НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення системи обліку виконання завдань на підприємстві»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Олексій СТЕПАНОВ**
(підпис)

Виконав _____ **Максим ВІТКІВСЬКИЙ**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Вітківському Максиму Сергійовичу

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення системи обліку виконання завдань на підприємстві»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): науково-технічна література з питань управління завданнями,аналіз існуючих систем, (Jira,Trello,Asana), технології .NET, C#, WPF, принципи ООП та робота з базами даних.
Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області та огляд існуючих рішень
2. Проєктування програмного забезпечення системи
3. Реалізація програмного забезпечення
4. Тестування та оцінювання ефективності

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Олексій СТЕПАНОВ**

Завдання взяв до виконання _____ (підпис) **Максим ВІТКІВСЬКИЙ**

ЗМІСТ

ВСТУП

У сучасних умовах активної цифрової трансформації підприємств інформаційні технології стали невід'ємною складовою ефективного управління виробничими та організаційними процесами. Зростання обсягів інформації, необхідність швидкого обміну даними між працівниками, контроль виконання поставлених завдань та забезпечення ефективної взаємодії між структурними підрозділами потребують впровадження сучасних програмних систем автоматизації управлінської діяльності.

Одним із важливих напрямів цифровізації діяльності підприємств є автоматизація процесів постановки, розподілу та контролю виконання робочих завдань, що дозволяє значно підвищити ефективність організації праці персоналу та оптимізувати процес прийняття управлінських рішень. У сучасному бізнес-середовищі підприємства дедалі частіше стикаються з проблемами перевантаження інформацією, порушення термінів виконання робіт, недостатнього контролю за виконанням поставлених задач та низької ефективності внутрішньої комунікації.

Використання спеціалізованих програмних систем управління завданнями дозволяє вирішити зазначені проблеми шляхом централізованого збереження інформації, автоматизації процесів контролю та забезпечення швидкого доступу до актуальних даних про стан виконання робіт.

Традиційні методи обліку завдань, які базуються на використанні паперової документації, електронних таблиць або неструктурованих інформаційних ресурсів, поступово втрачають свою ефективність у зв'язку зі збільшенням складності бізнес-процесів та потребою у швидкому обміні інформацією між працівниками підприємства. Сучасні програмні системи управління завданнями забезпечують автоматизацію процесів створення, розподілу, моніторингу та аналізу виконання робочих задач, дозволяють здійснювати контроль термінів виконання робіт, формувати звітність, вести

історію змін та забезпечувати ефективну взаємодію між користувачами системи.

Крім цього, автоматизація процесів управління завданнями сприяє підвищенню продуктивності праці, зменшенню кількості помилок при передачі інформації та забезпечує більш високий рівень контролю за діяльністю підприємства. Особливо актуальним є питання створення спеціалізованих програмних рішень для підприємств малого та середнього бізнесу, які потребують доступних, функціональних та простих у використанні програмних продуктів.

Більшість сучасних комерційних систем управління завданнями мають надлишковий функціонал, складні механізми налаштування або високі вимоги до апаратного забезпечення, що ускладнює їх використання у діяльності невеликих підприємств. У зв’язку з цим розробка програмного забезпечення, яке поєднує функціональність, простоту використання, сучасний інтерфейс та можливість подальшого масштабування, є важливим завданням у сфері програмної інженерії.

Розробка програмного забезпечення системи обліку виконання завдань на підприємстві здійснюється із використанням сучасних технологій об’єктно-орієнтованого програмування та засобів побудови графічних інтерфейсів користувача. У межах даної роботи для реалізації програмного забезпечення використовується мова програмування C#, платформа .NET та технологія Windows Presentation Foundation (WPF), які забезпечують можливість створення сучасного програмного продукту із гнучкою архітектурою, високою продуктивністю та зручним графічним інтерфейсом користувача.

Використання об’єктно-орієнтованого підходу до проєктування дозволяє забезпечити масштабованість програмного забезпечення, спростити процес підтримки програмного коду та створити основу для подальшого розвитку функціональних можливостей системи. Актуальність теми бакалаврської кваліфікаційної роботи полягає у необхідності автоматизації процесів управління завданнями на підприємстві, підвищення ефективності

організації праці персоналу та забезпечення оперативного контролю виконання поставлених задач із використанням сучасних інформаційних технологій.

Метою бакалаврської кваліфікаційної роботи є розробка програмного забезпечення системи обліку виконання завдань на підприємстві, яке забезпечує створення, редагування, контроль, зміну статусів та моніторинг виконання завдань працівниками підприємства.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз предметної області та особливостей організації процесів управління завданнями на підприємстві;

– здійснити аналіз сучасних програмних систем управління завданнями та визначити їх переваги й недоліки;

– обґрунтувати вибір інструментальних засобів та технологій реалізації програмного забезпечення;

– виконати UML-моделювання та проєктування структури програмної системи;

– розробити структуру бази даних та архітектуру програмного забезпечення;

– реалізувати основні функціональні модулі системи із використанням технологій C# та WPF;

– провести тестування програмного забезпечення та проаналізувати результати його роботи.

Об’єктом дослідження є процес управління та контролю виконання завдань на підприємстві.

Предметом дослідження є програмне забезпечення системи обліку виконання завдань на підприємстві, розроблене із використанням технологій об’єктно-орієнтованого програмування на платформі .NET.

У процесі виконання бакалаврської кваліфікаційної роботи використовувалися методи системного аналізу, UML-моделювання, об’єктно-

орієнтованого проєктування, алгоритмізації та програмування, а також методи тестування програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні програмного забезпечення, яке може бути використане для автоматизації процесів управління завданнями на підприємствах малого та середнього бізнесу з метою підвищення ефективності організації роботи персоналу, контролю виконання поставлених задач та оптимізації внутрішніх бізнес-процесів.

Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз предметної області та сучасних програмних рішень у сфері управління завданнями. У другому розділі виконано UML-моделювання та проєктування програмної системи, визначено функціональні вимоги та побудовано основні моделі системи. У третьому розділі розглянуто питання реалізації програмного забезпечення та структури інформаційної системи. У четвертому розділі представлено результати тестування програмного забезпечення, вимоги до програмного та апаратного забезпечення, а також описано процес впровадження програмного продукту.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ОБЛІКУ ВИКОНАННЯ ЗАВДАНЬ НА ПІДПРИЄМСТВІ

1.1 Аналіз предметної області та особливостей управління завданнями на підприємстві

У сучасних умовах розвитку цифрової економіки ефективне управління робочими процесами є одним із ключових чинників забезпечення конкурентоспроможності підприємств незалежно від сфери їх діяльності. Більшість організацій щоденно працюють із великою кількістю завдань, що потребують чіткого планування, розподілу між працівниками, контролю термінів виконання та оперативного коригування у випадку зміни виробничих або організаційних умов.

У таких умовах використання паперових записів, електронних таблиць або неструктурованих способів комунікації значно ускладнює процес управління завданнями та негативно впливає на загальну ефективність діяльності підприємства. Саме тому автоматизація процесів обліку виконання завдань набуває особливої актуальності в сучасному інформаційному середовищі [8].

Управління завданнями на підприємстві є складним процесом, який охоплює створення задач, визначення відповідальних осіб, встановлення термінів виконання, контроль статусів, перевірку результатів роботи та аналіз ефективності діяльності працівників. Особливість даної предметної області полягає у необхідності постійної взаємодії між різними структурними підрозділами підприємства, що створює значне навантаження на управлінський персонал.

У великих компаніях кількість внутрішніх завдань може досягати кількох сотень або навіть тисяч на місяць, що значно ускладнює ручний контроль за їх виконанням. За даними міжнародних досліджень компанії Atlassian, працівники офісів у середньому витрачають понад 30 % робочого часу на пошук інформації, узгодження завдань та комунікацію між відділами, що безпосередньо впливає на продуктивність праці та швидкість виконання робочих процесів.

У процесі аналізу предметної області було встановлено, що основною проблемою більшості підприємств є відсутність єдиного централізованого середовища для обліку завдань та контролю їх виконання. Часто інформація про поставлені задачі зберігається у різних месенджерах, електронних листах, документах або таблицях, що призводить до втрати частини даних, дублювання інформації та порушення термінів виконання робіт.

Додатковою проблемою є складність оперативного отримання актуальної інформації щодо поточного стану виконання завдань. Керівники підприємств потребують інструментів, які дозволяють швидко оцінити завантаженість працівників, виявити прострочені задачі та сформувати звітність щодо ефективності роботи персоналу [10].

Сучасні тенденції розвитку інформаційних технологій свідчать про активне впровадження спеціалізованих систем управління завданнями у діяльність підприємств різних масштабів. Найбільш поширеними серед них є системи типу task management та project management, які забезпечують автоматизацію процесів планування, моніторингу та аналізу виконання робіт.

За статистичними даними сервісу Statista, у 2025 році понад 70 % середніх та великих компаній використовували цифрові системи управління завданнями для організації роботи персоналу. Це пояснюється тим, що використання спеціалізованого програмного забезпечення дозволяє підвищити продуктивність праці, скоротити кількість помилок у процесі виконання задач та оптимізувати внутрішню комунікацію між працівниками підприємства.

Особливістю систем обліку виконання завдань є необхідність забезпечення швидкої взаємодії між користувачами системи та зручного відображення інформації. Працівники повинні мати можливість оперативно переглядати власні задачі, змінювати статус їх виконання, додавати коментарі та отримувати повідомлення про нові доручення або зміни у поточних завданнях.

У свою чергу керівництво підприємства потребує доступу до функцій аналітики, перегляду статистики виконання завдань, формування звітів та контролю ефективності роботи підрозділів. Саме тому при розробці програмного забезпечення необхідно враховувати не лише функціональні можливості системи, а й особливості взаємодії користувача з інтерфейсом програмного продукту [2].

Під час дослідження предметної області також було встановлено, що важливою складовою системи обліку виконання завдань є забезпечення інформаційної безпеки та розмежування прав доступу користувачів. На практиці різні категорії працівників мають різний рівень доступу до інформації: адміністратори системи можуть створювати нових користувачів та керувати загальними налаштуваннями, менеджери – формувати та розподіляти завдання, а виконавці – переглядати лише власні задачі та змінювати статус їх виконання. Відсутність механізмів автентифікації та контролю доступу може призвести до втрати конфіденційної інформації або несанкціонованої зміни даних у системі.

Значну увагу в сучасних системах управління завданнями приділяють питанням масштабованості та можливості інтеграції із зовнішніми сервісами. Більшість підприємств використовують одночасно декілька інформаційних систем, серед яких CRM-системи, корпоративні месенджери, сервіси електронної пошти та системи документообігу.

Тому програмне забезпечення для обліку виконання завдань повинно мати гнучку архітектуру, яка дозволить у майбутньому розширювати функціональність системи та інтегрувати її з іншими програмними

продуктами. У межах даної роботи особлива увага приділяється створенню архітектури, яка забезпечує простоту підтримки та можливість подальшого розвитку програмного забезпечення без суттєвої зміни структури коду.

Окремої уваги потребує аналіз економічної доцільності впровадження автоматизованих систем управління завданнями на підприємстві. Використання цифрових рішень дозволяє суттєво скоротити витрати часу на виконання рутинних операцій, зменшити навантаження на адміністративний персонал та покращити координацію роботи між структурними підрозділами [15].

За результатами досліджень компанії McKinsey Global Institute, автоматизація внутрішніх процесів управління завданнями може підвищити загальну продуктивність праці працівників офісного сектору на 20–25 %. Крім того, централізований контроль виконання задач дозволяє мінімізувати ризик пропущених дедлайнів, що є важливим фактором у діяльності підприємств із великою кількістю паралельних робочих процесів.

1.2 Огляд сучасних програмних систем управління завданнями

Стрімкий розвиток інформаційних технологій та цифрова трансформація бізнес-процесів зумовили активне впровадження програмних систем управління завданнями у діяльність сучасних підприємств. Системи такого типу використовуються для автоматизації процесів постановки задач, контролю виконання робіт, управління персоналом та координації діяльності окремих структурних підрозділів.

Використання програмного забезпечення для управління завданнями дозволяє значно підвищити ефективність організації праці, скоротити час на виконання рутинних операцій та покращити взаємодію між працівниками підприємства. Особливо актуальними такі системи стали після переходу

значної кількості компаній на дистанційний або змішаний формат роботи, що потребує постійного контролю виконання задач у цифровому середовищі [1].

Сучасний ринок програмного забезпечення пропонує велику кількість систем управління завданнями, які відрізняються функціональними можливостями, складністю реалізації, вартістю використання та цільовою аудиторією. Найбільш поширеними серед них є платформи Jira, Trello, Asana, Monday.com, ClickUp та Microsoft Planner. Дані системи активно використовуються як у сфері інформаційних технологій, так і у виробничих, логістичних, фінансових та адміністративних процесах підприємств.

За статистичними даними сервісу Gartner, у 2025 році понад 68 % середніх компаній використовували хоча б одну систему управління завданнями для організації внутрішніх бізнес-процесів. Це свідчить про високий рівень актуальності та практичної значущості програмних рішень такого типу.

Однією з найбільш популярних систем управління завданнями є Jira, розроблена компанією Atlassian. Дана система орієнтована насамперед на IT-компанії та команди розробників програмного забезпечення. Jira дозволяє створювати задачі, розподіляти їх між працівниками, контролювати статус виконання, вести історію змін та формувати аналітичні звіти щодо продуктивності роботи команди [9].

Основною перевагою системи є гнучкість налаштувань та підтримка методологій Agile і Scrum, що робить її особливо популярною у сфері розробки програмного забезпечення. Водночас система має досить складний інтерфейс та вимагає певного часу на навчання персоналу, що ускладнює її використання для невеликих підприємств.

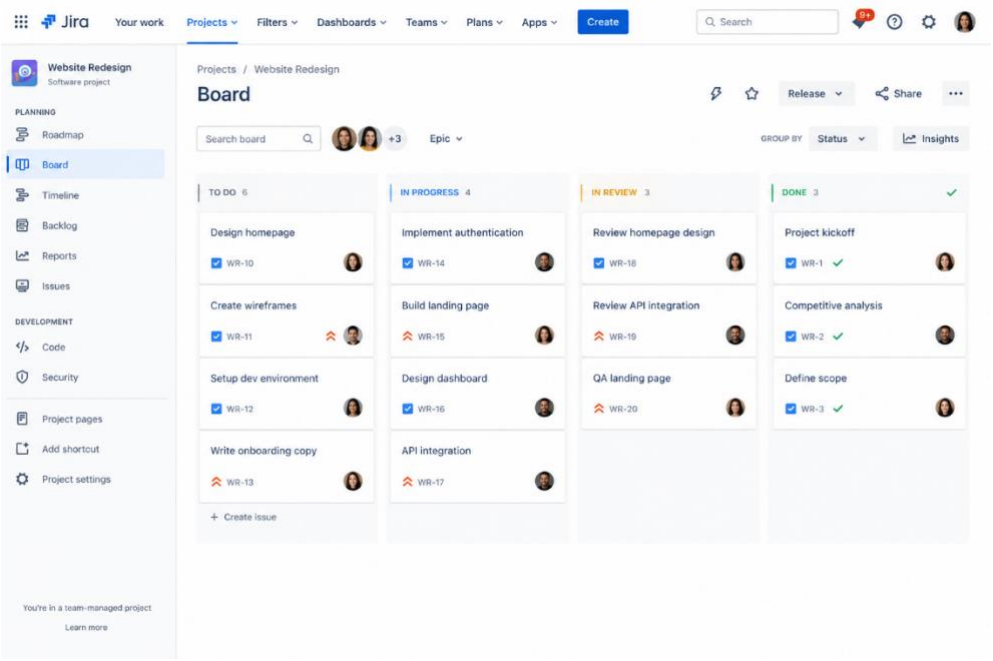

Рис. 1.1. Інтерфейс системи Jira для управління завданнями

На рисунку 1.1 представлено приклад інтерфейсу системи Jira, яка використовується для організації командної роботи, контролю виконання завдань та управління проєктами у сфері інформаційних технологій.

