Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методу та програмного засобу для організації роботи ІТ-команд»

Виконав: студент 2-го курсу, групи ЗПІ-22м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Давиденко І.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ:
Бабюк Н.П.
(прізвище та ініціали)
«18» грудня 2023 р.

Опонент:
к.т.н. доц.каф. ЗІ Маліновський В.І.
(прізвище та ініціали)
«18» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н. проф. Романюк О.Н.
(прізвище та ініціали)
«18» грудня 2023 р.

ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Давиденку Іллі Сергійовичу

1. Тема роботи – Розробка методу та програмного засобу для організації роботи ІТ-команд.

Керівник роботи: Бабюк Наталя Петрівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи 5 грудня 2023 року

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова розробки Javascript, система керування базами даних – Firebase, операційна система – Windows 11, метод моніторингу працівників ІТ-команд, метод надання відгуків.

4. Зміст текстової частини: вступ; аналіз стану розвитку засобів оптимізації робочих завдань ІТ-команди; розробка структури і алгоритмів роботи програми; розробка методу і програмного засобу для оптимізації робочих завдань ІТ-команди; тестування програми; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: титульний слайд; мета, предмет і об'єкт дослідження; новизна та практична цінність; актуальність розробки; задачі дослідження; розробка структури інтерфейсу програмного засобу; використані технології; реалізація методу моніторингу працівників ІТ-команд; реалізація методу надання відгуків; тестування програми; висновки; публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Бабюк Н.П., к.т.н., доцент кафедри ПЗ	19.09.23	05.12.23
5	Причепа І.В, к.е.н., доцент кафедри ЕПВМ	22.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку засобів оптимізації робочих завдань ІТ-команди	20.09.2023 28.09.2023	вик
2	Розробка структури і алгоритмів роботи програми	29.09.2023 14.10.2023	вик
3	Розробка методу і програмного засобу для оптимізації робочих завдань ІТ-команди	15.10.2023 05.11.2023	вик
4	Тестування програми	06.11.2023 21.11.2023	вик
5	Економічна частина	22.11.2023 01.12.2023	вик

Студент _____ (підпис) Давиденко І.С. (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ (підпис) Бабюк Н.П. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Поліщук М.О. Методи і програмні засоби для моніторингу діяльності персоналу: магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 115 с.

На укр. мові. Бібліогр.: 24 назв; рис.: 50; табл. 12.

Магістерська кваліфікаційна робота присвячена розробці методу та програмного засобу для організації роботи IT-команд. Для вирішення поставленої задачі була використана мова програмування Javascript з використанням бібліотеки React.js а також фреймворки Node.js. В якості бази даних була використана Firebase.

Розроблено метод організації роботи IT-команд, а саме: моніторинг працівників з використанням відгуків та зберіганням всієї важливої для робітників інформації в одному застосунку.

Розроблено програмний засіб, який має на меті допомогу працівникам IT-компаній з організацією IT-команд. Даний додаток має зручний інтерфейс, який є легким в розумінні для нових працівників та допомагає їм краще орієнтуватися і адаптуватися в компанії. Застосунок містить в собі всю інформацію, якою працівник може цікавитися про себе в компанії. Присутня можливість залишати відгуки про своїх колег та отримувати відгуки від них. Програмний засіб допомагає менеджерам і звичайним працівникам економити час на пошуку інформації та полегшує взаємодію з колегами що суттєво покращить продуктивність команд.

В магістерській роботі були також виконані економічні розрахунки по визначенню доцільності нового програмного продукту.

ABSTRACT

UDC 004.42 Polishchuk M.O. Methods and Software Tools for Personnel Activity Monitoring: Master's Qualification Work in the field of 121 – Software Engineering, Educational Program – Software Engineering. Vinnytsia: VNTU, 2023. 115 p.

In Ukrainian. Bibliography: 24 titles; figures: 50; tables: 12.

The master's qualification work is dedicated to the development of a method and software tool for organizing the work of IT teams. To address the stated task, the Javascript programming language was used, utilizing the React.js library and the Node.js frameworks. Firebase was chosen as the database.

A method for organizing the work of IT teams was developed, specifically focusing on monitoring employees through feedback and storing all essential information for workers in a single application.

A software tool was developed with the aim of assisting IT company employees in organizing IT teams. This application features a user-friendly interface that is easy for new employees to understand, helping them better orient and adapt within the company. The application contains all the information that employees may be interested in about themselves within the company. It provides the ability to leave feedback about colleagues and receive feedback from them. The software tool aids both managers and regular employees in saving time on information retrieval and facilitates interaction with colleagues, significantly improving team productivity.

In the master's thesis, economic calculations were also performed to determine the feasibility of a new software product.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Всі IT-компанії складаються з команд працівників, роботу яких потрібно правильно організовувати. До організації роботи відноситься багато факторів, такі як розподілення людей по командам, моніторинг їх взаємодії, перевірка стану працівників тощо. Кожен з цих факторів відслідковується менеджментом, і без правильного підходу це дуже важкий процес.

Для організації роботи потрібно щоб всі працівники працювали в комфорті, а це означає що їм повинні подобатися умови в яких вони працюють, їх колеги і відношення до них. Чим більше людина відчуває, що її цінують, тим більш продуктивною вона буде. Існує багато методів, щоб досягнути цього, але більшість з них займають занадто багато ресурсів [1].

Тому розробка методу та програмного засобу для організації роботи IT-команд є доцільною.

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

Мета та завдання дослідження. Метою розробки є підвищення ефективності організації праці працівників та їх стану за рахунок розробки програмного засобу з використанням методу для організації роботи IT-команд.

Відповідно до поставленої мети в дипломному проекті потрібно вирішити такі завдання:

- провести аналіз існуючих методів і засобів організації роботи IT-команд для визначення напрямків підвищення їх продуктивності та ефективності;

- удосконалити метод моніторингу працівників IT-команд;

- розробити програмні компоненти та систему візуалізації на основі запропонованих методів;

- розробити основний алгоритм організації роботи IT-команд з використанням методу обміну відгуками та покращенням методу моніторингу працівників;

- розробити інтерфейс програмного засобу;

- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

Об’єкт дослідження – процес розробки програмного засобу організації роботи IT-команд. Принципи поєднання клієнтської та серверної частини проектів.

Предмет дослідження – моделі, методи, алгоритми та програмні засоби для організації роботи IT-команд.

Методи дослідження. У процесі досліджень використовувались: теорія чисел та чисельних методів, теорія прийняття рішень, теорія алгоритмів.

Наукова новизна отриманих результатів.

Подальшого розвитку отримав метод організації роботи IT-команд, що відрізняється від існуючих можливістю обміну відгуками, що дозволяє учасникам команд та їх менеджерам підвищити ефективність праці та комунікації.

Практична цінність отриманих результатів.

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в роботі алгоритмів було розроблено програмний засіб для оптимізації робочих завдань IT-команди.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2022), Молодь в науці: дослідження, проблеми, перспективи (МН-2023).

Публікації. За результатами досліджень опубліковано 2 наукові публікації.

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану проблеми

Сфера інформаційних технологій приваблює все більше людей з кожним днем. Багатьом людям вона здається роботою їх мрії, адже це виглядає як проста робота за комп’ютером для потрапляння в яку можна вивчитися повністю самому. Через такі хибні сподівання люди зіштовхуються з неоправданими очікуваннями і робота в цій сфері для них виявляється дуже складною. Робітники можуть звикнути до своєї нової позиції, але особливо важко це для менеджменту, якому потрібно вмотивувати людей щоб вони продовжували працювати.

Проблеми з персоналом можуть виникнути не лише з новими працівниками, але й з тими які працюють вже давно. Під проблемами мається на увазі не лише продуктивність, але й задоволення людей від своєї роботи, взаємодії з колегами, цінування їх вкладу, тощо. Слідкувати за станом працівників дуже важливо, бо якщо легковажно до цього відноситися вони можуть «вигоріти». Емоційне вигорання – це синдром постійної втоми, емоційного виснаження, який посилюється з часом. Необхідність постійно бути на зв’язку, вислуховувати інших і говорити самому, стежити за важливими новинами, давати вказівки підлеглим або навпаки виконувати доручення керівництва, намагатися виправдати чиїсь очікування, перевершити конкурентів, продемонструвати хороші показники – все це призводить до того, що в якийсь момент у людини просто «сідають батарейки» [2]. Фізично неможливо одночасно пам'ятати взаємовідносини між всіма працівниками і їх моральний стан, тому цей процес потребує автоматизації.

Для контролю роботи і продуктивності можна використовувати різні методи:

- Ключові показники ефективності (KPI) – Ключові показники ефективності є важливим інструментом для вимірювання рівня виконання

поставлених завдань;

- 360-градусний зворотний зв'язок – Цей метод включає збирання відгуків від керівників, колег та підлеглих;
- Оцінка компетенцій та навичок – Важливо оцінити технічні та міжособистісні навички кожного члена команди. Це можна зробити через технічні інтерв'ю, оцінку професійних сертифікатів, або спеціальні тести.

Існує велика кількість методів для відслідковування продуктивності персоналу, вище була наведена лише частина з них. Який би з методів не було обрано, їх потрібно впроваджувати за допомогою певних засобів або застосунків. Можна використовувати веб-сайти, наприклад, такі веб-сайти, як Jira, Asana, або Slack, надають інструменти для створення завдань, відстеження їх виконання та аналізу KPI. Також можна використовувати більш традиційні засоби, тобто використовувати звіти та таблиці.

Окрім відслідковування прогресу чи продуктивності персоналу з сторони менеджменту, самі працівники теж повинні мати змогу бачити певну інформацію про себе в компанії. Це може бути навіть щось незначене, бачити скільки днів відпустки в них залишилось чи нагадування про дні народження чи річниці колег, це все гарно впливає на стан працівників.

Чим ціннішим себе відчуває працівник, тим більше користі і прибутку він принесе компанії, а для цього потрібно створити всі умови для його комфорту і вчасно реагувати якщо працівник перестає працювати так продуктивно, як він працював раніше, або коли він починає показувати перші признаки вигорання.

Забезпечення прозорості та доступу до особистої інформації для працівників – це не лише елемент доброзичливого ставлення до них, але і стратегічний крок для підтримки їхнього ефективного внеску у розвиток компанії.

Окрім наведених прикладів, надання працівникам можливості відстежувати свої успіхи та досягнення може бути ключовим елементом

корпоративної культури. Впровадження системи самооцінки або регулярних обговорень професійного зростання може допомогти працівникам усвідомити свій внесок та визначити області для подальшого розвитку.

Крім того, створення механізмів зворотного зв'язку та системи підтримки працівників в разі стресових ситуацій чи вигорання може значно поліпшити їхню психологічну та емоційну стійкість. Важливо виявляти розуміння та готовність реагувати на сигнали, які вказують на можливі проблеми зі станом працівників.

Загалом, інтеграція засобів, які підтримують взаємодію та взаєморозуміння між керівництвом та працівниками, створить сприятливе середовище для розвитку команди та забезпечить більшу відданість та продуктивність на робочому місці.

Досягнення цієї взаємодії вимагає інноваційних підходів до управління персоналом. Використання сучасних інструментів та технологій, таких як корпоративні портали, додатки для ефективної комунікації та онлайн-платформи для навчання, може стати кроком у майбутнє. Постійна адаптація та вдосконалення таких інструментів дозволяють компаніям не лише відповідати потребам працівників, але й виходити за межі, забезпечуючи стабільну та продуктивну робочу обстановку.

Отже, враховуючи усе вищесказане, зроблено висновок що необхідно розробити програмний засіб для організації роботи IT-команд який буде надавати змогу для відслідковування продуктивності працівників, збирати про них відгуки та слідкувати за їх задоволенням роботи в компанії.

1.2 Порівняльний аналіз аналогів розроблюваного застосунку

Найкращим шляхом визначення актуальності розробки застосунку є проведення аналізу його аналогів та їх порівняння за найголовнішими для схожих продуктів критеріями оцінювання.

Менеджмент дуже важлива складова сфери інформаційних технологій, тому додатки для керування персоналом користуються популярністю серед

користувачів, але більшість з них не надають всього важливого функціоналу, і для детального аналізу працівники змушені використовувати декілька застосунків.

При розробці застосунку для керування персоналом також важливо враховувати сучасні тенденції у сфері технологій та забезпечувати його сумісність з різноманітними платформами та пристроями. Крім того, врахування високих стандартів кібербезпеки є ключовим аспектом у забезпеченні захисту конфіденційної інформації та великого обсягу даних, що обробляються застосунком.

Необхідно також враховувати потреби користувачів у зручному інтерфейсі, який дозволяє легко орієнтуватися в функціоналі та використовувати всі можливості застосунку без надмірного навантаження на користувача.

Крім того, створення механізмів зворотнього зв'язку та підтримки може сприяти постійному вдосконаленню та адаптації застосунку до змінних потреб користувачів і підвищити його конкурентоспроможність на ринку програмного забезпечення для управління персоналом.

Успішний розвиток застосунку також передбачає регулярне оновлення та вдосконалення його функціоналу на основі отриманого зворотного зв'язку від користувачів та аналізу динаміки змін у сфері управління персоналом. Окрім цього, важливо звертати увагу на інноваційні рішення, які можуть покращити ефективність управління персоналом та вплинути на підвищення продуктивності команди.

Розглянемо найпопулярніші застосунки для управління персоналом.

Workday (див. рис. 1.1) є постачальником хмарного програмного забезпечення, яке спеціалізується на програмах для управління фінансами, планування ресурсів підприємства (ERP) і управління людським капіталом (HCM). Workday є піонером у сфері програмного забезпечення для управління персоналом як послуги (SaaS) і має штаб-квартиру, яка знаходиться в Плезантоні, Каліфорнія [3].

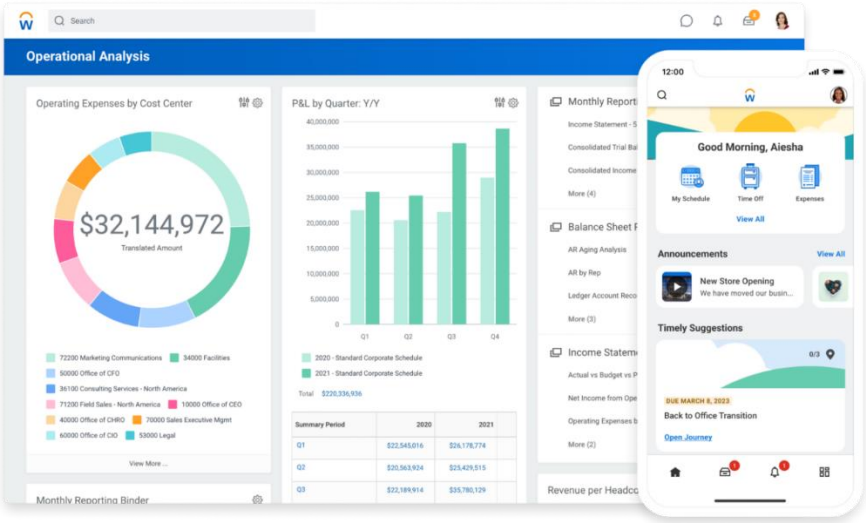

Рисунок 1.1 – Workday

SAP SuccessFactors Employee Central (див. рис. 1.2) – це програмне забезпечення SAP для керування персоналом. Хмарне рішення може допомогти стандартизувати процеси у глобальному масштабі та забезпечити прозорість для прийняття більш виважених рішень. Вона включає можливості управління профілями користувачів, організаційними діаграмами, глобальним керуванням пільгами та керуванням відсутністю [4].

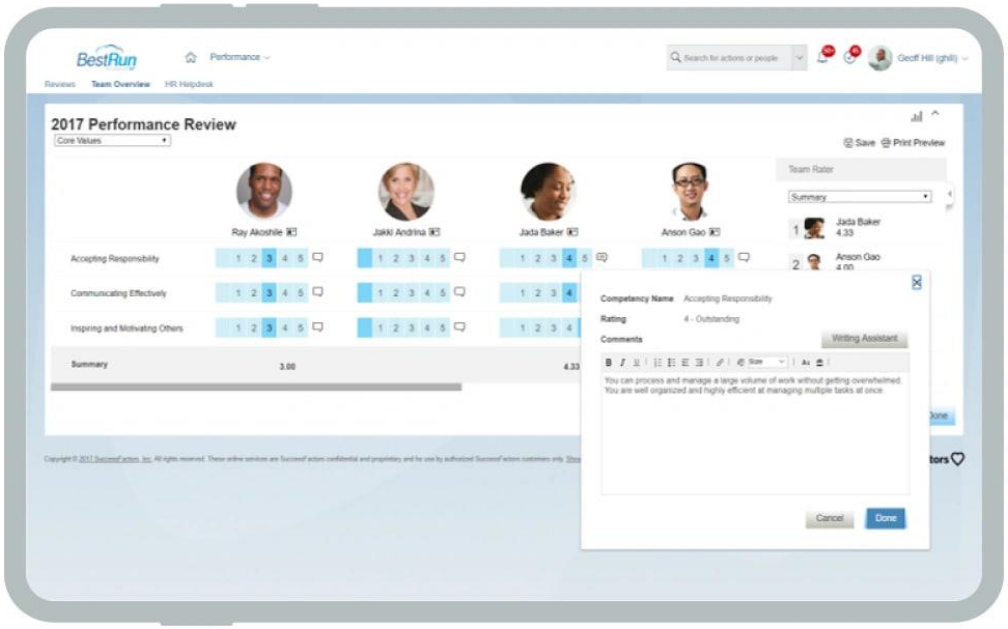

Рисунок 1.2 – SAP SuccessFactors Employee Central

BambooHR (див. рис. 1.3) – це інформаційна система кадрових ресурсів (HRIS), яка об’єднує в одній системі такі функції управління персоналом, як розрахунок заробітної плати, відстеження робочого часу, залучення співробітників, систему відстеження заявників тощо. Це хмарне програмне забезпечення ідеально підходить для малого та середнього бізнесу, які хочуть оптимізувати свої кадрові процеси для підвищення ефективності, оскільки воно зберігає всю інформацію про співробітників в одному місці для легкого доступу для співробітників, кадрових фахівців, рекрутерів і менеджерів [5].

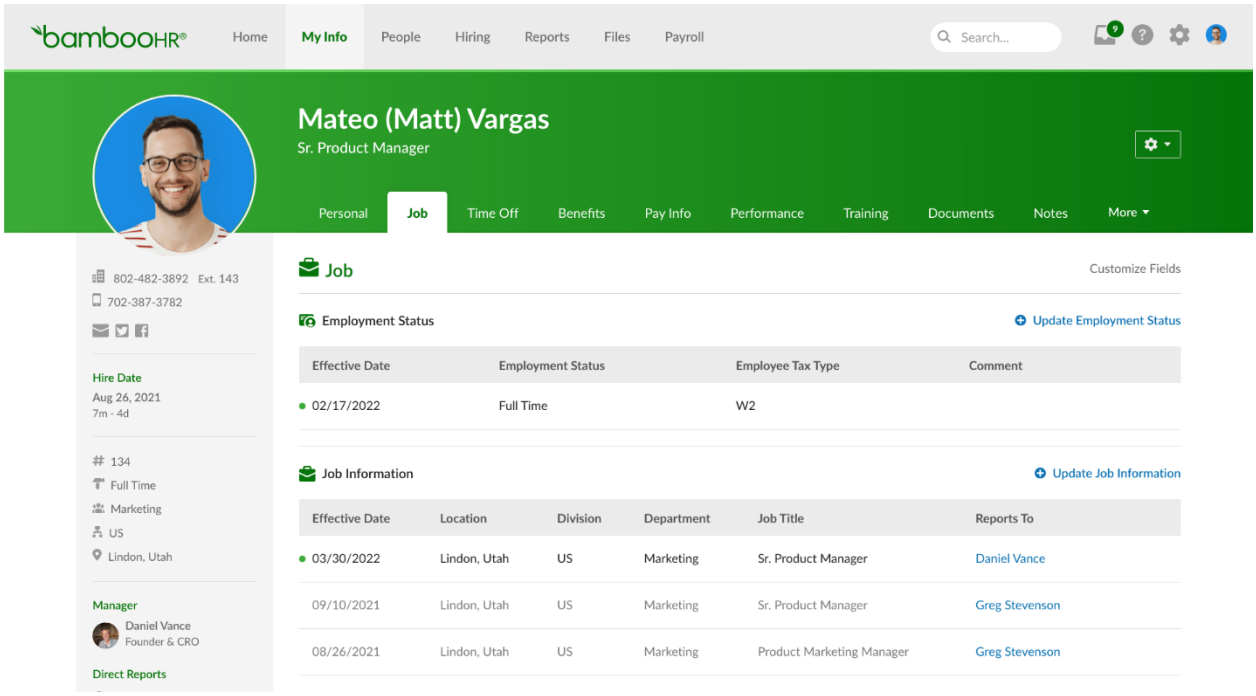

Рисунок 1.3 – Вікно роботи програми BambooHR

Порівняльний аналіз застосунків для керування персоналом є складним завданням, і його результативність значно полегшується завдяки систематичному підходу до збору та обробки інформації. Для досягнення цієї мети була створена спеціалізована таблиця, яка містить оцінку кожного з розглянутих застосунків з урахуванням важливих функціональних аспектів. Результати порівняння аналогів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів

Критерії оцінки	Workday	SAP SuccessFactors Employee Central	BambooHR	EasyHR
Можливість залишення відгуків	-	-	-	+
Можливість оцінки продуктивності	+	+	-	+
Збереження інформації про працівників	+	+	+	+
Легке впровадження	-	-	+	+

Розроблюваний застосунок «EasyHR» було порівняно з аналогами за такими критеріями:

1. Можливість залишення відгуків. Відповідає за можливість працівників залишати відгуки про своїх колег, анонімні або ні.

2. Можливість оцінки продуктивності – наявність в системи застосунків для вирахування продуктивності працівників.

3. Збереження інформації про працівників – можливість системи зберігати і відображати дані працівників в їх особистих кабінетах.

4. Легке впровадження – сукупність факторів які роблять впровадження системи в компанію легким. Факторами вважаються: кількість функціоналу, ціна, оцінка часу для навчання персоналу користуванню з системою тощо.

Виходячи з результатів порівняльного аналізу аналогів застосунку «EasyHR» можна зробити висновок, що розробка програмного засобу є актуальною та що він має переваги які роблять його конкурентоспроможним.

1.3 Аналіз методів вирішення поставленої задач

Одним з найголовніших і найбільш відповідальних завдань в ІТ-компанії є утримання персоналу і підтримка їх морального стану на високому рівні. Враховуючи те, що кожна людина різна і до кожної потрібно знайти окремий підхід ця задача виглядає дуже складною. Як все ж можна досягти успіху в цій нелегкій справі?

Не дивлячись на досвід, рівень ієрархії чи інші визначальні фактори до кожного працівника потрібно знайти особистий підхід, але це не повинно виглядати нав'язливо, бо на робочому місці людина в першу чергу виконує поставлені їй задачі, а лише потім вже є людиною відкритою до вільної комунікації. Враховуючи це, потрібно слідкувати за її моральним станом в компанії за допомогою методів які не будуть перешкоджати робочому процесу, а навпаки, будуть відчуватися працівником як можливість ввести щось в його покращення.

Такими методами є:

– Проведення анонімних або не анонімних опитувань щодо робочих процесів або атмосфери. Таким чином можна слідкувати чи задоволені працівники своєю роботою або цінностями компанії в цілому. Дуже важливо проводити такі опитування, бо вони допомагають зрозуміти загальну картину задоволення людей на їх робочих місцях. Цей метод є не таким простим, адже щоб він надавав реальні результати працівники завжди повинні бачити реальні результати цих опитувань. Тобто, якщо багато людей відмітили погане ставлення до персоналу, компанія повинна почати робити кроки для виправлення цієї ситуації, або ж якщо це неможливо зробити в короткі терміни, проінформувати команду що в найближчому майбутньому будуть прийняті міри для усунення цієї проблеми.

– Проведення діалогів з головами команд. Наприклад, дуже добре взяти за практику проводити зустрічі в форматі 1-1 працівників з їх менеджерами по проекту. Таким чином менеджери будуть знати чи подобається людині працювати над тим чи іншим проектом, чи задоволена вона своїм

навантаженням, тощо. Також це допомагає менеджерам зібрати інформацію щодо покращень процесів.

– Збір анонімних або не анонімних відгуків працівників один про одного. Так можна зрозуміти сильні чи слабкі сторони працівника, якщо мати інформацію про нього від людей з якими він працює. Якщо ж відгуки анонімні, то можна отримати корисну інформацію про працівника без можливого погіршення відносин в колективі в разі того що цей відгук буде поганим.

– Аналіз емоційного стану працівників шляхом опитувань і намагання покращити їх стан шляхом надання їм інформаційних ресурсів для покращення ситуації з їх станом.

– Навчання та розвиток Створення системи внутрішнього навчання, яка адаптується до потреб працівників, базується на аналізі їхнього професійного росту. Збираючи дані про результати проходження курсів та відображаючи навички в профілях працівників, система визначає, які аспекти потребують уваги та подальшого розвитку.

– Співпраця та комунікація Оптимізація системи комунікації передбачає вдосконалення інструментів для спілкування та спільної роботи. Це може включати в себе інтеграцію онлайн-спілкування та інших інструментів для полегшення комунікації в команді. Забезпечення можливості відгуків на комунікацію допомагає покращити взаєморозуміння та сприяє ефективності командних завдань. При цьому, система може аналізувати здатність працівників до співпраці та вносити рекомендації для подальшого покращення.

Будь-який метод буде майже неможливим для втілення якщо в команді не побудована атмосфера довіри один до одного, тому починати втілення будь-яких покращень потрібно лише після обговорення серед всього вищого менеджменту.

