Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ РОБОТИ СПІВРОБІТНИКІВ

Виконав: студент 4 курсу, групи 2КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Афросімова А. А.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О. В.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., доцент каф. АІІТ
Козачко О. М.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН проф.,
д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Афросімовій Анастасії Андріївній

1. Тема роботи: Розробка WEB-додатку для оцінювання продуктивності роботи співробітників.
керівник роботи: Сілагін Олексій Віталійович, к.т.н., доц.
затверджені наказом вищого навчального закладу «11» 03 2024 року № 80

2. Термін подання студентом роботи 27.05.2024 р.

3. Вихідні дані до роботи:
Мова програмування – об'єктно орієнтована з можливістю маніпулювання даними і управління базами даних, серевер яких підтримує HTTP протокол;
Відповідність до бази даних ODBC;
Функціональні вимоги: реєстрація користувачів, перегляд усіх оцінювань, проходження оцінювання, отримання результатів оцінювання, отримання зворотного зв'язку шляхом написанням відгуків;
Інтерфейс користувача має бути інтуїтивно зрозумілий.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; обґрунтування доцільності розробки WEB-додатку для оцінювання продуктивності роботи співробітників; аналіз програм-аналогів, їх переваги та недоліки; проектування WEB-додатку для оцінювання продуктивності роботи співробітників; програмна реалізація WEB-додатку для оцінювання продуктивності роботи співробітників; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму роботи WEB-додатку для оцінювання

продуктивності роботи співробітників; UML-діаграма прецедентів WEB-додатку для оцінювання продуктивності роботи співробітників; UML-діаграма взаємодій WEB-додатку для оцінювання продуктивності роботи співробітників; ER-діаграма бази даних; загальна UML-діаграма класів.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Сілагін О. В., к.т.н., доц. каф. КН		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Обґрунтування доцільності розробки WEB-додатку для оцінювання продуктивності роботи співробітників	06.05.24	10.05.24	
2	Аналіз програм-аналогів, їх переваги та недоліки	11.05.24	15.05.24	
3	Проектування WEB-додатку для оцінювання продуктивності роботи співробітників	16.05.24	19.05.24	
4	Програмна реалізація WEB-додатку для оцінювання продуктивності роботи співробітників	20.05.24	24.05.24	
5	Розробка інструкції користувача	27.05.24	29.05.24	
6	Висновки та аналіз отриманих результатів	30.05.24	03.06.24	
7	Оформлення матеріалів до захисту БДР	04.06.24	06.06.24	

Студент _____ **Афросімова А.А.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Сілагін О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 104 сторінок формату А4, на яких є 30 рисунків, 7 таблиць, список використаних джерел містить 33 найменування.

Метою роботи є розширення функціональних можливостей WEB-додатку для оцінювання продуктивності роботи співробітників.

У загальній частині роботи на основі аналізу існуючих технічних рішень, методів та технологій доведена доцільність розробки WEB-додатку для оцінювання продуктивності роботи співробітників.

Для WEB-додатку розроблено алгоритм роботи. Програмний продукт розроблений в інтегрованому середовищі IntelliJ IDEA за допомогою мови програмування Java та фреймворків Java Spring Framework, Vaadin.

Розроблено програмний продукт та проведено його тестування. Результати тестування розробки вказують на те, що основні функції WEB-додатку для оцінювання продуктивності роботи співробітників реалізовано.

Ключові слова: веб-додаток, продуктивність роботи, оцінювання, співробітники.

ABSTRACT

The bachelor's thesis consists of 104 pages of A4 format, which contain 30 figures, 7 tables, and a list of 33 references contains titles.

The aim of the work is to expand the functional capabilities of the WEB-application for evaluating the performance of employees.

In the general part of the work, based on the analysis of existing technical solutions, methods and technologies, the feasibility of developing a WEB-application for evaluating the performance of employees is proved.

An algorithm of work of its software module was developed for the WEB-application. The software product was developed in the integrated environment IntelliJ IDEA using programming language Java and frameworks Java Spring Framework, Vaadin.

The software product was developed and tested. The results of testing the development indicate that the main functions of the WEB-application for evaluating the performance of employees have been implemented.

Keywords: web-application, work productivity, assessment, employees.

ЗМІСТ

ВСТУП

Актуальність досліджень. У сучасному світі, де інформаційні технології невпинно розвиваються, автоматизація різноманітних бізнес-процесів стає дедалі важливішою складовою успішного функціонування компаній. Однією з таких критично важливих сфер є оцінювання продуктивності роботи співробітників. Традиційні методи оцінювання, що включають ручне заповнення форм та періодичні формальні зустрічі, виявляються малоефективними в умовах сучасного динамічного бізнес-середовища.

Серед ключових переваг впровадження автоматизованих систем оцінювання продуктивності є швидкий зворотний зв'язок. Працівники все частіше віддають перевагу частим та неформальним обговоренням їхньої продуктивності замість щорічних формальних оцінок. Автоматизовані системи дозволяють менеджерам і відділу кадрів підтримувати постійний зв'язок з працівниками, що сприяє своєчасній комунікації та підвищенню ефективності роботи працівників.

Ще однією суттєвою перевагою є оптимізація процесів оцінювання. Автоматизація значно зменшує навантаження на відповідальних осіб, що дозволяє проводити частіші оцінки та ефективніше моніторити прогрес і потреби у розвитку працівників. Крім того, автоматизація ручних завдань звільняє час для вдосконалення самих процесів оцінювання.

Легкий доступ до інформації також є вагомою перевагою автоматизованих систем. Менеджери та співробітники можуть швидко звертатися до минулих записів, що допомагає надавати чесні та прозорі огляди продуктивності, а також значущі рекомендації щодо розвитку.

Таким чином, актуальність розробки WEB-додатку для оцінювання продуктивності роботи співробітників є очевидною. Розробка цього додатку не тільки покращить ефективність роботи співробітників, але й сприятиме

більш об'єктивному та прозорому процесу оцінювання, забезпечить своєчасний зворотний зв'язок та підвищить загальну конкурентоспроможність компанії на ринку.

Метою дослідження є розширення функціональних можливостей WEB-додатку для оцінювання продуктивності роботи співробітників.

Об’єктом дослідження є процес розробки WEB-додатку для оцінювання продуктивності роботи співробітників.

Предметом дослідження є алгоритми та програмні засоби для реалізації розробки WEB-додатку для оцінювання продуктивності роботи співробітників.

Задачі дослідження:

1. Обґрунтувати доцільність розробки WEB-додатку для оцінювання продуктивності роботи співробітників;

2. Проаналізувати програми-аналоги, їх переваги та недоліки;

3. Спроектувати WEB-додаток для оцінювання продуктивності роботи співробітників;

4. Програмно реалізувати WEB-додаток для оцінювання продуктивності роботи співробітників;

5. Виконати тестування програми та провести аналіз отриманих результатів.

Практична цінність: розроблені алгоритми та методи можуть бути використані при створенні подібних WEB-додатків.

Апробація роботи. Результати досліджень було апробовано на LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024) [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-ДОДАТКУ ДЛЯ ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ РОБОТИ СПІВРОБТІНИКІВ

1.1 Аналіз сучасних методологій оцінювання продуктивності роботи співробітників

Автоматизоване оцінювання роботи працівників за допомогою WEB-додатків стало популярним інструментом у сучасних організаціях. Такий підхід дозволяє систематично і об'єктивно оцінювати продуктивність співробітників, використовуючи чіткі критерії оцінки та кількісні показники [2].

WEB-додатки для оцінювання продуктивності дозволяють зменшити час і ресурси, необхідні для проведення оцінювання, що підвищує загальну ефективність процесу. Співробітники можуть отримувати чітке уявлення про те, як оцінюється їхня робота, які показники використовуються та як вони впливають на загальну оцінку. Керівники можуть відстежувати результати оцінювання у режимі реального часу, що дозволяє оперативно виявляти проблемні ділянки, приймати своєчасні управлінські рішення та оптимізувати процеси для досягнення кращих результатів. На основі даних, зібраних системою, можна розробляти індивідуальні плани розвитку та навчання для співробітників, що сприяє їхньому професійному росту та підвищенню кваліфікації. Автоматизація процесу оцінювання зменшує обсяг паперової роботи та адміністративних завдань, дозволяючи сфокусуватися на основних робочих процесах.

Можна виділити такі переваги автоматизованого оцінювання продуктивності роботи співробітників:

- зручність опрацювання результатів;
- зменшення витраченого часу;
- прозорість процесу;

Оцінка знань і продуктивності співробітників є ключовим аспектом управління персоналом у будь-якій організації. У сучасному світі існує декілька методів тестування знань, які можуть бути використані для оцінювання результативності працівників.

Метод самооцінки передбачає, що працівники самостійно оцінюють свої знання, навички та досягнення [3]. Цей метод дозволяє співробітникам аналізувати свою роботу, виявляти сильні та слабкі сторони, а також встановлювати цілі для самовдосконалення. Основною перевагою методу самооцінки є розвиток самосвідомості і відповідальності за власну продуктивність. Однак, недоліком є можливість суб'єктивності, оскільки співробітники можуть недооцінювати так і переоцінювати свої здібності.

Метод оцінки керівником передбачає оцінювання знань і продуктивності співробітників безпосереднім керівником [4]. Цей метод забезпечує об'єктивну оцінку з боку досвідченого менеджера, який добре знає роботу працівника. Керівник може надавати зворотний зв'язок, визначати зони для розвитку та допомагати у встановленні цілей. Основна перевага цього методу полягає в його об'єктивності та досвіді оцінювача. Проте, можлива упередженість з боку керівника може стати недоліком цього підходу.

Метод 180° передбачає оцінювання знань і продуктивності співробітника з двох сторін: самим працівником і його безпосереднім керівником [5]. Цей метод поєднує самооцінку з об'єктивною оцінкою керівника, що забезпечує більш збалансоване і точне оцінювання. Працівник може порівняти свою самооцінку з оцінкою керівника, що сприяє кращому розумінню своїх сильних і слабких сторін, а також визначенню напрямків для розвитку.

Основною перевагою методу 180° є його об'єктивність і комплексність. Він дозволяє враховувати різні точки зору, що робить оцінку більш точною та корисною для працівника. Крім того, цей метод сприяє розвитку відкритого діалогу між працівником і керівником, що може покращити загальний робочий процес та підвищити продуктивність.

Метод 360° включає оцінювання знань і продуктивності співробітника з усіх можливих сторін: самого працівника, керівника, колег, підлеглих та клієнтів [6]. Цей метод забезпечує всебічну оцінку, яка враховує різні аспекти діяльності працівника. Основна перевага методу 360 градусів полягає в його комплексності і детальності. Він дозволяє отримати повну картину про працівника, його взаємодію з різними сторонами та його роль у команді.

Проте, метод 360° має і свої недоліки. Він може бути досить трудомістким і затратним за часом, оскільки потребує залучення великої кількості оцінювачів. Крім того, можливість суб'єктивності та упередженості з боку оцінювачів може вплинути на результати.

Результати порівняння методів для оцінювання продуктивності роботи співробітників наведено в таблиці 1.1.

Таблиця 1.1. – Порівняння методів для оцінювання продуктивності роботи співробітників

	Об’єктивність	Комплексність	Трудомісткість	Суб’єктивність
Метод самооцінка	-	-	+	+
Метод оцінка керівником	+	-	-	-
Метод 180° (працівник – керівник)	+	+	+	-
Метод 360° (працівник – керівник – колеги)	-	+	-	+

Отже, виконавши аналіз важливості застосування електронного оцінювання роботи співробітників та виділивши методи для його виконання, можна зробити висновок, що метод 180° є одним із найефективніших. Він поєднує самооцінку і оцінку керівником, що забезпечує збалансоване і об'єктивне оцінювання. Цей метод сприяє розвитку відкритого діалогу та співпраці між працівником і керівником, що може покращити загальну продуктивність команди. Тому метод 180° є оптимальним вибором для оцінювання знань і продуктивності співробітників.

1.2 Сучасні засоби оцінювання роботи співробітників

Існує велика кількість застосунків для оцінювання роботи співробітників в електронному вигляді. Варто виділити декілька відомих та провести аналіз ключових функцій, плюсів та недоліків їх використання. Виділимо наступні програми для електронного оцінювання роботи працівників:

1. BambooHR [7] – це програмне забезпечення для управління людськими ресурсами, яке працює в хмарі, що означає, що до нього можна отримати доступ з будь-якого пристрою з інтернетом (рис. 1.1). Цей додаток призначений для малих та середніх підприємств, допомагаючи їм автоматизувати та спрощувати різні HR процеси.

Одна з ключових функцій BambooHR – це зберігання та організація всіх даних про співробітників в одному місці. Всі особисті дані, історія зайнятості, контактна інформація та інша важлива інформація зберігається в захищеній базі даних, доступній для HR менеджерів.

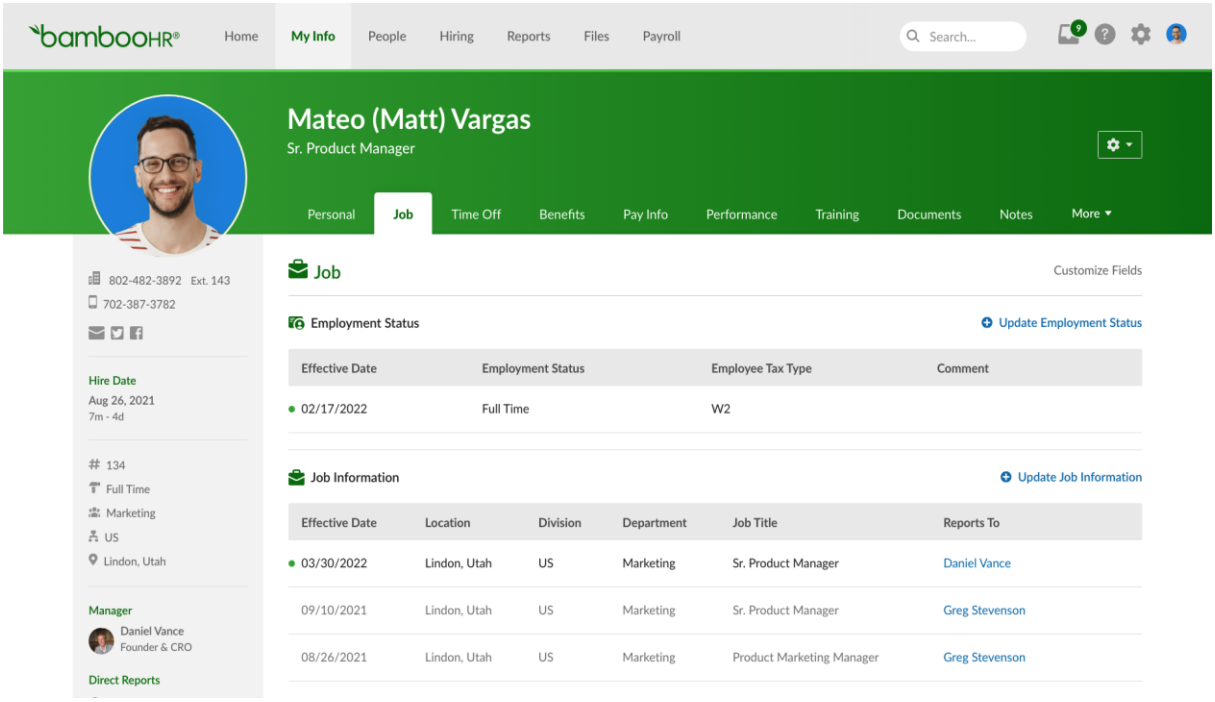

Рисунок 1.1 – Зовнішній вигляд системи BambooHR

Недоліки додатку BambooHR:

1. Інтерфейс користувача: деякі користувачі можуть вважати інтерфейс складним або неінтуїтивним, що може ускладнити процес налаштування та проведення оцінювань. Відсутність дружнього інтерфейсу може знизити ефективність використання додатку.
2. Обмежені функції звітності: хоча BambooHR пропонує базові звіти, деякі користувачі можуть знаходити, що відсутні більш глибокі аналітичні можливості та кастомізовані звіти, необхідні для повного аналізу результатів оцінювань.
3. Висока вартість для малих підприємств: BambooHR може бути відносно дорогим для малих підприємств або стартапів, що може обмежити їх можливість використовувати всі функції додатку. Висока вартість може бути недоліком для компаній з обмеженим бюджетом.

2. Kissflow HR Cloud – це інтуїтивна платформа для управління людськими ресурсами [8], яка дозволяє організаціям автоматизувати та оптимізувати процеси оцінювання працівників (рис. 1.2). Основні функції

оцінювання в Kissflow HR Cloud включають: створення річних та квартальних оцінювальних циклів з індивідуальними метриками та критеріями, автоматизоване нагадування про заплановані оцінювання, визначення та оцінювання ключових компетенцій працівників, порівняння компетенцій з очікуваннями компанії, оцінок та зворотного зв'язку. Кожна з цих функцій спрямована на забезпечення прозорого, об'єктивного та ефективного процесу оцінювання, що сприяє професійному росту працівників та досягненню бізнес-цілей організації.

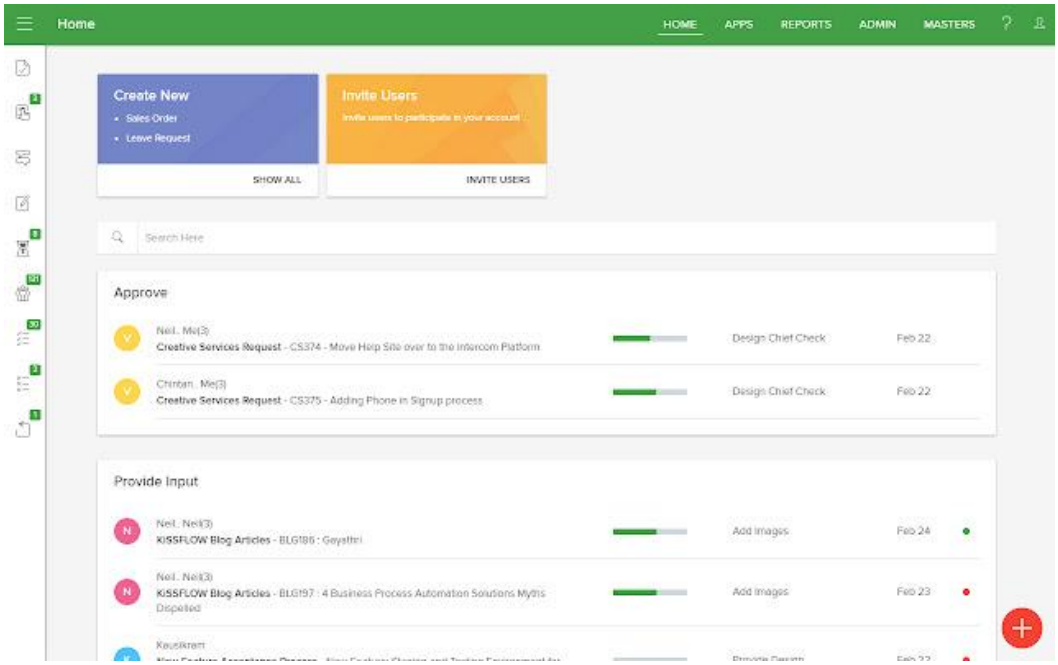

Рисунок 1.2 – Зовнішній вигляд системи Kissflow HR Cloud

1. Складність у налаштуванні: для нових користувачів може бути складно налаштувати систему відповідно до специфічних потреб організації. Процес налаштування може вимагати часу і технічних знань.
2. Відсутність гнучкості у звітуванні: хоча Kissflow пропонує звітні інструменти, деяким користувачам може не вистачати гнучкості у створенні спеціалізованих звітів, які б відповідали їхнім конкретним потребам.

3. Вартість: вартість підписки на Kissflow HR Cloud може бути високою для деяких компаній, особливо для малих і середніх підприємств.

3. SurveyMonkey [9] — це програмна платформа для опитування, яка дозволяє організаціям збирати й аналізувати відгуки від співробітників та інших зацікавлених сторін. Платформа пропонує низку шаблонів опитувань і варіантів налаштування, що полегшує організаціям створення та розповсюдження опитувань для вимірювання продуктивності роботи співробітників, задоволеності роботою та залучення. Однією з ключових особливостей SurveyMonkey є його конструктор опитувань, який дає змогу організаціям створювати персоналізовані опитування з різними типами запитань, включаючи множинні відповіді, відкриті та рейтингові шкали (рис. 1.3). Платформа також пропонує готові шаблони опитувань, які можна використовувати для вимірювання залученості, задоволеності та продуктивності співробітників.

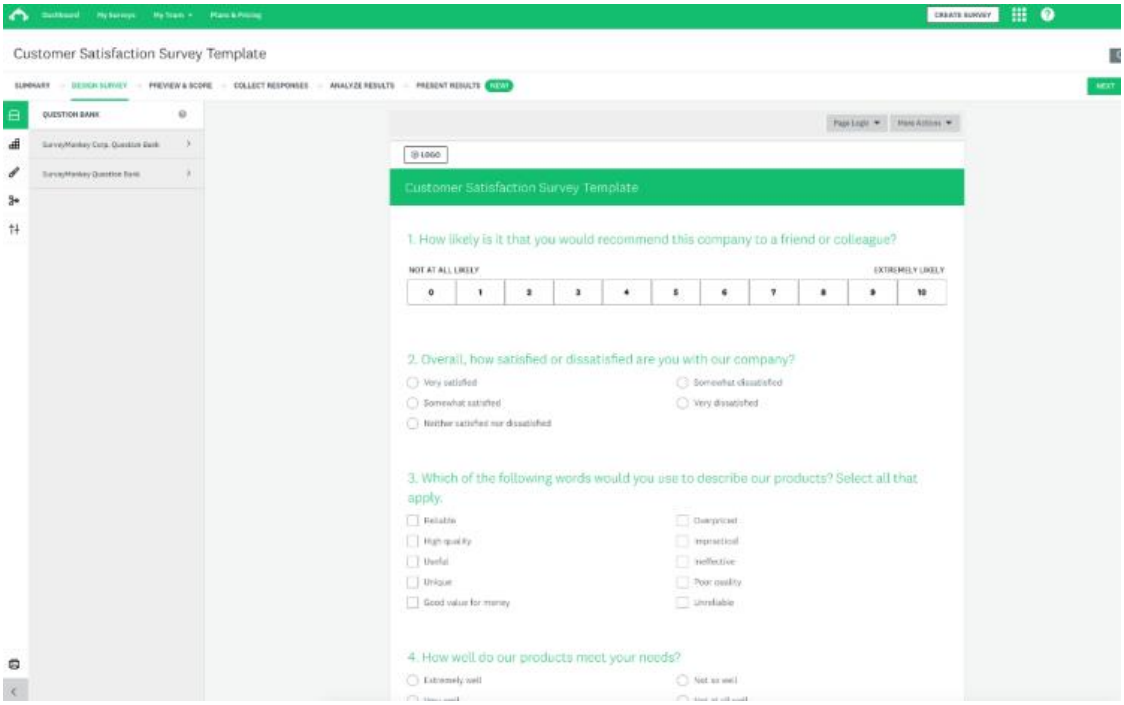

Рисунок 1.3 − Приклад форми опитування у додатку SurveyMonkey

Недоліки додатку SurveyMonkey:

1. Обмежена функціональність у безкоштовній версії: багато корисних функцій, таких як логіка опитування, персоналізація бренду та детальні звіти, доступні лише у платних планах.
2. Обмеження на кількість респондентів: у безкоштовній версії існує обмеження на кількість відповідей на одне опитування, що може бути недостатнім для великих опитувань.
3. Проблеми з користувацьким інтерфейсом: деякі користувачі можуть вважати інтерфейс недостатньо інтуїтивним або незручним для навігації, особливо при створенні складних опитувань.

Кожен з вище наведених додатків має свої переваги та недоліки при використанні. Для наглядності оцінки їх застосування, створимо таблицю. Таблиця порівняння додатків наведена в таблиці 1.2.

Таблиця 1.2. – Порівняння додатків для оцінювання продуктивності роботи співробітників

	BambooHR	Kissflow HR Cloud	Survey Monkey
Створення універсальних оцінювань	+	+	-
Авторизація	+	+	+
Написання відгуків до пройденого оцінювання	-	-	+
Зручний інтерфейс	+	-	+

Отже, виконавши аналіз засобів оцінювання роботи співробітників, можна зробити висновок, що їх суттєвим недоліком є відсутність можливості написання відгуку на пройдені оцінювання та відсутність можливості створення універсальних оцінювань.

1.3 Формулювання вимог та постановка задачі

Розробка WEB-додатків для оцінювання продуктивності роботи співробітників є актуальною темою в сучасному бізнес-середовищі. На ринку існує безліч додатків, спрямованих на вирішення цієї задачі, зокрема такі відомі рішення, як Asana, Trello, Monday.com, та Jira.

Asana є потужним інструментом для ефективністю роботи працівників, який дозволяє відстежувати прогрес виконання завдань, встановлювати дедлайни та призначати відповідальних. Проте, її недоліком є складність налаштувань для специфічних потреб компаній.