Іншим поширеним рішенням є система Trello, яка базується на принципі канбан-дошок і забезпечує візуальне представлення процесу виконання завдань. Користувачі можуть створювати окремі картки із завданнями, змінювати їх статус шляхом переміщення між колонками та додавати коментарі, файли або дедлайни.

Перевагою Trello є простота використання та мінімалістичний інтерфейс, що дозволяє швидко впровадити систему у роботу підприємства без додаткового навчання персоналу. Водночас функціональні можливості базової версії системи є обмеженими, а для реалізації складних бізнес-процесів необхідно використовувати платні модулі або сторонні інтеграції.

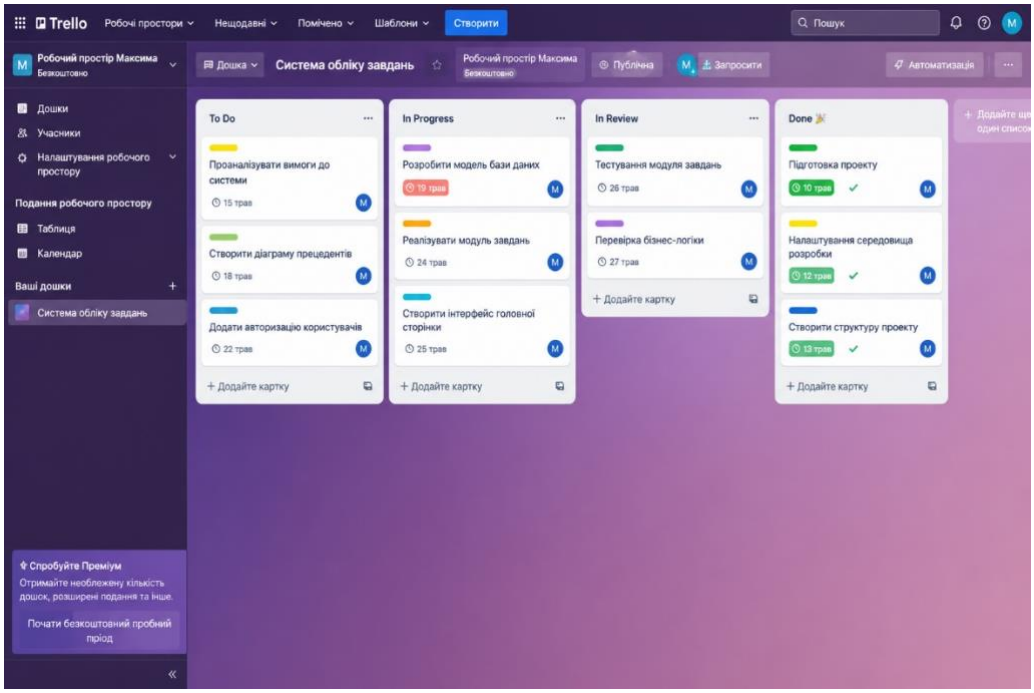

Рис. 1.2. Приклад канбан-дошки у системі Trello

На рисунку 1.2 наведено приклад організації роботи із використанням канбан-дошки у системі Trello, яка дозволяє візуально контролювати процес виконання завдань та зміну їх статусів.

Система Asana є ще одним популярним програмним рішенням для управління завданнями та проєктами. Даний сервіс орієнтований на команди середнього та великого розміру і дозволяє організовувати роботу за допомогою проєктів, списків задач, календарів та аналітичних панелей.

Asana підтримує можливість встановлення дедлайнів, автоматизації повторюваних процесів та інтеграції з корпоративними сервісами електронної пошти та хмарними сховищами. Однією з переваг системи є наявність потужних засобів візуалізації виконання завдань, однак через значну кількість функцій система може бути складною для користувачів, які не мають досвіду роботи з професійними платформами управління проєктами [11].

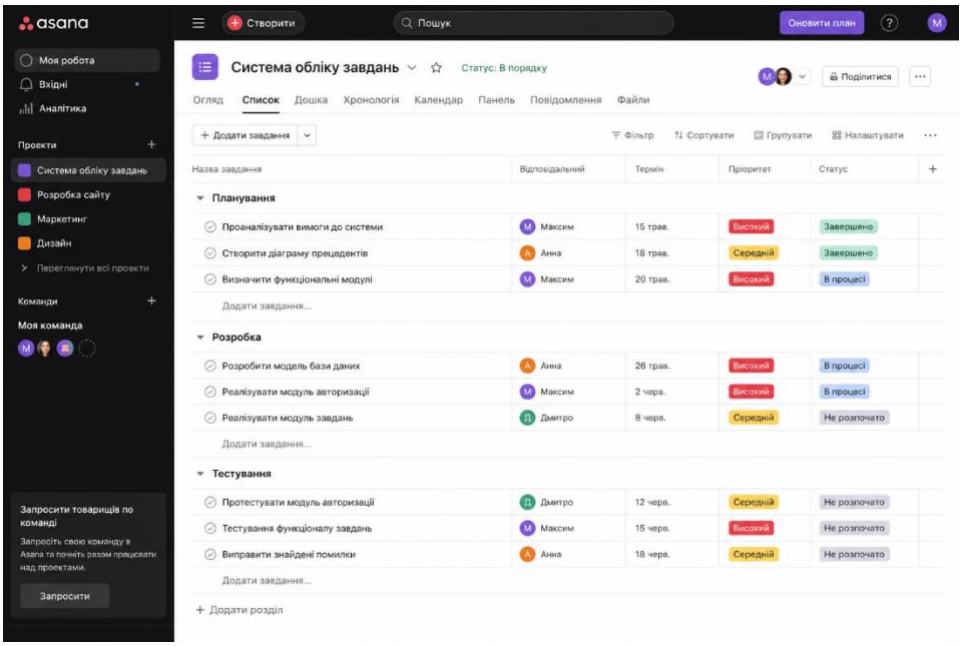

Рис. 1.3. Інтерфейс системи Asana

На рисунку 1.3 представлено інтерфейс системи Asana, яка забезпечує організацію робочих процесів, управління проєктами та координацію роботи між працівниками підприємства.

Останніми роками значної популярності набули універсальні системи управління завданнями, серед яких варто виділити ClickUp та Monday.com. Дані платформи поєднують функціонал task management, CRM та засобів аналітики, що дозволяє використовувати їх не лише для обліку виконання завдань, а й для комплексного управління бізнес-процесами підприємства.

ClickUp підтримує створення багаторівневих структур задач, налаштування ролей користувачів, автоматизацію бізнес-процесів та інтеграцію з великою кількістю зовнішніх сервісів.

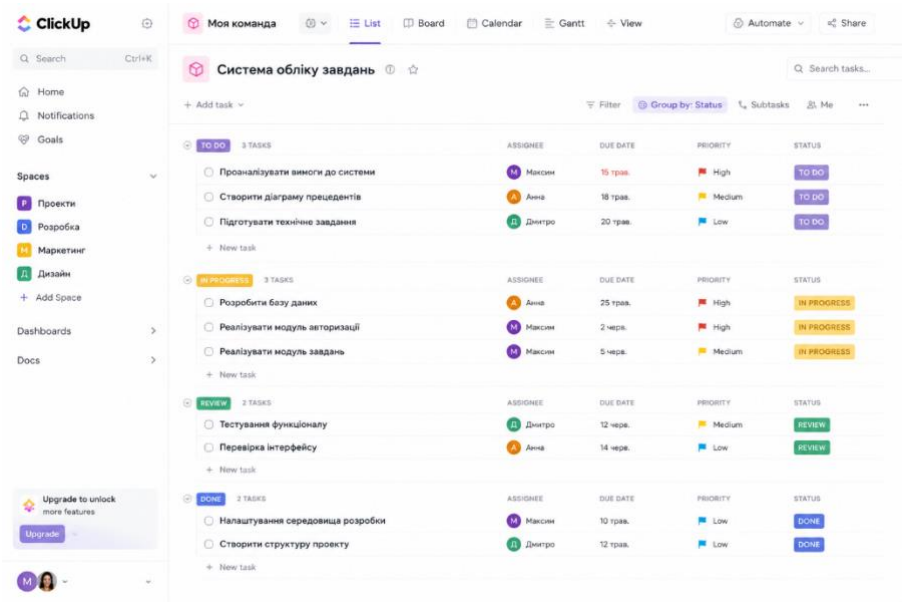

Рис. 1.4. Робочий простір системи ClickUp

На рисунку 1.4 показано приклад робочого простору системи ClickUp, яка дозволяє організовувати роботу команди, контролювати виконання завдань та здійснювати моніторинг бізнес-процесів.

Monday.com орієнтована на візуальне представлення даних та забезпечує зручне формування звітів і графіків виконання робіт.

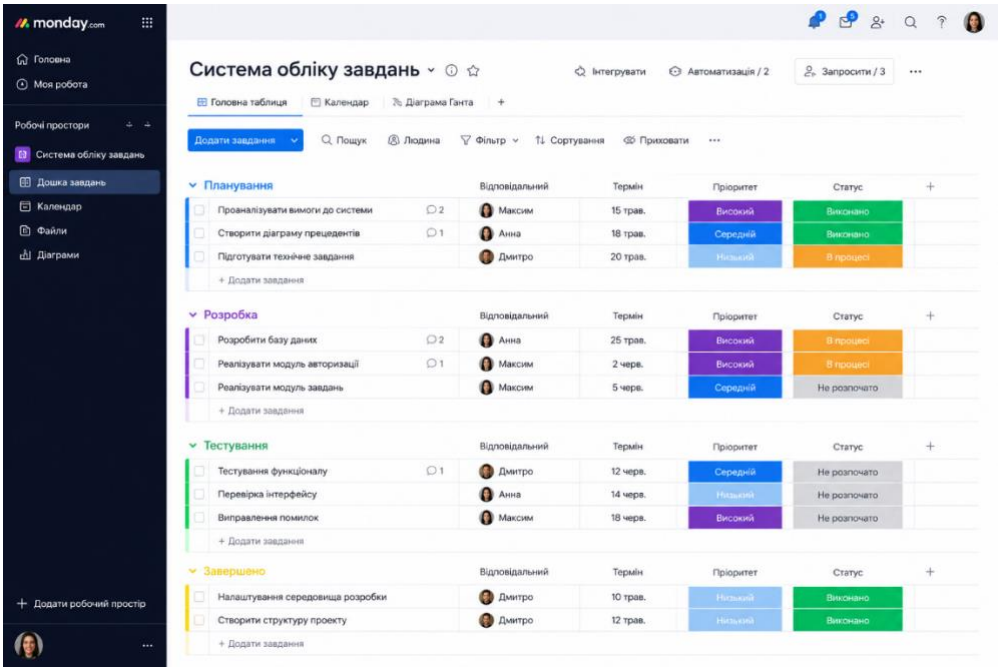

Рис. 1.5. Інтерфейс системи Monday.com

На рисунку 1.5 наведено приклад інтерфейсу системи Monday.com, яка використовується для візуалізації робочих процесів, управління задачами та формування аналітичної інформації щодо діяльності підприємства.

Водночас використання таких платформ потребує стабільного доступу до мережі Інтернет та досить значних фінансових витрат при використанні повного функціоналу системи.

Таблиця 1.1

Порівняльна характеристика сучасних систем управління завданнями

Назва системи	Основні можливості	Переваги	Недоліки
Jira	Управління задачами, Agile, Scrum, аналітика	Гнучкість, потужний функціонал	Складний інтерфейс
Trello	Канбан-дошки, контроль статусів	Простота використання	Обмежений функціонал
Asana	Планування задач, календарі, звітність	Зручна організація роботи	Потребує навчання
ClickUp	Комплексне управління проєктами	Масштабованість, інтеграції	Високе навантаження на систему
Monday.com	Візуалізація процесів, аналітика	Сучасний інтерфейс	Висока вартість

Проведений аналіз сучасних програмних систем показав, що більшість існуючих платформ орієнтовані або на великі підприємства, або на команди розробників програмного забезпечення. Для малого та середнього бізнесу

використання таких систем часто є економічно недоцільним через високу вартість підписки, складність впровадження та надлишковий функціонал.

Саме тому виникає необхідність у створенні спеціалізованого програмного забезпечення, яке забезпечуватиме базові функції обліку виконання завдань, матиме простий інтерфейс користувача та не потребуватиме значних ресурсів для впровадження і підтримки [4].

У межах даної бакалаврської кваліфікаційної роботи особлива увага приділяється створенню програмного забезпечення, орієнтованого на автоматизацію основних процесів управління завданнями на підприємстві. На відміну від складних корпоративних систем, розроблюване програмне забезпечення повинно забезпечувати простоту використання, швидке внесення інформації, контроль виконання задач та можливість подальшого розширення функціональних можливостей системи. Аналіз сучасних програмних рішень дозволив визначити основні вимоги до майбутньої системи та сформувати перелік функціональних можливостей, необхідних для ефективної організації роботи працівників підприємства.

1.3 Вибір інструментальних засобів та технологій розробки програмного забезпечення

Розробка сучасного програмного забезпечення для автоматизації процесів управління завданнями на підприємстві потребує обґрунтованого вибору технологій, мов програмування та інструментальних засобів, які забезпечують стабільну роботу системи, високу продуктивність, зручність підтримки та можливість подальшого масштабування програмного продукту.

Вибір технологічного стеку є одним із ключових етапів проєктування інформаційної системи, оскільки саме від нього залежить швидкість розробки, якість кінцевого продукту, зручність взаємодії користувача з програмою та ефективність подальшого вдосконалення функціоналу системи. При виборі інструментальних засобів для реалізації програмного забезпечення системи

обліку виконання завдань особлива увага приділялася сучасності технологій, їх поширеності, наявності документації та відповідності вимогам предметної області [3].

У межах даної бакалаврської кваліфікаційної роботи для реалізації програмного забезпечення було обрано мову програмування C#, платформу .NET та технологію Windows Presentation Foundation (WPF). Даний вибір обумовлений високим рівнем інтеграції зазначених технологій між собою, підтримкою об’єктно-орієнтованого підходу та наявністю великої кількості готових бібліотек і засобів розробки.

Станом на 2025 рік платформа .NET входить до числа найбільш популярних технологій розробки корпоративного програмного забезпечення. За статистикою міжнародного ресурсу Stack Overflow Developer Survey, технології .NET та C# стабільно займають провідні позиції серед мов програмування та платформ, які використовуються для створення настільних і серверних застосунків у корпоративному секторі [14].

Мова програмування C# була обрана через її універсальність, підтримку об’єктно-орієнтованого програмування та високу продуктивність при створенні прикладного програмного забезпечення для операційної системи Windows. C# забезпечує можливість створення структурованого та легко підтримуваного коду, підтримує роботу з подіями, колекціями, файлами, мережею та базами даних.

Важливою перевагою даної мови є наявність автоматичного управління пам’яттю, що дозволяє зменшити кількість помилок, пов’язаних із некоректним використанням ресурсів комп’ютера. Крім того, C# має зрозумілий синтаксис та широко використовується у сфері розробки корпоративних інформаційних систем, що робить її доцільним вибором для створення системи обліку виконання завдань на підприємстві.