В багатьох компаніях ці процеси досі не автоматизовані, тому менеджерам приходиться питати в кожного особисто їх думки про того чи іншого працівника якщо вони бажають взнати відношення людей один до

одного. Головним недоліком цього підходу є те, що людині про яку питають відгук можуть не повністю або не дуже детально передати, що їй потрібно покращити в своїй роботі, а що вона робить добре. Маючи можливість надавати ці відгуки через певний програмний засіб і надавати можливість як звичайним працівникам так і людям які є менеджерами цих працівників бачити відгуки це дуже зручний метод взаємодії між людьми в компанії.

Взаємодія з іншими працівниками це не лише відгуки один про одного, це також знання коли в ваших колег день народження або річниця в компанії. Якщо працівник буде мати інформацію про це в доступному місці він зможе краще зближатися з людьми з якими він працює, а це дуже добре впливає на робочу атмосферу. Також людям дуже добре взнавати інформацію про нових працівників або про тих хто пішов з компанії в певному ресурсі, бо так можна привітати або попрощатися з людьми з якими працюєш без необхідності питань в менеджерів про зміни в людському ресурсі компанії. Все це можна покращити шляхом впровадження застосунку для відображення цієї інформації.

Також проблемою на робочих місцях є складність розуміння працівником того, що надає йому компанія. Наприклад, часто бувають ситуації коли працівнику потрібно самому слідкувати за кількістю днів відпустки яка в нього залишилася, або які переваги надає йому компанія. Якщо ця інформація знаходиться в різних місцях які складно знайти то це зайва витрата часу одного або декількох працівників, адже якщо один не знайде місце з певною інформацією він піде питати в інших. Також людям набагато приємніше мати одне місце з всією корисною інформацією, так як вони можуть в будь-який час зайти і побачити все що їх цікавить про себе в компанії.

З наявністю застосунка менеджменту стає набагато легше взаємодіяти з працівниками. Вони можуть переглядати інформацію про них в одному місці, бачити відгуки інших колег про них, змінювати важливу інформацію про працівників тощо. Працівники в свою чергу можуть теж надавати

менеджменту інформацію про себе прямо в профілі, без зайвих зустрічей з людьми. Це забирає багато рутинної роботи з життя людей, і вони можуть витрачати свій час на щось більш важливе.

В результаті аналізу існуючих проблем та методів їх вирішення, було вирішено розробити метод для

1.4 Постановка задач на проектування

Магістерська кваліфікаційна робота присвячена розробка методу та програмного засобу для організації роботи IT-команд. Для його інтерфейсу було висунуто такі вимоги:

- зрозумілий для нових працівників;
- лаконічний дизайн з невеликою кількістю кольорів;

Для алгоритмів програмного засобу було висунуто такі вимоги:

- розробити алгоритм реалізації можливості формування звітності та перегляду статистики;
- розробити алгоритм створення та перегляду відгуків;
- розробити алгоритм для організації роботи IT-команд;
- розробити програмний продукт, призначений для вирішення поставлених проблем;
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

1.5 Висновки

Було проведено аналіз сучасного стану питання та порівняли існуючі аналоги, щоб визначити можливості для поліпшень у майбутньому програмному продукті. В результаті порівнянь даних вже існуючих додатків було підтверджено доцільність розробки. Було поставлено задачі та методи для проектування програмного продукту.

2 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вимог до застосунків для організації IT-команд

Дуже важливою частиною застосунків для організації персоналу є їх структура і наповнення. Ці застосунки не мають мати занадто багато контенту, але мають мати його достатньо щоб всі його користувачі отримували достатню кількість важливої для них інформації.

Так як розглядається додаток для організації IT-команд, на головній сторінці він повинен мати новини в командах або якісь події колег (підвищення, день народження, річниця в компанії тощо). При переході в профіль працівник має бачити всю інформацію яка може його цікавити, бо це допоможе йому не мати відчуття що якась інформація від нього приховується. При перегляді профілів інших працівників необхідно відображати лише ту інформацію яка не є чутливою. День народження, позиція, набір навичок тощо – це та інформація яку можна відображати іншим користувачам. Якщо ж профілі переглядає менеджер, цей список може бути розширено.

При розробці структури сторінок необхідно враховувати яка інформація може бути найбільш цікава користувачу. Таким чином, ім’я, дата народження, адреса тощо повинні бути з самого верху, бо так користувач буде точно впевнений що він переглядає саме свій профіль. Ім’я можна відображати на кожній сторінці, замість посилання на авторизацію, так працівник буде знати що він авторизований і має змогу в будь-яку мить потрапити до свого особистого кабінету. Блок з відгуками потрібно помістити на окрему сторінку яка буде доступна по посиланню в профілю, адже на завжди працівник має змогу чи ресурс на перегляд відгуків тому цей блок має бути доступним лише по прямому переходу на нього [6].

Важливою частиною є навігація. Вона повинна мати в собі все необхідне і нічого зайвого. Застосунок повинен виконувати лише свою задачу, і не мати в собі інформації про компанію, її новини, тощо. При пересуванні веб-сайтом

працівник повинен знати де він зараз знаходиться і яку інформацію переглядає.

Гарна структура веб-сайту Підвищує впізнаваність. Добре спроектований інтерфейс може стати важливим елементом впізнаваності сайту. Користувачі запам'ятовують позитивний досвід взаємодії та асоціюють його з конкретним продуктом чи брендом. Чітка структура дозволяє уникнути непорозумінь та помилок під час взаємодії з продуктом. Користувачі легше розуміють, що вони роблять та як впливає їхні дії на результат. Ефективна структура інтерфейсу робить користування продуктом більш комфортним та приємним. Це включає в себе зрозумілість взаємодії, чіткість повідомлень та інтуїтивно зрозумілі елементи управління.

Швидкодія веб-сайту визначається не лише технічними характеристиками серверів та мережі, але і ефективністю самої структури сторінок. Щоб забезпечити швидку та ефективну роботу додатка для організації IT-команд, необхідно враховувати кілька ключових аспектів.

По-перше, важливо оптимізувати завантаження сторінок. Це можна досягти за допомогою кешування ресурсів, мінімізації файлів CSS та JavaScript, а також використання стиснення зображень. Ефективне використання кешування дозволить прискорити завантаження сторінок для повторних відвідувачів, а мінімізація файлів дозволить зменшити обсяг передаваного контенту.

По-друге, важливо дбати про оптимальну структуру бази даних. Додаток для організації персоналу має швидко взаємодіяти з великою кількістю даних, тому ефективні запити до бази та оптимізована структура таблиць грають ключову роль. Використання індексів, кешування популярних запитів та регулярне апгрейдування бази даних допомагають забезпечити швидкодію та надійність роботи системи.

По-третє, асинхронні технології можуть покращити реакцію веб-сайту на дії користувачів. Застосування AJAX та інших асинхронних запитів дозволяє оновлювати частини сторінок без перезавантаження всього контенту.

Це сприяє більш швидкій і відзивчивій роботі сайту, особливо при великій кількості користувачів.

Загалом, при розробці структури додатка важливо не лише забезпечити доступність потрібної інформації, але й оптимізувати її відображення та обробку, щоб забезпечити швидку та ефективну роботу веб-сайту.

Хороша структура веб-сайту є ключовим елементом для досягнення успіху в онлайн-середовищі, оскільки вона впливає на різноманітні аспекти, які стосуються користувачів, пошукових систем і загальної ефективності веб-проекту. Враховуючи це, можна визначити важливість грамотної структури:

1. Пошукова оптимізація (SEO): Чітка структура сприяє кращій індексації веб-сайту пошуковими системами. Правильне використання метатегів, ключових слів у URL і логічна ієрархія сторінок полегшують завдання пошуковим роботам, що може позитивно впливати на рейтинг в пошукових результатах.

2. Швидкість завантаження: Ієрархічна та логічна структура сприяє покращенню швидкості завантаження сторінок, що є важливим фактором для задоволення користувачів і позитивного впливу на SEO.

3. Аналітика та відстеження: Грамотна структура дозволяє налаштовувати аналітичні інструменти для збору даних про відвідуваність та поведінку користувачів, що надає цінні інсайти для вдосконалення веб-сайту.

При розробці структури дуже важливим елементом є SEO, або Search Engine Optimization (Оптимізація для пошукових систем), – це комплекс заходів і стратегій, спрямованих на підвищення видимості веб-сайту в результатах пошукових систем для певних ключових слів чи фраз. Основна мета SEO – забезпечити, щоб ваш веб-сайт виглядав більш привабливим та релевантним для пошукових систем, що, в свою чергу, призводить до покращення його рейтингу та привертання більшої кількості органічного трафіку [7].

Важливість SEO в бізнес-середовищі важко переоцінити, і ось чому:

• Збільшення видимості: SEO допомагає вашому сайту виходити в топові позиції пошукових результатів, забезпечуючи більшу видимість для потенційних користувачів.

• Залучення цільового трафіку: Якщо ваш веб-сайт відповідає запитам користувачів у пошукових системах, ймовірність привертання цільового трафіку, тобто відвідувачів, які активно шукають продукти чи послуги, значно зростає.

• Підвищення авторитету: Оптимізований сайт зазвичай вважається авторитетнішим у очах пошукових систем, що може позитивно вплинути на позиції в результатах пошуку.

• Покращення користувацького досвіду: Багато елементів SEO, таких як швидкість завантаження сторінок, мобільна оптимізація та якісний контент, також сприяють поліпшенню загального користувацького досвіду.

• Конкурентоспроможність: У сучасному світі конкуренція в Інтернеті завжди велика. SEO допомагає виділитися серед конкурентів, забезпечуючи кращі позиції в пошукових результатах.

• Ефективність маркетингу: SEO володіє високим рівнем ефективності в порівнянні з іншими видами цифрового маркетингу, зокрема за рахунок великої точності в налаштуванні аудиторії.

Загалом, SEO визнається критично важливим елементом цифрової стратегії будь-якого веб-проекту, оскільки він впливає на його відвідуваність, успішність та взаємодію з аудиторією.

При розробці структури сайту аналітика та відстеження грають важливу роль у забезпеченні ефективності та користувацької дружелюбності. Ці інструменти дозволяють збирати та аналізувати дані, що визначають поведінку користувачів. Далі представлено кілька ключових причин, чому аналітика та відстеження є важливим етапом у створенні ефективної структури сайту [8].

1. Розуміння поведінки користувачів. Аналітика дозволяє відслідковувати, як користувачі взаємодіють з сайтом. Важливо визначити, як вони переходять між сторінками, на що клікають, як довго залишаються, і що саме цікавить або відводить їх увагу. Ця інформація визначає ефективність поточної структури та вказує на можливі покращення.

2. Оптимізація взаємодії. Вивчення аналітики дозволяє ідентифікувати слабкі місця в структурі сайту. Засновані на цих даних зміни можуть бути внесені для покращення навігації та взаємодії з користувачами. Оптимізована структура підвищить ефективність та забезпечить кращий досвід для відвідувачів.

3. Визначення популярних контенту та продуктів. Аналітика вказує на те, який контент або продукти найбільше цікавлять аудиторію. Це може вказати на те, як краще організувати структуру сайту, розміщувати важливий контент та як привертати увагу до ключових елементів.

4. Покращення конверсій. Шлях користувача через сайт може впливати на конверсії. Аналіз відстеження конверсій допомагає зрозуміти, як ефективно розміщені елементи та як можна покращити процес конвертації. Важливо визначити, на яких етапах користувачі можуть відхилятися та внести зміни для оптимізації цих точок.

5. Адаптація до змін. Потреби та поведінка користувачів можуть змінюватися з часом. Аналітичні дані важливі для виявлення цих змін та адаптації структури сайту для відповіді на нові вимоги та очікування аудиторії.

Врахування аналітики та відстеження на ранніх етапах розробки дозволяє побудувати ефективну та користувацько-орієнтовану структуру, що сприяє успіху та задоволенню від користування сайтом.

Отже, було розглянуто важливість та основні можливості структурування продукту.

2.2 Удосконалення методу моніторингу працівників для організації

роботи IT-команд

При розробці засобу який працює з даними людей, потрібно спочатку обрати які саме дані потрібно зберігати. Враховуючи, що додаток розробляється для IT-компаній, потрібно враховувати робочу специфіку і корегувати дані, які потрібно зберігати.

При зберіганні інформації потрібно враховувати GDPR (General Data Protection Regulation). GDPR, або Загальний регламент з питань захисту персональних даних, є нормативним актом, який набрав чинності в Європейському Союзі (ЄС) 25 травня 2018 року. GDPR регулює збір, обробку та зберігання персональних даних громадян ЄС та Європейського Економічного Простору (ЄЕП), а також надає права та контроль користувачам щодо їх особистої інформації [12].

Основні принципи та вимоги GDPR включають:

1. Збір та обробка даних: Організації повинні збирати лише ті персональні дані, які є необхідними для виконання конкретного визначеного ціллю. Обробка даних також повинна здійснюватися легально, справедливо та прозоро.

2. Інформування та згода: Особи, чиї дані збираються, повинні бути належним чином інформовані про цілі та обсяги збору даних. Згода на обробку персональних даних повинна бути вільною, конкретною, інформованою та однозначною.

3. Права осіб: GDPR надає ряд прав особам, чиї дані обробляються, включаючи право на доступ до своїх даних, виправлення неточностей, видалення даних (право на забування), обмеження обробки, перенос даних та право висловлювати свою позицію стосовно автоматизованого прийняття рішень.

4. Безпека даних: Організації зобов'язані забезпечувати адекватний рівень захисту персональних даних від несанкціонованого доступу, втрати, зміни чи розголошення.

5. Обов'язкове повідомлення про порушення: У випадку порушення безпеки, що може призвести до ризику для прав та свобод осіб, організації повинні невідкладно повідомити компетентні державні органи та заінтересовані особи.

6. Відповідальність та облік: Організації повинні дотримуватися принципу обліку та вести документацію щодо своєї відповідальності за виконання вимог GDPR.

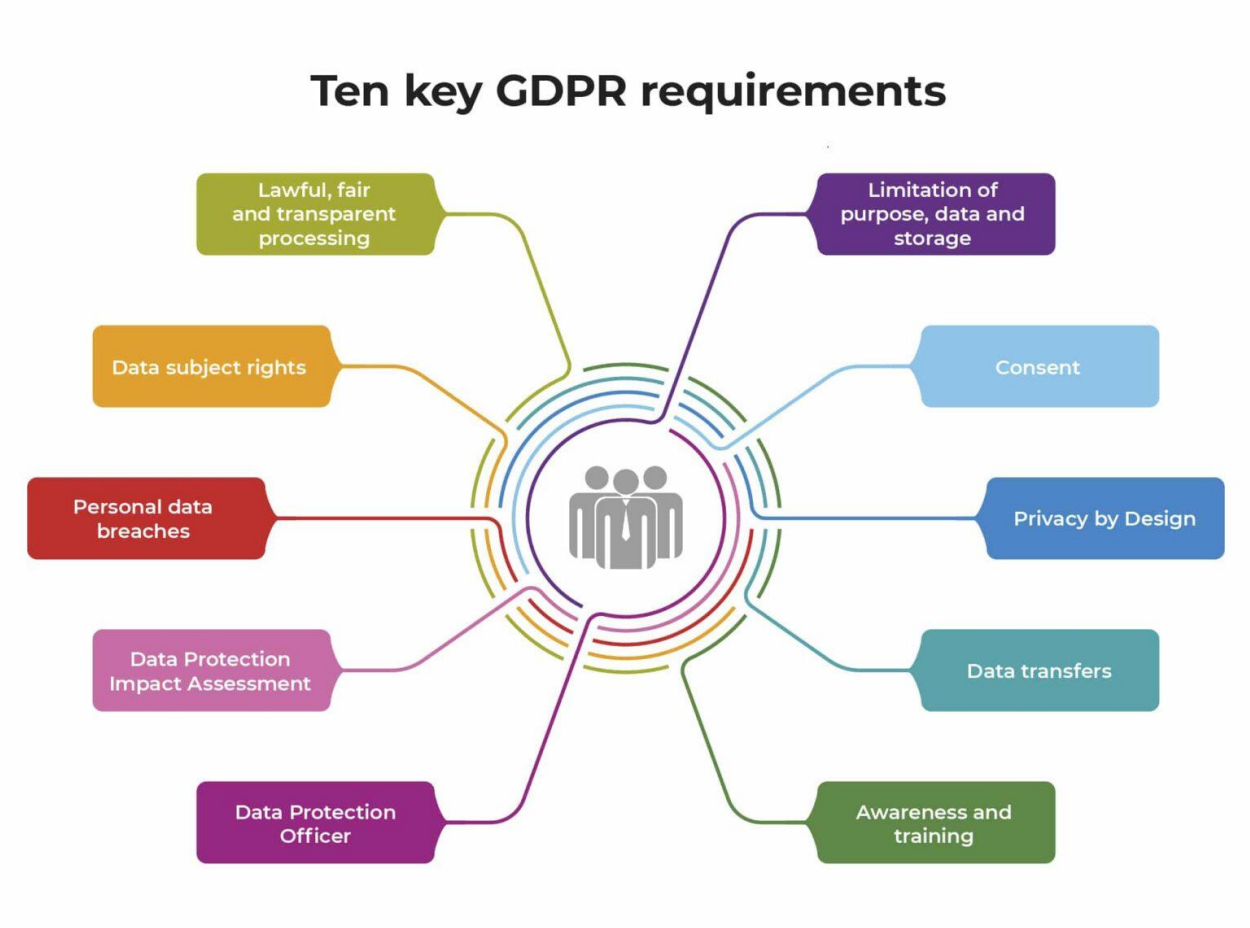

Рисунок 2.1 – Основні принципи та вимоги GDPR включають

GDPR застосовується до всіх організацій, які обробляють персональні дані громадян ЄС, незалежно від того, чи розташована організація в межах ЄС чи поза ним. Цей регламент важливий для захисту приватності осіб та підвищення стандартів безпеки обробки персональних даних.

GDPR встановлює жорсткі стандарти для збору та обробки персональних даних громадян Європейського Союзу. Персональні дані - це будь-яка інформація, що дозволяє ідентифікувати конкретну особу. Зберігання таких даних підпадає під обов'язки та обмеження, визначені GDPR. Ось кілька категорій персональних даних, які можна зберігати в рамках цих вимог:

Основна інформація:

- Ім'я та прізвище.
- Контактна інформація, така як адреса електронної пошти та номер телефону.

Інформація про ідентифікацію:

- Паспортні дані чи інші ідентифікаційні номери.
- Інформація про місце проживання:
- Адреса проживання.

Дані для фінансових операцій:

- Банківські реквізити для проведення транзакцій.

Інформація про роботу:

- Дані про зайнятість та робоче місце.
- Інша особиста інформація:
- Інші дані, які визначають особистість, такі як дата народження, стать, сімейний стан.

Зазначено, що обробка чутливих персональних даних (таких як релігійні переконання, стан здоров'я чи етнічна приналежність) зазвичай заборонена без явної згоди особи. Особливий захист передбачений для даних дітей. Важливо дотримуватися прозорих правил збору та обробки даних, і особи мають мати можливість контролювати використання їх персональних даних.

Кожна організація, яка обробляє персональні дані в межах ЄС, повинна дотримуватися принципів GDPR та враховувати права та інтереси осіб, чиї дані вона збирає та обробляє.

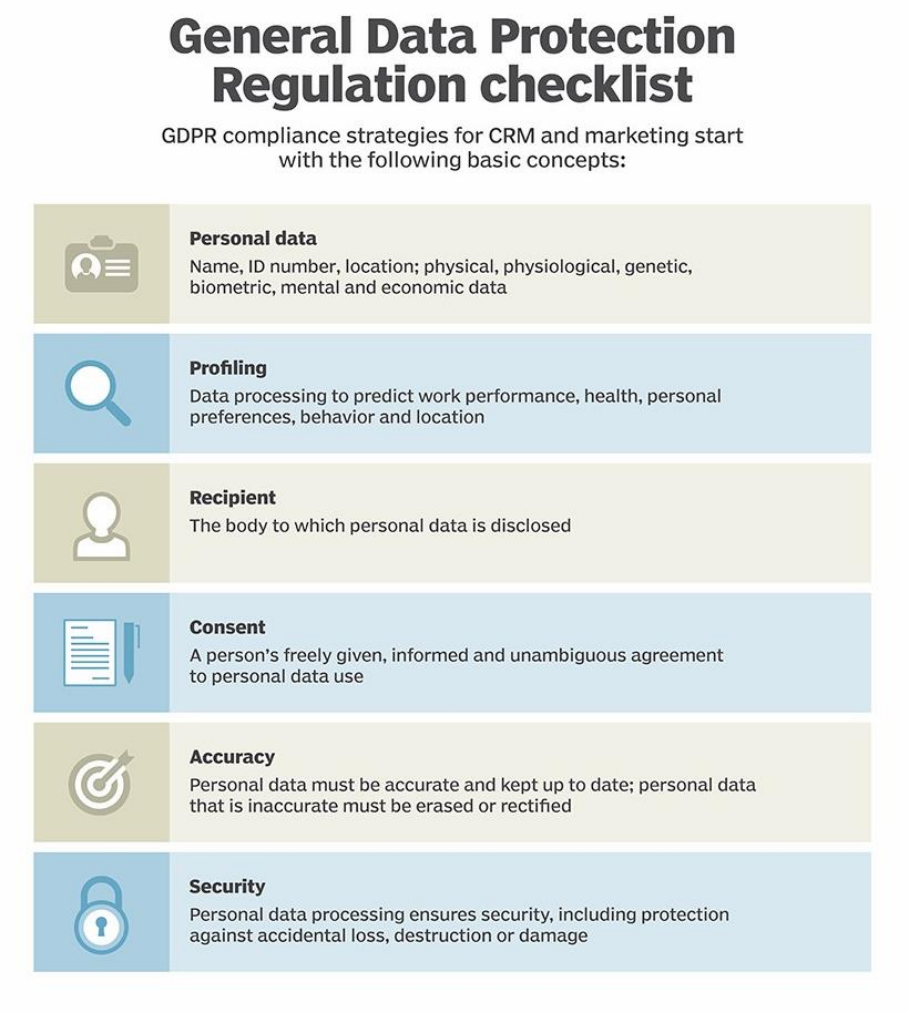

Рисунок 2.2 – Стандарти для збору та обробки персональних даних

GDPR визначає категорії особливо чутливих персональних даних, обробка яких зазвичай заборонена або обмежена без явної згоди особи. Ось кілька основних категорій, які потребують особливого захисту:

- Расова чи етнічна приналежність: Інформація про расу чи етнічне походження особи.
- Політичні погляди: Відомості про політичні переконання або партійні приналежності.
- Релігійні та філософські переконання: Дані, що стосуються релігійних або філософських переконань.
- Членство в професійних асоціаціях: Інформація про членство в професійних, вірменних або відпочинкових організаціях.

- Генетичні дані: Інформація про генетичні особливості особи.
- Стан здоров'я: Дані про стан здоров'я та медичні записи.
- Сексуальна орієнтація та особистість: Деталі щодо сексуальної орієнтації, особистого життя та сексуальних відносин.
- Біометричні дані: Інформація, яка забезпечує ідентифікацію особи за біометричними даними, такими як відбитки пальців чи обличчя.

Ці категорії визначаються як особливо чутливі, і для їх обробки, в тому числі зберігання, потрібно мати згоду особи чи ґрунтовані підстави відповідно до законодавства. Організації повинні утримуватися від зберігання та обробки цих даних без необхідної легальної підстави та дозволу від осіб, яких вони стосуються.

В особистому кабінеті вся інформація буде розміщена по розділам. Розділи будуть наступними:

- особиста інформація;
- робота – інформація про посаду, компенсацію та технології які знає працівник;
- відгуки;
- документи;
- надзвичайні ситуації (контактна інформація близьких працівнику людей);
- бенефіти;
- здоров’я.

Всі дані в профілі працівника може переглядати тільки сам працівник та менеджер, який за нього відповідає, ці люди також можуть редагувати ці дані. Враховуючи, що всі дані про працівників знаходяться в одному місці, це дуже полегшує роботу менеджерів, яким потрібно знати важливу інформацію про працівників.

Для покращення процесу моніторингу працівників до всіх присутніх даних також буде додано рекомендовані ресурси які можуть покращувати

продуктивність працівників. В розділі «Робота» буде додано список курсів або сторінок з корисною інформацію про працівників які можуть допомогти їм розвиватися. Список цих ресурсів буде узгоджуватися з технічним керівництвом команд та вищим менеджментом, щоб максимально ефективно витрачати час людей на самонавчання.

Моніторинг працівників IT-команд буде покращено шляхом введення відгуків працівників один про одного. Відгуки працівників один про одного, також відомі як «360-градусовий зворотний зв'язок» або «міжособистісний фідбек» можуть мати декілька переваг для підприємств та команд.

