Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмної системи для автоматизації процесів управління персоналом»

Виконав: студент IV курсу
групи ЗПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Лозан О. В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Рейда О. М.
(прізвище та ініціали)

Рецензент: д.т.н., проф. каф. ОТ МартинюкТ.Б.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ д.т.н., проф. Романюк О.Н.
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Лозану Олександру Володимировичу

1. Тема роботи – розробка програмної системи для автоматизації процесів управління персоналом.

Керівник роботи: Рейда Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: тип програми – комп'ютерна система; модель розробки – ітеративна; СУБД для керування базою даних: SQLLite; вхідні дані: інформація для входу в особистий кабінет, особисті дані персоналу, метадані запитів; вихідні дані: відображення даних про персонал, завдання; середовище розробки – MS Visual Studio; мова програмування: C#; використані додаткові бібліотеки та фреймворки: .NET.

4. Зміст текстової частини: вступ, обґрунтування вибору методу розробки та постановка задач дослідження; розробка програмного забезпечення; тестування програмного забезпечення; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми дослідження; мета, об'єкт та предмет дослідження; задачі дослідження; новизна і практична цінність одержаних результатів; аналоги; методи та засоби розробки; блок-схема алгоритму роботи модуля для ведення обліку співробітників; блок-схема алгоритму роботи модуля створення завдань та їх призначення виконавцеві; загальний алгоритм системи у вигляді діаграми станів; інтерфейс користувача; інтерфейс вікна створення завдання та ведення обліку працівників; висновки; публікації й апробації; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Рейда О. М. к.т.н., доцент кафедри ПЗ	13.03.24	03.06.2

7. Дата видачі завдання 13.03.24

КАЛЕНДАРНИЙ ПЛАН

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примі
1	Аналіз розвитку автоматизованих систем управління персоналом та постановка задач дослідження	13.03.24 – 22.03.24	Виконано
2	Розробка алгоритмів програмного продукту	13.03.24 – 22.03.24	Виконано
3	Розробка автоматизованої системи управління персоналом	25.03.24 – 19.04.24	Виконано
4	Тестування програмного забезпечення	22.04.24 – 17.05.24	Виконано
5	Оформлення матеріалів до захисту БДР	20.05.24 – 03.06.24	Виконано

Студент _____ **Лозан (**
(підпис) (ініціали та пр

Керівник бакалаврської дипломної роботи _____ **Рейда (**
(підпис) (ініціали та пр

АНОТАЦІЯ

УДК 331:004.4

Лозан О. В. Розробка програмної системи для автоматизації процесів управління персоналом: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення. Вінниця : ВНТУ, 2024. 60 с.

На укр. мові. Бібліогр. : 20 назв.; рис. : 39; табл. 3: .

У бакалаврській дипломній роботі було визначено доцільність, практичну цінність та мету розробки, яка полягає в удосконаленні та автоматизації систем управління персоналом.

Удосконалено метод створення завдань та вибору виконавців, що на відміну від існуючого, використовує регулярні вирази з ваговими коефіцієнтами до професійних навиків, для покращення відповідності виконавців задачам, що сприятиме підвищенню ефективності роботи.

Система для автоматизації процесів управління персоналом розроблена із використанням засобів розробки MS Visual Studio та мови програмування C#. Під час розробки використано систему управління базами даних SQLLite. Розроблено методику тестування та проведено тестування автоматизованої системи згідно методики. Працездатність та ефективність роботи системи перевірено. Розроблено інструкцію користувача та визначено системні вимоги.

Отримані в бакалаврській дипломній роботі результати можна використати для впровадження в бізнес-середовищі, зокрема для вдосконалення та автоматизації основних процесів.

Ключові слова: бізнес-середовище, конфіденційність, ефективність роботи.

ABSTRACT

UDC 331:004.4

Lozan O. V. Development of an automated personnel management system: Vinnytsia: VNTU, 2024. 60 p.

In Ukrainian speech Bibliogr. : 20 name; Fig. : 39; table : 3.

In the bachelor's thesis, the feasibility, practical value, and purpose of the development were determined, which aim at improving and automating personnel management systems. The method for task creation and selection of performers has been enhanced. Unlike the existing method, it uses regular expressions with weighted coefficients for professional skills to improve the match between performers and tasks, which will contribute to increased work efficiency.

The system for automating personnel management processes was developed using MS Visual Studio development tools and the C# programming language. The development used the SQLLite database management system. A testing methodology was developed, and the automated system was tested according to this methodology. The functionality and efficiency of the system were verified. A user manual was developed, and system requirements were determined.

The results obtained in the bachelor's thesis can be used for implementation in a business environment, particularly for improving and automating key processes.

Keywords: business environment, confidentiality, work efficiency.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У змінному та конкурентному середовищі сучасного бізнесу, де кожен ресурс, включаючи час та людський капітал, має величезну цінність, автоматизовані системи управління персоналом стають невід'ємною складовою успішності організацій. Розробка таких систем передбачає не лише впровадження технічних рішень, але й аналіз процесів, взаємодію зі співробітниками та стратегічне планування кадрової політики.

Підходячи до створення автоматизованої системи управління персоналом, необхідно врахувати унікальність кожної компанії: її розмір, галузь діяльності, корпоративну культуру та потреби персоналу. Такий підхід дозволяє розробити індивідуалізовану систему, яка оптимізує рутинні процеси, підвищує ефективність управління та сприяє збільшенню задоволеності та залученості співробітників.

На сьогоднішній день існує ряд проблем у наявних аналогів систем управління персоналом, які можуть обмежувати їх ефективність і впливати на результативність організацій:

1. Ручне керування та рутинні процеси. Багато аналогових систем вимагають великої кількості ручних операцій, що призводить до збільшення часу на адміністративні завдання та може призвести до помилок у веденні документації.

2. Обмежені можливості аналізу та звітності. Деякі аналогові системи не забезпечують достатніх інструментів для аналізу даних про персонал та створення звітів, що ускладнює прийняття обґрунтованих управлінських рішень.

3. Недостатня автоматизація процесів. Багато аналогових систем не мають достатнього рівня автоматизації, що може призводити до затримок у виконанні завдань та збільшення трудомісткості роботи з персоналом.

4. Нестабільність та низька безпека даних. Деякі аналогові системи можуть бути вразливими перед загрозами кібербезпеки, а також мають обмежені

можливості забезпечення захисту конфіденційності та цілісності даних про персонал.

5. Недостатня гнучкість та адаптованість. Багато аналогових систем не можуть ефективно адаптуватися до змін у вимогах бізнесу та ринкових умов, що ускладнює їх використання в динамічних умовах сучасного бізнесу.

Тому реалізація автоматизованої системи управління персоналом є актуальною розробкою у наш час.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є вдосконалення та автоматизація процесів управління персоналом за рахунок впровадження сучасних технологій та шляхом розробки автоматизованої системи, що сприятиме підвищенню ефективності роботи, зменшенню необхідного часу для ведення обліку співробітників та моніторингу виконання завдань.

Основними задачами роботи є:

- провести аналіз існуючих методів і засобів розробки системи для автоматизації процесів управління персоналом для визначення напрямків підвищення продуктивності виконання поставлених задач;
- розробка модуля графічного інтерфейсу, призначеного для відображення контенту та його управлінням;
- запропонувати метод створення завдань та вибору виконавців;
- розробка модуля для створення завдань та їх управлінням;
- розробка модуля для відображення та редагування даних співробітників;
- розробка моделі даних та архітектури системи;
- проведення експериментальних досліджень розроблених засобів і тестування системи.

Об'єктом дослідження є процес автоматизації процесів у сфері управління персоналом.

Предметом дослідження є методи та засоби розробки програмної системи для автоматизації управління персоналом.

Методи дослідження. У процесі дослідження використано такі методи дослідження: комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень; теорія реляційних баз даних; UML-діаграми; аналіз потоків даних; прототипування; методи оптимізації даних; методи проведення тестування програмних застосунків, методи статистичної обробки даних.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод створення завдань та вибору виконавців, що на відміну від існуючого, використовує регулярні вирази з ваговими коефіцієнтами до професійних навиків, для покращення відповідності виконавців задачам, і як наслідок підвищення продуктивності і якості виконаних робіт.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положеннях було запропоновано сучасні підходи до автоматизованих систем управління персоналом та розроблено програмні засоби для вдосконалення процесу управління.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024) » [1].

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференцій.

1 АНАЛІЗ РОЗВИТКУ АВТОМАТИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ ПЕРСОНАЛОМ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану автоматизованих систем управління персоналом

У світі стрімкого технологічного прогресу та постійних змін у бізнес-середовищі, використання автоматизованих систем управління персоналом стає необхідністю для організацій, які прагнуть досягти ефективності та конкурентоспроможності. Сучасні підходи до управління персоналом вимагають не лише кваліфікованих кадрів, але й використання інноваційних інструментів та технологій для оптимізації процесів та підвищення продуктивності.

Автоматизовані системи управління персоналом забезпечують компаніям можливість ефективно керувати всіма аспектами персонального управління, від найму та розвитку персоналу до оцінки та мотивації працівників. Проте, незважаючи на широке використання таких систем, існують деякі виклики та проблеми, що потребують уваги та вирішення.

1. Зміни в організаційній культурі. Впровадження нової системи може призвести до змін в організаційній культурі, що може викликати опір серед деяких співробітників. Важливо провести ефективну комунікаційну стратегію та навчальні програми для зменшення опору та підтримки переходу.

2. Проблеми із сумісністю та інтеграцією. Якщо організація вже використовує інші програмні рішення, можуть виникнути проблеми з інтеграцією автоматизованої системи управління персоналом з існуючими системами. Це може призвести до неспрощених процесів та втрати часу на ручне введення даних.

3. Ризик залежності від технологій. Використання автоматизованих систем може зробити організацію більш вразливою до відмови систем або кібератак. Це може призвести до перерв у роботі та втрати даних, яка може значно вплинути на діяльність компанії.

4. Потреба в оновленнях та підтримці. Автоматизовані системи потребують постійного оновлення та підтримки, щоб забезпечити їхню ефективну роботу та відповідність законодавству. Це може призвести до додаткових витрат на обслуговування та навчання персоналу.

5. Проблеми з конфіденційністю та етикою. Збір та збереження великої кількості особистих даних про працівників може породжувати етичні та юридичні проблеми з конфіденційністю даних. Організації повинні бути дбайливими щодо збору, збереження та використання таких даних.

Використання автоматизованих систем управління персоналом надає компаніям ряд переваг, які підвищують ефективність та забезпечують успішне ведення бізнесу:

- підвищена продуктивність – автоматизовані системи допомагають у спрощенні та автоматизації рутинних завдань, таких як облік робочого часу, опрацювання відпусток та відгуків, що звільняє час кадрових спеціалістів для вирішення стратегічних завдань;
- покращений доступ до інформації – завдяки централізованій базі даних персоналу, всі відомості про співробітників доступні у одному місці, що спрощує пошук інформації та забезпечує швидкий обмін даними;
- оптимізація процесу найму та рекрутингу – системи автоматизованого управління персоналом дозволяють швидко опрацьовувати кандидатів, проводити автоматизований відбір та оцінку кандидатів, що прискорює процес найму та допомагає залучати кращі кадри;
- системи моніторингу виконання завдань – встановлення систем, що дозволяють відстежувати виконання завдань співробітниками, включаючи статус проектів, терміни виконання та результативність.

Отже, розробка системи для управління персоналом є важливим етапом для будь-якої організації, що прагне досягти ефективності, конкурентоспроможності та успіху на ринку. На основі аналізу автоматизованих систем управління персоналом можна зробити висновок, що вони принесуть ряд вигод для організації, включаючи підвищену продуктивність, оптимізацію

процесів найму та рекрутингу, покращений доступ до інформації, підвищення якості прийняття рішень та покращення комунікації з персоналом, важливо бути ознайомленим з усіма стандартами та рекомендаціями, щоб розробити надійну та, в першу чергу, ефективну систему.

1.2 Порівняльний аналіз аналогів

Використання систем для автоматизації управління персоналом стає все більш актуальним на фоні різкого зростання кількості даних та необхідності їх зберігання у безпечному середовищі, переходячи від колишніх методів, що полягали у використання фізичних паперів. Розробка автоматизованої системи зберігання та обробки даних може значно спростити та допомогти у автоматизації роботи бізнесу, та допоможе підвищити їх продуктивність. Проведення порівняльного аналізу цих систем допоможе визначити їх переваги та недоліки, а також визначити потенційні напрямки для вдосконалення власної автоматизованої системи управління персоналом.

Розглянемо популярні ресурси як аналоги розроблюваної системи, такі як:

PeopleHR – ця система дозволяє користувачам переглядати дані співробітників, оцінювати ефективність співробітників. Також можна розмежовувати права доступу, що не дасть доступу до інформації співробітникам про своїх колег (див. рисунок 1.1) [2].

Основний функціонал системи:

1. Повний спектр функцій управління персоналом. PeopleHR пропонує широкий спектр функцій, включаючи управління наймом, заробітною платою.
2. Хмарне рішення та легкий доступ. Будучи хмарною платформою, PeopleHR забезпечує легкий доступ до даних та функцій з будь-якого пристрою з підключенням до Інтернету. Це особливо корисно для компаній з розподіленими командами або віддаленими співробітниками.

3. Персоналізація та конфігурування. Платформа дозволяє налаштовувати та адаптувати процеси управління персоналом під конкретні потреби та вимоги компанії.

4. Підтримка клієнтів та навчання. PeopleHR надає добру підтримку клієнтів та навчальні ресурси для впровадження та використання системи.

До недоліків програмного продукту належать:

1. Вартість. Помітним мінусом PeopleHR може бути вартість. Хоча ціни можуть варіюватися залежно від обсягу функціональності та розміру компанії, для деяких малих підприємств це може бути високою витратою.

2. Складність інтеграції. Інтеграція PeopleHR з іншими системами, такими як бухгалтерське програмне забезпечення, може бути складною та вимагати додаткових зусиль.

3. Створення завдань. PeopleHR не надає функціонал для створення, призначення завдань відповідальній особі та відстеження виконання завдань.

Рисунок 1.1 – Зображення програмного продукту «PeopleHR»

HiBob – це хмарна платформа для управління персоналом, яка пропонує інтегровані рішення для керування кадрами та забезпечення зручного спілкування та співпраці між співробітниками (див. рисунок 1.2) [3].

Основний функціонал системи:

1. Управління персоналом. Відслідковування основних даних про співробітників, таких як контактна інформація, підрозділи, посади тощо.

2. Електронний документообіг. Можливість зберігати та обробляти документи, пов'язані з управлінням персоналом, такі як контракти, заяви про відпустки, оцінки роботи тощо.

3. Оцінка роботи та процеси звітності: Проведення оцінок роботи, створення звітів та аналіз ефективності персоналу.

До недоліків програмного продукту належать:

1. Вартість. Як і більшість хмарних платформ для управління персоналом, вартість використання HiBob може бути високою, особливо для малих компаній або стартапів.

2. Складність інтеграції. Інтеграція HiBob з іншими системами, такими як бухгалтерське програмне забезпечення або програми для спілкування в компанії, може бути складною та вимагати додаткових зусиль.

3. Створення завдань. HiBob не надає функціонал для створення, призначення завдань відповідальній особі та відстеження виконання завдань.

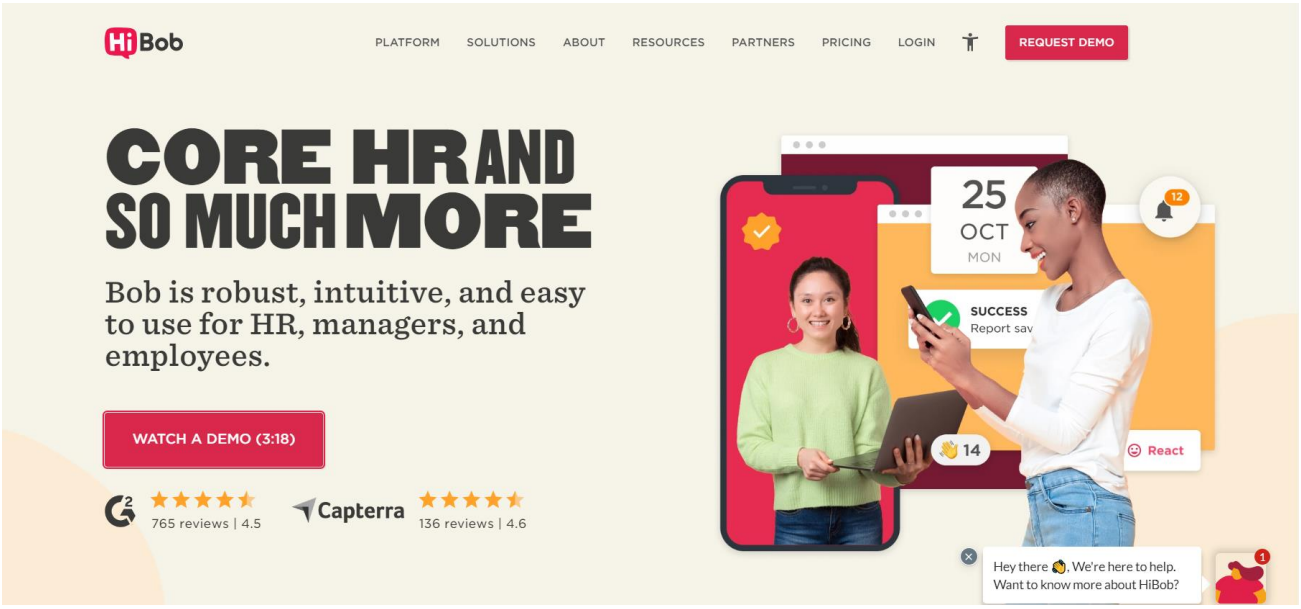


Рисунок 1.2 – Зображення програмного продукту «HiBob»

WebWork – це вебсистема, що дозволяє відстежувати відвідуваність працівників, не надає доступу до персональних даних працівників іншим співробітникам, окрім високопосадових осіб. (див. рисунок 1.3) [4].

Основний функціонал системи:

1. Управління проектами. Створення проектів, призначення завдань, встановлення термінів виконання та відстеження прогресу роботи.

2. Облік робочого часу. Відстеження часу, проведеного на кожному проекті або завданні, за допомогою таймерів або ручного введення даних.

3. Керування ресурсами. Планування та розподіл ресурсів, враховуючи їх доступність та вміння.

До недоліків програмного продукту належать:

1. Обмежені можливості безпеки та конфіденційності. Деякі користувачі можуть виявити, що система має обмежені можливості щодо захисту конфіденційних даних та безпеки інформації.

2. Складність використання для початківців. Для новачків система може виглядати складною та вимагати деякого часу на освоєння її функціоналу та інтерфейсу.