Trello є більш простим у використанні інструментом, що базується на методології Kanban. Він дозволяє візуально відстежувати прогрес завдань, але має обмежений функціонал для глибокого аналізу продуктивності.

Monday.com пропонує розширений набір функцій для планування та управління ресурсами, однак його вартість є досить високою, що може бути неприйнятним для малих підприємств.

Jira, яка в основному використовується для управління трудовою діяльністю в IT-сфері, забезпечує потужні можливості для відстеження продуктивності роботи, але потребує значного часу для освоєння та має вузьку специфіку використання.

Загалом, проведення оцінювання продуктивності роботи співробітників у електронному варіанті відкриває можливості для більш ефективного збору даних, швидшого аналізу результатів та покращення загальної об'єктивності процесу. Важливо враховувати як переваги, так і недоліки кожного інструменту та вибирати той, який найкраще задовольняє потреби працівників у конкретних ситуаціях.

Розроблюваний WEB-додаток для оцінювання продуктивності роботи співробітників має на меті підвищити ефективність роботи працівників за

рахунок більш точного і прозорого аналізу їхньої діяльності. В результаті розробки цього WEB-додатку очікується підвищення загальної продуктивності компанії шляхом своєчасного виявлення слабких місць і заохочення кращих працівників, а також забезпечення більш ефективного управління ресурсами та даними. Це дозволить керівництву приймати більш обґрунтовані рішення, спрямовані на оптимізацію робочих процесів і мотивацію персоналу до досягнення більш високих результатів.

WEB-додаток для оцінювання продуктивності роботи співробітників матиме такі функції:

- Можливість авторизації до відповідної посади, ролі (співробітник, керівник, адміністратор).

- Можливість проходження оцінювання та отримання звітності результатів.

- Можливість написання відгуку на пройдене оцінювання.

- Можливість регістрації, нового співробітника адмністратором.

- Можливість створення нового оцінювання, та його редагування адміністратором.

- Можливість створення нового питання ло оцінювання, та його редагування адміністратором.

Усі ці функції мають забезпечити правильний функціонал роботи WEB-додатку. Розроблюваний WEB-додаток має бути зручним у використанні. Під зручністю розуміємо, можливість користувача в будь-який момент пройти необхідне оцінювання, по завершенню отримати його результати, змога переглянути свої попередні оцінювання, використовуючи інтуїтивно зрозумілий та простий WEB-додаток. Також зі сторони адміністратора, змога реєстрації нового працівника, створення та редагування нового оцінювання.

1.4 Висновок до розділу 1

У цьому розділі проаналізовано сучасні методи оцінки продуктивності працівників та обрано найефективніший – 180°, через його збалансоване і об'єктивне оцінювання. Розглянуто вже відомі варіанти програм-аналогів, які використовуються при оцінюванні ефективності роботи працівників. Під час дослідження обговорено проблеми пов’язані з використанням тих чи інших програм, наведено основні переваги та недоліки.

WEB-додаток для оцінювання продуктивності роботи співробітників є актуальним, оскільки він дозволяє автоматизувати процеси оцінки, забезпечує прозорість і об'єктивність у визначенні ефективності роботи, сприяє виявленню сильних та слабких сторін співробітників. Він також допомагає керівництву приймати обґрунтовані рішення щодо розвитку персоналу та покращення загальної продуктивності компанії. Тому створення такого WEB-додатку є доцільним. Такий додаток допоможе зекономити час, підвищити точність оцінки продуктивності співробітників.

Виконано постановку задачі та опис призначення розроблюваного WEB-додатку, як наслідок – поставлено задачі для подальшого дослідження.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ ДЛЯ ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ РОБОТИ СПІВРОБІТНИКІВ

2.1 Обґрунтування вибору архітектури для WEB-додатку

Розробка WEB-додатку вимагає ретельного вибору архітектури, яка забезпечить ефективну та надійну роботу системи [10]. Ось основні причини, чому важливо ретельно вивчити та обґрунтувати вибір архітектури:

1. Масштабованість: архітектура визначає, як побудовано та організовано додаток, що безпосередньо впливає на його здатність до розширення. Правильно обрана архітектура дозволяє ефективно адаптувати систему до зростаючих вимог користувачів і збільшення обсягів даних. Якщо масштабованість недостатня, це може призвести до незадоволеності користувачів, повільної роботи додатка та неконтрольованого зростання витрат на інфраструктуру.

2. Ефективність: правильно підібрана архітектура забезпечує високу продуктивність додатка, що включає швидку обробку запитів, оперативний доступ до даних та мінімальні затримки. Недостатня ефективність може призвести до невдоволення користувачів, зниження продуктивності та обмеження можливостей для подальшого розвитку функціональності.

3. Розширюваність: Архітектура повинна бути гнучкою, щоб легко додавати нові функції, модулі та адаптуватися до майбутніх змін. Правильний вибір архітектури дозволяє інтегрувати новий функціонал без необхідності кардинально змінювати систему. Відсутність розширюваності може обмежувати розвиток додатка та збільшувати витрати на його модифікацію.

На сьогоднішній день існує кілька популярних архітектур, кожна з яких характеризується своїми унікальними особливостями та методами організації компонентів і логіки програмного забезпечення. Серед архітектурних підходів

для веб-додатків можна виділити наступні: монолітну архітектуру, мікросервісну архітектуру та Clean архітектуру.

Монолітна архітектура є традиційним підходом до розробки програмного забезпечення, при якому весь додаток виконується як єдиний, нероздільний блок [11]. Основні компоненти монолітної архітектури включають базу даних, серверну логіку та користувацький інтерфейс.

Переваги монолітної архітектури:

1. Простота: Монолітна архітектура проста у розробці, використанні та супроводі. Усі компоненти додатку знаходяться в єдиному кодовому базисі, що значно полегшує внесення змін та виявлення помилок.

2. Висока продуктивність: Монолітна архітектура не вимагає додаткових витрат на комунікацію між різними компонентами системи. Це дозволяє швидше виконувати запити і забезпечувати стабільно високу продуктивність.

3. Єдина база даних: Усі дані додатку зберігаються в одній базі даних, що значно полегшує управління та міграцію даних.

Недоліки монолітної архітектури:

1. Труднощі з масштабуванням: оскільки додаток розгортається як єдиний блок, масштабувати окремі компоненти складно. При збільшенні кількості користувачів або навантаження на систему можуть виникати проблеми з продуктивністю та масштабованістю.

2. Складність підтримки та оновлення: внесення змін у монолітний додаток є складним процесом, оскільки всі зміни вносяться до єдиного кодового базису. Це може призводити до конфліктів, ускладнювати відстеження помилок і подовжувати цикл розробки.

3. Взаємозалежність компонентів: у монолітній архітектурі компоненти тісно інтегровані, що може створювати проблеми при збоях

одного з компонентів або потребі в його зміні, оскільки це може вплинути на роботу всього додатку.

Незважаючи на недоліки, монолітна архітектура є досить популярним варіантом для невеликих проектів, де простота розробки та супроводу є більш пріоритетною за масштабованість та гнучкість. Монолітна архітектура схематично зображена на рисунку 2.1.

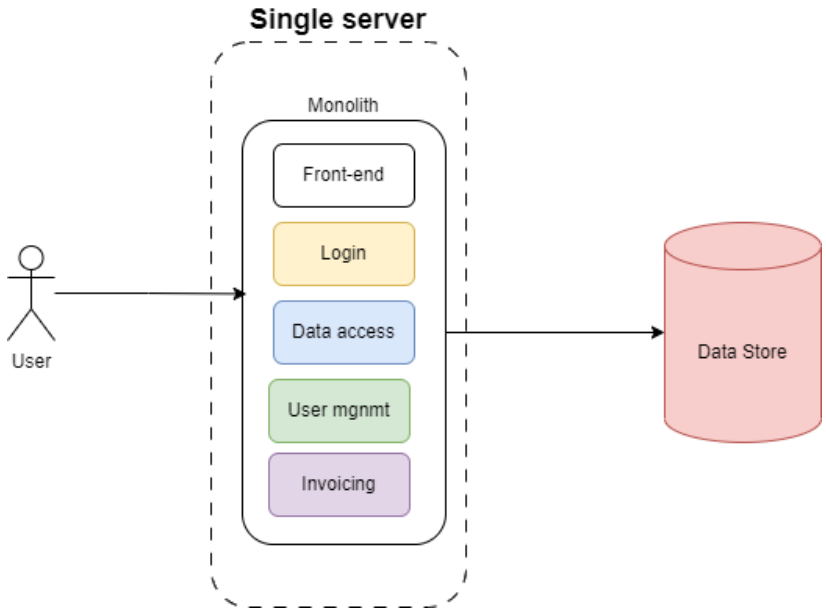

Рисунок 2.1 – Схематичне зображення монолітної архітектури

Мікросервісна архітектура — це метод розробки програмного забезпечення, де додаток розділяється на маленькі, автономні сервіси. Ці сервіси взаємодіють, щоб реалізувати бізнес-логіку системи [13]. Кожен сервіс відповідає за певну функцію, має власну базу даних та інтерфейс.

Основні особливості мікросервісної архітектури:

1. Поділ на функціональні компоненти: бізнес-логіка поділена на невеликі, автономні сервіси. Кожен сервіс виконує чітко визначену функцію, таку як автентифікація користувачів, обробка замовлень або керування продуктами. Це спрощує процеси розробки, тестування та підтримки системи.

2. Автономність і легкість масштабування: кожен сервіс можна розгорнути, масштабувати і оновити незалежно від інших. Це дозволяє ефективно використовувати ресурси, оптимально розподіляти навантаження та забезпечувати високу доступність і гнучкість масштабування системи.

Переваги мікросервісної архітектури:

1. Гнучке масштабування: окремі сервіси можуть легко масштабуватися, дозволяючи системі швидко адаптуватися до змін у навантаженні та підтримувати високу продуктивність.

2. Автономна розробка: кожен сервіс розробляється та вдосконалюється окремо, що спрощує процеси розробки, зменшує ризики та дозволяє командам працювати більш незалежно та ефективно.

Недоліки мікросервісної архітектури:

1. Складність керування: управління численними сервісами та їх взаємодією потребує значних зусиль і розвинених механізмів контролю, що може ускладнювати адміністрування всієї системи.

2. Труднощі тестування**:** розподілена структура мікросервісів робить процес тестування більш складним, оскільки необхідно перевіряти взаємодію між багатьма окремими сервісами, що може бути трудомістким і складним завданням.

Мікросервісна архітектура набирає популярності у розробці WEB-додатків завдяки своїй здатності забезпечувати високу масштабованість, гнучкість і швидке впровадження змін. Проте, перед тим як вибрати цей підхід, важливо уважно оцінити потреби та особливості вашого проекту, оскільки мікросервісна архітектура не завжди є найкращим рішенням і може спричинити певні труднощі та обмеження. Мікросервісна архітектура схематично зображена на рисунку 2.2.

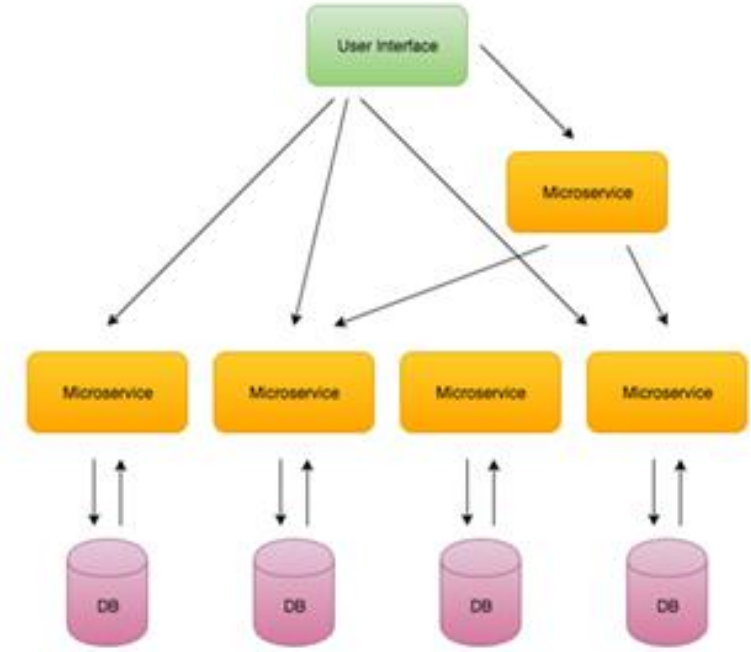

Рисунок 2.2 – Схематичне зображення мікросервісної архітектури

Clean-архітектура — це інноваційний підхід до розробки серверної частини WEB-додатків [14]. Вона акцентує увагу на чіткому розподілі обов'язків і незалежності компонентів. Ця архітектура організована у вигляді шарів, де кожен шар має специфічну роль і взаємодіє з іншими лише в обмеженому обсязі. Такий підхід покращує модульність та спрощує підтримку і розвиток системи.

Основні шари Clean архітектури включають:

1. Доменний шар (Domain Layer): це ядро додатку, яке містить бізнес-логіку та основні правила домену. Він незалежний від зовнішніх фреймворків та інфраструктури, що дозволяє легко переносити його на інші платформи.

2. Шар застосунків (Application Layer): цей шар координує взаємодію між інтерфейсом користувача і доменним шаром. Він містить бізнес-процеси та правила, які визначають обробку зовнішніх запитів.

3. Шар інтерфейсів (Interface Layer): цей шар відповідає за всі зовнішні інтерфейси системи, такі як REST API, веб-інтерфейси та інші методи

комунікації з клієнтами. Його завданням є забезпечення ефективної взаємодії між користувачами та системою, перетворюючи зовнішні дані у зрозумілий формат для доменного шару.

4. Інфраструктурний шар (Infrastructure Layer): він включає реалізацію всіх зовнішніх компонентів, таких як бази даних, зовнішні сервіси та кешування. Цей шар відповідає за інтеграцію додатку із зовнішнім середовищем та доступ до необхідних ресурсів.

5. Шар представлення (Presentation Layer): цей додатковий шар відповідає за відображення даних користувачам і обробку їх введення. Він включає в себе інтерфейси, такі як веб-сторінки, мобільні додатки та настільні програми, що безпосередньо взаємодіють з користувачами.

Переваги Clean архітектури включають:

1. Модульність і незалежність компонентів: кожен шар має чітко визначену функцію, що дозволяє змінювати або замінювати його без впливу на інші частини системи. Це сприяє гнучкості та полегшує обслуговування.

2. Зручність тестування**:** завдяки чітким межам між шарами, тестування окремих компонентів стає простішим і ефективнішим. Це забезпечує легкість у виявленні та виправленні помилок.

3. Незалежність від зовнішніх фреймворків: Доменний шар, який містить бізнес-логіку, не залежить від конкретних технологій або фреймворків. Це робить систему більш універсальною, гнучкою та легкою для масштабування.

Недоліки Clean архітектури включають:

1. Висока складність: clean-архітектура може бути складною для розуміння та впровадження, особливо для новачків у розробці. Це вимагає додаткового навчання та часу для адаптації.

2. Збільшений обсяг коду: поділ функціональності на окремі шари може призвести до значного збільшення кількості коду, який потрібно писати та підтримувати, що ускладнює процес розробки.

3. Повільний розвиток: чіткі межі між шарами можуть уповільнити впровадження нових функцій, особливо на початкових етапах проекту, коли структура системи ще формується.

Незважаючи на ці недоліки, Clean-архітектура є ефективним підходом до розробки програмного забезпечення, який дозволяє створювати модульні, гнучкі та легко тестовані системи, що спрощує їхню підтримку та розвиток у довгостроковій перспективі. Clean архітектура схематично зображена на рисунку 2.3.

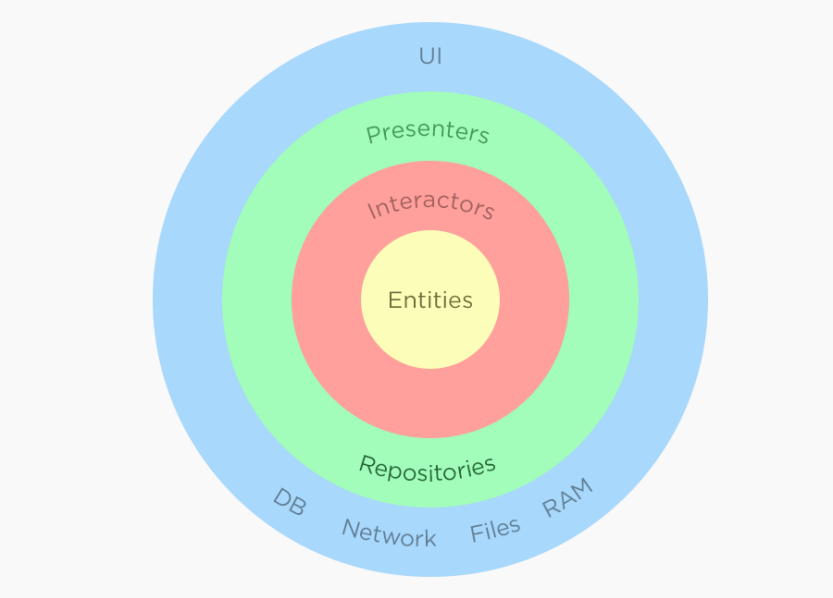

Рисунок 2.3 – Схематичне зображення Clean архітектури

Під час аналізу існуючих архітектур для WEB-додатку, їхніх переваг і недоліків, було обрано Clean архітектуру через її переваги у сфері модульності, тестованості та гнучкості. Ця архітектура дозволяє ефективно організувати структуру WEB-додатку, зменшити залежність від зовнішніх факторів та сприяє подальшому розвитку та підтримці WEB-додатку.

2.2 Проектування WEB-додатку із застосуванням UML-діаграм

Суть UML-діаграми варіантів використання полягає в документуванні функціональних вимог до системи з точки зору користувачів [15]. Вона надає загальне уявлення про взаємодію користувачів із системою та операції, які вони можуть виконувати. Це допомагає зрозуміти основні сценарії використання системи та їх послідовність.

У процесі розробки UML-діаграми варіантів використання спочатку визначаються основні актори, які взаємодіють із системою. У нашому випадку такими акторами є адміністратор, працівник та керівник. Потім ідентифікуються ключові варіанти використання, які описують основні дії та функціональні можливості системи. Кожен варіант використання складається з акторів, сценаріїв та можливих результатів.

У процесі розробки системи розглядаються різноманітні сценарії її використання різними типами користувачів. Для моделювання цих сценаріїв створюється UML-діаграма варіантів використання, яка демонструє послідовність дій та взаємодію акторів із системою. Ця діаграма слугує важливим інструментом для розуміння потреб користувачів і планування функціональних можливостей системи.

Кожен актор має свої функціональні можливості та цілі при взаємодії із системою. При вході до додатку користувачу буде надана можливість обрати роль: співробітник, керівник або адміністратор.

Співробітник може переглядати свої вже пройдені оцінювання, проходити нові, якщо те необхідно. Він також може після завершення оцінювання побачити свої результати, та надати, отримати відгук.

Керівник має можливість переглядати свої та своїх підлеглих вже пройдені оцінювання. Він також може проходити нові оцінювання для себе та своїх підлеглих. Після проходження тестування, керівник має змогу побачити

результати та за ними оцінити наскільки працівник ефективно працює. Як результат, можна написати відгук на його результати.

Адміністратор відповідає за управління користувачами. Він може створювати, редагувати та видаляти аккаунти співробітників, що дозволяє підтримувати безпеку та дотримання правил. Адміністратор також має доступ до створення та редагування оцінювань та питань до оцінювань.

Розроблена UML-діаграма варіантів використання зображена на рисунку 2.4. Вона відображає функціональні можливості кожного актора та дозволяє зрозуміти, як вони взаємодіють з системою. Це сприяє створенню ефективної платформи, де керівники можуть відстежувати прогрес, надавати зворотній зв'язок та визначати області для покращення, тоді як співробітники можуть отримувати чітке розуміння своїх досягнень та цілей.

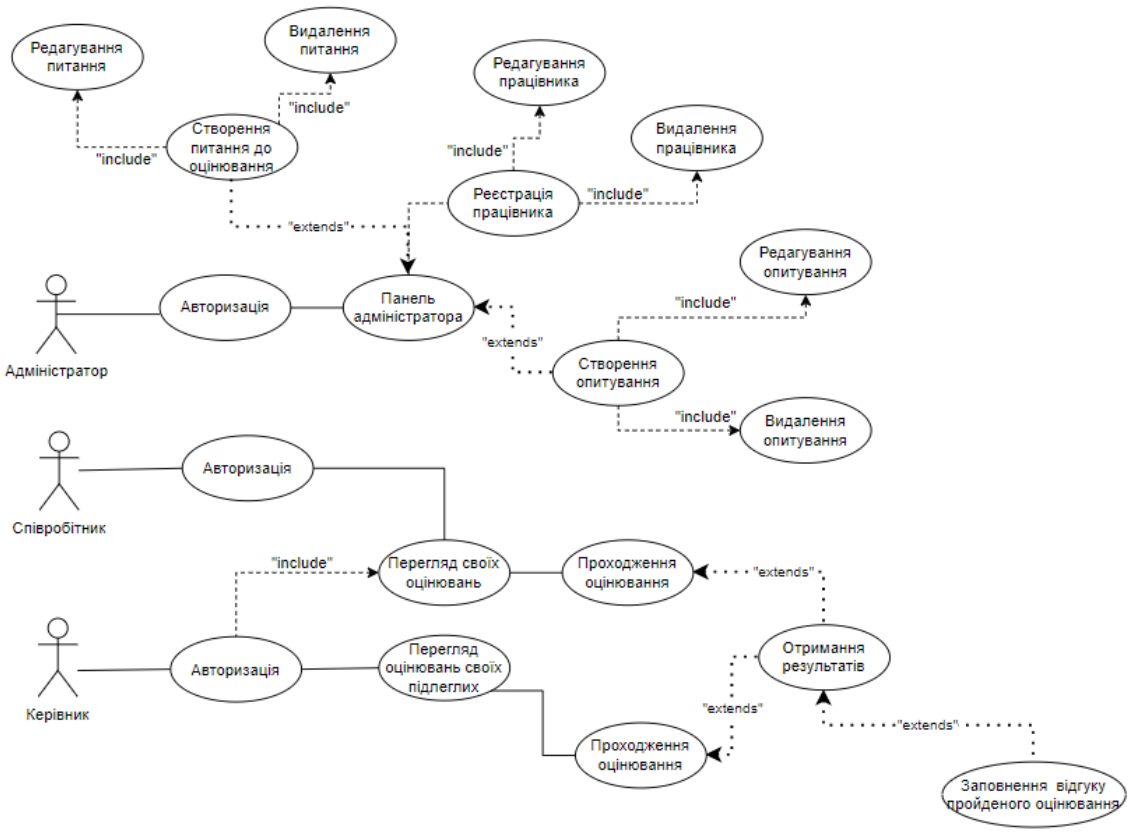

Рисунок 2.4 – UML-діаграма прецедентів WEB-додатку для оцінювання продуктивності роботи співробітників

Діаграми взаємодій (Use Case Diagrams) є важливим інструментом у моделюванні системного аналізу та розробки програмного забезпечення [16]. Вони допомагають наочно представити функціональні вимоги системи і демонструють, як користувачі (актори) взаємодіють із системою.

Основною метою UML-діаграм взаємодій є:

1. Забезпечення чіткого розуміння вимог: UML-діаграми взаємодій дозволяють точно визначити, які функції повинна виконувати система і які дії можуть здійснювати користувачі. Це допомагає проектувальникам і розробникам краще уявити роботу системи.

2. Визначення акторів та їхніх ролей: Діаграма взаємодій дозволяє ідентифікувати різних акторів (користувачів, системи, зовнішні сервіси тощо), які взаємодіють із системою. Вона вказує на ролі акторів у процесі взаємодії із системою.

3. Встановлення меж системи: Діаграма взаємодій дозволяє чітко окреслити кордони системи, визначаючи, які взаємодії відбуваються всередині системи, а які поза її межами.

4. Виявлення основних сценаріїв взаємодії: Діаграма взаємодій окреслює ключові сценарії взаємодії між акторами та системою, допомагаючи виявити основні функції і процеси, що відбуваються в системі.

5. Надання основи для подальшого аналізу та проектування: Діаграма взаємодій служить базою для більш детального аналізу і проектування системи. Вона використовується для створення детальніших моделей і діаграм, таких як діаграми послідовностей та діяльності.

На рисунку 2.5 наведено UML-діаграму взаємодій для WEB-додатку для оцінювання продуктивності роботи співробітників.