Компетенція	Мій	Менеджер	GAP
Управління процесами	4	3,8	0,2
Системне мислення	4	3,7	0,3
Роботав команді	4,5	3,7	0,8
Осмисленість	3,5	3,3	0,2
Комунікація	4,5	3	1,5
Клієнтоорієнтованість	3,5	4	-0,5

Рисунок 2.3 – Метод 360-градусного зв’язку

Відгуки колег та підлеглих дозволяють компаніям отримати детальні та об'єктивні відомості про навички своїх працівників. Оскільки вони надходять від тих, хто працює безпосередньо з цими людьми, такі відгуки є важливим інструментом для розвитку і росту працівників. Взаємодія з відгуками працівників може вказати на ті аспекти, де працівники можуть зростати та

розвивати свої навички. Це дозволяє компанії створювати та впроваджувати цільові тренінгові програми для покращення компетентностей свого персоналу. Однією з головних переваг відгуків один про одного є те, що вони надходять від різних джерел, таких як колеги та підлеглі. Це допомагає оцінити не тільки індивідуальні навички, але й взаємодію в команді, сприяючи більш ефективній роботі та покращенню командного духу. Взаємодія з відгуками також дозволяє компанії визначити потенційні лідерські якості серед працівників. Розуміння того, як працівники взаємодіють з іншими, може сприяти розвитку та підтримці лідерів у компанії. Отримання відгуків від колег та підлеглих може позитивно вплинути на мотивацію працівників. Вони відчуваються визнаними за свої досягнення, що сприяє підвищенню задоволеності роботою та більш активній участі в командних завданнях. Важливо, щоб оцінка працівників була об'єктивною.

Оцінюючи їхню роботу з різних точок зору, відгуки допомагають уникнути однобічних оцінок і забезпечити більш реалістичний погляд на ефективність працівників. Відгуки працівників один про одного створюють позитивне середовище, де працівники взаємодіють та висловлюють свої думки. Це підвищує загальну задоволеність роботою та залученість працівників в життя компанії. Важливим елементом є створення відкритої та чесної культури, де відгуки відображають реальну картину роботи. Це сприяє ефективній комунікації та взаєморозумінню всередині організації.

Відгуки будуть виділені як окремий блок в особистому кабінеті працівника, звідти він зможе як писати відгуки про інших так і читати відгуки про себе. Можливість ділитися цією інформацією з менеджерами також допоможе більш розгорнуто бачити вплив працівника на його співробітників і їх відношення до нього.

Дуже важливим фактором є емоційний стан працівників, і так як вигоряння на робочих місцях останнім часом стає все частіше дуже важливо намагатися допомогти працівникам всюди де це можливо. В розділі «Здоров’я» буде додана можливість написати відгук про те, що турбує

працівника. Для цього буде використовуватися той самий модуль який використовується для звичайних відгуків, але він буде направлений тільки до відповідального за працівника менеджера. Далі надана інформація за необхідності буде передана до менеджерів з людських ресурсів компанії щоб допомогти працівнику з його проблемою. Також там будуть присутні ресурси від команди по роботі з людьми щодо покращення самопочуття і робочої етики працівників, щоб спонукати працювати в нормальному режимі і не перепрацьовувати. Для багатьох людей перепрацювання відбуваються на основі стресу, вони можуть хвилюватися що вони не виправдовують очікувань чи думати що вони недостатньо продуктивно працюють, і тому вдаються до того що працюють поза робочим часом щоб доказати свою користь. Відслідкувати це дуже важко, бо такі люди рідко в цьому признаються, але потрібно робити все можливе щоб запобігти людині дійти до такого стану. Якщо все це буде знаходитися в одному застосунку, менеджери зможуть одразу реагувати на такі повідомлення, і вони точно не загубляться в потоці повідомлень в робочих чатах.

Отже, було розглянуто метод моніторингу працівників IT-команд та визначено як його можна покращити.

2.3 Розробка структури інтерфейсу застосунку для організації IT-команд

Розробка структури веб-сайту – це планування та реалізація навігації по сторінках чи блоках веб-сайту. Її метою є спрощення використання сайту користувачем і забезпечення швидкого та зрозумілого переміщення по ньому. Головними вимогами до структури веб-сайту є: назва посилань повинна відповідати сторінкам, на які вони посилаються, блоки повинні бути розташовані в логічній послідовності, зв’язок сторінок між собою має бути зрозумілим, доступ до головних сторінок має бути легким.

До структури веб-сайту також відносять дизайн. Дизайн та стиль інтерфейсу – це ключові аспекти, що визначають вигляд та взаємодію

користувача з продуктом. Колірна палітра впливає на емоційний тон інтерфейсу, а правильно вибрані шрифти додають зручності читанню та сприяють загальному враженню.

Іконки та графічні елементи можуть полегшити сприйняття інформації та додати естетичний аспект. Важливо розміщати їх стратегічно, щоб вони служили інформаційній функції та не заважали користувачеві.

Організація простору грає ключову роль у визначенні читабельності та загального порядку. Використання простору дозволяє уникнути перенасиченості і створити чіткі області, що підкреслюють важливі частини інтерфейсу.

Консистентність – це основний принцип дизайну. Всі елементи інтерфейсу повинні мати однаковий стиль, включаючи кольори, форми та типографію. Це допомагає створити єдиний, логічний та професійний вигляд.

Брендовий дизайн визначається стильовими елементами, характерними для певного бренду. Дотримання цього стилю в інтерфейсі допомагає створити єдиний образ бренду та зберегти його впізнаваність.

Додатково, важливо забезпечити адаптивність інтерфейсу до різних пристроїв. Респонсивний дизайн гарантує, що інтерфейс залишається ефективним та привабливим навіть на пристроях з різними розмірами екранів.

Якщо структура сайту є логічною та зрозумілою, користувач з більшою вірогідністю продовжить використовувати його в майбутньому, а якщо ні, то він просто почне шукати новий, іноді навіть незважаючи на те, що інші можуть мати в собі меншу кількість контенту. Чим більше той, хто використовує продукт, не розуміє, тим швидше росте його незадоволеність, а саме це відбувається якщо структура веб-додатку виконана погано.

Структуру веб-сайту поділяють на 3 види:

1. Лінійна. В цій структурі передбачений перехід між сторінками одна за одною. Відсутність підрозділів та зрозумілість кожного переходу робить цей вид чудовим вибором для малих сайтів, або тих, де інформація йде послідовно, без необхідності створювати додаткові розділи з сторінками.

2. Деревоподібна. Така структура передбачає, що доступ до сторінок здійснюється на різних «гілках», тобто до деяких сторінок користувач може отримати доступ лише якщо він перед цим знаходився на сторінці цієї ж «гілки».

3. Довільна. Доступ до сторінок відбувається в довільному порядку, тобто з головної, користувач може потрапити до найвіддаленішої без необхідності відвідувати перед цим сторінки які логічно зв’язані з нею. Такий підхід зручний, якщо всі сторінки не зв’язані між собою напряму або якщо на сайті міститься багато інформації по різним темам і користувач обирає лише одну, яка йому необхідна.

Для застосунку було обрано деревоподібну структуру, так як більшість дій в ньому відбуваються саме в особистому кабінеті, і більшість сторінок доступні по посиланням саме з нього.

До структури додатку також відноситься його технічна частина. Додаток розроблено за технологією Single page application. SPA (односторінкова програма) – це реалізація веб-програми, яка завантажує лише один веб-документ, а потім оновлює основний вміст цього єдиного документа за допомогою JavaScript API, таких як XMLHttpRequest і Fetch, коли має бути показано інший вміст. Таким чином, це дозволяє користувачам використовувати веб-сайти без завантаження цілих нових сторінок із сервера, що може призвести до підвищення продуктивності та більш динамічного досвіду, з деякими компромісними недоліками, такими як SEO, більше зусиль, необхідних для підтримки стану, реалізації навігації та досягнення значущої продуктивності моніторинг [9].

Розроблену структуру особистого кабінету наведено на рисунку 2.1.

Рисунок 2.4 – Структура особистого кабінету

Для кращого розуміння взаємодії користувачів з застосунком було розроблено UML діаграми. UML (Unified Modeling Language) – це стандартизована мова моделювання, призначена для визначення, візуалізації, будування та документування програмних систем. UML виникла як результат об'єднання кількох методологій моделювання в 1990-х роках та стала широко використовуваною в області розробки програмного забезпечення.

Основна мета UML полягає в тому, щоб створювати єдиний стандарт для визначення моделей програмних систем та сприяти взаєморозумінню між розробниками, аналітиками та іншими учасниками розробки програмного забезпечення. UML надає графічні засоби для створення різних видів діаграм, які відображають різні аспекти програмних систем [10].

Деякі основні види діаграм UML включають:

1. Діаграми використання (Use Case Diagrams): Визначають функціональність системи та взаємодії між системою та її зовнішнім середовищем.

2. Діаграми класів (Class Diagrams): Моделюють структуру системи, її класи, атрибути та взаємодії між класами.

3. Діаграми послідовності (Sequence Diagrams): Показують взаємодії між об'єктами в часі порядку.

4. Діаграми активності (Activity Diagrams): Використовуються для моделювання бізнес-процесів, потоків роботи та діяльності в системі.

5. Діаграми компонентів (Component Diagrams): Показують фізичну структуру системи та взаємодії між її компонентами.

6. Діаграми розгортання (Deployment Diagrams): Моделюють фізичну інфраструктуру та конфігурацію системи в середовищі виконання.

Діаграму прецедентів яка буде описувати можливості звичайних працівників та менеджерів. Вона зображена на рисунку 2.2. В ній можна побачити, що працівник має змогу увійти до свого особистого кабінету де він може взаємодіяти з блоком відгуків, тобто переглядати відгуки про себе та залишати відгуки про своїх колег; переглядати або редагувати свою особисту інформацію або ж через пошук знайти колег і переглянути інформацію про них. Менеджер теж має свій особистий кабінет, але окрім звичайних дій він може переглядати більш детальну інформацію про працівників, таку як відгуки про нього та його показники продуктивності.

Рисунок 2.5 – Діаграма прецедентів

Для кращого відображення роботи програмного засобу було розроблено діаграми послідовностей. Розглянемо дії користувача при виконанні авторизації:

1. Користувач заходить на сайт та авторизується.
2. Веб-сервер надсилає запит на backend.
3. Перевірка існування користувача.
4. Перевірка пароля.
5. Авторизація успішна.
6. Веб-сервер надсилає відповідь користувачеві.

Діаграму діяльності для відображення процесу авторизації представлено на рисунку 2.3.

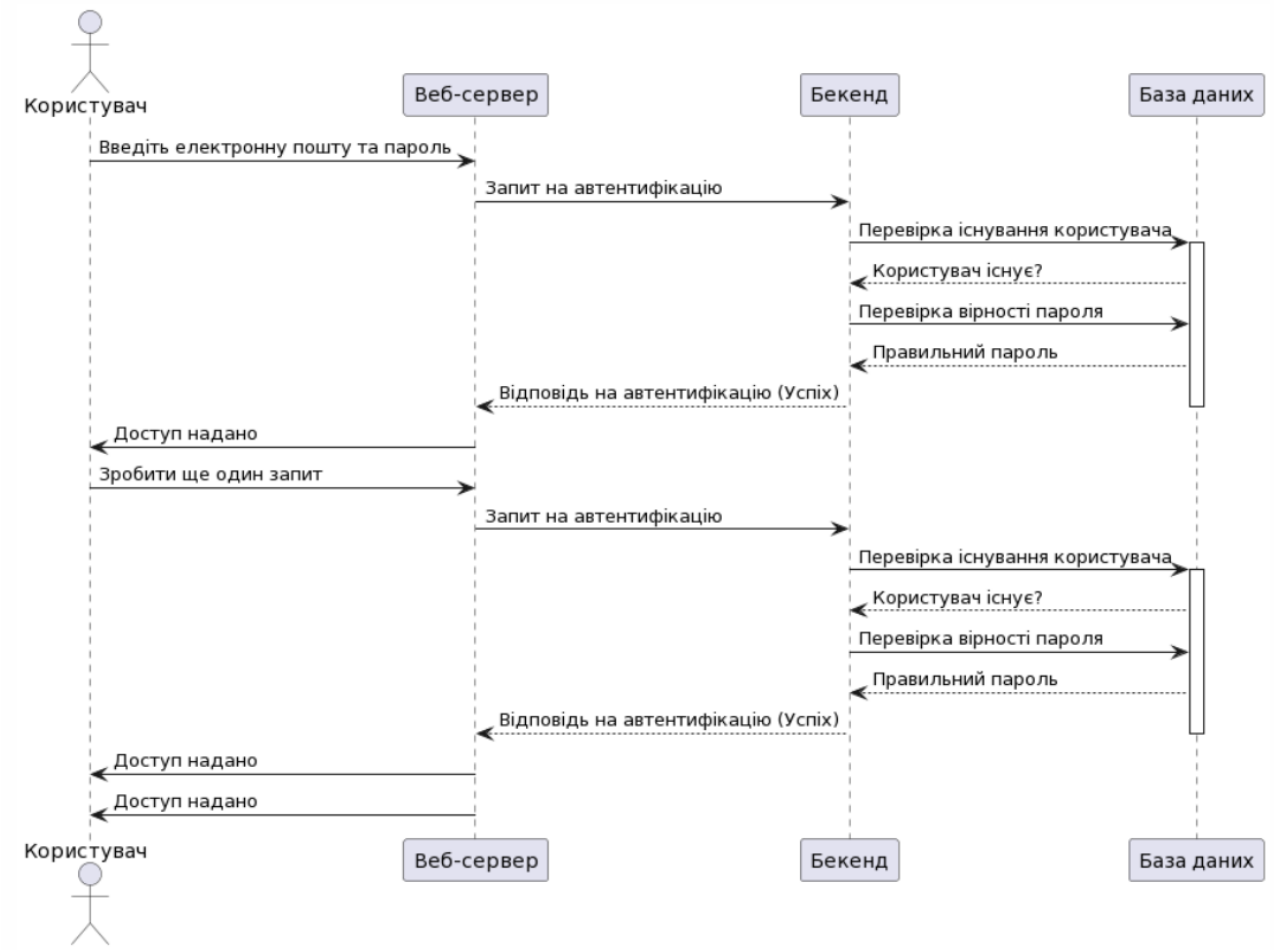

Рисунок 2.6 – Діаграма послідовності авторизації користувача

Також було розроблено діаграму послідовності для створення відгуку. Опис дій для неї виглядає наступним чином:

1. Користувач відкриває свій профіль.

2. Користувач заходить в розділ відгуків.

3. Робиться запит на бекенд, який робить запит на базу даних і користувачу показуються всі відгуки про нього.

4. Користувач натискає кнопку «Написати відгук».

5. Користувач заповнює поля відгуку та відправляє відгук.

6. Робиться запит на бекенд який знаходить користувача якому потрібно записати відгук.

7. Бекенд відправляє запит на базу даних щоб знайти користувача.

8. База даних повертає користувача на бекенд.

9. Бекенд записує відгук користувачу і відправляє ці дані на базу даних.

Діаграму відображено на рисунку 2.4.

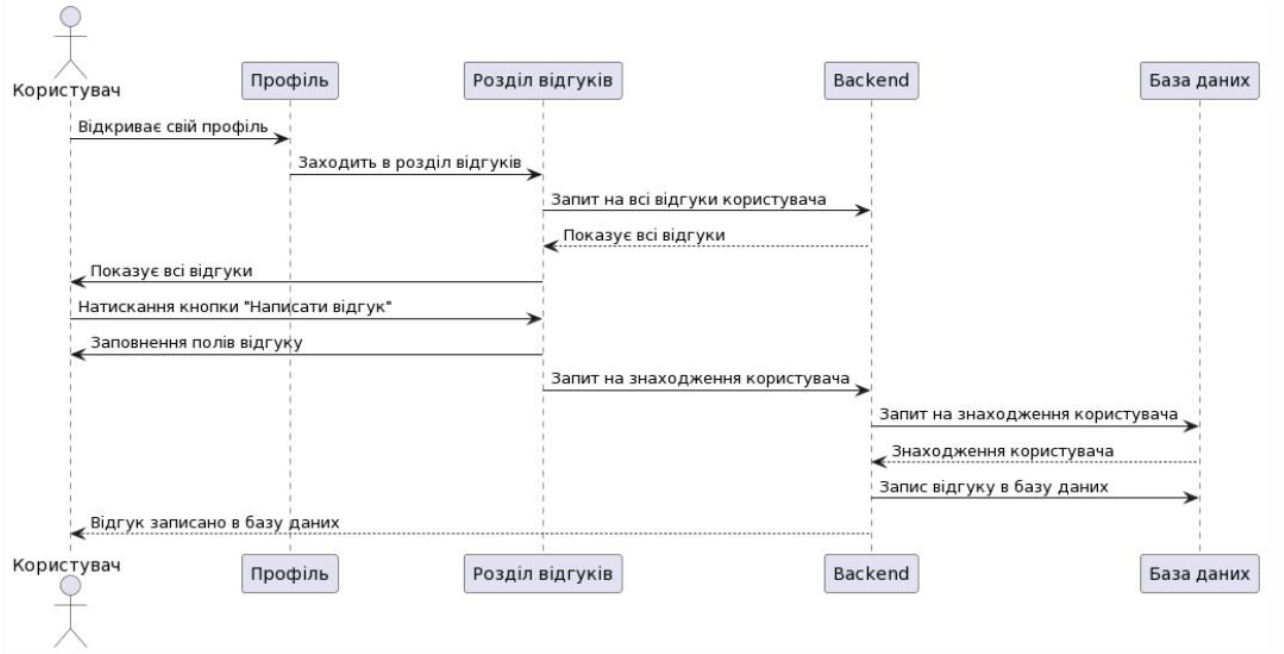

Рисунок 2.7 – Діаграма діяльності створення відгуку

Отже, було проаналізовано структуризацію веб-сторінок та розроблено структуру особистого кабінету працівника для програмного засобу для організації роботи IT-команд.

2.3 Розробка алгоритмів роботи програми

Розробка алгоритмів є необхідною складовою сфери інформатики та комп'ютерних наук. Вона становить фундаментальний етап у процесі створення програм та розробки програмного забезпечення. Важливість цього процесу визначається його впливом на різні аспекти комп'ютерної діяльності.

Алгоритми є своєрідними «рецептами» для вирішення конкретних завдань чи проблем. Вони надають можливість програмістам та розробникам створювати ефективні та оптимальні рішення для різноманітних задач. Вірно спроектовані алгоритми дозволяють не лише досягати бажаних результатів,

але й раціонально використовувати обчислювальні ресурси, такі як час та пам'ять.

У сучасному світі, де швидкість розвитку технологій є ключовою, розробка нових та оптимізація існуючих алгоритмів стають факторами, що визначають конкурентоспроможність програмного забезпечення. Це стосується не лише практичних застосувань, але й наукових досліджень, де розробка нових алгоритмів відкриває нові можливості та сприяє розвитку обчислювальної науки.

Блок-схема – це графічне представлення послідовності кроків або дій в програмі, процесі або системі. Вона використовується для візуального зображення логіки або порядку виконання конкретного завдання.

У блок-схемах кожна дія або операція представлена блоком, який з'єднується з іншими блоками за допомогою ліній. Відповідно до стандартів графічного представлення, різні види блоків та символів використовуються для позначення різних видів дій, рішень, початку чи завершення процесу [11].

Основні елементи блок-схем включають:

- блоки: Представляють конкретні кроки або операції.
- лінії: З'єднують блоки та показують порядок виконання дій.
- ромби: Використовуються для представлення рішень або вибору між двома або більше альтернативами.
- овали або круги: Вказують початок чи кінець процесу.

Блок-схеми є важливим інструментом в розробці програмного забезпечення та інженерії процесів. Вони надають графічне представлення логічної структури та послідовності дій у системі чи програмі, полегшуючи візуалізацію та розуміння комплексних логічних структур та алгоритмів.

Блок-схеми допомагають розробникам швидко зрозуміти процеси та послідовності виконання завдань. Вони служать ефективним інструментом для візуалізації та комунікації логічної структури програми чи системи.

Ці графічні представлення дозволяють легше виявляти потенційні помилки чи невірні послідовності дій. Також, вони стають важливим засобом документування проектів та процесів, сприяючи зрозумілості для розробників та інших учасників проекту.

Блок-схеми також важливі при проектуванні та оптимізації систем. Вони можуть виявити зайву складність, непотрібні етапи, чи можливості оптимізації шляхом зміни логічної структури чи алгоритмів.

Крім того, блок-схеми є ефективним інструментом для навчання та тренування нових членів команди, допомагаючи швидко засвоювати та навчати принципи роботи системи. Усі ці аспекти роблять блок-схеми важливим елементом у досягненні ясності, систематизації та успішної реалізації завдань у процесі розробки та інженерії.

Блок-схеми стають невід'ємною частиною процесу тестування програмного забезпечення. Їх використання дозволяє тестувальникам швидко зрозуміти логіку та послідовність функціональностей, що сприяє більш ефективному тестуванню та виявленню помилок.

Застосування блок-схем допомагає покращити командну роботу. Вони стають чіткими та об'єктивними елементами для обговорення, планування та спільного розуміння завдань. Розробники можуть використовувати блок-схеми для спілкування зі співробітниками, навіть якщо вони не мають глибокого технічного розуміння коду чи системи [11].

Крім цього, блок-схеми можуть слугувати основою для автоматизованої генерації коду або моделювання процесів. Вони можуть бути використані як вихідний пункт для розробки програмного забезпечення та створення деталізованих моделей системи.

Загалом, блок-схеми не лише сприяють ясності та систематизації, але й інтегруються в різноманітні етапи життєвого циклу розробки та управління проектом, покращуючи розуміння, комунікацію та результативність.

Для авторизації користувача було розроблено блок-схему щоб відобразити як саме вона відбувається. Її відображено на рисунку 2.5.

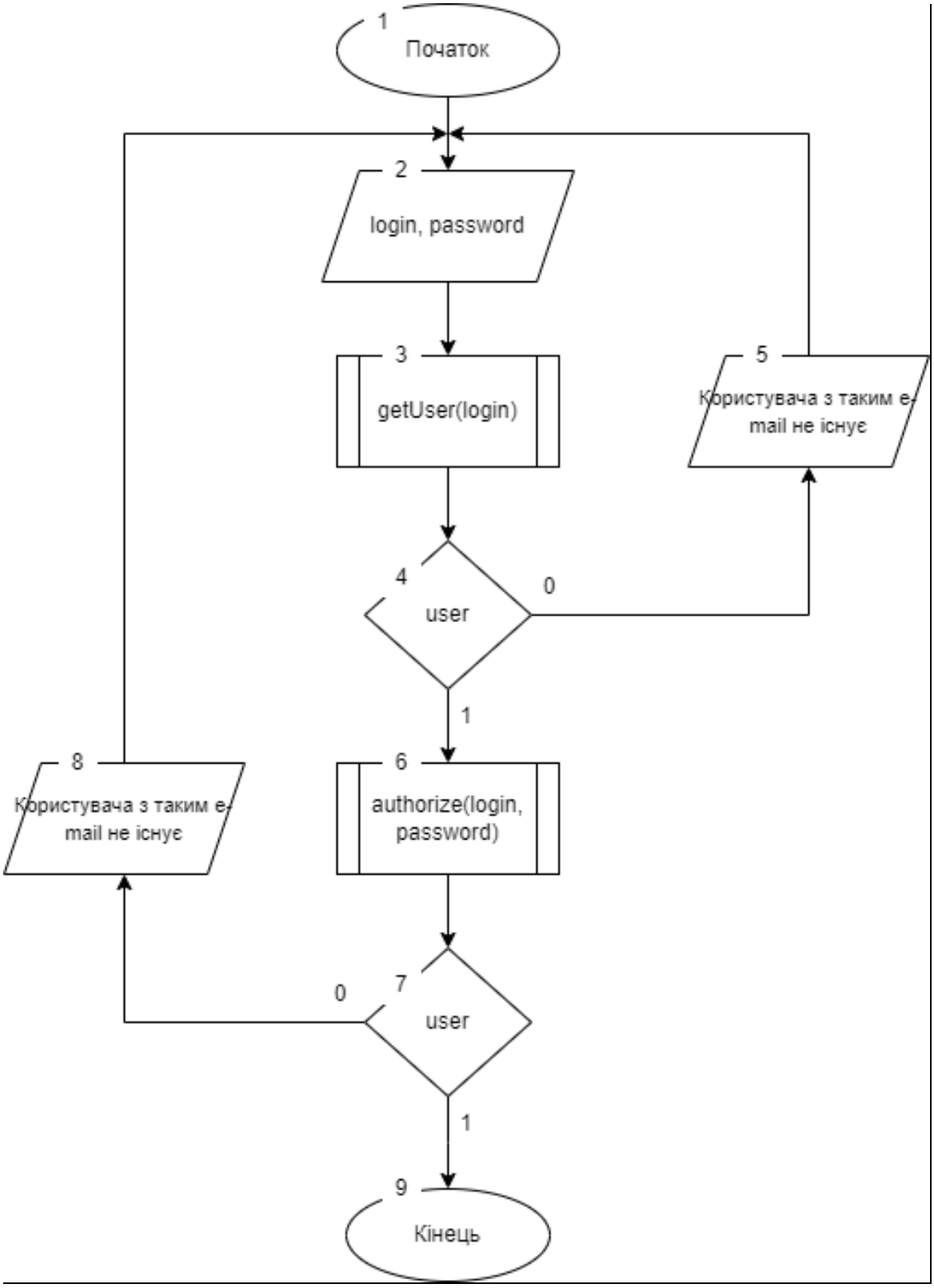

Рисунок 2.8 – Блок схема авторизації користувача

Також було розроблено блок-схему роботи блоку з привітаннями працівників компанії. Цю блок схему відображено на рисунку 2.6.