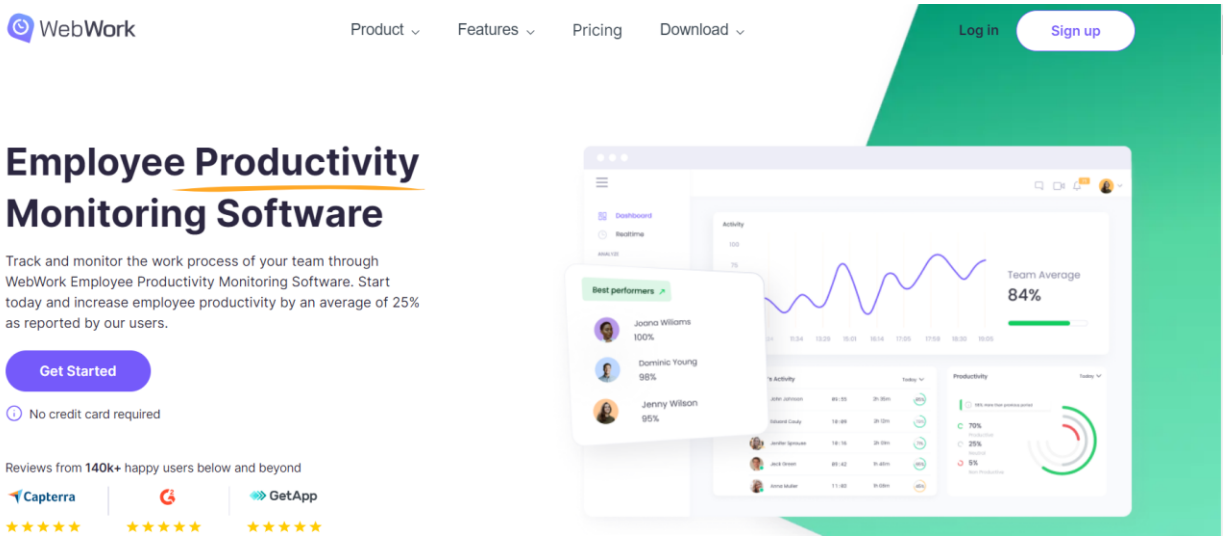


Рисунок 1.3 – Зображення програмного продукту «WebWork»

Asana – це популярна хмарна платформа для керування проектами, яка дозволяє командам організувати та відстежувати робочі завдання. На рисунку 1.4 зображено інтерфейс програмного продукту Asana.

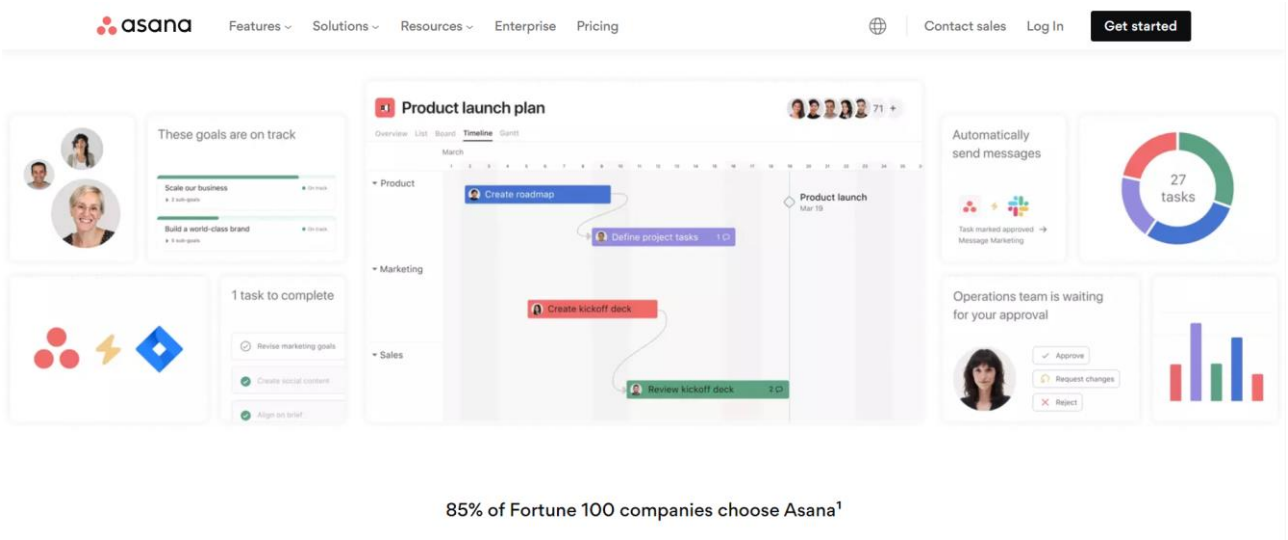


Рисунок 1.4 – Зображення програмного продукту «Asana»

Основний функціонал системи:

1. Можливість створювати нові завдання, описувати їх, встановлювати терміни та призначати відповідальних.
2. Можливість організовувати завдання у вигляді списків, проектів, а також за допомогою міток та категорій.
3. Можливість додавати коментарі, прикріплювати файли та спільно працювати над завданнями з учасниками команди.
4. Можливість відстежувати прогрес виконання завдань, переглядати завершені та незавершені завдання, а також отримувати сповіщення про важливі події.
5. Можливість перегляду графіків проектів, встановлення термінів та пріоритетів завдань.

Недоліки системи Asana можуть включати:

1. Для деяких користувачів інтерфейс Asana може виглядати складним та вимагати деякого часу для освоєння.

2. Деякі функції та можливості Asana доступні лише у платних версіях, що може відлякувати деяких користувачів.

3. Деякі користувачі можуть відчувати обмежені можливості інтеграції Asana з іншими програмами та сервісами.

Для наочної демонстрації відмінностей розглянутих додатків було зведено їх переваги й недоліки у таблицю порівняння (див. таблицю 1.1).

Таблиця 1.1. – Порівняльні характеристики програмних продуктів

	PeopleHR	HiBob	WebWork	Asana	Manage Employee
Моніторинг часу	0	1	1	0	0
Відображення даних співробітників	1	0	1	1	1
Редагування даних	1	1	0	1	1
Автоматична система	0	0	1	1	1
Захист даних	0	1	0	0	1
Сумарний коефіцієнт	2	3	3	3	4

Відповідно до таблиці порівняльних характеристик розробка власної системи Manage Employee для керування персоналом є доцільною. Отриманий продукт дасть можливість розробникам та потенційним зацікавленим особам використовувати ефективну та надійну систему.

Отже, порівняльний аналіз аналогів у сфері управління персоналом розкриває різноманітні підходи та можливості, що відкриваються перед розробниками програмної системи. Розуміння переваг та обмежень існуючих аналогів дозволяє налагодити оптимальні технічні рішення та функціонал, які відповідають вимогам та потребам користувачів.

Одержані в ході аналізу знання про існуючі рішення стають основою для вдосконалення та розробки нової автоматизованої системи управління

персоналом. Переваги та недоліки аналогів надають цінний вихідний матеріал для створення інноваційного продукту, спроможного задовольнити потреби сучасного ринку та підняти ефективність адміністрування на новий рівень.

1.3 Аналіз методів розв’язання задач

Розробка автоматизованої системи управління персоналом вимагають ретельного розгляду, щоб досягти максимальної ефективності. Існують різні методи, кожен з яких має свої переваги та недоліки. Для ефективного вирішення поставленої задачі потрібно ретельно вивчити і порівняти різні підходи та методи, що застосовуються у сфері автоматизованого керування персоналом.

Один з можливих методів полягає у застосуванні методів аналізу даних [5]. Використання аналітики даних для оцінки продуктивності, ефективності та задоволеності персоналу. Це може включати аналіз ключових показників продуктивності, таких як час виконання завдань, рівень відгуку клієнтів або показники якості роботи.

Другий метод включає в себе використання алгоритмів машинного навчання для автоматичного аналізу та передбачення патернів у поведінці персоналу [6]. Наприклад, моделі прогнозування відповідей співробітників на зміни в умовах роботи або рекомендації щодо оптимальних стратегій мотивації персоналу.

Третій метод полягає у використанні методів оптимізації, таких як Lean або Six Sigma, для пошуку та усунення непотрібних витрат часу та ресурсів у процесах управління персоналом [7].

Четвертий метод полягає у використанні аналізу мережі для вивчення взаємозв'язків та взаємодій між співробітниками, що дозволяє виявити ключових впливових осіб та покращити комунікацію та співпрацю в організації.

П’ятий метод полягає у використанні стратегій та методів управління змінами для успішного впровадження нових систем та процесів управління

персоналом, включаючи залучення та підтримку персоналу під час переходу до нових систем [8].

Для аналізу методів розв'язання задачі також важливо дослідити існуючі програмні продукти та сервіси, які вже здійснюють автоматизований підбір одягу. Це дозволить виявити переваги та недоліки існуючих рішень і врахувати їх при розробці власної системи.

Таким чином, аналіз методів розв'язання поставленої задачі дозволить обрати оптимальний шлях для розробки автоматизованої системи управління персоналом. Аналіз цих методів підтверджує важливість обрання комплексного підходу до розробки системи, який забезпечить якість та ефективність її функціонування.

1.4 Постановка задач для розробки автоматизованої системи

Проаналізувавши переваги та недоліки існуючих систем було визначено наступні завдання, які необхідно виконати для розроби автоматизованої системи:

- розробити загальний алгоритм роботи системи;
- розробити модуль графічного інтерфейсу, призначеного для відображення контенту та його управлінням;
- розробити модуль для створення завдань та їх управлінням;
- розробити модуль для відображення та редагування даних співробітників;
- розробити модель даних та архітектури системи;
- провести тестування системи.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто стан додатків з автоматизованою системою управління персоналом. Було проведено порівняння існуючих аналогів, таких як: PeopleHR, HiBob, WebWork, Asana.

У результаті порівняння було обґрунтовано актуальність розробки нової, незалежної системи управління персоналом, також, проаналізувавши наявні системи та методи розв’язання задач, було поставлено завдання на розробку із врахуванням усіх переваг та недоліків.

2 РОЗРОБКА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних та вихідних даних системи

Вхідні та вихідні дані є важливими компонентами будь-якої програми. Як правило, джерелом вхідних даних є користувач, який вводить інформацію за допомогою графічного інтерфейсу. До користувачів належать адміністратори, які контролюють ведення обліку співробітників та виконання завдань, та звичайні працівники, які бачать завдання, що необхідно виконати та їх терміни.

Насамперед користувач мусить додати до програми дані персоналу. Для цього необхідно вказати ім’я, посаду, заробітну плату тощо.

Не менш важливою є загальна інформація про співробітників, які може редагувати та переглядати адміністратор за допомогою графічного інтерфейсу.

Іншими функціями є створення завдань та їх призначення відповідному співробітнику з вказанням термінів виконання. Адміністратор може редагувати список завдань, зміни відображуються працівникові після оновлення списку завдань, які йому необхідно виконати.

Працівник може переглядати список завдань, де буде вказано його опис, статус та терміни виконання, що дозволить вчасно виконувати необхідні завдання.

Також працівник може переглядати та редагувати деякі свої дані у системі такі як логін та пароль для входу до свого облікового запису

Вхідні дані зберігаються у локальній базі даних. У процесі своєї роботи застосунок обробляє цю інформацію і відображає користувачу за допомогою графічного інтерфейсу у візуально привабливому вигляді.

До вихідних даних застосунку для управління персоналом можна віднести список працівників, до якого адміністратор може вносити необхідні зміни та список завдань які бачить працівник.

Крім того система веде статистику виконаних та невиконаних завдань, за чим може слідкувати адміністратор та оцінювати роботу співробітників.

2.2 Аналіз вибору архітектури програмної системи

Для створення програмного забезпечення (ПЗ) існують різноманітні архітектурні підходи, кожен з яких має свої особливості, переваги та недоліки. Ось кілька ключових архітектур, які часто використовуються при розробці програмного забезпечення:

1. Монолітна архітектура. У монолітній архітектурі весь функціонал програмного забезпечення об'єднаний у єдиний блок (моноліт), який виконується як один процес. Це простий підхід, особливо для невеликих проектів, де всі компоненти розроблені та підтримуються в межах одного додатка [9].

2. Клієнт-серверна архітектура. У цій архітектурі функціональність програмного забезпечення розділена між клієнтом і сервером. Клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію з користувачем, тоді як серверна частина відповідає за обробку даних та бізнес-логіку [10].

3. Мікросервісна архітектура [11]. У мікросервісній архітектурі функціональність програмного забезпечення розбивається на невеликі незалежні сервіси, які взаємодіють між собою за допомогою API. Це дозволяє швидко реагувати на зміни, підтримувати та масштабувати окремі компоненти системи.

Для створення власної програмної системи було вирішено використовувати монолітну архітектуру, що надає ряд переваг перед іншими:

1. Монолітні додатки мають просту інфраструктуру та не вимагають складних механізмів розгортання та керування. Це дозволяє швидко впроваджувати та підтримувати додаток.

2. У монолітних додатках весь код знаходиться в одному місці, що спрощує відлагодження та тестування. На підприємствах нерідко виникають проблеми в роботі програмного забезпечення, а використання монолітної архітектури дозволить розробникам швидко знаходити та виправляти помилки.

3. Завдяки простоті структури та єдності функціоналу, розробка та впровадження нового функціоналу в монолітних додатках може відбуватися швидше.

4. Монолітні додатки можуть бути більш ефективними з точки зору витрат, оскільки не вимагають складних механізмів підтримки. Це дозволить заощадити кошти підприємства на підтримку такого програмного забезпечення та витратити їх на інші потреби.

2.3 Розробка моделі системи

Для кращого усвідомлення принципів роботи автоматизованої системи управління персоналом, що складається з користувацького інтерфейсу та локальної бази даних, було вирішено побудувати модель її функціонування з використанням структурної UML діаграми компонентів [12]. Ця діаграма дозволяє наглядно відображати послідовність дій та процесів, які відбуваються в системі під час виконання адміністрування даних.

Структурна UML діаграма — включає діаграми класів та відображає структуру класів у системі та його взаємозв'язку, компонентів, що дозволяє уявити структуру високорівневих компонентів системи та його залежності і об'єктів, що фокусується на конкретних об'єктах у системі та його взаємодіях.

На рисунку 2.1 представлена UML діаграма компонентів, яка ілюструє взаємодію різних компонентів системи управління персоналом. Ця модель допомагає зрозуміти процеси взаємодії користувачем із системою. Розглядання такої діаграми сприяє кращому розумінню функціональності та логіки роботи системи.

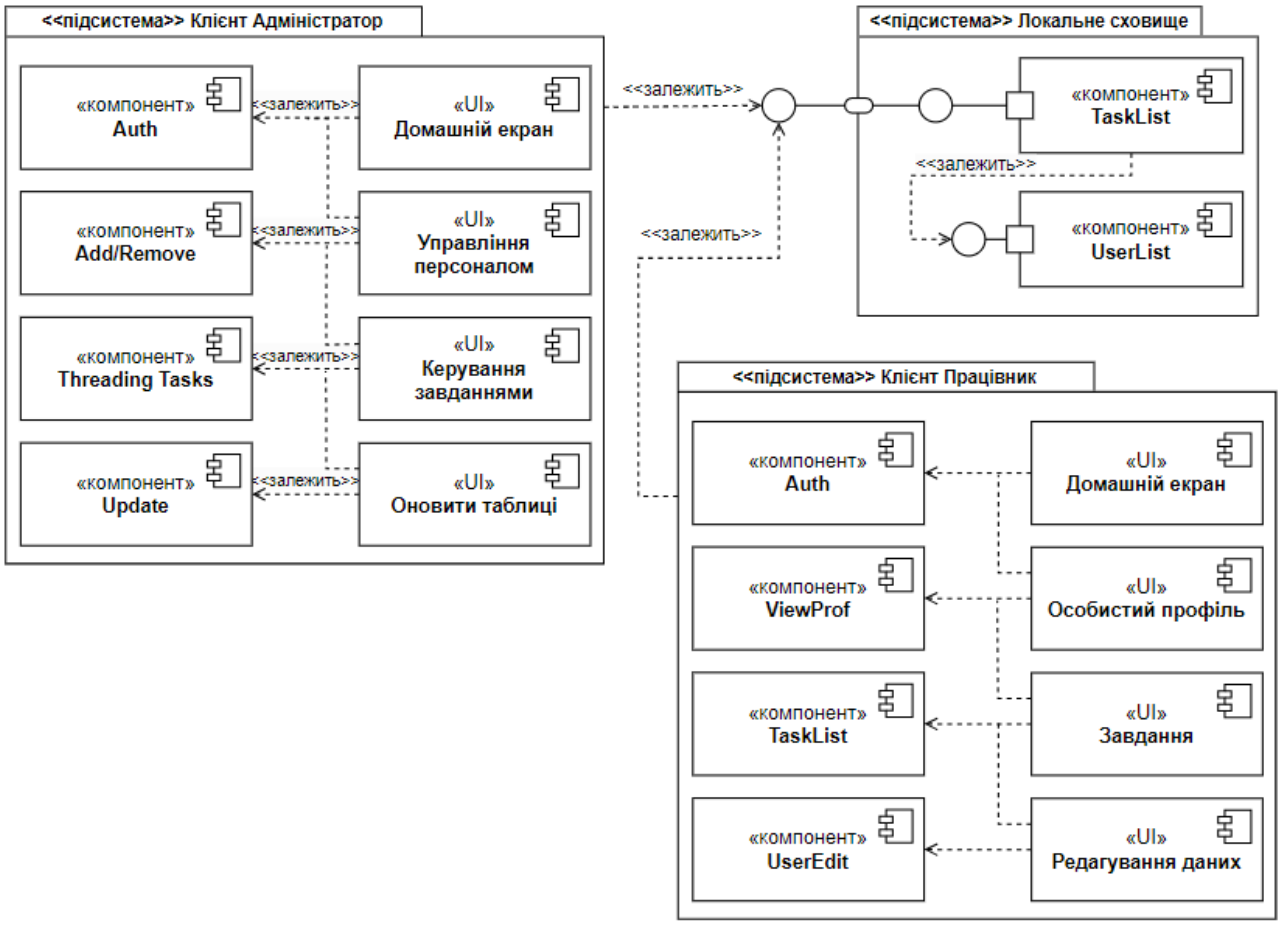


Рисунок 2.1 – Модель системи у вигляді діаграми компонентів

Ця модель дозволяє аналізувати та вдосконалювати функціональність системи управління персоналом, забезпечуючи оптимальний доступ до даних. Використання UML діаграм компонентів допомагає зрозуміти взаємозв'язки між різними компонентами системи та процесами взаємодії між ними.

Локальне сховище є важливою частиною моделі програми для управління персоналом. Вона відповідає за зберігання всієї інформації, пов'язаної з співробітниками та завданнями. Локальна БД дозволяє додатку надавати швидкий і оперативний доступ до збережених даних без необхідності підключення до віддаленого сервера.

База даних може бути реалізована за допомогою різних технологій і бібліотек. Для запланованої розробки було обрано базу даних SQLLite [13]. Вона є вбудованою базою даних, тобто не потребує окремого сервера для її роботи, а база даних зберігається в окремому файлі у файловій системі. Вона забезпечує

надійне та масштабоване рішення для зберігання та управління великими обсягами даних.