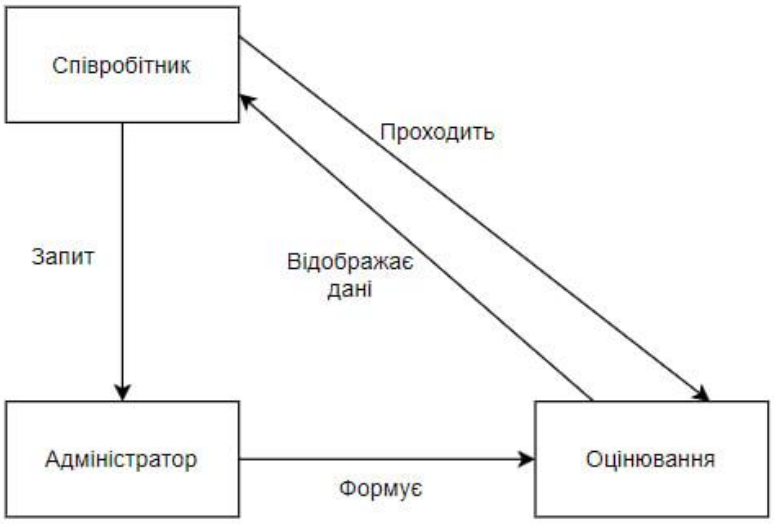

Рисунок 2.5 – UML-діаграма взаємодій WEB-додатку для оцінювання продуктивності роботи співробітників

Розробка програм на мові програмування Java ґрунтується на об’єктно-орієнтованому підході, де ключовими концепціями є "клас" і "об'єкт" [18]. Об'єктно-орієнтоване програмування (ООП) дозволяє програмістам структурувати завдання на логічно завершені одиниці, що реалізовуються через класи. Кожен клас функціонує як шаблон або опис, за допомогою якого можна створювати конкретні екземпляри класів.

Підхід ООП передбачає, що розробник коду повинен вміти:

- аналізувати задачу та розглядати її з точки зору об'єктів, розбиваючи її на класи з конкретними функціями або відповідальностями;
- встановлювати та управляти зв'язками між класами для досягнення необхідного функціоналу та забезпечення ефективної комунікації;
- виділяти та реалізовувати багаторазово використовуваний код у вигляді окремих класів з метою уникнення дублювання коду та дотримання принципу DRY (Don't Repeat Yourself);

- створювати ієрархію класів від простого до складного для організації коду за рівнем абстракції та полегшення повторного використання;
- розуміти деталі реалізації можливостей, які надають класи, такі як інкапсуляція, спадковість та поліморфізм, для ефективного створення програм.

В програмному модулі, що розробляється будуть такі класи: User, Survey, Position, Employee, Answers та Question, що слугують для відображення сутностей сторінок.

Клас User: цей клас представляє користувача системи, що може бути як співробітником, так і менеджером або адміністратором.

Атрибути: user_id: унікальний ідентифікатор користувача, name: ім'я користувача, email: електронна адреса користувача, password: пароль для входу в систему.

Функції: login(email, password): функція для входу користувача в систему. Перевіряє правильність введених даних. logout(): Функція для виходу користувача з системи. isAdmin(): Функція перевірки чи є користувач адміністратором.

Клас Survey: цей клас представляє оцінювання, що може бути використане для оцінки продуктивності співробітників.

Атрибути: survey_id: унікальний ідентифікатор оцінювання, title: назва оцінювання, description: опис оцінювання, questions: список питань, що входять до оцінювання.

Функції: add_question(question): Додає нове питання до оцінювання. remove_question(question_id): Видаляє питання з оцінювання за допомогою його ідентифікатора.

Клас Position: цей клас представляє посаду співробітника.

Атрибути: position_id: унікальний ідентифікатор посади, title: назва посади, department: відділ, до якого належить посада.

Функції: update_title(new_title): Оновлює назву посади. update_department(new_department): Оновлює відділ, до якого належить посада.

Клас Answers: цей клас представляє відповіді на питання оцінювання.

Атрибути: answer_id: унікальний ідентифікатор відповіді, employee_id: ідентифікатор співробітника, що дав відповідь, question_id: ідентифікатор питання, на яке було дано відповідь, answer_text: текст відповіді.

Функції: update_answer(new_answer_text): Оновлює текст відповіді.

Клас Question: цей клас представляє питання оцінювання.

Атрибути: question_id: унікальний ідентифікатор питання, text: текст питання, question_type: тип питання (наприклад, відкрите, закрите, з вибором тощо).

Функції: update_text(new_text): Оновлює текст питання. update_type(new_type): Оновлює тип питання.

На рис. 2.6 показано загальну UML-діаграму класів програмного модуля.

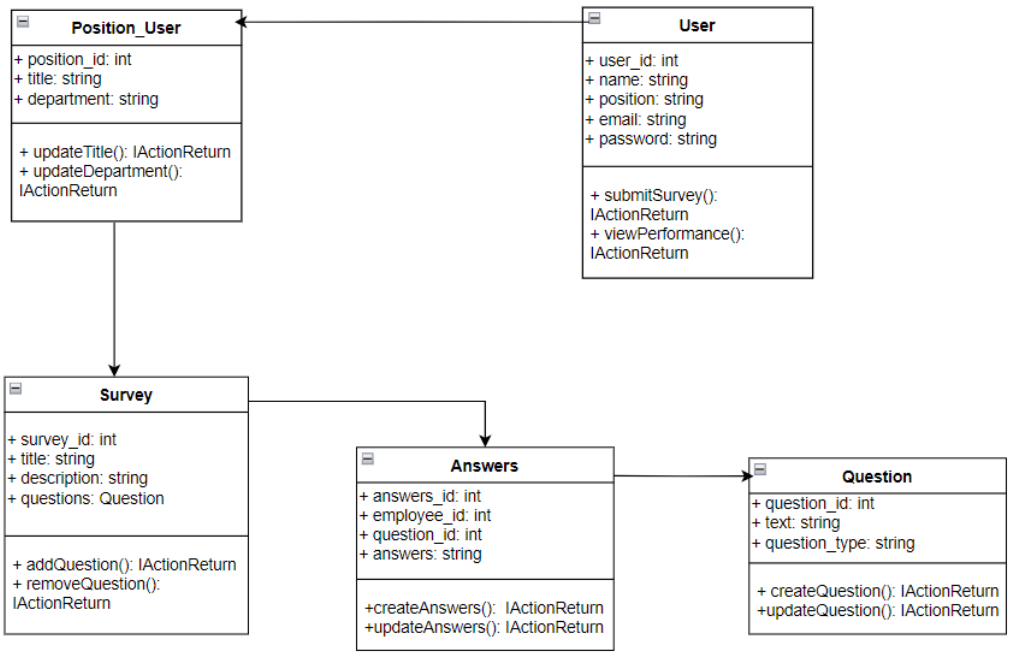

Рисунок 2.6 – Загальна UML-діаграма класів

UML-діаграма розгортання є одним із видів UML-діаграм (Unified Modeling Language) і використовується для моделювання фізичної архітектури системи. Головна мета цієї діаграми – показати, як різні компоненти WEB-додатку будуть розміщені на фізичних серверах. Для цього визначаються фізичні вузли (сервери) та встановлюються зв'язки з компонентами системи.

При створенні діаграми розгортання слід враховувати такі аспекти [17]:

- Фізичні сервери: Визначаються сервери, на яких будуть розгорнуті компоненти веб-додатку. Необхідно провести детальний аналіз щодо вибору серверів, їх налаштування та забезпечення необхідних ресурсів для оптимальної роботи системи.
- Компоненти системи: Вказується, які компоненти веб-додатку будуть розміщені на кожному сервері або платформі. Це можуть бути WEB-сервери, сервери баз даних, кешувальні сервери, сервери обробки запитів, а також інші спеціалізовані сервери і залежності між компонентами.

Використання UML-діаграми розгортання допомагає глибше зрозуміти фізичну структуру веб-додатку та визначити ідеальне розташування його складових. Цей інструмент не лише надає можливість оцінити, як система буде масштабуватися, але і гарантує високу доступність та ефективність функціонування. Діаграма розгортання WEB-додатку зображена на рисунку 2.7.

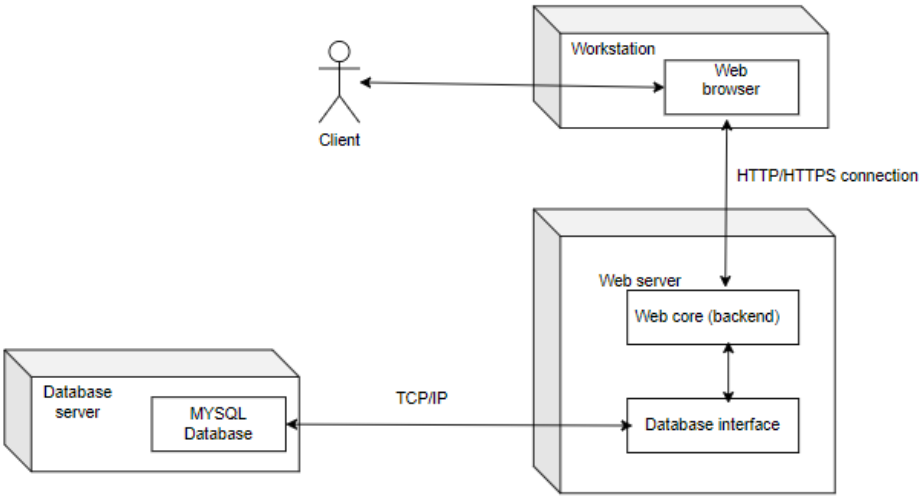

Рисунок 2.7 – Діаграма розгортання WEB-додатку для оцінювання продуктивності роботи співробітників

При розгортанні розробленого WEB-додатку використовуються наступні компоненти:

1. Клієнтська частина WEB-додатку: Це інтерфейс, через який користувачі отримують доступ до функціоналу системи. Вона буде розгорнута на клієнтських пристроях, таких як комп'ютери, смартфони або планшети, і забезпечить взаємодію користувачів із системою та передачу даних між клієнтом і сервером.

2. Серверна частина WEB-додатку: Цей компонент відповідає за обробку запитів від клієнтів, бізнес-логіку системи та зберігання даних. Вона буде розгорнута на серверних вузлах, що забезпечують високу доступність та швидкість відповіді. Серверна частина обробляє запити, взаємодіє з базою даних та надає необхідну функціональність клієнтам.

3. Система управління базами даних (СУБД): Компонент, що відповідає за зберігання та управління даними, які використовуються в системі. Вона забезпечує ефективну організацію та доступ до даних, а також цілісність і безпеку інформації. Система управління базами даних може використовувати реляційну модель для зберігання структурованих даних.

2.3 Розробка схеми алгоритму роботи WEB-додатку для оцінювання продуктивності роботи співробітників

Алгоритм – це скінчена послідовність інструкцій для виконання дій, спрямованих на розв'язання певної задачі [19]. Алгоритм може бути візуалізований за допомогою блок-схем. Для комп'ютерних програм алгоритм є списком деталізованих інструкцій, що реалізують процес обчислення.

Опис алгоритму може бути поданий у вигляді псевдокоду, блок-схеми, програмного коду або простого тексту, залежно від контексту та характеру задачі.

Псевдокод: це спеціальний формат, схожий на програмний код, але не є повністю синтаксично правильним. Він використовується для передачі загальної логіки алгоритму та послідовності операцій.

Блок-схема: це графічне зображення алгоритму, яке складається з блоків та зв'язків між ними. Блок-схема допомагає візуалізувати послідовність дій, умови та повторення.

Програмний код: це конкретна реалізація алгоритму у вигляді коду для певної мови програмування. Код визначається вибраною мовою програмування і має відповідний синтаксис та точність для виконання певних завдань.

Отже, для розробки WEB-додатку для оцінювання продуктивності роботи співробітників оберемо блок-схему для опису алгоритму. Схема алгоритму подана на рис. 2.8.

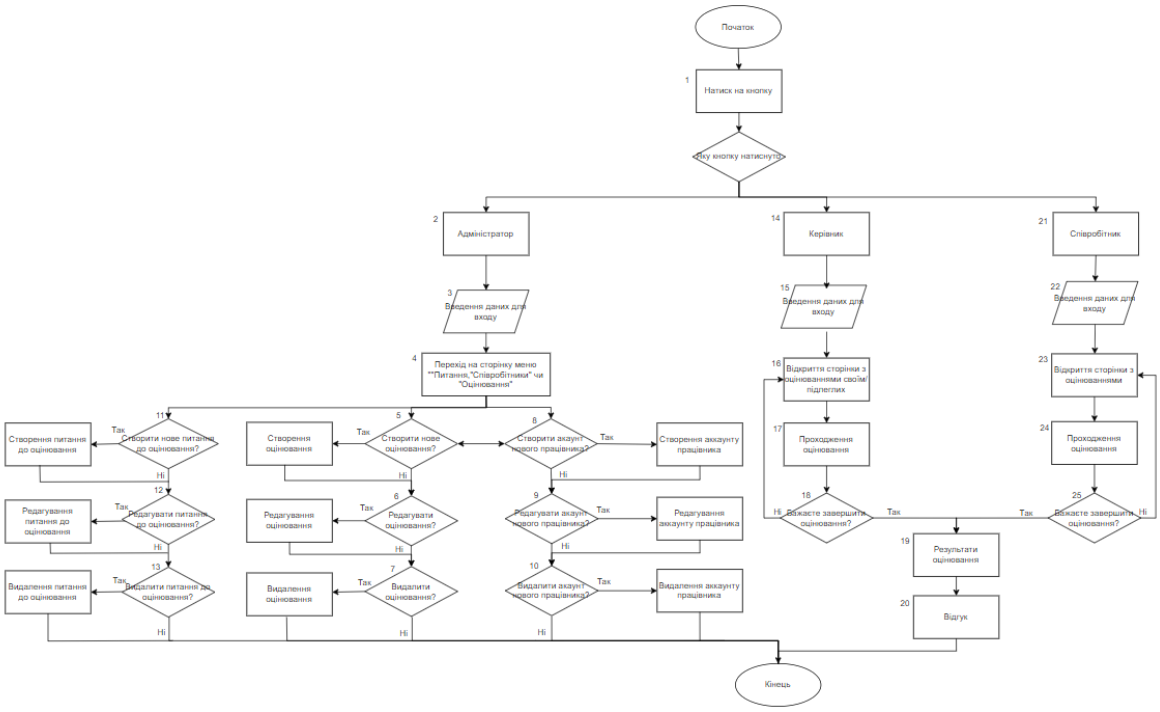

Рисунок 2.8 – Схема алгоритму роботи WEB-додатку для оцінювання продуктивності роботи співробітників

І випадок

Алгоритм у випадку натиснення на кнопку «Співробітник» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає «Співробітник», «Керівник», «Адміністратор»;

Крок 2. Введення вхідних даних: логін, пароль;

Крок 3. Перевірка чи є такий користувач у БД;

Крок 4. Вхід у систему.

ІІ випадок

Алгоритм у випадку натиснення на кнопку «Керівник» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає «Співробітник», «Керівник», «Адміністратор»;

Крок 2. Введення вхідних даних: логін, пароль;

Крок 3. Перевірка чи є такий користувач у БД;

Крок 4. Вхід у систему.

III випадок

Алгоритм у випадку натиснення на кнопку «Адміністратор» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає «Співробітник», «Керівник», «Адміністратор»;

Крок 2. Введення вхідних даних: логін, пароль;

Крок 3. Перевірка чи є такий користувач у БД;

Крок 4. Вхід у систему.

IV випадок

Алгоритм у випадку «Обрання оцінювання» складається з таких кроків:

Крок 1. Завантаження початкових даних»;

Крок 2. Отримання даних про оцінювання з БД;

Крок 3. Користувач обирає оцінювання яке необхідно пройти;

Крок 4. Перевірка чи існує таке оцінювання у БД.

Крок 5. Користувач зайшов на оцінювання.

V випадок

Алгоритм у випадку «Проходження оцінювання» складається з таких кроків:

Крок 1. Завантаження початкових даних»;

Крок 2. Отримання даних про питання для оцінювання з БД;

Крок 3. Користувач відповідає на питання;

Крок 4. Можна скористатись кнопкою «Зберегти» та закінчити оцінювання наступного разу;

Крок 5. Користувач надсилає відповіді на пройдене оцінювання.

VI випадок

Алгоритм у випадку натисненні кнопки «Зберегти» складається з таких кроків:

Крок 1. Завантаження початкових даних»;

Крок 2. Отримання даних про оцінювання та питання для оцінювання з БД;

Крок 3. Збереження усіх відповідей;

VII випадок

Алгоритм у випадку натиснення кнопки «Готово» складається з таких кроків:

Крок 1. Завантаження початкових даних»;

Крок 2. Отримання даних про оцінювання та питання для оцінювання з БД;

Крок 3. Перевірка чи користувач відповів на усі питання;

Крок 4. Оцінювання пройдено.

VIII випадок

Алгоритм у випадку, коли і «Співробітник і керівник пройшли оцінювання» складається з таких кроків:

Крок 2. Отримання даних про оцінювання від працівника та керівника з БД;

Крок 3. Виведення результатів оцінювання;

Крок 4. Співробітник і керівник заповнюють форму відгуку на пройдене оцінювання.

2.4 Висновок до розділу 2

У цьому розділі проаналізовано та сформульовано основні вимоги до програмного забезпечення.

Обрано архітектуру Clean для WEB-додатку для оцінювання продуктивності роботи співробітників, що спростить процес розробки. WEB-додаток міститиме такі сторінки: сторінка авторизації, сторінка з оцінюваннями, сторінка самого оцінювання та сторінка з результатами.

Також створено UML-діаграми варіантів використання, взаємодій, розгортання та загальну UML-діаграму класів. Використання UML-діаграм класів дозволяє візуалізувати структуру програми та методи, присутні в кожному класі. Це надає швидкий і зручний спосіб отримати інформацію про класи та їхні взаємозв'язки. За допомогою цих діаграм можна аналізувати структуру системи, розкривати деталі та проектувати програму з точки зору об'єктно-орієнтованого підходу. Розробка діаграм класів є важливою складовою процесу розробки програмного забезпечення. Вона дозволяє провести декомпозицію предметної галузі, визначити необхідні класи та їхні взаємозв'язки. Це допомагає краще розуміти сутність проблеми та визначати, які аспекти треба враховувати, а які можна ігнорувати.

Крім того, була розроблена схема алгоритму функціонування WEB-додатку. Подана схема алгоритму допомагає крок за кроком уявити, як програма буде працювати та взаємодіяти з іншими елементами, що сприяє кращому розумінню поставленого завдання та допомагає уникнути помилок при реалізації функціоналу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ ДЛЯ ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ РОБОТИ СПІВРОБІТНИКІВ

3.1 Обґрунтування вибору мови програмування та фреймворків

На початковому етапі розробки програмного забезпечення, що відбувається на стадії проектування, вибір мови програмування має вирішальне значення. Для здійснення оптимального вибору проводиться аналіз вимог і порівняння різних мов програмування. Оцінюється можливість кожної мови задовольнити необхідні функціональні вимоги та відтворити всі необхідні функції у розроблюваному додатку.

Для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників ми проаналізуємо мови програмування Python, C++ та Java [20-22].

C++ використовується для створення різноманітних програмних продуктів і вважається однією з найбільш популярних мов програмування. За допомогою C++ розробляються операційні системи, програми з різних галузей, драйвери пристроїв та ігри.

Ця мова є розширенням мови програмування C і включає об'єктно-орієнтований підхід. Вона надає можливість програмувати як у стилі C, так і в об'єктно-орієнтованому стилі. Основними концепціями C++ є класи, які можуть містити поля та методи. Поля можуть бути публічними, захищеними або приватними, а методи - функціями-членами для реалізації певної функціональності.

Однією з привабливих особливостей C++ є можливість перевантаження операторів, наприклад, оператора додавання. Крім того, мова містить різноманітні оператори, такі як оператори порівняння, арифметичні оператори, оператори обробки бітів і логічні оператори.

Стандартна бібліотека C++ включає стандартну бібліотеку C з невеликими змінами для кращої сумісності з C++. Це робить її більш зручною для використання у програмах, написаних на C++.

Python - це загальнопризначена високорівнева мова програмування, що базується на принципах простоти та зрозумілості коду. Заснована Гвідо ван Россумом у 1991 році, вона відома своєю легкістю вивчення та розробки.

Синтаксис Python відрізняється від мови C, що робить його дуже зрозумілим і доступним навіть для новачків. Використання відступів (пробілів або табуляції) для визначення блоків коду робить його більш структурованим і легким для читання.

Це інтерпретована мова, що означає, що код виконується рядок за рядком без необхідності попередньої компіляції. Це сприяє ітеративній розробці та тестуванню, а також зменшує час розгортання проекту.

Python підтримує об'єктно-орієнтоване програмування, що дозволяє розробникам створювати класи та об'єкти з властивостями та методами. Він також підтримує інші парадигми програмування, такі як функціональне програмування, що розширює можливості розробників та сприяє використанню різних підходів при розв'язанні завдань.

Однією з головних переваг Python є великий вибір бібліотек та фреймворків, які спрощують розробку різноманітних додатків. Наприклад, Django та Flask - популярні веб-фреймворки, що дозволяють швидко створювати веб-додатки. Існують також спеціалізовані бібліотеки для наукових обчислень (NumPy, Pandas), машинного навчання (TensorFlow, scikit-learn) та багатьох інших галузей.

Java — це мова програмування, яка зазвичай використовується для написання Інтернет-додатків. Мільйони програм на Java використовуються й сьогодні. Java — це кросплатформна об’єктно-орієнтована мова, яку можна використовувати як окрему платформу. Це швидка, безпечна та надійна мова програмування для всього: від мобільних програм і корпоративного

програмного забезпечення до програм для великих даних і серверних технологій.

Java була розроблена з метою нагадувати мову програмування C++, але бути більш зручною у використанні. Її використовують для створення невеликих модулів програм або аплетів, які можна використовувати як частину веб-сторінки. Також Java можна використовувати для створення повноцінних програм, які можуть працювати на одному комп’ютері або розподілятися по мережі між серверами та клієнтами.

Використовуючи бібліотеки мови Java, можна швидко розробляти веб-сайти, створювати ігри, розробляти програми для ПК або мобільних пристроїв чи планшетів. Бібліотеки розширюють функціональність мови, і після вивчення базового синтаксису можна робити майже все, що завгодно. Наприклад, якщо потрібно запрограмувати мікроконтролер або обробити великі обсяги даних, Java впорається з цим завданням.

Java також використовує автоматичний збирач сміття для управління життєвим циклом об'єктів. На дизайн Java вплинула мова C++, яка зробила Java насамперед об'єктно-орієнтованою мовою. Це означає, що інструкції не можуть містити адреси даних, що зберігаються в інших програмах або в самій операційній системі, що може призвести до завершення роботи програм.

Java забезпечує надійність коду, оскільки об'єкти Java, на відміну від C++, не містять посилань на зовнішні дані або інші об'єкти. Це означає, що інструкції не можуть містити адреси даних, що зберігаються в інших програмах або в самій операційній системі, що може призвести до завершення роботи програм.

Дані, перетворені Java в байт-код, не є читабельними, і Java запускає програми всередині пісочниці, щоб запобігти змінам з невідомих джерел. Java є легкою для вивчення мовою програмування.

Крім того, існує кілька різних рівнів складності та різних підходів фреймворків для розробки вебзастосунків, таких як фреймворки для

структурування застосунків на основі патерну MVC (Struts, Spring), бібліотеки для створення шаблонів вебсторінок (Velocity, Java ServerFaces), і бібліотеки для відображення реляційної таблиці на об’єкти і навпаки (Hibernate).

Наявність переваг та недоліків у вище перелічених мов програмування, які демонструють їх характерні відмінності, продемонстровано в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика мов програмування

	C++	Python	Java
Зрозумілість коду	-	+	+
Швидкодія	-	-	+
Підтримка ООП	+	+	+
Зручність для обчисленнята аналізу даних	+	-	+
Надійність передачі даних у мережі пристроїв	+	-	+

Проаналізувавши результати, можна зробити висновок, що Java є лідером серед конкурентів, тому для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників буде використана вона.

Для розробки основного функціоналу та модуля безпеки вибраний потужний фреймворк Spring [23]. У сучасних ентерпрайз-проєктах важко уявити Java-проєкт, який обходиться без використання Spring. Варто зазначити, що Spring – це не лише один клас чи технологія, яка складається з різних пакетів, незалежних один від одного, об’єднаних під загальною назвою.

Перед тим, як розпочати проєктування, важливо проаналізувати та зрозуміти, які пакети можна використовувати. Spring Framework (або просто

Spring) – це універсальний фреймворк з відкритим кодом для Java-платформи, який надає добре документовані та легкі у використанні засоби для розв’язання проблем, що виникають при створенні масштабних застосунків.

Особливості ядра Spring можна використовувати в будь-якій Java аплікації, і наявність безлічі розширень і покращень дозволяє будувати веб-застосунки на Java Enterprise платформі. Це призвело до великої популярності Spring серед програмістів, і його вважають стратегічно важливим фреймворком (рис. 3.1).

Один з складових у фреймворку – Spring Security [24], є інструментом, що забезпечує механізми для побудови систем впізнавання та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Основні характеристики Spring Security включають:

- аутентифікацію та авторизацію користувачів;
- захист від атак, таких як фіксація сесії, підробка міжсайтових запитів і інші;
- опціонально підключається модуль Spring Web MVC.