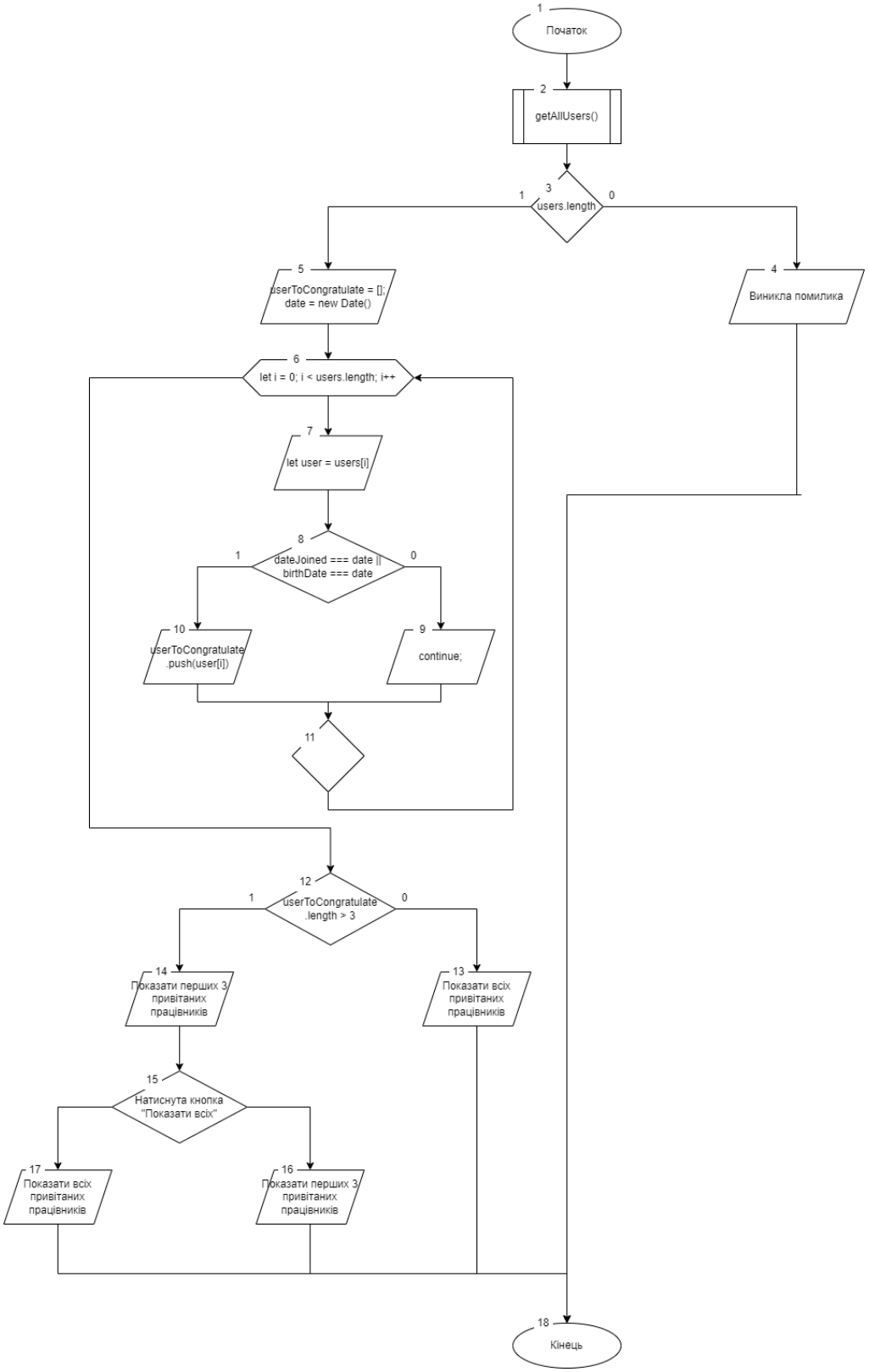

Рисунок 2.9 – Блок схема блоку привітань працівників

Отже, було розглянуто що таке алгоритми та розроблено блок-схеми для блоків програмного засобу.

2.5 Висновки

У другому розділі було проаналізовано підходи до структурування застосунків для організації IT-команд та створено структуру для розроблюваного програмного застосунку. Було проаналізовано можливості удосконалення методу моніторингу працівників IT-команд.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір та опис мови програмування та інструментів розробки

Раніше розробка веб-сторінок переважно покладалася на єдиний стек технологій, оскільки альтернативи вважалися непрактичними або складними для впровадження. У сучасному ландшафті веб-розробка пропонує широкий спектр технологій, і вибір серед них безпосередньо впливає на масштабованість і продуктивність продукту.

Вибраний стек розробки включає HTML для розмітки, каскадні таблиці стилів (CSS) для стилізації, JavaScript з бібліотекою React.js і платформу Node.js. HTML служить мовою для гіпертекстової розмітки, що дозволяє браузерам відображати вміст на сторінці. Його структура повністю складається з тегів, кожен з яких передає певні інструкції браузеру на основі свого імені. Рішення використовувати HTML у розробці веб-системи для вивчення JavaScript було мотивовано його здатністю демонструвати вміст на веб-сторінці [13].

CSS, або каскадні таблиці стилів, використовуються для стилізації HTML-сторінок, дозволяючи повністю змінити зовнішній вигляд сторінки та представити її не просто як простий текст, а в стилізованих блоках. CSS полегшує зміну кольору фону, кольору тексту, відображення вмісту, розміру блоку, форми та інтерактивної поведінки, включаючи анімацію. Вибір використання CSS виправданий його здатністю посилити залучення користувачів, оскільки сторінка, що складається виключно з тексту, може не привернути увагу користувача, тоді як стилізована сторінка, швидше за все, приверне її [14].

Щоб покращити використання CSS, був використаний препроцесор SCSS. SCSS, мета-мовний синтаксис SASS з можливістю сценаріїв, перетворює написаний код у стилі CSS. Його переваги включають можливість вкладати правила одне в одне, що робить написання коду більш простим;

створення змінних для часто використовуваних правил, сприяючи розумінню коду; і встановлення спеціальних змінних із набором правил, які можна повторно використовувати, усуваючи надмірність коду [15].

Для компіляції коду SCSS у стилі CSS потрібен компілятор. Для цієї мети було використано Webpack, що дозволяє специфікувати правила для остаточного зовнішнього вигляду продукту. Окрім компіляції стилів, Webpack може об’єднати кілька файлів JavaScript в один, дозволяючи писати окремі модулі коду з іменами файлів, що описують їхній вміст, і збирати їх в один файл для кінцевого продукту [16].

JavaScript, мова програмування високого рівня, вдихає життя у веб-сторінки, дозволяючи додавати новий вміст, оновлювати наявний вміст, обробляти введені користувачем дані у формах і керувати застосуванням стилів до блоків сторінок. Без JavaScript сторінка була б обмежена пасивним переглядом, і єдиною взаємодією, доступною для користувачів, був би перехід на інші сторінки за допомогою посилань. Рішення використовувати JavaScript випливає з його незамінності для взаємодії з користувачем, наприклад для навігації по розділах, збереження прогресу навчання, реєстрації та різноманітних інших функцій [17].

React.js є бібліотекою JavaScript, розробленою Facebook, яка використовується для створення динамічних та ефективних інтерфейсів веб-застосунків. Однією з ключових особливостей React є використання віртуального DOM, що сприяє оптимізації швидкості відображення змін, дозволяючи ефективно оновлювати тільки необхідні частини сторінки.

Основний принцип React – це компонентний підхід, коли інтерфейс розбивається на невеликі, незалежні компоненти, що полегшує розробку та підтримку коду. Ця архітектурна модель робить код більш зрозумілим та легко відтримуваним.

React широко використовується для створення односторінкових застосунків, де зміни відбуваються динамічно без перезавантаження сторінки. Популярність React обумовлена не лише його функціональністю, але й

широкою спільнотою, що забезпечує доступ до великої кількості ресурсів та підтримки. Його оголошені практики програмування та активна екосистема роблять його потужним інструментом для розробників, спрощуючи створення високоякісних та масштабованих веб-застосунків [18].

Node.js – це платформа яка дозволяє виконувати JavaScript скрипти на сервері та відправляти користувачеві результат їх виконання. Завдяки їй, можна розробляти серверну частину сайту на JavaScript, тобто можна обробляти форми, працювати з базами даних, надсилати користувачу HTML сторінки тощо. Було вирішено використовувати цю платформу через можливість використання мови Javascript для серверної частини, найголовніше для зв'язку з базою даних та обробкою форм [19].

Сервер безпосередньо взаємодіятиме з базою даних, проводитиме обробку інформації та повертатиме результати клієнту у форматі JSON. JSON (JavaScript Object Notation) - це текстовий формат обміну даними, заснований на JavaScript. Він володіє властивістю легкості читання як для людини, так і для машини і часто використовують у REST API, переважно над XML.

Це універсальний формат, який сприяє ефективній передачі даних між сервером та клієнтом, надаючи зручність читання та обробки. JSON дозволяє структурувати дані у вигляді пар ключ-значення, що полегшує їх інтерпретацію та використання у різноманітних програмах та платформах.

Безперечно, використання JSON у взаємодії сервера та клієнта ускладнюється мінімальним обсягом даних та забезпечує швидкий обмін інформацією. Це особливо актуально в архітектурі REST API, де легкість взаємодії та швидкість обробки даних визначають ефективність та продуктивність веб-системи.Усі дані програми зберігаються у базі даних, яка, в свою чергу, є основою інформаційної системи. База даних (БД) – це організована структура, призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів. Головною перевагою БД є швидкість внесення та використання необхідної інформації. Завдяки

спеціальним алгоритмам, що використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд.

Також в БД існує певна взаємозв'язок інформації: зміна в одному рядку може призвести до змін в інших рядках, що допомагає працювати з інформацією простіше і швидше. Тому теорії створення та практиці використання баз даних приділяється достатня увага протягом періоду функціонування інформаційних систем.

В якості бази даних було обрано Firebase. База даних Firebase Realtime – це база даних, розміщена в хмарі. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. При створенні міжплатформних програм за допомогою платформ Apple, Android і JavaScript SDK, усі клієнти спільно використовують один екземпляр Realtime Database і автоматично отримують оновлення з найновішими даними [20].

Отже, було проведено варіативний аналіз і обґрунтовано вибір засобів для реалізації програмного засобу для організації роботи IT-команд.

3.2 Розробка інтерфейсу програмного засобу

Інтерфейс (або інтерфейс користувача, UI) – це просторова площина, на якій взаємодіють користувачі та комп'ютерні програми. Це область, де відбувається обмін інформацією між користувачем і системою. В контексті веб-сайтів інтерфейс – це спосіб, за допомогою якого користувачі взаємодіють з веб-сайтом, включаючи елементи дизайну, навігаційні можливості, форми і т. д.

Важливість хорошого інтерфейсу веб-сайтів важко переоцінити. Ось деякі аспекти, які роблять його важливим:

1. Перші враження. Ваш інтерфейс – це перше, що бачить користувач під час візиту на веб-сайт. Чим приємніше та інтуїтивно зрозуміле перше враження, тим більше шансів, що відвідувач залишиться та досліджуватиме ваш вміст.

2. Навігація. Інтерфейс повинен забезпечувати зручну та легку навігацію. Чітка структура меню, логічне розташування кнопок і посилань допомагають користувачам швидше знаходити необхідну інформацію.

3. Дизайн. Естетика грає важливу роль. Привабливий та сучасний дизайн може збільшити визначеність бренду та зробити взаємодію з веб-сайтом більш приємною.

4. Доступність. Інтерфейс повинен бути доступним для різних користувацьких груп, включаючи людей з обмеженими можливостями. Доступність включає в себе врахування різних видів користувацьких потреб та можливостей.

5. Швидкість та ефективність. Швидкий та ефективний інтерфейс дозволяє користувачам швидше досягати своїх цілей. Оптимізований інтерфейс покращує загальний досвід користувача.

6. Взаємодія. Взаємодія користувачів з веб-сайтом повинна бути інтуїтивно зрозумілою. Елементи управління, такі як кнопки, форми та меню, повинні бути чітко виділені та легко розпізнаватися.

В цілому, вдалий інтерфейс веб-сайту сприяє позитивному враженню користувачів, забезпечує їм зручну навігацію та стимулює взаємодію з вмістом, що може позитивно позначитися на репутації вашого бренду та задоволенні користувачів [21].

Інтерфейс веб-сайту зазвичай складається з різних частин, кожна з яких виконує свою функцію для поліпшення користувацького досвіду. Основні частини веб-інтерфейсу включають:

- хедер (Верхня Частина);
- меню та Навігація – меню навігації зазвичай розташоване в хедері або на його близькому відстані. Це може бути горизонтальним чи вертикальним списком посилань на ключові розділи веб-сайту.;
- основний Вміст;

• бічна Панель (Сайдбар) – деякі веб-сайти використовують бічну панель для відображення додаткової інформації, посилань на розділи, тегів чи інших корисних елементів. Це може бути розташоване з боку або з обох боків вмісту.;

• підвал (Футер);

• форми та віджети – форми для зворотного зв'язку, підписки на розсилки, пошукові віджети та інші елементи, які сприяють взаємодії користувача з веб-сайтом.;

• попапи та сповіщення – можуть використовуватися для виведення важливої інформації, збору даних або надання користувачеві певних опцій.;

• контекстні Меню;

Ці частини спільно працюють для створення повного інтерфейсу веб-сайту, який має бути як зручним, так і ефективним для користувача, сприяючи навігації та взаємодії з контентом.

Для початку було розроблено верхню частину сайту. Верхня частина веб-сайту, або «Хедер» – верхня частина сторінки, яка містить ключову інформацію та елементи навігації. Зазвичай включає логотип бренду, головне меню, пошукову строку та інші важливі елементи. Верхня частина виконує важливу роль у створенні першого враження та спрощенні доступу користувачів до ключових розділів сайту. Її дизайн повинен бути привабливим та функціональним, забезпечуючи зручну навігацію та консистентність із загальним стилем веб-сайту. Розроблена верхня частина сайту представлена на рисунку 3.1.

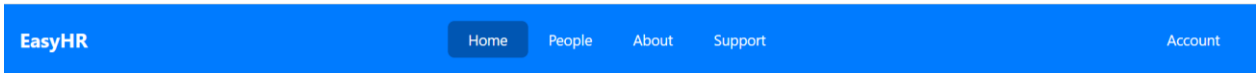

Рисунок 3.1 – Верхня частина сайту

Також було розроблену нижню частину сайту, або його «Підвал». В ньому звичайно відображається логотип сайту та інформація про розробника

або більше інформації про сам ресурс. Нижню частину сайту відображено на рисунку 3.2.

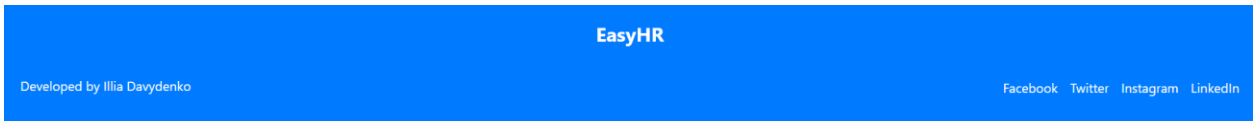

Рисунок 3.2 – Нижня частина сайту

На головній сторінці та сторінці з профілем користувача зліва буде присутній блок з річницями чи іншими подіями колег працівника. Цей блок відображено на рисунку 3.3.

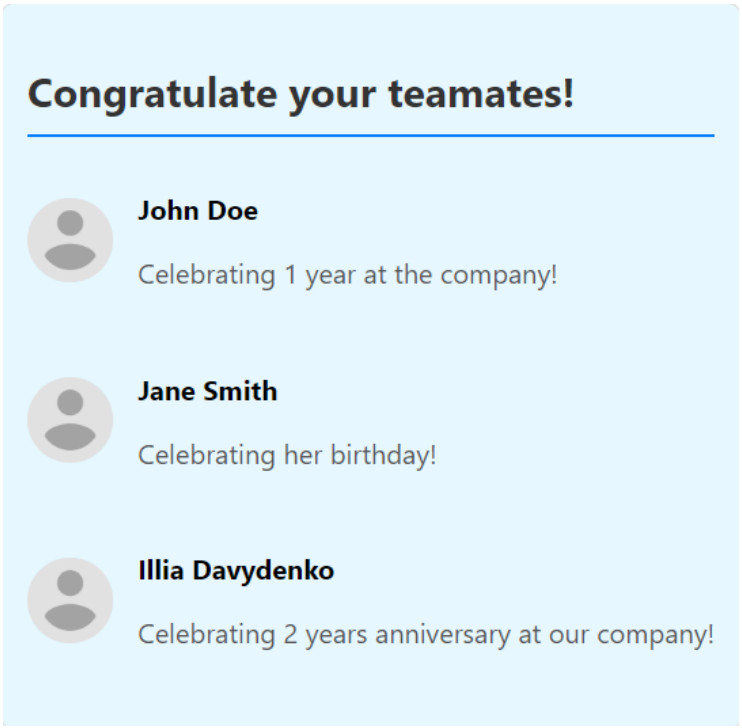

Рисунок 3.3 – Блок вітань

Було розроблено інтерфейс профілю користувача. В ньому присутня інформація про самого працівника, навігація по різним блокам профіля та блок з привітаннями для колег під фотографією. Вся інформація в розділі особистої інформації може бути змінена користувачем, після внесення змін

вони відправляються на сервер і зберігаються там дозволяючи зберегти зміни. Інтерфейс цього блоку представлено на рисунку 3.4.

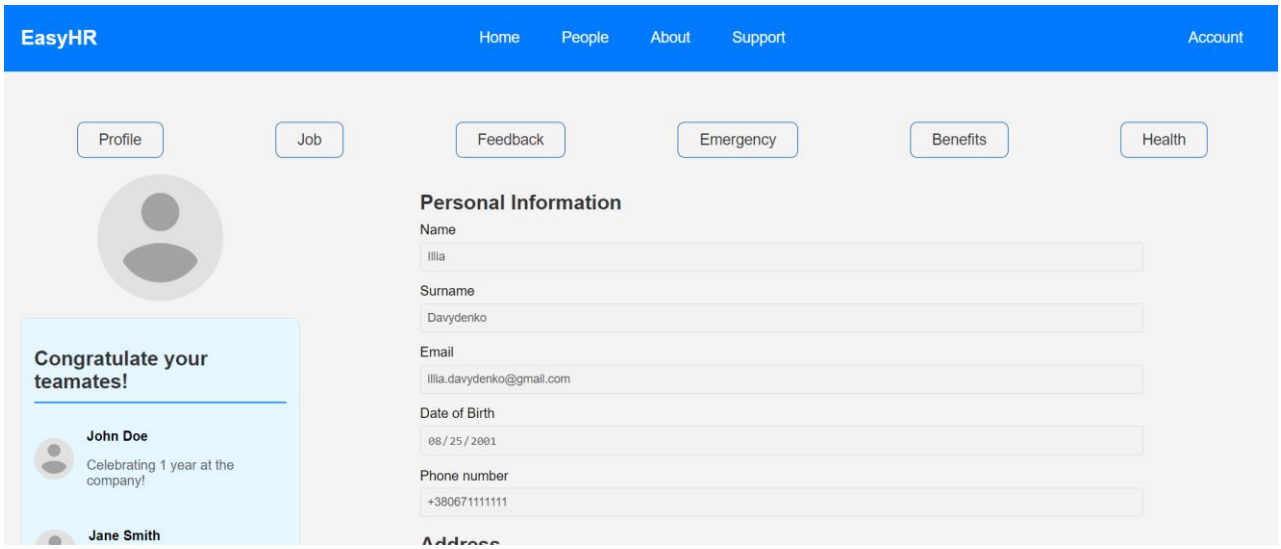

Рисунок 3.4 – Блок персональної інформації користувача

Отже, було розглянуто важливість інтерфейсу в додатках та розроблено інтерфейс програмного засобу для організації роботи IT-команд.

3.3 Реалізація алгоритму роботи програми

Щоб почати роботу з програмним засобом спочатку потрібно пройти авторизацію. Авторизація – це процес визначення та контролю прав доступу користувачів до конфіденційної інформації чи функціоналу системи. Вона включає ідентифікацію користувача, перевірку облікових даних, визначення прав доступу та можливість управління цими правами. Забезпечує безпеку та конфіденційність даних, а також обмежує доступ до ресурсів відповідно до повноважень користувача. Авторизація – це власне дозвіл, який визначає можливості користувача у взаємодії з системою. Для ефективної авторизації часто використовуються різноманітні методи аутентифікації, такі як паролі, двофакторна аутентифікація, відбитки пальців або розпізнавання обличчя. Комбінація цих методів забезпечує надійний рівень захисту від несанкціонованого доступу. Вона важлива для підтримання приватності та

безпеки, адже дозволяє ефективно керувати тим, хто має доступ до чутливої інформації. Вікно авторизації відображене на рисунку 3.5.

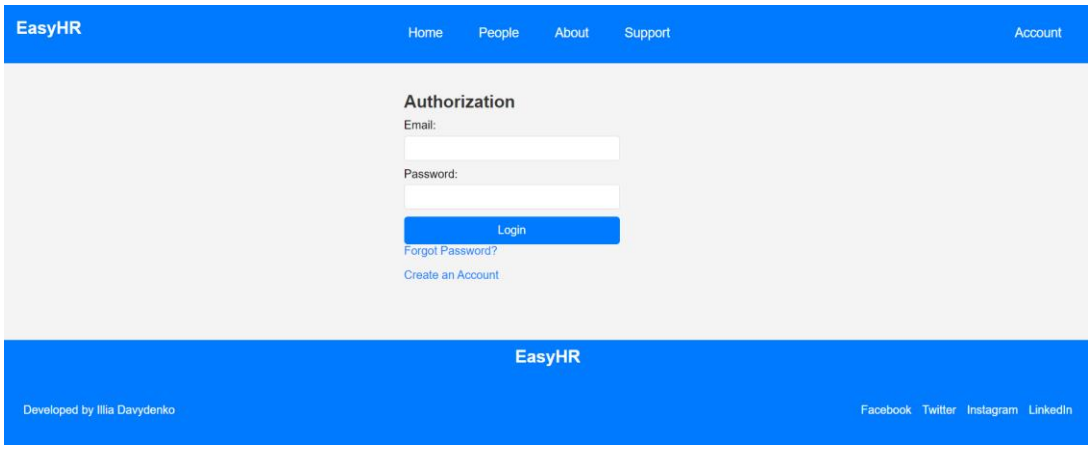

Рисунок 3.5 – Блок авторизації

Якщо користувач вводить правильні дані, він зможе користуватися застосунком, а якщо дані були неправильними (користувач з такою поштою не існує або неправильний пароль) то йому буде видано помилку з проханням перевірити його дані (див. рис. 3.6). Для перевірки даних посилається запит на сервер який перевіряє чи існує користувач по ключу (полю пошти) в базі даних. Після перевірки користувач бачить результат.

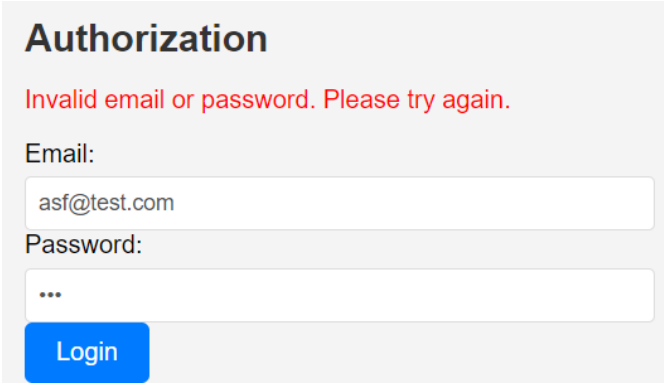

Рисунок 3.6 – Помилка при неправильному введенні даних

Для обміну даними між клієнтом та сервером використовуються HTTP запити. HTTP – це протокол передачі гіпертексту, що використовується для

комунікації між веб-браузерами та веб-серверами. Він визначає, як повинні взаємодіяти клієнт і сервер під час обміну даними. HTTP базується на моделі "запит-відповідь", де клієнт висилає запит на сервер, а сервер відправляє відповідь. В основі HTTP лежать методи запитів, такі як GET (отримання ресурсу), POST (створення ресурсу), PUT (оновлення ресурсу), DELETE (видалення ресурсу) та інші. Кожен запит і відповідь мають заголовки, які містять інформацію про запит чи відповідь. REST (Representational State Transfer) та HTTP (Hypertext Transfer Protocol) є двома тісно пов'язаними поняттями в області розробки веб-сервісів та взаємодії між клієнтом і сервером.

REST є архітектурним стилем для розробки мережевих аплікацій та веб-сервісів. Введений Ройсом Філдінгом у 2000 році, REST надає принципи для створення легких, масштабованих та ефективних веб-сервісів.

Основні принципи REST включають:

1. Ресурси:Все у світі REST є ресурсами, які можуть бути доступні або взаємодіяти між собою. Кожен ресурс має унікальний ідентифікатор (URI).

2. Представлення:Ресурси можуть мати різні представлення (наприклад, JSON, XML), залежно від потреб клієнта.

3. Стан: Клієнт та сервер можуть взаємодіяти без збереження стану між ними.Кожен запит від клієнта містить всю необхідну інформацію для обробки запиту на сервері.

4. Єдність інтерфейсу – Всі ресурси використовують спільний інтерфейс для спрощення взаємодії та зменшення складності.

5. Система Кешування: REST може використовувати систему кешування для підвищення ефективності взаємодії між клієнтом і сервером.

REST може використовуватися поверх протоколу HTTP, тож терміни часто використовуються разом, і RESTful веб-сервіси використовують HTTP як свій протокол комунікації.

Однією з переваг REST є його безстійність. Кожен запит вважається самодостатнім, не залежить від попередніх запитів, та містить всю необхідну

інформацію для обробки сервером. Це робить систему більш простою та ефективною.

Гнучкість та легкість розширення є іншими перевагами REST, що роблять його популярним вибором для веб-сервісів. Крім того, стандартизований підхід до взаємодії між клієнтом та сервером сприяє створенню розширюваних та добре архітектурно задокументованих систем.