2.4 Розробка загального алгоритму роботи системи

При розробці складної програмної системи дуже важливо мати чітке уявлення про її загальну роботу. Це включає в себе визначення функціональності програми, взаємодію різних компонентів між собою та загальний потік даних і процесів. Один із способів досягти цього – розробка загального алгоритму.

Загальний алгоритм може допомогти отримати комплексне уявлення про функціональність застосунку, а також визначити потенційні вузькі місця і проблеми, які можуть виникнути в процесі розробки. Таким чином, можна оптимізувати дизайн програми, підвищити її ефективність та результативність.

Загальний алгоритм роботи програми для управління персоналом можна представити у вигляді діаграми станів, яка відображає основний спосіб використання програми: керування даними персоналу та створення і відстеження виконання завдань. Розроблена діаграма зображена на рисунку 2.2.

UML-діаграма станів - це тип діаграми, який моделює різні стани об'єкта або системи, а також переходи між цими станами. Ця діаграма дозволяє візуалізувати різні стани, події, що спричиняють переходи між станами, і дії, які відбуваються в кожному стані або при переході між ними [14].

Основні елементи UML-діаграми станів включають:

1. Стани. Представляють можливі стани системи або об'єкта. Кожен стан має унікальний ідентифікатор і може мати пов'язані дії та події.

2. Переходи. Показують переходи між різними станами. Вони відображають, як система або об'єкт переходить від одного стану до іншого при виникненні певної події або умови.

3. Події. Події є спричинюючими факторами для переходів між станами. Вони вказують, коли відбувається перехід від одного стану до іншого.

4. Дії. Дії виконуються в певних станах або під час переходів між станами. Вони вказують, які дії виконуються системою або об'єктом у конкретному контексті.

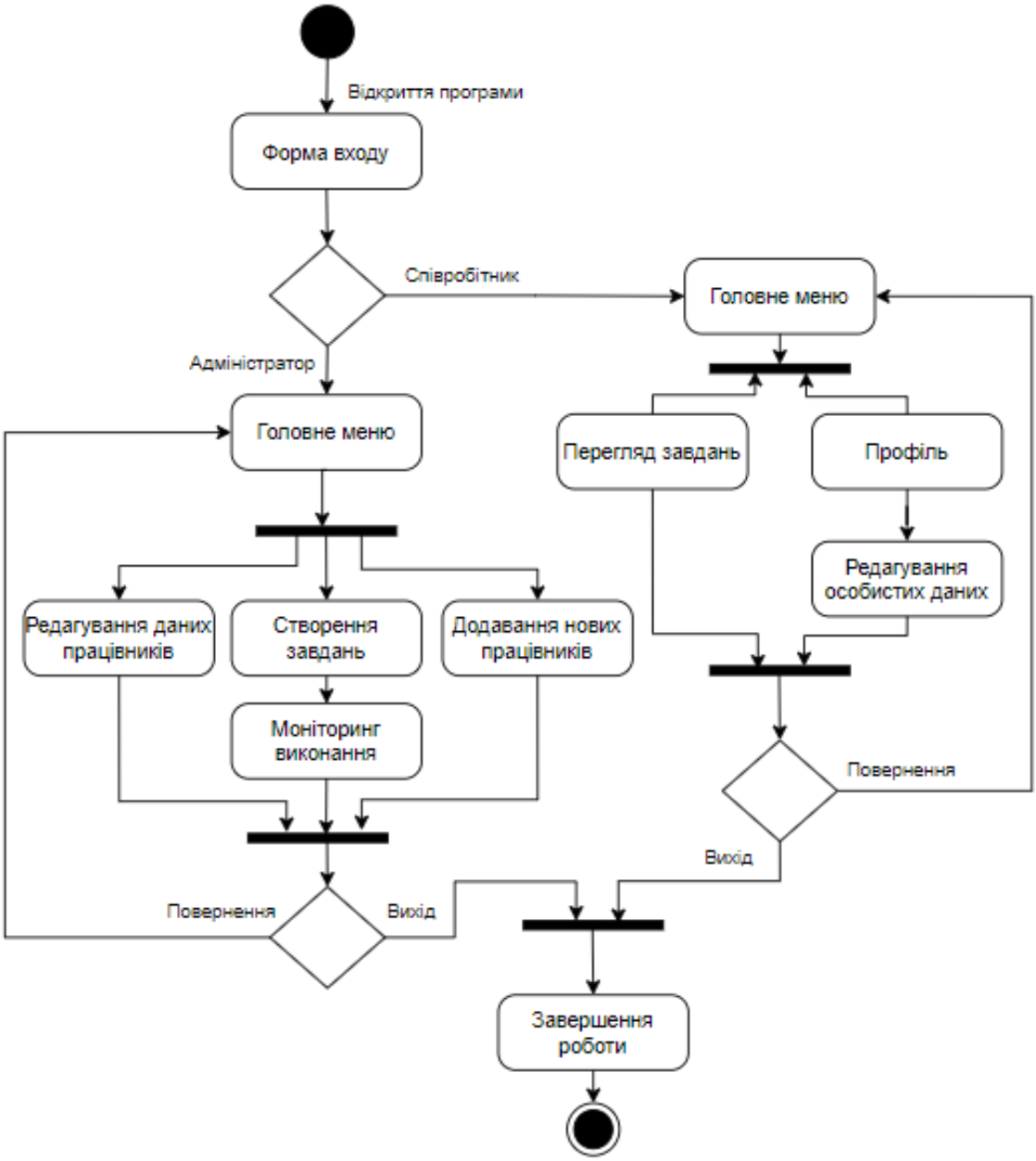


Рисунок 2.2 – Зображення діаграми станів загального алгоритму системи

Діаграма відображає різні сценарії роботи програми залежно від того який обліковий запис використовує користувач: адміністратора чи працівника. Це показано за допомогою розгалуження на діаграмі. Кінцева точка діаграми буде

досягнута, якщо всі дані є дійсними, були успішно додані до бази даних та користувач вийшов зі свого облікового запису.

2.5 Висновки

У розділі було проаналізовано вхідні та вихідні дані програми для відстеження читання. Далі, у цьому розділі було розроблено основний алгоритм роботи системи, який визначає послідовність операцій та взаємодію компонентів системи для досягнення поставленої мети.

Надалі, для кращого розуміння роботи програмного продукту, ми використали UML діаграми для моделювання його робочих процесів та структури. Це дозволяє нам чітко уявити взаємозв'язки між компонентами системи та спростити її розуміння та подальшу розробку.

3. РОЗРОБКА АВТОМАТИЗНОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Інтегровані середовища розробки (IDE) – це програмні системи, які надають комплексне середовище для розробки, тестування та розгортання програм. Коли мова йде про розробку додатків для Windows, існує кілька IDE, які можна використовувати.

MS Visual Studio – це інтегроване середовище розробки (IDE), розроблене компанією Microsoft для створення програмних продуктів для різних платформ, включаючи Windows, macOS, Android, iOS, веб-програмування та інші [15]. Ось деякі ключові характеристики Visual Studio:

- MS Visual Studio підтримує широкий спектр мов програмування, включаючи C#, C++, Visual Basic, F#, JavaScript, Python, і багато інших. Це робить його універсальним інструментом для розробки для різних платформ.
- MS Visual Studio має вбудовані інструменти для розробки, тестування, відлагодження та профілювання програм. Це включає в себе редактор коду з підсвічуванням синтаксису, інтелектуальне завершення коду, підтримку Git та інші корисні функції.
- MS Visual Studio надає потужні інструменти для відлагодження програм. Розробники можуть використовувати відлагоджувальник для пошуку й виправлення помилок, використовуючи різні режими відлагодження, такі як по кроці, пошук виразу, точка зупинки, а також додаткові інструменти, такі як відлагодження в паралельних процесах.
- MS Visual Studio має багато вбудованих шаблонів для швидкого створення різних типів проектів, таких як Windows Forms, WPF, ASP.NET, консольні програми та інші.

- MS Visual Studio має вбудовану підтримку спільної роботи, що дозволяє розробникам спільно працювати над проектами за допомогою інструментів командної розробки, таких як Azure DevOps або Team Foundation Server.

JetBrains Rider – це інтегроване середовище розробки, розроблене компанією JetBrains, відомою своїми продуктами для розробки програмного забезпечення, такими як IntelliJ IDEA, PyCharm, PhpStorm та інші. Ось деякі ключові особливості JetBrains Rider:

- Rider підтримує розробку на різних мовах програмування, включаючи C#, VB.NET, F#, JavaScript, TypeScript, HTML, CSS та інші. Він також має підтримку інших технологій, таких як ASP.NET, ASP.NET Core, Xamarin, Unity, WPF, WinForms, і багато інших.
- Rider надає потужний відлагоджувальник з підтримкою різних режимів відлагодження, включаючи по кроці, пошук виразу, точка зупинки, а також підтримку відлагодження в паралельних процесах та віддалених відлагоджувальників.
- Rider підтримується на різних операційних системах, включаючи Windows, macOS, Linux. Це робить його ідеальним вибором для розробки на різних платформах.
- Rider відомий своєю швидкістю та продуктивністю при роботі з великими проектами. Він має оптимізований движок для роботи з кодом, що дозволяє розробникам швидше і ефективніше робити свою роботу.

SharpDevelop – це безкоштовне інтегроване середовище розробки для платформи .NET. Він створений для розробки програм на мовах програмування, таких як C#, Visual Basic .NET та інші мови, що підтримуються платформою .NET Framework. Ось кілька ключових особливостей SharpDevelop:

- SharpDevelop підтримує розробку на мовах програмування C#, VB.NET, F#, а також підтримує редакцію файлів XML, HTML, CSS та інших.
- Інтерфейс користувача SharpDevelop приємний та інтуїтивно зрозумілий. Він має ряд зручних інструментів, таких як вбудований редактор

коду, відлагоджувальник, менеджер проектів та відкритих вікон, що робить роботу з проектами зручною та ефективною.

- SharpDevelop повністю інтегрований з .NET Framework, що дозволяє розробникам легко створювати та редагувати проекти, використовуючи функціональність, яка пропонується платформою .NET.
- SharpDevelop підтримує розширення та доповнення, що дозволяє розширювати його функціональність за допомогою сторонніх плагінів.
- SharpDevelop підтримує розробку програм для платформи Mono, що дозволяє розробникам створювати крос-платформенні додатки на основі платформи .NET.

Порівняльний аналіз на основі можливостей згаданих IDE наведено в таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз IDE

	MS Visual Studio	JetBrains Rider	SharpDevelop
Багатофункціональність	1	1	0
Швидкодія та продуктивність	1	1	1
Вартість	1	0	1
Інтегроване середовище	1	1	1
Інтеграція Git	1	1	0
Сумарний коефіцієнт	5	4	3

Як видно з таблиці, MS Visual Studio має перевагу над JetBrains Rider та SharpDevelop, що робить її кращим вибором для запланованої розробки. По-перше, Visual Studio це повнофункціональне інтегроване середовище розробки з широким спектром інтегрованих інструментів для роботи з кодом, відлагодження та тестування. По-друге, Visual Studio має як платні так і безкоштовні версії,

порівняно з JetBrains Rider, яка має лише безкоштовний пробний період. По-третє, Visual Studio має інтеграцію з Git, що дозволяє контролювати версії, порівняно з SharpDevelop. Таким чином, Visual Studio є кращим вибором для розробки системи для управління персоналом.

Існує багато мов програмування, які використовуються для розробки додатків для персональних комп'ютерів. Кожна мова має свої сильні та слабкі сторони, і її вибір залежить від вимог та цілей проєкту.

Java – це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems. Вона відома своєю переносимістю, безпекою та надійністю. Java використовується для розробки різноманітних програм та додатків, включаючи веб-додатки, мобільні додатки, корпоративні системи, ігри тощо.

C# – це об'єктно-орієнтована мова програмування, розроблена Microsoft. Вона часто використовується для розробки програм на платформі .NET. C# використовується для створення різноманітних програм, включаючи веб-додатки, настільні додатки, мобільні додатки (з використанням платформи Xamarin), ігри (з використанням Unity) тощо [16].

C++ – це мова програмування загального призначення, яка поєднує в собі можливості мови C з об'єктно-орієнтованим програмуванням. C++ використовується для розробки широкого спектру програм, включаючи системне програмування, розробку оперативних систем, програмування мікроконтролерів, графічні додатки, ігри тощо.

Порівняльний аналіз описаних мов програмування наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння мов програмування

	Java	C#	C++
Об'єктно-орієнтована	1	1	1
Синтаксис	1	1	0
Контроль ресурсів	0	1	1

Продовження таблиці 3.2

Широкий набір інструментів	0	1	1
Кросплатформеність	1	1	1
Сумарний коефіцієнт	3	5	4

Виходячи з порівняльних характеристик мов програмування, було вирішено використовувати для розробки власної автоматизованої системи управління персоналом мову програмування C# тому, що вона легша у вивченні порівняно з C++ та пропонує більший список інструментів для розробки додатків для Windows ніж Java.

3.2 Розробка інтерфейсу програмного застосунку

Розробка інтерфейсів користувача відіграє вирішальну роль у успіху будь-якого продукту чи сервісу. Добре спроектований UI робить користування продуктом чи сервісом більш зручним та ефективним, що призводить до ряду важливих переваг:

- інтуїтивний та зрозумілий інтерфейс дозволяє користувачам легко знаходити необхідні функції та виконувати завдання без зайвих зусиль, що збільшує їхню задоволеність від використання продукту;
- задоволені користувачі, які отримують позитивний досвід використання продукту, схильні залишатися лояльними до нього та рекомендувати його іншим;
- інтуїтивний інтерфейс допомагає уникнути плутанини та непорозумінь, що зменшує кількість звернень до служби підтримки користувачів та знижує витрати на підтримку;
- продукти та сервіси з якісним та ефективним UI мають конкурентну перевагу на ринку, оскільки залучають більше користувачів та забезпечують їм приємний досвід використання [17].

Інтерфейс програмного застосунку складається з форми входу, головного меню та вікна створення нового співробітника.

Форма входу користувача до системи складається із двох полів введення даних, кнопки входу. На рисунку 3.1 показано графічну схему форми входу, а на рисунку 3.2 розроблений інтерфейс.

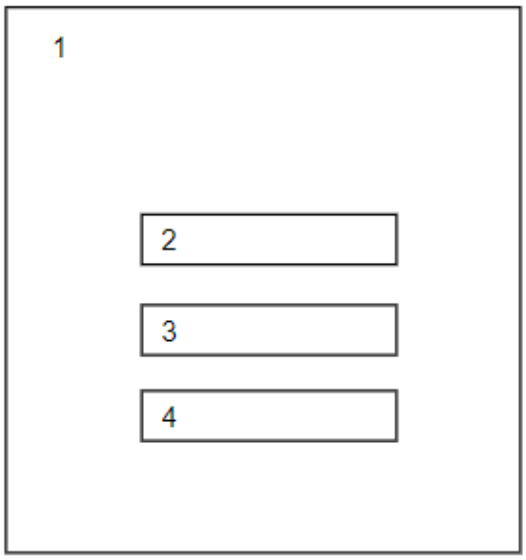


Рисунок 3.1. – Графічна схема вікна входу до системи

Елементи форми входу до системи:

1. Робоча область.
2. Поле введення логіну.
3. Поле введення паролю.
4. Кнопка Вхід.

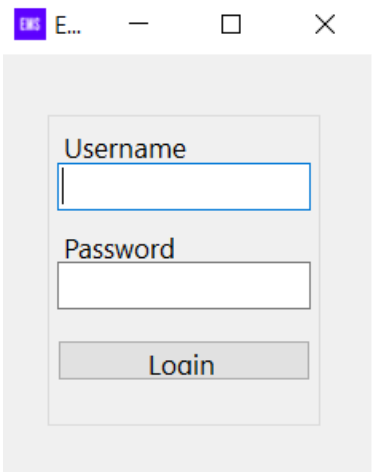


Рисунок 3.2. – Вікно входу до системи

Головне меню адміністратора є вікном, де користувачі мають можливість відобразити список завдань, які зараз виконуються, відобразити дані усіх співробітників, редагувати їх та зареєструвати анкету нового співробітника, створювати та призначати завдання працівникові з вказанням кінцевого терміну виконання. Вікно складається із кнопки для оновлення даних таблиці, таблиці де відображаються всі завдання. Меню швидкого доступу складається з трьох кнопок які відповідають за керування даними працівників, керування завданнями та виходом із програми або зміною користувача. На рисунку 3.3 показано графічну схему головного меню адміністратора, а на рисунку 3.4 розроблений інтерфейс за цією схемою.

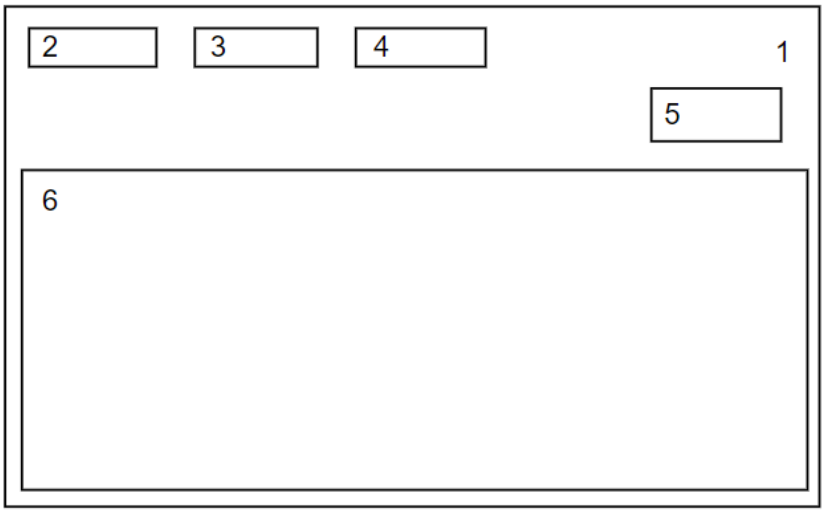


Рисунок 3.3 – Зображення графічної схеми головного меню адміністратора

Елементи головного меню адміністратора:

1. Робоча область.
2. Меню швидкого доступу Керування персоналом.
3. Меню швидкого доступу Керування завданнями.
4. Меню швидкого доступу Зміна користувача/Вихід.
5. Кнопка Оновлення даних таблиці завдань.
6. Таблиця завдань.