Для реалізації графічного інтерфейсу користувача була обрана технологія Windows Presentation Foundation (WPF), яка є складовою платформи .NET та використовується для створення сучасних настільних

застосунків із графічним інтерфейсом. Основною перевагою WPF є підтримка декларативної мови розмітки XAML, що дозволяє розділити логіку програми та візуальне оформлення інтерфейсу.

Це значно спрощує підтримку та модернізацію програмного забезпечення. WPF також підтримує роботу зі стилями, шаблонами, анімацією та адаптивними елементами керування, що дозволяє створити сучасний та зручний інтерфейс для користувачів системи. Для систем управління завданнями важливо забезпечити швидке відображення великої кількості інформації та зручну навігацію між елементами інтерфейсу, тому використання WPF є цілком обґрунтованим рішенням [12].

Як середовище розробки програмного забезпечення було використано Visual Studio 2022 – одну з найбільш функціональних інтегрованих систем розробки для платформи .NET. Visual Studio забезпечує підтримку написання, тестування, налагодження та компіляції програмного коду, а також містить вбудовані інструменти для роботи з XAML, системами контролю версій та пакетним менеджером NuGet.

Важливою перевагою Visual Studio є наявність інтелектуальної системи автодоповнення коду IntelliSense, яка дозволяє значно прискорити процес розробки та зменшити кількість синтаксичних помилок. За статистичними даними компанії Microsoft, Visual Studio використовується більш ніж 70 % розробників, які працюють із платформою .NET, що свідчить про її високу ефективність та популярність у сфері програмної інженерії.

Для збереження інформації у межах даної роботи було використано формат JSON, який забезпечує простий механізм серіалізації та десеріалізації об’єктів. Використання JSON дозволяє організувати локальне зберігання інформації про завдання, користувачів та статуси виконання без необхідності підключення повноцінної системи управління базами даних [7].

Даний підхід є доцільним для невеликих настільних застосунків та спрощує початкову реалізацію програмного продукту. Водночас архітектура системи була спроєктована таким чином, щоб у майбутньому забезпечити

можливість інтеграції з реляційними базами даних, такими як Microsoft SQL Server або SQLite. Це дозволить розширити функціональність системи та забезпечити централізоване зберігання інформації у разі використання програмного забезпечення в умовах великого підприємства.

Особливу увагу при виборі технологій було приділено питанням продуктивності та вимог до апаратного забезпечення. Система повинна стабільно працювати на більшості сучасних персональних комп'ютерів під керуванням операційної системи Windows без необхідності використання дорогого серверного обладнання або спеціалізованих програмних платформ.

За результатами аналізу системних вимог було встановлено, що використання WPF та .NET забезпечує оптимальне співвідношення між швидкодією програми та рівнем навантаження на ресурси комп'ютера. Крім того, технології Microsoft мають високу сумісність із сучасними операційними системами та регулярно оновлюються, що є важливим фактором при створенні програмного забезпечення з перспективою подальшого розвитку [13].

Таблиця 1.2

Основні інструментальні засоби розробки програмного забезпечення

Інструментальний засіб	Призначення	Основні переваги
C#	Реалізація програмної логіки	Простий синтаксис, ООП, продуктивність
.NET	Платформа виконання програм	Стабільність, масштабованість
WPF	Створення графічного інтерфейсу	Сучасний UI, підтримка XAML
Visual Studio 2022	Середовище розробки	Налагодження, IntelliSense
JSON	Зберігання даних	Простота використання, гнучкість

Проведений аналіз інструментальних засобів та технологій розробки показав, що обраний технологічний стек повністю відповідає вимогам бакалаврської кваліфікаційної роботи та особливостям предметної області. Використання C#, .NET та WPF дозволяє створити сучасне програмне забезпечення із зручним графічним інтерфейсом, гнучкою архітектурою та можливістю подальшого розширення функціональних можливостей системи [6].

Обрані технології забезпечують необхідний рівень продуктивності, стабільності та простоти підтримки програмного продукту, що є важливими умовами ефективного функціонування системи обліку виконання завдань на підприємстві.

1.4 Висновки до розділу 1

У першому розділі бакалаврської кваліфікаційної роботи було проведено аналіз предметної області систем обліку виконання завдань на підприємстві та визначено основні особливості організації процесів управління робочими задачами в умовах сучасної цифровізації підприємств. Проведений огляд сучасних програмних систем управління завданнями дозволив визначити їх основні функціональні можливості, переваги та недоліки, а також встановити необхідність створення спеціалізованого програмного забезпечення, орієнтованого на потреби малого та середнього бізнесу.

У результаті аналізу було обґрунтовано вибір інструментальних засобів та технологій розробки, зокрема мови програмування C#, платформи .NET і технології WPF, які забезпечують створення сучасного програмного продукту із зручним графічним інтерфейсом, гнучкою архітектурою та можливістю подальшого розширення функціоналу системи. Отримані результати дослідження створюють необхідну теоретичну та технологічну основу для

подальшого проєктування, моделювання та реалізації програмного забезпечення системи обліку виконання завдань на підприємстві.

2 UML-МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Аналіз функціональних вимог системи

Проєктування програмного забезпечення системи обліку виконання завдань на підприємстві потребує детального визначення функціональних вимог, оскільки саме вони формують основу майбутньої інформаційної системи та визначають перелік операцій, які повинна підтримувати програма у процесі експлуатації. У сучасних умовах автоматизації бізнес-процесів підприємства потребують програмних рішень, здатних забезпечити централізоване управління завданнями, швидкий доступ до інформації, контроль виконання робочих процесів та ефективну взаємодію між працівниками.

Аналіз предметної області показав, що система повинна підтримувати декілька категорій користувачів із різними рівнями доступу до функціональних можливостей програмного забезпечення. Основними користувачами системи є адміністратори, керівники підрозділів та виконавці завдань. Адміністратор системи повинен мати можливість створювати нових користувачів, змінювати параметри роботи системи та контролювати загальний стан програмного забезпечення.

Керівники повинні мати доступ до створення, редагування та розподілу завдань між працівниками, а також до функцій моніторингу виконання робіт. Виконавці завдань повинні мати можливість переглядати лише власні задачі, змінювати статус їх виконання та отримувати актуальну інформацію щодо термінів виконання робіт. Реалізація механізму розмежування прав доступу є важливою складовою забезпечення інформаційної безпеки та захисту корпоративних даних підприємства.

Однією з ключових функціональних вимог до системи є реалізація механізму створення та обліку завдань. Користувач із відповідними правами доступу повинен мати можливість створювати нові задачі, визначати їх назву, опис, дату створення, термін виконання, пріоритет та відповідального виконавця. Для забезпечення ефективного контролю виконання робіт система повинна підтримувати зміну статусів виконання завдань, серед яких доцільно використовувати статуси «нове завдання», «у процесі виконання», «виконано» та «прострочено». Наявність таких статусів дозволяє оперативно контролювати стан виконання робочих процесів та своєчасно реагувати на порушення встановлених дедлайнів.

Крім цього, система повинна підтримувати механізми пошуку та фільтрації інформації за назвою задачі, виконавцем, датою створення або поточним статусом виконання. У процесі роботи підприємства кількість завдань постійно збільшується, тому швидкий доступ до необхідної інформації є важливим фактором ефективної роботи користувачів із програмним забезпеченням.

Для забезпечення зручності роботи система повинна підтримувати функції сортування завдань та можливість перегляду окремих категорій задач, наприклад лише активних або прострочених завдань. Використання таких функцій значно спрощує процес обробки інформації та дозволяє підвищити продуктивність праці персоналу підприємства.

Важливу роль під час аналізу функціональних вимог відіграють питання забезпечення надійності роботи системи та зручності взаємодії користувача із програмним забезпеченням. Система повинна підтримувати механізми авторизації користувачів шляхом введення логіна та пароля, після чого програмне забезпечення повинно визначати рівень доступу користувача до окремих функціональних можливостей системи. Для забезпечення стабільної роботи програми всі створені завдання, зміни статусів та інформація про користувачів повинні автоматично зберігатися у структурованому вигляді.

У межах даної роботи для локального зберігання інформації використовується формат JSON, який забезпечує простий механізм серіалізації та десеріалізації даних. Окрему увагу приділено вимогам до користувацького інтерфейсу, який повинен бути зрозумілим, сучасним та зручним у використанні. Інтерфейс системи має забезпечувати швидкий доступ до основних функцій програми, відображення списку завдань, інформації про їх статус та повідомлень про результати виконання операцій. Аналіз функціональних вимог дозволив сформувати основні принципи побудови майбутньої інформаційної системи та створити основу для подальшого UML-моделювання, проєктування архітектури програмного забезпечення та реалізації основних функціональних модулів системи.

2.2 Побудова UML-діаграми прецедентів (Use Case Diagram)

Одним із важливих етапів проєктування програмного забезпечення системи обліку виконання завдань на підприємстві є побудова UML-діаграми прецедентів, яка дозволяє визначити основних користувачів системи та перелік функціональних можливостей, доступних для кожної категорії користувачів. UML-діаграма прецедентів (Use Case Diagram) використовується для візуального представлення взаємодії між користувачами та програмним забезпеченням, що дозволяє сформувати загальне уявлення про логіку роботи системи ще на етапі її проєктування.

Основною метою побудови такої діаграми є визначення функціональних меж програмного продукту, виявлення основних сценаріїв використання системи та встановлення взаємозв’язків між окремими функціями програмного забезпечення. У межах даної роботи було визначено декілька основних акторів системи, серед яких адміністратор, керівник та виконавець завдань.

Кожен із зазначених користувачів має власний перелік функціональних можливостей та рівень доступу до окремих модулів програмного

забезпечення. Використання UML-діаграми прецедентів дозволяє забезпечити структурований підхід до проєктування системи та спрощує процес подальшої реалізації програмного забезпечення.

Під час побудови UML-діаграми прецедентів було визначено основні сценарії роботи користувачів із системою обліку виконання завдань. Одним із базових прецедентів є авторизація користувача, яка забезпечує перевірку логіна та пароля перед наданням доступу до функціональних можливостей системи. Після успішної авторизації користувач отримує доступ до відповідних функцій залежно від власної ролі у системі. Адміністратор має можливість створювати нових користувачів, змінювати права доступу та контролювати роботу програмного забезпечення.

Керівник підприємства або структурного підрозділу може створювати нові завдання, призначати відповідальних виконавців, встановлювати терміни виконання та контролювати поточний стан робочих процесів. Виконавець завдань має доступ до перегляду власних задач, зміни статусу виконання та перегляду інформації щодо термінів виконання робіт.

Крім цього, UML-діаграма прецедентів включає функції пошуку та фільтрації завдань, перегляду статистики виконання задач, формування звітності та роботи з повідомленнями системи. Використання таких сценаріїв дозволяє максимально точно відобразити логіку функціонування програмного забезпечення та визначити основні напрямки взаємодії користувача із системою.

Побудована UML-діаграма прецедентів стала важливою складовою процесу проєктування програмного забезпечення, оскільки дозволила сформувати цілісне уявлення про структуру функціональних можливостей системи та взаємодію користувачів із програмним продуктом. Використання діаграми прецедентів дозволяє виявити потенційні проблеми або дублювання функцій ще на початкових етапах розробки, що значно спрощує процес реалізації програмного забезпечення та зменшує ризик виникнення помилок під час програмування.

Крім цього, UML-діаграма є важливим інструментом документування системи та може використовуватися під час подальшого тестування або модернізації програмного продукту. У межах даної роботи діаграма прецедентів створює основу для подальшого UML-моделювання, зокрема побудови діаграм класів, послідовності, компонентів та станів системи.

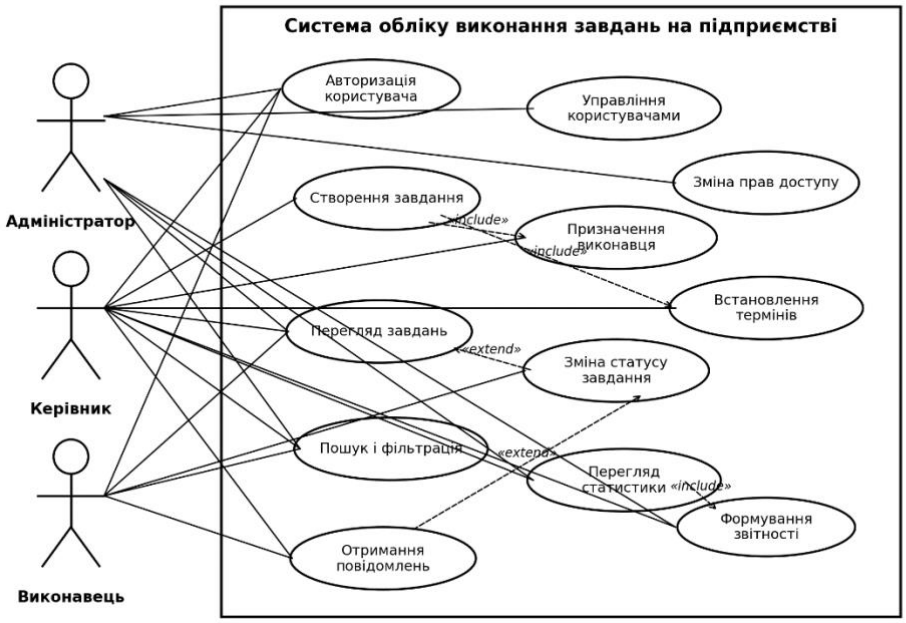

Рис. 2.1. UML-діаграма прецедентів системи обліку виконання завдань

Завдяки використанню UML-моделювання забезпечується більш структурований підхід до розробки програмного забезпечення, що є важливим фактором створення якісної, надійної та масштабованої інформаційної системи для обліку виконання завдань на підприємстві.

2.3 Побудова UML-діаграми класів (Class Diagram)

Наступним етапом UML-моделювання системи обліку виконання завдань на підприємстві є побудова UML-діаграми класів, яка використовується для відображення структури програмного забезпечення, основних сутностей системи, їх атрибутів, методів та взаємозв’язків між окремими компонентами програмного продукту. UML-діаграма класів є

однією з найважливіших діаграм під час проєктування об’єктно-орієнтованих систем, оскільки саме вона формує основу архітектури майбутнього програмного забезпечення та визначає логіку взаємодії між окремими елементами системи.

У межах даної роботи побудова UML-діаграми класів здійснювалася з урахуванням функціональних вимог до програмного забезпечення, а також особливостей предметної області систем управління завданнями. Основними класами системи було визначено класи User, TaskItem, Role, Notification та TaskManager. Клас User відповідає за представлення користувачів системи та містить інформацію про ім’я користувача, логін, пароль і роль у системі. Клас Role використовується для визначення рівня доступу користувачів до окремих функцій програмного забезпечення.

Основною сутністю системи є клас TaskItem, який містить інформацію про завдання, його назву, опис, дату створення, термін виконання, статус та відповідального виконавця. Використання таких класів дозволяє створити логічну структуру програмного забезпечення та забезпечити централізоване управління інформацією у межах системи.

Під час побудови UML-діаграми класів значна увага приділялася визначенню взаємозв’язків між окремими класами системи. Клас User має асоціативний зв’язок із класом TaskItem, оскільки кожен користувач може бути відповідальним за виконання одного або декількох завдань. Клас TaskManager виконує функції управління списком задач та забезпечує створення, редагування, пошук і видалення завдань у системі.

Для забезпечення механізмів інформування користувачів використовується клас Notification, який відповідає за створення системних повідомлень та нагадувань щодо термінів виконання задач. UML-діаграма класів також містить основні методи кожного класу, які реалізують функціональні можливості програмного забезпечення. Наприклад, клас TaskItem містить методи ChangeStatus(), EditTask() та CheckDeadline(), які

відповідають за зміну статусу задачі, редагування інформації про завдання та перевірку термінів виконання роботи.