Для того, щоб можна було перевірити існування корисутвача в базі даних, потрібно зробити на неї запит через сервер. Перед тим як робити запит в базу даних, потрібно мати ендпоінт на сервері на який можна буде давати запити. Спочатку потрібно підключитися до бази даних по даним які можна отримати в профілі проекту бази даних Firebase. Після підключення можна писати будь-який ендпоінт, по якому можна буде надсилати запити з клієнтської частини. Програмну реалізацію ендпоінта з отриманням даних користувача відображено на рисунку 3.8.

```
const express = require('express');
const admin = require('firebase-admin');

const app = express();
const port = process.env.PORT || 3001;

const serviceAccount = require('./easyhr-53d27-firebase-adminsdk-icl0r-e16277d7d4.json');
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://easyhr-53d27-default-rtdb.europe-west1.firebasedatabase.app/',
});

app.get('/getUser/:userId', async (req, res) => {
  const userId = req.params.userId;

  try {
    const snapshot = await admin.database().ref(`users/${userId}`).once('value');
    const user = snapshot.val();
    res.json(user);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Рисунок 3.8 – Ендпоінт отримання даних користувача

Після створення серверу, можна виконувати запити з клієнтської частини на сервер і отримувати дані про користувачів. Для цього була написана функція обробник яка буде викликатися на відправленні форми. В ній робиться виклик на бекенд який повертає користувача з бази даних. Після цього перевіряється Її реалізація представлена на рисунку 3.9.

```
24  const handleLogin = async (e) => {
25      e.preventDefault();
26      let user = null;
27
28      try {
29        const response = await fetch(`http://localhost:3001/getUser/${email}`);
30
31        if (!response.ok) {
32          setError('Invalid email or password. Please try again.');
33        }
34
35        user = await response.json();
36
37        if (password !== user.password) {
38            setError('Invalid email or password. Please try again.');
39        }
40
41      } catch (error) {
42        setError(error.message || 'Error during login.');
43      }
44
45      return user;
46    };
```

Рисунок 3.9 – Програмна реалізація запиту на сервер для отримання даних користувача

Для пошуку людей в системі або відображенню важливих дат колег необхідно мати можливість отримувати всіх працівників з бази даних. Для цього було реалізовано новий ендпоінт який дозволяє отримувати всіх користувачів при звернені. Завдяки цьому, можна буде отримувати дані для того щоб, наприклад знаходити важливі події для блоку з привітаннями колег. Його реалізацію представлено на рисунку 3.10.

```
app.get('/getAllUsers', async (req, res) => {
    try {
      const snapshot = await admin.database().ref('users').once('value');
      const users = snapshot.val();

      res.json(users);
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Internal Server Error' });
    }
  });
```

Рисунок 3.10 – Реалізація ендпоінта отримання всіх користувачів

Клієнтська частина може бути дуже великою по кількості коду, так як часто існують речі, які присутні на багатьох сторінках і є однаковими для них.

Клієнтська частина сайту це та частина, яку бачить користувач. Вона відіграє велику роль бо сам від неї формується перше враження про сайт, а вже потім користувач починає думати за його швидкодію та функціонал. Для спрощення написання коду в React.js існує можливість створення компонентів які можна перевикористовувати і інших місцях. Для того щоб використати компонент в іншому компоненті потрібно імпортувати його в нього та просто використати як звичайний HTML тег. Для прикладу розглянемо компонент NavigationItem який використовується для більшості посилань на веб-сайті (див. рис. 3.11).

```
import React from 'react';
import { Link } from 'react-router-dom';
import './NavigationItem.css';

const NavigationItem = ({ to, children }) => {
  return (
    <Link to={to} className="navigation-item">
      {children}
    </Link>
  );
};

export default NavigationItem;
```

Рисунок 3.11 – Компонент NavigationItem

Розглянемо розроблений модуль надання відгуків. На сторінці відображаються відгуки які вже отримав користувач та можливість створити свій відгук при натисканні на кнопку. Після натискання буде відправлено запит на сервер де береться запит на базу даних, отримується користувач, йому в поле відгуків додається новий відгук. Після записання нового відгука користувачу, це знову записується в базу даних щоб потім відобразити їх на сторінці. Сторінка відгуків представлена на рисунку 3.12

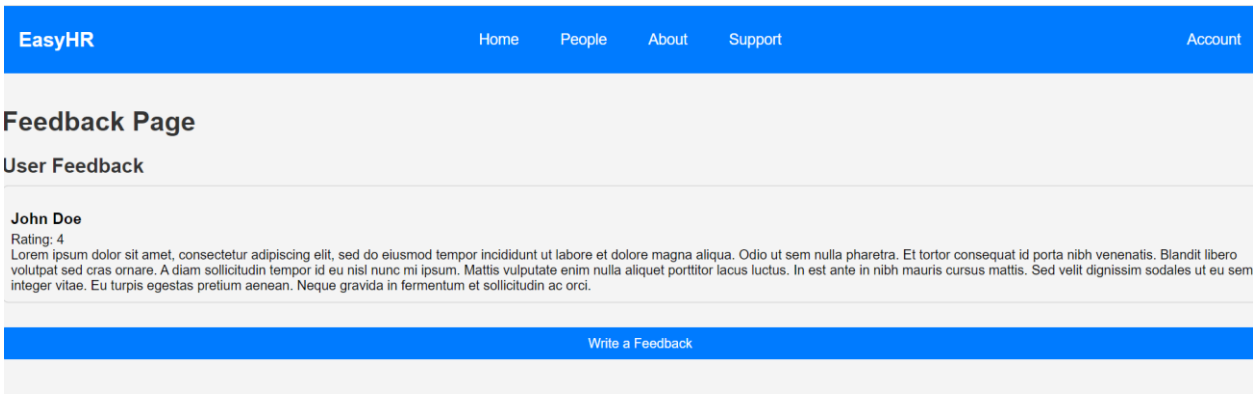

Рисунок 3.12 – Сторінка відгуків працівника

Реалізацію ендпоінту на сервері представлено на рисунку 3.13. При запиті на нього з клієнтської частини він запише дані користувачу в його профіль на базі даних. Цей код реалізує серверний ендпоінт для обробки HTTP POST-запитів за шляхом /sendFeedback/:userEmail. Його основна мета - додавати відгук до масиву feedbacks користувача в базі даних за його електронною адресою. Отримання параметрів запиту userEmail: Електронна адреса користувача, для якого додається відгук.feedbackData: Дані відгуку, які надходять у тілі POST-запиту. Запит до бази даних Firebase здійснюється за допомогою orderByChild та equalTo для пошуку користувача за його електронною адресою.Якщо користувач існує, отримується його ID та посилання (userRef) на відповідний об'єкт користувача.Масив feedbacks користувача оновлюється, додаючи новий відгук, який надходить з тіла POST-запиту.Оновлений об'єкт користувача з новим відгуком зберігається в базі

даних. Якщо користувач існує, сервер повертає статус 200 та об'єкт { success: true }.Якщо користувач не знайдений, сервер повертає статус 404 та об'єкт { error: 'User not found' }. В разі помилки сервер повертає статус 500 та об'єкт { error: 'Internal Server Error' }.

```
app.post('/sendFeedback/:userEmail', async (req, res) => {
  const userEmail = req.params.userEmail;
  const feedbackData = req.body;

  try {
    const snapshot = await admin.database().ref('users').orderByChild('email').equalTo(userEmail).once('value');

    if (snapshot.exists()) {
      const userId = Object.keys(snapshot.val())[0];
      const userRef = admin.database().ref(`users/${userId}`);

      const user = snapshot.val()[userId];
      user.feedbacks.push(feedbackData);

      await userRef.set(user);

      res.status(200).json({ success: true });
    } else {
      res.status(404).json({ error: 'User not found' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```

Рисунок 3.13 – Реалізація ендпоінту надсилання фідбеку на базу даних

Отже, було розглянуто розробку алгоритму роботи програмного засобу для оптимізації для організації роботи IT-команд.

3.4 Висновки

У третьому розділі було було проведено варіативний аналіз і обґрунтовано вибір засобів і технологій для реалізації програмного засобу для організації роботи IT-команд. Було розглянуто важливість інтерфейсу програмних засобів та розроблено інтерфейс для програми. Було розглянути програмну реалізацію алгоритмів роботи додатку.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування – це ітеративний процес перевірки та оцінки програмного забезпечення з метою гарантування його якості та правильного функціонування. Ця систематична діяльність включає в себе виконання програм чи систем з метою виявлення помилок та перевірки, чи вони відповідають вимогам [22].

Важливість тестування охоплює виявлення помилок на ранніх етапах розробки, забезпечення високої якості продукту, підтримку його надійності та визначення відповідності вимогам. Тестування також допомагає зменшити вартість виправлення проблем пізніше, поліпшує взаємодію з користувачем, забезпечує безпеку та підвищує довіру до продукту.

Автоматизація тестування, крім того, покращує ефективність та дозволяє швидше реагувати на зміни. Тестування продуктивності, безпеки, сумісності, оновлень та міграцій, відмовостійкості – це додаткові аспекти, що гарантують повноту та ефективність тестового процесу.

Функціональне тестування – це процес, під час якого перевіряється, чи виконує програма свої функції відповідно до визначених вимог. Під час функціонального тестування фахівці оцінюють, як елементи програми взаємодіють між собою та чи відповідає їхнє функціонування очікуваним результатам. Основна мета - переконатися, що програма працює так, як очікується від її користувачів та замовників.

Нефункціональне тестування – це оцінка аспектів програмного продукту, які не є його прямими функціями. Одним із ключових напрямків нефункціонального тестування є тестування продуктивності. В рамках цього виду тестування оцінюється швидкість роботи програми, її витрати пам'яті та здатність витримувати велике навантаження. Проведення тестів на швидкодію та реакцію системи на інтенсивність використання допомагає визначити,

наскільки продукт є ефективним та надійним в умовах реального використання. Окрім цього, нефункціональне тестування може включати в себе перевірку безпеки продукту, його сумісність з іншими системами, а також відмовостійкість. Сумісність програми з іншими системами також є ключовим аспектом. Переконання у тому, що програма працює на різних платформах та інтегрується з іншими програмами, забезпечує безперебійну роботу та задоволення користувачів.Всі ці аспекти важливі для створення повноцінного та конкурентоспроможного програмного продукту [22].

Важливість тестування включає:

- виявлення та усунення помилок на ранніх етапах розробки, що зменшує вартість та час, необхідний для виправлення проблем пізніше;
- забезпечення високої якості продукту, гарантуючи, що він відповідає вимогам та очікуванням користувачів;
- надійність програмного продукту являється ключовою для його успіху, і тестування дозволяє перевірити його працездатність та стійкість;
- тестування перевіряє, чи відповідає програма вимогам та функціональності, встановленим на етапі розробки;
- виявлені та виправлені на ранніх етапах помилки коштують дешевше, ніж ті, що виявлені після випуску продукту;
- тестування сприяє легкому розумінню та зручності використання продукту для користувачів;
- тестування виявляє потенційні вразливості та забезпечує безпеку програмного продукту;
- успішне тестування підвищує довіру користувачів та клієнтів до програми чи системи.

Тестування білого ящика (White Box Testing):

Під час тестування білого ящика розробник тестує програмне забезпечення з доступом до вихідного коду. Цей вид тестування часто використовується для модульного тестування, коли окремі частини системи

перевіряються на працездатність та стійкість. Розробник тестує код, пов'язаний з бібліотеками тестованого програмного забезпечення. Такий підхід дозволяє аналізувати правильність внутрішніх структур даних і виконувати тестування на ранніх етапах розробки. Однак цей метод вимагає від розробника знань програмування для розуміння коду.

Тестування чорного ящика (Black Box Testing):

Під час тестування чорного ящика тестувальник працює з програмою лише через її зовнішній інтерфейс, не знаючи деталей внутрішньої реалізації. Тестувальник базується на специфікаціях або інших документах, що описують вимоги до системи. Цей метод включає функціональне та нефункціональне тестування. Тестування чорного ящика полегшує створення тест-кейсів без прив'язки до конкретної специфікації і може бути проведене на різних етапах розробки. Однак цей метод не дозволяє вичерпно тестувати всі можливі шляхи виконання програми.

Тестування сірого ящика (Gray Box Testing):

Тестування сірого ящика є комбінацією методів білого та чорного ящика. Розробник тесту має обмежений доступ до вихідного коду програми, але тестування проводиться головним чином з позиції користувача. Такий підхід дозволяє аналізувати внутрішню структуру та алгоритми ПЗ для написання ефективних тест-кейсів, при цьому саме тестування проводиться з точки зору користувача.

Таким чином, враховуючи частковий доступ до вихідного коду, метод сірого ящика поєднує переваги інших двох методів, забезпечуючи максимальну ефективність тест-кейсів і покращуючи загальний рівень тестування.

4.2 Тестування розробленого програмного продукту

Було проведено тестування профілю користувача. В базі даних знаходиться запис про користувача з всіма даними про нього. При переході на

сторінку профілю робиться запит на сервер, який в свою чергу має робити запит на базу даних і отримувати дані про працівника. Після цього ці дані мають бути відображені на сторінці. На рисунку 4.1 відображений запис про працівника в БД.

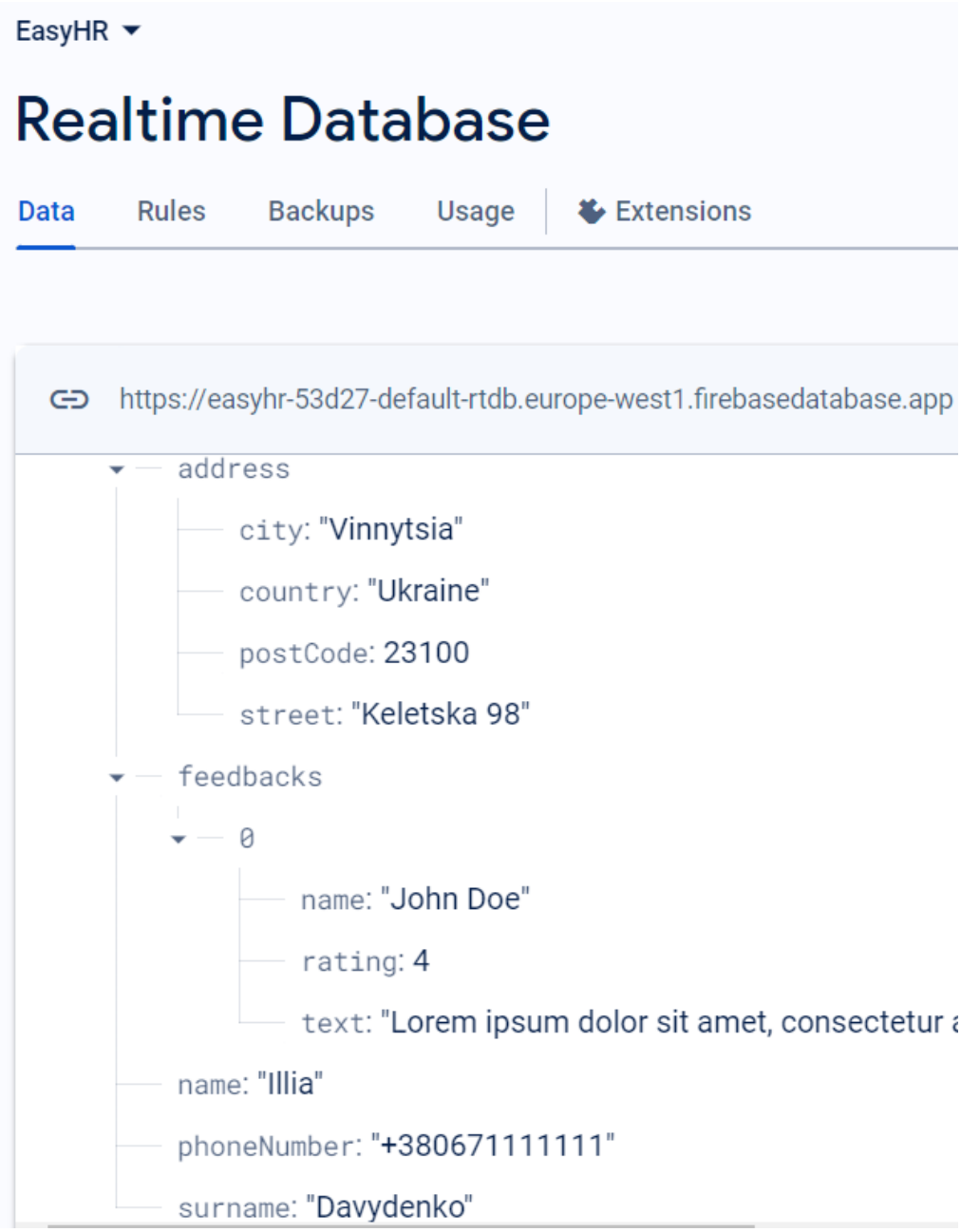

Рисунок 4.1 – Запис користувача в базі даних

На сторінці профілю повинні відображатися дані з бази даних. А при зміні цих даних в базі даних чи на сторінці профілю, дані повинні зберігатися в БД. Сторінка профіля з даними відображена на рисунку 4.2.

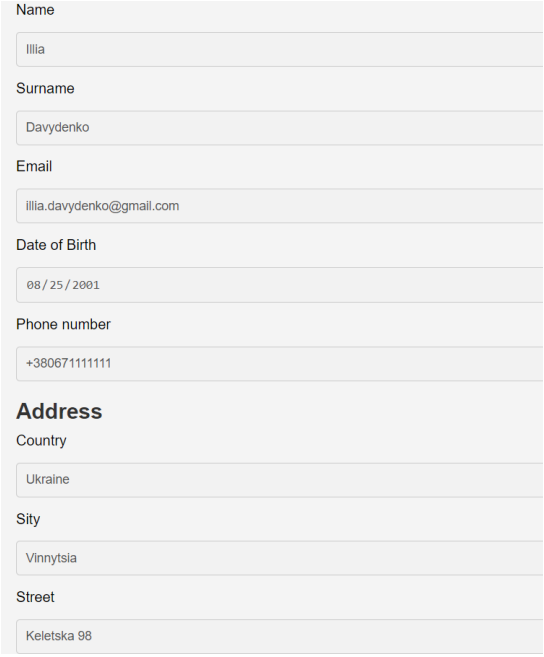

Рисунок 4.2 – Дані в профілі з бази даних

Блок з привітаннями відображає привітання в залежності від того чи є якісь річниці в колег. Наприклад, якщо в працівника з БД дата початку роботи в компанії співпадає з сьогоднішньою датою, то його відобразить в цьому блоці. Наприклад, в базі даних існують три користувачі в яких в сьогоднішній день річниця (див. рис. 4.3).

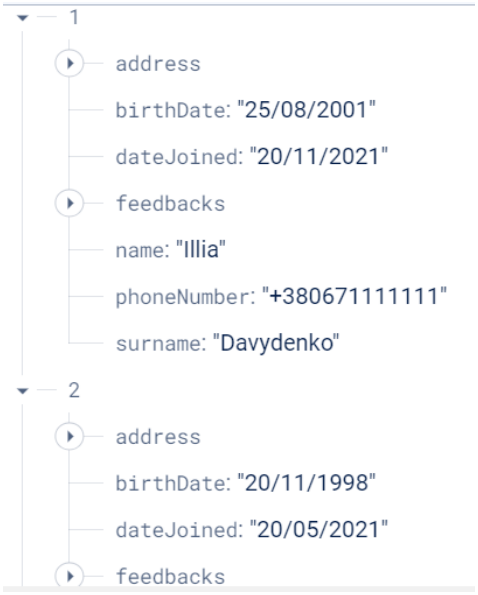

Рисунок 4.3 – Записи користувачів з річницями

Оскільки ми маємо користувачів з річницями в сьогоднішню дату, вони будуть відображені в блоці привітань (див. рис. 4.4). В залежності від того, яка саме в них річниця, буде відображено різне повідомлення. Таким чином при річниці в компанії буде привітання з річницею, при дні народження з днем народження, при першому дні в компанії теж є окреме привітання.

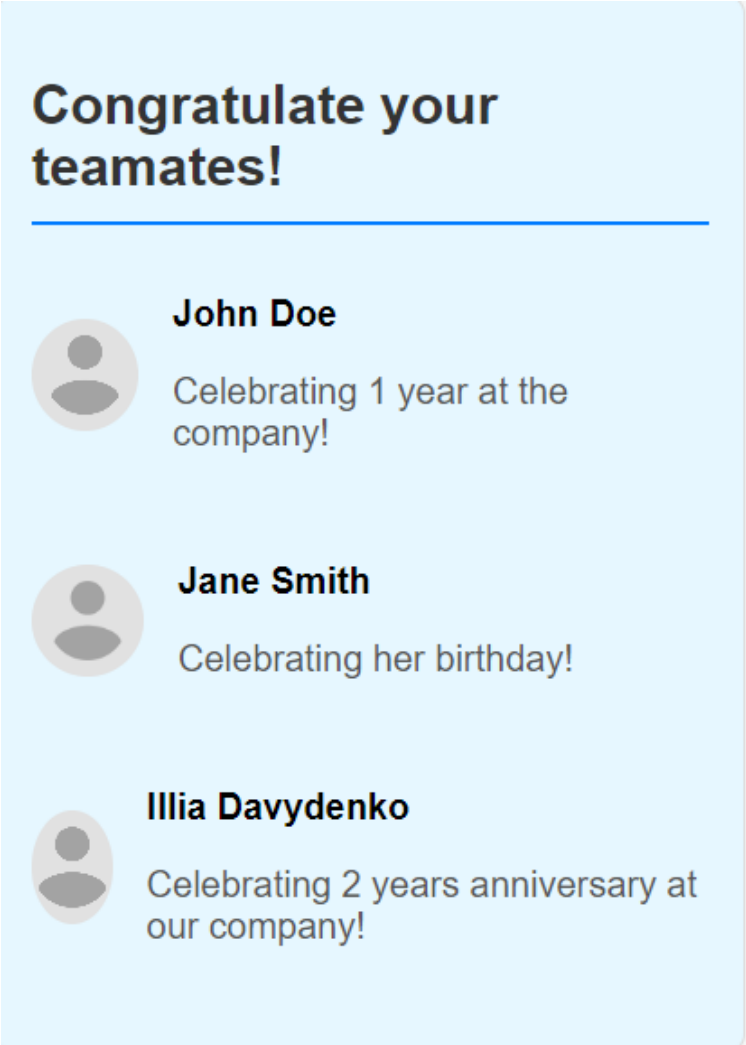

Рисунок 4.4 – Привітальний блок з користувачами з БД

Сторінка «Люди» має відображати всіх користувачів з можливістю пошуку. При початковому заході на сторінку робиться запит на сервер, який повертає на клієнтську частину користувачів які присутні в базі даних, а потім з отриманого списку формуються блоки з користувачами на сторінці. Сторінку з працівниками відображено на рисунку 4.5.

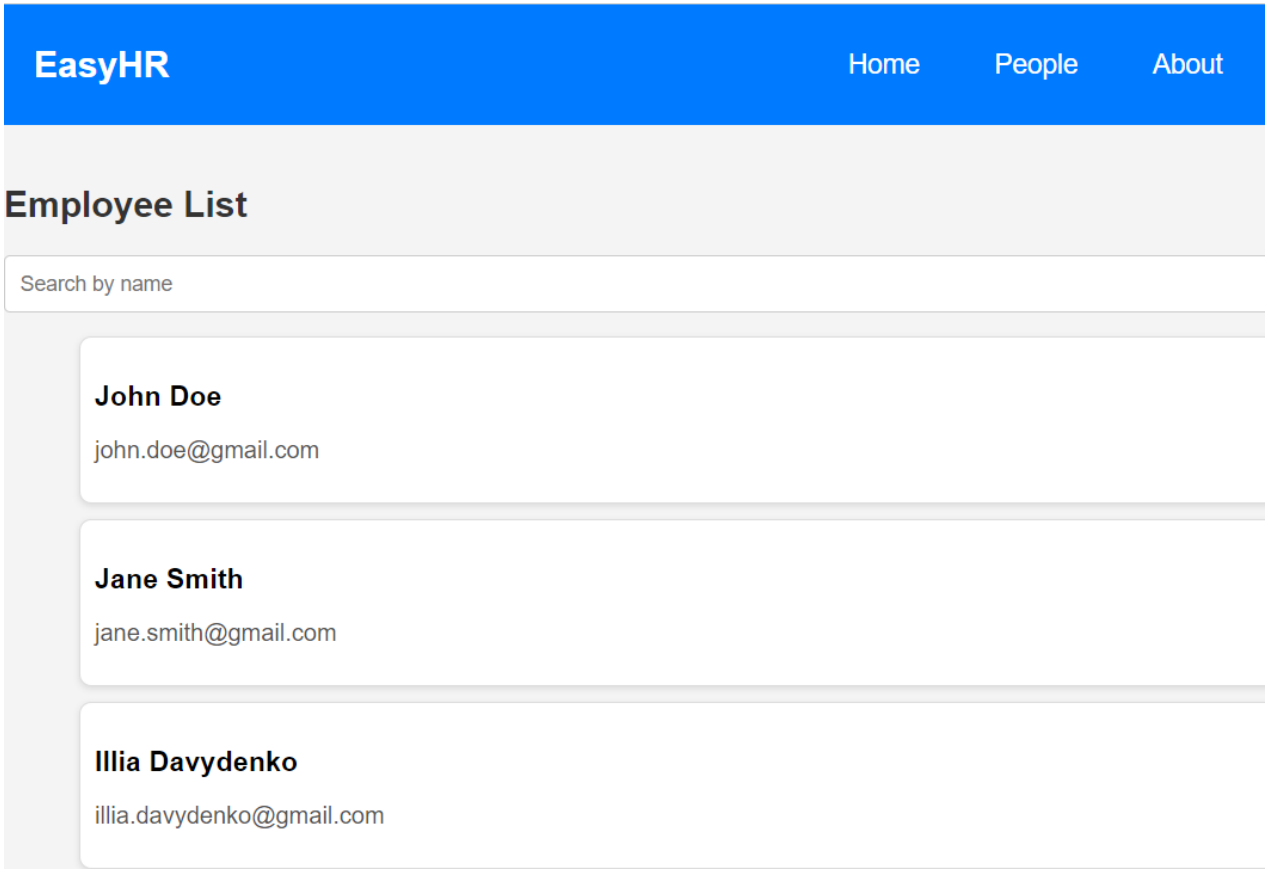

Рисунок 4.5 – Блок з відображенням всіх працівників

Якщо ввести ім’я користувача в стрічку пошуку то в списку відобразяться тільки користувачі з таким іменем (див. рис. 4.6).

