання
        messagebox.showinfo("Success", "Task created successfully.")
        # Вивід створеного завдання у консоль
        print("Task created:", task_name, task_desc)
        # Оновлення списку завдань у графічному інтерфейсі
        self.tasks_list.insert('', 'end', values=(task_name,
        task_desc)) # Оновлення списку завдань
    else:
        # Повідомлення про помилку, якщо дані неповні
        messagebox.showerror("Error", "Please enter both task name and
        description.")

# Метод create_project
def create_project(self):
    project_name = self.project_name_entry.get().strip()
    project_desc = self.project_desc_entry.get().strip()

    # Перевірка, чи введені дані не є порожніми
    if project_name and project_desc:
        # Виконання вставки даних в таблицю projects бази даних
        self.cursor.execute('INSERT INTO projects (name, description)
        VALUES (?, ?)', (project_name, project_desc))
        self.conn.commit()
        # Інформаційне повідомлення про успішне створення проєкту
        messagebox.showinfo("Success", "Project created successfully.")
        # Вивід створеного проєкту у консоль
        print("Project created:", project_name, project_desc)
        # Оновлення списку проєктів у графічному інтерфейсі
        self.projects_list.insert('', 'end', values=(project_name,
        project_desc)) # Оновлення списку проєктів
    else:
        # Повідомлення про помилку, якщо дані неповні
        messagebox.showerror("Error", "Please enter both project name and
        description.")
```

Останнім файлом модуля User_interface є lists_tables.py. Цей файл відповідає за створення графічного інтерфейсу для відображення списків задач і проєктів, а

також за їхнє редагування та видалення. Основний фрагмент коду, що відповідає за створення графічного інтерфейсу для відображення списків задач і проєктів, це розміщення списків у віджеті `ttk.Treeview` (лістинг 3.8)

Лістинг 3.8 — Ініціалізація віджетів `ttk.Treeview`

```
# Tasks List
self.tasks_list = ttk.Treeview(self, columns=('Task Name', 'Task
Description'), show='headings')
self.tasks_list.heading('Task Name', text='Task Name') #
Встановлення заголовку для стовпчика 'Task Name'
self.tasks_list.heading('Task Description', text='Task
Description') # Встановлення заголовку для стовпчика 'Task
Description'
self.tasks_list.pack(fill=tk.BOTH, expand=True) # Пакування
списку завдань для заповнення та розгортання у вікні
self.tasks_list.bind("<Double-1>", self.edit_task) #
Прив'язування події подвійного кліку до методу edit_task
self.tasks_list.bind("<Button-3>", self.show_context_menu) #
Прив'язування події правого кліку до методу show_context_menu
# Projects List
self.projects_list = ttk.Treeview(self, columns=('Project Name',
'Project Description'), show='headings')
self.projects_list.heading('Project Name', text='Project
Name') # Встановлення заголовку для стовпчика 'Project Name'
self.projects_list.heading('Project Description', text='Project
Description') # Встановлення заголовку для стовпчика 'Project
Description'
self.projects_list.pack(fill=tk.BOTH, expand=True) # Пакування
списку проєктів для заповнення та розгортання у вікні
self.projects_list.bind("<Double-1>", self.edit_project) #
Прив'язування події подвійного кліку до методу edit_project
self.projects_list.bind("<Button-3>", self.show_context_menu) #
Прив'язування події правого кліку до методу show_context_menu
```

Другим компонентом комп'ютерної системи є `Manage_tasks`. Цей компонент містить у собі файли: `task_creation_form.py`, `task_management.py`, `task_edit_form.py`, `assign_tasks.py`, `user_list.py`, `notification.py`, `task_deadlines.py`, `deadline_priority_form.py` та `calendar_display.py`.

Всі ці файли разом утворюють модуль для управління задачами й проєктами в системі. Вони надають функціонал для створення, редагування, видалення та призначення завдань користувачам, а також встановлення дедлайнів і пріоритетів. Вони також включають в себе механізм відправки повідомлень користувачам про

призначення нових завдань і відображення термінів виконання в календарі. Ці файли покликані полегшити організацію і контроль процесу виконання завдань і проєктів в системі.

І у цьому модулі є основні файли, які виконують найважливіші ролі і мають бути розглянуті першочергово.

Файл `task_management.py` містить основний клас для керування задачами. Він відповідає за створення, редагування та видалення завдань.

Основними частинами цієї програми є клас `TaskManagement`, який використовує бібліотеку `tkinter` для створення простого інтерфейсу користувача для керування задачами.

У коді створюються елементи управління (мітки та кнопки), що наведено у лістингу 3.9.

Лістинг 3.9 — Елементи управління в класі `TaskManagement`

```
self.label = ttk.Label(self, text="Task Management Module") #
Створення мітки
self.create_button = ttk.Button(self, text="Create Task",
command=self.create_task) # Виведення кнопки для створення
завдання
self.edit_button = ttk.Button(self, text="Edit Task",
command=self.edit_task) # Створення кнопки для редагування
завдання
self.delete_button = ttk.Button(self, text="Delete Task",
command=self.delete_task) # Створення кнопки для видалення
завдання
```

У програмі елементи на вікні програми розміщуються за допомогою методу `pack()`. Виклик методів об'єктів представлено у лістингу 3.10.

Лістинг 3.10 — Виклик методів об'єктів

```
self.label.pack()
self.create_button.pack()
self.edit_button.pack()
self.delete_button.pack()
```

Методи `create_task()`, `edit_task()`, `delete_task()` визначені для керування завданнями у файлі. Кожен з цих методів виконує певну дію, що відображається у вікні за допомогою виведення в консоль відповідного повідомлення. (лістинг 3.11).

Лістинг 3.11 — Функції на об'єкті класу TaskManagement

```
def create_task(self):
    # Логіка для створення нового завдання
    print("Creating a new task...")

def edit_task(self):
    # Логіка для редагування завдання
    print("Editing a task...")

def delete_task(self):
    # Логіка для видалення завдання
    print("Deleting a task...")
```

Отже, код файлу `task_management.py` створює вікно з назвою "TaskManagement", міткою "Task Management Module" і три кнопки: "Create Task", "Edit Task" та "Delete Task". При натисканні на кожну кнопку відповідний метод виводить повідомлення у консоль.

Наступними основними файлами модулю `Mange_tasks` є `task_creation_form.py` і `task_edit_form.py`. Ці файли містять класи для створення та редагування завдань. Вони важливі для забезпечення можливості користувачам вводити нові завдання та редагувати існуючі.

У файлі `task_creation_form.py` після оголошення класу `TaskCreationForm`, який успадковує від `Toplevel` (вікно верхнього рівня) визначається конструктор класу і створюються елементи введення (лістинг 3.12)

Лістинг 3.12 — Елементи введення

```
self.task_name_label = ttk.Label(self, text="Task Name:")
self.task_name_input = ttk.Entry(self)
self.task_description_label = ttk.Label(self, text="Task
Description:")
self.task_description_input = ttk.Entry(self)
```

Після створення елементів введення, додаються кнопки "Create" та "Cancel". А далі створюється розміщення елементів на вікні за допомогою сітки (grid), що наведено у лістингу 3.13.

Лістинг 3.13 — Розміщення елементів у вікні

```
self.task_name_label.grid(row=0, column=0, sticky="w", padx=5,
pady=5)
self.task_name_input.grid(row=0, column=1, padx=5, pady=5)
self.task_description_label.grid(row=1, column=0, sticky="w",
padx=5, pady=5)
self.task_description_input.grid(row=1, column=1, padx=5,
pady=5)
self.create_button.grid(row=2, column=0, columnspan=2, padx=5,
pady=5)
self.cancel_button.grid(row=3, column=0, columnspan=2, padx=5,
pady=5)
```

Елементи розміщуються у вікні за допомогою сітки з вказанням рядків та стовпців.

Метод `create_task()` викликається при натисканні кнопки "Create". Він отримує значення введених даних, створює об'єкт бази даних, додає нове завдання до таблиці завдань у базі даних, а потім закриває вікно форми (лістинг 3.14).

Лістинг 3.14 — Метод `create_task()`