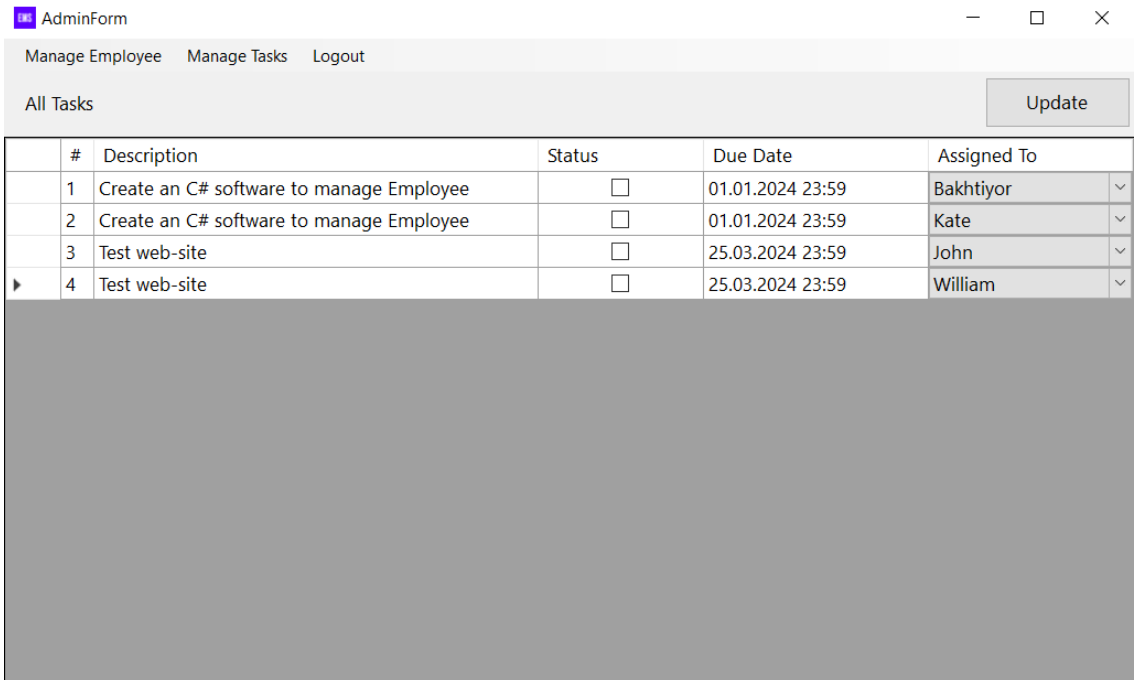


Рисунок 3.4 – Зображення головного меню

Вікно реєстрації нового працівника відкриває модальне вікно, де необхідно заповнити всі поля, після чого підтвердити реєстрацію, що вносить ці дані до бази даних та надає доступ до їх перегляду та редагування. Вікно складається із трьох полів, які необхідно заповнити та кнопка реєстрації. На рисунку 3.5 показано схему вікна реєстрації, а на рисунку 3.6 розроблений інтерфейс.

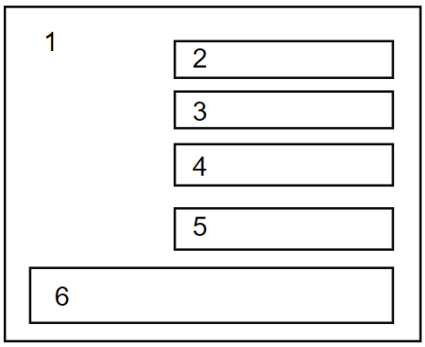


Рисунок 3.5 – Схема вікна реєстрації нового працівника

Елементи вікна реєстрації нового працівника:

1. Робоча область.
2. Поле введення Ім’я.
3. Поле введення Логіну.
4. Поле введення навичок працівника.
5. Поле введення Заробітної плати.
6. Кнопка Створити працівника.

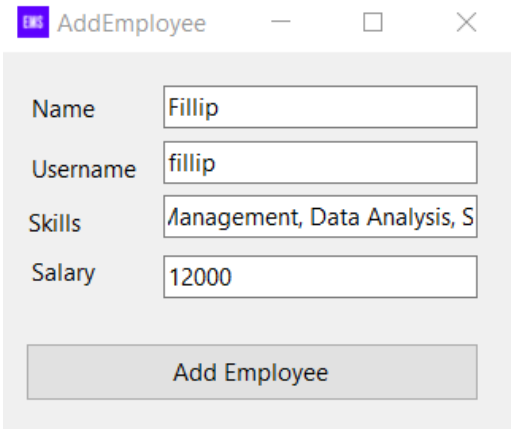


Рисунок 3.6 – Зображення розробленого інтерфейсу

Після натискання кнопки Створення завдання на головному екрані адміністратора відкривається модальне вікно, де необхідно додати опис завдання, вказати кінцевий термін виконання та вказати які навички працівника знадобляться для його виконання, після чого натиснути Додати завдання. На

рисунку 3.7 показано схему вікна створення завдання, а на рисунку 3.8 його графічна реалізація.

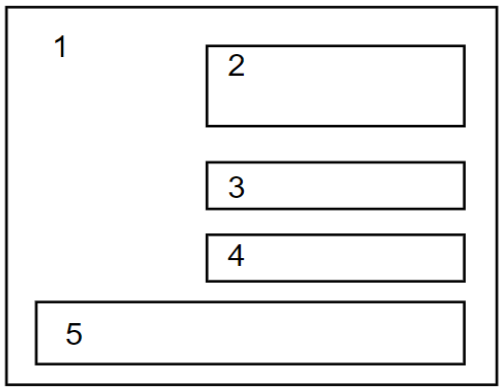


Рисунок 3.7 – Схема вікна створення завдання

Елементи вікна створення нового завдання:

1. Робоча область.
2. Додати опис завдання.
3. Обрати кінцеву дату виконання.
4. Обрати відповідальну особу.
5. Кнопка створити завдання.

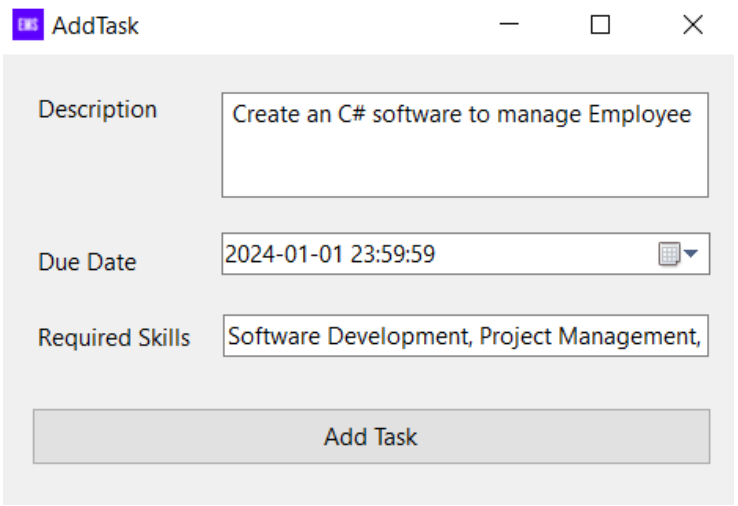


Рисунок 3.8 – Зображення розробленого інтерфейсу

Головний екран працівника це вікно, де співробітники мають можливість переглянути завдання, які їм призначені до виконання, переглянути особистий профіль та змінити свої дані. Головний екран складається із таблиці, яка відображає завдання, двох кнопок на панелі швидкого доступу, які призначені для перегляду свого профілю та його редагування, виходу або зміни користувача. На рисунку 3.9 показано схему головного вікна працівника, а на рисунку 3.10 розроблений інтерфейс.

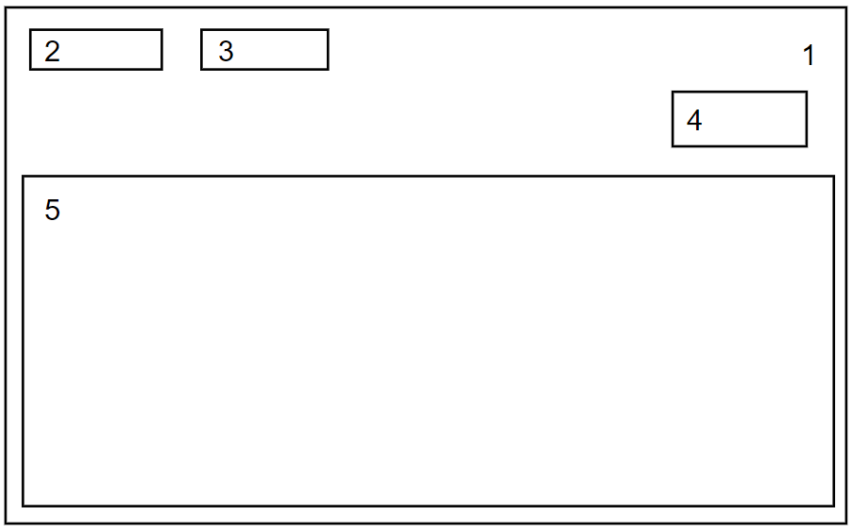


Рисунок 3.9 – Схема головного вікна працівника

Елементи головного вікна працівника:

1. Робоча область.
2. Кнопка Профіль.
3. Кнопка Вихід.
4. Кнопка Оновити дані.
5. Таблиці зі списком призначених завдань.

Рисунок 3.10 – Зображення розробленого інтерфейсу

Вікно редагування особистих даних складається з шести полів: ім'я, логін, навички, старий пароль, новий пароль та підтвердження паролю. Також вікно містить дві кнопки Застосувати зміни та Закрити вікно. На рисунку 2.11 показано схему вікна редагування особистих даних, а на 2.12 розроблений інтерфейс.

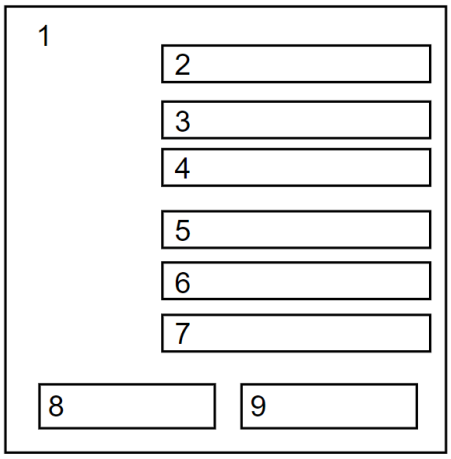


Рисунок 3.11 – Схема вікна редагування особистих даних

Елементи вікна редагування даних:

1. Робоча область.
2. Поле редагування ім’я.
3. Поле редагування логіну.

4. Поле редагування навичок.
5. Поле введення старого пароля.
6. Поле введення нового пароля.
7. Поле Підтвердити пароль.
8. Кнопка Закрити.
9. Кнопка Підтвердити.

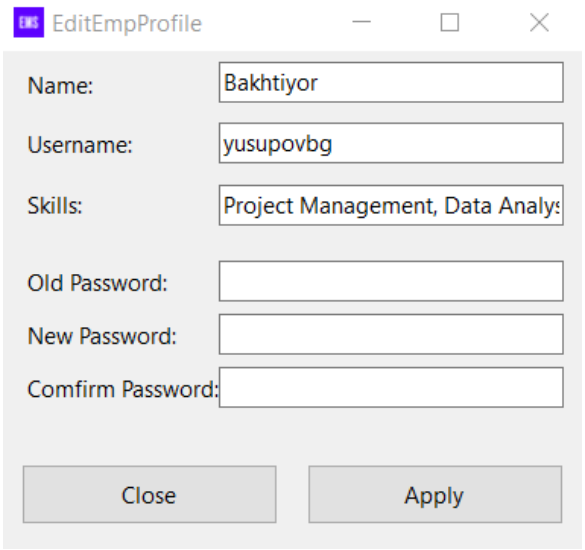


Рисунок 3.12 – Розроблений інтерфейс

Розробка інтерфейсу користувача для автоматизованої системи управління персоналом була проведена у редакторі коду Visual Studio із використанням C# та WinForms.

3.3 Розробка модуля ведення обліку співробітників

Модуль ведення обліку співробітників є однією з найважливіших частин програмного застосунку, адже він дозволяє менеджерам та адміністраторам керувати даними персоналу і слідкувати за виконанням певних завдань. Цей модуль забезпечує ефективне управління ресурсами компанії, підвищує прозорість процесів і дозволяє керівникам приймати обґрунтовані рішення на основі актуальних даних.

На рисунку 3.13 зображено блок схему роботи модуля ведення обліку співробітників.

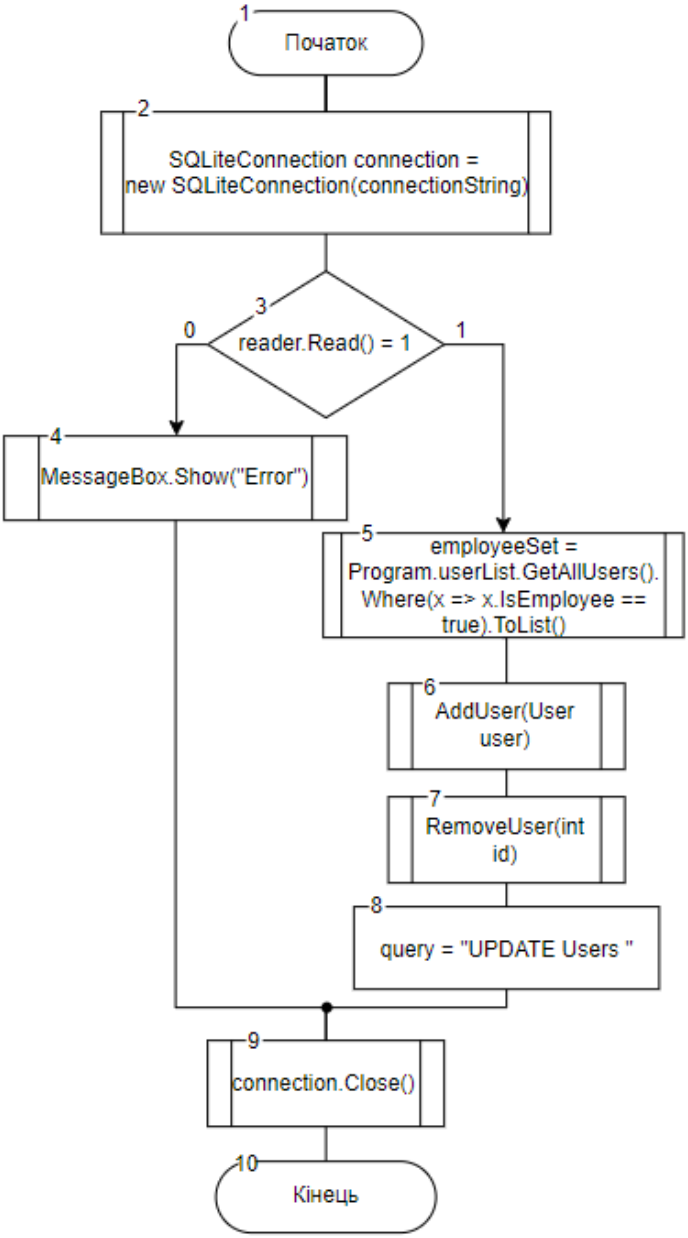


Рисунок 3.13 – Блок схема алгоритму роботи методу ведення обліку працівників

Крок 1. Початок роботи методу UserList.

Крок 2. Виконується підключення до бази даних за допомогою SQLLiteConnection.

Крок 3. За допомогою методу reader.Read() виконується витягування даних із таблиці Employee з бази даних.

Крок 4. Якщо виникає помилка при читанні бази даних, повертається значення «false» та відображається MessageBox.

Крок 5. Якщо було повернено значення «true» інформація про працівників виводиться до DataGridView.

Крок 6. Виконується метод AddUser(), який додає додає дані нового працівника до List<User> userSet за допомогою userSet.Add(user) та генерує новий унікальний номер.

Крок 7. Виконується метод RemoveUsers() де в класі Program об’єкту userList видаляється інформація про працівника.

Крок 8. Виконується оновлення бази даних за допомогою запиту query UPDATE.

Крок 9. Завершується з’єднання з базою даних.

Крок 10. Завершення роботи методу UserList.

Отже, було розроблено модуль ведення обліку співробітників, який зберігає дані співробітників та надає доступ до їх редагування у зручній формі таблиці.

3.4 Розробка модуля створення завдань

Для автоматизованої системи управління персоналом також важливо розробити модуль, який дозволить високопосадовим працівникам створювати завдання, додавати їх опис, вказувати терміни виконання та автоматично призначати їх працівникам. Такий модуль забезпечить ефективну делегацію завдань, прозорість процесів та контроль за виконанням робіт. Крім того, система буде автоматично призначати завдання відповідним працівникам, щоб уникнути перевантаження та оптимізувати розподіл робочого навантаження. Можливість автоматичного призначення завдань допоможе уникнути непорозумінь і перевантаження персоналу, що в свою чергу сприятиме підвищенню продуктивності та мотивації команди.

Модуль створення та автоматизованого призначення завдань було реалізовано за допомогою CRUD операцій з даними в базі даних SQLLite. Це рішення забезпечує простий та швидкий обмін даними, що знаходяться в базі даних. На рисунку 3.14 зображено блок-схему модуля створення та призначення завдань.

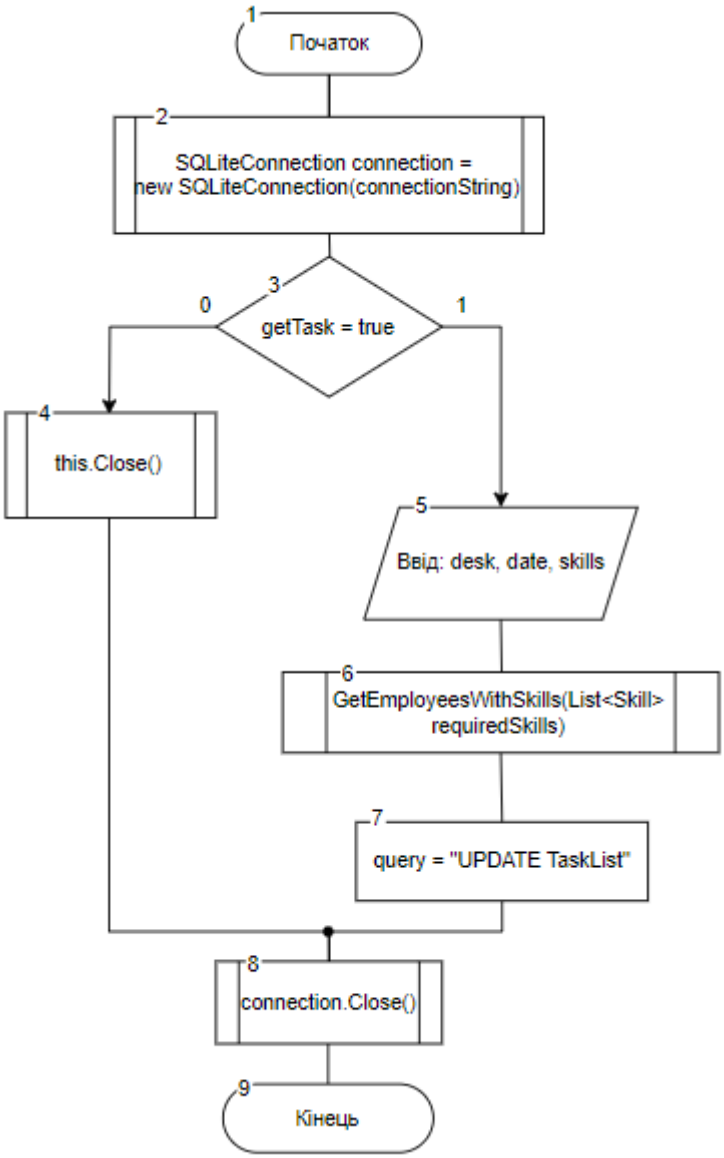


Рисунок 3.14 – Блок-схема алгоритму методу створення завдань

Крок 1. Початок роботи методу TaskList.