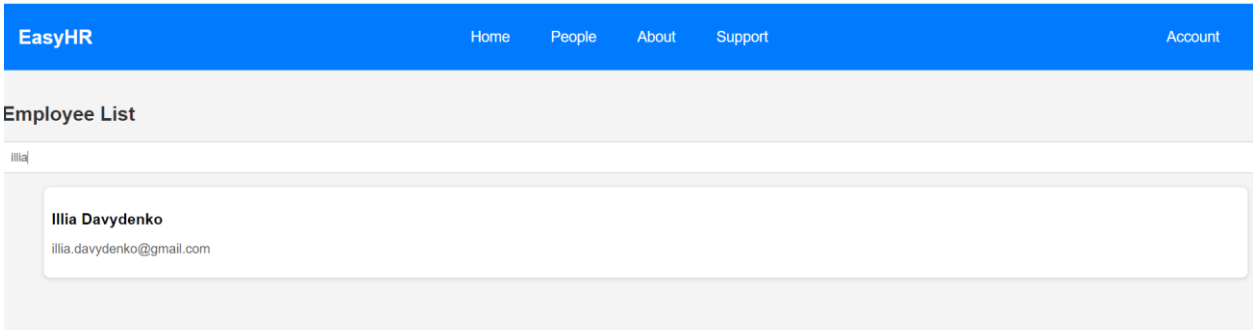

Рисунок 4.6 – Результат пошуку працівників по імені

Для більш детального тестування блоку відгуків, для нього було розроблено тест-кейси. Тест-кейси є важливою складовою процесу тестування програмного забезпечення. Вони визначають конкретні сценарії

тестування, які дозволяють перевірити різні аспекти програми. Кожен тест-кейс містить докладний опис тестового випробування, його умови та очікувані результати.

Основна мета тест-кейсів – це забезпечити повноту та систематичність тестування. Вони визначають, які аспекти програми слід перевірити, і як саме це треба робити. Крім того, вони можуть бути використані для автоматизації тестів, спрощуючи процес виконання повторюваних тестових сценаріїв.

Створення ефективних тест-кейсів вимагає ретельного розуміння вимог до програми та можливих сценаріїв використання. Кожен тест-кейс має бути простим у розумінні та виконанні, а його результати мають бути чітко інтерпретовані. Це допомагає забезпечити якість програмного продукту та раціонально використовувати ресурси для проведення тестування.

Було розроблено такі тест-кейси:

Сценарій 1: Перегляд відгуків

1. Користувач відкриває блок відгуків.

2. Перевіряється наявність відгуків.

3. Відгуки відображають ім'я автора, оцінку та текст відгуку.

4. Впевнитися, що відгуки відображаються вірно та відсортовані за датою (найновіші вгорі).

Сценарій 2: Залишення відгуку

1. Користувач натискає кнопку "Написати відгук".

2. Відкривається модальне вікно для написання відгуку.

3. Користувач вводить ім'я, оцінку та текст відгуку.

4. Після відправки перевіряється, що відгук з'явився у списку відгуків.

Сценарій 3: Відмова від відгуку без вводу імені отримувача

1. Користувач натискає кнопку "Написати відгук".

2. Відкривається модальне вікно для написання відгуку.

3. Користувач не вводить ім’я отримувача, але вказує текст відгука та оцінку.

4. При спробі відправити відгук перевіряється, що відгук не додався, і

користувач отримує повідомлення про необхідність введення імені.

Сценарій 4: Відмова від відгуку без вводу тексту

5. Користувач натискає кнопку "Написати відгук".

6. Відкривається модальне вікно для написання відгуку.

7. Користувач не вводить текст відгуку, але вказує ім'я та оцінку.

8. При спробі відправити відгук перевіряється, що відгук не додався, і користувач отримує повідомлення про необхідність введення тексту.

Сценарій 5: Відмова від відгуку без вказівки оцінки

1. Користувач натискає кнопку "Написати відгук".

2. Відкривається модальне вікно для написання відгуку.

3. Користувач вводить текст відгуку та вказує ім'я, але не вказує оцінку.

4. При спробі відправити відгук перевіряється, що відгук не додався, і користувач отримує повідомлення про необхідність вказання оцінки.

Сценарій 6: Відгук із максимальною кількістю символів

1. Користувач натискає кнопку "Написати відгук".

2. Відкривається модальне вікно для написання відгуку.

3. Користувач вводить текст відгуку з максимальною кількістю символів.

4. Перевіряється, що введений великий обсяг тексту правильно обробляється і відгук успішно додається.

Сценарій 7: Оцінка за межами діапазону

1. Користувач вводить текст відгуку, вказує ім'я, але вибирає оцінку, що не входить у діапазон від 1 до 5.

2. При спробі відправити відгук перевіряється, що відгук не додався, і користувач отримує повідомлення про некоректну оцінку.

Спочатку блок відгуків було протестовано за звичайним позитивним сценарієм. В блоці відгуків можна залишати відгуки про інших працівників. Для цього потрібно зайти в свій аккаунт, зайти в розділ «Відгуки» та залишити відгук про працівника. Після цього відгук буде додано в дані користувача в базі даних і він буде відображатися в цього працівника. Для того щоб

протестувати цей функціонал спочатку потрібно зайти в профіль іншого користувача. Щоб це виконати потрібно зайти в вікно авторизації і ввести дані працівника (див. рис. 4.7). Якщо дані були введені правильно, то працівник успішно зайде в свій аккаунт. Якщо ж дані введені неправильно, йому буде відображено помилку з повідомленням що його дані неправильні та запропоновано ввести дані знову.

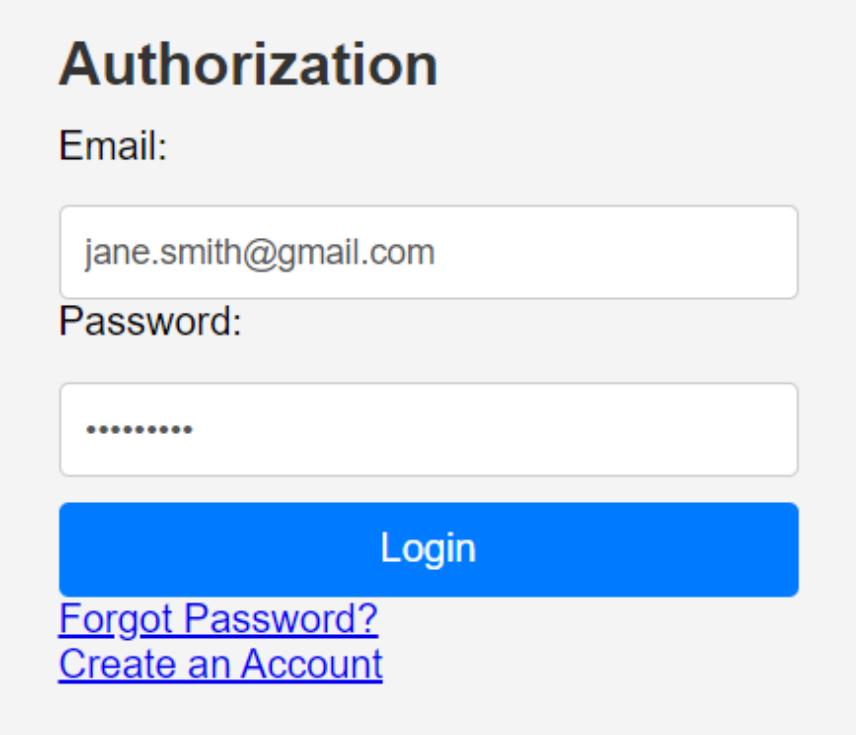

Рисунок 4.7 – Авторизація під іншим працівником

Після авторизації потрібно зайти в розділ відгуків, натиснути на кнопку «Написати відгук», обрати користувача якому буде написано цей відгук і відправити його (див. рис. 4.8). Якщо були введені всі поля, то відгук буде відправлено і додано користувачу якому його було відправлено. Після цього можна буде перевірити чи отримав інший користувач відгук і чи оновився його запис в базі даних.

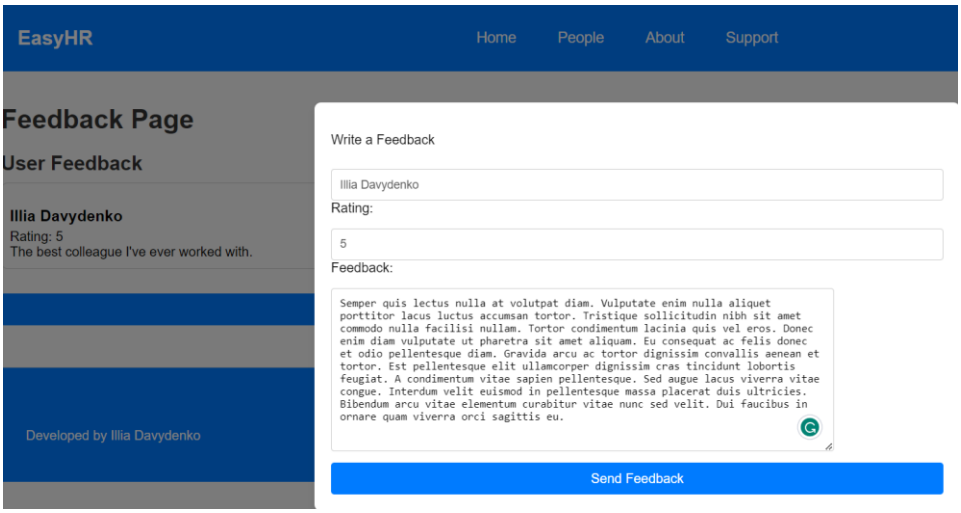

Рисунок 4.8 – Написаний відгук перед відправленням

Після того як було відправлено відгук, в користувача, якому його було відправлено, оновився запис в БД (див. рис. 4.9).

Рисунок 4.9 – Оновлений запис користувача в базі даних

Після цього можна перейти в профіль користувача якому було відправлено відгук, там повинен знаходитися новий відгук. Результат тестування відображено на рисунку 4.10.

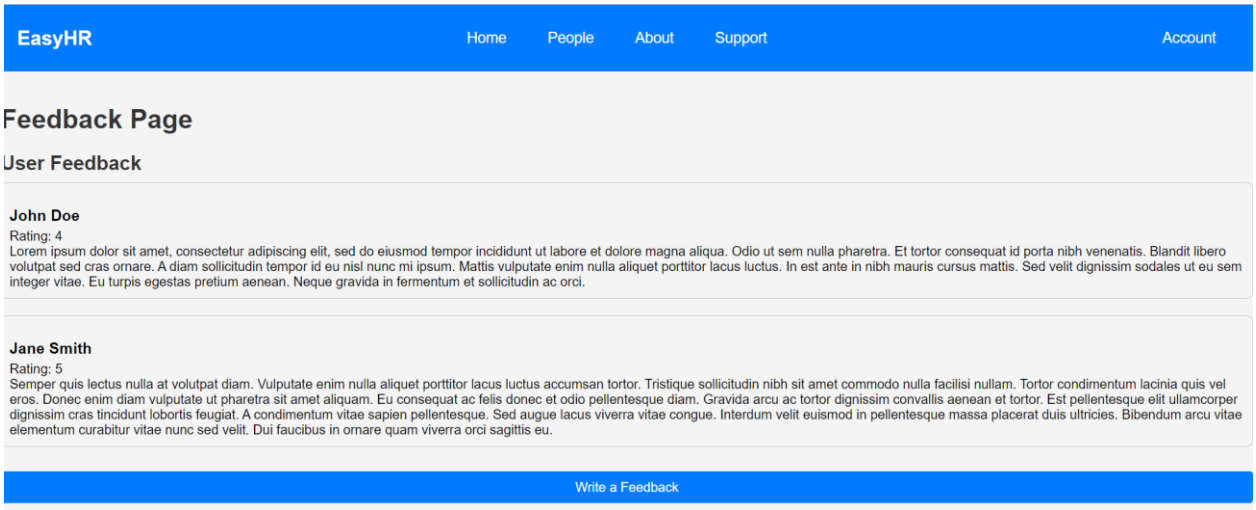

Рисунок 4.10 – Відгуки працівника після додавання нового відгука

Також було блок відгуків було протестовано за тест-кейсами. Перевірено Сценарій 3: Відмова від відгуку без вводу імені отримувача. За описом тест кейсу було проведено дії та перевірено результат (див. рис. 4.11).

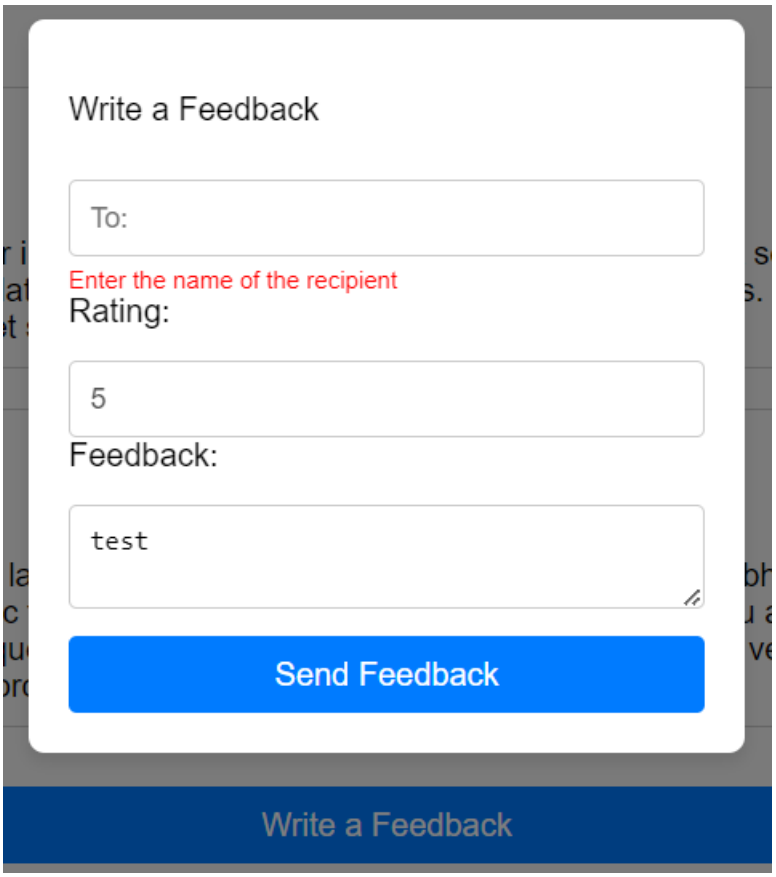

Рисунок 4.11 – Результат перевірки тест-кейсу з відсутністю отримувача

відгуку

Таким чином можна зробити висновок що сценарій 3 – пройдено. Також було перевірено сценарій 6: Відгук із максимальною кількістю символів. Проведено необхідні дії для перевірки відгуку (див. рис. 4.12).

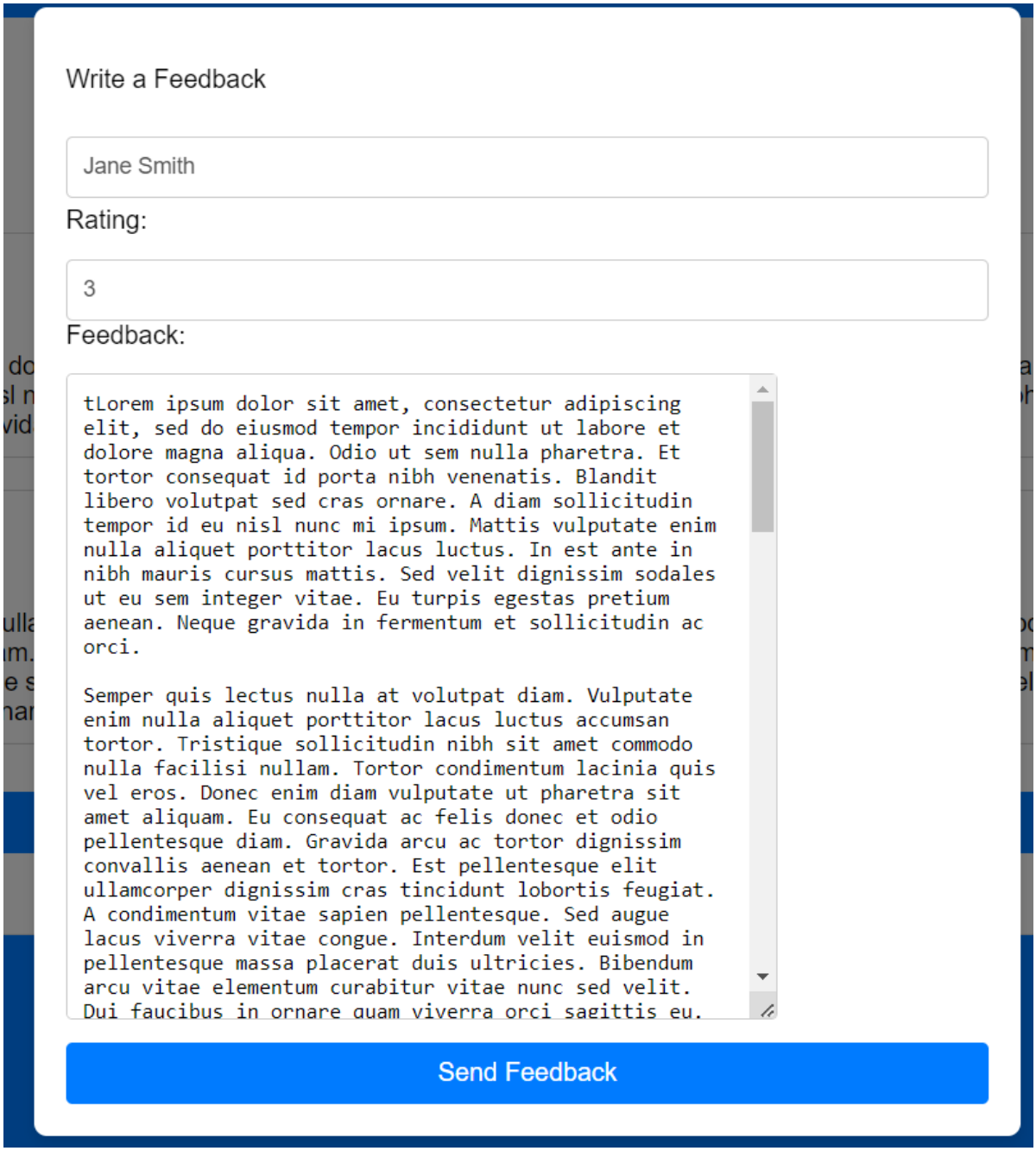

Рисунок 4.12 – Дії для перевірки тест кейсу з великою кількостю символів в тексті відгуку

Після відправки відгуку було перевірено чи був записаний відгук в базу даних (див. рис. 4.13)

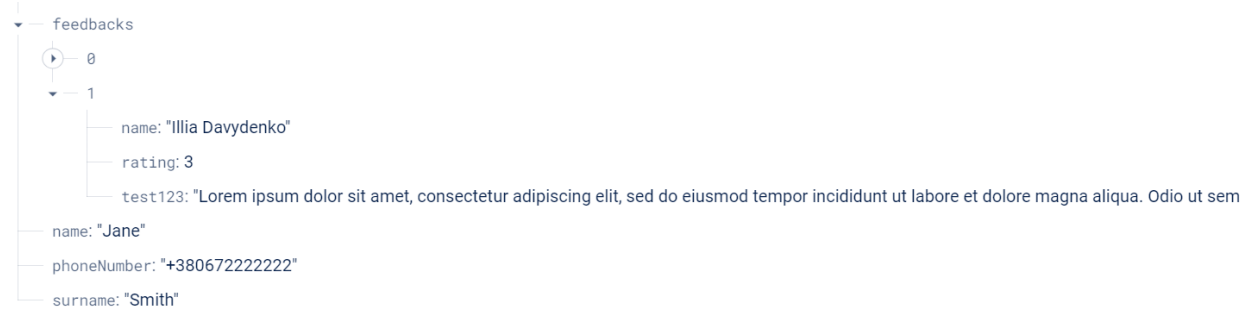

Рисунок 4.13 – Результат перевірки тест кейсу з великою кількостю символів в тексті відгуку

Запис в БД присутній, тому можна зробити висновок що тест кейс за сценарієм 6: Відгук із максимальною кількістю символів – пройдено.

Отже, було проведено тестування програмного засобу, розглянуто що таке тест кейси та створено їх для блоку відгуків.

4.3 Розробка інструкцій користувача

На початку роботи з програмним засобом потрібно пройти авторизацію, тільки після цього можна буде використовувати додаток. Після проведення авторизації користувач може перейти до свого аккаунту де він може побачити всю важливу для нього інформацію розподілену по розділам. В кожному розділі знаходиться відповідна йому інформація, наприклад, в розділі «Персональні дані» знаходяться особисті дані працівника, а в розділі «Відгуки» всі відгуки про нього і можливість залишати відгуки про колег.

Для того, щоб залишити відгук, потрібно відкрити вікно в розділі «Відгуки», обрати працівника, ввести текст та натиснути кнопку відправити. Після цього користувач зможе побачити цей відгук.

В різних користувачів різний доступ до програмного засобу. Їх рівень доступу можна змінювати безпосередньо в базі даних. Якщо працівник є менеджером, він може переглядати відгуки які були надіслані працівнику за якого він відповідає та має можливість змінювати дані його аккаунту.

В програмному засобі можна переглядати список працівників. Для цього потрібно перейти в розділ «Працівники», там ж можна ввести ім’я працівника щоб знайти його. Якщо цей розділ переглядає менеджер, він може зайти на профіль працівника при натисканні на його ім’я.

4.4 Висновки

У четвертому розділі було проаналізовано методи тестування та вирішено обрати для тестування метод «сірого ящика», адже алгоритми роботи додатку було описано частково і він є дуже зручним для створення тест-кейсів. Проведено тестування програмного засобу та розроблено інструкцію користувача для програмного застосунку.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методу та програмного засобу для організації роботи IT-команд» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методу та програмного засобу для організації роботи IT-команд» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [23].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту	Технічні та споживчі влас-тивості продукту	Технічні та споживчі властивості продукту на	Технічні та споживчі властивості продукту	Технічні та споживчі властивості продукту
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційни х витрат	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4

8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	40	39	41
Середньоарифметична сума балів $СБ_c$	40		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [23].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для організації роботи IT-команд» становить 40 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методу та програмного засобу для організації роботи IT-команд», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [23]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 30000{,}00 \cdot 60 / 22 = 81818{,}18 \text{грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	30000	1363,64	60	81818,18
Інженер-розробник програмного забезпечення	25000	1136,36	60	68181,82
Інженер-розробник програмного забезпечення	25000	1136,36	60	68181,82
Тестувальник	20000	909,09	30	27272,73
Консультант	20000	909,09	20	18181,82
Всього				263636,4

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М \cdot K_i \cdot K_c}{T_р \cdot t_{зм}} , \quad (5.3)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [23];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700,00 \cdot 1,10 \cdot 1,65 / (22 \cdot 8) = 69,09$ грн.
$З_{р1} = 69,09 \cdot 8,00 = 552,75$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн

Підготовка робочого місця дослідника	8	2	1,1	69,09	552,75
Інсталяція програмного забезпечення	3	5	1,7	106,78	320,34
Всього					873,09

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (263636,36 + 873,09) \cdot 11 / 100\% = 29096,04$ грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн = (263636,36 + 873,09 + 29096,04) \cdot 22 / 100\% = 64593,21$ грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1 = 3 \cdot 210{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 693{,}0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір	210	3	0	0	693
Канцелярське приладдя (набір офісного працівника)	175	2	0	0	385
Flesh-пам'ять	130	1	0	0	143
Всього					1221

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методу та програмного засобу для організації роботи IT-команд» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *18000· 2 · 1,1 = 39600 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
HP 290 G4 MT (123P7EA), монітор	2	18000	39600
HP 205 G8 Starry (6D452EA), монітор	2	25000	55000
Всього			94600

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 11600{,}00 \cdot 4 \cdot 1{,}1 = 51968$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	4	11600	51968
Прикладний пакет Microsoft Office 2019	4	5500	24640
Всього			76608

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (55000{,}00 \cdot 3) / (3 \cdot 12) =$ 4583,33*грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційн відрахування, грн
HP 290 G4 MT (123P7EA)-2шт	39 600	3	3	3300,00
HP 205 G8 Starry White (6D452EA)-2шт	55 000	3	3	4583,33
Робоче місце дослідника	10000	5	3	500,00
Оргтехніка	6000	4	3	375,00
ОС Windows 11 (4 пак)	51968	2	3	6496,00
Прикладний пакет Microsoft Office 2019 (4 пак)	24640	2	3	3080,00
Всього				18334,33

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7{,}50$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0{,}2 \cdot 480{,}0 \cdot 7{,}50 \cdot 0{,}95 / 0{,}97 = 2115{,}46$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
HP 290 G4 MT (123P7EA)-2шт	0,2	480	2115,46
HP 205 G8 Starry White (6D452EA)-2шт	0,2	480	2115,46
Робоче місце дослідника	0,15	480	581,75
Оргтехніка	0,45	10	33,05
Всього			4825,90

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (5.11)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

$B_{св}$ = (263636,36+ 873,09) · 20 / 100% = 52901,89грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (263636,36+ 873,09) · 30 / 100% = 92578,31грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де *$Н_{ів}$* – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (263636,36+ 873,09) · 50 / 100% = 132254,73грн.