```
def create_task(self):
    task_name = self.task_name_input.get() # Отримання значення назви
    завдання з поля введення
    task_description = self.task_description_input.get()
    db = Database() # Створення об'єкту бази даних
    db.task_table.add_task(task_name, task_description) # Додавання
    нового завдання до таблиці завдань у базі даних
    self.destroy()
```

У файлі `task_edit_form.py` також як і у файлі `task_creation_form.py` створюється екземпляр бази даних, використовується метод `grid()` для розміщення елементів у вікні за допомогою сітки, відповідно до їх позицій у вікні. Схожістю між цими двома файлами ще є методи `create_task()` / `save_task()`. Обидва файли мають ці методи, які виконують логіку для створення або редагування завдань у

базі даних. У файлі `task_edit_form.py` метод `save_task()` викликається при натисканні кнопки "Save" (лістинг 3.15). Він отримує нові значення назви та опису завдання з полів введення, а потім оновлює відповідний запис у базі даних.

Лістинг 3.15 — Визначення методу `save_task()`

```
def save_task(self):
    # Отримання нових значень змінених даних
    new_name = self.task_name_input.get()
    new_description = self.task_description_input.get()
```

У Модулі управління задачами є файл, який виконує функції призначення та розподілу завдань між користувачами. Він важливий для організації робочих процесів та розподілу завдань між учасниками команди. Цим файлом є `assign_tasks.py`. Програма файлу ініціалізує клас `AssignTasksDialog` (лістинг 3.16).

Лістинг 3.16 — Ініціалізація класу `AssignTasksDialog`

```
class AssignTasksDialog(tk.Toplevel):
    def __init__(self, parent, users, database):
        super().__init__(parent)
        self.users = users
        self.database = database
        self.init_ui()
```

Цей конструктор приймає параметри `parent` (батьківське вікно), `users` (список користувачів) та `database` (екземпляр бази даних). Він ініціалізує атрибути `users` та `database`, а потім викликає метод `init_ui()` для створення інтерфейсу користувача.

І не менш важливою частиною програми є логіка призначення завдання у методі `assign_task()` (лістинг 3.17).

Лістинг 3.17 — Метод `assign_task()`

```
def assign_task(self):
    selected_task = self.task_combobox.get()
    assigned_user = self.user_combobox.get()
    # Логіка для призначення завдання обраному користувачу та
    # передача даних у базу даних
    self.database.add_task_to_user(selected_task, assigned_user)
    # Після завершення дії закриття вікна
    self.destroy()
```

Ще одним важливим аспектом системи для сповіщення користувачів про зміни та оновлення в їхніх завданнях є файл `notification.py`. У цьому файлі є функція для відправлення сповіщень про призначення нових завдань.

І останнім компонентом системи є база даних. Модуль `Appdatabase` має у своєму складі 4 файли: `appdatabase.py`, `task_table.py`, `project_table.py` та `user_table.py`.

Файл `appdatabase.py` містить клас `Database`, який ініціалізує та управляє таблицями бази даних для задач, проєктів та користувачів. Він імпортує класи `TaskTable`, `ProjectTable` та `UserTable`, які визначені у відповідних файлах. Цей файл відповідає за ініціалізацію бази даних та встановлення зв'язків з таблицями (лістинг 3.18).

Лістинг 3.18 — Ініціалізація таблиць з файлів

```
# Ініціалізація таблиць
self.task_table = TaskTable()
self.project_table = ProjectTable()
self.user_table = UserTable()
```

У файлі `task_table.py` одним з головних компонентів є імпорт бібліотеки `SQLite`, яка дозволяє взаємодіяти з базою даних `SQLite` у `Python`. У програмі є клас `TaskTable`, який за допомогою методу `__init__`, який є конструктором класу `TaskTable`. Він ініціалізує з'єднання з базою даних та створює таблицю, якщо вона ще не існувала би (лістинг 3.19).

Лістинг 3.19 — Ініціалізація класу `TaskTable` та з'єднання з базою даних

```
def __init__(self, db_name='tasks.db'): # Ініціалізація методу
    __init__ з параметром db_name, який за замовчуванням має значення
    'tasks.db'
    self.db_name = db_name # Присвоєння значення атрибуту db_name
    self.conn = sqlite3.connect(db_name) # З'єднання з базою даних з
    використанням SQLite
    self.create_table() # Виклик методу create_table() для створення
    таблиці при ініціалізації об'єкта
```

У програмі `task_table.py`, як і у `project_table.py` використовуються методи `create_table`, `add_task`, `get_all_tasks` та `create_table`, `add_project`, `get_all_projects` відповідно. Метод `create_table` виконує SQL-запит для створення таблиці `projects` у випадку, якщо вона ще не існує (лістинг 3.20).

Лістинг 3.20 — Метод створення таблиці `projects/tasks`

```
def create_table(self): # Метод для створення таблиці в базі даних
    query = '''CREATE TABLE IF NOT EXISTS projects/tasks
    (
        id INTEGER PRIMARY KEY AUTOINCREMENT, # Поле для унікального
        ідентифікатора завдання
        name TEXT NOT NULL, # Поле для назви завдання, не може бути пустим
        description TEXT # Поле для опису завдання
    )
    self.conn.execute(query) # Виконання SQL-запиту для створення
    таблиці
    self.conn.commit() # Збереження змін у базі даних
```

Метод `get_all_projects` та `get_all_tasks` виконує SQL-запит для отримання всіх записів з таблиці `projects` або та `tasks` повертає список усіх проєктів/задач (лістинг 3.21).

Лістинг 3.21 — Метод отримання всіх проєктів/задач з бази даних

```
def get_all_projects(self):
def add_task(self, name,
description):
    query = 'SELECT * FROM projects/ tasks' # SQL-запит для вибору
    всіх записів з таблиці projects
    cursor = self.conn.execute(query) # Виконання запиту на базі
    даних
    return cursor.fetchall() # Повернення списку всіх проєктів
```

А за додавання нових користувачів у новостворені таблиці відповідає файл `user_table.py`. У цьому файлі, як і в усіх файлах компонента `Appdatabase` імпортується модуль `sqlite3`. Після цього Конструктор класу `UserTable`, який ініціалізує з'єднання з базою даних та викликає метод `create_table()` для створення таблиці користувачів (лістинг 3.22).

Лістинг 3.22 — Метод створення таблиці користувачів

```

def create_table(self): # Метод для створення таблиці користувачів
    query = '''CREATE TABLE IF NOT EXISTS users
    (
        id INTEGER PRIMARY KEY AUTOINCREMENT, # Первинний ключ
        username TEXT NOT NULL, # Поле для імені користувача
        email TEXT # Поле для електронної адреси
    )
    self.conn.execute(query) # Виконання SQL-запиту
    self.conn.commit() # Збереження змін у базі даних

```

Після збереження змін і створення бази даних виконується методи `add_user` та `get_all_users` (лістинг 3.23). Метод `add_user` одає нового користувача до таблиці. Він отримує два параметри: `username` і `email`, які вставляються у відповідні стовпці таблиці за допомогою SQL-запиту `INSERT INTO`. А метод `get_all_users` виконує SQL-запит `SELECT * FROM users`, щоб отримати всі записи з таблиці користувачів. Він повертає список кортежів, кожен з яких містить дані одного користувача.

Лістинг 3.23 — Методи для роботи з таблицею користувачів

```

def add_user(self, username, email): # Метод для додавання нового користувача
    query = '''INSERT INTO users (username, email) # SQL-запит для додавання запису
    VALUES (?, ?)''' # Параметризований SQL-запит
    self.conn.execute(query, (username, email)) # Виконання SQL-запиту з параметрами
    self.conn.commit() # Збереження змін у базі даних