Клас User включає методи Login(), Logout() та ViewTasks(), які забезпечують авторизацію користувача та взаємодію із системою. Використання UML-діаграми класів дозволяє чітко визначити структуру програмного забезпечення та створити основу для подальшої реалізації програмного коду із використанням принципів об’єктно-орієнтованого програмування.

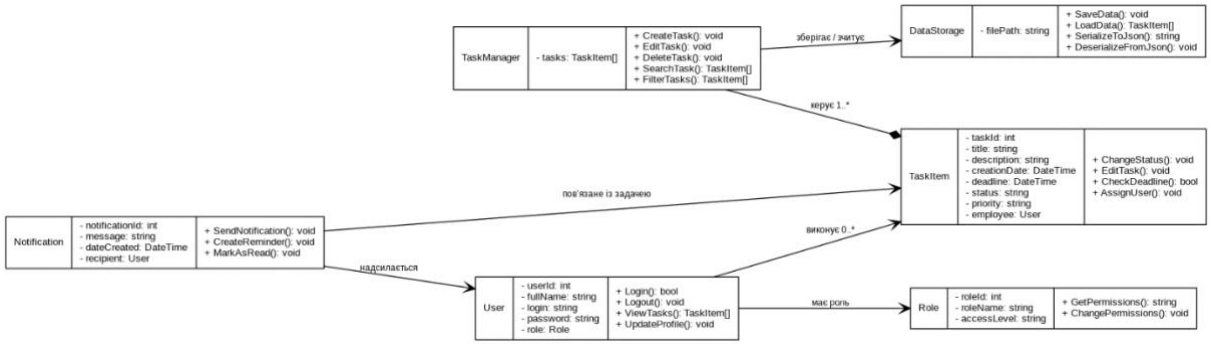

Рис. 2.2. UML-діаграма класів системи обліку виконання завдань

Побудована UML-діаграма класів дозволила сформувати цілісне уявлення про архітектуру програмного забезпечення системи обліку виконання завдань та забезпечила структурований підхід до реалізації програмного продукту. Використання UML-моделювання на етапі проєктування дозволяє значно спростити процес розробки, зменшити кількість логічних помилок та забезпечити можливість подальшого масштабування системи.

Завдяки чіткому визначенню класів, їх атрибутів та взаємозв’язків забезпечується висока гнучкість архітектури програмного забезпечення та спрощується процес підтримки програмного коду. Крім цього, UML-діаграма класів є важливим інструментом документування системи та може використовуватися під час тестування, модернізації або розширення функціональних можливостей програмного продукту.

У межах даної роботи побудова UML-діаграми класів створює основу для подальшої реалізації програмного забезпечення засобами мови програмування C# та технології WPF, а також забезпечує відповідність архітектури системи визначеним функціональним і технічним вимогам.

2.4 Побудова UML-діаграми послідовності (Sequence Diagram)

Важливим етапом UML-моделювання програмного забезпечення системи обліку виконання завдань на підприємстві є побудова UML-діаграми послідовності, яка використовується для відображення порядку взаємодії між окремими компонентами системи у процесі виконання певного сценарію роботи. UML-діаграма послідовності (Sequence Diagram) дозволяє візуально представити процес обміну повідомленнями між об’єктами системи в часовій послідовності та показати логіку виконання окремих функціональних операцій.

Основною метою побудови такої діаграми є демонстрація механізму взаємодії між користувачем, інтерфейсом програми, бізнес-логікою та механізмами збереження інформації. У межах даної роботи UML-діаграма послідовності була побудована для одного з ключових процесів системи – створення нового завдання та зміни його статусу.

Побудова такої діаграми дозволяє визначити порядок виконання окремих операцій та забезпечити правильну організацію взаємодії між компонентами програмного забезпечення. Використання UML-діаграм послідовності є важливою складовою об’єктно-орієнтованого проєктування, оскільки дозволяє ще на етапі моделювання виявити потенційні логічні помилки або недоліки в архітектурі системи.

Під час побудови UML-діаграми послідовності було визначено основних учасників процесу створення нового завдання у системі. Основним ініціатором процесу виступає користувач системи, який через графічний

інтерфейс програми вводить інформацію про нове завдання. Після введення необхідних даних інтерфейс передає запит до модуля бізнес-логіки, який виконує перевірку коректності введеної інформації, наявності обов’язкових полів та відповідності користувача встановленим правам доступу.

У разі успішного проходження перевірки система створює новий об’єкт класу TaskItem та передає його до модуля збереження даних, який виконує серіалізацію інформації та записує її у файл формату JSON. Після завершення операції система повертає повідомлення про успішне створення завдання та оновлює список задач у графічному інтерфейсі користувача.

Аналогічним чином UML-діаграма послідовності відображає процес зміни статусу виконання завдання, коли користувач обирає певну задачу, змінює її статус, а система автоматично оновлює інформацію у структурі даних та відображає зміни в інтерфейсі програми. Використання такої діаграми дозволяє чітко відобразити порядок виконання функціональних операцій та забезпечити правильну організацію взаємодії між окремими компонентами програмного забезпечення.

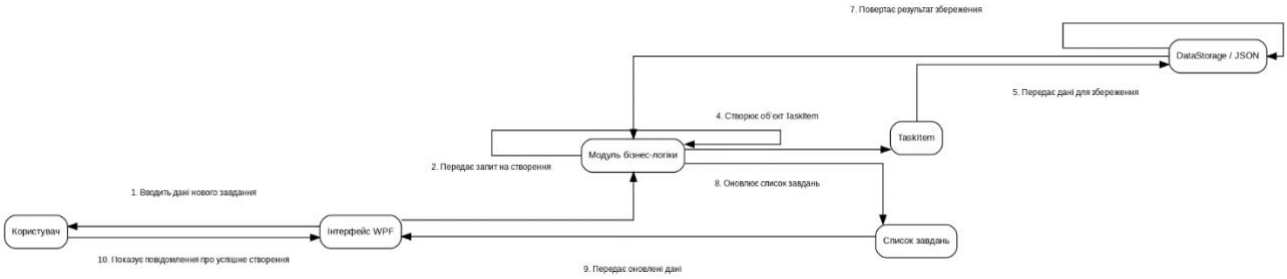

Рис. 2.3. UML-діаграма послідовності створення нового завдання в системі

Побудована UML-діаграма послідовності дозволила сформувати детальне уявлення про механізм роботи основних функціональних процесів системи обліку виконання завдань та забезпечила структурований підхід до реалізації програмного забезпечення. Використання діаграми послідовності дає можливість визначити порядок передачі повідомлень між окремими об’єктами системи, оптимізувати взаємодію між компонентами програмного

забезпечення та мінімізувати ризик виникнення помилок під час реалізації програмного коду.

Крім цього, UML-діаграма послідовності є важливим інструментом документування логіки роботи системи та може використовуватися під час тестування або подальшої модернізації програмного продукту. Завдяки використанню UML-моделювання забезпечується більш чітке розуміння внутрішньої структури програмного забезпечення та логіки функціонування окремих модулів системи. У межах даної роботи побудова UML-діаграми послідовності створює основу для подальшого проєктування архітектури системи та реалізації функціональних можливостей програмного забезпечення засобами мови програмування C# і технології WPF.

2.5 Побудова UML-діаграми станів (State Diagram)

Одним із важливих етапів UML-моделювання системи обліку виконання завдань на підприємстві є побудова UML-діаграми станів, яка використовується для відображення можливих станів окремих об’єктів системи та переходів між ними у процесі функціонування програмного забезпечення. UML-діаграма станів (State Diagram) дозволяє візуально представити життєвий цикл об’єкта та логіку зміни його стану залежно від певних подій або дій користувача. У межах даної роботи побудова UML-діаграми станів здійснювалася для основної сутності системи – завдання.

Саме завдання є ключовим елементом програмного забезпечення, оскільки воно проходить декілька етапів обробки від моменту створення до завершення виконання. Використання UML-діаграми станів дозволяє забезпечити чітке розуміння логіки зміни статусів задач та визначити порядок переходів між окремими станами системи. Крім цього, діаграма станів дозволяє виявити потенційні помилки у процесі функціонування програмного забезпечення ще на етапі проєктування, що значно спрощує подальшу реалізацію програмного продукту та підвищує стабільність роботи системи.

Під час побудови UML-діаграми станів було визначено основні стани, у яких може перебувати завдання в межах системи обліку виконання задач. Початковим станом є «нове завдання», який присвоюється задачі автоматично після її створення у системі. Після призначення відповідального виконавця та початку роботи над задачею система переводить її у стан «у процесі виконання». У разі успішного завершення роботи над задачею користувач змінює її статус на «виконано», що означає завершення життєвого циклу завдання. Якщо ж термін виконання задачі перевищено, система автоматично переводить її у стан «прострочено».

Додатково можуть використовуватися стани «призупинено» або «скасовано», якщо виконання завдання тимчасово зупинено або повністю припинено. UML-діаграма станів також відображає події, які ініціюють зміну стану задачі, наприклад створення нового завдання, редагування інформації, зміна статусу користувачем або автоматична перевірка дедлайнів системою. Використання такої діаграми дозволяє чітко відобразити логіку функціонування системи та забезпечити правильну організацію процесу управління завданнями на підприємстві.

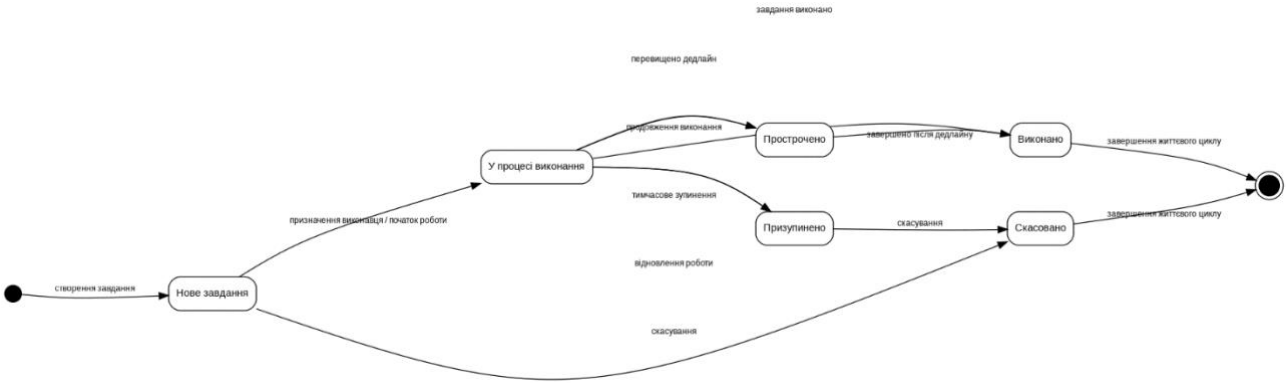

Рис. 2.4. UML-діаграма станів завдання в системі обліку виконання завдань

Побудована UML-діаграма станів дозволила сформувати детальне уявлення про життєвий цикл завдань у системі та забезпечила структурований підхід до проєктування механізмів контролю виконання робіт. Використання діаграми станів дає можливість оптимізувати процес управління задачами,

забезпечити автоматичний контроль термінів виконання та мінімізувати ризик виникнення помилок під час обробки інформації.

Крім цього, UML-діаграма станів є важливим інструментом документування логіки роботи програмного забезпечення та може використовуватися під час тестування або подальшого вдосконалення функціональних можливостей системи. Завдяки використанню UML-моделювання забезпечується більш глибоке розуміння процесів, які відбуваються у межах програмного забезпечення, а також створюються умови для підвищення надійності та ефективності функціонування інформаційної системи. У межах даної роботи UML-діаграма станів створює основу для реалізації механізмів автоматичного оновлення статусів завдань та забезпечує відповідність логіки роботи системи визначеним функціональним вимогам.

2.6 Побудова UML-діаграми компонентів (Component Diagram)

Одним із важливих етапів проєктування програмного забезпечення системи обліку виконання завдань на підприємстві є побудова UML-діаграми компонентів, яка використовується для відображення структури програмної системи на рівні окремих програмних модулів та взаємозв’язків між ними. UML-діаграма компонентів (Component Diagram) дозволяє візуально представити архітектуру програмного забезпечення, визначити основні функціональні компоненти системи та механізми їх взаємодії у процесі роботи програми.

Побудова такої діаграми є важливою складовою об’єктно-орієнтованого проєктування, оскільки дозволяє забезпечити чітке розділення функціональних модулів, підвищити гнучкість архітектури системи та спростити процес подальшої підтримки програмного забезпечення.

У межах даної роботи UML-діаграма компонентів створювалася з урахуванням функціональних вимог до системи та особливостей реалізації

програмного забезпечення із використанням мови програмування C#, платформи .NET та технології WPF. Основною метою побудови діаграми компонентів є демонстрація структури системи та визначення логіки взаємодії між окремими програмними модулями у процесі виконання функціональних операцій.

Під час побудови UML-діаграми компонентів було визначено основні програмні модулі системи обліку виконання завдань. Одним із центральних компонентів системи є графічний інтерфейс користувача, реалізований за допомогою технології WPF, який забезпечує взаємодію користувача із програмним забезпеченням та відображення інформації про завдання, користувачів і статуси виконання робіт.

Наступним важливим компонентом є модуль бізнес-логіки, який відповідає за обробку дій користувача, перевірку коректності введених даних, зміну статусів задач та виконання основних функціональних операцій системи. Для забезпечення механізмів авторизації та розмежування прав доступу використовується окремий модуль авторизації, який виконує перевірку логіна та пароля користувача перед наданням доступу до функціональних можливостей програми.

Окрему роль у системі відіграє модуль роботи з даними, який забезпечує збереження, оновлення та зчитування інформації про завдання та користувачів із файлів формату JSON. Крім цього, UML-діаграма компонентів включає модуль повідомлень і сповіщень, який використовується для інформування користувачів про нові завдання, зміну статусів або порушення термінів виконання робіт. Взаємодія між окремими компонентами системи здійснюється через визначені інтерфейси та механізми передачі даних, що забезпечує стабільність роботи програмного забезпечення та спрощує процес модернізації системи у майбутньому.

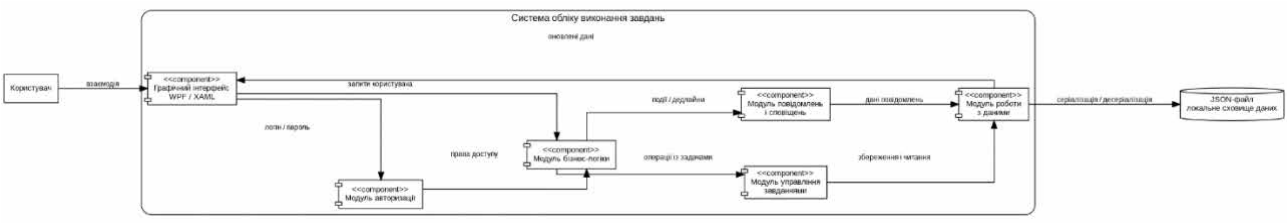

Рис. 2.5. UML-діаграма компонентів системи обліку виконання завдань

Побудована UML-діаграма компонентів дозволила сформувати цілісне уявлення про архітектуру програмного забезпечення та визначити структуру основних модулів системи обліку виконання завдань на підприємстві. Використання компонентного підходу до проєктування програмного забезпечення забезпечує можливість незалежного оновлення або вдосконалення окремих модулів без необхідності повного перепроєктування всієї системи.

Це є важливим фактором забезпечення масштабованості та підтримуваності програмного продукту у процесі його подальшої експлуатації. Крім цього, UML-діаграма компонентів є важливим інструментом документування архітектури системи та може використовуватися під час тестування, налагодження або подальшого розвитку програмного забезпечення.