5.2.12 *Накладні* (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = *(263636,36+ 873,09) · 120 / 100% = 317 411,35грн.*

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

$В_{заг}$=263636,36+873,09+29096,04+64593,21+1221+0,00+94600+76608+ 18334,33+ 4825,90+52901,89+92578,31+132254,73+317 411,35= 1148934,22грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 1148934,22/ 0,9 = 1276593,58грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 5000 компаній користувачів/рік;

2-й рік – 6000 компаній користувачів/рік;

3-й рік – 3500 компаній користувачів/рік.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 50000 компаній користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 88800,00 грн /за рік користування;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [24]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 30\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (50000,00 \cdot 1500,00 + 90300 \cdot 5000) \cdot 0,83 \cdot 0,3 \cdot (1\text{-}0,18/100\%) = 107889327$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (50000{,}00 \cdot 1500{,}00 + 90300 \cdot (5000 + 6000)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1 - 0{,}18/100\%) =$ 218913899,4грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (50000{,}00 \cdot 1500{,}00 + 90300 \cdot (5000 + 6000 + 3500)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1 - 0{,}18/100\%) =$ 283678233,3 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП =$

$107889327/(1+0{,}25)^1 + 218913899{,}4/(1+0{,}25)^2 + 283678233{,}3/(1+0{,}25)^3 = 37165961$

2,67 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.19}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1276593,58грн.

$PV = k_{інв} \cdot ЗВ$= 3· 1276593,58= 3829780,739грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 371659612,67 грн;

PV – теперішня вартість початкових інвестицій, 3829780,739грн.

$E_{абс} = ПП - PV$ = 371659612,67 - 3829780,739= 367829831,93грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 367829831,93грн;

PV – теперішня вартість початкових інвестицій, 3829780,739грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+367829831,93/3829780,739)^{1/3} = 3,6.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{5.22}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 3,6 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методу та програмного засобу для організації роботи ІТ-команд» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.23}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 3,6 = 0,28 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для організації роботи IT-команд» становить 40 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,28 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методу та програмного засобу для організації роботи IT-команд».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. І.С. Давиденко, Н.П. Бабюк «Аналіз програмних засобів для організації роботи ІТ-команд» - ВНТУ, 2023. Молодь в науці: дослідження, проблеми, перспективи.

2. Емоційне вигорання [Електронний ресурс] – Режим доступу до ресурсу: https://healthcenter.od.ua/psyhichne-zdorovya/emoczijne-vygorannya/

3. What is Workday [Електронний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/advisor/business/software/what-is-workday/

4. SAP [Електронний ресурс] – Режим доступу до ресурсу: https://www.sap.com/products/hcm.html

5. BambooHR [Електронний ресурс] – Режим доступу до ресурсу: https://www.bamboohr.com/

6. Структура веб-сайту [Електронний ресурс] – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/

7. Що таке SEO [Електронний ресурс] – Режим доступу до ресурсу: https://ag.marketing/blog/shcho-take-seo/

8. Інструменти веб-аналітики [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/instrumenty-veb-analitiki

9. SPA [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

10. UML для бізнес моделювання [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html

11. Бхаргава А., Грокаємо алгоритми / А. Бхаргава, – Київ: ArtHuss, 2023. 256 с.

12. Що таке GDPR [Електронний ресурс] – Режим доступу до ресурсу: https://legalaid.ua/ua/shho-take-gdpr/

13. Роббінс Д. HTML5: кишеньковий довідник 5-е видання. / Д. Роббінс – Київ: Діалектика, 2015. 192 с.

14. Мейер Е., Уейл Е. CSS повний довідник. / Е. Мейер, Е. Уейл – Київ: Діалектика, 2017. 1088 с.

15. Уроки SASS / SCSS. [Електронний ресурс]. – Режим доступу: https://itproger.com/course/sass

16. Webpack: керівництво для початківців. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/514838/

17. Давиденко І. С. Бабюк Н. П. Аналіз мови програмування JavaScript. / LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця: ВНТУ, 2022. 2 с.

18. Знайомство з React.js. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.legacy.reactjs.org/tutorial/tutorial.html

19. Що таке Node.js. [Електронний ресурс]. – Режим доступу: https://uk.theastrologypage.com/node-js

20. Що таке Firebase [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/44058/

21. Анатомія веб-сторінки [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/web-page-anatomy/

22. Ляхов О.Л. Методи тестування і оцінки якості програмного забезпечення. / О. Л. Ляхов, О.О. Бородіна – Полтава: ПолтНТУ, 2015. – 372 с.

23. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

24. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

Додаток А. Технічне завдання

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

ЗАТВЕРДЖУЮ
д.т.н., проф. О.Н. Романюк
"20" вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційної роботу
«Розробка методу та програмного засобу для організації роботи IT-команд»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Бабюк Н.П.
" 20 " вересня 2023 р.

Виконав:
студент гр. ЗПІ-22м Давиденко І.С.
" 20 " вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу і програмного засобу оптимізації робочих завдань ІТ-команди.».

Галузь застосування – інформаційні системи управління проектами.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від «18» вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення якості оптимізації завдань ІТ-команди за рахунок розробки програмного засобу з використанням методу оптимізації прийняття рішень.

Призначення роботи – розробка методу, алгоритмів та програмного засобу оптимізації робочих завдань ІТ-команди.

4. Вихідні дані для проведення НДР

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

1. Структура веб-сайту [Електронний ресурс] – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/

2. Інструменти веб-аналітики [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/instrumenty-veb-analitiki

3. Що таке GDPR [Електронний ресурс] – Режим доступу до ресурсу: https://legalaid.ua/ua/shho-take-gdpr/

4. Структура веб-сайту [Електронний ресурс] – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/

5. Технічні вимоги

МКР повинна задовольняти такі вимоги:

– запропонувати нові методи організації роботи ІТ-команд;

– розробити програмні компоненти та систему візуалізації на основі запропонованих методів;

– розробити основний алгоритм оптимізації з використанням методу прийняття рішень, розрахуванням відповідних критеріїв в залежності від заданих умов;

– вихідні дані – методи організації роботи ІТ-команд;

– результат роботи, метод організації роботи ІТ-команд та програмний продукт у вигляді системи для управління працівниками з додатковим функціоналом, що розроблено на основі методу організації роботи.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні. Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки.

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку засобів оптимізації робочих завдань ІТ-команди	20.09.2023 – 28.09.2023
2	Розробка структури і алгоритмів роботи програми	29.09.2023 – 14.10.2023
3	Розробка методу і програмного засобу для оптимізації робочих завдань ІТ-команди	15.10.2023 – 05.11.2023
4	Тестування програми	06.11.2023 – 21.11.2023
5	Економічна частина	22.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. Кафедрою згідно з графіком.

Додаток Б. Протокол перевірки роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методу та програмного засобу для організації роботи IT-команд

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Unicheck	
Оригінальність	92.8%
Схожість	7.2%

Науковий керівник: к.т.н., доцент кафедри ПЗ Бабюк Н. П.

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності 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 приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Давиденко І.С.

Керівник роботи _____ Бабюк Н. П.

Додаток В. Лістинг коду програмного засобу

Server.js

```
const express = require('express');
const admin = require('firebase-admin');

const app = express();
const port = process.env.PORT || 3001;

const serviceAccount = require('./easyhr-53d27-firebase-adminsdk-icl0r-
e16277d7d4.json');
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://easyhr-53d27-default-rtdb.europe-
west1.firebasedatabase.app/',
});

app.get('/getUser/:userId', async (req, res) => {
  const userId = req.params.userId;

  try {
    const snapshot = await
admin.database().ref(`users/${userId}`).once('value');
    const user = snapshot.val();
    res.json(user);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

app.get('/getAllUsers', async (req, res) => {
    try {
      const snapshot = await admin.database().ref('users').once('value');
      const users = snapshot.val();

      res.json(users);
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Internal Server Error' });
    }
  });

  app.post('/sendFeedback/:userEmail', async (req, res) => {
    const userEmail = req.params.userEmail;
    const feedbackData = req.body;

    try {
      const snapshot = await
admin.database().ref('users').orderByChild('email').equalTo(userEmail).once('
value');

      if (snapshot.exists()) {
        const userId = Object.keys(snapshot.val())[0];
        const userRef = admin.database().ref(`users/${userId}`);

        const user = snapshot.val()[userId];
        user.feedbacks.push(feedbackData);
```

```
        await userRef.set(user);

        res.status(200).json({ success: true });
      } else {
        res.status(404).json({ error: 'User not found' });
      }
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Internal Server Error' });
    }
  });

  app.get('/getUserFeedbacks/:userEmail', async (req, res) => {
    const userEmail = req.params.userEmail;

    try {
      const snapshot = await
admin.database().ref('users').orderByChild('email').equalTo(userEmail).once('
value');

      if (snapshot.exists()) {
        const userId = Object.keys(snapshot.val())[0]; // Отримуємо
ідентифікатор користувача
        const user = snapshot.val()[userId];
        const feedbacks = user.feedbacks || [];

        res.status(200).json(feedbacks);
      } else {
        res.status(404).json({ error: 'User not found' });
      }
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Internal Server Error' });
    }
  });

  app.post('/authorize', async (req, res) => {
    const { email, password } = req.body;

    try {
      const snapshot = await
admin.database().ref('users').orderByChild('email').equalTo(email).once('valu
e');
      const user = snapshot.val();

      if (user) {
        const userData = Object.values(user)[0];

        if (userData.password === password) {
          res.json({ success: true, message: 'Authorization successful',
user: userData });
        } else {
          res.status(401).json({ success: false, message: 'Invalid password'
});
        }
      } else {
        res.status(404).json({ success: false, message: 'User not found' });
      }
    } catch (error) {
      console.error(error);
      res.status(500).json({ success: false, message: 'Internal Server Error'
});
```

```
    }
  });

  app.post('/register', async (req, res) => {
    const { name, surname, email, password } = req.body;

    try {
      const snapshot = await
admin.database().ref('users').orderByChild('email').equalTo(email).once('valu
e');
      const existingUser = snapshot.val();

      if (existingUser) {
        res.status(409).json({ success: false, message: 'User with this email
already exists' });
      } else {
        const newUser = {
          name,
          surname,
          email,
          password,
          feedbacks: [],
          dateJoined: new Date().toISOString(),
        };

        await admin.database().ref('users').push(newUser);
        res.json({ success: true, message: 'Registration successful', user:
newUser });
      }
    } catch (error) {
      console.error(error);
      res.status(500).json({ success: false, message: 'Internal Server Error'
});
    }
  });
app.post('/editCustomer', async (req, res) => {
  const formData = req.body;

  try {
    const userId = formData.userId;

    // Перевірте, чи користувач існує перед оновленням
    const userSnapshot = await
admin.database().ref(`users/${userId}`).once('value');
    if (!userSnapshot.exists()) {
      return res.status(404).json({ error: 'User not found' });
    }

    // Оновіть дані користувача з тіла запиту
    await admin.database().ref(`users/${userId}`).update(formData);

    res.json({ message: 'User data updated successfully' });
  } catch (error) {
    console.error('Error updating user data:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});

// FeedbackBlock.js
import React, { useState, useEffect } from 'react';
```

```
import { createPortal } from 'react-dom';
import './FeedbackBlock.css';
import Feedback from '../Feedback/Feedback';

const FeedbackBlock = () => {
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [selectedUser, setSelectedUser] = useState('');
  const [rating, setRating] = useState(1);
  const [feedbackText, setFeedbackText] = useState('');
  const [users, setUsers] = useState([]);
  const userEmail = localStorage.getItem('currUserEmail');

  useEffect(() => {
    // Здійснюємо запит до бази даних для отримання списку імен
    const fetchData = async () => {
      try {
        const response = await fetch('http://localhost:3001/getAllUsers'); //
Змініть URL на свій
        const data = await response.json();

        if (Array.isArray(data)) {
          // Якщо дані є масивом, оновіть стан користувачів
          setUsers(data);
        } else {
          console.error('Invalid data format received from the server.');
        }
      } catch (error) {
        console.error('Error fetching user data:', error);
      }

    };

    fetchData();
  }, []); // Запускається лише раз при монтажі компонента

  const getUserFeedbacks = async (email) => {
    try {
      const response = await fetch(`http://localhost:3001/getUser/${email}`);
      const userData = await response.json();

      if (userData && userData.feedbacks) {
        return userData.feedbacks;
      } else {
        console.error('User data or feedbacks not found');
        return [];
      }
    } catch (error) {
      console.error('Error fetching user feedbacks:', error);
      return [];
    }
  };

  const handleOpenModal = () => {
    setIsModalOpen(true);
  };

  const handleCloseModal = () => {
    setIsModalOpen(false);
  };

  const handleSendFeedback = async () => {
    try {
      // Перевіряємо, чи вибрано користувача та введено текст відгуку
```

```
      if (!selectedUser || !feedbackText) {
        console.error('Please select a user and enter feedback text.');
        return;
      }

      // Формуємо об'єкт з даними відгуку
      const feedbackData = {
        to: selectedUser,
        rating: rating,
        text: feedbackText,
      };

      // Відправляємо дані на сервер
      const response = await fetch('http://localhost:3001/sendFeedback', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(feedbackData),
      });

      // Перевіряємо статус відповіді
      if (!response.ok) {
        console.error('Failed to send feedback.');
        return;
      }

      console.log('Feedback sent successfully!');

      // Після відправки можна очистити стан для наступного відгуку
      setSelectedUser('');
      setRating(1);
      setFeedbackText('');
      setIsModalOpen(false);
    } catch (error) {
      console.error('Error sending feedback:', error);
    }
  };

  const getUserOptions = () => {
    return users.map((user) => user.fullName);
  };
  const userFeedbacks = getUserFeedbacks(userEmail);

  return (
    <div>
      <h2>User Feedback</h2>

      {feedbacks.map((feedback) => (
        <Feedback name={feedback.name} rating={feedback.rating}
text={feedback.text} />
      ))}

      <button onClick={handleOpenModal}>Write a Feedback</button>
      {/* Modal */}
      {isModalOpen && (
        <div className="modal-overlay" onClick={handleCloseModal}>
          <div className="modal" onClick={(e) => e.stopPropagation()}>
            <span className="close" onClick={handleCloseModal}>
              &times;
            </span>
            <p>Write a Feedback</p>
            {/* Dropdown for selecting name */}
```

```
            <input
              type="text"
              placeholder="To:"
              value={selectedUser}
              onChange={(e) => setSelectedUser(e.target.value)}
              list="userOptions"
              required  // Додаємо атрибут required
            />
            <datalist id="userOptions">
              {getUserOptions().map((option) => (
                <option key={option} value={option} />
              ))}
            </datalist>
            <span className="form-error">
              {selectedUser.trim() === '' && 'Enter the name of the
recipient'}
            </span>
            {/* Rating input */}
            <label>Rating:</label>
            <input
              required
              type="number"
              min="1"
              max="5"
              value={rating}
              onChange={(e) => setRating(Number(e.target.value))}
            />
            {/* Feedback text input */}
            <label>Feedback:</label>
            <textarea
              value={feedbackText}
              onChange={(e) => setFeedbackText(e.target.value)}
            />
            {/* Button to send feedback */}
            <button onClick={handleSendFeedback}>Send Feedback</button>
          </div>
        </div>
      )}
    </div>
  );
};

export default FeedbackBlock;
```

PersonalInfo.jsx

```
import React from 'react';
import PersonalInfoField from '../PersonalInfoField/PersonalInfoField';
import './PersonalInfo.css';

const PersonalInfo = () => {
  const [formData, setFormData] = useState({});
  const [isEditing, setIsEditing] = useState(false);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('http://localhost:3001/getAllUsers'); //
Змініть URL на свій
        const data = await response.json();

        if (Array.isArray(data) && data.length > 0) {
```

```
          // Якщо дані є масивом, встановіть початкові дані для користувача
          setFormData(data[0]); // Ви можете вибрати першого користувача, або
використовувати інші логіку вибору
        } else {
          console.error('Invalid data format received from the server.');
        }
      } catch (error) {
        console.error('Error fetching user data:', error);
      }
    };

    fetchData();
  }, []); // Перший запуск при завантаженні компонента

  const handleEditSave = async () => {
    if (isEditing) {
      try {
        // Ваша логіка відправки даних на сервер
        await fetch('http://localhost:3001/editCustomer', {
          method: 'POST', // або PUT, залежно від вашої реалізації
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify(formData),
        });

        // Після відправки можна переключити режим редагування
        setIsEditing(false);
      } catch (error) {
        console.error('Error updating user data:', error);
      }
    } else {
      // Якщо ми в режимі редагування, змінюємо стан на збереження та
дозволяємо редагування
      setIsEditing(true);
    }
  };

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });
  };
  return (
    <form className="personal-info">
      <h2>Personal Information</h2>
      <PersonalInfoField label="Name" type="text" value={formData.name}
disabled={!isEditing} onChange={handleInputChange} />
      <PersonalInfoField label="Surname" type="text" value={formData.surname}
disabled={!isEditing} onChange={handleInputChange} />
      <PersonalInfoField label="Email" type="email" value={formData.email}
disabled={!isEditing} onChange={handleInputChange} />
      <PersonalInfoField label="Date of Birth" type="date"
value={formData.birthDate} disabled={!isEditing} onChange={handleInputChange}
/>
      <PersonalInfoField label="Phone number" type="string"
value={formData.mobilePhone} disabled={!isEditing}
onChange={handleInputChange} />
      <h2>Address</h2>
```

```
      <PersonalInfoField label="Country" type="text"
value={formData.address.country} disabled={!isEditing}
onChange={handleInputChange} />
      <PersonalInfoField label="City" type="text"
value={formData.address.city} disabled={!isEditing}
onChange={handleInputChange} />
      <PersonalInfoField label="Street" type="text"
value={formData.address.street} disabled={!isEditing}
onChange={handleInputChange} />
      <PersonalInfoField label="Postal Code" type="text"
value={formData.address.postCode} disabled={!isEditing}
onChange={handleInputChange} />

      <button type="button" onClick={handleEditSave}>
        {isEditing ? 'Save Changes' : 'Edit Data'}
      </button>
    </form>
  );
};

export default PersonalInfo;
```

Login.jsx

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';

const AuthorizationForm = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

const handleLogin = async (e) => {
    e.preventDefault();
    let user = null;

    try {
      const response = await fetch(`http://localhost:3001/getUser/${email}`);

      if (!response.ok) {
        setError('Invalid email or password. Please try again.');
      }

      user = await response.json();

      if (password !== user.password) {
          setError('Invalid email or password. Please try again.');
      }

    } catch (error) {
      setError(error.message || 'Error during login.');
    }

    return user;
  };

  return (
    <div className='authorization-form'>
      <h2>Authorization</h2>
      {error && <p style={{ color: 'red' }}>{error}</p>}
      <form onSubmit={handleLogin}>
        <label htmlFor="email">Email:</label>
        <input
          type="email"
```

```
            id="email"
            name="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
          />

          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            name="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
          />

          <button type="submit">Login</button>
        </form>

        <div>
          <Link to="/forgot-password">Forgot Password?</Link>
        </div>

        <div>
          <Link to="/create-account">Create an Account</Link>
        </div>
      </div>
    );
};

export default AuthorizationForm;
```

EmployeeList.jsx

```
import React, { useState, useEffect } from 'react';
import User from '../User/User';
import './EmployeeList.css';

const EmployeeList = () => {
  const [employees, setEmployees] = useState([]);
  const [searchTerm, setSearchTerm] = useState('');
  const employees1 = [{name: 'John Doe', email: 'john.doe@gmail.com'},
        {name: 'Jane Smith', email: 'jane.smith@gmail.com'},
        {name: 'Illia Davydenko', email: 'illia.davydenko@gmail.com'},
        {name: 'Dmytro Korol', email: 'dmytro.korol@gmail.com'},
        {name: 'Artur Dushenko', email: 'artur.dushenko@gmail.com'}
    ]
  useEffect(() => {
    const fetchEmployees = async () => {
      try {
        const response = await fetch('http://localhost:3001/getAllUsers');
        const data = await response.json();

        setEmployees(data);
      } catch (error) {
        console.error('Error fetching employees', error);
      }
    };

    fetchEmployees();
  }, []);

  const filteredEmployees = employees1.filter((employee) =>
    employee.name.toLowerCase().includes(searchTerm.toLowerCase())
  );
```

```
  return (
    <div>
      <h2>Employee List</h2>
      <input
        type="text"
        placeholder="Search by name"
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />
      <ul>
        {filteredEmployees.map((employee) => (
          <User key={employee.email} name={employee.name}
email={employee.email} />
        ))}
      </ul>
    </div>
  );
};

export default EmployeeList;

import React from 'react';
import PropTypes from 'prop-types';
import './Feedback.css';

const Feedback = ({ name, rating, text }) => (
  <div className="feedback-block">
    <h3>{name}</h3>
    <p>Rating: {rating}</p>
    <p>{text}</p>
  </div>
);

Feedback.propTypes = {
  name: PropTypes.string.isRequired,
  rating: PropTypes.number.isRequired,
  text: PropTypes.string.isRequired,
};

export default Feedback;
import React from 'react';
import NavigationItem from '../../NavigationItem/NavigationItem';
import PeopleEvents from '../../PeopleEvents/PeopleEvents';
import PersonalInfo from '../PersonalInfo/PersonalInfo';
import './AccountPage.css'; // Import your AccountPage styles
import DefaultImage from '../../../avatar-placeholder.png'

const AccountPage = () => {
  return (
    <div className="account-page">
      <nav className="account-navigation">
        <NavigationItem to="/account"
label="Profile">Profile</NavigationItem>
        <NavigationItem to="/account/job" label="Job">Job</NavigationItem>
        <NavigationItem to="/account/feedback"
label="Feedback">Feedback</NavigationItem>
        <NavigationItem to="/account/emergency"
label="Emergency">Emergency</NavigationItem>
        <NavigationItem to="/account/benefits"
label="Benefits">Benefits</NavigationItem>
        <NavigationItem to="/account/health"
label="Health">Health</NavigationItem>
```

```
      </nav>
      <div className='account-info'>
        <div className="account-left-block">
                <div className="user-photo">
                  <img src={DefaultImage} alt="" />
                </div>
                <PeopleEvents />
        </div>

              <div className="account-right-block">
                <PersonalInfo />
              </div>
        </div>

    </div>
  );
};

export default AccountPage;
import React from 'react';
import './Event.css';
import DefaultImage from '../../avatar-placeholder.png'

const Event = ({ name, caption, avatar }) => {
    return (
    <div className="event">
      <div className="avatar">

        <img src={DefaultImage} alt={name} />
      </div>
      <div className="event-info">
        <p className="event-name">{name}</p>
        <p className="event-caption">{caption}</p>
      </div>
    </div>
  );
};

export default Event;

import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Header from './components/Header/Header';
import Footer from './components/Footer/Footer';
import Home from './components/Home/Home';
import AccountPage from './components/Account/AccounPage/AccountPage';
import AuthorizationForm from
'./components/AuthorizationForm/AuthorizationForm';
import FeedbackPage from './components/Account/FeedbackPage/FeedbackPage';
import EmployeeList from './components/EmployeeList/EmployeeList';

const App = () => {
  return (
    <Router>
      <div>
        <Header />
        <div className="content">
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/account" element={<AccountPage />} />
            <Route path="/authorize" element={<AuthorizationForm />} />
            <Route path="/account/feedback" element={<FeedbackPage />} />
            <Route path="/employees" element={<EmployeeList />} />
```

```
          </Routes>
          <div id="modal-root"></div>
        </div>
        <Footer />
      </div>
    </Router>
  );
};

export default App;


import React from 'react';
import NavigationItem from '../NavigationItem/NavigationItem';
import './Header.css';

const Header = () => {
  return (
    <header className="header">
      <div className="logo">EasyHR</div>
      <nav className="navigation">
        <NavigationItem to="/">Home</NavigationItem>
        <NavigationItem to="/employees">People</NavigationItem>
        <NavigationItem to="/about">About</NavigationItem>
        <NavigationItem to="/support">Support</NavigationItem>
      </nav>
      <NavigationItem to="/account">Account</NavigationItem>
    </header>
  );
};

export default Header;

index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import './styles.css'; // Import your global styles