def get_all_users(self): # Метод для отримання всіх користувачів
    query = 'SELECT * FROM users' # SQL-запит для вибору всіх записів
    cursor = self.conn.execute(query) # Виконання SQL-запиту
    return cursor.fetchall() # Повернення результатів запиту у вигляді списку кортежів

```

Структурованість файлів дозволяє дієво взаємодіяти між усіма компонентами розроблюваної комп'ютерної системи. І використання об'єктно-орієнтованого підходу під час створення програми створює стабільну та гнучку систему управління задачами та проєктами, що відповідає потребам користувачів та дозволить ефективно виконувати поставлені завдання і отримувати бажані результати.

4 СТРУКТУРА БАЗИ ДАНИХ ТА ІНТЕРФЕЙС СИСТЕМИ

4.1 Структура бази даних

Розробляючи комп'ютерну систему управління задачами та проєктами, ключовим аспектом зберігання вихідної інформації та її організація. Для цього при розробці комп'ютерної системи була використана реляційна база даних, яка складається з набору таблиць, атрибутів та зв'язків між ними.

Реляційна база даних — це тип бази даних, яка зберігає та надає доступ до точок даних, пов'язаних між собою. Реляційні бази даних базуються на реляційній моделі — інтуїтивно зрозумілому, простому способі представлення даних у вигляді таблиць.

У сфері управління проєктами, реляційні бази даних є дуже важливим інструментом для ефективного відстеження задач, контролю над ресурсами та координації дій команди. Вони дозволяють зберігати великі обсяги даних про проєкти, їх стан і прогрес, а також забезпечують зручний доступ до цієї інформації.

Реляційна база даних може бути корисною для будь-якої потреби в сфері управління проєктами, де дані різних аспектів проєктів, які пов'язані між собою та потребують управління. Це містить у собі відстеження завдань, ресурсів, строків, витрат та багато іншого.

У розробленій системі управління задачами та проєктами створена структурована база даних, що дозволяє зберігати та організовувати інформацію про користувачів, проєкти та їхні задачі.

База даних розробленої комп'ютерної системи складається з 3 таблиць. Ці таблиці визначають структуру та взаємозв'язки між даними системи.

Таблиця «Users» містить інформацію про користувачів системи, зокрема їхні ідентифікатори, імена та електронні адреси. Ця таблиця дозволяє ідентифікувати користувачів та зв'язувати їх зі своїми проєктами та задачами (табл.4.1).

Таблиця 4.1 – Склад відношення «Users»

Атрибут	Тип даних	Значення за умовчанням	Обов'язковість	Примітки
id	Лічильник	-	Так	Первинний ключ
username	Текстовий	-	Так	
email	Текстовий	-	Так	

Таблиця «Projects» містить дані про самі проєкти, такі як їхні назви та описи. Кожен проєкт має унікальний ідентифікатор, який дозволяє однозначно ідентифікувати проєкт в системі. Атрибути наведені у таблиці 4.2.

Таблиця 4.2 – Склад відношення «Projects»

Атрибут	Тип даних	Значення за умовчанням	Обов'язковість	Примітки
id	Лічильник	-	Так	Первинний ключ
name	Текстовий	-	Так	Не може бути порожнім
description	Текстовий	-	Ні	
user_id	Лічильник	-	Ні	Зовнішній ключ, посилання на проєкт

Таблиця «Tasks» визначає задачі, що пов'язані з кожним проєктом. Ця таблиця містить інформацію про назви та описи задач, а також зв'язує їх з конкретними проєктами, які їм відповідають та з користувачами за допомогою зовнішніх ключів.

Таблиця 4.3 – Склад відношення «Tasks»

Атрибут	Тип даних	Значення за умовчанням	Обов'язковість	Примітки
id	Лічильник	-	Так	Первинний ключ
name	Текстовий	-	Так	Не може бути порожнім
description	Текстовий	-	Ні	
project_id	Лічильник	-	Ні	Зовнішній ключ, посилання на проєкт
user_id	Лічильник	-	Ні	Зовнішній ключ, посилання на проєкт

Ці таблиці разом створюють основу для ефективного управління проєктами та задачами в комп'ютерній системі, дозволяючи зберігати, організовувати та взаємодіяти з інформацією зручним та узгодженим способом.

Для отримання бази даних необхідним є визначення зв'язків між сутностями. Виходячи з структури бази даних, можна визначити наступні взаємозв'язки між сутностями:

- «Users» → «Projects» (Б:Б);
- «Projects» → «Tasks» (1:Б);
- «Tasks» → «Users» (Б:1).

Зв'язок між таблицею «Users» і «Projects» є багато-до-багатьох (many-to-many). Кожен проєкт може мати багато користувачів, і кожен користувач може брати участь у багатьох проєктах.

Зв'язок між таблицею «Projects» і «Tasks» також є один-до-багатьох. Кожен проєкт може мати багато завдань, і кожне завдання може належати лише одному проєкту.

Водночас, в таблиці «Tasks», зв'язок з таблицею «Users» існує також як один-до-багатьох. Кожен користувач може мати багато завдань, але кожне завдання належить лише одному користувачеві.