Spring Security дозволяє ефективно впроваджувати механізми безпеки в корпоративних застосунках, що додає значний рівень захисту від потенційних загроз.

Рисунок 3.1 – Схематичне зображення основних технологій Spring

Spring Boot - це комплексний фреймворк для створення і виведення застосунків з мінімальними зусиллями і конфігураціями [25]. Він розділяється на дві високотехнологічні стопки: заснований на API сервлетах Spring MVC і реактивний Spring WebFlux.

Основні характеристики Spring Boot включають:

- вбудовані контейнери Tomcat, Jetty або Undertow, які працюють безпосередньо без розгортання War-файлів;
- готові стартові залежності, що спрощують конфігурацію збірки;
- можливість конфігурування проєкту прямо в браузері за допомогою Spring Initializr;
- автоматичне налаштування сторонніх бібліотек (по можливості).

У проєктах часто застосовують також Spring Data. Spring Data – це компонент, який дозволяє застосункам доступ до даних через різні типи баз даних, фреймворки map-reduce і хмарні послуги. Spring Data має підпроекти для різних СУБД, таких як MySQL, MongoDB, Redis, а також можливість

використання підмодулів для більш специфічних баз даних, таких як ArangoDB, Google Datastore, Microsoft Azure Cosmos DB та інші.

Spring Data реалізує основний механізм через репозиторії, які є набором інтерфейсів для взаємодії з даними за допомогою JPA Entity. Вони мають ряд функцій, таких як:

- створення динамічних запитів до бази даних;
- базові класи для різних завдань;
- прозорий аудит об’єктів;
- можливість інтеграції власного коду в репозиторії;
- легка інтеграція з Spring через JavaConfig або кастомні XML-простори імен;
- розширена інтеграція з контролерами Spring MVC.

Spring Data використовується широко для забезпечення доступу до даних і легко інтегрується з іншими модулями Spring.

Для написання клієнтської частини WEB-додатку існує кілька популярних фреймворків, серед яких варто виділити Vaadin, React та Angular. Кожен з них має свої особливості та переваги, які впливають на вибір для конкретного проекту.

Vaadin — це фреймворк для створення сучасних веб-додатків на Java [26]. Він дозволяє розробникам створювати повнофункціональні веб-інтерфейси, використовуючи Java, без необхідності писати код на JavaScript або HTML. Vaadin надає багатий набір UI-компонентів, що дозволяє швидко створювати складні інтерфейси.

Переваги Vaadin:

1. Java як основна мова: Дозволяє розробникам Java працювати у звичному середовищі.
2. Повноцінні UI-компоненти: Бібліотека компонентів Vaadin включає таблиці, форми, графіки тощо.

3. Безпека: Vaadin автоматично обробляє безпеку на рівні клієнт-сервер.
4. Продуктивність: Підтримує серверний рендеринг, що покращує продуктивність.

React — це бібліотека JavaScript, створена компанією Facebook, яка використовується для створення користувацьких інтерфейсів [27]. React зосереджений на компонентному підході та дозволяє створювати повторно використовувані UI-компоненти.

Переваги React:

1. Компонентний підхід: Легко створювати та повторно використовувати компоненти.
2. Віртуальний DOM: Забезпечує високу продуктивність при оновленні інтерфейсу.
3. Велика спільнота: Багато готових рішень та бібліотек.
4. Гнучкість: Можна використовувати з різними технологіями та фреймворками.

Angular — це платформа для розробки веб-додатків, створена компанією Google [28]. Angular надає повний набір інструментів для розробки, включаючи двосторонню прив'язку даних, DI (Dependency Injection), шаблони, роутінг тощо.

Переваги Angular:

1. Повний стек: Надає всі необхідні інструменти для розробки веб-додатків.
2. Двостороння прив'язка даних: Автоматично синхронізує модель та вигляд.
3. DI (Dependency Injection): Полегшує тестування та повторне використання коду.
4. Роутінг: Інтегрований модуль для навігації між сторінками.

В таблиці 3.2 продемонстровано переваги та недоліки вище зазначених фреймворків розробки для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників.

Таблиця 3.2 – Порівняльна характеристика фреймворків розробки

	Vaadin	React	Angular
Мова розробки	Java	JavaScript	TypeScript
Компонентний підхід	Так	Так	Так
Продуктивність	Висока	Висока	Висока
Повнота інструментарію	Висока (все включено)	Середня (потрібні додаткові бібліотеки)	Висока (все включено)
Легкість навчання	Середня (Java розробникам)	Висока (для розробників JavaScript)	Середня (TypeScript + Angular специфіка)
Безпека	Висока (вбудовані механізми)	Залежить від імплементації	Висока (вбудовані механізми)
Гнучкість	Низька (Java-орієнтована)	Висока (можливість інтеграції з іншими технологіями)	Середня (висока інтеграція, але обмежена специфікою)

У випадку WEB-додатку для оцінювання продуктивності роботи співробітників, було обрано Vaadin як фреймворк для клієнтської частини WEB-додатку. Цей вибір обґрунтовується такими факторами як зручність

використання Java, багатий набір готових UI-компонентів, висока безпека, продуктивність та легкість інтеграції з іншими Java-технологіями. Використання Vaadin дозволяє розробникам швидко створювати складні та функціональні інтерфейси, забезпечуючи при цьому високу якість та надійність програмного забезпечення.

Вибір технологій та фреймворків виглядає добре продуманим, спрямованим на уникнення конфліктів несумісності та максимальної ефективності розробки. Використання мови програмування Java, підтримка системи автоматичного збирання проєкту Maven, і фреймворк Vaadin для швидкого розвитку клієнтської частини, спільно з Spring Security для забезпечення безпеки, Spring Data для взаємодії з базою даних MySQL, та Spring Boot для об’єднання всіх модулів та запуску застосунку на IntelliJ IDEA, обіцяють ефективну та добре структуровану розробку WEB-додатку.

3.2 Обгрунтування вибору мови програмування для управління організованою базою даних

OQL (Object Query Language) і SQL (Structured Query Language) − це дві мови запитів, які використовуються для управління базами даних [29]. Обидві мови мають свої особливості і використовуються в різних контекстах.

OQL є мовою запитів, спрямованою на роботу з об'єктно-орієнтованими базами даних. Вона дозволяє виконувати різні операції з об'єктами, такі як отримання, додавання, видалення та оновлення об'єктів у колекціях. OQL визначає синтаксис та правила для складання запитів до бази даних. Вона надає можливість виразно вибирати потрібні дані та виконувати операції над ними з використанням різноманітних операторів, таких як SELECT, INSERT, DELETE, UPDATE, CREATE OBJECT, DROP OBJECT і багатьох інших. OQL є широко використовуваною мовою в об'єктно-орієнтованих базах даних і має багато переваг, таких як незалежність від конкретних систем управління

базами даних, можливість інтерактивного взаємодії з базою даних та програмного доступу до неї.

SQL є реляційною мовою запитів, яка базується на роботі з реляційними базами даних. Вона використовується для формулювання запитів до бази даних, звертаючись до таблиць, їх стовпців та рядків. SQL дозволяє виконувати різні операції з даними, такі як вибірка, вставка, видалення та оновлення даних у таблицях. Вона надає можливість виразно вибирати потрібні дані та виконувати операції над ними з використанням різноманітних операторів, таких як SELECT, INSERT, DELETE, UPDATE, CREATE TABLE, DROP TABLE і багатьох інших. SQL інтегрується з мовами програмування та забезпечує можливість виклику операцій, реалізованих у програмах, з запитів SQL.

Отже, OQL і SQL є двома різними мовами запитів, призначеними для операцій з різними типами баз даних. OQL спрямована на роботу з об'єктно-орієнтованими базами даних і забезпечує широкий набір можливостей для взаємодії з об'єктами та їх атрибутами, тоді як SQL призначена для реляційних баз даних і дозволяє виконувати різноманітні операції над таблицями. Обидві мови мають свої переваги і використовуються відповідно до потреб конкретного типу бази даних.

Для роботи з базою даних обрано мову SQL, яка має переваги інтерактивності, незалежності від СУБД і широкого застосування в реляційних базах даних. SQL дозволяє швидко і легко взаємодіяти з даними, формулюючи запити з ключових термінів та інструкцій. Ключові терміни, такі як SELECT, FROM, WHERE, вказують умови вибірки, джерела даних та умови фільтрації.

Для опису предметної області необхідні такі сутності: працівник, оцінювання, запитання, посада, відповідь. Щоб визначити ступінь універсального відношення потрібно перерахувати всі атрибути, що описують ці сутності. Отже, універсальне відношення для цієї бази даних буде мати такий вигляд:

R (юзернейм працівника, id-працівника, пароль працівника, ім’я працівника, прізвище користувача, id-посади, id-оцінювання, дата початку оцінювання, дата завершення оцінювання, назва оцінювання, id-запитання, назва запитання, опис запитання, тип запитання, відповідь, назва посади, id-відповіді, назва відповіді).

Ступінь універсального відношення – 18.

ER-діаграма є візуальним відтворенням, що використовується для моделювання структури даних у базах даних. Вона показує елементи, взаємодії між елементами і властивості кожного елемента. Основна мета ER-діаграми − зображення структури даних та їх взаємодій у зрозумілій формі. Вона допомагає розробникам і аналітикам уявити, описати і зрозуміти взаємодії між різними компонентами бази даних [14].

Використання ER-діаграми має декілька переваг.

По-перше, вона дозволяє виявити елементи та їх властивості, ідентифікувати взаємодії між елементами і відобразити цю інформацію в зручній формі. Це сприяє кращому розумінню структури даних і полегшує комунікацію між розробниками та зацікавленими сторонами.

По-друге, ER-діаграми допомагають визначити правильний спосіб збереження даних у базі даних, враховуючи їхні залежності та взаємодії. Вони можуть використовуватись як основа для створення схеми бази даних і планування структури таблиць.

ER-діаграма бази даних зображено на рис. 3.2.

Реляційна модель даних − це модель даних, де текстова або числова інформація, відображується за допомогою таблиць. Фрагмент реляційної моделі даних предметної галузі наведено на рис. 3.3.

Вона використовує математичне поняття відношення, яке можна представити у формі таблиці, де кожен рядок є кортежем, а кожен стовпець – атрибутом, визначеним на деякому домені. Реляційна модель даних має три складові частини: структурну, маніпуляційну і цілісну.

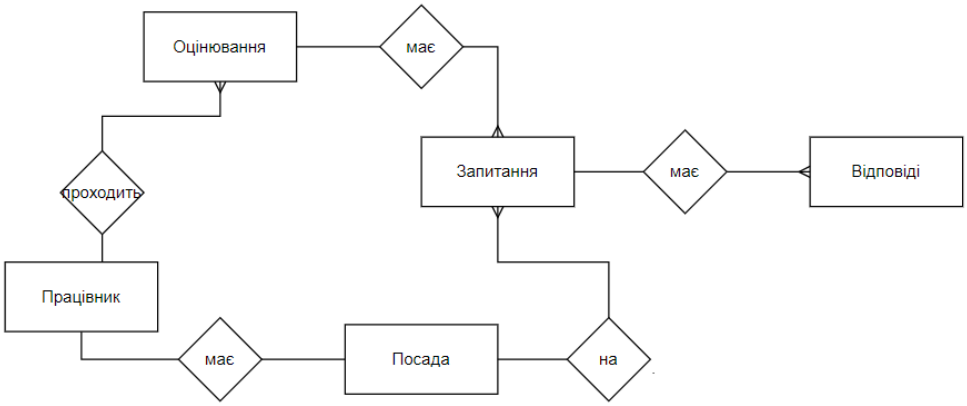

Рисунок 3.2 – ER-діаграма бази даних

	EMPLOYEE_ID	USER_ID	FIRST_NAME	LAST_NAME	POSITION_ID	MANAGER_ID	EMAIL
1	1	2	Anastasiia	Afrosimova	1	6	aa@gmail.com
2	2	3	Simple	Jun	1	6	junior@gmail.com

Рисунок 3.3 – Фрагмент реляційної моделі бази даних

Структурна складова стверджує, що головною структурою даних є нормалізоване n-арне відношення. Вона потребує наявності первинного ідентифікатора, який чітко ідентифікує кожен рядок у таблиці, а також запобігає наявності копій рядків у відношенні.

Маніпулятивна складова моделі включає реляційну алгебру і реляційне числення. Реляційна алгебра надає набір операцій для маніпулювання відношеннями, таких як об'єднання, перетин, різниця, проекція, відбір, приєднання і т.д. Реляційне числення використовує логічні умови для опису бажаних вихідних відношень.

Цілісна складова моделі встановлює вимоги щодо цілісності сутностей і посилань. Цілісність сутностей означає, що значення первинного ідентифікатора не може бути порожнім або невизначеним. Цілісність посилань потребує наявності кортежа з відповідним значенням первинного ідентифікатора для кожного значення зовнішнього ключа, яке посилається на інше відношення.

Реляційна модель інформаційної структури забезпечує організований та ефективний спосіб управління даними у базах даних. Вона дозволяє використовувати реляційну алгебру та числення для виконання різноманітних операцій з даними, забезпечуючи цілісність даних та спрощуючи виконання запитів та аналітичних операцій. Реляційна модель має свої переваги та недоліки. Переваги включають:

1. Простота та зрозумілість для користувача. Основним засобом організації інформації є "таблиця";
2. Незалежність даних від способу їх фізичного збереження. Представлення даних не залежить від методів зберігання на диску чи в пам'яті;
3. Стандартизація мови запитів. Реляційна модель використовує реляційну алгебру та числення як математичну основу для мови запитів;
4. Підтримка цілісності даних. Реляційна модель визначає вимоги цілісності сутностей та посилань, що забезпечує унікальність та консистентність даних.

Недоліки включають:

1. Низька продуктивність для опрацювання великих обсягів даних. Модель даних на основі взаємозв'язків потребує значних зусиль та ресурсів для виконання операцій над таблицями, особливо у випадку, якщо таблиці мають велику кількість рядків та стовпців;
2. Потреба у нормалізації бази даних. Модель даних на основі взаємозв'язків потребує нормалізації бази даних для уникнення аномалій та дублювання даних. Нормалізація може призвести до збільшення кількості таблиць та складності запитів.

Отже, розглянемо дві реляційні системи управління базами даних (СУБД) такі як SQL Server та MySQL.

SQL Server — це платформа даних, розроблена Microsoft, що забезпечує середовище для створення та управління базами даних [30]. Основна перевага

SQL Server полягає в роботі з реляційною моделлю даних. Крім того, SQL Server підтримує стандартну мову запитів SQL. Однією з великих переваг SQL Server є його здатність масштабуватися. Він може легко обробляти як невеликі, так і дуже великі обсяги даних, а також підтримує високонавантажені додатки. Забезпечення безпеки є ще однією важливою характеристикою SQL Server. Він надає широкий спектр засобів для захисту даних, включаючи автентифікацію та авторизацію користувачів, шифрування даних та контроль доступу. SQL Server також інтегрується з іншими продуктами Microsoft, що робить його ще більш потужним та зручним у використанні. Його можна поєднувати з .NET Framework, Visual Studio, Azure та багатьма іншими інструментами.

Недоліками SQL Server є комерційне ліцензування, що може зробити використання деяких його версій дороговартісним. Крім того, він може бути вимогливим до обладнання та ресурсів, особливо при обробці великих обсягів даних.

MySQL є реляційною системою управління базами даних (СУБД), призначеною для зберігання, керування та доступу до великих обсягів даних [31]. Основою MySQL є мова структурованих запитів SQL, яка дозволяє ефективно взаємодіяти з базою даних та виконувати різноманітні операції, такі як створення, читання, оновлення та видалення записів.

Серед основних переваг MySQL виділяється її висока продуктивність, яка забезпечує швидку обробку запитів і значну пропускну здатність. Крім того, MySQL підтримує розподілені операції, що дозволяє масштабувати систему та підвищувати продуктивність шляхом розподілу навантаження між кількома серверами. Ще однією важливою перевагою MySQL є її відкритий вихідний код і активна спільнота розробників, що сприяє постійному вдосконаленню системи відповідно до потреб користувачів.

Однак MySQL має й недоліки. Серед них — відсутність деяких розширених функцій, які доступні у комерційних СУБД, а також обмежені можливості масштабування в порівнянні з іншими системами.

Порівняльна характеристику СУБД баз даних наведено у табл. 3.3

Таблиця 3.3 – Порівняння характеристик СУБД баз даних

Характеристика	SQL Server	MySQL
Розробник	Microsoft	Oracle Corporation
Ліцензія	Комерційна, пропрієтарна	Відкрита
Операційна система	Підтримує Windows	Підтримує різні операційні системи
Мови програмування	T-SQL, .NET Framework	Python, Java та ін.
Підтримка стандартів SQL	Висока	Має деякі відхилення
Масштабованість	Добре масштабований	Менш масштабований
Інструменти управління	SQL Server Management Studio	MySQL Workbench
Підтримка реплікації	Має потужні засоби	Має базові засоби
Надійність	Висока	Висока
Реляційність	Підтримка реляційної моделі даних	Підтримка реляційної моделі даних
Міграція даних	Має вбудовані засоби для міграції даних, такі як SQL Server Integration Services	Має різні інструменти та підходи для міграції даних, такі як MySQL Workbench

Після аналізу баз даних SQL Server та MySQL було обрано MySQL. MySQL надає ширший набір інструментів для управління базами даних, включаючи високий рівень безпеки та можливості реплікації даних. Крім того, враховуючи використання мови програмування Java, MySQL забезпечує глибшу інтеграцію, сприяючи швидкому та ефективному розробленню веб-додатків.

3.3 Обґрунтування вибору середовища розробки

Для створення Java-додатків є кілька варіантів: Eclipse, NetBeans, IntelliJ IDEA та інші. Всі ці середовища розробки доступні з відкритим вихідним кодом.

Переваги використання цих середовищ розробки:

- безкоштовне розповсюдження;
- можливість додавання нових функцій та отримання нових можливостей;
- можливість покращення IDE. Чим більше людей хочуть внести покращення, тим легше тестувати нові функції, що зменшує кількість помилок у роботі.

На сьогодні найпопулярнішими є Eclipse та IntelliJ IDEA. Вони мають приблизно однакові функціональні можливості.

Стандартні можливості вищезазначених IDE:

- підсвічування синтаксису;
- форматування коду;
- компіляція коду;
- система підказок;
- автодоповнення;
- інтеграція з бібліотеками та програмними каркасами;
- автогенерація коду;
- налагоджувач коду;
- перевірка помилок.

Eclipse – це безкоштовне модульне інтегроване середовище для розробки програмного забезпечення, яке розробляється та підтримується Eclipse Foundation [32]. До нього входять проекти, такі як платформа Eclipse, набір інструментів для програмування на мові Java, системи контролю версій,

конструктори GUI тощо. Написане переважно на Java, Eclipse може використовуватись для розробки застосунків на Java, а також за допомогою різних плагінів на інших мовах програмування, включаючи C++, C#, COBOL, Fortran, PHP, Python. Середовища розробки включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

IntelliJ IDEA – це комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та інші) від компанії JetBrains.

Community версія IntelliJ IDEA підтримує інструменти (у вигляді плагінів) для проведення тестування TestNG та JUnit, системи контролю версій Mercurial і Git, засоби складання Maven, Ant, мови програмування Java, Clojure, Groovy і Dart. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, GitHub, YouTrack, Lighthouse.

Комерційна версія "Ultimate Edition" відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML-діаграм, підрахунком покриття коду, можливістю роботи з фреймворками (Rails, Spring, Play Framework і Hibernate), засобами інтеграції з Microsoft Team Foundation Server і Rational ClearCase.

Якщо необхідно отримати красиві іконки, платформу для створення настільних програм або IDE для C++, то Eclipse ймовірно краще. Якщо потрібна система, яка дозволяє швидко та зручно вести розробку, зосереджуючись на проблемі, то IDEA – це те, що потрібно.

В таблиці 3.4 продемонстровано переваги та недоліки вище зазначених середовищ розробки для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників.

Таблиця 3.4 – Порівняльна характеристика середовищ розробки

	IntelliJ IDEA	Eclipse
Автодоповнення і перевірка коду	+	+
Форматування коду	+	+
Швидкодія	+	+
Можливість відкриття декілька проектів в одному вікні	-	+
Швидкий візуальний редактор форм	+	-
Розбиття коду на графічні блоки	-	+

За результатами порівняння популярних інтегрованих середовищ розробки для мови програмування Java, наведені у таблиці, можна зробити висновок, що IntelliJ IDEA є найкращим вибором серед усіх розглянутих варіантів для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників.

3.4 Розробка серверної частини

Для розробки програмного модуля використано такі технології:

Vaadin Framework – це популярний фреймворк для створення веб-додатків на Java. Він надає інструменти для побудови сучасних веб-інтерфейсів, що виглядають і працюють як нативні десктопні додатки. Однією з головних особливостей Vaadin є можливість писати весь інтерфейс користувача на Java без необхідності використовувати HTML, CSS або JavaScript. Це робить розробку простішою та більш узгодженою, особливо для тих, хто добре володіє Java.

Spring Framework – це потужний фреймворк для розробки Java-додатків, який надає всебічну підтримку для створення корпоративних додатків. Spring відомий своєю гнучкістю та модульністю, що дозволяє розробникам

використовувати лише ті частини фреймворку, які потрібні для конкретного проекту.

Java Persistence API (JPA) – це стандартна специфікація Java для збереження, управління і доступу до даних між об'єктами Java і реляційними базами даних. Вона забезпечує простий та ефективний спосіб об'єктно-реляційного відображення (ORM), що дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтовані концепції.

Основні характеристики JPA:

1. Об'єктно-реляційне відображення (ORM): JPA дозволяє розробникам визначати, як об'єкти Java відображаються на таблиці бази даних, і навпаки. Це спрощує роботу з базами даних, оскільки дозволяє маніпулювати даними, використовуючи об'єктно-орієнтований підхід.

2. Анотації та XML-конфігурації: JPA підтримує конфігурацію за допомогою анотацій, що дозволяє визначати відображення об'єктів на таблиці прямо в коді. Також можлива конфігурація через XML-файли, що забезпечує гнучкість у налаштуваннях.

Liquibase – це інструмент для управління схемами баз даних та версіями змін, який допомагає автоматизувати процеси міграції баз даних. Він дозволяє розробникам відстежувати, версіонувати та розгортати зміни у базі даних в керований і передбачуваний спосіб.

Основні характеристики Liquibase:

1. Контроль версій: Liquibase дозволяє відстежувати всі зміни у схемі бази даних через механізм контролю версій. Кожна зміна зберігається у вигляді change set, що дозволяє легко відкотити або застосувати конкретні зміни.

2. Підтримка різних форматів файлів: Liquibase підтримує кілька форматів для опису змін у базі даних, включаючи XML, YAML, JSON та SQL. Це дозволяє розробникам вибирати найбільш зручний для них спосіб опису змін.

3. Міграції баз даних: Інструмент дозволяє автоматизувати процес міграції баз даних між різними середовищами (розробка, тестування, продуктивне середовище), що зменшує ризик помилок та знижує час на налаштування.

Розглянемо деякі приклади реалізації серверної частини:

Розглянемо приклад, коли користувач заходить на сторінку авторизації.

```
@Route("login")
@PageTitle("Login")
public class LoginView extends VerticalLayout {

    public LoginView() {
        TextField username = new TextField("Username");
        PasswordField password = new PasswordField("Password");
        Button loginButton = new Button("Login", event -> {
            // логіка для авторизації
        });

        add(
                new H1("Login"),
                username,
                password,
                new Label("Welcome! Please login to continue."),
                loginButton
        );
    }
}
```

У цій функції LoginView() в якості аргументу використовується об'єкт VerticalLayout, який забезпечує вертикальне розташування компонентів на сторінці. Створюються об'єкти TextField та PasswordField для введення імені користувача та пароля відповідно, а також об'єкт Button для авторизації. Цей фрагмент коду демонструє створення форми для входу в систему з використанням Vaadin.

Приклад, коли користувач бере участь в опитуванні.

```
@Route("survey")
@PageTitle("Survey")
public class SurveyView extends VerticalLayout {

    public SurveyView() {
        Div surveyContent = new Div();
        surveyContent.add(new Div(new Text("Survey content goes here")));

        Button submitButton = new Button("Submit", event -> {
            // логіка для надсилання оцінювання
        });

        add(surveyContent, submitButton);
    }
}
```

У цій функції SurveyView() створюється об'єкт Div для відображення контенту оцінювання та об'єкт Button для надсилання результатів. Цей фрагмент коду демонструє створення сторінки оцінювання з використанням компонентів Vaadin.