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);
```

Додаток Г. Ілюстративна частина

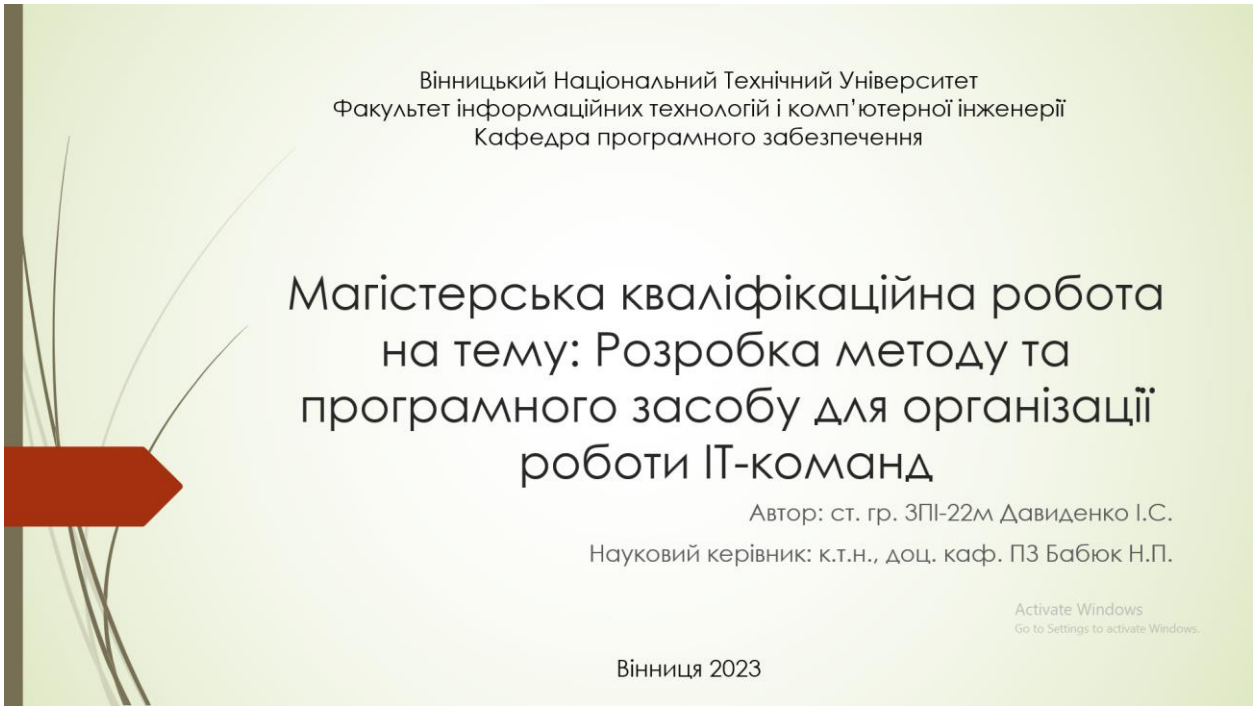

Рисунок Г.1 – Титульний слайд

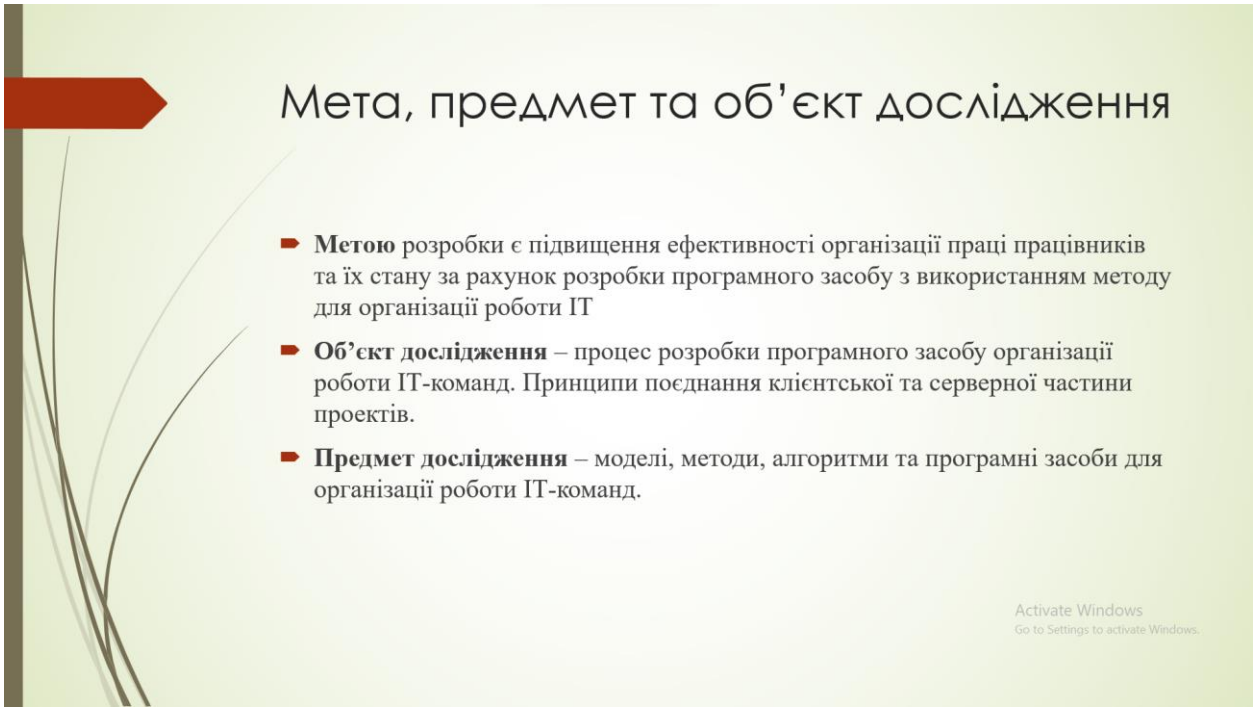

Рисунок Г.2 – Мета, предмет та об'єкт дослідження

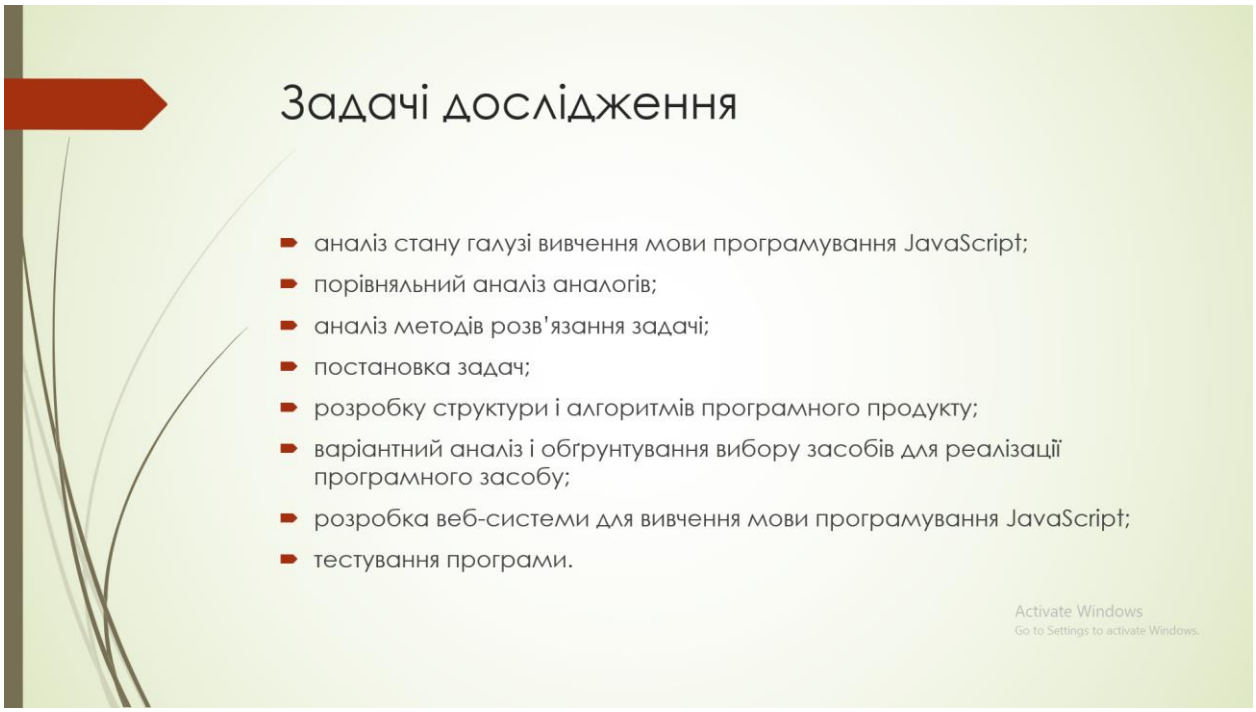

Рисунок Г.3 – Задачі дослідження

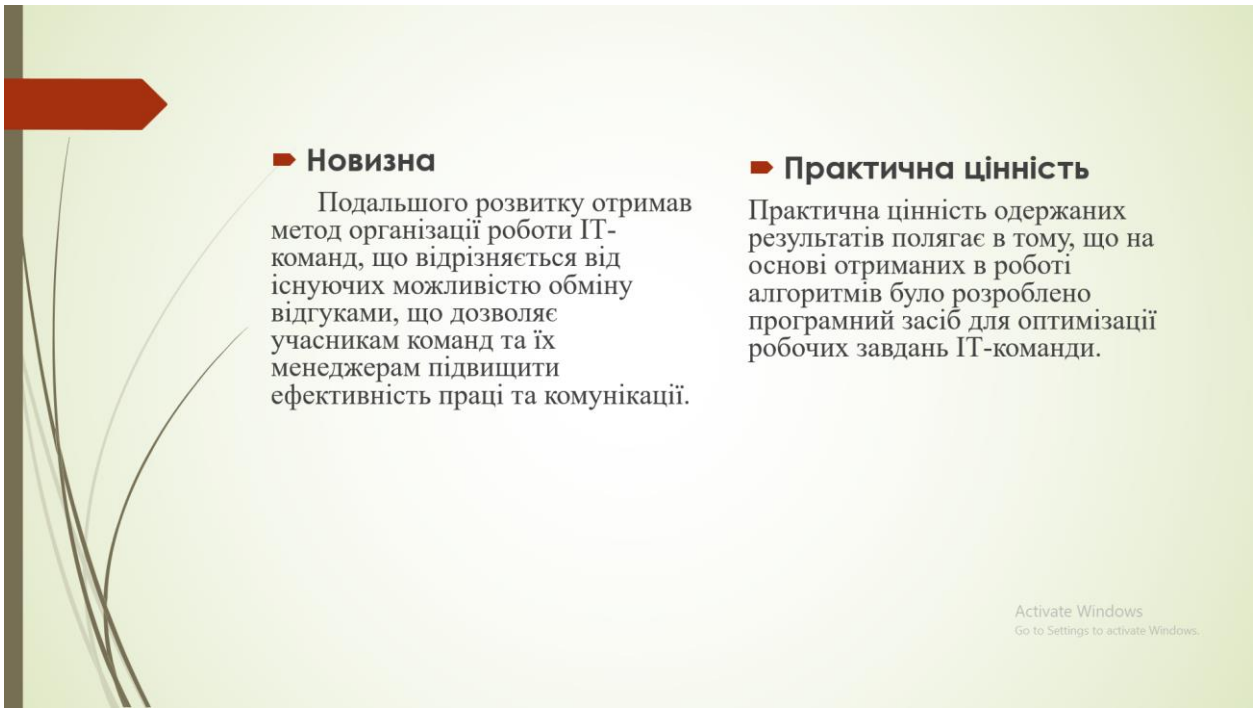

Рисунок Г.4 – Новизна і практична цінність

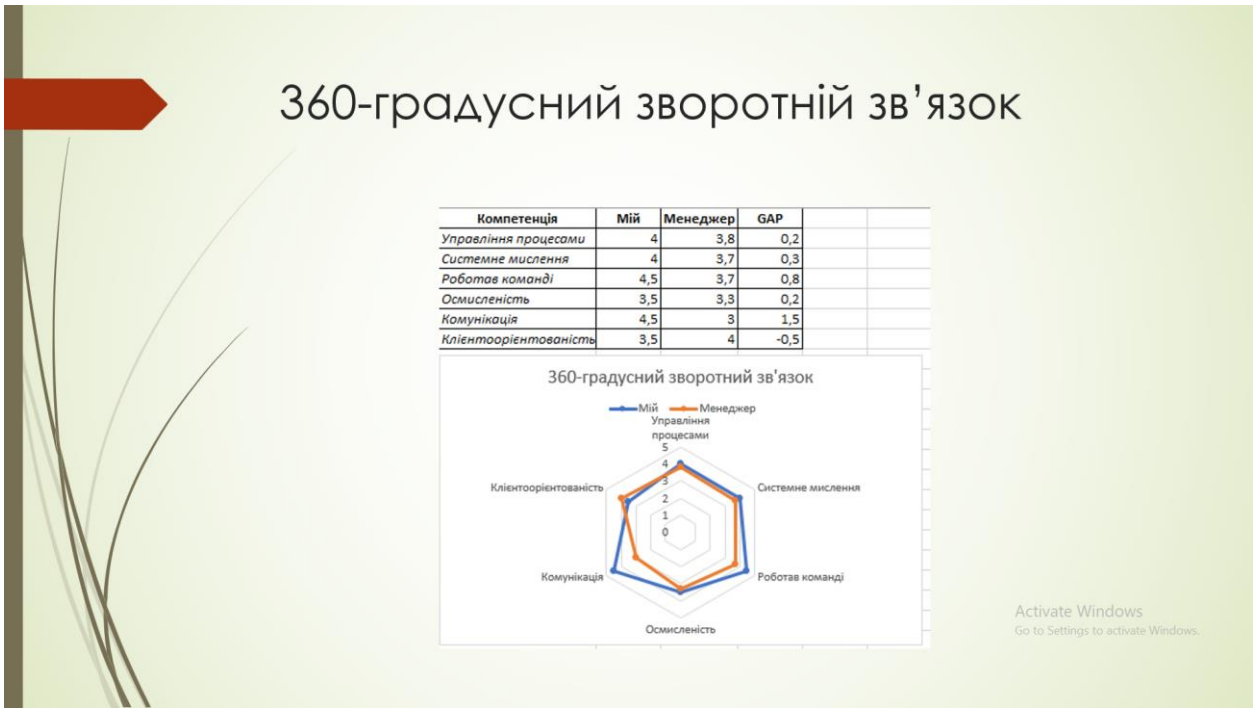

Рисунок Г.5 – 360-градусний зворотній зв’язок

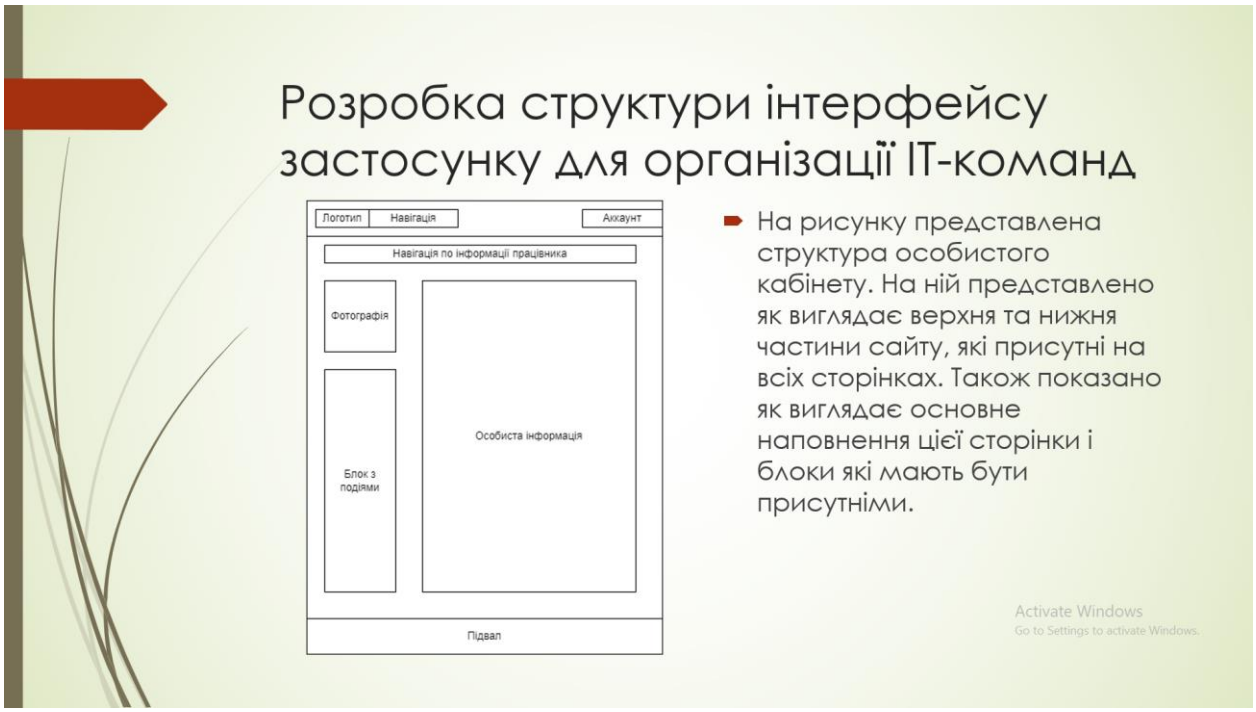

Рисунок Г.6 – Розробка структури інтерфейсу застосунку для організації IT-команд

Рисунок Г.7 – Блок схема блоку привітань працівників

Рисунок Г.8 – Використані технології

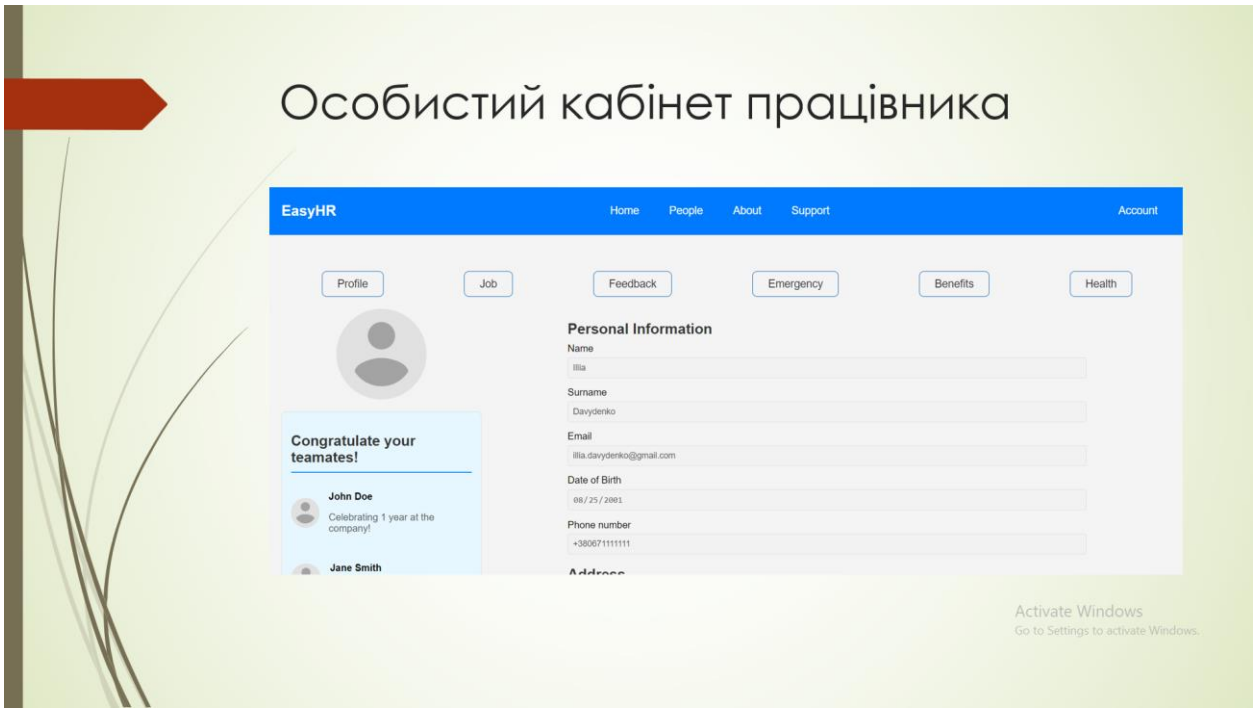

Рисунок Г.9 – Особостий кабінет працівника

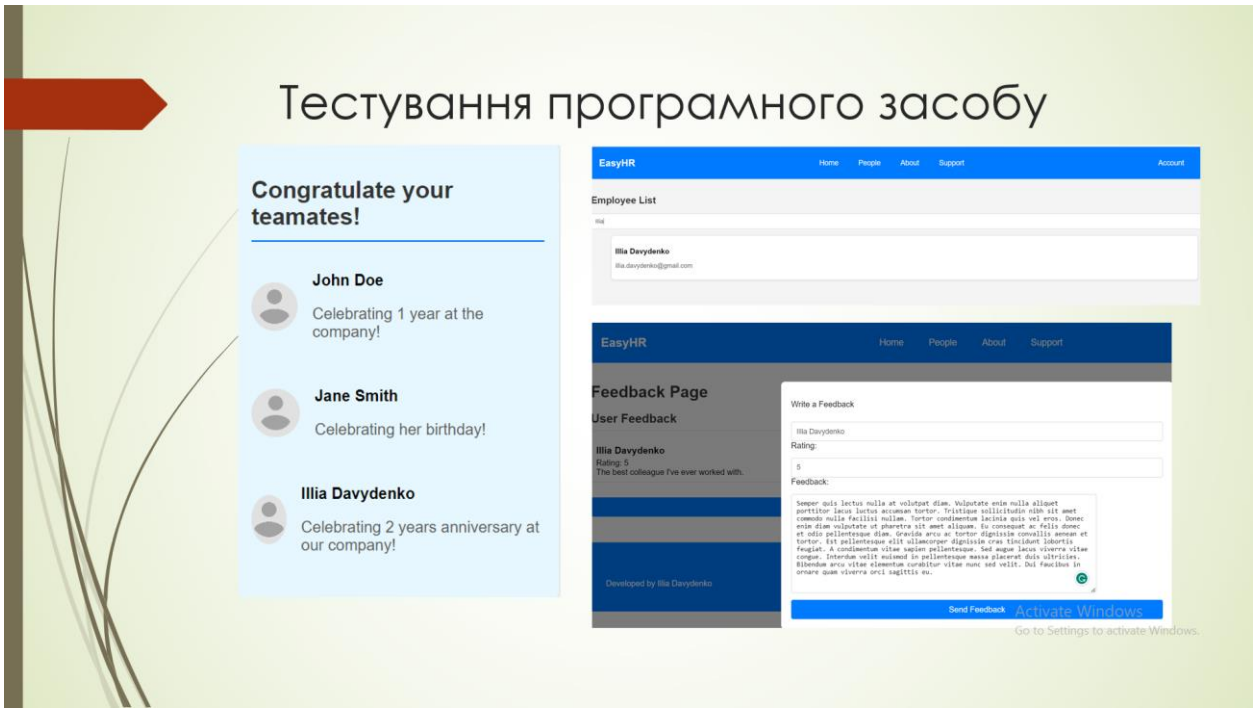

Рисунок Г.10 – Тестування програмного засобу

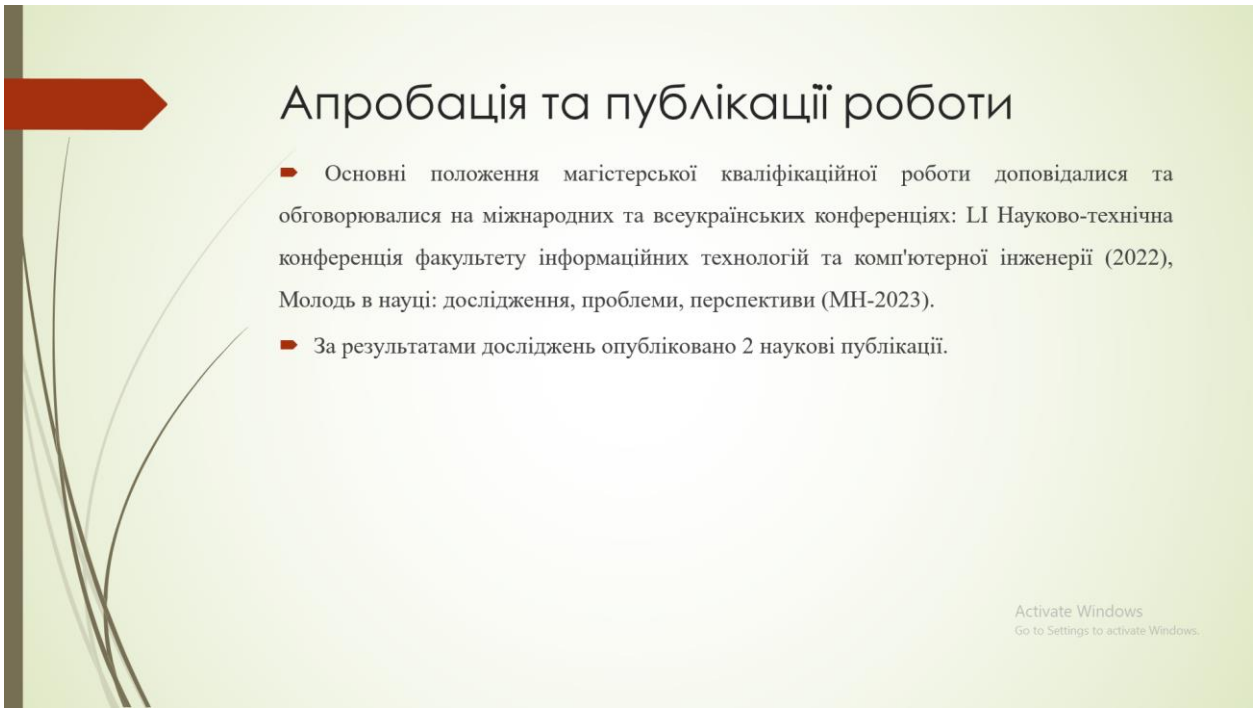

Рисунок Г.11 – Апробація та публікації роботи

Рисунок Г.12 – Фінальний слайд