Проаналізувавши всі зв'язки між відношеннями бази даних, що використовуються у системі, можна отримати ER-модель бази даних системи управління задачами та проєктами, яку наведено на рисунку 4.1.

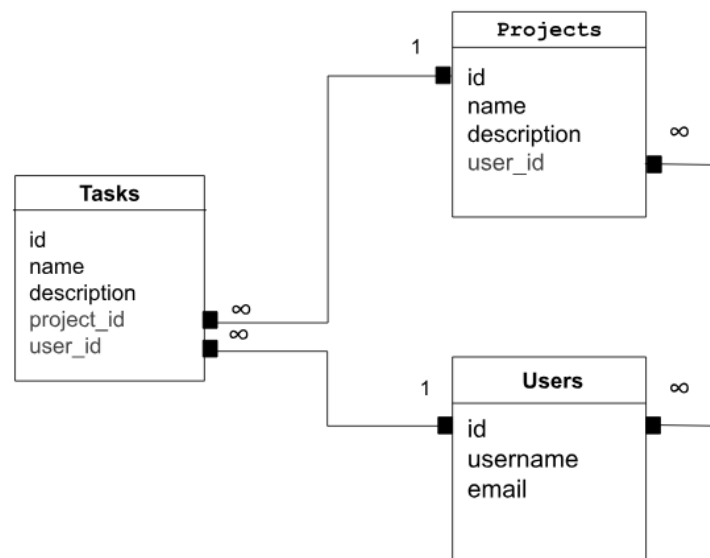


Рисунок 4.1 – ER-модель бази даних системи управління задачами та проєктами

4.2 Розробка інтерфейсу системи

Під час розробки комп'ютерної системи управління задачами та проєктами було обрано мінімалістичний дизайн.

У комп'ютерних технологіях, мінімалізм у контексті комп'ютерних систем означає застосування мінімалістичної філософії та принципів.

Ідея, що лежить в основі цього функціонального мінімалізму, ґрунтується на загальному спостереженні: хороший інструмент часто може робити тільки одну

річ, але робити цю річ досить добре. Концентрація на єдиній меті дозволяє добре адаптуватися до завдання, жертвуючи при цьому гнучкістю. Зосередженість на цілому перекладається як сфокусована компетенція інструменту: оскільки функціонал інструменту чітко визначений, то функціональність інструменту чітко визначена, всі сили можуть бути сконцентровані на вдосконаленні його продуктивності.

Цей підхід передбачає проєктування систем, які використовують якомога менше апаратних і програмних ресурсів, забезпечуючи при цьому ефективну і безперебійну роботу системи.

Такий дизайн було обрано з метою допомогти користувачеві концентруватися на основних функціях, не розсіюючи увагу.

Було дотримано принципів структурного і композиційного мінімалізму для досягнення цієї мети. Структурний мінімалізм спрямований на звільнення користувача від непотрібної навігації, тоді як композиційний мінімалізм допомагає прибрати непотрібні робочі процеси, які можуть завадити його планам дій. Це дозволяє користувачам мати більше свободи і ефективно керувати своїми задачами та проєктами.

Так як завантажений дизайн може зробити навігацію по системі складною через велику кількість деталей та непотрібні деталі. Також недоліком використання не мінімалістичного дизайну є перевантаження інформацією користувача. Використання складних графічних ефектів, анімованих кнопок можуть сповільнювати роботу програми і відволікати користувача від його основної мети — виконання задач і закриття проєктів.

Наголошуючи на доцільності використання у цій системі мінімального дизайну також варто відзначити і інші переваги.

По-перше, перевагою є збільшення доступності. Менше елементів у дизайні може сприяти покращенню доступності для користувачів з обмеженими можливостями або на старих пристроях. Мінімалістичний дизайн може знизити навантаження на обробку графіки та ресурсів пристрою, що робить його доступнішим для більшого кола користувачів.

По-друге, спрощені форми та шрифти. У розробленій системі шрифти були простими, без зайвих деталей. Саме за рахунок цього текст легко читається і користувач буде зосередженим на основній інформації.

Отже, основний фокус розміщений на функціональності системи, а не на її зовнішньому вигляді. Адже важливо, щоб користувач легко знаходив та використовував необхідні інструменти для управління задачами та проєктами.

Головною візуальною частиною комп'ютерної системи є модуль Користувацький інтерфейс, який містить у собі ряд під модулів для забезпечення зручного та ефективного взаємодії користувача з програмою. Навігація у програмі здійснюється за допомогою кнопок у верхній частині вікна та меню. У системі створені форми для роботи із задачами та проєктами, а також є оповіщення про помилки у разі недотримання вимог формування завдань.

Першою складовою Користувацького інтерфейсу є головне вікно комп'ютерної системи (рис.4.2). Це основне вікно програми, яке відображається при запуску програми. На ньому розміщені панель інструментів з кнопками для додавання нових завдань та проєктів, а також списки для відображення завдань та проєктів.

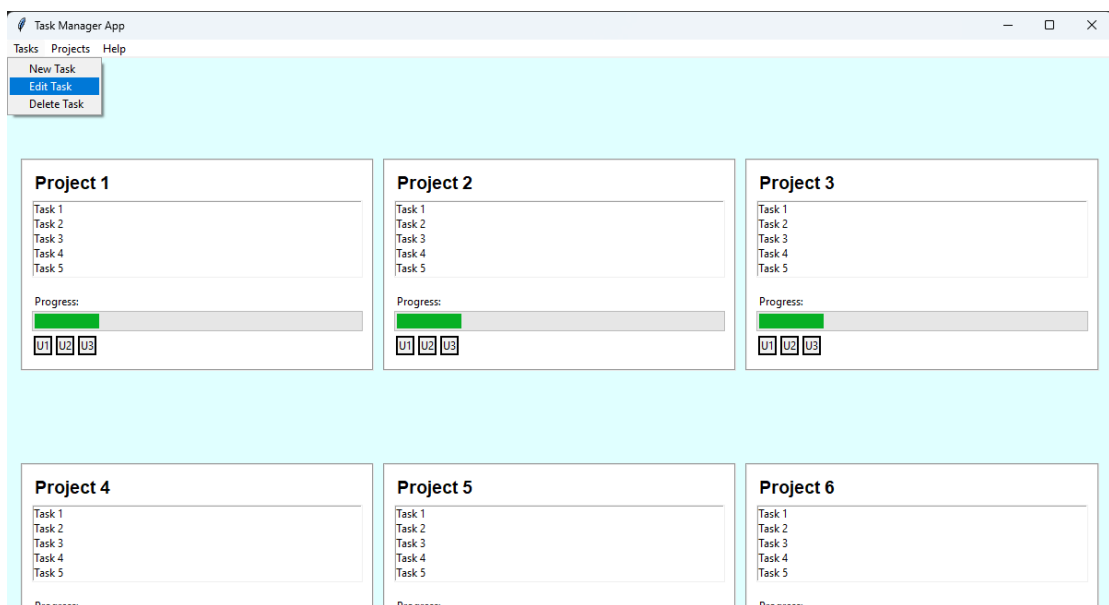


Рисунок 4.2 — Головне вікно комп'ютерної системи

Другою складовою візуальної частини є меню, яке надає доступ до розділів «Tasks», «Projects» та «Help». Реалізація доступу до різних функцій програми через випадаюче меню дає користувачу швидкий і зручний спосіб до потрібних йому опцій та функцій системи (рис.4.3).

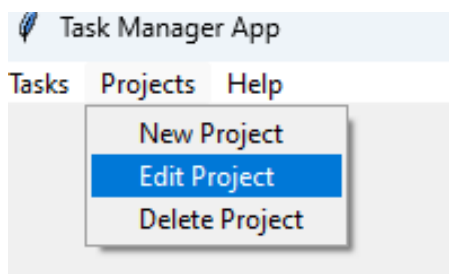
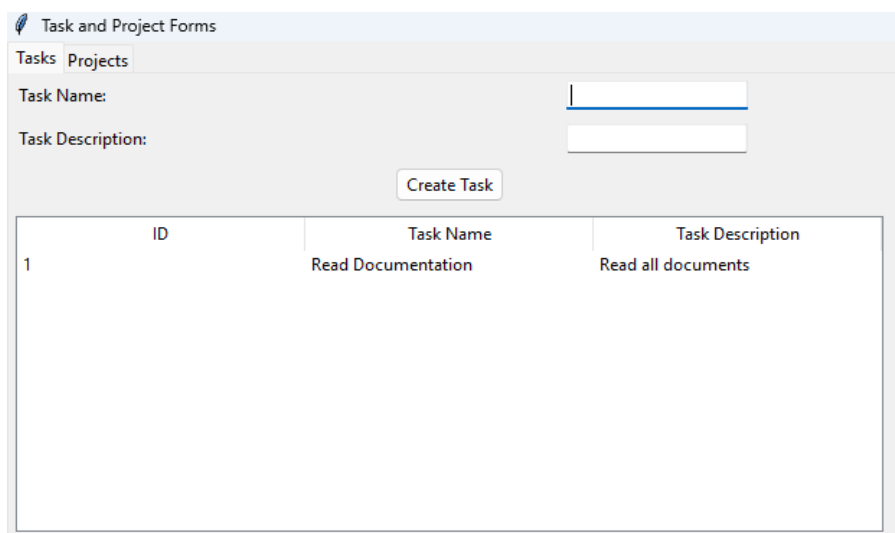


Рисунок 4.3 — Меню системи з обиранням розділу «Projects»

Також доступними до користувача є форми створення, редагування та видалення завдань та проєктів. Ці форми дозволяють користувачеві легко створювати, редагувати та видаляти завдання та проєкти в системі (рис.4.5).