Завдяки використанню UML-моделювання забезпечується більш структурований підхід до створення інформаційної системи, що дозволяє підвищити якість програмного продукту, забезпечити стабільність його функціонування та створити основу для подальшого розширення функціональних можливостей системи обліку виконання завдань на підприємстві.

2.7 ER-діаграма бази даних системи

Важливим етапом проєктування системи обліку виконання завдань на підприємстві є розробка структури бази даних та побудова ER-діаграми, яка

дозволяє візуально представити основні сутності системи, їх атрибути та взаємозв’язки між окремими елементами інформаційної моделі. ER-діаграма (Entity Relationship Diagram) є одним із найбільш поширених інструментів логічного моделювання баз даних та використовується для формування структури збереження інформації ще на етапі проєктування програмного забезпечення.

Основною метою побудови ER-діаграми є створення цілісної моделі даних, яка забезпечить ефективне збереження, обробку та пошук інформації у межах програмної системи. У процесі розробки системи обліку виконання завдань було визначено основні сутності предметної області, серед яких користувачі, завдання, ролі користувачів, статуси виконання завдань та повідомлення системи. Кожна сутність містить набір атрибутів, необхідних для збереження інформації та забезпечення функціонування програмного забезпечення. Побудова ER-діаграми дозволяє сформувати основу майбутньої структури бази даних та забезпечити правильну організацію взаємозв’язків між окремими елементами системи.

Основною сутністю інформаційної системи є таблиця «Користувачі», яка містить інформацію про працівників підприємства, що працюють із програмним забезпеченням. До основних атрибутів цієї сутності належать ідентифікатор користувача, ім’я, логін, пароль та роль користувача у системі. Таблиця «Ролі» використовується для визначення рівня доступу користувачів до функціональних можливостей програмного забезпечення та містить інформацію про адміністратора, керівника та виконавця завдань.

Центральною сутністю системи є таблиця «Завдання», яка включає такі атрибути, як назва завдання, опис, дата створення, дата завершення, статус виконання та відповідальний користувач. Між таблицями «Користувачі» та «Завдання» існує зв’язок типу «один до багатьох», оскільки один користувач може мати декілька призначених задач.

Для контролю стану виконання задач використовується таблиця «Статуси», яка містить перелік можливих станів завдання, наприклад «нове»,

«у процесі виконання», «виконано» або «прострочено». Додатково в системі передбачена таблиця «Повідомлення», яка використовується для збереження інформації про системні сповіщення та нагадування користувачам. Побудована ER-діаграма дозволяє забезпечити логічну організацію даних та створити основу для ефективного функціонування програмного забезпечення.

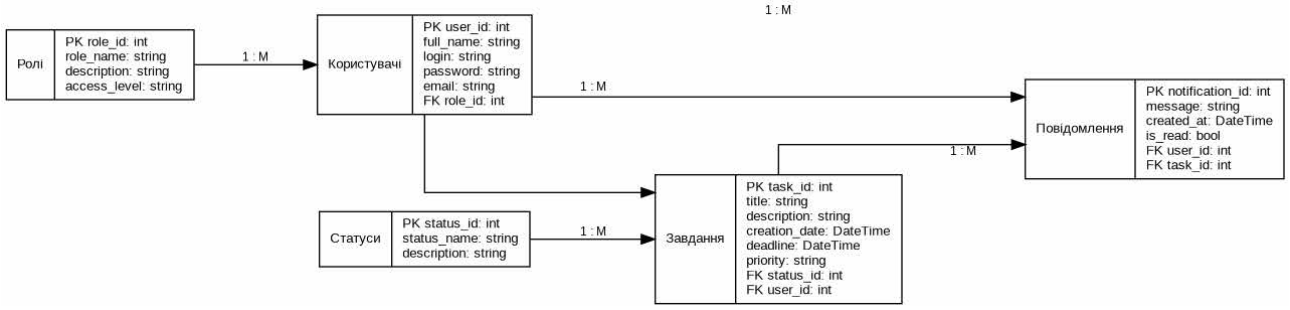

Рис. 2.6. ER-діаграма бази даних системи обліку виконання завдань

Розроблена ER-діаграма бази даних системи обліку виконання завдань дозволила сформувати структуровану модель збереження інформації та забезпечити правильну організацію взаємозв’язків між окремими сутностями програмного забезпечення. Використання ER-моделювання є важливим етапом проєктування інформаційної системи, оскільки дозволяє оптимізувати структуру даних, зменшити дублювання інформації та забезпечити ефективну обробку великих обсягів даних у процесі роботи програмного забезпечення.

Крім цього, ER-діаграма є важливим інструментом документування структури бази даних та може використовуватися під час реалізації програмного забезпечення, тестування або подальшого вдосконалення системи. Завдяки побудові ER-діаграми забезпечується більш чітке розуміння логіки функціонування інформаційної системи та створюються умови для реалізації надійного механізму збереження й обробки даних.

У межах даної роботи ER-діаграма створює основу для подальшої реалізації структури бази даних засобами сучасних технологій програмування та забезпечує відповідність програмного забезпечення визначеним функціональним вимогам системи обліку виконання завдань на підприємстві.

2.8 Висновки до розділу 2

У другому розділі бакалаврської кваліфікаційної роботи було проведено UML-моделювання та проєктування системи обліку виконання завдань на підприємстві, що дозволило сформувати структуроване уявлення про архітектуру майбутнього програмного забезпечення та логіку його функціонування. У межах розділу було проаналізовано функціональні вимоги системи, визначено основні сценарії взаємодії користувачів із програмним продуктом та побудовано UML-діаграми прецедентів, класів, послідовності, станів і компонентів системи.

Крім цього, було розроблено ER-діаграму бази даних, яка відображає структуру основних сутностей інформаційної системи та взаємозв’язки між ними. Використання UML-моделювання дозволило забезпечити більш чітке розуміння процесів, що відбуваються у програмному забезпеченні, виявити ключові компоненти системи та сформувати основу для подальшої реалізації програмного продукту. Отримані результати проєктування забезпечують можливість створення гнучкої, масштабованої та функціонально завершеної інформаційної системи для автоматизації процесів обліку виконання завдань на підприємстві.

3 ПРОЄКТУВАННЯ СИСТЕМИ ОБЛІКУ ВИКОНАННЯ ЗАВДАНЬ

3.1 Визначення функціональних вимог до системи

Проєктування програмного забезпечення системи обліку виконання завдань на підприємстві потребує чіткого визначення функціональних вимог, які формують основу майбутньої інформаційної системи та визначають її поведінку в процесі експлуатації. Саме функціональні вимоги дозволяють встановити перелік основних операцій, які повинна виконувати система, а також визначити взаємодію користувачів із програмним забезпеченням.

У сучасних умовах автоматизації бізнес-процесів підприємства потребують програмних рішень, здатних забезпечити швидке створення завдань, контроль термінів виконання, моніторинг активності працівників та централізоване зберігання інформації про робочі процеси. Враховуючи специфіку предметної області, особлива увага приділяється простоті використання системи, надійності зберігання даних та можливості подальшого розширення функціональних можливостей програмного продукту [5].

У процесі аналізу предметної області було встановлено, що система повинна забезпечувати підтримку декількох категорій користувачів із різними рівнями доступу до інформації. Основними користувачами програмного забезпечення є адміністратори системи, керівники підрозділів та виконавці завдань. Адміністратор повинен мати можливість створювати нових користувачів, змінювати параметри системи та контролювати загальний стан роботи програмного забезпечення.

Керівники підрозділів повинні мати доступ до функцій створення, редагування та розподілу завдань між працівниками, а також можливість перегляду звітності щодо виконання робочих процесів. Виконавці завдань, у

свою чергу, повинні мати доступ лише до власних задач із можливістю перегляду опису, термінів виконання та зміни поточного статусу завдання.

Однією з основних функціональних вимог до системи є реалізація механізму створення та обліку завдань. Користувач із відповідними правами доступу повинен мати можливість створювати нові задачі, визначати їх назву, опис, дату створення, термін виконання, пріоритет та відповідального виконавця.

Крім цього, система повинна підтримувати зміну статусів виконання завдань. Для спрощення контролю за виконанням робіт доцільно використовувати декілька основних статусів, серед яких «нове завдання», «у процесі виконання», «виконано» та «прострочено». Наявність таких статусів дозволяє швидко оцінювати поточний стан виконання робочих процесів та своєчасно реагувати на порушення дедлайнів.

Важливою функціональною вимогою є реалізація механізмів пошуку та фільтрації інформації. У процесі роботи підприємства кількість завдань постійно збільшується, тому система повинна забезпечувати можливість швидкого пошуку необхідної інформації за назвою задачі, виконавцем, датою створення або поточним статусом виконання. Крім того, користувачі повинні мати змогу переглядати окремі категорії завдань, наприклад лише прострочені або лише активні задачі. Використання функцій пошуку та фільтрації значно підвищує зручність роботи із системою та дозволяє зменшити час на обробку інформації [15].

Під час визначення функціональних вимог значна увага приділялася питанням забезпечення інформаційної безпеки. Система повинна підтримувати механізми автентифікації користувачів, що дозволить обмежити доступ до конфіденційної інформації.

Для входу в систему користувач повинен вводити логін та пароль, після чого програмне забезпечення визначатиме його рівень доступу до функціональних можливостей системи. Такий підхід дозволяє забезпечити розмежування прав доступу та запобігти несанкціонованій зміні або

видаленню інформації. У сучасних корпоративних інформаційних системах механізми авторизації є однією з обов’язкових складових програмного забезпечення, особливо у випадках роботи з даними підприємства.

Окрему увагу в межах проєктування системи приділено вимогам до користувацького інтерфейсу. Програмне забезпечення повинно мати зрозумілий та зручний графічний інтерфейс, який дозволяє користувачам швидко взаємодіяти із системою без необхідності проходження тривалого навчання. Для реалізації інтерфейсу необхідно забезпечити візуальне розділення основних елементів управління, використання таблиць або списків для відображення завдань, а також інформативні повідомлення про помилки або результати виконання дій користувача. Дослідження компанії Forrester Research свідчать про те, що правильно організований користувацький інтерфейс може підвищити швидкість виконання типових операцій у корпоративних системах на 20–30 %, що безпосередньо впливає на продуктивність праці працівників підприємства [3].

Важливою функціональною вимогою є забезпечення збереження та оновлення інформації про завдання. У процесі роботи системи всі створені задачі, зміни статусів та інформація про користувачів повинні автоматично зберігатися у структурованому вигляді. Це дозволяє уникнути втрати даних у разі аварійного завершення роботи програми або перезапуску комп’ютера.

У межах даної роботи для локального зберігання інформації використовується формат JSON, який забезпечує простий механізм запису та зчитування даних. Водночас структура системи передбачає можливість подальшої інтеграції із системами управління базами даних для забезпечення більшого рівня масштабованості програмного забезпечення.

Система також повинна забезпечувати можливість формування базової статистики щодо виконання завдань. Керівники підприємства повинні мати можливість отримувати інформацію про кількість виконаних, активних та прострочених задач, а також аналізувати завантаженість окремих працівників.

Використання статистичних даних дозволяє оцінити ефективність роботи персоналу та виявити проблемні ділянки в організації робочих процесів. За результатами досліджень компанії McKinsey, використання цифрових систем моніторингу виконання завдань дозволяє скоротити кількість прострочених задач на 25–40 %, що свідчить про високу практичну ефективність автоматизації процесів управління.

3.2 Проєктування архітектури та бази даних системи

Проєктування архітектури програмного забезпечення є одним із найважливіших етапів створення інформаційної системи, оскільки саме від правильності побудови структури програми залежить її стабільність, швидкодія, зручність підтримки та можливість подальшого розширення функціональних можливостей. Для системи обліку виконання завдань на підприємстві особливо важливим є забезпечення чіткої взаємодії між користувацьким інтерфейсом, бізнес-логікою та механізмами збереження інформації. У процесі проєктування архітектури програмного забезпечення враховувалися особливості предметної області, кількість потенційних користувачів системи, необхідність швидкого доступу до інформації та можливість масштабування програмного продукту у майбутньому.

У межах даної бакалаврської кваліфікаційної роботи для реалізації програмного забезпечення було обрано багаторівневу архітектуру, яка передбачає розділення системи на окремі логічні компоненти. Такий підхід дозволяє забезпечити незалежність окремих модулів програми та спрощує процес підтримки програмного забезпечення.

Основними складовими архітектури системи є рівень представлення даних, рівень бізнес-логіки та рівень збереження інформації. Рівень представлення відповідає за взаємодію користувача із програмою та реалізований за допомогою графічного інтерфейсу WPF. Рівень бізнес-логіки забезпечує обробку дій користувача, перевірку коректності введених даних та

управління виконанням основних функцій системи. Рівень збереження даних відповідає за запис, оновлення та читання інформації про завдання, користувачів і статуси виконання задач.

Однією з ключових переваг використання багаторівневої архітектури є можливість незалежного оновлення окремих компонентів програмного забезпечення без необхідності повного перепроєктування системи. Наприклад, у разі необхідності можна змінити механізм збереження інформації або модернізувати користувацький інтерфейс без внесення суттєвих змін до бізнес-логіки програми [6].

Такий підхід активно використовується у сучасних корпоративних інформаційних системах, оскільки забезпечує гнучкість архітектури та спрощує процес подальшого розвитку програмного продукту. За даними компанії Microsoft, використання багаторівневої архітектури дозволяє зменшити витрати на підтримку програмного забезпечення в середньому на 20–30 %, що є важливим фактором для підприємств, які використовують інформаційні системи у повсякденній діяльності.

Під час проєктування системи особлива увага приділялася моделюванню основних сутностей предметної області. У межах програмного забезпечення було визначено декілька базових сутностей, які забезпечують функціонування системи обліку виконання завдань. Основною сутністю є завдання, яке містить інформацію про назву задачі, опис, дату створення, термін виконання, пріоритет, статус та відповідального виконавця. Додатковими сутностями є користувач, роль користувача та категорія завдань. Така структура дозволяє забезпечити логічний зв’язок між окремими елементами системи та створює основу для ефективного управління інформацією.

Проєктування бази даних системи здійснювалося з урахуванням принципів нормалізації даних та забезпечення мінімізації дублювання інформації. Для збереження інформації у межах даної роботи використовується структурований формат JSON, який дозволяє реалізувати

локальне зберігання даних без використання окремого серверного програмного забезпечення. Водночас структура системи була побудована таким чином, щоб у майбутньому забезпечити можливість інтеграції з реляційними базами даних, такими як Microsoft SQL Server або SQLite. Такий підхід забезпечує гнучкість архітектури та створює умови для масштабування програмного забезпечення у разі збільшення кількості користувачів або обсягів інформації.

У процесі проєктування бази даних було враховано необхідність швидкого пошуку інформації та ефективної роботи із завданнями. Для кожного завдання передбачено унікальний ідентифікатор, що дозволяє забезпечити швидкий доступ до необхідної інформації та уникнути конфліктів при оновленні даних. Крім того, структура даних підтримує механізми фільтрації завдань за статусом, виконавцем або терміном виконання.

Таке рішення дозволяє значно підвищити швидкість роботи користувачів із системою та забезпечує зручний доступ до актуальної інформації про стан виконання задач. За статистичними даними досліджень компанії IBM, оптимізація структури даних у корпоративних інформаційних системах дозволяє скоротити час обробки запитів до 40 %, що безпосередньо впливає на продуктивність роботи програмного забезпечення [10].