Розглянемо приклад, коли користувач керує оцінюваннями.

```
@Route(value = "manager-survey")
@PageTitle("Manager Survey")
public class ManagerSurveyView extends VerticalLayout {

    private Grid<Survey> grid = new Grid<>(Survey.class);

    public ManagerSurveyView() {
        addClassName("manager-survey-view");
        setSizeFull();

        configureGrid();
```

```
        add(grid);
    }

    private void configureGrid() {
        grid.setColumns("name", "description", "createdDate");
        grid.getColumns().forEach(col -> col.setAutoWidth(true));
    }
}
```

У цій функції configureGrid() налаштовується сітка Grid для відображення списку опитувань. Сітка налаштовується таким чином, щоб колонки автоматично підлаштовувалися під вміст. Цей фрагмент коду демонструє використання сітки для відображення даних опитувань.

3.5 Тестування роботи програми та аналіз результатів

Тестування є важливою складовою розробки WEB-додатку для оцінювання продуктивності роботи співробітників, що дозволяє виявити та виправити помилки, перевірити відповідність функціоналу вимогам та забезпечити належну роботу програми.

Після запуску WEB-додатку з’являється початкова сторінка (рис. 3.4). Вона містить функції увійти як «Співробітник», «Керівник», «Адміністратор». Щоб розпочати роботу з WEB-додатком як співробітик, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Співробітник».

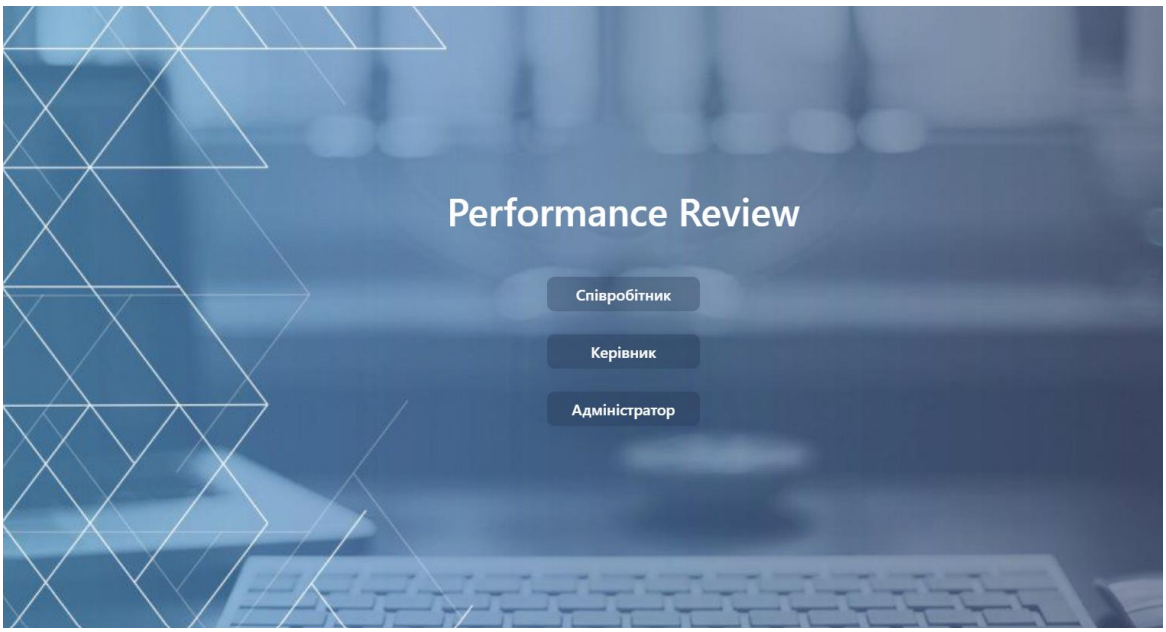

Рис 3.4 – Головна сторінка WEB-додатку

Після цього на сторінці авторизації для співробітника, ввести дані та натиснути кнопку Login (рис. 3.5).

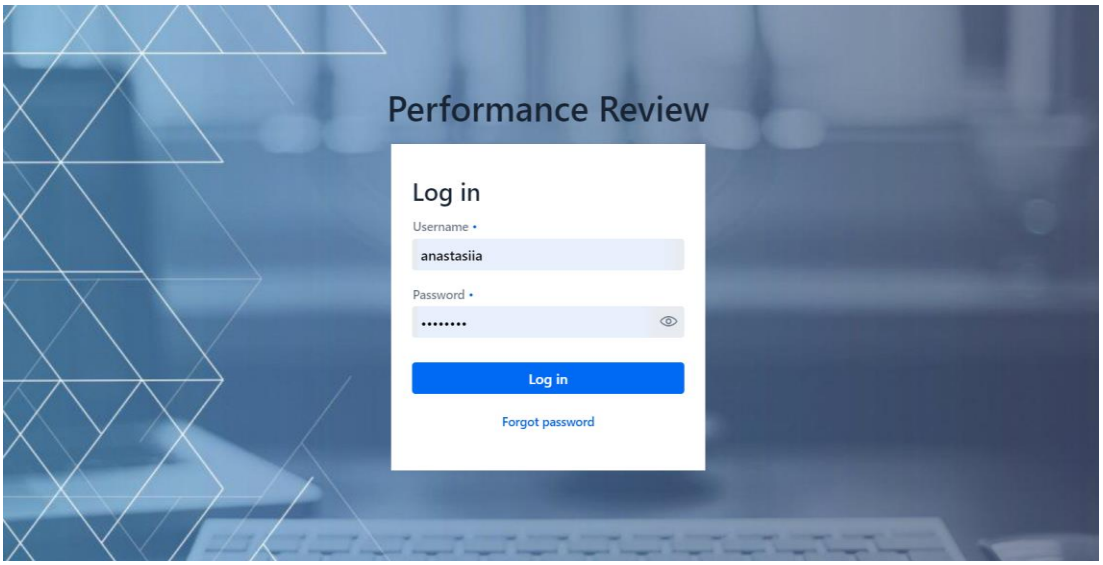

Рис 3.5 – Сторінка входу у ролі «Співробітника»

Співробітник потрапляє на основну сторінку WEB-додатку (рис. 3.6). Він потрапляє на сторінку меню «Співробітник». На цій сторінці співробітник має можливість переглядати усі оцінювання, включаючи назву, дату початку та

завершення, керівника та поле з відміткою про проходження оцінювання працівником, керівником. Після перегляду співробітник може обрати оцінювання, яке йому необхідно пройти.

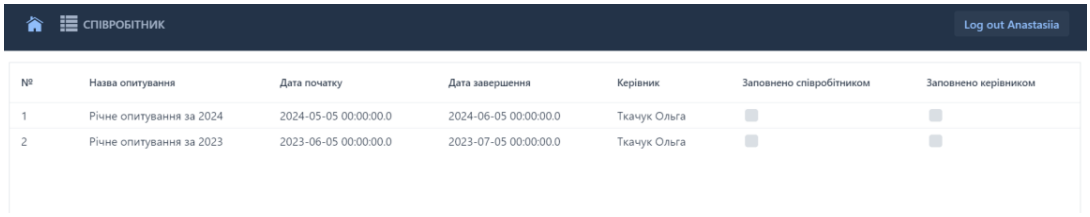

Рис 3.6 – Головна сторінка системи у ролі «Співробітника»

Коли співробітник відкрив форму з необхідним оцінюванням (рис. 3.7). Він може давати відповіді на питання у письмовій формі або за допомогою чекбоксів, випадаючого списку чи радіокнопок. При проходженні оцінювання у співробітника є можливість зберегти свої результати натиснувши на кнопку «Зберегти» або по завершенню проходження натиснути кнопку «Готово».

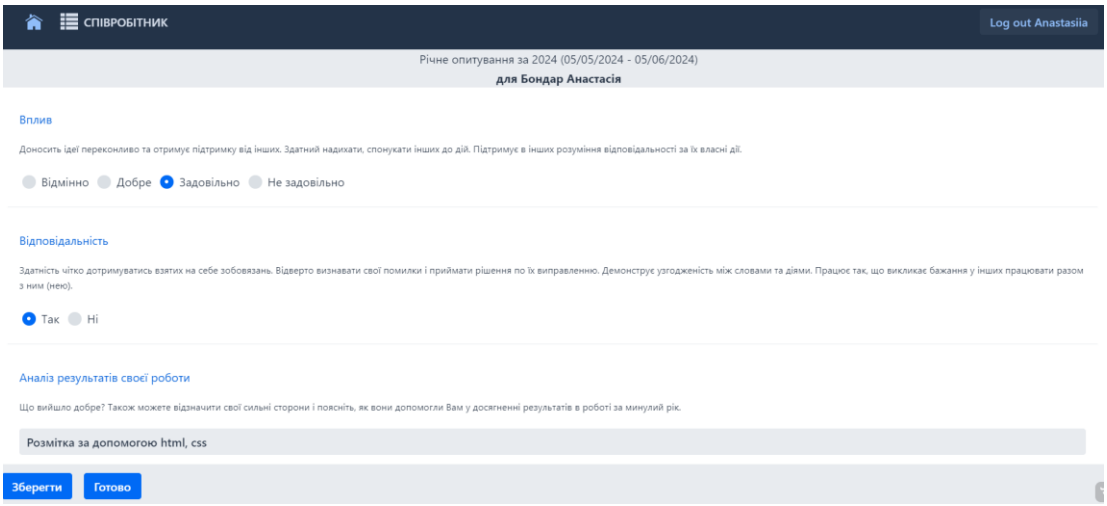

Рис 3.7 – Проходження оцінювання у ролі «Співробітника»

Після цього співробітник перейде назад до головної сторінки, де буде видно що оцінювання пройдено (рис 3.8). Отримати свої результати з оцінювання можна буде тільки після проходження його керівником.

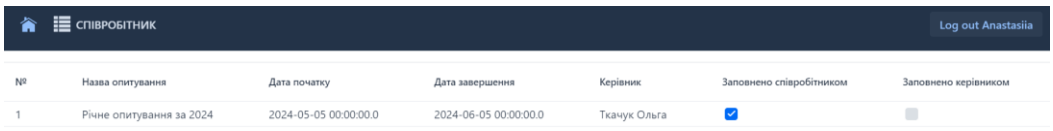

Рис 3.8 – Зміни на головній сторінці після проходження оцінювання у ролі «Співробітника»

Щоб розпочати роботу з WEB-додатком як керівник, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Керівник».

Після цього на сторінці авторизації для керівника, ввести дані та натиснути кнопку Login (рис. 3.9).

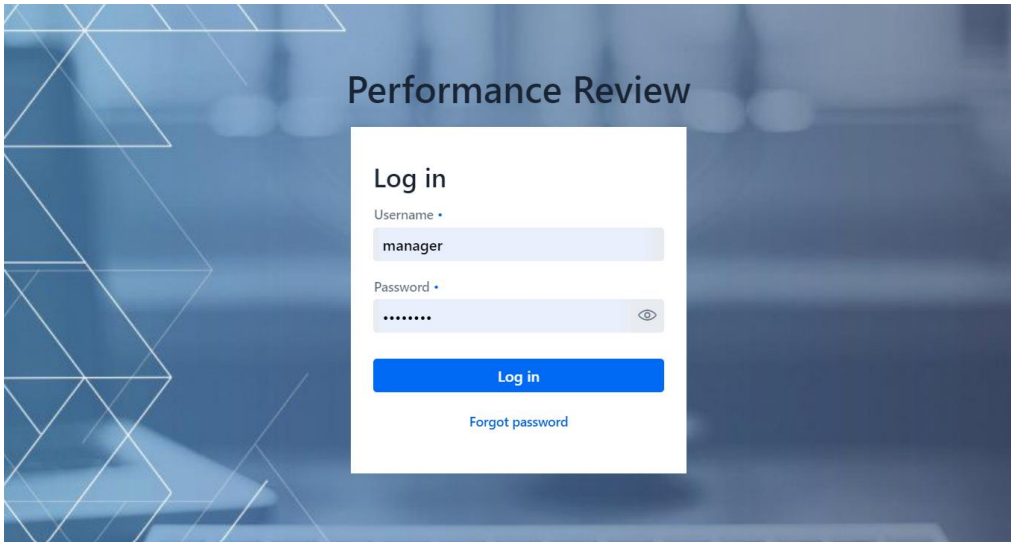

Рис 3.9 – Сторінка входу у ролі «Керівника»

Керівник потрапляє на основну сторінку WEB-додатку. Так як керівник, є також і співробітником, тому він потрапляє на головну сторінку меню «Співробітник». На цій сторінці він, як працівник, має можливість переглядати усі оцінювання, включаючи назву, дату початку та завершення, керівника та поле з відміткою про проходження оцінювання працівником, керівником.

Також додається нова сторінка меню «Керівник» (рис. 3.10), в якій буде видно усі оцінювання підлеглих керівника.

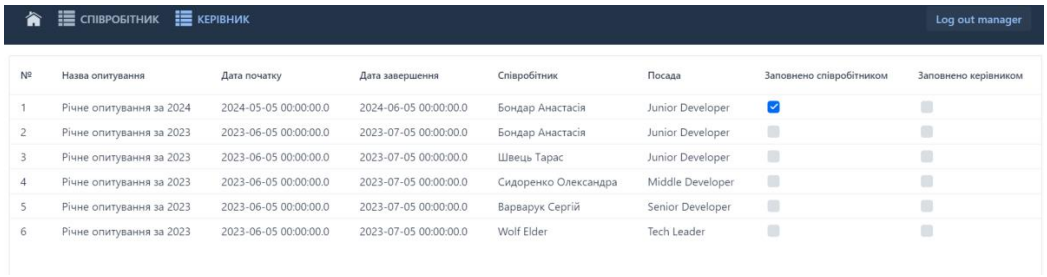

Рис 3.10 – Сторінка меню «Керівник»

Керівник має змогу пройти оцінювання як в ролі співробітника так і в ролі керівника. Коли керівник відкрив форму з необхідним оцінюванням (рис. 3.11). Він може давати відповіді на питання у письмовій формі або за допомогою чекбоксів, випадаючого списку чи радіокнопок. При проходженні оцінювання у керівника є можливість зберегти свої результати натиснувши на кнопку «Зберегти» або по завершенню проходження натиснути кнопку «Готово».

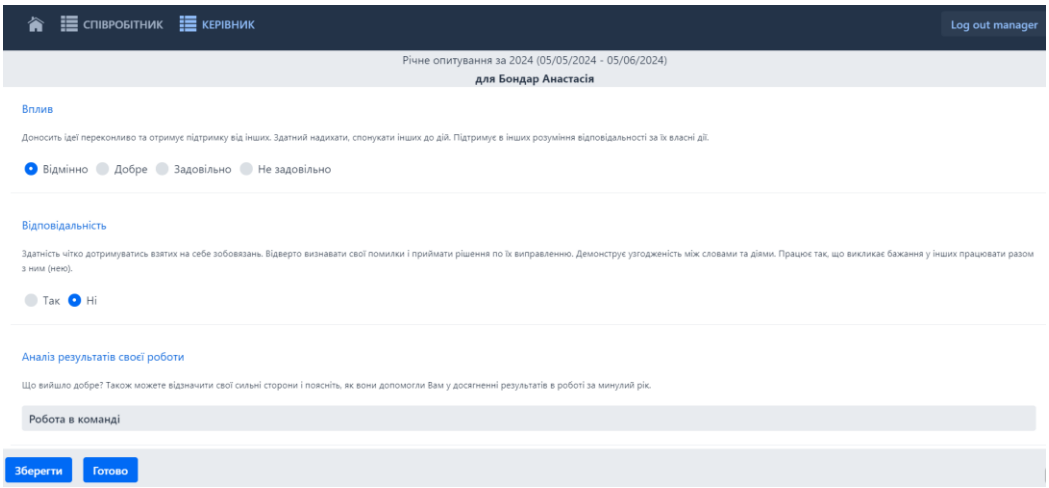

Рис 3.11 – Проходження оцінювання у ролі «Керівника»

Після завершення проходження, якщо оцінювання було вже пройдене підлеглим відкриється форма з результатами оцінювання (рис. 3.12). В якій буде видно наскільки подібно відповіли керівник та співробітник. У тому разі, якщо поле питання світиться червоним кольором означає, що керівник відповів гірше, ніж співробітник, якщо зеленим – краще, якщо не підсвічується – однаково. В кінці результатів додасться поле для відгуку (рис. 3.13), де керівник і співробітник зможуть надати зворотній зв’язок щодо пройденого оцінювання.

Рис 3.12 – Результати оцінювання

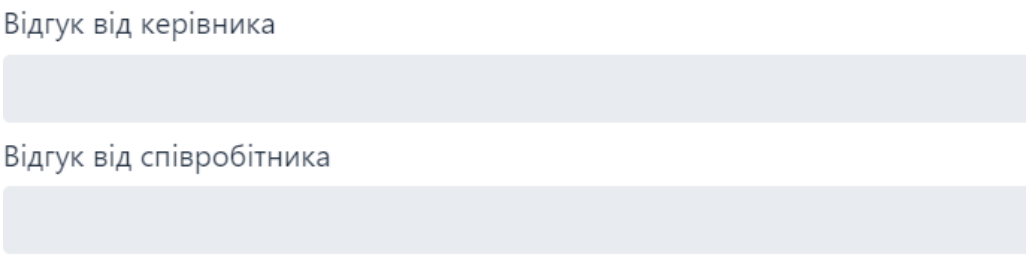

Рис 3.13 – Поле для створення відгуку

Щоб розпочати роботу з WEB-додатком як адміністратор, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Адміністратор».

Після цього на сторінці авторизації для адміністратора, ввести дані та натиснути кнопку Login (рис. 3.14).

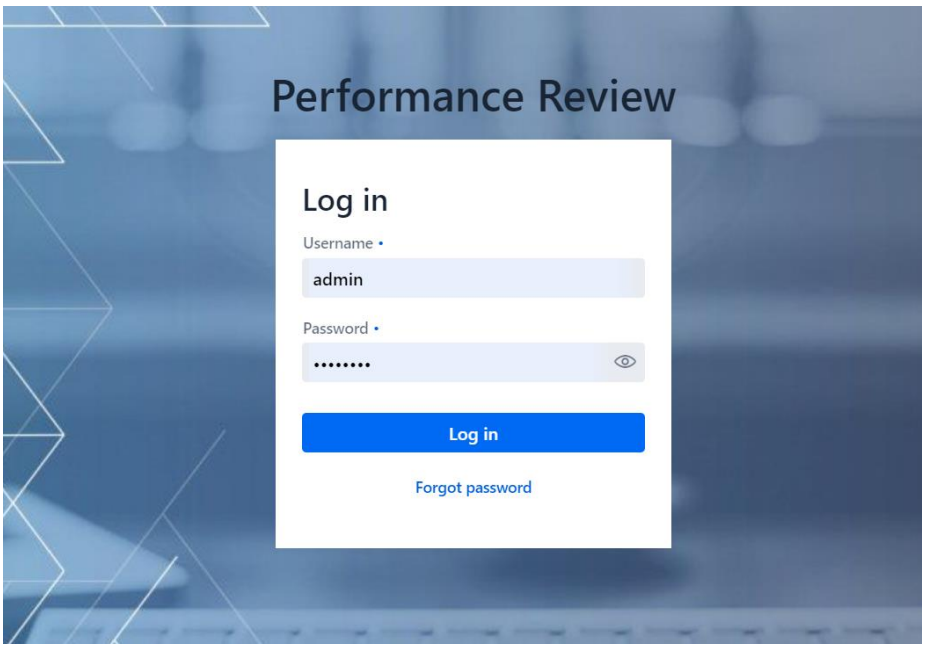

Рис 3.14 – Сторінка входу у ролі «Адміністратора»

Керівник потрапляє на основну сторінку WEB-додатку (рис. 3.15). На лівій частині головної сторінки знаходиться меню. На правій частині знаходиться інформація з вибраного меню. Після авторизації, адміністратор потрапляє на сторінку меню «Користувачі». На цій сторінці адміністратор має можливість переглядати усіх створених користувачів. При натисненні на користувача відкриється форма редагування, видалення користувача з системи (рис. 3.16). Також можна зареєструвати нового користувача натиснувши на кнопку «Зареєструвати співробітника». Відкриється форма, де можна буде вказати логін та пароль створеного користувача.

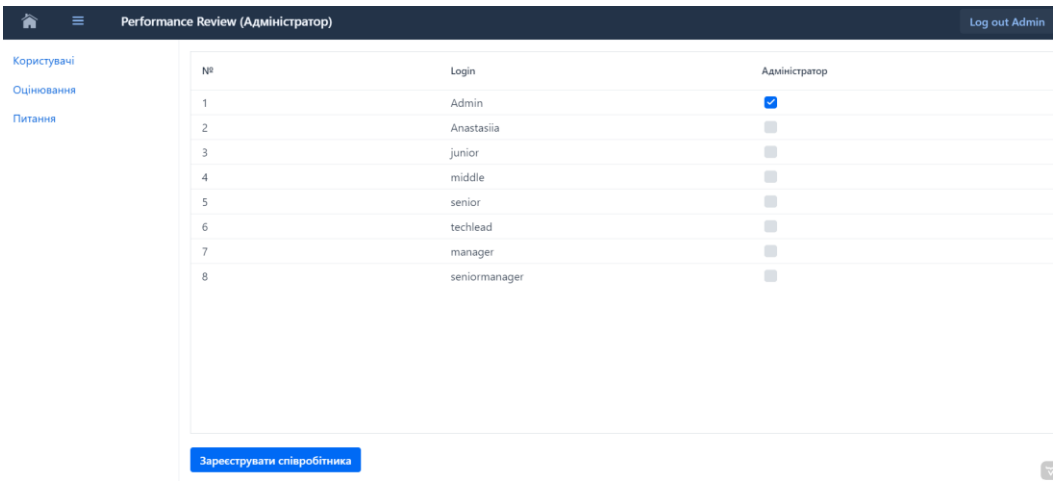

Рис 3.15 – Головна сторінка системи з оцінюваннями у ролі «Адміністратора»

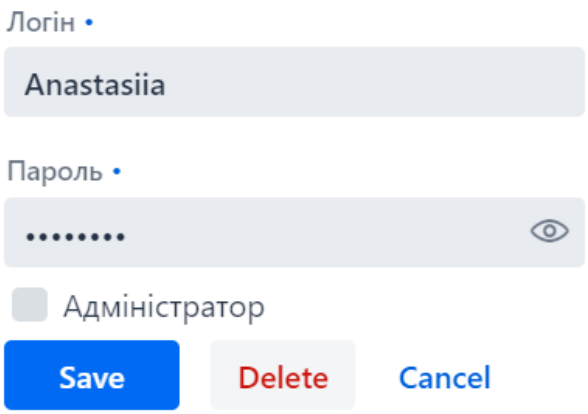

Рис 3.16 – Форма створення, редагування та видалення користувача з системи

Наступна сторінка меню - це сторінка з оцінюваннями (рис. 3.17). На цій сторінці адміністратор має можливість переглядати усі створені оцінювання. При натисненні на оцінювання відкриється форма редагування, видалення оцінювання з системи (рис. 3.18). Також можна створити нове оцінювання натиснувши на кнопку «Створити оцінювання». Відкриється форма, де можна буде вказати назву, дату початку та завершення оцінювання.

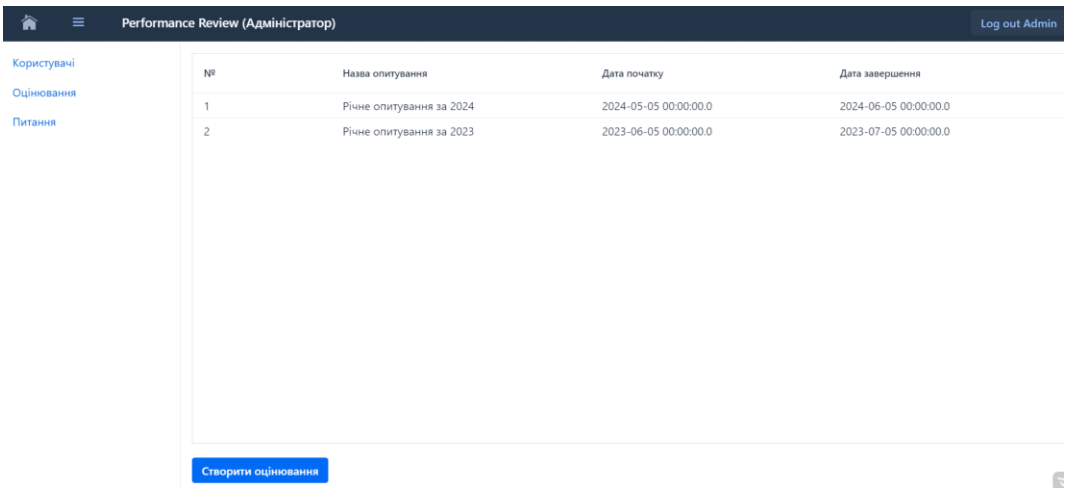

Рис 3.17 – Сторінка меню з оцінюваннями у ролі «Адміністратора»

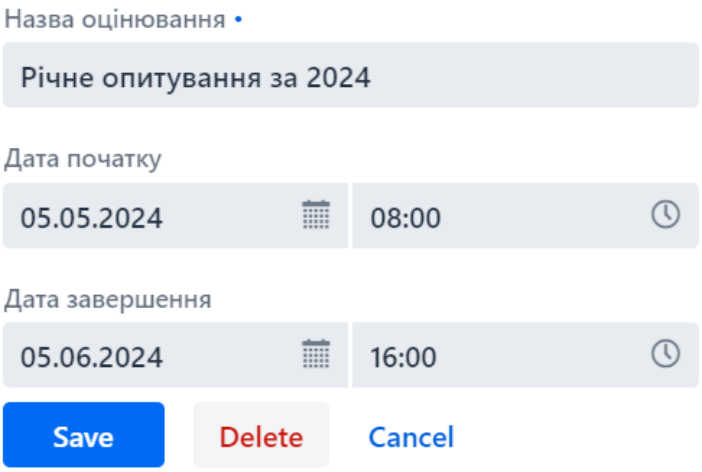

Рис 3.18 – Форма створення, редагування та видалення оцінювання з системи

Наступна сторінка меню - це сторінка з питаннями (рис. 3.19). На цій сторінці адміністратор має можливість переглядати усі створені питання. При натисненні на питання відкриється форма редагування, видалення питання з системи (рис. 3.20). Також можна створити нове питання натиснувши на кнопку «Створити питання». Відкриється форма, де можна буде вказати назву питання, опис, тип відповіді та кількість варіантів відповідей.