The image shows a software interface titled "Task and Project Forms". It has two tabs: "Tasks" and "Projects", with "Tasks" currently selected. Below the tabs, there are two input fields: "Task Name:" and "Task Description:". Below these fields is a button labeled "Create Task". At the bottom of the form is a table with three columns: "ID", "Task Name", and "Task Description". The table contains one row of data.

ID	Task Name	Task Description
1	Read Documentation	Read all documents

Рисунок 4.5 — Форма створення нового завдання

Створюючи нову задачу необхідно додавати її опис та назву. Так само необхідно зробити при формуванні нового проєкту. За відсутності опису або назви користувача буде сповіщено про помилку і запропоновано ввести всі необхідні дані (рис.4.6)

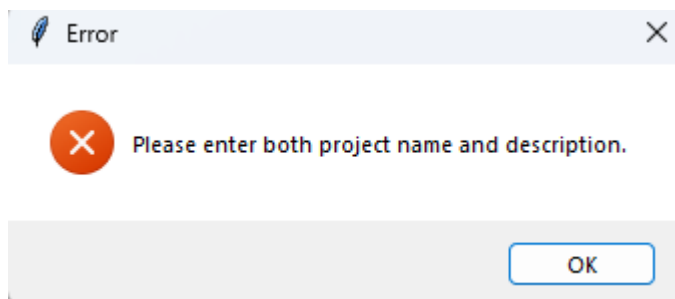


Рисунок 4.6 — Повідомлення про помилку під час створення проєкту

Останньою частиною Користувацького інтерфейсу є списки та таблиці для відображення задач та проєктів (рис.4.7).

Task Name		Task Description	
Project Name		Project Description	

Рисунок 4.7 — Списки та таблиці задач та проєктів

Ключовим елементами дизайну розробленої комп'ютерної системи управління задачами та проектами можна назвати використання мінімалістичних інтерфейсів для спрощеного користування.

4.3 Тестування розробленої системи

Тестування системи є дуже важливим етапом у процесі розробки системи. Тестування необхідне для перевірки системи та підтвердження того, що система відповідає вимогам та специфікаціям. Також не менш важливим є виявлення та виправлення помилок та недоліків перед використання системи. Функціональні та нефункціональні характеристики системи є важливою частиною у процесі тестування для отримання критеріїв оцінки успішності комп'ютерної системи.

Функціональні характеристики описують функції та можливості системи. Вони визначають конкретні дії, які користувачі можуть виконувати за допомогою системи. Ці характеристики є дуже важливими, оскільки вони:

- забезпечують відповідність вимогам;
- слугують основою для тестування системи;
- полегшують розробку та підтримку.

А нефункціональні характеристики описують, як система виконує свої функції. Вони містять у собі важливі аспекти системи, такі як продуктивність, безпека, зручність використання та надійність. Ці характеристики також важливі, тому що вони:

- впливають на користувацький досвід;
- забезпечують відповідність стандартам та захисту даних;
- допомагають забезпечити стабільну роботу системи.

Для проведення тестування розробленої комп'ютерної системи використовувався комп'ютер з наступними характеристиками: процесор Intel Pentium, 4GB RAM, 64-розрядна операційна система на базі процесора x64.

Програмне забезпечення містило у собі операційну систему Windows 10 та Python 3.10. Попередньо були встановлені всі необхідні бібліотеки та залежності через requirements.txt. Тестування проводилось з використанням PyTest — інструментом для автоматизованого тестування, та використовувалася SQLite для роботи з базою даних.

Тестування системи охоплювало кілька ключових сценаріїв для перевірки основної функціональності та надійності роботи всієї системи. Перш за все, перевірявся процес запуску системи, включаючи ініціалізацію головного вікна програми та завантаження меню з доступом до основних функцій. Наступним етапом було тестування сценаріїв створення нових задач: відкриття форми створення задачі, заповнення необхідних полів (назва, опис, терміни виконання, пріоритети) і збереження задачі у базу даних. Перевірялися можливості редагування існуючих задач, зокрема вибір задач зі списку, внесення змін через форму редагування і оновлення даних у базі. Також були протестовані функції призначення задач користувачам, де перевірялося відкриття модуля призначення завдань, вибір відповідального користувача зі списку та оновлення даних у базі з надсиланням відповідних сповіщень.

Додатково тестувалися сценарії встановлення термінів виконання та пріоритетів для задач, зокрема відкриття форми для встановлення цих параметрів, заповнення відповідних полів і оновлення даних у базі, а також відображення термінів на календарі.

Крім того, перевірялися функції відображення списків та таблиць задач і проєктів, де система отримувала дані з бази та відображала їх у відповідному форматі.

Кожен з цих тестів мав свої критерії успішності, які містили у собі виконання заявлених функцій без помилок і в межах допустимих часових рамок, що дозволило переконатися в надійності та ефективності роботи системи.