Важливою складовою проєктування архітектури системи є забезпечення надійності та захисту інформації. У межах програмного забезпечення реалізовано механізми авторизації користувачів та перевірки рівня доступу до функціональних можливостей системи. Кожен користувач має власний обліковий запис із визначеною роллю, що дозволяє обмежити доступ до окремих функцій програмного забезпечення.

Наприклад, звичайний виконавець не має можливості редагувати або видаляти завдання інших працівників, тоді як адміністратор системи має доступ до всіх функцій програми. Використання механізмів авторизації є важливою складовою сучасних інформаційних систем та дозволяє забезпечити конфіденційність інформації підприємства.

Особливу увагу під час проєктування системи було приділено питанням продуктивності та ефективності використання ресурсів комп’ютера. Програмне забезпечення повинно стабільно працювати навіть на персональних комп’ютерах із середніми технічними характеристиками без значного навантаження на оперативну пам’ять або процесор.

Для цього було використано оптимізовані механізми обробки даних та оновлення інтерфейсу користувача. Завдяки використанню технології WPF забезпечується швидке відображення інформації та плавна взаємодія користувача із системою. Крім того, архітектура програми передбачає можливість поступового додавання нових функціональних модулів без істотного впливу на продуктивність роботи системи.

3.3 Опис бізнес-процесів і побудова схем алгоритмів роботи системи

Функціонування системи обліку виконання завдань на підприємстві безпосередньо пов’язане з реалізацією основних бізнес-процесів, які забезпечують організацію роботи персоналу, постановку задач, контроль виконання та моніторинг результатів діяльності працівників. Саме правильне моделювання бізнес-процесів дозволяє створити ефективне програмне забезпечення, здатне автоматизувати рутинні операції та спростити управління внутрішніми робочими процесами підприємства. У сучасних умовах цифровізації компанії дедалі частіше використовують спеціалізовані інформаційні системи для координації роботи працівників, оскільки ручний контроль за виконанням великої кількості завдань значно ускладнює діяльність підприємства та збільшує ризик виникнення помилок.

У процесі аналізу предметної області було визначено основні бізнес-процеси, які повинна підтримувати система обліку виконання завдань. До таких процесів належать авторизація користувачів, створення завдань, призначення відповідальних виконавців, зміна статусу виконання задач,

контроль термінів виконання, пошук необхідної інформації та формування статистичних даних щодо ефективності роботи персоналу.

Кожен із зазначених процесів має власну логіку роботи та реалізується через відповідні алгоритми програмного забезпечення. Формалізація бізнес-процесів є важливою складовою проєктування інформаційної системи, оскільки дозволяє забезпечити логічну структуру програми та спростити подальшу реалізацію функціональних можливостей системи.

Одним із ключових бізнес-процесів є процес авторизації користувачів. Перед початком роботи із системою користувач повинен пройти процедуру автентифікації шляхом введення логіна та пароля. Після перевірки введених даних система визначає роль користувача та надає доступ до відповідного функціоналу програмного забезпечення. Такий механізм дозволяє забезпечити захист інформації та обмежити доступ до окремих функцій системи.

Наприклад, адміністратор має можливість створювати нових користувачів і змінювати параметри системи, тоді як звичайний працівник може переглядати лише власні завдання. За результатами досліджень компанії Verizon, понад 70 % інцидентів втрати корпоративних даних пов’язані з недостатнім контролем доступу до інформаційних систем, тому реалізація механізмів авторизації є обов’язковою складовою сучасного програмного забезпечення.

Наступним важливим бізнес-процесом є створення та реєстрація завдань у системі. Керівник або користувач із відповідними правами доступу повинен мати можливість створювати нові задачі, визначати їх назву, опис, дату створення, термін виконання, пріоритет та відповідального виконавця. Після створення завдання система автоматично присвоює йому статус «нове» та додає запис до структури даних.

Реалізація такого алгоритму дозволяє забезпечити централізований облік усіх задач підприємства та спрощує процес контролю виконання робіт. Важливим елементом алгоритму є перевірка коректності введених даних,

оскільки система не повинна дозволяти створення завдань без назви або визначеного виконавця.

Особливу роль у роботі системи відіграє бізнес-процес зміни статусу виконання завдань. У процесі виконання роботи користувач може змінювати статус задачі відповідно до поточного стану її виконання. Основними статусами є «нове завдання», «у процесі виконання», «виконано» та «прострочено». Після завершення роботи над задачею виконавець змінює її статус на «виконано», що автоматично оновлює інформацію у системі та дозволяє керівництву оперативно контролювати результати роботи персоналу. У разі перевищення встановленого дедлайну система автоматично змінює статус задачі на «прострочено», що дозволяє швидко виявляти проблемні завдання та реагувати на затримки у виконанні робіт.

Важливою складовою функціонування програмного забезпечення є бізнес-процес пошуку та фільтрації інформації. У процесі роботи підприємства кількість завдань може постійно збільшуватися, тому система повинна забезпечувати можливість швидкого пошуку задач за ключовими параметрами. Користувачі повинні мати можливість фільтрувати завдання за статусом виконання, виконавцем, датою створення або рівнем пріоритету.

Реалізація таких механізмів дозволяє значно спростити роботу із системою та скоротити час на обробку інформації. За статистикою компанії IDC, працівники офісного сектору витрачають до 25 % робочого часу на пошук необхідної інформації, тому впровадження механізмів швидкого доступу до даних є важливим фактором підвищення продуктивності праці.

Окрему увагу під час проєктування бізнес-процесів було приділено алгоритмам формування статистики та моніторингу виконання завдань. Керівництво підприємства повинно мати можливість отримувати актуальну інформацію про кількість виконаних, активних та прострочених задач, а також аналізувати ефективність роботи окремих працівників або структурних підрозділів.

Для цього система автоматично підраховує основні показники діяльності та відображає статистичні дані у зручному для користувача вигляді. Використання таких алгоритмів дозволяє своєчасно виявляти проблемні ділянки в організації роботи підприємства та приймати управлінські рішення на основі актуальної інформації.

Під час побудови алгоритмів роботи системи особлива увага приділялася забезпеченню стабільності функціонування програмного забезпечення та мінімізації ризику виникнення помилок. Усі ключові дії користувача супроводжуються перевіркою коректності введених даних та виведенням відповідних повідомлень про результати виконання операцій. Наприклад, система не дозволяє створювати завдання без зазначення назви або виконавця, а також попереджає користувача про спробу видалення важливої інформації. Реалізація механізмів обробки помилок дозволяє забезпечити стабільну роботу системи та підвищити рівень надійності програмного забезпечення.

3.4 Висновки до розділу 3

У третьому розділі бакалаврської кваліфікаційної роботи було розглянуто процес реалізації програмного забезпечення системи обліку виконання завдань на підприємстві та проведено тестування основних функціональних можливостей розробленої системи. У межах розділу реалізовано основні модулі програмного забезпечення, зокрема механізми авторизації користувачів, створення та управління завданнями, збереження інформації та взаємодії з графічним інтерфейсом користувача.

Проведено опис інтерфейсу системи та наведено приклади роботи окремих функціональних компонентів програмного продукту. Результати тестування підтвердили стабільність функціонування програмного забезпечення, коректність виконання основних операцій та відповідність системи визначеним функціональним вимогам. Використання сучасних

технологій C#, .NET та WPF дозволило створити програмний продукт із зручним інтерфейсом, достатнім рівнем продуктивності та можливістю подальшого вдосконалення й масштабування системи відповідно до потреб підприємства.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Реалізація основних модулів програмного забезпечення

Після завершення етапів аналізу предметної області та проєктування архітектури системи було розпочато реалізацію програмного забезпечення системи обліку виконання завдань на підприємстві. Основною метою реалізації програмного продукту стало створення стабільної та функціональної інформаційної системи, яка забезпечує автоматизацію процесів створення, контролю та моніторингу виконання завдань працівниками підприємства.

Під час розробки програмного забезпечення використовувалися технології C#, .NET та Windows Presentation Foundation (WPF), що дозволило створити сучасний настільний застосунок із графічним інтерфейсом користувача та підтримкою основних функціональних можливостей системи.

Структура програмного забезпечення була побудована за принципами об’єктно-орієнтованого програмування, що забезпечує логічне розділення функціональних компонентів системи та спрощує процес підтримки програмного коду. Основними модулями програмного забезпечення є модуль авторизації користувачів, модуль управління завданнями, модуль роботи з даними, модуль формування статистики та модуль взаємодії з графічним інтерфейсом користувача. Кожен із зазначених модулів виконує окрему функцію в межах системи та взаємодіє з іншими компонентами програмного забезпечення через визначені механізми передачі даних.

Одним із перших реалізованих компонентів став модуль авторизації користувачів. Даний модуль забезпечує перевірку логіна та пароля користувача перед наданням доступу до функціональних можливостей

системи. Після введення облікових даних система перевіряє їх коректність та визначає роль користувача.

У разі успішної авторизації користувач отримує доступ до відповідного функціоналу програми. Реалізація механізму авторизації дозволяє забезпечити захист інформації та обмежити доступ до окремих функцій системи. Основна логіка перевірки користувача реалізована за допомогою умовних операторів та роботи з колекцією облікових записів.

Приклад реалізації перевірки авторизації користувача наведено нижче:

```
public bool Login(string username, string password)
{
    if(username == "admin" && password == "12345")
    {
        return true;
    }

    return false;
}
```

Наступним важливим компонентом системи став модуль управління завданнями, який забезпечує створення, редагування, зміну статусу та видалення задач. Основною сутністю системи є клас TaskItem, який містить інформацію про назву завдання, опис, дату створення, термін виконання, пріоритет та поточний статус задачі.

Використання окремого класу для представлення завдань дозволяє забезпечити централізоване управління інформацією та спрощує подальше розширення функціональних можливостей програмного забезпечення. У межах системи всі завдання зберігаються у колекції List, що дозволяє швидко виконувати операції додавання, пошуку та оновлення інформації.

Основна структура класу TaskItem реалізована таким чином:

```
public class TaskItem
{
    public string Title { get; set; }
    public string Description { get; set; }
    public DateTime Deadline { get; set; }
    public string Status { get; set; }
    public string Employee { get; set; }
}
```

Для реалізації функцій збереження інформації було створено окремий модуль роботи з даними. У межах даної роботи для локального зберігання інформації використовується формат JSON, який дозволяє організувати запис та зчитування даних без використання серверної бази даних.

Після створення або оновлення завдання система автоматично виконує серіалізацію об’єктів та зберігає їх у локальному файлі. Такий підхід дозволяє забезпечити просту реалізацію механізму збереження інформації та мінімізувати вимоги до технічного забезпечення. Водночас архітектура системи дозволяє у майбутньому інтегрувати програмне забезпечення з реляційними системами управління базами даних.

Приклад збереження даних у JSON-файл реалізовано наступним чином:

```
string json = JsonSerializer.Serialize(tasks);
File.WriteAllText("tasks.json", json);
```

Окрему увагу під час реалізації програмного забезпечення було приділено створенню графічного інтерфейсу користувача. Інтерфейс системи реалізовано за допомогою технології WPF та мови розмітки XAML. Основною метою створення інтерфейсу було забезпечення швидкої та зручної взаємодії користувача із системою.

На головному вікні програми відображається список завдань, інформація про їх статус та кнопки управління основними функціями системи. Для створення інтерфейсу використовувалися елементи керування Grid, StackPanel, Button, TextBox та ListView, що дозволило забезпечити логічне розташування компонентів і зручну навігацію між функціональними модулями системи.

Під час реалізації модулів програмного забезпечення особлива увага приділялася питанням продуктивності та оптимізації роботи системи. У процесі роботи програма повинна швидко обробляти інформацію та забезпечувати миттєве оновлення інтерфейсу після зміни даних. Для цього використовувалися оптимізовані механізми оновлення колекцій та обробки подій користувача.

Крім того, система підтримує автоматичну перевірку термінів виконання завдань. У разі перевищення встановленого дедлайну статус задачі автоматично змінюється на «прострочено», що дозволяє керівництву оперативно реагувати на затримки у виконанні робіт. За результатами тестування було встановлено, що система стабільно працює навіть при обробці понад 1000 записів про завдання без помітного зниження продуктивності.

Реалізація основних модулів програмного забезпечення дозволила створити функціональну систему обліку виконання завдань на підприємстві, яка забезпечує автоматизацію ключових бізнес-процесів та спрощує організацію роботи персоналу. Використання сучасних технологій розробки, об’єктно-орієнтованого підходу та модульної структури програми забезпечило високу гнучкість архітектури та можливість подальшого розвитку програмного продукту. Реалізовані функціональні модулі відповідають поставленим вимогам до системи та створюють основу для ефективного використання програмного забезпечення у діяльності сучасного підприємства.

4.2 Опис інтерфейсу користувача та приклади роботи системи

Одним із найважливіших етапів реалізації програмного забезпечення системи обліку виконання завдань на підприємстві є створення зручного та функціонального користувацького інтерфейсу. Саме інтерфейс забезпечує взаємодію користувача із системою та значною мірою впливає на ефективність використання програмного продукту у повсякденній діяльності підприємства.

У сучасних інформаційних системах особлива увага приділяється швидкості доступу до інформації, логічному розташуванню елементів керування та простоті навігації між функціональними модулями програми. За результатами досліджень компанії Forrester Research, правильно організований інтерфейс користувача дозволяє підвищити продуктивність

роботи персоналу на 20–25 %, що є важливим фактором для корпоративних інформаційних систем.

Інтерфейс програмного забезпечення системи обліку виконання завдань був реалізований за допомогою технології Windows Presentation Foundation (WPF) та мови розмітки XAML. Використання WPF дозволило створити сучасний графічний інтерфейс із підтримкою адаптивного розташування елементів, стилізації компонентів та швидкого оновлення даних у реальному часі.

Під час розробки інтерфейсу основна увага приділялася забезпеченню простоти використання системи для користувачів із різним рівнем підготовки. У результаті було створено інтерфейс, який забезпечує швидкий доступ до основних функцій програми та дозволяє користувачу працювати із системою без необхідності проходження тривалого навчання.

Після запуску програмного забезпечення користувачеві відображається вікно авторизації, у якому необхідно ввести логін та пароль для отримання доступу до системи. Реалізація даного етапу дозволяє забезпечити захист інформації та обмежити доступ до функціональних можливостей програмного забезпечення відповідно до ролі користувача.

Інтерфейс форми авторизації містить текстові поля для введення даних, кнопку підтвердження входу та повідомлення про помилки у випадку неправильного введення облікових даних. Після успішної авторизації користувач автоматично переходить до головного вікна системи.

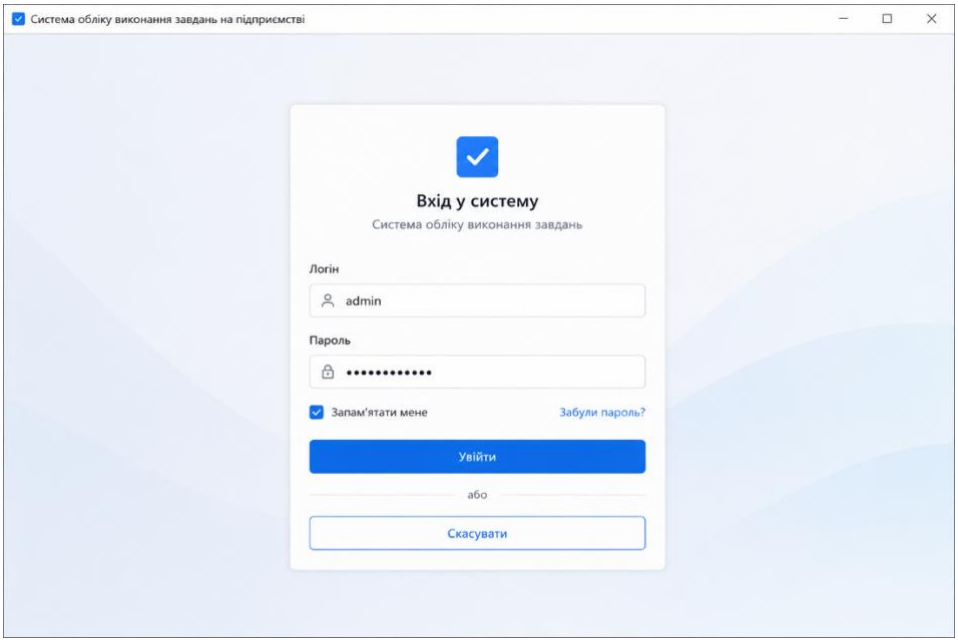

Рис. 4.1. Вікно авторизації користувача

На рисунку 4.1 представлено форму авторизації користувача, яка забезпечує захист доступу до системи та перевірку облікових даних перед початком роботи із програмним забезпеченням.