Крок 2. Виконується підключення до бази даних за допомогою SQLLiteConnection.

Крок 3. За допомогою методу GetAllTasks() виконується витягування даних із таблиці TaskList з бази даних та перевірка.

Крок 4. Якщо метод getTask повертає значення «false» виконується команда break та завершується робота модуля.

Крок 5. Якщо метод getTask повертає значення «true» ініціалізуються змінні з даних введених користувачем.

Крок 6. Виконується виклик функції GetEmployeesWithSkills(), яка приймає як параметр List<Skill> requiredSkills, що зберігає інформацію про навички та виконує автоматичну вибірку працівників які задовольняють вимоги.

Крок 7. Виконується оновлення бази даних командою query UPDATE.

Крок 8. Завершується з’єднання з базою даних командою connection.Close().

Крок 9. Завершення роботи методу AddTask.

Також для кращого розуміння алгоритму роботи модуля, який відповідає за призначення завдань було створено блок-схему алгоритму функції, яка обирає відповідних працівників базуючись на їхніх навичках (див. рисунок 3.15).

Крок 1. Початок роботи функції GetEmployeesWithSkills().

Крок 2. Відкриття з’єднання з базою даних.

Крок 3. Створення запиту для отримання даних працівників, які мають необхідні навички, використовуючи query SELECT та HAVING SkillCount >= 3, для того щоб обрати лише людей у яких кількість відповідних навичок більша або дорівнює трьом.

Крок 4. Команда виконує запит, а зчитувач reader проходить через результати.

Крок 5. Для кожного рядка результатів створюється об’єкт Employee, який додається до списку працівників.

Крок 6. Метод return повертає список працівників які мають мінімум три навички з введених.

Крок 7. Завершення з'єднання з базою даних методом Close().

Крок 8. Завершення роботи функції GetEmployeesWithSkills().

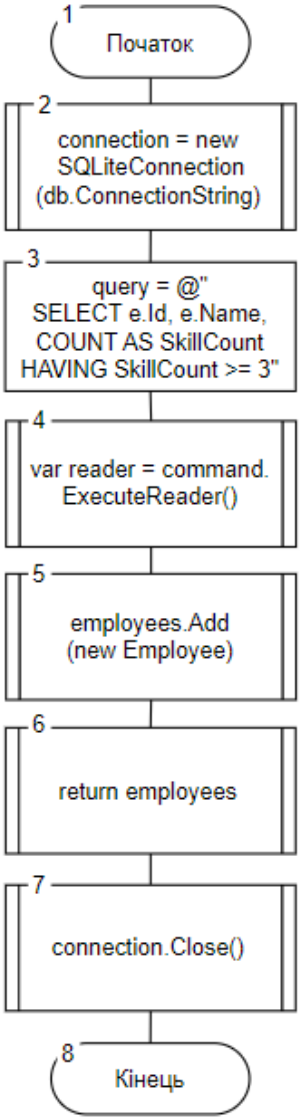


Рисунок 3.15 – Блок-схема алгоритму функції для вибірки працівників

Отже, було розроблено модуль для створення та призначення завдань певному працівнику, який дозволяє в короткі терміни створити нове завдання та зменшує час необхідний для комунікації з персоналом.

3.5 Розробка модуля для видалення завдань

Видалення завдань є не менш важливою складовою застосунку для управління персоналом. Це дозволяє зберігати актуальність і чистоту даних, видаляючи завдання, які більше не потрібні або були виконані. Видалення старих або завершених завдань допомагає запобігти перевантаженню користувачів

зайвою інформацією, що може підвищити продуктивність та ефективність роботи команди.

На рисунку 3.15 зображено блок схему методу для видалення завдань.

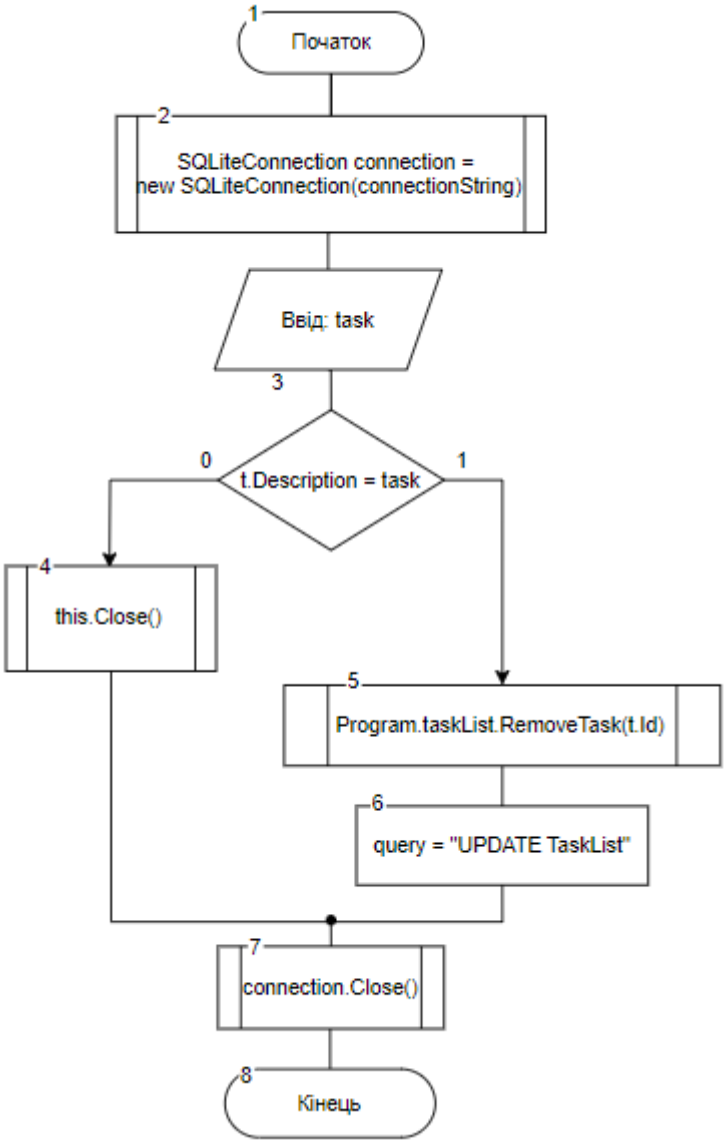


Рисунок 3.15 – Блок-схема алгоритму видалення завдань

Крок 1. Початок роботи методу RemoveTask.

Крок 2. Виконується підключення до бази даних за допомогою SQLLiteConnection.

Крок 3. Ініціалізація змінної task, яка отримує значення виділеного рядка за допомогою властивості SelectedItem, що перетворюється методом ToString() з

об’єкту списку на рядок.

Крок 4. Виконується перевірка для кожного завдання чи його опис збігається зі значенням змінної task та повертає значення «true» або «false».

Крок 5. Якщо повертається значення«false» вікно закривається.

Крок 6. Якщо повертається значення «true», завдання видаляється зі списку за його ідентифікатором Id.

Крок 8. Виконується оновлення бази даних командою query UPDATE.

Крок 9. Завершується з’єднання з базою даних командою connection.Close().

Крок 10. Завершення роботи методу RemoveTask.

Отже, було розроблено модуль для видалення завдань, який дозволяє видаляти виконані або більше не актуальні завдання адміністраторам та сприяти збереженню чистоти даних.

3.5 Висновки

У розділі було обґрунтовано вибір мови програмування для розробки системи та його інтерфейсу. Проаналізувавши популярні технології для розробки програмних систем, було обрано такі технології, як C#, SQLLite та засобів розробки MS Visual Studio. Також було описано розробки модулів для ведення обліку даних співробітників, створення та призначення завдань відповідальній особі з вказанням термінів виконання, та видалення завдань.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Тестування системи

В сучасних умовах конкуренція на ринку зростає, а очікування користувачів стають все вищими. Тому тестування систем стало надзвичайно важливим . Ось декілька причин, які пояснюють необхідність тестування [18].

- Тестування допомагає виявляти та виправляти помилки ще до того, як вони стануть проблемами для користувачів. Це може заощадити час, гроші та запобігти пошкодженню репутації.
- Тестування допомагає гарантувати, що системи відповідають очікуванням користувачів і вимогам бізнесу. Це може підвищити задоволення користувачів, їх лояльність та прибутковість.
- Тестування допомагає виявляти недоліки та інші неефективності в системах, що може призвести до підвищення продуктивності та зниження витрат.
- Тестування допомагає гарантувати, що системи можуть бути легко модифіковані та розширені. Це дозволяє компаніям пристосовуватися до змінних потреб ринку.
- Тестування допомагає виявляти та усувати вразливості в системах, що може захистити дані користувачів та активи компанії.

Щодо типів тестування, є декілька підходів, таких як функціональне, продуктивне, надійне, безпечне та тестування на юзабіліті. Кожен з цих підходів важливий для забезпечення якості та ефективності системи.

- Функціональне тестування. Перевірка функціональних вимог програми, тобто тестування функціональності .
- Нефункціональне тестування. Тестування, що не стосується конкретної функціональності програми, але важливе для забезпечення якості, наприклад, тестування продуктивності, надійності, безпеки тощо.

- Модульне тестування. Тестування окремих модулів або компонентів програми.

- Інтеграційне тестування. Тестування взаємодії між різними компонентами або модулями програми.

- Системне тестування. Перевірка системи як цілісності, включаючи всі її компоненти та їх взаємодію.

- Приймальне тестування. Тестування з метою підтвердження відповідності програми вимогам замовника або стандартам якості.

Під час розробки комп'ютерної системи для автоматизації процесів управління персоналом було створено класи та функції для проведення юніт-тестування, використовуючи бібліотеку Nunit.

NUnit — це платформа модульного тестування для всіх мов .NET. Ця платформа з відкритим кодом спочатку була перенесена з JUnit [19].

Фреймворк включає кілька методів підтвердження, щоб переконатися, що код, що тестується, працює належним чином. Він також має низку атрибутів, які можна використовувати для керування виконанням тесту та налаштування та демонтажу тестового середовища. NUnit є популярним інструментом серед розробників, щоб гарантувати, що їхній код є високою якістю та без дефектів.

Ці класи слугують для перевірки та валідації електронної пошти та паролю під час виконання входу до системи. Діаграма класів зображена на рисунку 4.1.

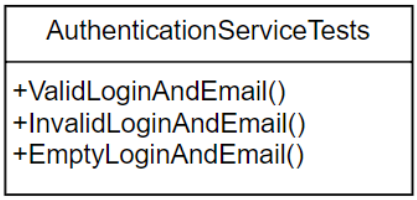


Рисунок 4.1 – Діаграма класу AuthenticationServiceTests

У класі тестування використовується створена тестова база даних яка заповнена тестовими даними для входу користувачів.

Основним методом для перевірки провильності вводу даних є метод ValidLoginAndEmail(). Блок-схема цього методу зображена на рисунку 4.2.

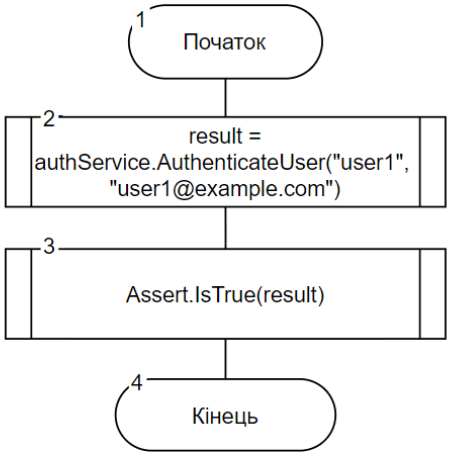


Рисунок 4.2 – Метод перевірки електронної пошти за шаблоном

1. Виклик методу тестування ValidLoginAndEmail().
2. Ініціалізація змінної використовуючи правильні дані для входу, які збережені у тестовій базі даних.
3. Повертає значення «true» або «false», відповідно до того, чи відповідають введені дані правильним, використовуючи метод Assert.
4. Вихід із методу ValidLoginAndEmail() – перевіряє значення паролю та .

Для проведення юніт-тестування на введення неправильних даних користувачем було створено метод InvalidLoginAndEmail() (рисунок 4.3).

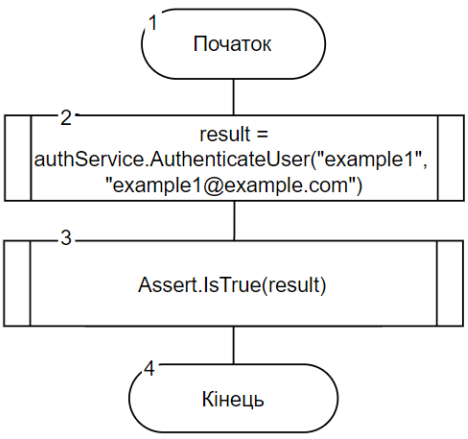


Рисунок 4.3 – Блок схема перевірки методу введення неправильних даних

1. Виклик методу InvalidLoginAndEmail().
2. Ініціалізація змінної result та збереження введених даних.
3. Використовується метод Assert.isTrue(), який приймає аргументи або методи, які повинні повернути «true». Якщо при перевірці було повернено значення «false» тест вважається непройденим.
4. Вихід із методу InvalidLoginAndEmail().

Для проведення юніт-тестування на наявність незаповнених полів користувачем було створено метод EmptyLoginAndEmail(), що перевіряє значення email і password. Блок-схема цього методу зображена на рисунку 4.4.

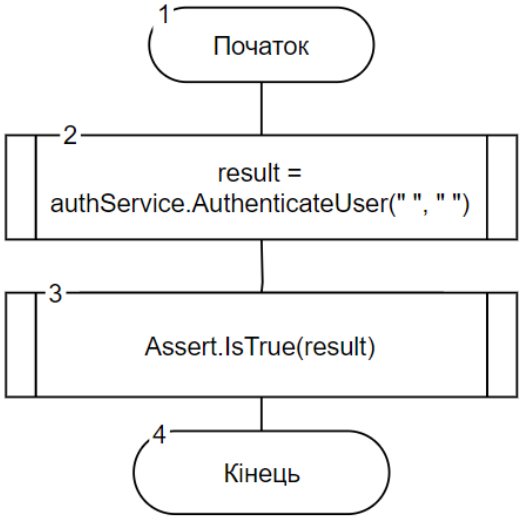


Рисунок 4.4 – Блок схема алгоритму методу EmptyLoginAndEmail()

1. Виклик методу тестування EmptyLoginAndEmail().
2. Ініціалізація змінної result використовуючи пусті поля.
3. Повертає значення «true» або «false», відповідно до того, чи відповідають введені дані правильним, використовуючи метод Assert.
4. Завершення методу EmptyLoginAndEmail().

У класі тестування для додавання і видалення завдань використовується два шаблони, які перевіряють створення завдання та його видалення. Діаграма класу для тестування керування завданнями зображена на рисунку 4.5.

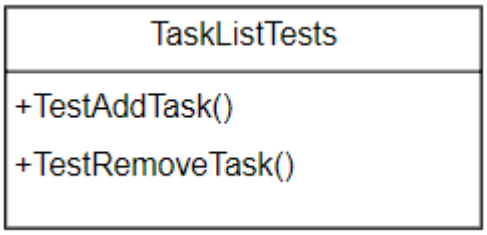


Рисунок 4.5 – Діаграма класу TaskListTest

Для проведення базової перевірки модуля створення завдань було створено метод TestAddTask(). Блок-схема алгоритму цього методу показана на рисунку 4.6.

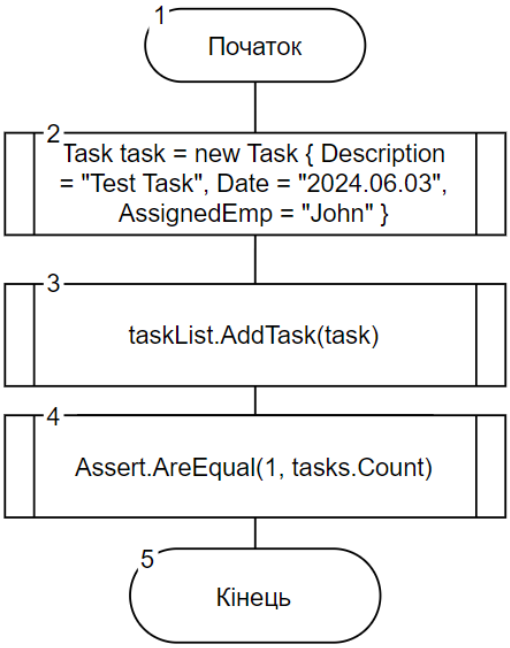


Рисунок 4.6 – Блок-схема алгоритму методу TestAddTask()

1. Виклик методу тестування TestAddTask().
2. Ініціалізується змінна task з присвоєними значеннями опису, дати та призначеного працівника.
3. Використовується метод taskList.AddTask() для запису до таблиці.
4. Якщо запис до таблиці був успішним повертається значення «true», що вказує на успішне проходження тесту.
5. Завершення методу TestAddTask().

Для перевірки видалення завдань було створено метод тестування TestRemoveTask(). Блок-схема алгоритму методу показана на рисунку 4.7.

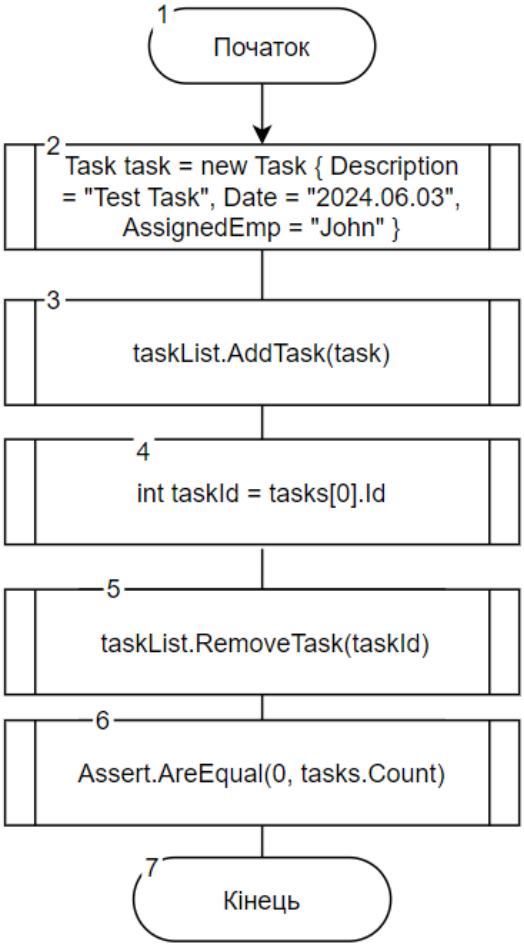


Рисунок 4.6 – Блок-схема алгоритму методу TestRemoveTask()

1. Виклик методу тестування TestRemoveTask().
2. Ініціалізується змінна task з присвоєними значеннями опису, дати та призначеного працівника.
3. Використовується метод taskList.AddTask() для запису до таблиці.
4. Ініціалізується змінна Id, як містить унікальний номер завдання.
5. Використовується метод taskList.RemoveTask(), де в якості аргументів передається змінна, яка містить унікальний номер завдання.
6. Виконується перевірка, що метод RemoveTask() успішно видаляє завдання зі списку за його ідентифікатором та повертає значення «true».
7. Завершення методу TestRemoveTask().