У процесі тестування системи перевірялося виконання ключових функцій для забезпечення стабільності та коректної роботи системи.

Тестування відбувалося за послідовними діями, такими як:

- а) запуск системи:
 - 1) ініціалізація головного вікна програми (`main_window.py`);
 - 2) завантаження меню (`menu.py`) з доступом до основних функцій;
- б) створення нової задачі:
 - 1) вибір опції створення нової задачі у меню;
 - 2) заповнення форми створення задачі (`task_creation_form.py`);
- в) редагування існуючої задачі:
 - 1) вибір задачі для редагування із списку задач (`lists_tables.py`);
 - 2) внесення змін через форму редагування задач (`task_edit_form.py`);
- г) призначення задачі користувачу системи:
 - 1) вибір задачі для призначення відповідального;
 - 2) вибір користувача із списку (`user_list.py`);
- д) встановлення термінів виконання та пріоритетів задач і проєктів:
 - 1) вибір задачі для встановлення термінів або пріоритетів;
 - 2) заповнення форми встановлення термінів та пріоритетів (`deadline_priority_form.py`).

Очікуваними результатами роботи комп'ютерної системи були:

- система правильно ініціалізує головне вікно і завантажує меню;
- успішне створення, редагування, призначення задач, а також встановлення термінів і пріоритетів без помилок;
- коректне відображення списків задач та проєктів;
- надсилання сповіщень користувачам про створення, редагування та призначення задач.

Після тестування були отримані дійсні результати роботи системи.

На етапі запуску системи, головне вікно ініціалізувалося правильно, меню завантажувалося без помилок (рис. 4.8).

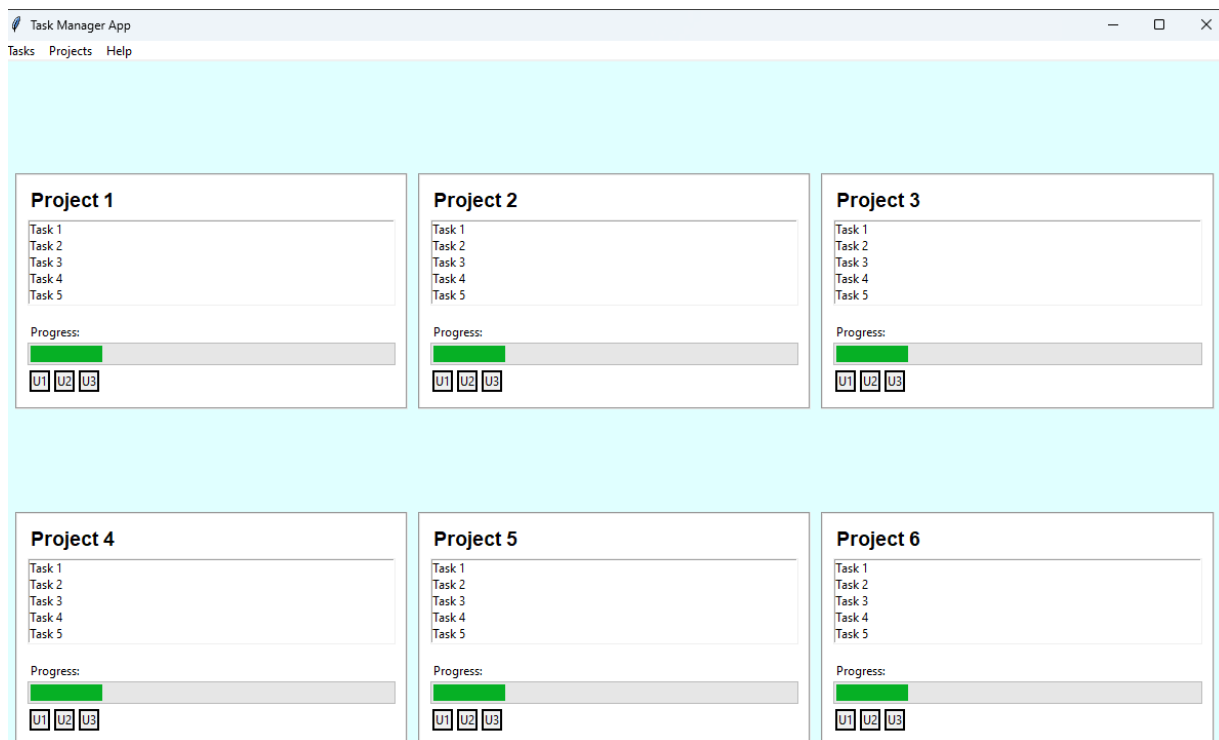


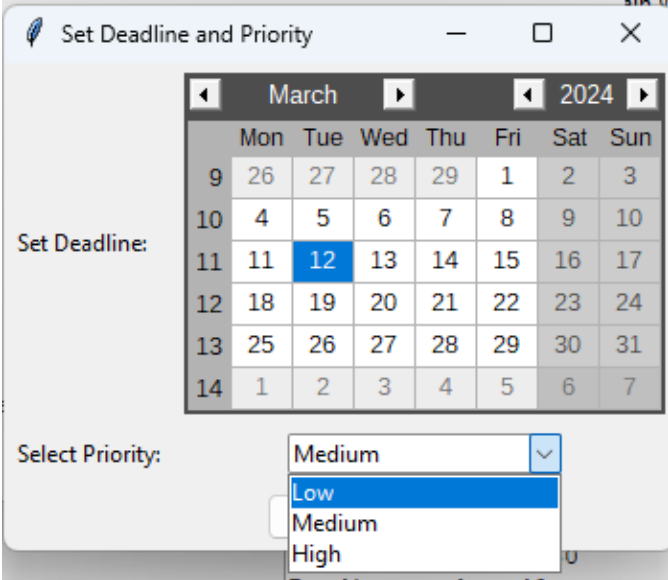
Рисунок 4.8 — Запуск головного вікна та меню системи

Етап створення та редагування задач також пройшов успішно. Форми створення та редагування задач працювали без помилок, дані зберігалися у базі даних (рис.4.9)

ID	Task Name	Task Description
1	Read Documentation	Read all documents

Рисунок 4.9 — Форма для створення задач

Користувач заповнив необхідні поля у формі створення задачі (назва, опис, терміни виконання, пріоритети) (рис.4.10)



Set Deadline and Priority

Set Deadline:

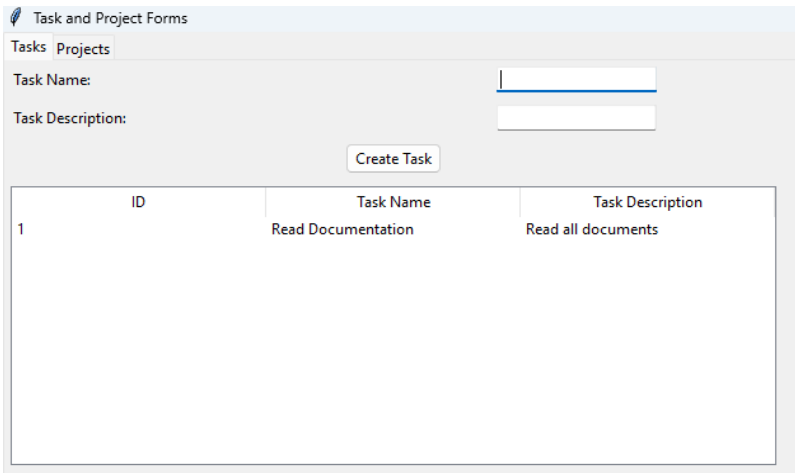
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
9	26	27	28	29	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

Select Priority:

- Medium
- Low
- Medium
- High

Рисунок 4.11 — Списки та таблиці задач та проєктів

Система зберегла нову задачу у базу даних (task_table.py). Потім у таблиці форми створення задачі відобразилася нова задача, яку також баче користувач (рис.4.12).



Task and Project Forms

Tasks Projects

Task Name:

Task Description:

Create Task

ID	Task Name	Task Description
1	Read Documentation	Read all documents

Рисунок 4.12 — Таблиця з новоствореною задачею

Редагування існуючої задачі також відбулося успішно. Користувач обрав задачу для редагування з таблиці (lists_tables.py). Форма редагування задач відкрилася і користувач вніс всі необхідні зміни (рис.4.13).

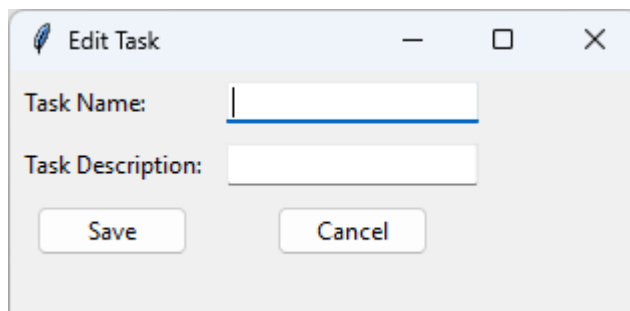


Рисунок 4.13 — Форма редагування задачі

Після внесення всіх змін, система оновила дані задачі у базі даних (task_table.py).

Користувач також вибрав задачу для призначення відповідального за неї. Відкрився модуль призначення задач (assign_tasks.py). Наступним кроком був вибір відповідального із списку користувачів (user_list.py) (рис.4.14).

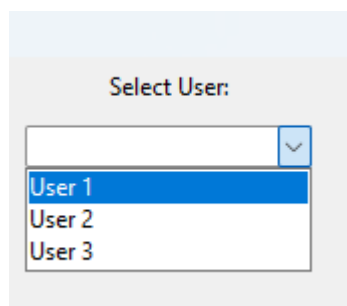


Рисунок 4.14 — Вибір користувача, відповідального за задачу

Система оновила дані про призначення в базі даних і надіслала сповіщення користувачу (notification.py) (рис.4.15)

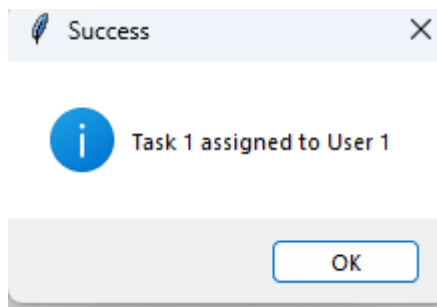


Рисунок 4.15 — Сповіщення про успішність призначення задачі

Отже, списки завдань відображаються правильно, сповіщення надсилаються вчасно.

По завершенню роботи системи проведено аналіз результатів, який показав, що більшість тестових сценаріїв пройшли успішно. Це підтверджує відповідність системи встановленим вимогам.

Отже, функціональними можливостями системи управління задачами та проєктами є:

- а) навігація та доступність:
 - 1) користувачам надається можливість мати навігацію по системі та отримувати доступ до основних функцій для зручного використання;
 - 2) меню швидкого доступу дозволяє користувачам швидко переходити до різних опцій системи, таких як створення, редагування та видалення задач і проєктів;
- б) ефективне керування задачами та проєктами:
 - 1) можливість дієво керувати задачами і проєктами, за допомогою форм;
 - 2) таблиці та списки допомагають відслідковувати стан та прогрес завдань та проєктів;