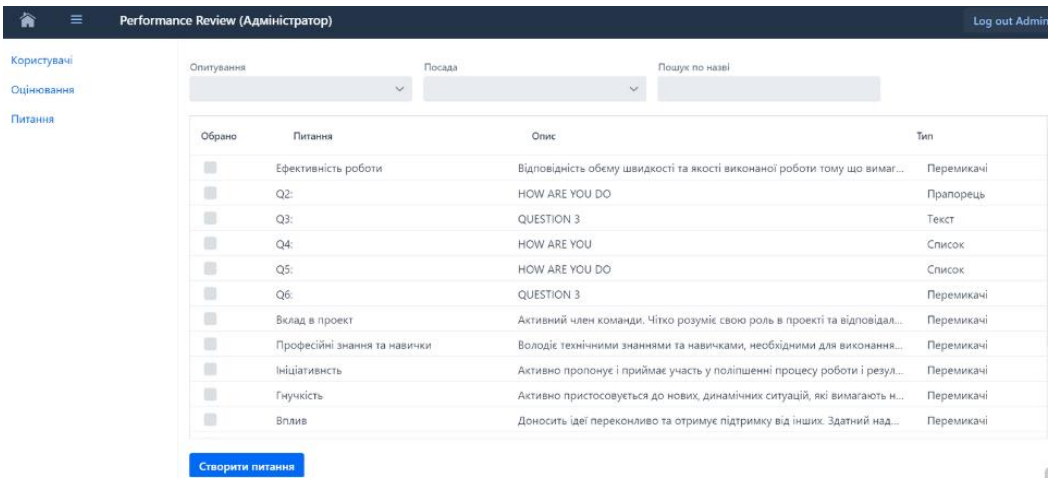

Рис 3.19 – Сторінка меню з питаннями до оцінювання у ролі «Адміністратора»

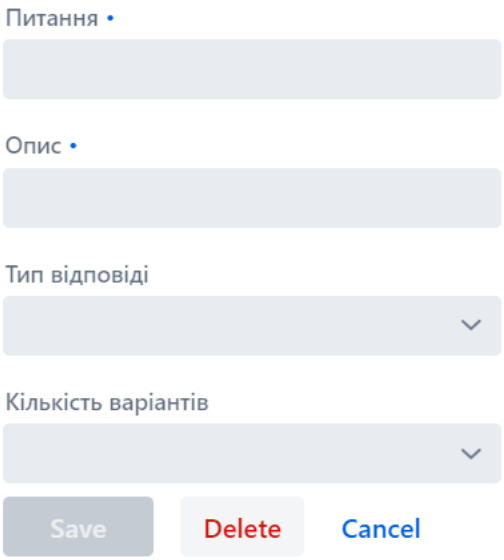

Рис 3.20 – Форма створення, редагування та видалення питань з системи

Під час тестування WEB-додатку для оцінювання продуктивності роботи співробітників було проведено реєстрацію для десятьох користувачів, що займають різні посади та відповідають різним ролям (співробітник, керівник, адміністратор). Також було створено десять оцінювань, та додані відповідно питання до кожного з них. Було ретельно перевірено функціональність

реєстрації під різними ролями, можливість перегляду усіх існуючих оцінювань, правильність обробки відповідей оцінювання, можливість перегляду результатів та коректність заповнення відгуку на пройдене оцінювання. Окрім цього, тестувалася можливість додавання, редагування та видалення усіх можливих користувачів, оцінювань та питань з боку «Адміністратора», а також правильність переадресації посилань та роботи кнопок. Проведене тестування показало, що функціонал WEB-додатку працює належним чином та повністю відповідає визначеним вимогам.

В табл. 3.5 подано порівняння результатів тестування розробленого WEB- додатку для оцінювання продуктивності роботи співробітників з програмами-аналогами.

Таблиця 3.5 – Результати тестування розробленого WEB-додатку порівняно з аналогами

	Розроблений додаток	BambooHR	Kissflow HR Cloud	Survey Monkey
Створення універсальних оцінювань	+	+	+	-
Авторизація	+	+	+	+
Написання відгуків	+	-	-	+
Зручний інтерфейс	+	+	-	+

Отже, проаналізувавши та протестувавши розроблений додаток можна стверджувати, що поставлених цілей досягнуто. За результатами порівняння табл. 3.3 можна побачити, що розроблений програмний засіб має покращений функціонал, принаймні на одну можливість у порівнянні з програмами-аналогами, який забезпечить вищу продуктивність роботи працівника.

Розроблений WEB-додаток для оцінювання продуктивності роботи співробітників має ширший функціонал зарахунок доданої функції «Відгуків».

Висновок до розділу 3

У даному розділі було проведено аналіз та обґрунтування вибору мов програмування, фреймворків і бази даних, а також тестування розробленого WEB-додатку. Ці кроки були важливими для успішної реалізації проекту та досягнення його цілей.

Обґрунтування вибору мов програмування та фреймворків включало ретельний аналіз потреб проекту та порівняння характеристик доступних варіантів. Враховуючи такі фактори, як продуктивність, швидкість розробки, підтримка спільноти та масштабованість, було прийнято рішення використовувати мову програмування Java та для реалізації серверної частини WEB-додатку, а також веб-фреймворки Vaadin, Spring Java Framework. Усе це було вирішено писати у середовищі розробки IntelliJ IDEA.

При обґрунтуванні вибору бази даних було проведено порівняльний аналіз різних варіантів, таких як SQL Server та MySQL. Враховуючи фактори надійності, масштабованості, швидкодії та зручності розробки, було прийнято рішення використовувати MySQL як базу даних для WEB-додатку.

Останнім етапом було тестування розробленого WEB-додатку, в результаті якого було проведено порівняльну характеристику розробленого WEB-додатку з уже існуючими. Було розширено функціональні можливості WEB-додатку для оцінювання продуктивності роботи співробітників, а саме: можливість створення універсальних оцінювань та додання функції відгуків на пройдені оцінювання. Таким чином, мету дослідження було досягнуто.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи було розроблено WEB-додаток для оцінювання продуктивності роботи співробітників

Усі задачі, поставлені для реалізації WEB-додатку для оцінювання продуктивності роботи співробітників, були виконані в повному обсязі, а саме:

- обґрунтовано доцільність розробки WEB-додатку для оцінювання продуктивності роботи співробітників;
- проаналізовано програми-аналоги, їх переваги та недоліки;
- спроектовано WEB-додаток для оцінювання продуктивності роботи співробітників;
- програмно реалізовано WEB-додаток для оцінювання продуктивності роботи співробітників;
- виконано тестування програми та проведено аналіз отриманих результатів.

Проаналізовано існуючі методології реалізації додатку для оцінювання продуктивності роботи співробітників та обрано найбільш прийнятний – 180°, через його об’єктивність та універсальність. На основі проведеного огляду програм-аналогів поставлено задачі та визначено основні вимоги до функціоналу.

Під час проектування було представлено загальну UML-діаграму класів та описано їх методи, на основі якої відбулася подальша розробка. Також було представлено UML-діаграми прецедентів, розгортання та взаємодій, що допомогли у проектуванні та визначили процес взаємодії користувача з системою. Було обрано архітектуру та розроблено схему алгоритму функціонування WEB-додатку. Схема алгоритму допомогла зрозуміти, як покроково має працювати програма. Під час проектування було враховано

потреби користувачів для забезпечення оптимального функціонування додатку.

Також було здійснено фактичну реалізацію спроектованого WEB-додатку. Було обґрунтовано вибір інструментів, а саме: мов програмування, фреймворків та бази даних для реалізації WEB-додатку. Також було проведено тестування розробленого WEB-додатку для перевірки його функціональності, коректності та взаємодії компонентів системи. Окрім того, було проведено порівняльну характеристику розробленого WEB-додатку з програмами-аналогами. Розроблений WEB-додаток успішно пройшов тестування та у повній мірі відповідає заданим вимогам.

Мета роботи, яка полягала в розширенні функціональних можливостей WEB-додатку для оцінювання продуктивності роботи співробітників, досягнута за рахунок того, що у порівнянні з аналогами, введено 2 нові можливості, а саме: можливість створення універсального оцінювання та написання відгуку на пройдене оцінювання. Це сприяло покращенню зворотного зв'язку між співробітниками та керівництвом, а також забезпечило більш детальну і точну оцінку професійних навичок та компетенцій працівників.

Отже, всі поставлені задачі було виконано, а мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Афросімова А.А., Сілагін О.В. «Розробка веб-додатку для оцінки продуктивності роботи співробітників». Матеріали конференції: «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)». [Електронний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20701

2. Цифровізація процесу оцінювання працівників. [Електронний ресурс] – Режим доступу: http://ppeu.stu.cn.ua/article/view/299142.

3. Метод самооцінки на роботі. [Електронний ресурс] – Режим доступу: https://jobs.ua/articles/yak-otsniti-chi-dobre-vi-spravlyatesya-z-robotoyu-14528.

4. Метод оцінки керівників. [Електронний ресурс] – Режим доступу: http://lib-net.com/content/9560_Ocinka_kerivnikiv.html.

5. Оцінка методом 180 [Електронний ресурс] – Режим доступу: https://peopleforce.io/uk/blog/gid-metodi-otsinki-personalu

6. Оцінка методом 360. [Електронний ресурс] – Режим доступу: https://collaborator.biz/blog/360-degree-feedback-how-to-evaluate-staff-and-improve-the-work/

7. BambooHR [Електронний ресурс] – Режим доступу: https://app.bamboohr.com/.

8. Kissflow HR Cloud. [Електронний ресурс] – Режим доступу: https://kissflow.com/hr

9. Survey Monkey. [Електронний ресурс] – Режим доступу: https://www.surveymonkey.com/

10. Архітектура програмного забезпечення. [Електронний ресурс] – Режим доступу: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya

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

12. N-рівнева архітектура програмного забезпечення. [Електронний ресурс] – Режим доступу: https://studfile.net/preview/5219680/page:4/

13. Мікросервісна архітектура програмного забезпечення. [Електронний ресурс] – Режим доступу: https://qalight.ua/baza-znaniy/shho-take-mikroservisna-arhitektura-pz/

14. Clean-архітектура програмного забезпечення. [Електронний ресурс] – Режим доступу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

15. У чому суть UML діаграм. [Електронний ресурс] – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html

16. Діаграми взаємодії. [Електронний ресурс] – Режим доступу: https://moodle.znu.edu.ua/mod/resource/view.php?id=265912&forceview=1

17. Діаграми розгортання. [Електронний ресурс] – Режим доступу: https://www.guru99.com/uk/deployment-diagram-uml-example.html

18. Основні принципи ООП. [Електронний ресурс] – Режим доступу: https://training.epam.ua/ua/blog/275

19. Алгоритми їх основні функції. [Електронний ресурс] – Режим доступу: https://cloud.itstep.org/blog_3/building-and-understanding-algorithms-a-step-by-step-guide-for-beginners

20. Переваги та недоліки C++. [Електронний ресурс] – Режим доступу: https://informatikpm11.blogspot.com/p/c.html

21. Python. Найкраще застосування. [Електронний ресурс] – Режим доступу: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python

22. Java. Реалізація у веб-розробці [Електронний ресурс] – Режим доступу: https://goit.global/ua/articles/shcho-take-java-i-de-vona-vykorystovuietsia/

23. Spring Java Framework. [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework

24. Documentation Spring Security. [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-security

25. Spring Boot переваги і недоліки. [Електронний ресурс] – Режим доступу: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-java-spring-boot

26. Vaadin inctrodurction. [Електронний ресурс] – Режим доступу: https://www.baeldung.com/vaadin

27. React. [Електронний ресурс] – Режим доступу: https://ru.legacy.reactjs.org/

28. Angular. [Електронний ресурс] – Режим доступу: https://foxminded.ua/angular/

29. SQL and OQL. [Електронний ресурс] – Режим доступу: https://cr.openjdk.org/~sundar/8022483/webrev.01/raw_files/new/src/share/classes/com/sun/tools/hat/resources/oqlhelp.html

30. База даних SQL Server. [Електронний ресурс] – Режим доступу: https://support.microsoft.com/uk-ua/topic/

31. База даних MySQL. [Електронний ресурс] – Режим доступу: https://timeweb.cloud/tutorials/mysql/osnovy-mysql

32. Середовище розробки Eclipse. [Електронний ресурс] – Режим доступу: https://foxminded.ua/ru/eclipse-ide/

33. Середовище розробки InteliJ IDEA. [Електронний ресурс] – Режим доступу: https://itpro.kiev.ua/product/jetbrains-intellij-idea/?tab=description

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка WEB-додатку для оцінювання продуктивності роботи співробітників

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93,06% Схожість 6,94%

Аналіз звіту подібності (відмітити потрібне):

✓ 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 виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 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 Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Афросімова А.А.

Керівник роботи Сілагін О.В.

ДОДАТОК Б
ІНСТРУКЦІЯ КОРИСТУВАЧА

Після запуску WEB-додатку з’являється початкова сторінка (рис. Б.1). Вона містить функції увійти як «Співробітник», «Керівник», «Адміністратор». Щоб розпочати роботу з WEB-додатком як співробітик, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Співробітник».

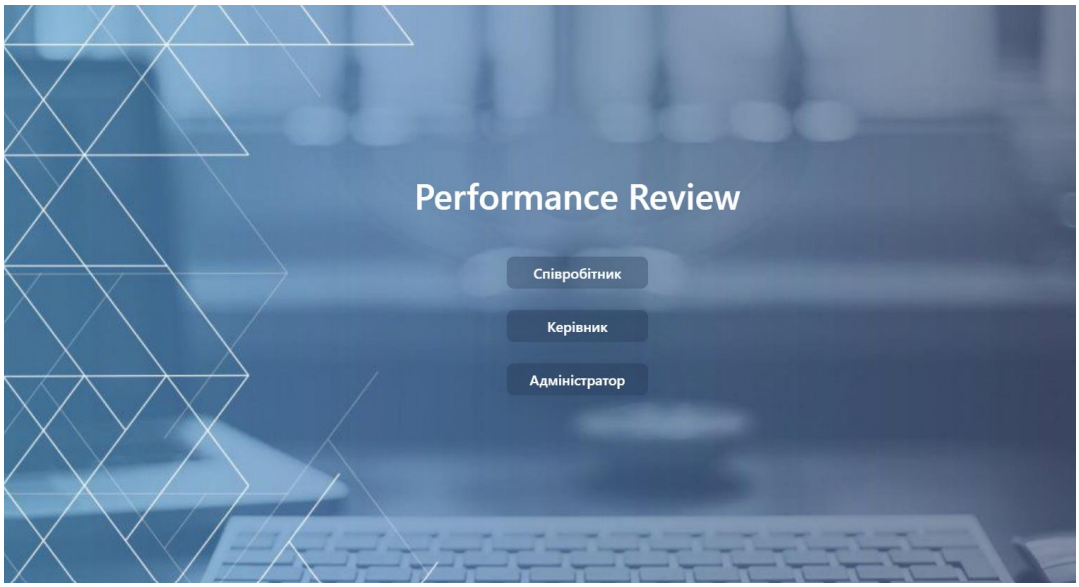

Рис. Б.1 – Головна сторінка WEB-додатку

Після цього на сторінці авторизації для співробітника, ввести дані та натиснути кнопку Login (рис. Б.2).

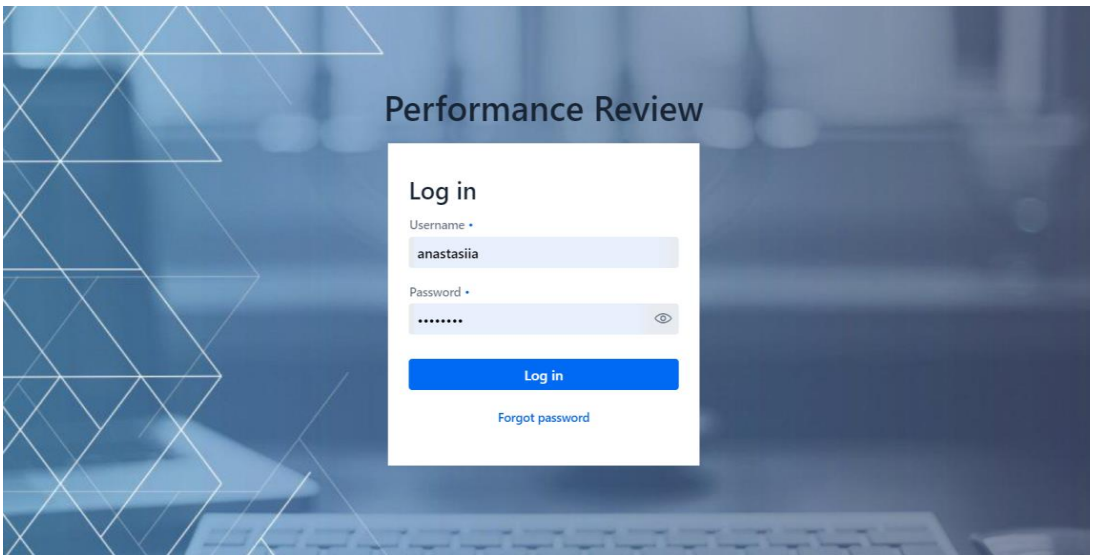

Рис. Б.2 – Сторінка входу у ролі «Співробітника»

Співробітник потрапляє на основну сторінку WEB-додатку (рис. Б.3). Він потрапляє на сторінку меню «Співробітник». На цій сторінці співробітник має можливість переглядати усі оцінювання, включаючи назву, дату початку та завершення, керівника та поле з відміткою про проходження оцінювання працівником, керівником. Після перегляду співробітник може обрати оцінювання, яке йому необхідно пройти.

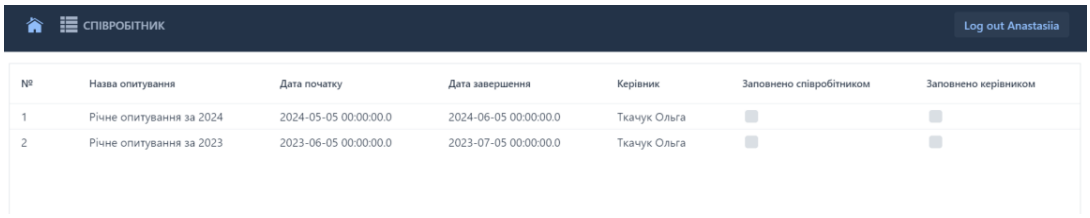

Рис. Б.3 – Головна сторінка системи у ролі «Співробітника»

Коли співробітник відкрив форму з необхідним оцінюванням (рис. Б.4). При проходженні оцінювання у співробітника є можливість зберегти свої результати натиснувши на кнопку «Зберегти» або по завершенню проходження натиснути кнопку «Готово».

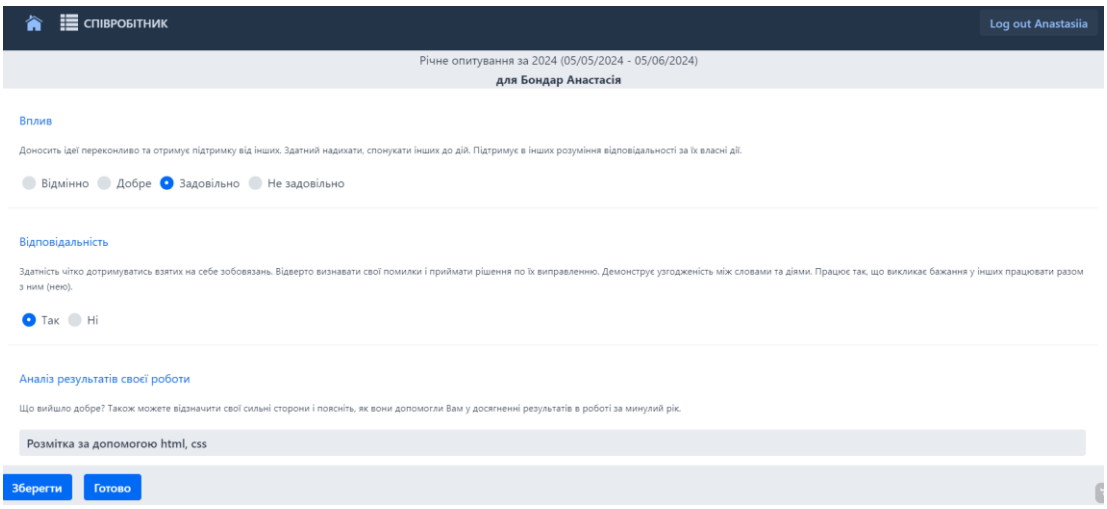

Рис. Б.4 – Проходження оцінювання у ролі «Співробітника»

Щоб розпочати роботу з WEB-додатком як керівник, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Керівник».

Після цього на сторінці авторизації для керівника, ввести дані та натиснути кнопку Login (рис. Б.5).

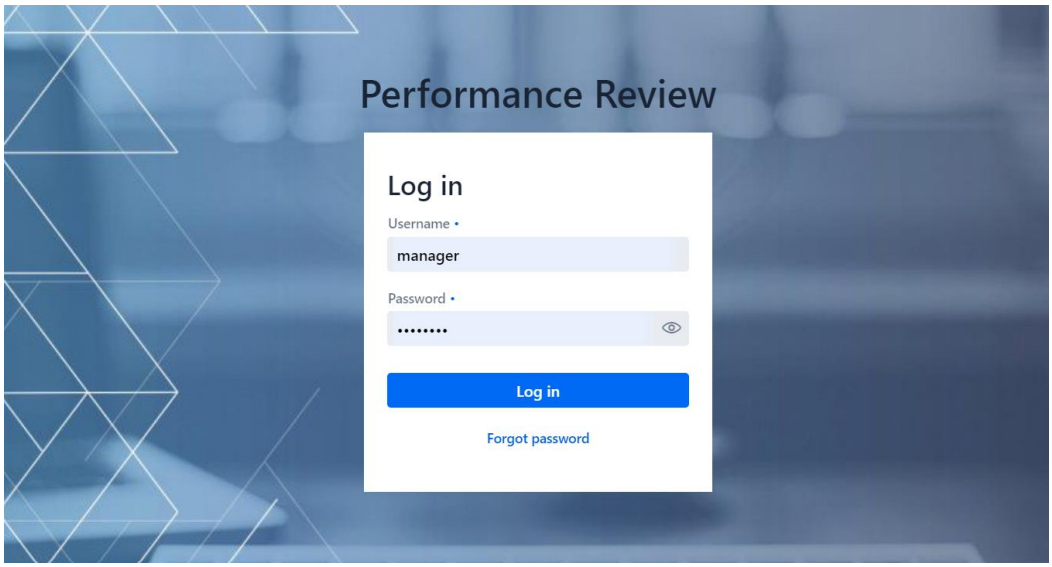

Рис. Б.5 – Сторінка входу у ролі «Керівника»

Керівник потрапляє на основну сторінку WEB-додатку. Так як керівник, є також і співробітником, тому він потрапляє на головну сторінку меню «Співробітник» (рис. Б.4). На цій сторінці він, як працівник, має можливість переглядати усі оцінювання, включаючи назву, дату початку та завершення, керівника та поле з відміткою про проходження оцінювання працівником, керівником. Також додається нова сторінка меню «Керівник» (рис. Б.6), в якій буде видно усі оцінювання підлеглих керівника.