4.2 Розробка інструкції користувача

Інструкція користувача (ІК) є важливим компонентом будь-якого продукту, оскільки вона допомагає користувачам зрозуміти, як використовувати продукт ефективно та безпечно [20].

Розробка інструкції користувача є важливою з кількох причин:

1. Спрощує використання продукту: інструкція користувача допомагає новим користувачам швидше оволодіти продуктом і використовувати його ефективно. Вона надає докладні пояснення щодо функцій, можливостей та правильного використання програми.

2. Мінімізує помилки і непорозуміння: наявність чітких інструкцій дозволяє уникнути помилок та непорозумінь під час використання продукту. Користувачі можуть дотримуватися кроків інструкції, щоб досягти бажаного результату без неправильних кроків.

3. Забезпечує однорідне використання: інструкція користувача стандартизує процес використання продукту, що дозволяє користувачам виконувати одні й ті ж дії в однаковий спосіб. Це допомагає уникнути ситуацій, коли користувачі використовують продукт по-різному і отримують різні результати.

4. Покращує задоволення від використання: якщо користувачі можуть швидко і легко зрозуміти, як використовувати продукт, це збільшує їх задоволення від використання його. Інструкція користувача допомагає знизити рівень стресу та невпевненості, пов'язаних з використанням нового програмного забезпечення.

5. Зменшує потребу в підтримці: чіткі інструкції користувача можуть зменшити кількість запитів до служби підтримки, оскільки користувачі можуть самостійно знайти відповіді на свої запитання у документації.

Для початку роботи із системою користувачеві пропонується увійти у свій особистий кабінет, де в залежності від його ролі потрапляє у своє головне вікно. На рисунку 4.1 зображено форму входу до системи.

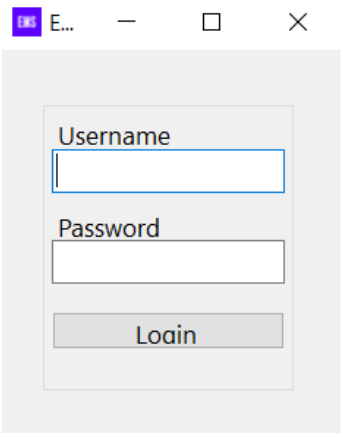


Рисунок 4.1 – Вікно авторизації

Після виконання входу користувач потрапляє у головне меню (див. рисунок 4.1), де може виконувати необхідні функції управління, якщо вхід відбувся з акаунту адміністратора.

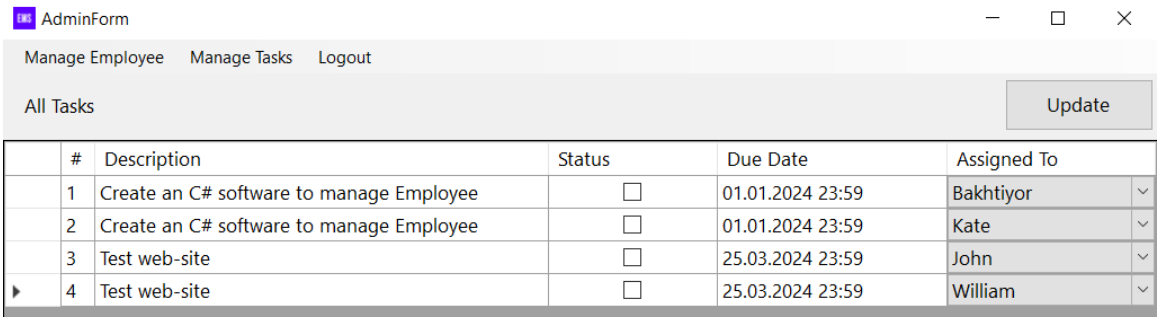


Рисунок 4.2 – Зображення інтерфейсу користувача адміністратора

Увесь необхідний функціонал знаходиться у вікні «AdminForm», що дозволяє виконувати основні функції управління. Оновлення даних таблиці активних завдань актуальною інформацією про поточні завдання натискаючи кнопку «Update» на головному вікні;

Для перегляду інформації про працівників необхідно на панелі швидкого доступу натиснути кнопку «ManageEmployee» => «Employees», що дозволить переглядати та редагувати інформацію про працівників (див. рисунок 4.3).

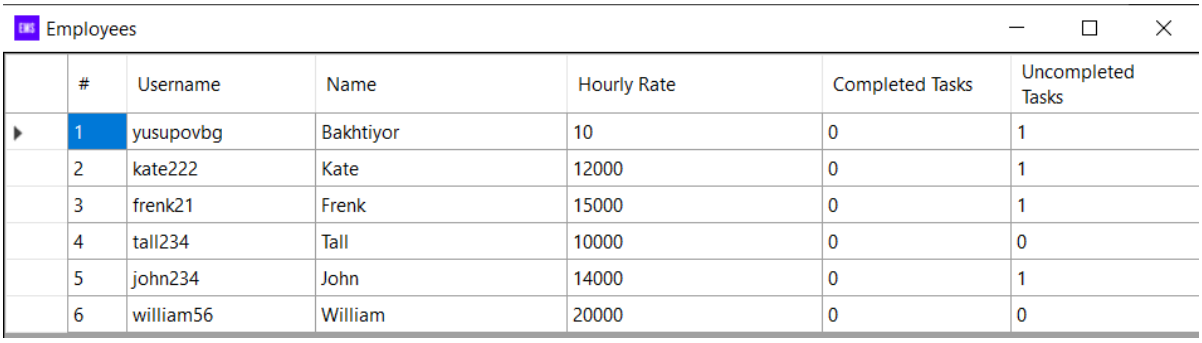


Рисунок 4.3 – Вікно «Employees»

Якщо необхідно створити анкету нового працівника, на головному на головному екрані потрібно натиснути кнопку «ManageEmployee» => «Add Employee», що відкриє нове вікно, де необхідно заповнити всі поля. Вікно створення нового працівника зображено на рисунку 4.4.

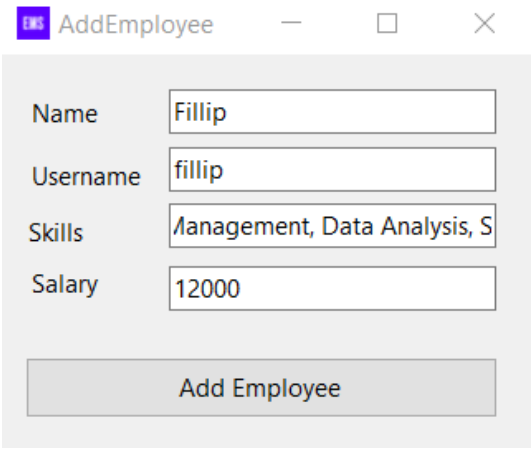


Рисунок 4.4 – Зображення вікна «Add Employee»

Для того щоб видалити співробітника потрібно натиснути кнопку «ManageEmployee» => «Remove Employee» на панелі швидкого доступу головного вікна. У вікні, яке відкрилось обрати необхідне ім’я та натиснути «Remove» (див. рисунок 4.5).

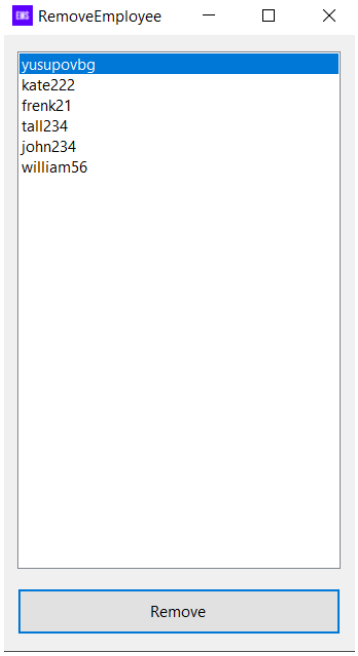


Рисунок 4.5 – Вікно «Remove Employee»

Якщо натиснути кнопку «Manage Tasks» на панелі швидкого доступу головного вікна, відкриється нове вікно де можна створити нове завдання. Необхідно заповнити поле опису, обрати час виконання завдання та вказати необхідні навички для виконання завдання (див. рисунок 4.6).

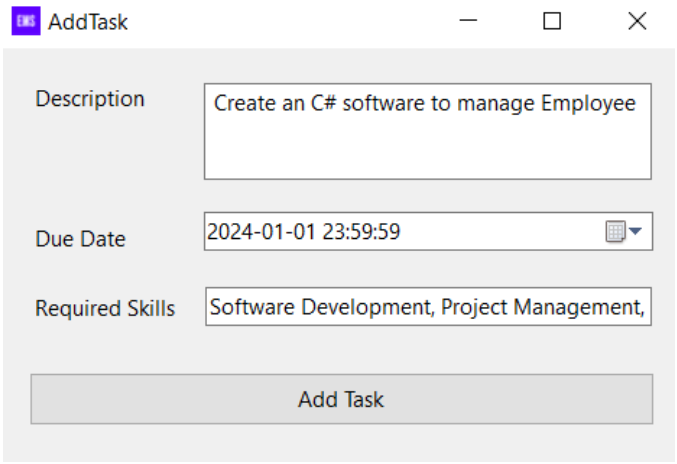


Рисунок 4.6 – Вікно «Add Task»

Після натискання кнопки «Add Task» з’явиться вікно яке інформує про те, що завдання було успішно створено та призначено відповідним працівникам, які задовольняють вимогам (див. рисунок 4.7).

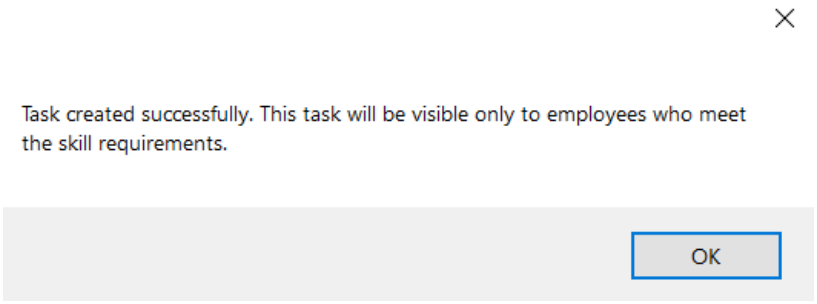


Рисунок 4.7 – Вікно інформування

Якщо вхід до системи відбувся звичайним працівником він потрапляє у своє головне меню, де на початковому екрані відображується таблиця із списком призначених йому завдань, які він може відмітити як виконане, кнопка «Update» для оновлення відомостей та кнопка «Profile», яка знаходиться на панелі швидкого доступу (див. рисунок 4.8) Також працівник може бачити коли адміністратор підтвердить виконання завдання, що буде відображуватись у полі «Finish Time».

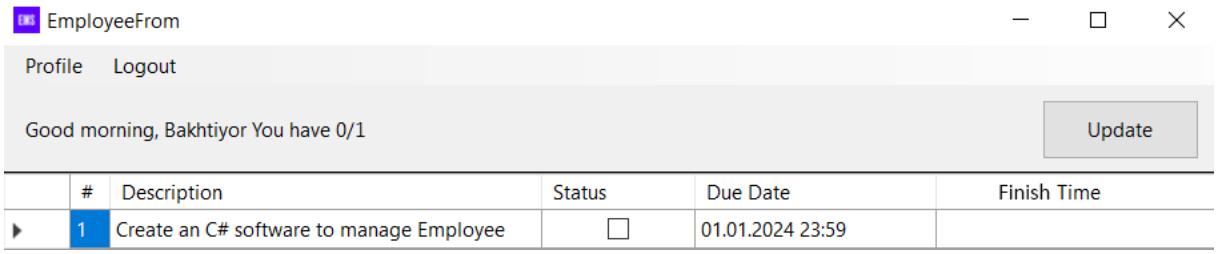


Рисунок 4.8 – Вікно «Employee Form»

Працівник може переглянути свій профіль, де буде вказано його відпрацьований час та особисті дані. Для цього потрібно натиснути кнопку «Profile» та обрати пункт меню «View my profile» (див. рисунок 4.9).

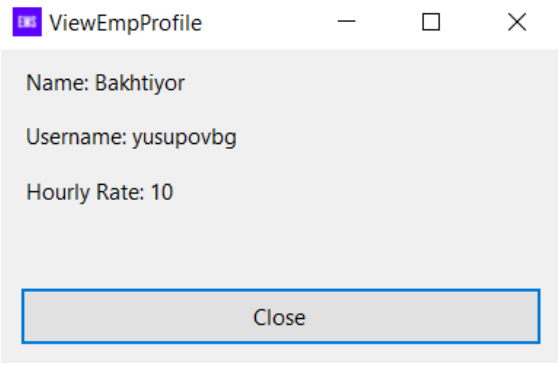


Рисунок 4.9 – Вікно «View Employee Profile»

Для редагування свого профілю користувачеві необхідно натиснути кнопку «Profile» та в меню обрати «Edit profile». Користувач може змінити ім’я, логін, пароль та вказати свої навички. Вікно редагування особистих даних зображено на рисунку 4.10.

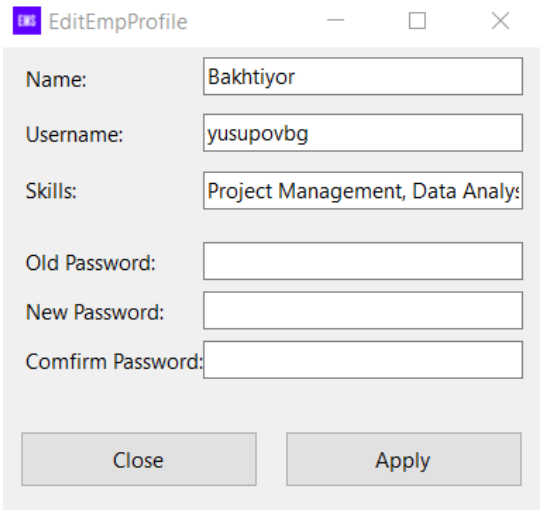


Рисунок 4.10 – Вікно «Edit Employee Profile»

Для виходу із свого облікового запису необхідно використовувати кнопку «Logout» яка є на панелі швидкого доступу у головному вікні як адміністратора, так і працівника.

4.3 Висновки

Під час тестування системи ми провели ретельний аналіз функціональності та коректності роботи програми на десктопній платформі. Виявлені та виправлені помилки дозволили забезпечити надійну та стабільну роботу додатку.

У результаті успішного проходження тестів на продуктивність було підтверджено, що програма працює швидко та ефективно, не надмірно використовуючи ресурси пристрою.

Крім того, було розроблено інструкцію користувача застосунку, яка допоможе правильно його використовувати.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмну систему для автоматизації процесів управління персоналом на базі операційної системи Windows. Розроблена програмна система призначена для удосконалення процесу керування даними співробітників та керуванням виконання завдань, що сприятиме підвищенню конкурентоспроможності серед подібних рішень.

Розглянуто стан питання автоматизації процесів управління персоналом та методи розв’язання задачі, проведено порівняльний аналіз аналогів. Також окреслено напрямки удосконалення подібних систем та підходи до організації процесу керування.

Розроблено модель даних та структуру програмної системи. Розроблено загальний алгоритм роботи застосунку; удосконалено метод створення завдань та вибору виконавців, що на відміну від існуючого, використовує регулярні вирази з ваговими коефіцієнтами до професійних навиків, для покращення відповідності виконавців задачам, і як наслідок підвищення продуктивності і якості виконаних робіт.

Розглянуто переваги й недоліки методів програмування та засобів розробки. На основі аналізу для розробки обрано середовище MS Visual Studio та мову програмування C#. Описано процес розробки основних модулів програмної системи: інтерфейс користувача, структуру даних, модуля створення завдань та призначення їх виконавцеві та відстеження статистики виконаних завдань.

Розроблено методику і проведено тестування програмної системи, доведено доцільність створення модульних тестів для перевірки працездатності розробленої програмної системи. Було створено ряд тест-кейсів для тестування реакції програми у відповідь на введення некоректних даних. За результатами тестування проблем не виявлено. Також було розроблено інструкцію користувача застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лозан О. В. Проблеми та перспективи використання програмних застосунків у бізнес-середовищі / О. В. Лозан , О. М. Рейда // Матеріали LIII Науково-технічної конференції Вінницького національного технічного університету, Вінниця, 2024, [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/author/submission/21144

2. PeopleHR [Електронний ресурс] – Режим доступу до ресурсу: https://peopleforce.io/uk/products/peoplehr

3. HiBob [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/thzfj

4. WebWork [Електронний ресурс] – Режим доступу до ресурсу: https://www.webwork-tracker.com/productivity-monitoring-software

5. BitImpulse – Що таке система аналізу даних? [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tiahm

6. Telegraph – Що таке machine learning? [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tiakr

7. SixSigma це? [Електронний ресурс]. – Режим доступу: https://worksection.com/ua/blog/six-sigma.html

8. Стратегії організаційних змін та їх успішна реалізація [Електронний ресурс]. – Режим доступу: https://britishmba.in.ua/stratehii-orhanizatsiinykh-zmin-ta-ikh-uspishna-realizatsiia/

9. Монолітна архітектура ПЗ [Електронний ресурс]. – Режим доступу: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/

10. Клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

11. Мікросервісна архітектура [Електронний ресурс]. – Режим доступу: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/

12. Як будувати UML діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/

13. What is SQLLite [Електронний ресурс]. – Режим доступу до ресурсу: http://surl.li/twrjr

14. Проектування програмного забезпечення засобами UML [Електронний ресурс]. – Режим доступу до ресурсу: http://surl.li/twrjy

15. Visual Studio 2022 [Електронний ресурс]. – Режим доступу до ресурсу: http://surl.li/twrkg

16. C# Programming [Електронний ресурс]. – Режим доступу до ресурсу: https://www.coursera.org/articles/c-sharp

17. Що таке UX/UI дизайн. [Електронний ресурс] – Режим доступу до ресурсу: https://redstone.media/shcho-take-ux-ui-dysayn

18. Що таке тестування ПЗ? [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/

19. Nunit [Електронний ресурс] – Режим доступу до ресурсу: https://nunit.org/

20. Інструкція користувача: User Manual [Електронний ресурс] – Режим доступу до ресурсу: https://qatestlab.com/resources/knowledge-center/sample-deliverables/user-manuals/

ДОДАТКИ

62

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
к.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмної системи для автоматизації процесів управління персоналом» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доцент О. М. Рейда
«12» березня 2024 р.
Виконав:
студент гр. ЗПІ-20б О. В. Лозан
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмної системи для автоматизації процесів управління персоналом».