- в) редагування задач та їхній розподіл:
 - 1) можливість користувачам створювати, редагувати та видаляти задачі;
 - 2) можливість призначати та розподіляти задачі між учасниками проєкту;
- г) керування термінами виконання задач.

Нефункціональними характеристиками комп'ютерної системи є:

- ефективність та швидкодія реакцій та обробки задач та проєктів;
- надійність зберігання даних та стійкість до відновлення в разі аварій за допомогою використання вбудованої реляційної бази даних SQLite;
- безпека даних шляхом зберігання інформації в захищеній базі даних із застосуванням механізмів автентифікації та авторизації доступу до даних;
- система готова до масштабування і може легко пристосовуватися до збільшення обсягів даних та користувацького трафіку без втрати продуктивності;
- зручність використання системи за рахунок мінімалістичного та інтуїтивно зрозумілого інтерфейсу системи;
- використання SQLite для зберігання даних дозволяє зберігати всю базу даних в одному файлі, що полегшує перенесення системи між різними пристроями та платформами;
- використання сучасних технологій та бібліотек гарантує актуальність та доступність нових функцій та оновлень.

Загалом, система відповідає основним функціональним вимогам та успішно пройшла тестування, підтвердивши свою функціональність і відповідність вимогам.

ВИСНОВКИ

У процесі виконання роботи було виявлено, що існуючі системи управління задачами та проєктами мають свої недоліки, які знижують продуктивність команд, які реалізують проєкти. І результати аналізу підкреслюють необхідність вдосконалення та розробки нової системи, яка буде орієнтованою на вирішення конкретних проблем управління проєктами та буде зручною у використанні.

Впровадження комп'ютерної системи управління задачами та проєктами є кроком до підвищення дієвості планування, координації та контролю виконання різних завдань та проєктів. Це дозволить знизити ризики та покращити якість бажаних користувачами результатів. Такий підхід стає важливим для забезпечення конкурентоздатності та успішної діяльності команд та організацій в умовах, де постійно необхідно приймати швидкі рішення та планувати більше нових проєктів.

Недоліки існуючих систем підкреслили необхідність впровадження нових функціональностей та поліпшення користувацького досвіду. Розробляючи нову систему управління було успішно впроваджено нові функції, які не мають проблем з кросплатформеністю, складним інтерфейсом для користувачів.

Отже, розроблена система управління задачами та проєктами відповідає поставленим цілям та вимогам. Її впровадження сприятиме покращенню дієвості та продуктивності управлінських процесів у різних сферах діяльності.

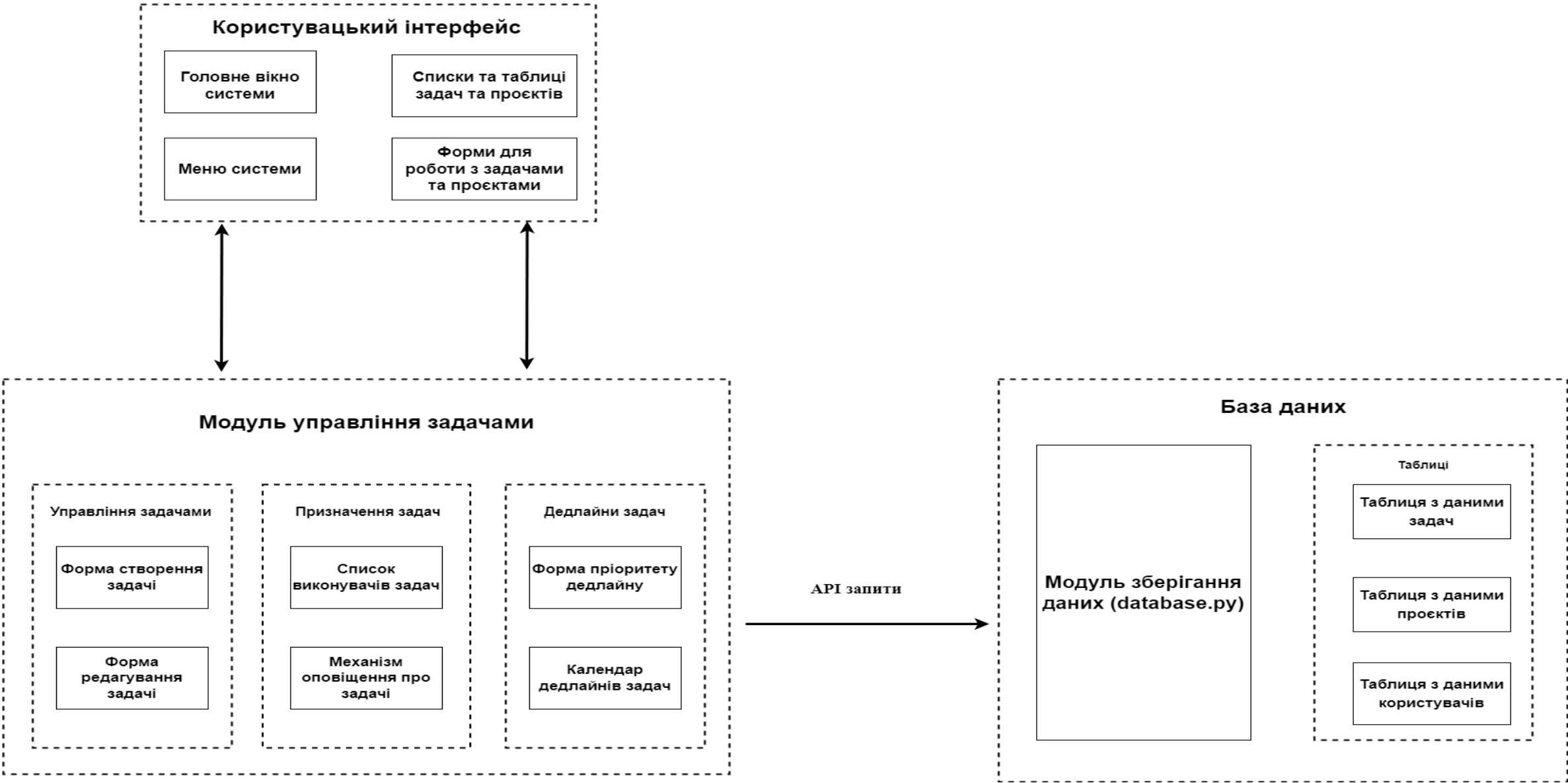
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гайдаржи В. І. Ізварін І. В. Базы даних в інформаційних системах: підручник. Ун-т "Україна", 2018. 418 с.
2. Хігні Дж. Основи управління проєктами: навч. посіб. 2016-те вид. Фабула, 2020. 272с.
3. Architectural styles and the design of network-based software architectures. Home - UC Irvine Donald Bren School of Information & Computer Sciences. URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm> (date of access: 29.03.2024).
4. Association for Project Management Staff. APM body of knowledge. Association for Project Management, 2019.
5. Cojoaca E. S. D. Interoperability of SME databases from the project management point of view. Informatica economica. 2017. Vol. 21, no. 4/2017. P. 15—28. URL: <https://doi.org/10.12948/issn14531305/21.4.2017.02> (date of access: 03.05.2024).
6. Create table. SQLite Home Page. URL: https://www.sqlite.org/lang_createtable.html (date of access: 10.04.2024).
7. Date C. J. Database design and relational theory: normal forms and all that jazz. Apress, 2019. 472 p.
8. Devaux S. A. Managing projects as investments: earned value to business value. Taylor & Francis Group, 2014. 255 p.
9. Graphical user interfaces with tk. Python documentation. URL: <https://docs.python.org/3/library/tk.html#index-0> (date of access: 25.04.2024).
10. Harrin E. Managing multiple projects: how project managers can balance priorities, manage expectations and increase productivity. Kogan Page, Limited, 2022.
11. IBM B2B advanced communications 1.0.0. IBM in Deutschland, Österreich und der Schweiz. URL: <https://www.ibm.com/docs/en/b2badv-communication/1.0.0?topic=concepts-mime-multipart-message-overview> (date of access: 25.04.2024).