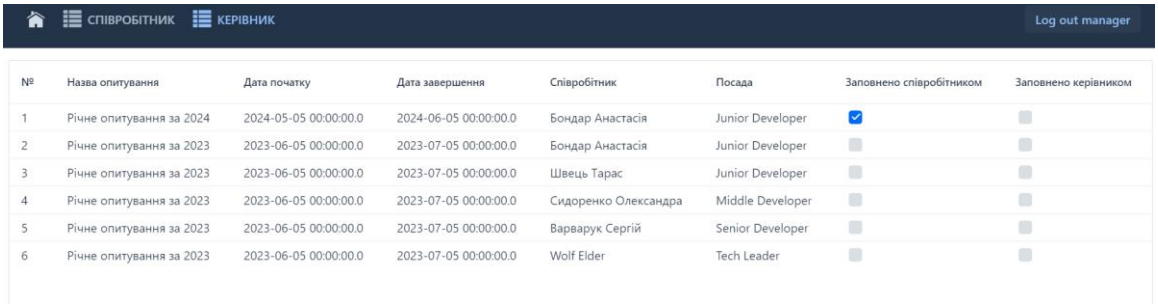

Рис. Б.6 – Сторінка меню «Керівник»

Керівник має змогу пройти оцінювання як в ролі співробітника так і в ролі керівника. При проходженні оцінювання у керівника є можливість зберегти свої результати натиснувши на кнопку «Зберегти» або по завершенню проходження натиснути кнопку «Готово».

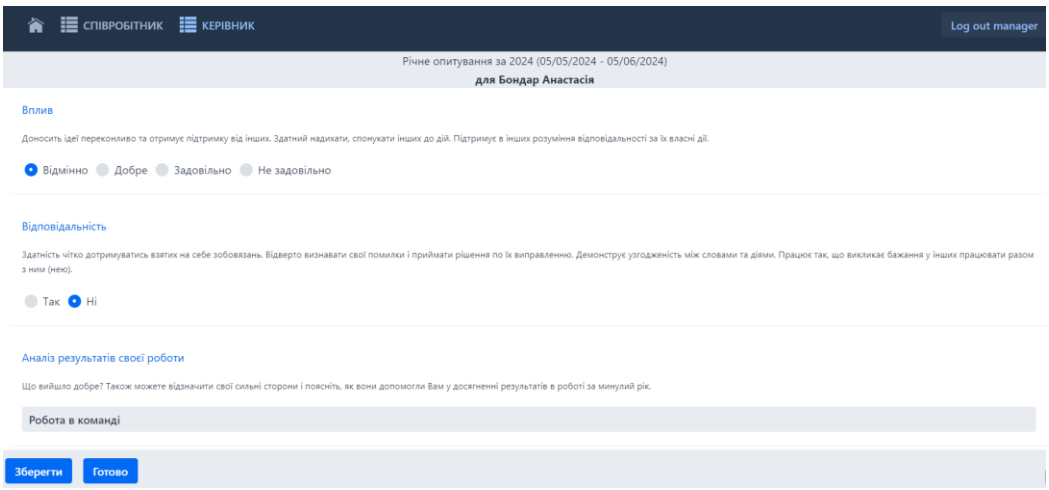

Рис. Б.7 – Проходження оцінювання у ролі «Керівника»

Після завершення проходження, якщо оцінювання було вже пройдене підлеглим відкриється форма з результатами оцінювання (рис. Б.8). В якій буде видно наскільки подібно відповіли керівник та співробітник. У тому разі, якщо поле питання світиться червоним кольором означає, що керівник відповів гірше, ніж співробітник, якщо зеленим – краще, якщо не підсвічується – однаково. В кінці результатів додасться поле для відгуку (рис. Б.9), де керівник і співробітник зможуть надати зворотній зв’язок щодо пройденого оцінювання.

Рис. Б.8 – Результати оцінювання

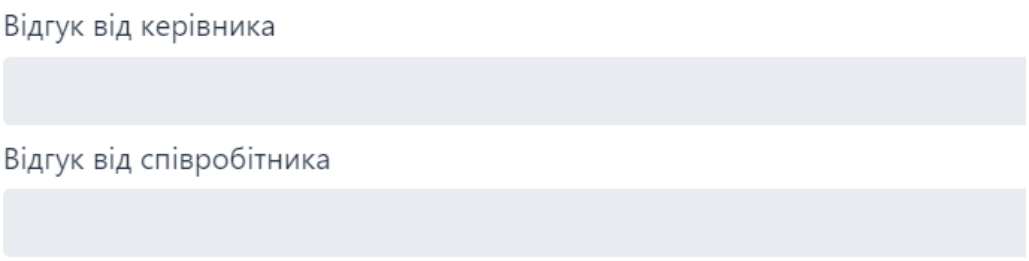

Рис. Б.9 – Поле для створення відгуку

Щоб розпочати роботу з WEB-додатком як адміністратор, необхідно пройти процес авторизації. Для цього на головній сторінці системи натиснути кнопку «Адміністратор».

Після цього на сторінці авторизації для адміністратора, ввести дані та натиснути кнопку Login (рис. Б.10).

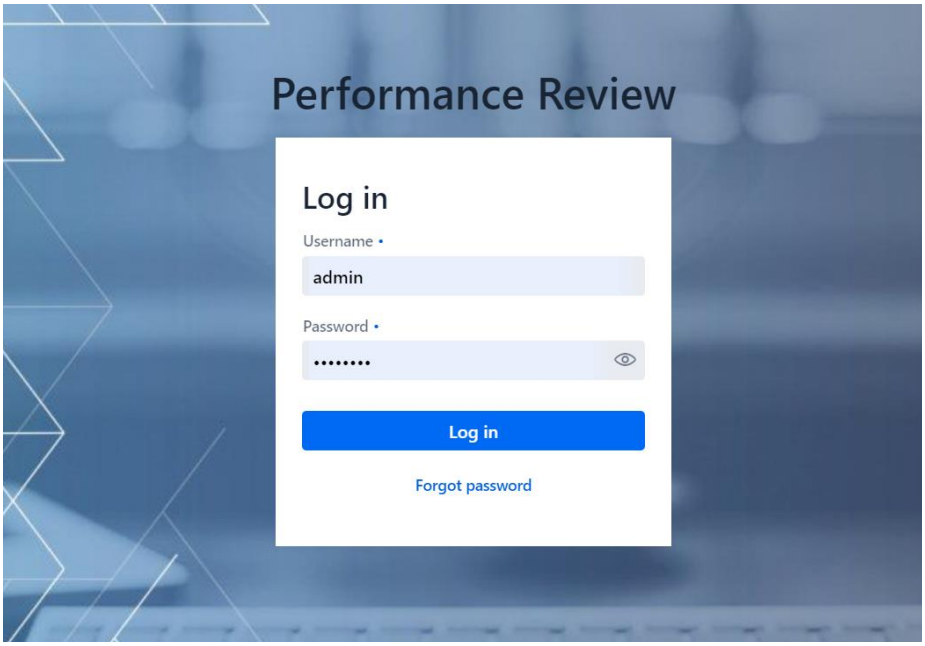

Рис. Б.10 – Сторінка входу у ролі «Адміністратора»

Керівник потрапляє на основну сторінку WEB-додатку (рис. Б.11). На лівій частині головної сторінки знаходиться меню. На правій частині знаходиться інформація з вибраного меню. Після авторизації, адміністратор потрапляє на сторінку меню «Користувачі». На цій сторінці адміністратор має можливість переглядати усіх створених користувачів. При натисненні на користувача відкриється форма редагування, видалення користувача з системи (рис. 3. Б.12). Також можна зареєструвати нового користувача натиснувши на кнопку «Зареєструвати співробітника». Відкриється форма, де можна буде вказати логін та пароль створеного користувача.

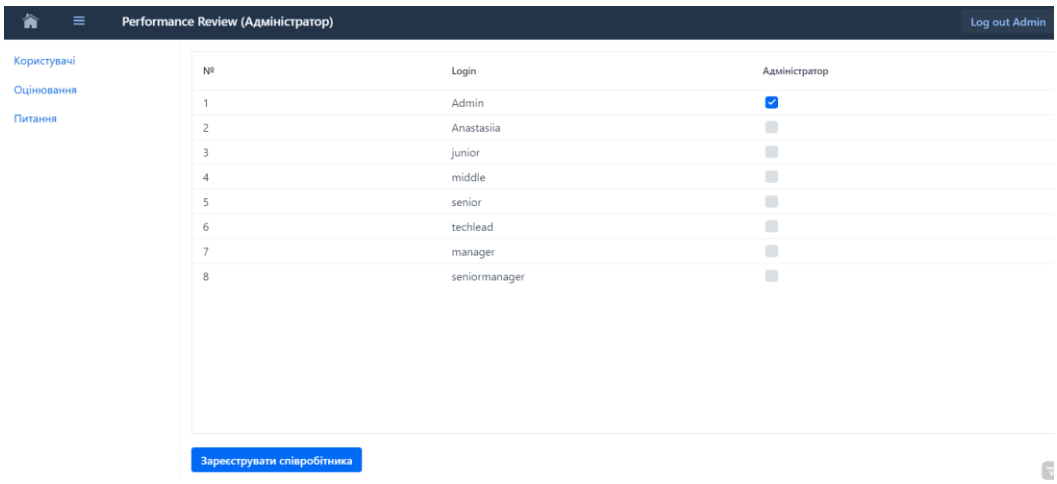

Рис. Б.11 – Головна сторінка системи з оцінюваннями у ролі «Адміністратора»

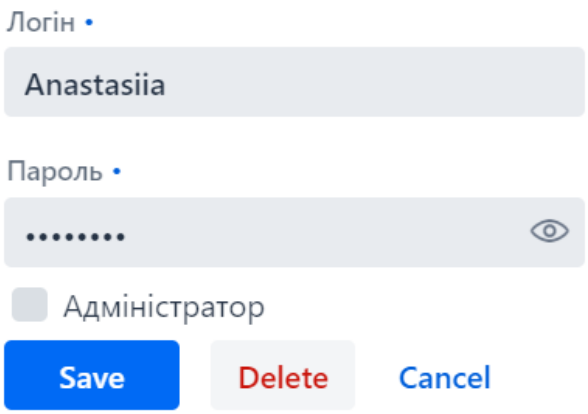

Рис. Б.12 – Форма створення, редагування та видалення користувача з системи

Наступна сторінка меню - це сторінка з оцінюваннями (рис. Б.13). На цій сторінці адміністратор має можливість переглядати усі створені оцінювання. При натисненні на оцінювання відкриється форма редагування, видалення оцінювання з системи (рис. Б.14). Також можна створити нове оцінювання натиснувши на кнопку «Створити оцінювання». Відкриється форма, де можна буде вказати назву, дату початку та завершення оцінювання.

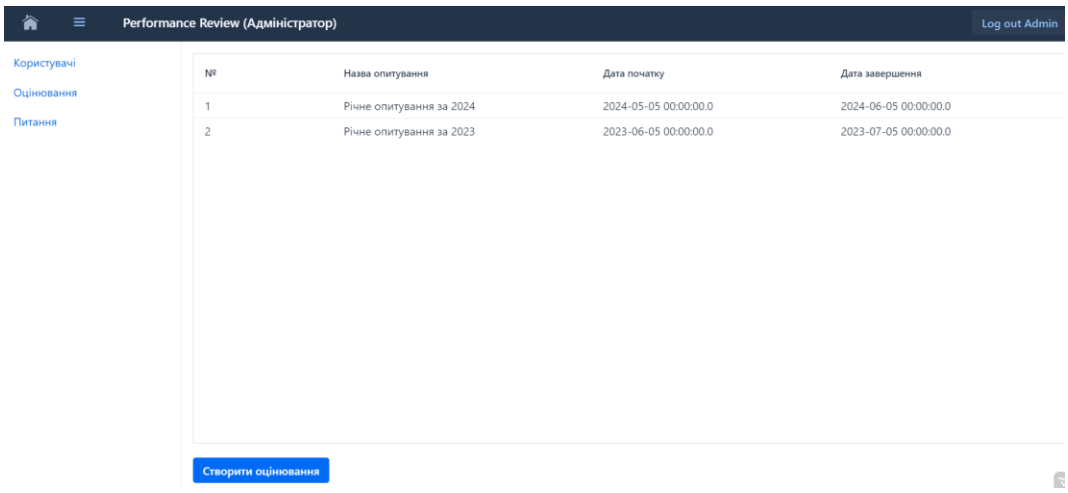

Рис. Б.13 – Сторінка меню з оцінюваннями у ролі «Адміністратора»

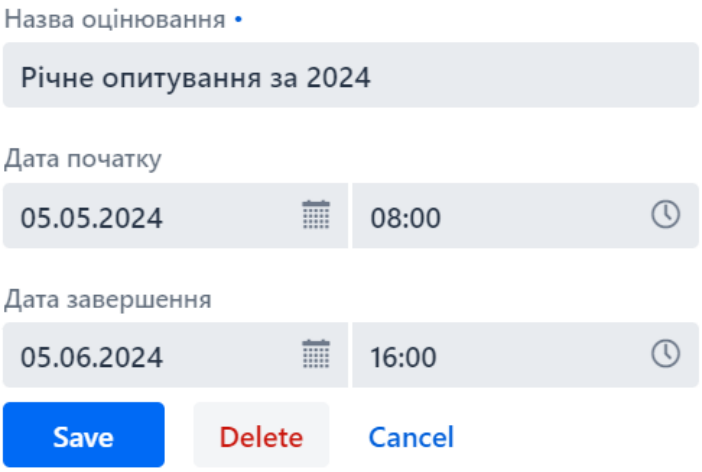

Рис. Б.14 – Форма створення, редагування та видалення оцінювання з системи

Наступна сторінка меню - це сторінка з питаннями (рис. Б.15). На цій сторінці адміністратор має можливість переглядати усі створені питання. При натисненні на питання відкриється форма редагування, видалення питання з системи (рис. Б.16). Також можна створити нове питання натиснувши на кнопку «Створити питання». Відкриється форма, де можна буде вказати назву питання, опис, тип відповіді та кількість варіантів відповідей.

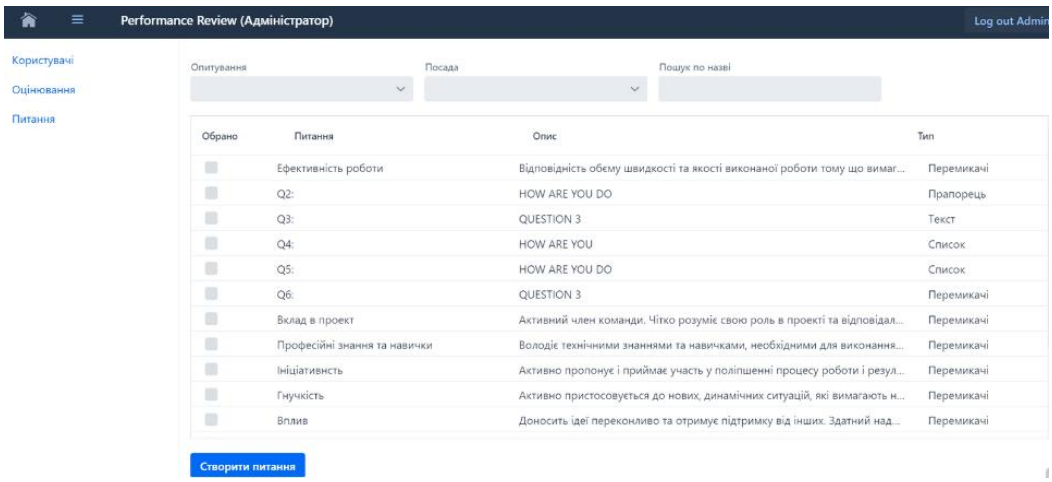

Рис. Б.15 – Сторінка меню з питаннями до оцінювання у ролі «Адміністратора»

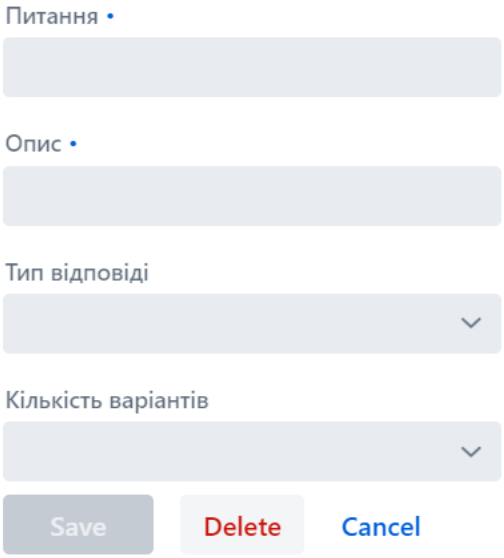

Рис. Б.16 – Форма створення, редагування та видалення питань з системи

ДОДАТОК В

ЛІСТИНГ ПРОГРАМИ

```
package com.afrosimova.prmanager;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PrmanagerApplication {

 public static void main(String[] args) {
        SpringApplication.run(PrmanagerApplication.class, args);
 }
}
```

```
package com.afrosimova.prmanager.views.employee;

import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Label;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.server.auth.AnonymousAllowed;
import com.vaadin.flow.theme.lumo.LumoUtility;

@Route(value = "")
@AnonymousAllowed
public class LobbyView extends VerticalLayout {
   public LobbyView() {
      setClassName("lobby-view");
      setSizeFull();
      getElement().getStyle().set("background-image",
"url('https://i.pinimg.com/originals/bc/fe/36/bcfe3678767235a08366a55d6ef7ac64.jpg')");

      //getElement().getStyle().set("height", "50%");
      //getElement().getStyle().set("width", "50%");
      getElement().getStyle().set("background-size", "cover");
```

```
        getElement().getStyle().set("background-position", "center");
        getElement().getStyle().set("background-repeat", "no-repeat");
        getElement().getStyle().set("display", "flex");
        getElement().getStyle().set("align-items", "center");
        getElement().getStyle().set("justify-content", "center");
//        getStyle().set("background-image", "url('img/test.png')");
//        getStyle().set("background-size", "cover");
//        getElement().getStyle().set("background-image", "url('img/test.png')");
//        getElement().getStyle().set("background-size", "cover");

        Component title = new H1("Performance Review");
        title.addClassNames(LumoUtility.TextColor.SUCCESS_CONTRAST,
                LumoUtility.TextAlignment.CENTER,
                LumoUtility.AlignSelf.CENTER,
                LumoUtility.Padding.LARGE);
        add(title);
        addClassNames(LumoUtility.JustifyContent.CENTER);
        Button employeeButton = new Button("Співробітник",
                //employeeButton.getStyle().set("background-color", "blue"),
        e -> getUI().ifPresent((a -> a.navigate("employee_surveys"))));
        styleButton(employeeButton);
        Button managerButton = new Button("Керівник",
                e -> getUI().ifPresent((a -> a.navigate("manager_surveys"))));
        styleButton(managerButton);
        Button adminButton = new Button("Адміністратор",
                e -> getUI().ifPresent((a -> a.navigate("admin/users"))));
        styleButton(adminButton);
        add(employeeButton, managerButton, adminButton);
    }
    private void styleButton(Button button) {
        button.addClassNames(LumoUtility.AlignSelf.CENTER,
                LumoUtility.Background.CONTRAST_40,
                LumoUtility.TextColor.PRIMARY_CONTRAST,
                LumoUtility.BoxShadow.XLARGE,
                LumoUtility.BorderRadius.LARGE);
        button.setWidth("160px");
    }
}
package com.afrosimova.prmanager.views.employee;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.login.LoginForm;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
```

```
import com.vaadin.flow.router.BeforeEnterEvent;
import com.vaadin.flow.router.BeforeEnterObserver;
import com.vaadin.flow.router.PageTitle;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.server.auth.AnonymousAllowed;

@Route("login")
@PageTitle("Login")
@AnonymousAllowed
public class LoginView extends VerticalLayout implements BeforeEnterObserver {

    private final LoginForm login = new LoginForm();

    public LoginView(){
        addClassName("login-view");
        setSizeFull();

        getElement().getStyle().set("background-image",
"url('https://i.pinimg.com/originals/bc/fe/36/bcfe3678767235a08366a55d6ef7ac64.jpg')");

        //getElement().getStyle().set("height", "50%");
        //getElement().getStyle().set("width", "50%");
        getElement().getStyle().set("background-size", "cover");
        getElement().getStyle().set("background-position", "center");
        getElement().getStyle().set("background-repeat", "no-repeat");
        getElement().getStyle().set("display", "flex");
        getElement().getStyle().set("align-items", "center");
        getElement().getStyle().set("justify-content", "center");

        setAlignItems(Alignment.CENTER);
        setJustifyContentMode(JustifyContentMode.CENTER);

        login.setAction("login");

        add(new H1("Performance Review"), login);
    }

    @Override
    public void beforeEnter(BeforeEnterEvent beforeEnterEvent) {
        // inform the user about an authentication error
        if(beforeEnterEvent.getLocation()
                .getQueryParameters()
                .getParameters()
```

```
            .containsKey("error")) {
        login.setError(true);
      }
    }
}
package com.afrosimova.prmanager.views.employee;

import com.afrosimova.prmanager.security.SecurityService;
import com.afrosimova.prmanager.views.employee.EmployeeSurveysView;
import com.afrosimova.prmanager.views.employee.LobbyView;
import com.afrosimova.prmanager.views.employee.ManagerSurveysView;
import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.applayout.AppLayout;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.html.Anchor;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.icon.VaadinIcon;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.FlexComponent.JustifyContentMode;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.tabs.Tab;
import com.vaadin.flow.component.tabs.Tabs;
import com.vaadin.flow.component.tabs.TabsVariant;
import com.vaadin.flow.router.RouteConfiguration;
import com.vaadin.flow.router.RouterLink;
import com.vaadin.flow.theme.lumo.LumoUtility;

import java.util.Optional;

/**
 * The main view is a top-level placeholder for other views.
 */
public class MainLayout extends AppLayout {

    private final Tabs menu;
    private final SecurityService securityService;

    public MainLayout(SecurityService securityService) {
        this.securityService = securityService;
        menu = createMenu();
        addToNavbar(createHeaderContent());
    }
```

```
private Component createHeaderContent() {
    final HorizontalLayout layout = new HorizontalLayout();
    layout.getThemeList().add("dark");
    layout.setPadding(true);
    layout.setSpacing(false);
    layout.setId("header");
    layout.setWidthFull();
    layout.setAlignItems(FlexComponent.Alignment.END);
    layout.setJustifyContentMode(JustifyContentMode.BETWEEN);

    final Span brand = new Span();
    final Anchor brandLink = new Anchor("/", brand);
    brandLink.addClassName("navbar-brand");
    layout.add(brandLink);
    layout.add(menu);

    String u = securityService.getAuthenticatedUser().getUsername();
    Button logout = new Button("Log out " + u, e -> securityService.logout()); // <2>
    var header = new HorizontalLayout(logout);
    //header.getStyle().set( "border" , "6px dotted DarkOrange" ) ;
    header.setJustifyContentMode(JustifyContentMode.END);
    header.setWidthFull();
    header.addClassNames(
            LumoUtility.Padding.Vertical.NONE,
            LumoUtility.Padding.Horizontal.MEDIUM);

    layout.add(header);

    return layout;
}

private Tabs createMenu() {
    final Tabs tabs = new Tabs();
    tabs.setOrientation(Tabs.Orientation.HORIZONTAL);
    tabs.addThemeVariants(TabsVariant.LUMO_MINIMAL);
    tabs.setId("tabs");
    tabs.add(createMenuItems());
    return tabs;
}

private Tab[] createMenuItems() {
```

```
        return new Tab[]{
                new Tab(createRouterLink("", VaadinIcon.HOME, LobbyView.class)),
                new Tab(createRouterLink("Співробітник", VaadinIcon.LIST,
EmployeeSurveysView.class)),
                new Tab(createRouterLink("Керівник", VaadinIcon.LIST, ManagerSurveysView.class)),
        };
    }

    private RouterLink createRouterLink(String name, VaadinIcon viewIcon,
                                        Class<? extends Component> navigationTarget) {
        final RouterLink routerLink = new RouterLink(name, navigationTarget);
        routerLink.addComponentAsFirst(viewIcon.create());
        routerLink.addClassNames("flex", "gap-s", "uppercase");
        return routerLink;
    }

    @Override
    protected void afterNavigation() {
        super.afterNavigation();
        updateChrome();
    }

    private void updateChrome() {
        getTabWithCurrentRoute().ifPresent(menu::setSelectedTab);
    }

    private Optional<Tab> getTabWithCurrentRoute() {
        final String currentRoute = RouteConfiguration.forSessionScope()
                .getUrl(getContent().getClass());
        return menu.getChildren().filter(tab -> hasLink(tab, currentRoute))
                .findFirst().map(Tab.class::cast);
    }

    private boolean hasLink(Component tab, String currentRoute) {
        return tab.getChildren().filter(RouterLink.class::isInstance)
                .map(RouterLink.class::cast).map(RouterLink::getHref)
                .anyMatch(currentRoute::equals);
    }
```

ДОДАТОК Г

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ РОБОТИ СПІВРОБІТНИКІВ»

Виконала: студентка 4 курсу, групи 2КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Афросімова А. А.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О. В.
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