Галузь застосування – керування персоналом.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою роботи є вдосконалення та автоматизація процесів управління персоналом за рахунок впровадження сучасних технологій та шляхом розробки автоматизованої системи, що сприятиме підвищенню ефективності роботи, зменшенню необхідного часу для ведення обліку співробітників та моніторингу виконання завдань.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. А.М. Пєтух, О.В. Романюк, О.Н. Романюк. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних. Вінниця : ВНТУ, 2016. 97 с.

2. O'Regan Gerard. Concise Guide to Software Testing. Хам, Швейцарія : Springer International Publishing, 2019. 293 с

5. Технічні вимоги

Тип програми – комп’ютеринй застосунок; модель розробки каскадна; засоби організаціїї даних програми – SQLLite; вхідні дані: персональні дані входу користувачів, особи; вихідні дані: відображення даних про персонал, завдання; засоби розробки – MS Visual Studio; мова програмування – C#.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку автоматизованих систем управління персоналом та постановка задач дослідження	13.03.24 –22.03.24
2	Розробка алгоритмів програмного продукту	13.03.24 –22.03.24
3	Розробка автоматизованої системи управління персоналом	25.03.24 –19.04.24
4	Тестування програмного забезпечення	22.04.24 –17.05.24
5	Оформлення матеріалів до захисту БДР	20.05.24 – 03.06.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмної системи для автоматизації процесів управління персоналом.

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Рейда О. М.

Оригінальність	88%
Схожість	12%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Лозан О. В.

Керівник роботи Рейда О. М.

Додаток В – Лістинг програми

```
// Program.cpp
#include "LoginForm.h"
#include <iostream>
#include "MainForm.h"
#include "RegisterForm.h"
#include "Worker.h"
#include "User.h"
using namespace System;
using namespace System::Windows::Forms;
void main(array<String^>^ args)
{
Application::EnableVisualStyles();
Application::SetCompatibleTextRenderingDefault(false);
ManagePersonal::LoginForm loginform;
loginform.ShowDialog();
User^ user = loginform.user;
if (user != nullptr) {

        // If authentication succeeded, create an instance of MainForm
        ManagePersonal::MainForm mainForm(user);

        // Run the MainForm
        Application::Run(% mainForm);
                                }
else {
        MessageBox::Show("Authentication cancelled", "Authentication
Cancelled", MessageBoxButtons::OK);
```

```
}
}
```