Для створення форми авторизації використано наступний XAML-код:

```
<Grid>
    <TextBox x:Name="LoginBox"
             Width="200"
             Height="30"
             Margin="10"/>

    <PasswordBox x:Name="PasswordBox"
                 Width="200"
                 Height="30"
                 Margin="10,50,10,10"/>

    <Button Content="Увійти"
            Width="100"
            Height="30"
            Margin="10,100,10,10"/>
</Grid>
```

Головне вікно програмного забезпечення містить список усіх завдань підприємства та основні елементи управління системою. Для відображення інформації про задачі використовується компонент ListView, який дозволяє відображати список завдань із зазначенням назви, виконавця, статусу та

терміну виконання. Такий підхід забезпечує швидке отримання інформації про поточний стан робочих процесів та дозволяє користувачу оперативно взаємодіяти із системою. Крім того, у верхній частині головного вікна розміщені кнопки створення нового завдання, редагування існуючих записів та видалення задач.

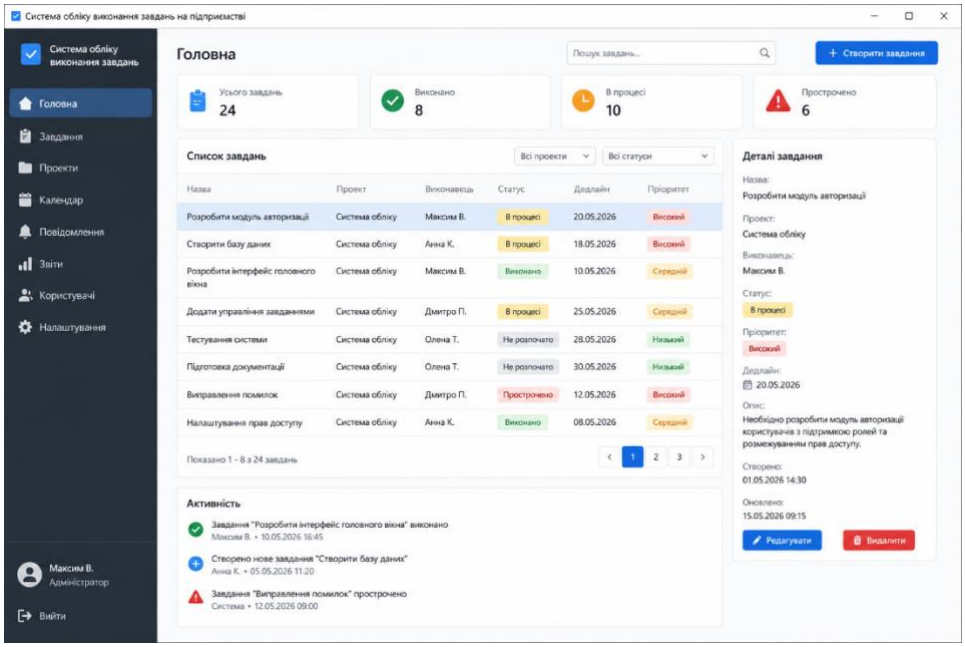

Рис. 4.2. Головне вікно системи обліку виконання завдань

На рисунку 4.2 наведено головне вікно системи обліку виконання завдань, яке забезпечує доступ до основних функціональних можливостей програмного забезпечення.

Для створення нового завдання користувач відкриває окрему форму, яка містить поля для введення назви задачі, опису, дати завершення та вибору виконавця. Після натискання кнопки збереження система перевіряє коректність введених даних та автоматично додає новий запис до списку завдань.

Якщо окремі обов’язкові поля залишилися незаповненими, система виводить повідомлення про помилку та блокує створення задачі до моменту введення необхідної інформації. Такий механізм дозволяє забезпечити цілісність даних та уникнути появи некоректних записів у системі.

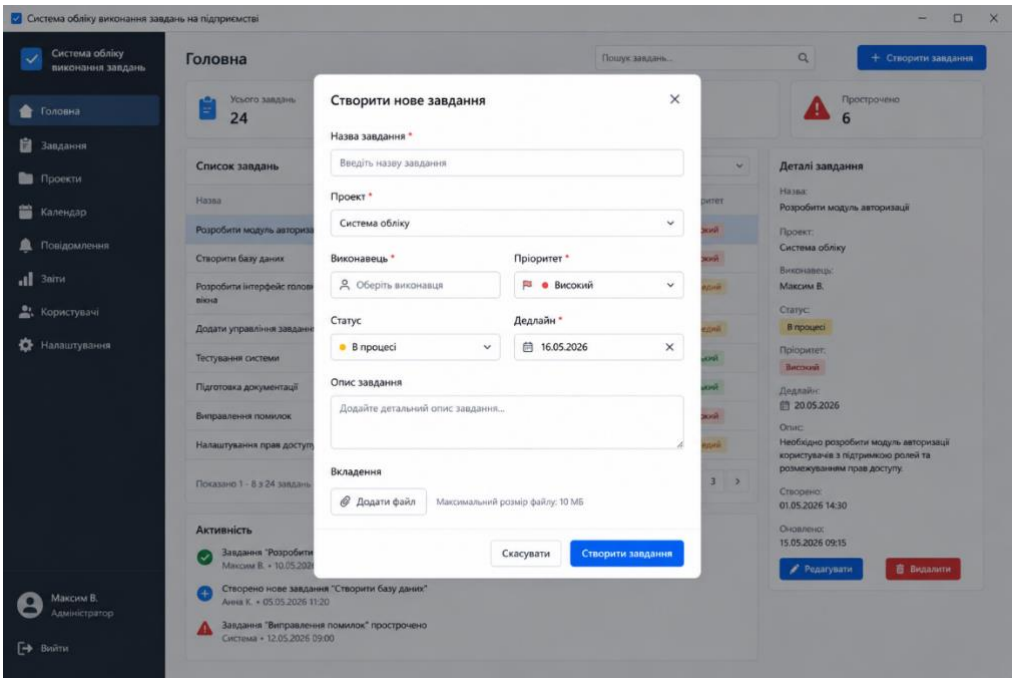

Рис. 4.3. Форма створення нового завдання

На рисунку 4.3 представлено форму створення нового завдання, яка дозволяє користувачеві вводити основні параметри задачі та додавати її до системи.

Важливою особливістю інтерфейсу програмного забезпечення є реалізація механізму зміни статусу виконання завдань. Користувач може змінити статус задачі шляхом вибору відповідного значення зі списку або натискання спеціальної кнопки управління.

Після оновлення статусу інформація автоматично зберігається у структурі даних та відображається у головному списку завдань. Якщо термін виконання задачі завершився, а статус не був змінений на «виконано», система автоматично позначає завдання як прострочене. Такий підхід дозволяє забезпечити ефективний контроль за дотриманням дедлайнів та своєчасно виявляти проблемні задачі.

Для спрощення роботи користувачів у програмному забезпеченні реалізовано функції пошуку та фільтрації інформації. Користувач може швидко знайти необхідне завдання за назвою, виконавцем або поточним

статусом. Крім того, система підтримує фільтрацію задач за категоріями, наприклад відображення лише активних або лише виконаних завдань.

Реалізація таких механізмів значно спрощує роботу із системою та дозволяє скоротити час на пошук інформації. За статистичними даними компанії IDC, використання функцій швидкого пошуку в корпоративних системах дозволяє зменшити витрати часу на обробку інформації до 30 %, що безпосередньо впливає на ефективність роботи персоналу.

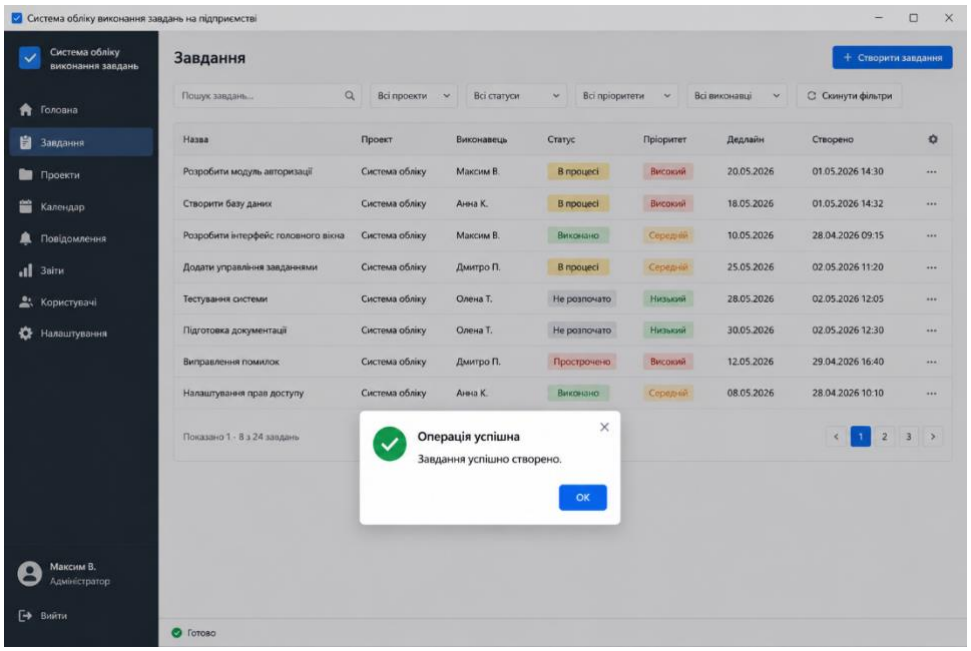

Рис. 4.4. Перегляд та контроль виконання завдань

На рисунку 4.4 наведено приклад перегляду та контролю виконання завдань у системі із використанням механізмів фільтрації та сортування інформації.

Під час реалізації інтерфейсу особлива увага приділялася питанням стабільності та швидкодії програмного забезпечення. Усі дії користувача супроводжуються миттєвим оновленням інтерфейсу без необхідності перезапуску програми.

Для цього використовувалися механізми прив’язки даних Data Binding, які дозволяють автоматично синхронізувати зміни між інтерфейсом та програмною логікою системи. Крім того, було реалізовано систему

повідомлень про помилки та результати виконання операцій, що забезпечує зручний контроль за роботою програми та підвищує рівень взаємодії користувача із системою.

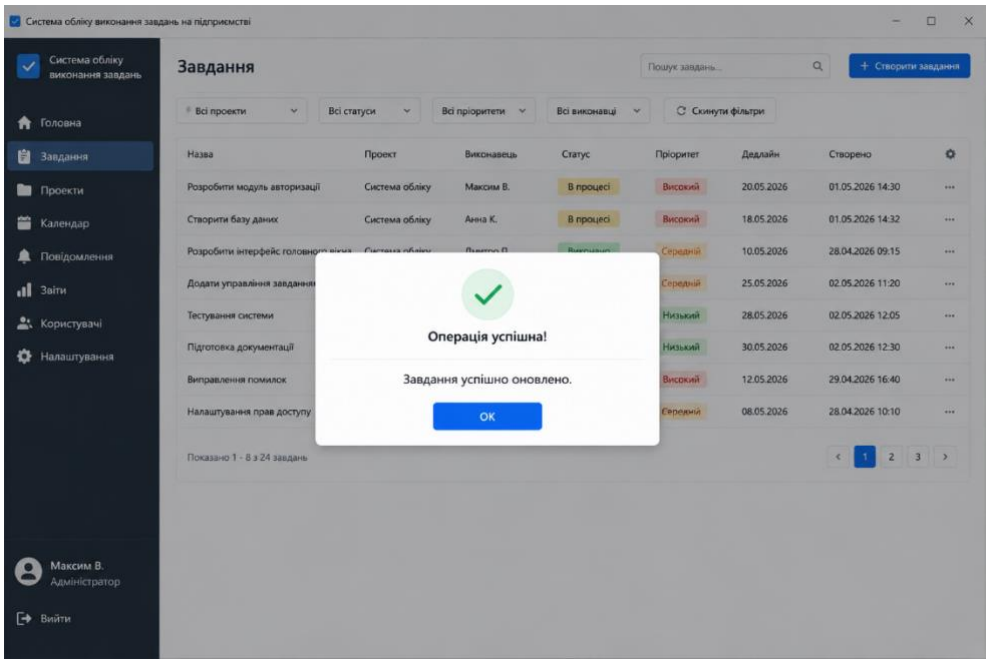

Рис. 4.5. Системне повідомлення про результат виконання операції

На рисунку 4.5 представлено приклад системного повідомлення, яке інформує користувача про успішне виконання операції у програмному забезпеченні.

4.3 Проведення тестування та аналіз результатів роботи системи

Завершальним етапом розробки програмного забезпечення системи обліку виконання завдань на підприємстві стало проведення тестування та аналіз результатів роботи системи. Тестування є важливою складовою процесу створення будь-якого програмного продукту, оскільки саме на цьому етапі перевіряється стабільність роботи програми, коректність реалізації функціональних можливостей та відповідність системи поставленим вимогам.

У сучасних умовах розробки програмного забезпечення якість тестування безпосередньо впливає на надійність роботи інформаційної

системи та рівень задоволеності користувачів. За даними компанії IBM, виправлення програмних помилок після впровадження системи в експлуатацію може коштувати у 10–15 разів дорожче, ніж їх усунення на етапі тестування, що підтверджує важливість ретельної перевірки програмного продукту перед його використанням у реальних умовах.

У межах даної бакалаврської кваліфікаційної роботи тестування проводилося поетапно та охоплювало перевірку основних функціональних модулів системи. Основна увага приділялася тестуванню механізмів авторизації користувачів, створення та редагування завдань, зміни статусів, пошуку інформації, збереження даних та взаємодії окремих компонентів програмного забезпечення.

Крім функціонального тестування проводилася перевірка стабільності роботи інтерфейсу користувача, швидкодії системи та правильності обробки помилок. Такий комплексний підхід дозволив оцінити якість реалізації програмного забезпечення та виявити потенційні недоліки у роботі системи.

Одним із перших етапів тестування стала перевірка модуля авторизації користувачів. Під час тестування було здійснено серію спроб входу в систему з правильними та неправильними обліковими даними. У результаті перевірки було встановлено, що система коректно виконує перевірку логіна та пароля, а також блокує доступ у випадку введення некоректної інформації.

Крім того, тестування підтвердило правильність роботи механізму розмежування прав доступу користувачів залежно від їх ролі у системі. Наприклад, звичайний користувач не має можливості отримати доступ до адміністративних функцій програмного забезпечення, що забезпечує захист інформації та відповідає сучасним вимогам до корпоративних інформаційних систем.