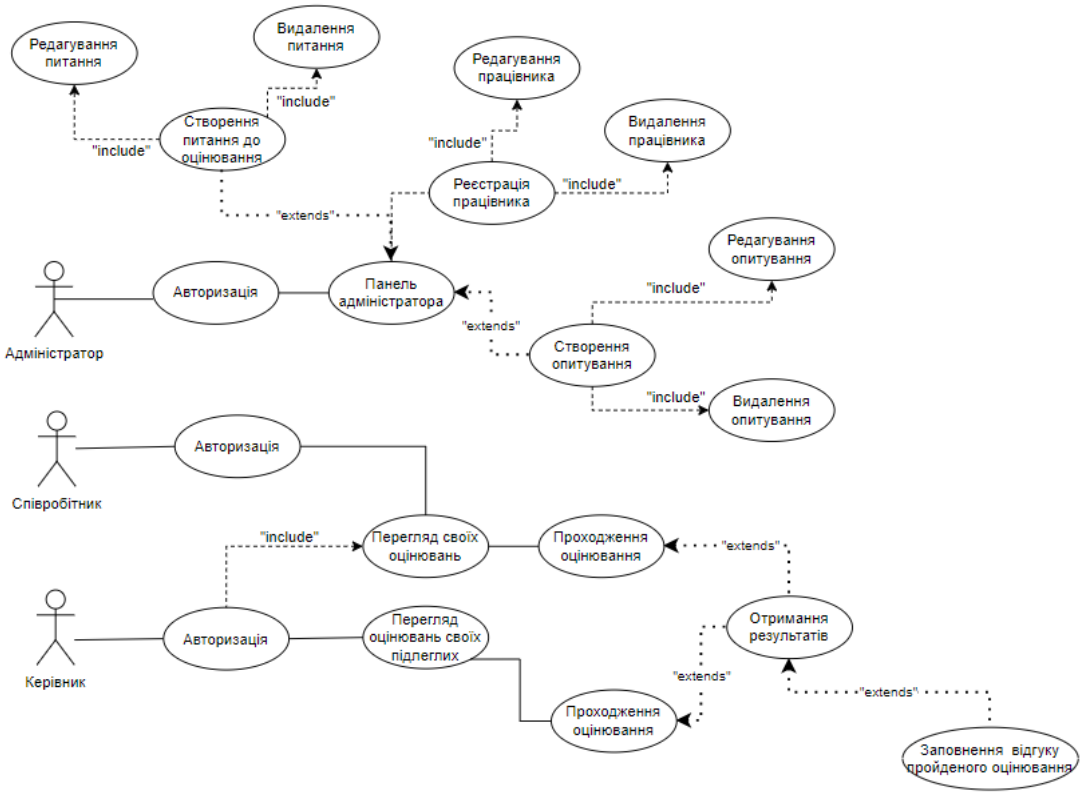

Рисунок Г.1 – UML-діаграма прецедентів WEB-додатку для оцінювання продуктивності роботи співробітників

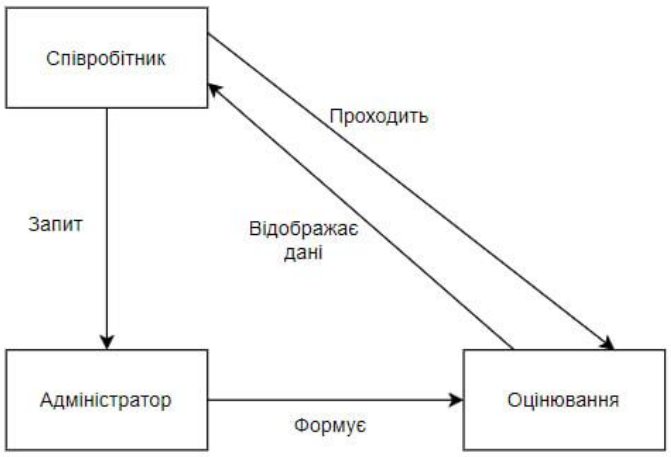

Рисунок Г.2 – UML-діаграма взаємодій WEB-додатку для оцінювання продуктивності роботи співробітників

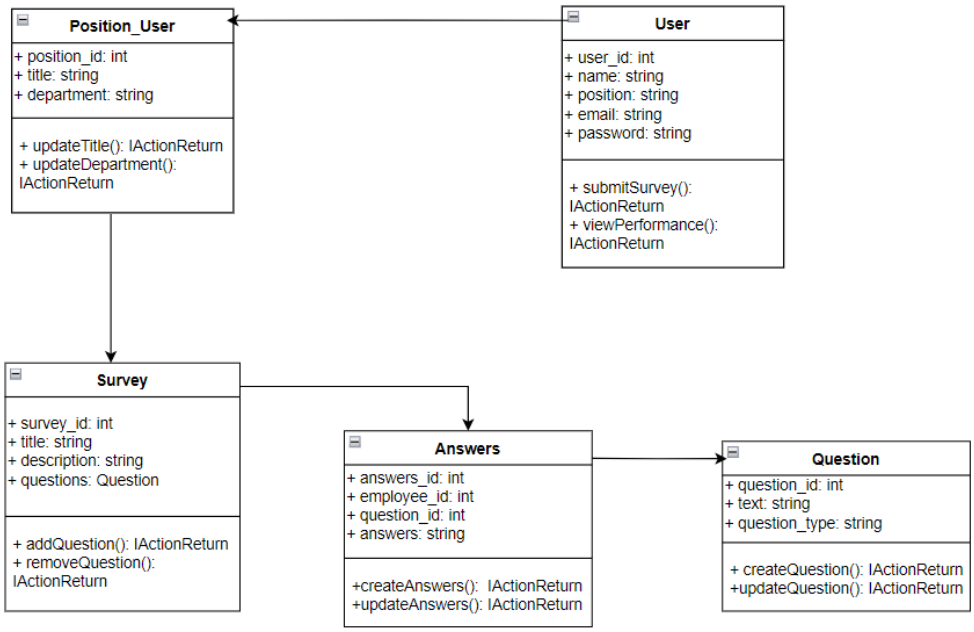

Рисунок Г.3 – Загальна UML-діаграма класів

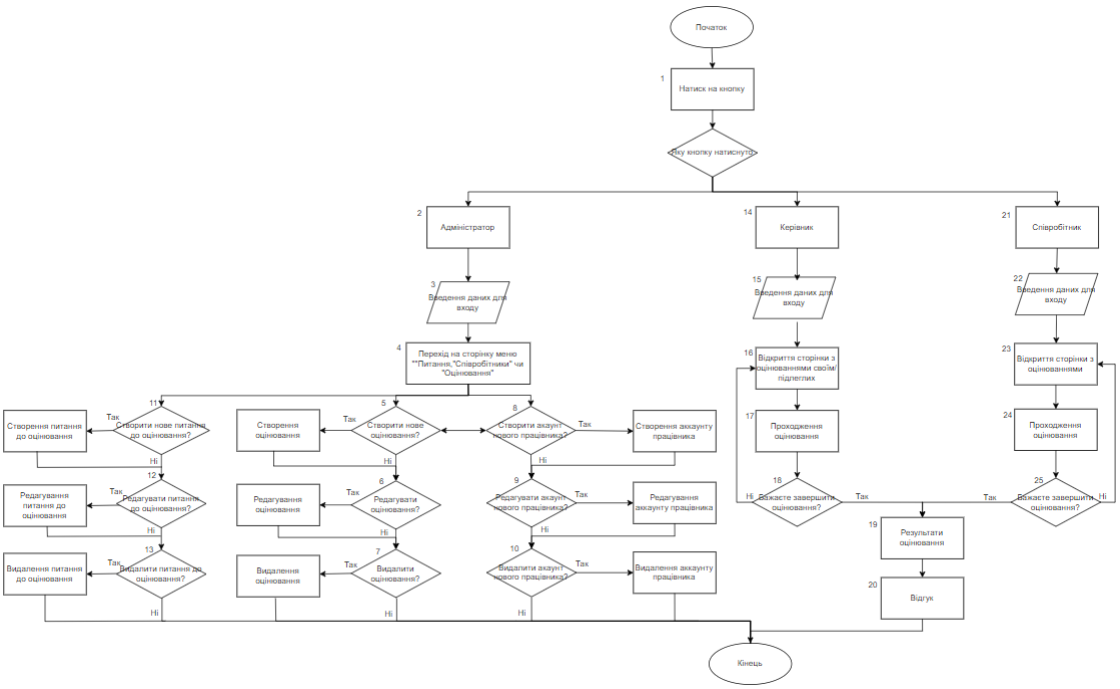

Рисунок Г.4 – Схема алгоритму роботи WEB-додатку для оцінювання продуктивності роботи співробітників

Рисунок Г.5 – ER-діаграма бази даних

	EMPLOYEE_ID	USER_ID	FIRST_NAME	LAST_NAME	POSITION_ID	MANAGER_ID	EMAIL
1	1	2	Anastasiia	Afrosimova	1	6	aa@gmail.com
2	2	3	Simple	Jun	1	6	junior@gmail.com

Рисунок Г.6 – Фрагмент реляційної моделі бази даних

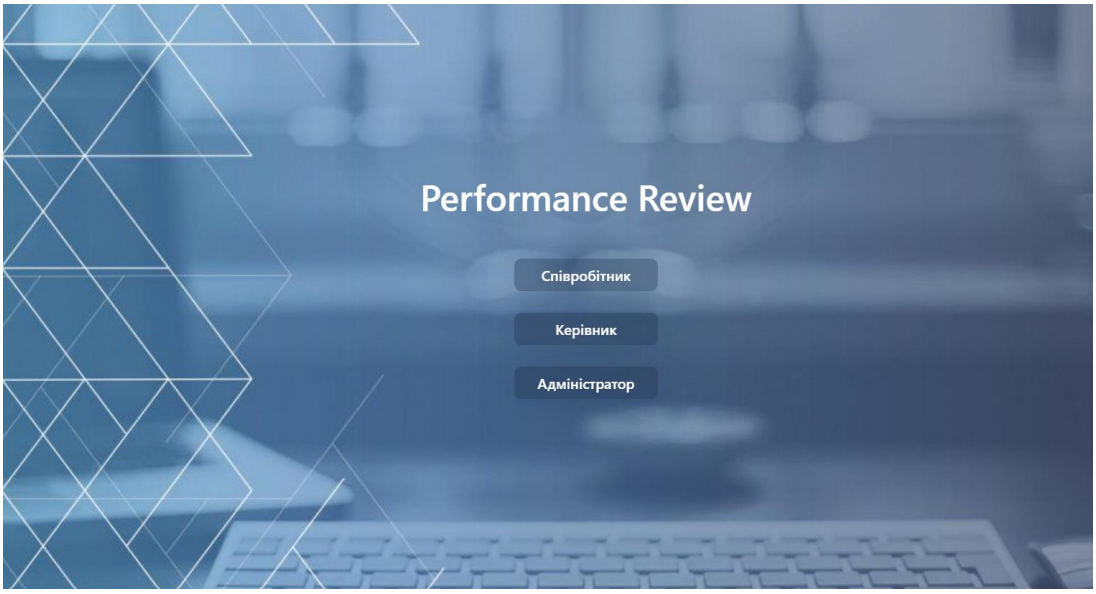

Рисунок Г.7 – Головна сторінка WEB-додатку

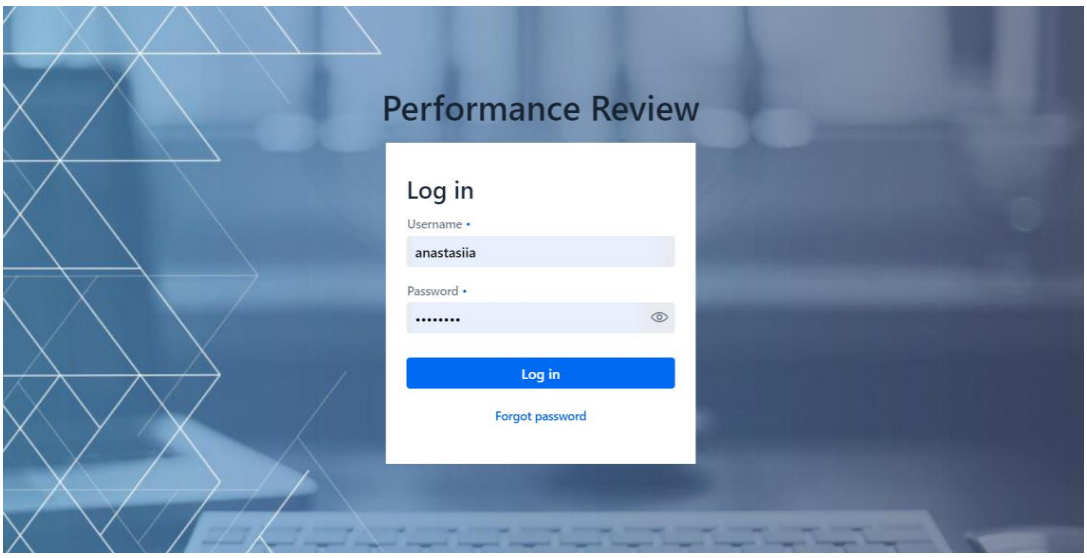

Рисунок Г.8 – Сторінка авторизації у ролі «Співробітника», «Керівника», «Адміністратора»

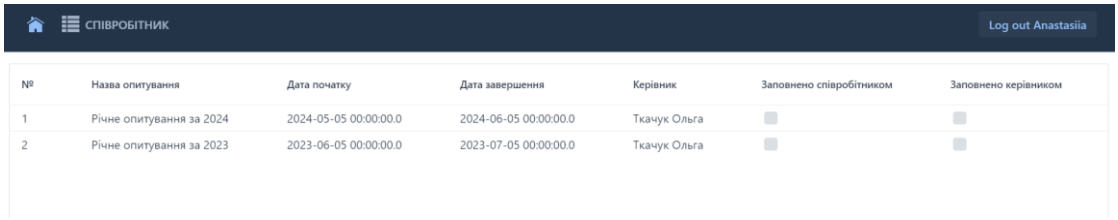

Рисунок Г.9 – Сторінка після авторизації у ролі «Співробітника», «Керівника»

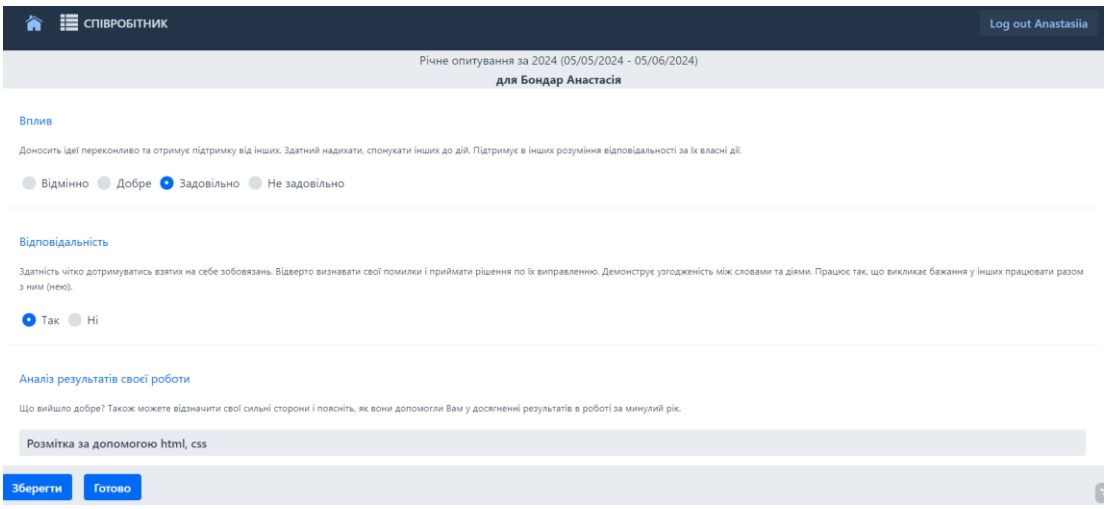

Рисунок Г.10 – Проходження оцінювання у ролі «Співробітника», «Керівника»

Рисунок Г.11 – Результати оцінювання у ролі «Співробітника», «Керівника»

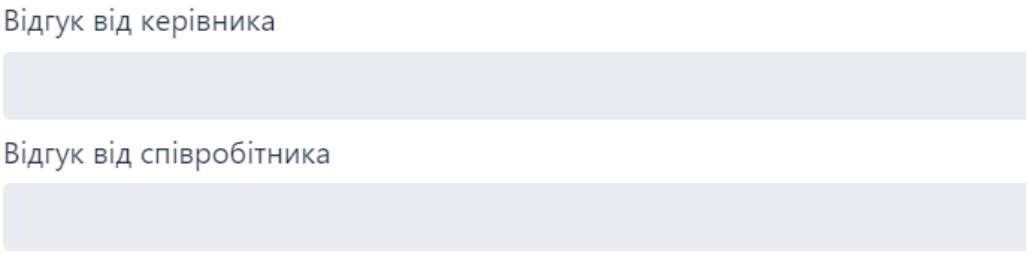

Рисунок Г.12 – Створення відгуку у ролі «Співробітника», «Керівника»

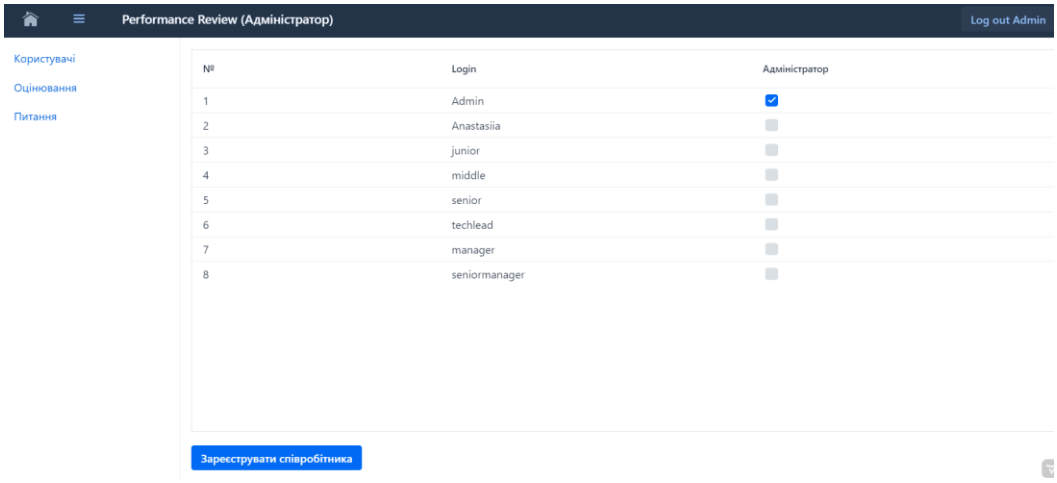

Рисунок Г.13 – Головна сторінка системи з оцінюваннями у ролі «Адміністратора»

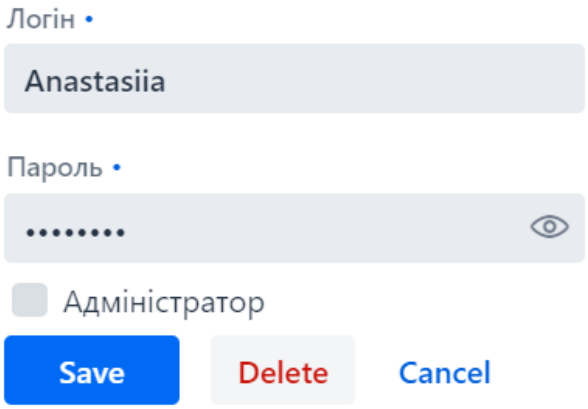

Рисунок Г.14 – Форма створення, редагування та видалення користувача з системи

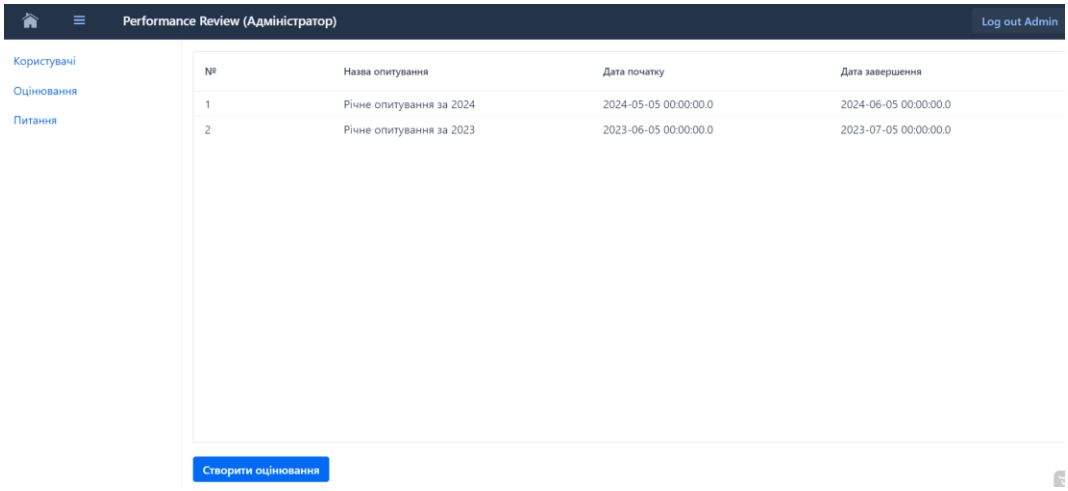

Рисунок Г.15 – Сторінка меню з оцінюваннями у ролі «Адміністратора»

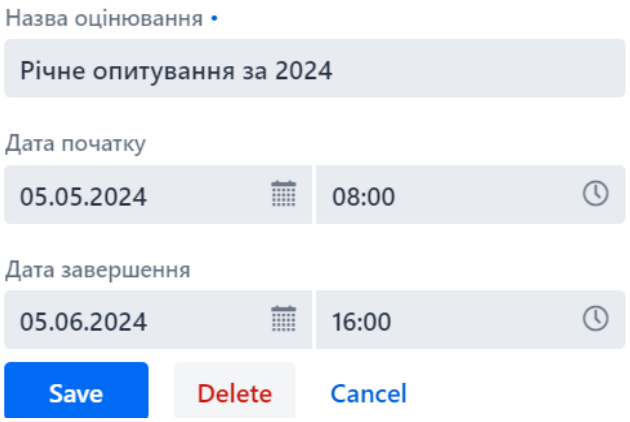

Рисунок Г.16 – Форма створення, редагування та видалення оцінювання з системи

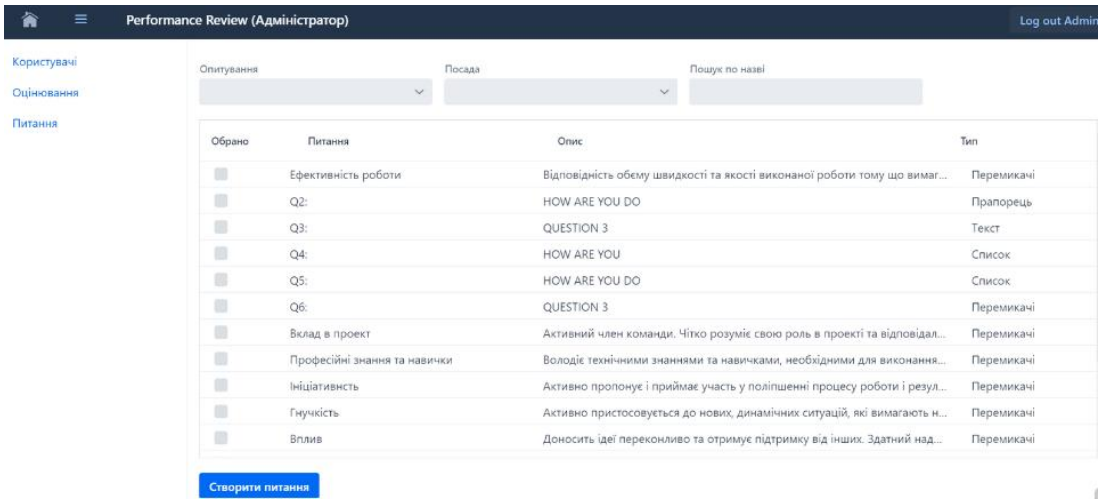

Рисунок Г.17 – Сторінка меню з питаннями до оцінювання у ролі «Адміністратора»

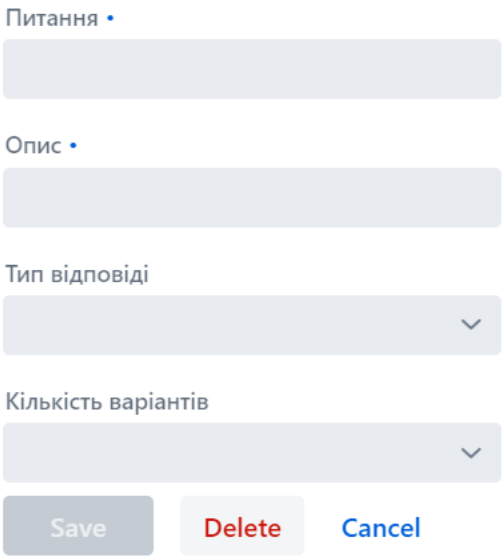

Рисунок Г.18 – Форма створення, редагування та видалення питань з системи