12. Importance of application software: a complete guide | hyperlink infosystem. Top Mobile App Development Company USA, India & UAE - Hyperlink InfoSystem. URL: <https://www.hyperlinkinfosystem.com/blog/importance-of-application-software> (date of access: 24.04.2024).
13. Institute P. M. PMBOK guide: the project management body of knowledge. Booksmith Publishing LLC, 2021.
14. Lutkevich B., Wigmore I. What is system software? — definition from whatis.com. WhatIs. URL: <https://www.techtarget.com/whatis/definition/system-software> (date of access: 20.04.2024).
15. Must-Know project management statistics [current data] • gitnux. GITNEX. URL: <https://gitnux.org/project-management-statistics/> (date of access: 23.03.2024).
16. Notebook — ttkbootstrap documentation. ttkbootstrap - ttkbootstrap. URL: <https://ttkbootstrap.readthedocs.io/en/version-0.5/widgets/notebook.html> (date of access: 04.04.2024).
17. Querying — peewee 3.17.3 documentation. peewee — peewee 3.17.3 documentation. URL: <https://docs.peewee-orm.com/en/latest/peewee/querying.html> (date of access: 28.04.2024).
18. Query methods — pygresql 6.0.1. Welcome to PyGreSQL — PyGreSQL 6.0.1. URL: <https://www.pygresql.org/contents/pg/query.html> (date of access: 28.04.2024).
19. Smtplib — клієнт протоколу smtp. Python documentation. URL: <https://docs.python.org/uk/3/library/smtplib.html> (дата звернення: 10.04.2024).
20. Sqlite3 — DB-API 2.0 interface for SQLite databases. Python documentation. URL: <https://docs.python.org/3/library/sqlite3.html> (date of access: 21.04.2024).
21. Traditional vs. object-oriented approaches. Saylor Academy | Saylor Academy. URL: <https://learn.saylor.org/mod/book/tool/print/index.php?id=29145> (date of access: 18.03.2024).

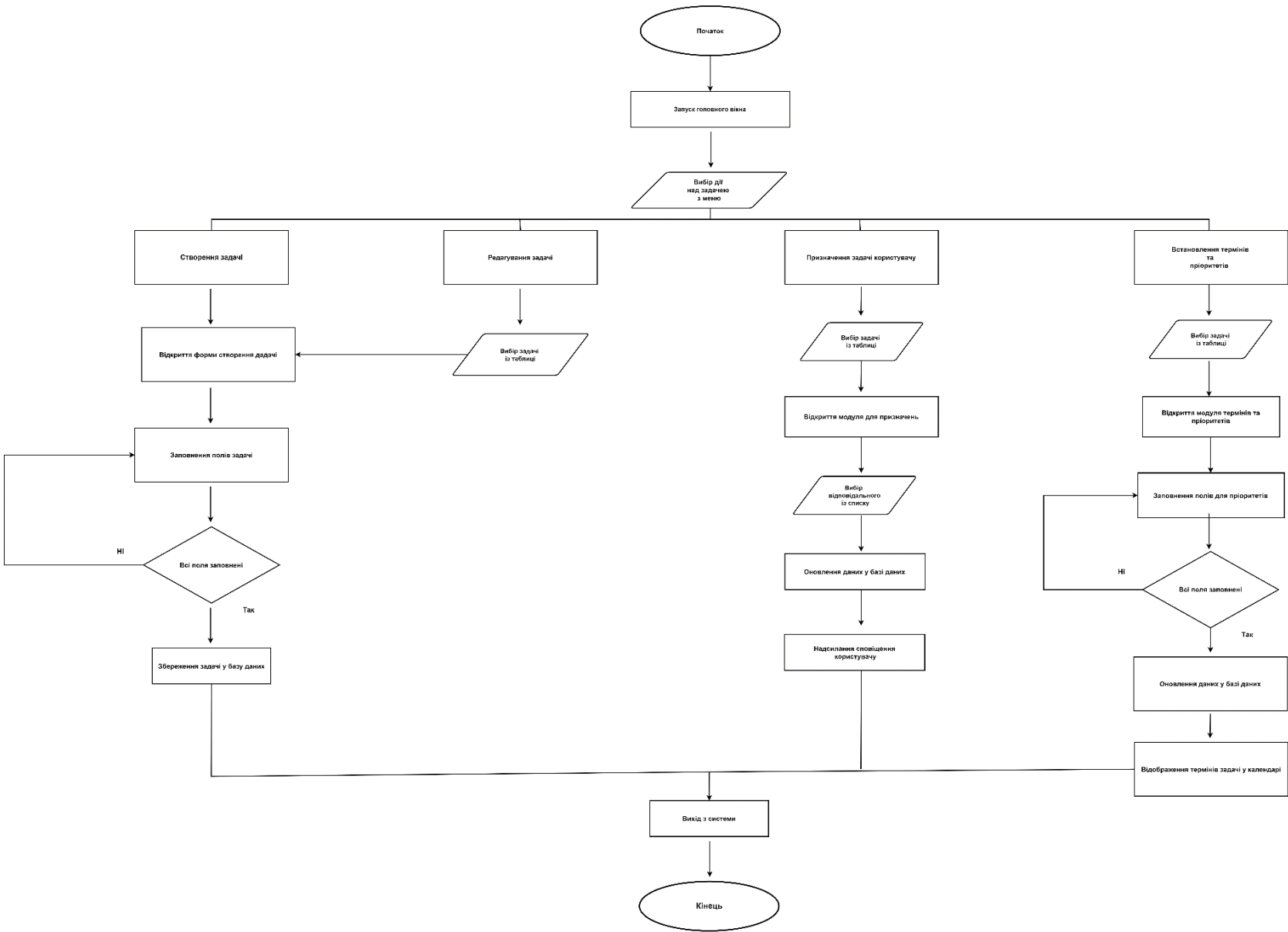
22. What is A relational database (RDBMS)? | google cloud. *Google Cloud*.
URL: <https://cloud.google.com/learn/what-is-a-relational-database> (date of access: 29.04.2024).
23. What is a relational database?. *Oracle | Cloud Applications and Cloud Platform*. URL: <https://www.oracle.com/database/what-is-a-relational-database/> (date of access: 30.04.2024).
24. What is software development? | IBM. IBM in Deutschland, Österreich und der Schweiz. URL: <https://www.ibm.com/topics/software-development> (date of access: 18.04.2024)

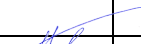
Структурна схема реалізації системи управління задачами та проєктами



					13.02070849.00057 ПЛ1					
Зм.	Лист	№ докум.	Підп.	Дата	Розробка комп'ютерної системи управління задачами та проєктами Структурна схема управління задачами та проєктами	Літ.			Маса	Маштаб
Розроб.		Півень А.С.								
Перевір.		Касьян К.М.								
Т. контр.						Лист 1			Листів 1	
						НУ «Запорізька політехніка» КНТЗ-510				
Н. контр.		Польська О.В.								
Затвердж.		Кудерметов Р.К.								

Алгоритм управління задачами у системі



					13.02070849.00057 ПЛ2					
Зм.	Лист	№ докум.	Підп.	Дата	Розробка комп'ютерної системи управління задачами та проєктами Алгоритм управління задачами у системі	Лім.			Маса	Маштаб
Розроб.		Півень А.С.								
Перевір.		Касьян К.М.								
Т. контр.										
						Лист 1		Листів 1		
Н. контр.		Польська О.В.				НУ «Запорізька політехніка» КНТЗ-510				
Затвердж.		Кудерметов Р.К.								

Екранна форма системи

Task Manager App

TasksProjectsHelp

New Task

Edit Task

Delete Task

Project 1

Task 1
Task 2
Task 3
Task 4
Task 5

Progress:

U1U2U3

Project 2

Task 1
Task 2
Task 3

Progress:

U1U2U3

Project 3

Task 1
Task 2
Task 3
Task 4
Task 5

Progress:

U1U2U3

Project 4

Task 1
Task 2
Task 3
Task 4

Progress:

U1U2U3

Project 5

Task 1
Task 2

Progress:

U1U2U3

Project 6

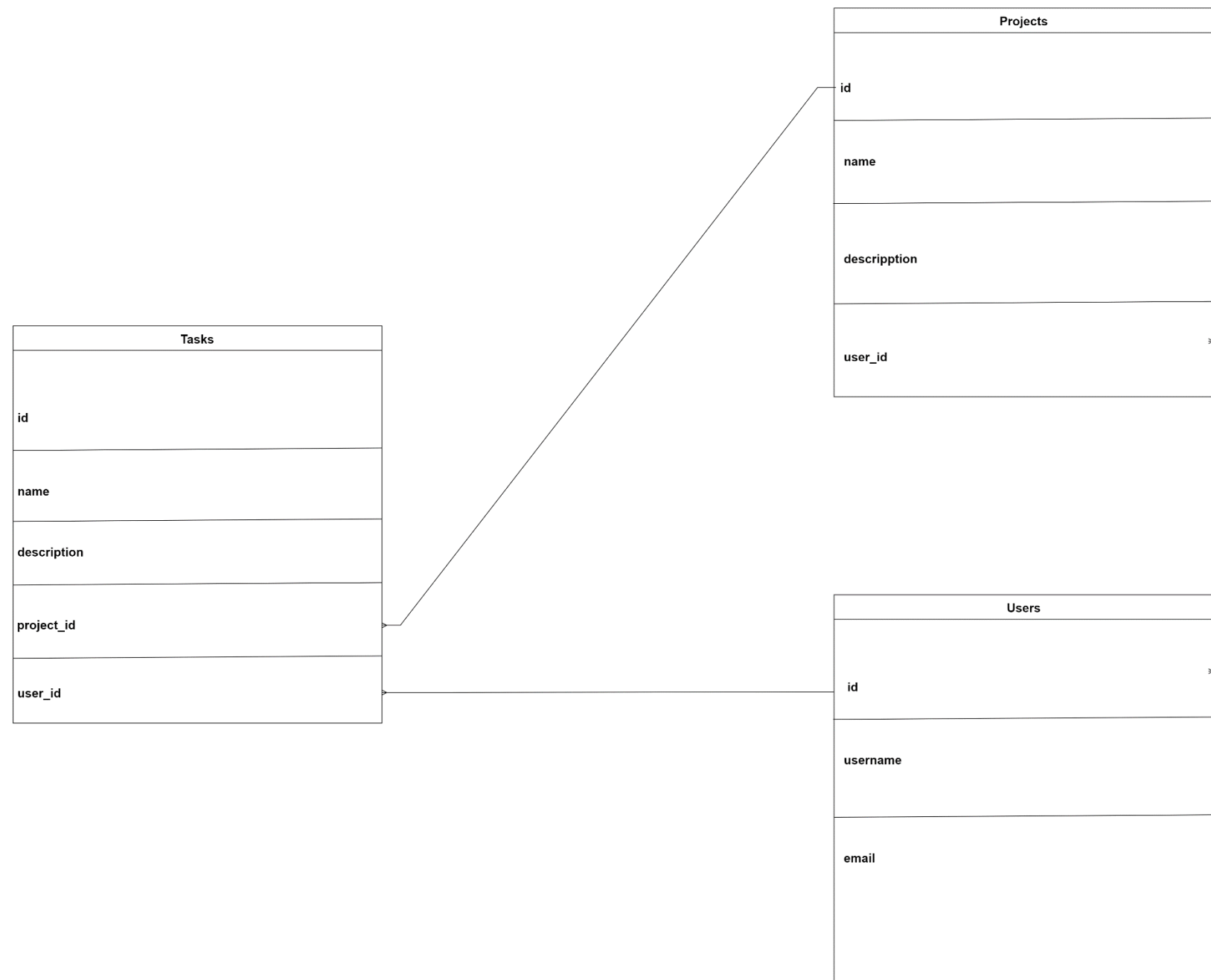
Task 1
Task 2
Task 3
Task 4
Task 5

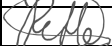
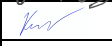
Progress:

U1U2U3

					13.02070849.00057 ПЛЗ					
Зм.	Лист	№ докум.	Підп.	Дата	Розробка комп'ютерної системи управління задачами та проєктами Екранна форма системи	Лім.			Маса	Маштаб
Розроб.		Півень А.С.								
Перевір.		Касьян К.М.								
Т. контр.						Лист 1			Листів 1	
						НУ «Запорізька політехніка» КНТЗ-510				
Н. контр.		Польська О.В.								
Затвердж.		Кудерметов Р.К.								

Реляційна база даних



					13.02070849.00057 ПЛ4						
					Розробка комп'ютерної системи управління задачами та проектами Реляційна база даних	Лім.		Маса	Маштаб		
Зм.	Лист	№ докум.	Підп.	Дата							
Розроб.	Півень А.С.										
Перевір.	Касьян К.М.										
Т. контр.											
						Лист 1		Листів 1			
Н. контр.	Польська О.В.					НУ «Запорізька політехніка»					
Затвердж.	Кудерметов Р.К.					КНТЗ-510					