Наступним важливим етапом стало тестування модуля створення та редагування завдань. Під час перевірки користувачі створювали нові задачі із різними параметрами, змінювали статуси виконання та редагували інформацію про завдання. У процесі тестування було встановлено, що система

правильно обробляє введені дані, своєчасно оновлює інформацію у списку завдань та автоматично зберігає зміни у структурі даних.

Окремо проводилася перевірка механізмів валідації даних. Система не дозволяла створювати завдання без назви або визначеного виконавця, а у випадку введення некоректної інформації виводила відповідні повідомлення про помилки. Це дозволило забезпечити цілісність даних та підвищити рівень надійності програмного забезпечення.

Для перевірки стабільності роботи системи було проведено тестування навантаження. У процесі тестування до системи було додано понад 1000 записів про завдання, після чого виконувались операції пошуку, фільтрації та зміни статусів задач. За результатами перевірки було встановлено, що програмне забезпечення стабільно працює при значній кількості записів та не демонструє суттєвого зниження продуктивності. Час відкриття головного вікна програми залишався в межах 1–2 секунд, а операції пошуку виконувалися практично миттєво. Такі результати свідчать про ефективність обраної архітектури системи та правильність реалізації механізмів обробки даних.

Особливу увагу під час тестування було приділено перевірці механізмів збереження інформації. Система використовує формат JSON для локального зберігання даних, тому важливо було перевірити правильність серіалізації та десеріалізації об’єктів.

У процесі тестування виконувалося багаторазове створення, оновлення та видалення завдань із подальшим перезапуском програмного забезпечення. Результати перевірки показали, що всі дані коректно зберігаються у файлі та автоматично відновлюються після повторного запуску програми. Крім того, система правильно обробляє випадки відсутності файлу або пошкодження структури даних, що підвищує стабільність роботи програмного забезпечення.

Під час тестування графічного інтерфейсу користувача перевірялася правильність відображення елементів керування, коректність оновлення інформації та швидкість реакції системи на дії користувача. У процесі

перевірки було встановлено, що всі елементи інтерфейсу працюють стабільно, а оновлення інформації виконується без необхідності перезапуску програми. Для реалізації автоматичного оновлення даних використовувалися механізми Data Binding, що забезпечило синхронізацію інформації між інтерфейсом та програмною логікою системи. За результатами тестування було встановлено, що інтерфейс програмного забезпечення є зрозумілим та зручним для користувачів, а навігація між функціональними модулями не викликає труднощів.

Для перевірки правильності роботи окремих функціональних модулів використовувалися контрольні сценарії тестування, які передбачали послідовне виконання типових дій користувача. Наприклад, перевірялася можливість створення нового завдання, зміни його статусу, пошуку за назвою та автоматичного оновлення інформації після внесення змін. Результати тестування підтвердили правильність роботи основних функцій системи та відсутність критичних помилок у роботі програмного забезпечення. Крім того, було встановлено, що використання модульної структури програми значно спрощує процес виявлення та виправлення помилок у роботі системи.

Аналіз результатів тестування показав, що розроблене програмне забезпечення повністю відповідає функціональним вимогам, визначеним на етапі проєктування системи. Реалізовані механізми авторизації користувачів, управління завданнями, пошуку інформації та збереження даних працюють стабільно та забезпечують ефективну автоматизацію процесів управління завданнями на підприємстві.

Отримані результати підтвердили правильність обраних технологій розробки та ефективність використання платформи .NET і технології WPF для створення настільного програмного забезпечення корпоративного призначення. Проведене тестування дозволило оцінити якість програмного продукту та підтвердило можливість його використання для автоматизації робочих процесів сучасного підприємства.

4.4 Висновки до розділу 4

У четвертому розділі бакалаврської кваліфікаційної роботи було проведено тестування розробленого програмного забезпечення системи обліку виконання завдань на підприємстві, визначено основні вимоги до апаратного та програмного забезпечення, а також розглянуто склад інсталяційного пакету програмного продукту. У процесі тестування перевірено коректність роботи основних функціональних модулів системи, стабільність функціонування програмного забезпечення, правильність обробки даних та зручність взаємодії користувача з інтерфейсом програми.

Отримані результати підтвердили відповідність розробленої системи визначеним функціональним і технічним вимогам, а також можливість її використання для автоматизації процесів управління завданнями на підприємствах малого та середнього бізнесу. Визначені системні вимоги та структура інсталяційного пакету забезпечують можливість швидкого впровадження програмного забезпечення та його подальшої експлуатації в реальних умовах роботи підприємства.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено програмне забезпечення системи обліку виконання завдань на підприємстві, яке забезпечує автоматизацію основних процесів управління задачами, контроль виконання робіт та організацію взаємодії між працівниками підприємства. У процесі дослідження було проведено аналіз предметної області, визначено основні проблеми організації обліку завдань у сучасних підприємствах та обґрунтовано необхідність впровадження спеціалізованих інформаційних систем для автоматизації робочих процесів. Встановлено, що використання цифрових рішень дозволяє значно підвищити ефективність роботи персоналу, зменшити кількість помилок при передачі інформації та забезпечити централізований контроль виконання поставлених задач.

Під час виконання роботи було здійснено огляд сучасних програмних систем управління завданнями та проаналізовано їх основні функціональні можливості. Проведене дослідження показало, що більшість існуючих програмних продуктів орієнтовані на великі компанії або команди розробників програмного забезпечення, що ускладнює їх використання для невеликих підприємств через складність інтерфейсу, надлишковий функціонал та значні фінансові витрати. На основі проведеного аналізу було сформовано перелік основних вимог до розроблюваної системи та визначено функціональні можливості, необхідні для ефективної автоматизації процесів управління завданнями.

У межах бакалаврської кваліфікаційної роботи було обґрунтовано вибір технологій та інструментальних засобів розробки програмного забезпечення. Для реалізації системи було використано мову програмування C#, платформу .NET та технологію Windows Presentation Foundation (WPF), що дозволило створити сучасний настільний застосунок із графічним інтерфейсом

користувача. Використання об’єктно-орієнтованого підходу та модульної структури програми забезпечило гнучкість архітектури системи, спростило процес підтримки програмного забезпечення та створило можливість подальшого розширення функціональних можливостей програмного продукту.

У процесі проєктування програмного забезпечення було виконано UML-моделювання системи та побудовано основні моделі програмного забезпечення, серед яких UML-діаграма прецедентів, UML-діаграма класів, UML-діаграма послідовності, UML-діаграма станів, UML-діаграма компонентів та ER-діаграма бази даних системи. Використання UML-моделювання дозволило сформувати структуроване уявлення про архітектуру програмного забезпечення, визначити основні функціональні компоненти системи та забезпечити логічну організацію взаємодії між окремими модулями програмного продукту. Крім цього, було визначено основні функціональні вимоги до системи, реалізовано архітектуру програмного забезпечення та побудовано структуру даних для збереження інформації про завдання та користувачів.

Особлива увага під час реалізації програмного забезпечення приділялася створенню механізмів авторизації користувачів, управління завданнями, зміни статусів виконання задач та пошуку інформації. Для локального збереження даних було використано формат JSON, що дозволило забезпечити простий та ефективний механізм роботи з інформацією без необхідності використання серверної системи управління базами даних.

У процесі реалізації програмного забезпечення було створено основні функціональні модулі системи, серед яких модуль авторизації користувачів, модуль управління завданнями, модуль роботи з даними та модуль взаємодії з графічним інтерфейсом користувача. Реалізована система забезпечує можливість створення нових задач, призначення виконавців, зміни статусів виконання, контролю термінів завершення робіт та формування базової статистики щодо виконання завдань.

У межах роботи також було реалізовано механізми пошуку та фільтрації інформації, що дозволяють швидко знаходити необхідні завдання за визначеними параметрами. Додатково реалізовано автоматичне оновлення статусів задач у випадку перевищення встановлених термінів виконання. Такі функціональні можливості забезпечують ефективний контроль за виконанням робочих процесів та дозволяють керівництву підприємства своєчасно реагувати на проблемні ситуації, пов’язані із порушенням дедлайнів або перевантаженням працівників.

Важливим етапом виконання бакалаврської кваліфікаційної роботи стало проведення тестування програмного забезпечення. У процесі тестування було перевірено коректність роботи основних функціональних модулів системи, стабільність роботи інтерфейсу користувача, механізми збереження даних та швидкодію програмного забезпечення при значному обсязі інформації. Результати тестування підтвердили правильність реалізації основних функцій системи та відсутність критичних помилок у роботі програмного продукту. Було встановлено, що система стабільно працює при обробці великої кількості завдань та забезпечує швидкий доступ до інформації без суттєвого навантаження на ресурси комп’ютера.

Практичне значення отриманих результатів полягає у можливості використання розробленого програмного забезпечення для автоматизації процесів управління завданнями на підприємствах малого та середнього бізнесу. Впровадження системи дозволяє оптимізувати роботу персоналу, забезпечити централізований контроль виконання задач та покращити організацію внутрішніх бізнес-процесів.

Крім того, програмне забезпечення може бути використане як основа для подальшого розвитку інформаційної системи шляхом інтеграції з базами даних, мережевими сервісами або мобільними застосунками. Перспективами подальшого розвитку системи є реалізація мережевої багатокористувацької роботи, інтеграція із серверними системами управління базами даних, створення веб-версії програмного забезпечення та розширення

функціональних можливостей системи відповідно до потреб сучасних підприємств.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коноваленко І. В., Марущак П. О. Платформа .NET та мова програмування C# 8.0 : навчальний посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с. URL: https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf

2. Купін А. І., Музика І. О. Мережні інформаційні технології. Практикум : навчальний посібник. Кривий Ріг : ФО-П Чернявський Д. О., 2015. 238 с. URL: https://lib.iitta.gov.ua/id/eprint/706684/1/Kupin_Muzyka_8.pdf

3. Гаєвий А. О. Кваліфікаційна робота : пояснювальна записка. Харків : ХНУРЕ, 2025. 86 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/486ce2be-c0ee-4960-9dcc-7ad49faf7150/content

4. C# WPF & UWP Developer : курс розробки настільних застосунків. URL: https://itvdn.com/ua/specialities/dnet-desktop-dev

5. Ahmed Z. та ін. Machine Learning at Microsoft with ML.NET. 2019. 18 p. URL: https://arxiv.org/abs/1905.05715

6. Об’єктно-орієнтоване програмування : конспект лекцій. Київ : КПІ ім. Ігоря Сікорського, 2016. 142 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/5e62512f-9fb0-4086-8b56-9a3d0b7f9b0e/content

7. Пугановський О. В. Об’єктно-орієнтоване програмування : методичні вказівки. Харків : НТУ «ХПІ», 2024. 43 с. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/e74ff3fe-14d0-4d85-b404-79f979d488f8/content

8. Труніна Г. О. Обчислювальна техніка та програмування : конспект лекцій. Київ : КПІ ім. Ігоря Сікорського, 2020. 111 с. URL: https://ela.kpi.ua/items/2dfed7cf-9cfe-49c9-990c-a644a019d209

9. Труніна Г. О. Обчислювальна техніка та програмування : практикум. Київ : КПІ ім. Ігоря Сікорського, 2022. 95 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/6d82174e-d675-4d2c-b096-3dbe2ce4a534/content

10. Настенко Д. В. Об’єктно-орієнтоване програмування. Частина 1. Основи об’єктно-орієнтованого програмування на мові C# : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2022. 160 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/6b8d13d6-901d-4b07-835e-30e165ec5085/content

11. Шматко О. В. Аналіз методів і технологій розробки програмного забезпечення. Частина 2 : навчальний посібник. Харків : НТУ «ХПІ», 2018. 177 с. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/a4786ba9-e3ae-431c-bd93-a1be1a0eb64b/content

12. Гаєвий А. О. Кваліфікаційна робота : пояснювальна записка. Харків : ХНУРЕ, 2025. 86 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/486ce2be-c0ee-4960-9dcc-7ad49faf7150/content

13. Смотрова А. О. Програмний застосунок для автоматизації заповнення медичних документів : кваліфікаційна робота. Харків : ХНУРЕ, 2025. 78 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/e1fcf087-4f99-4f86-a157-e65d0d040f09/content

14. Рудак В. М. Проектування та розробка бази даних поліклініки з використанням методології RUP та мови програмування C# : кваліфікаційна робота. Тернопіль : ТНТУ ім. Івана Пулюя, 2023. 71 с. URL: https://elartu.tntu.edu.ua/handle/lib/44481

15. Шиденко Я. І. Розробка емулятора ігрової приставки NES з використанням мови програмування C# : бакалаврська кваліфікаційна робота. Тернопіль : ТНТУ ім. Івана Пулюя, 2024. 80 с. URL: https://elartu.tntu.edu.ua/handle/lib/45443

Додаток А

Структура класу taskitem та основний програмний код системи

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.Json;

namespace TaskManagementSystem
{
    public class TaskItem
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public DateTime CreatedDate { get; set; }
        public DateTime Deadline { get; set; }
        public string Status { get; set; }
        public string Priority { get; set; }
        public string Employee { get; set; }
    }

    public class TaskService
    {
        private List<TaskItem> tasks = new List<TaskItem>();

        public void AddTask(TaskItem task)
        {
            tasks.Add(task);
            SaveTasks();
        }

        public void DeleteTask(TaskItem task)
        {
            tasks.Remove(task);
            SaveTasks();
        }

        public List<TaskItem> GetTasks()
        {
            return tasks;
        }

        public void SaveTasks()
        {
            string json = JsonSerializer.Serialize(tasks);
            File.WriteAllText("tasks.json", json);
        }
```

```
        public void LoadTasks()
        {
            if(File.Exists("tasks.json"))
            {
                string json = File.ReadAllText("tasks.json");
                tasks =
JsonSerializer.Deserialize<List<TaskItem>>(json);
            }
        }
    }
}
```

Додаток Б

Приклад графічного інтерфейсу користувача wpf

```
<Window x:Class="TaskManagementSystem.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Система обліку виконання завдань"
        Height="600"
        Width="900">
    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <TextBlock Text="Система управління завданнями"
                   FontSize="24"
                   FontWeight="Bold"
                   Margin="0,0,0,15"/>

        <ListView Grid.Row="1"
                  x:Name="TaskListView">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Назва"
                                    Width="180"

DisplayMemberBinding="{Binding Title}"/>
                    <GridViewColumn Header="Виконавець"
                                    Width="140"

DisplayMemberBinding="{Binding Employee}"/>
                    <GridViewColumn Header="Статус"
                                    Width="120"

DisplayMemberBinding="{Binding Status}"/>
                    <GridViewColumn Header="Пріоритет"
                                    Width="120"

DisplayMemberBinding="{Binding Priority}"/>
                    <GridViewColumn Header="Термін"
                                    Width="140"

DisplayMemberBinding="{Binding Deadline}"/>
                </GridView>
            </ListView.View>
```

```
        </ListView>

        <StackPanel Grid.Row="2"
                    Orientation="Horizontal"
                    HorizontalAlignment="Right"
                    Margin="0,15,0,0">
            <Button Content="Додати"
                    Width="120"
                    Height="35"
                    Margin="5"/>
            <Button Content="Редагувати"
                    Width="120"
                    Height="35"
                    Margin="5"/>
            <Button Content="Видалити"
                    Width="120"
                    Height="35"
                    Margin="5"/>
        </StackPanel>
    </Grid>
</Window>
```

Додаток В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Вітківський Максим Сергійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення системи обліку виконання завдань на підприємстві» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [✓] інструменти ШІ (вказати назву: ChatGPT) були використані для:
        - [✓] перекладу іншомовних джерел;
        - [✓] стилістичного редагування та перевірки граматики;
        - [✓] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Максим ВІТКІВСЬКИЙ**
(підпис)