\\Worker.h

```
#pragma once
using namespace System;
public ref class Worker {
public:
    int id;
    String^ name;
    String^ email;
    String^ phone;
    String^ adress;
    String^ position;
    String^ password;
    property bool IsWorker;
public:
        Worker() {
        IsWorker = true;
    }
    property int Id {
        int get() {
            return id;
        }
        void set(int value) {
            id = value;
        }
    }
    property String^ Name {
        String^ get() {
            return name;
```

```
    }
    void set(String^ value) {
        name = value;
    }
}
property String^ Email {
    String^ get() {
        return email;
    }
    void set(String^ value) {
        email = value;
    }
}
property String^ Phone {
    String^ get() {
        return phone;
    }
    void set(String^ value) {
        phone = value;
    }
}
property String^ Address {
    String^ get() {
        return adress;
    }
    void set(String^ value) {
        adress = value;
    }
}
property String^ Position {
```

```
        String^ get() {
            return position;
        }
        void set(String^ value) {
            position = value;
        }
    }
    property String^ Password {
        String^ get() {
            return password;
        }
        void set(String^ value) {
            password = value;
        }
    }
};

\\RegisterForm.h
#pragma once
#include "LoginForm.h"
#include "User.h"
#include "Worker.h"
namespace ManagePersonal {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
```

```
/// <summary>
/// Summary for RegisterForm
/// </summary>
public ref class RegisterForm : public System::Windows::Forms::Form
{
public:
	RegisterForm(void)
	{
		InitializeComponent();
		//
		//TODO: Add the constructor code here
		//
	}
protected:
	/// <summary>
	/// Clean up any resources being used.
	/// </summary>
	~RegisterForm()
	{
		if (components)
		{
			delete components;
		}
	}
private: System::Windows::Forms::Label^ label1;
protected:
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Label^ label5;
```

```
private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::Label^ label8;
private: System::Windows::Forms::TextBox^ tbConfirmPassword;
private: System::Windows::Forms::TextBox^ tbPassword;
private: System::Windows::Forms::TextBox^ tbPosition;
private: System::Windows::Forms::TextBox^ tbAdress;
private: System::Windows::Forms::TextBox^ tbPhone;
private: System::Windows::Forms::TextBox^ tbEmail;
private: System::Windows::Forms::TextBox^ tbName;
private: System::Windows::Forms::Button^ btnRegister;
private: System::Windows::Forms::Button^ btnCancel;
private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        void InitializeComponent(void)
        {
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->label5 = (gcnew System::Windows::Forms::Label());
                this->label6 = (gcnew System::Windows::Forms::Label());
```

```
this->label7 = (gcnew System::Windows::Forms::Label());
this->label8 = (gcnew System::Windows::Forms::Label());
this->tbConfirmPassword = (gcnew
System::Windows::Forms::TextBox());
this->tbPassword = (gcnew System::Windows::Forms::TextBox());
this->tbPosition = (gcnew System::Windows::Forms::TextBox());
this->tbAdress = (gcnew System::Windows::Forms::TextBox());
this->tbPhone = (gcnew System::Windows::Forms::TextBox());
this->tbEmail = (gcnew System::Windows::Forms::TextBox());
this->tbName = (gcnew System::Windows::Forms::TextBox());
this->btnRegister = (gcnew System::Windows::Forms::Button());
this->btnCancel = (gcnew System::Windows::Forms::Button());
this->SuspendLayout();
//
// label1
//
this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 16.2F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label1->Location = System::Drawing::Point(12, 22);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(985, 71);
this->label1->TabIndex = 0;
this->label1->Text = L"Реєстрація нового працівника";
this->label1->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;
//
// label2
//
```

```
this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label2->Location = System::Drawing::Point(61, 132);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(207, 39);
this->label2->TabIndex = 1;
this->label2->Text = L"Ім\'я";
this->label2->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label3
//
this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label3->Location = System::Drawing::Point(61, 171);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(207, 39);
this->label3->TabIndex = 2;
this->label3->Text = L"Ел. адреса";
this->label3->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label4
//
```

```
this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label4->Location = System::Drawing::Point(61, 210);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(207, 39);
this->label4->TabIndex = 3;
this->label4->Text = L"Телефон";
this->label4->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label5
//
this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label5->Location = System::Drawing::Point(61, 249);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(207, 39);
this->label5->TabIndex = 4;
this->label5->Text = L"Адреса";
this->label5->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label6
//
```

```
this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label6->Location = System::Drawing::Point(61, 288);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(207, 39);
this->label6->TabIndex = 5;
this->label6->Text = L"Посада";
this->label6->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label7
//
this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 13.8F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->label7->Location = System::Drawing::Point(61, 327);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(207, 39);
this->label7->TabIndex = 6;
this->label7->Text = L"Пароль";
this->label7->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
//
// label8
//
```

```
this->label8->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 13.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
this->label8->Location = System::Drawing::Point(61, 366);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(304, 39);
this->label8->TabIndex = 7;
this->label8->Text = L"Повторити пароль";
this->label8->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
//
// tbConfirmPassword
//
this->tbConfirmPassword->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
this->tbConfirmPassword->Location = System::Drawing::Point(305, 370);
this->tbConfirmPassword->Name = L"tbConfirmPassword";
this->tbConfirmPassword->Size = System::Drawing::Size(614, 34);
this->tbConfirmPassword->TabIndex = 8;
//
// tbPassword
//
this->tbPassword->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
```

```
                static_cast<System::Byte>(204)));
            this->tbPassword->Location = System::Drawing::Point(305, 327);
            this->tbPassword->Name = L"tbPassword";
            this->tbPassword->Size = System::Drawing::Size(614, 34);
            this->tbPassword->TabIndex = 9;
            //
            // tbPosition
            //
            this->tbPosition->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->tbPosition->Location = System::Drawing::Point(305, 288);
            this->tbPosition->Name = L"tbPosition";
            this->tbPosition->Size = System::Drawing::Size(614, 34);
            this->tbPosition->TabIndex = 10;
            //
            // tbAdress
            //
            this->tbAdress->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->tbAdress->Location = System::Drawing::Point(305, 248);
            this->tbAdress->Name = L"tbAdress";
            this->tbAdress->Size = System::Drawing::Size(614, 34);
            this->tbAdress->TabIndex = 11;
            //
            // tbPhone
            //
```

```
this->tbPhone->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));
this->tbPhone->Location = System::Drawing::Point(305, 210);
this->tbPhone->Name = L"tbPhone";
this->tbPhone->Size = System::Drawing::Size(614, 34);
this->tbPhone->TabIndex = 12;
//
// tbEmail
//
this->tbEmail->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));
this->tbEmail->Location = System::Drawing::Point(305, 168);
this->tbEmail->Name = L"tbEmail";
this->tbEmail->Size = System::Drawing::Size(614, 34);
this->tbEmail->TabIndex = 13;
//
// tbName
//
this->tbName->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));
this->tbName->Location = System::Drawing::Point(305, 128);
this->tbName->Name = L"tbName";
this->tbName->Size = System::Drawing::Size(614, 34);
this->tbName->TabIndex = 14;
```

```
            //
            // btnRegister
            //
            this->btnRegister->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 13.8F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->btnRegister->Location = System::Drawing::Point(464, 482);
            this->btnRegister->Name = L"btnRegister";
            this->btnRegister->Size = System::Drawing::Size(217, 50);
            this->btnRegister->TabIndex = 15;
            this->btnRegister->Text = L"Зареєструвати";
            this->btnRegister->UseVisualStyleBackColor = true;
            this->btnRegister->Click += gcnew System::EventHandler(this,
&RegisterForm::btnRegister_Click);
            //
            // btnCancel
            //
            this->btnCancel->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 13.8F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->btnCancel->Location = System::Drawing::Point(738, 482);
            this->btnCancel->Name = L"btnCancel";
            this->btnCancel->Size = System::Drawing::Size(217, 50);
            this->btnCancel->TabIndex = 16;
            this->btnCancel->Text = L"Закрити";
            this->btnCancel->UseVisualStyleBackColor = true;
            this->btnCancel->Click += gcnew System::EventHandler(this,
&RegisterForm::btnCancel_Click);
```

```
            //
            // RegisterForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
            this->BackColor = System::Drawing::SystemColors::ButtonFace;
            this->ClientSize = System::Drawing::Size(1009, 610);
            this->Controls->Add(this->btnCancel);
            this->Controls->Add(this->btnRegister);
            this->Controls->Add(this->tbName);
            this->Controls->Add(this->tbEmail);
            this->Controls->Add(this->tbPhone);
            this->Controls->Add(this->tbAdress);
            this->Controls->Add(this->tbPosition);
            this->Controls->Add(this->tbPassword);
            this->Controls->Add(this->tbConfirmPassword);
            this->Controls->Add(this->label8);
            this->Controls->Add(this->label7);
            this->Controls->Add(this->label6);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Name = L"RegisterForm";
            this->Text = L"RegisterForm";
            this->Load += gcnew System::EventHandler(this,
&RegisterForm::RegisterForm_Load);
            this->ResumeLayout(false);
```

```
            this->PerformLayout();
        }
    #pragma endregion
    private: System::Void RegisterForm_Load(System::Object^ sender,
System::EventArgs^ e) {
    }
    public: User^ user = nullptr;
    public: bool switchToMain = false;
    private: System::Void linkLabel1_LinkClicked(System::Object^ sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {
        }
    private: System::Void btnCancel_Click(System::Object^ sender,
System::EventArgs^ e) {
        this->Close();
    }
    public: Worker^ worker = nullptr;
    private: System::Void btnRegister_Click(System::Object^ sender,
System::EventArgs^ e) {
        String^ name = tbName->Text;
        String^ email = tbEmail->Text;
        String^ phone = tbPhone->Text;
        String^ adress = tbAdress->Text;
        String^ position = tbPosition->Text;
        String^ password = tbPassword->Text;
        String^ confirmPassword = tbConfirmPassword->Text;
        if (email->Length == 0 || password->Length == 0 || name->Length == 0 ||
            phone->Length ==0 || adress->Length == 0 || position->Length ==
0 ||
            password->Length == 0 || confirmPassword->Length == 0)
        {
```

```
                MessageBox::Show("Please enter all the fields",
                        "One or more empty fields", MessageBoxButtons::OK);
                return;
        }

        if (String::Compare(password, confirmPassword) != 0) {
                MessageBox::Show("Passwords are not match", "Password error",
MessageBoxButtons::OK);
        }
        try {

                String^ connString = "Data
Source=localhost\\MSSQLSERVER01;Initial
Catalog=ManagePersonalBusiness;Integrated Security=True";
                SqlConnection sqlConn(connString);
                sqlConn.Open();
                String^ sqlQuery = "INSERT INTO worker " +
                        "(name, email, phone, adress, position, password) VALUES
" +
                        "(@name, @email, @phone, @adress, @position,
@password);";
                SqlCommand command(sqlQuery, % sqlConn);
                command.Parameters->AddWithValue("@name", name);
                command.Parameters->AddWithValue("@email", email);
                command.Parameters->AddWithValue("@phone", phone);
                command.Parameters->AddWithValue("@adress", adress);
                command.Parameters->AddWithValue("@position", position);
                command.Parameters->AddWithValue("@password", password);
                command.ExecuteNonQuery();
```

```
                worker = gcnew Worker;
                worker->name = name;
                worker->email = email;
                worker->phone = phone;
                worker->adress = adress;
                worker->position = position;
                worker->password = password;
                this->Close();
        }
        catch(Exception^ ex) {
                MessageBox::Show("Failed to register new user"  + ex->Message,
                        "Register failture", MessageBoxButtons::OK);
        }
}
};
}
\\MainForm.h
#pragma once
#include "User.h"
#include "RegisterForm.h"
#include <string>
#include <vector>
#include <fstream>
namespace ManagePersonal {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
```

```
using namespace System::Data::SqlClient;
using namespace System::Runtime::InteropServices;
using namespace System::IO;
using namespace Microsoft::Office::Interop::Excel;
/// <summary>
/// Summary for MainForm
/// </summary>
public ref class MainForm : public System::Windows::Forms::Form
{
public:
	MainForm(User^ user)
	{
		InitializeComponent();
		//
		//TODO: Add the constructor code here
		//
		// Establish connection
		}
protected:
	/// <summary>
	/// Clean up any resources being used.
	/// </summary>
	~MainForm()
	{
		if (components)
		{
			delete components;
		}
	}
private: System::Windows::Forms::Button^ button1;
```

```
        private: System::Windows::Forms::DataGridView^ dataWorkers;
        private: System::Windows::Forms::Button^ button2;
        private: System::Windows::Forms::MenuStrip^ menuStrip1;
        private: System::Windows::Forms::ToolStripMenuItem^ toolStripMenuItem1;
        private: System::Windows::Forms::ToolStripMenuItem^ toolStripMenuItem2;
        private: System::Windows::Forms::ToolStripMenuItem^
exitToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
exitToolStripMenuItem1;
        protected:
        protected:
        protected:
        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;

        #pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                this->button1 = (gcnew System::Windows::Forms::Button());
                this->dataWorkers = (gcnew
System::Windows::Forms::DataGridView());
                this->button2 = (gcnew System::Windows::Forms::Button());
```

```
this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
this->toolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->toolStripMenuItem2 = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->exitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->exitToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataWorkers))->BeginInit();
this->menuStrip1->SuspendLayout();
this->SuspendLayout();
//
// button1
//
this->button1->BackColor = System::Drawing::SystemColors::ButtonFace;
this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei", 10.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->button1->Location = System::Drawing::Point(790, 102);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(231, 43);
this->button1->TabIndex = 0;
this->button1->Text = L"Реєстрація працівника";
this->button1->UseVisualStyleBackColor = false;
```

```
this->button1->Click += gcnew System::EventHandler(this,
&MainForm::button1_Click);
//
// dataWorkers
//
this->dataWorkers->Anchor =
static_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::A
nchorStyles::Top | System::Windows::Forms::AnchorStyles::Left)
| System::Windows::Forms::AnchorStyles::Right));
this->dataWorkers->AutoSizeColumnsMode =
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;
this->dataWorkers->AutoSizeRowsMode =
System::Windows::Forms::DataGridViewAutoSizeRowsMode::AllCells;
this->dataWorkers->BackgroundColor =
System::Drawing::SystemColors::ButtonFace;
this->dataWorkers->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize
;
this->dataWorkers->Location = System::Drawing::Point(12, 199);
this->dataWorkers->Name = L"dataWorkers";
this->dataWorkers->RowHeadersWidth = 51;
this->dataWorkers->RowTemplate->Height = 24;
this->dataWorkers->Size = System::Drawing::Size(1043, 271);
this->dataWorkers->TabIndex = 1;
this->dataWorkers->CellContentClick += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this,
&MainForm::dataWorkers_CellContentClick);
this->dataWorkers->CellEndEdit += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this,
&MainForm::dataWorkers_CellEndEdit);
```

```
this->dataWorkers->CellValueChanged += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this,
&MainForm::dataWorkers_CellValueChanged);
//
// button2
//
this->button2->BackColor =
System::Drawing::SystemColors::ButtonFace;
this->button2->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 10.8F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->button2->Location = System::Drawing::Point(54, 103);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(237, 42);
this->button2->TabIndex = 2;
this->button2->Text = L"Оновити дані таблиці";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this,
&MainForm::button2_Click);
//
// menuStrip1
//
this->menuStrip1->BackColor =
System::Drawing::SystemColors::ButtonFace;
this->menuStrip1->ImageScalingSize =
System::Drawing::Size(20, 20);
this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(4) {
this->toolStripMenuItem1,
```

```
                this->toolStripMenuItem2, this->exitToolStripMenuItem, this->exitToolStripMenuItem1
            });
            this->menuStrip1->Location = System::Drawing::Point(0, 0);
            this->menuStrip1->Name = L"menuStrip1";
            this->menuStrip1->Size = System::Drawing::Size(1067, 28);
            this->menuStrip1->TabIndex = 3;
            this->menuStrip1->Text = L"menuStrip1";
            this->menuStrip1->ItemClicked += gcnew System::Windows::Forms::ToolStripItemClickedEventHandler(this, &MainForm::menuStrip1_ItemClicked);
            //
            // toolStripMenuItem1
            //
            this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
            this->toolStripMenuItem1->Size = System::Drawing::Size(117, 24);
            this->toolStripMenuItem1->Text = L"Оновити дані";
            //
            // toolStripMenuItem2
            //
            this->toolStripMenuItem2->Name = L"toolStripMenuItem2";
            this->toolStripMenuItem2->Size = System::Drawing::Size(145, 24);
            this->toolStripMenuItem2->Text = L"Новий працівник";
            //
            // exitToolStripMenuItem
            //
            this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";
```

```
this->exitToolStripMenuItem->Size = System::Drawing::Size(124,
24);
this->exitToolStripMenuItem->Text = L"Про програму";
//
// exitToolStripMenuItem1
//
this->exitToolStripMenuItem1->Name =
L"exitToolStripMenuItem1";
this->exitToolStripMenuItem1->Size = System::Drawing::Size(60,
24);
this->exitToolStripMenuItem1->Text = L"Вихід";
//
// MainForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(12, 27);
this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ButtonFace;
this->ClientSize = System::Drawing::Size(1067, 482);
this->Controls->Add(this->button2);
this->Controls->Add(this->dataWorkers);
this->Controls->Add(this->button1);
this->Controls->Add(this->menuStrip1);
this->Font = (gcnew System::Drawing::Font(L"Microsoft YaHei",
12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
this->MainMenuStrip = this->menuStrip1;
this->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
this->Name = L"MainForm";
```

```
            this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
            this->Text = L"ManageWorkers";
            this->Load += gcnew System::EventHandler(this,
&MainForm::MainForm_Load);

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataWorkers))->EndInit();
            this->menuStrip1->ResumeLayout(false);
            this->menuStrip1->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();
        }
    #pragma endregion
    private: System::Void label1_Click(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void MainForm_Load(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {

        // Create an instance of RegisterForm
        ManagePersonal::RegisterForm^ registerForm = gcnew
ManagePersonal::RegisterForm();
        // Show RegisterForm
        registerForm->ShowDialog();
    }
```

```
private: System::Void dataWorkers_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {
        }
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
        // Establish connection
        String^ connString = "Data Source=localhost\\MSSQLSERVER01;Initial Catalog=ManagePersonalBusiness;Integrated Security=True;";
        SqlConnection^ sqlConn = gcnew SqlConnection(connString);
        sqlConn->Open();
        try {
            // Execute query
            String^ sqlQuery = "SELECT Id, name, email, phone, adress, position /* Exclude password column */ FROM worker";
            SqlCommand^ command = gcnew SqlCommand(sqlQuery, sqlConn);
            SqlDataReader^ reader = command->ExecuteReader();

            // Load data into DataTable and bind to DataGridView
            System::Data::DataTable^ dataTable = gcnew System::Data::DataTable();
            dataTable->Load(reader);
            dataWorkers->DataSource = dataTable;
        }
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ ПЕРСОНАЛОМ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

"РОЗРОБКА СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ ПЕРСОНАЛОМ"

СТУДЕНТ: ст. гр. 3Пі-20Б
Лозан О.В.
КЕРІВНИК: к.т.н., доцент
Рейда О.М.

Вінниця 2024

Рисунок Г.1 – Титульний слайд

Актуальність теми дослідження

- **Недостатня автоматизація процесів.** Багато аналогових систем не мають достатнього рівня автоматизації, що може призводити до затримок у виконанні завдань та збільшення трудомісткості роботи з персоналом.
- **Ефективність і продуктивність.** Автоматизовані системи управління персоналом дозволяють значно підвищити ефективність робочих процесів.
- **Підтримка віддаленої роботи.** В умовах зростання популярності віддаленої роботи автоматизовані HRM системи надають необхідні інструменти для управління персоналом незалежно від їхнього місцезнаходження.
- **Підвищення конкурентоспроможності.** Організації, що використовують передові технології в управлінні персоналом, мають перевагу перед конкурентами.

Рисунок Г.2 – Актуальність теми дослідження

Мета, предмет та об’єкт дослідження

Основною метою є вдосконалення та автоматизація процесів управління персоналом за рахунок впровадження сучасних технологій та шляхом розробки автоматизованої системи, що сприятиме підвищенню ефективності роботи, зменшенню необхідного часу для ведення обліку співробітників та моніторингу виконання завдань.

Предмет дослідження – методи та засоби розробки програмної системи для автоматизації управління персоналом

Об’єктом дослідження – процес автоматизації процесів у сфері управління персоналом.

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Задачі дослідження

– провести аналіз існуючих методів і засобів розробки системи для автоматизації процесів управління персоналом для визначення напрямків підвищення продуктивності виконання поставлених задач;

– розробка модуля графічного інтерфейсу, призначеного для відображення контенту та його управлінням;

– запропонувати метод створення завдань та вибору виконавців;

– розробка модуля для створення завдань та їх управлінням;

– розробка модуля для відображення та редагування даних співробітників;

– розробка моделі даних та архітектури системи;

– проведення експериментальних досліджень розроблених засобів і тестування системи.

Рисунок Г.4 – Задачі дослідження

Наукова новизна

Подальшого розвитку отримав метод створення завдань та вибору виконавців, що на відміну від існуючого, використовує регулярні вирази з ваговими коефіцієнтами до професійних навиків, для покращення відповідності виконавців задачам, і як наслідок підвищення продуктивності і якості виконаних робіт.

Рисунок Г.5 – Наукова новизна

Практичне значення отриманих результатів

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положеннях було запропоновано сучасні підходи до автоматизованих систем управління персоналом та розроблено програмні засоби для вдосконалення процесу управління.

Рисунок Г.6 – Практичне значення отриманих результатів

Аналоги

	PeopleHR	HiBob	WebWork	Asana	Manage Employee
Моніторинг часу	0	1	1	0	0
Відображеня даних співробітників	1	0	1	1	1
Редагування даних	1	1	0	1	1
Автоматична система	0	0	1	1	1
Захист даних	0	1	0	0	1
Сумарний коефіцієнт	2	3	3	3	4

Рисунок Г.7 – Аналоги

Методи та засоби розробки

Мова програмування C#

Засоби розробки MS Visual Studio

СУБД для керування БД

Монолітна архітектура

Рисунок Г.8 – Методи та засоби розробки

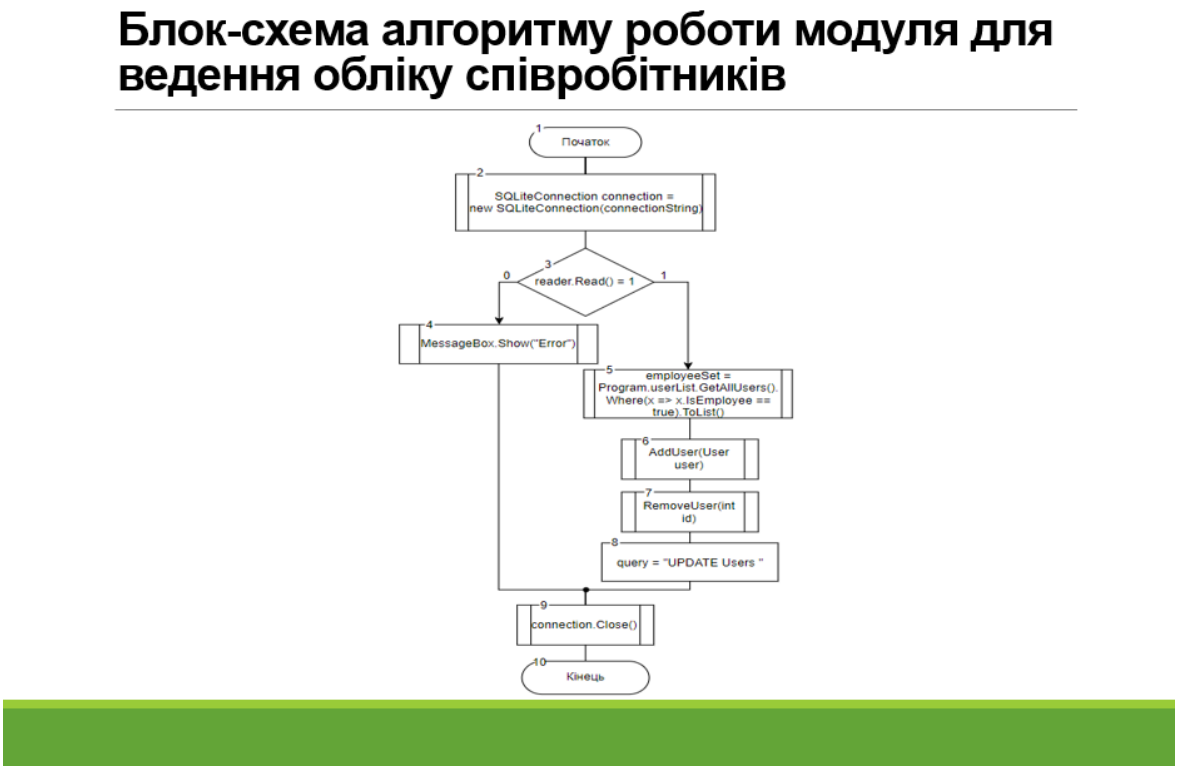


Рисунок Г.9 – Блок-схема алгоритму роботи модуля для ведення обліку співробітників

Рисунок Г.10 – Блок-схема алгоритму роботи модуля створення завдань та їх призначення виконавцеві

Рисунок Г.11 – Загальний алгоритм системи у вигляді діаграми станів

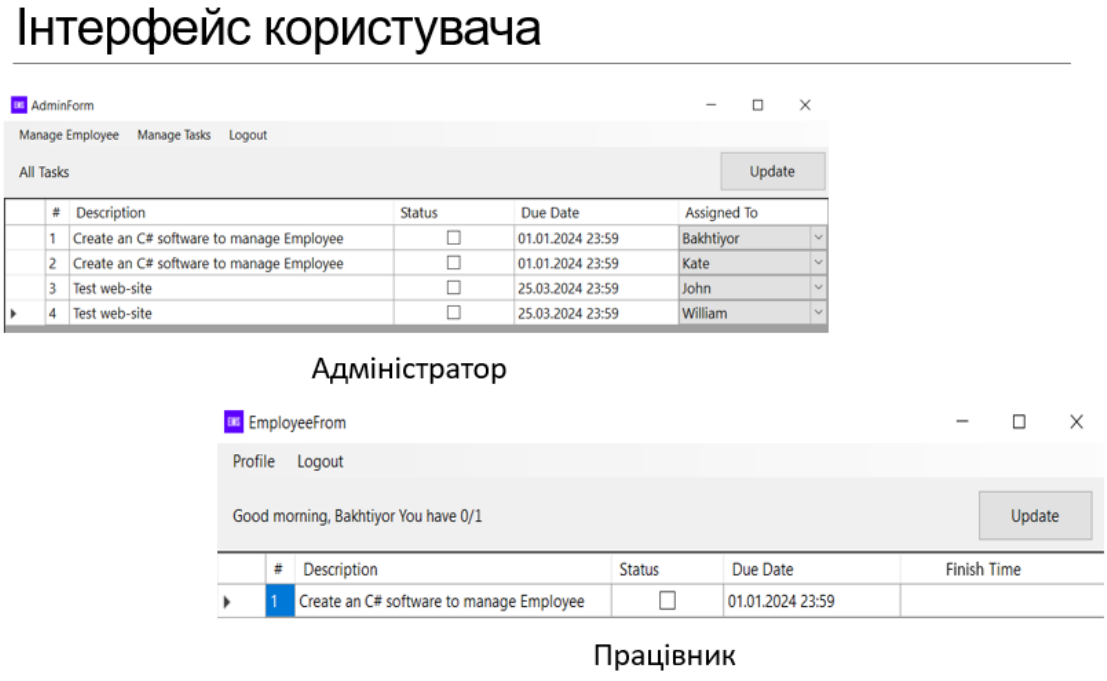
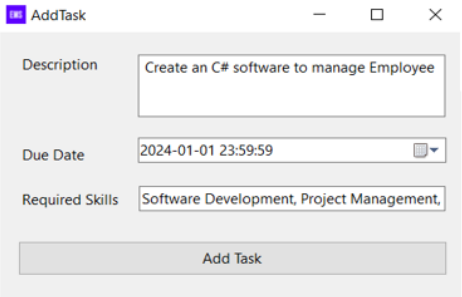


Рисунок Г.12 – Інтерфейс користувача

Інтерфейс вікна створення завдання



Інтерфейс вікна для ведення обліку працівників

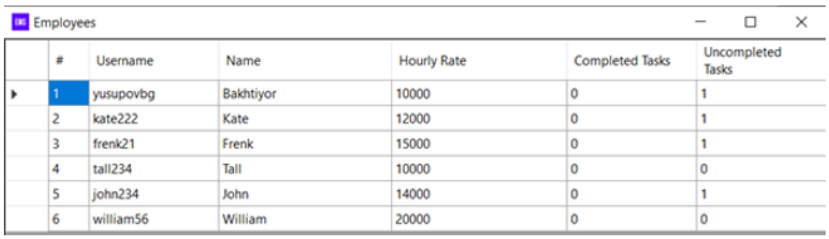


Рисунок Г.13 – Інтерфейс вікна створення завдання та ведення обліку працівників

Висновки

- Розглянуто стан питання автоматизації процесів управління персоналом та методи розв'язання задачі, проведено порівняльний аналіз аналогів.
- Розроблено загальний алгоритм роботи системи.
- Удосконалено метод створення завдань та вибору виконавців, що на відміну від існуючого, використовує регулярні вирази з ваговими коефіцієнтами до професійних навиків, для покращення відповідності виконавців задачам, і як наслідок підвищення продуктивності і якості виконаних робіт.
- Розглянуто переваги й недоліки методів програмування та засобів розробки.
- Розроблено методику і проведено тестування програмної системи.
- Розроблено інструкцію користувача системи.

Рисунок Г.14 – Висновки

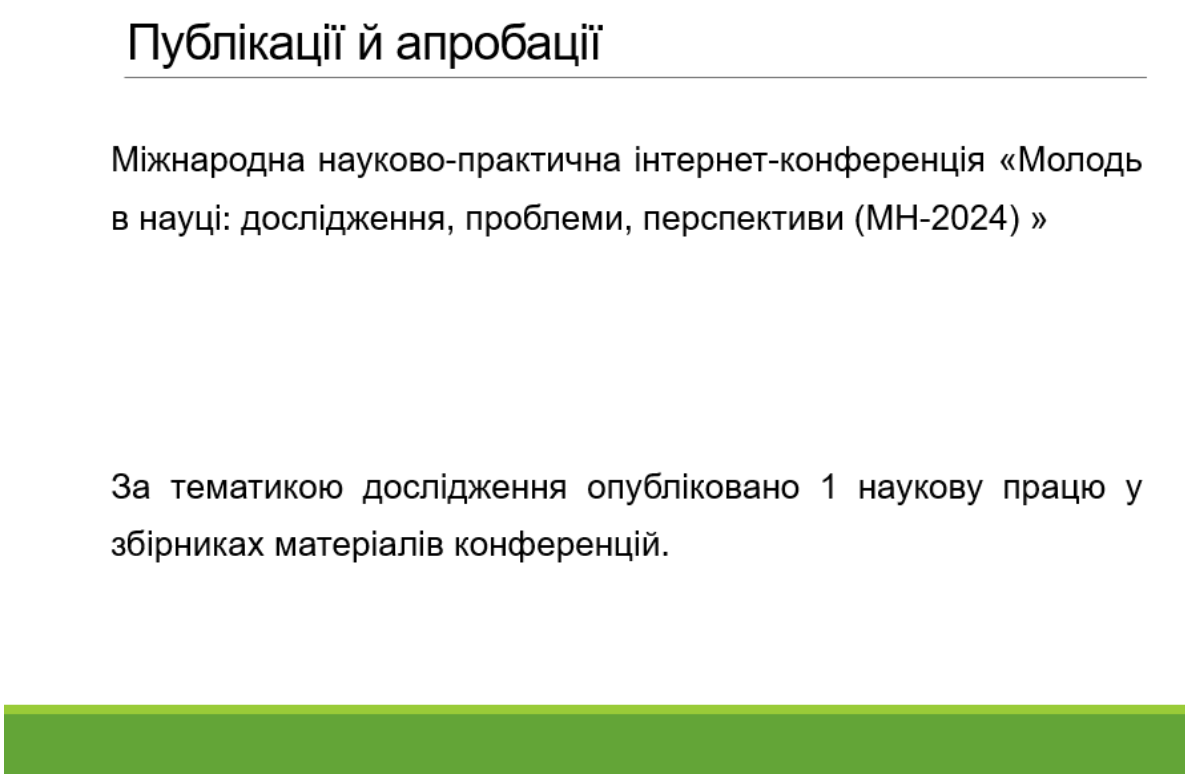


Рисунок Г.15 – Публікації та апробації матеріалів бакалаврської роботи

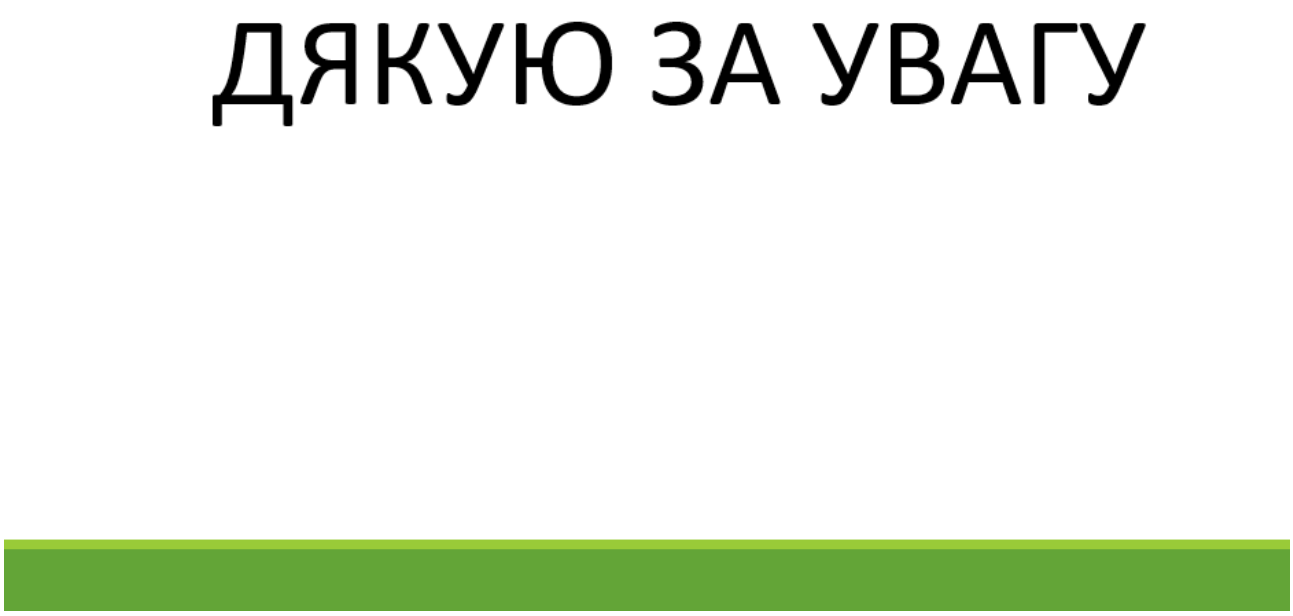


Рисунок Г.16 – Фінальний слайд