

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інформаційна веб-орієнтована HRM-система
для цифрової трансформації та автоматизації
процесів управління персоналом»**

Виконав студент групи ІПЗ-21-5
БУЛАНОВ Олександр Павлович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. каф. інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БУЛАНОВ Олександр Павлович**

Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**

Тема роботи: **«Інформаційна веб-орієнтована HRM-система для цифрової трансформації та автоматизації процесів управління персоналом»**

затверджена Наказом закладу вищої освіти від «23» лютого 2025 р., №74/с

Вихідні дані для роботи: **розробка системи управління людськими ресурсами, що дозволить ефективно керувати персоналом всередині компаній.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Толстой І.А.	27.02.2025р.	27.02.2025р.
2	Толстой І.А.	24.04.2025р.	24.04.2025р.
3	Толстой І.А.	02.05.2025р.	02.05.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу - інформаційної HRM-системи для управління персоналом у компанії - та пояснювальної записки. Пояснювальна записка містить 106 сторінок, з яких основний текст займає 75 сторінок, додатки - 31 сторінку. У роботі наведено 34 ілюстрації, 14 таблиць та використано 40 джерел інформації.

Головною метою є створення зручної, функціональної HRM-платформи, яка автоматизує основні процеси роботи з персоналом, забезпечує ефективне ведення кадрової документації, рекрутингу, бухгалтерського обліку та внутрішньої комунікації.

У роботі визначено ключові завдання, що стосуються проєктування архітектури системи, визначення ролей користувачів та їх функціоналу, аналізу аналогічних рішень на ринку, а також запропоновано власну реалізацію системи. Представлено огляд загальної структури проєкту, описано алгоритми взаємодії між модулями, архітектуру бази даних та діаграми класів.

У процесі розробки було реалізовано систему з кількома типами користувачів (адміністратор, HR, працівник, кандидат, сторонній користувач), кожен з яких має доступ до визначеного функціоналу. Також проведено тестування програмного забезпечення, в результаті якого були виявлені та усунуті певні помилки для забезпечення стабільної роботи системи.

КЛЮЧОВІ СЛОВА: HRM, СИСТЕМА, РЕКРУТИНГ, ПЕРСОНАЛ, БАЗА ДАНИХ, WEB-ІНТЕРФЕЙС, PHP, SYMFONY, BACKEND, FRONTEND.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна веб-орієнтована HRM-система для цифрової трансформації та автоматизації процесів управління персоналом Пояснювальна записка	Літ.	Арк.	Аркуші	
Розроб.		Буланов О.П.							
Керівник		Толстой І.А.					3	106	
						Житомирська політехніка, група ІПЗ-21-5			
Н. контр.									
Зав. каф.		Вакалюк Т.А.							

ABSTRACT

The bachelor's final qualification work consists of a software complex - an information HRM system for personnel management in a company - and an explanatory note. The explanatory note contains 106 pages, of which the main text takes up 75 pages, the appendix - 31 pages. The work contains 34 illustrations, 14 tables and uses 40 sources of information.

The main goal is to create a convenient, functional HRM platform that automates the main processes of working with personnel, ensures effective maintenance of personnel documentation, recruiting, accounting and internal communication.

The work identifies key tasks related to the design of the system architecture, definition of user roles and their functionality, analysis of similar solutions on the market, and also proposes its own implementation of the system. An overview of the general structure of the project is presented, algorithms for interaction between modules, database architecture and class diagrams are described.

During the development process, a system with several types of users (administrator, HR, employee, candidate, third-party user) was implemented, each of which has access to a specific functionality. Software testing was also conducted, as a result of which certain errors were identified and eliminated to ensure stable operation of the system.

KEYWORDS: HRM, SYSTEM, RECRUITMENT, PERSONNEL, DATABASE, WEB INTERFACE, PHP, SYMFONY, BACKEND, FRONTEND.

		Буланов О.П.			ПІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1 Поняття, завдання та роль HRM-систем в управлінні персоналом	9
1.2 Цифрова трансформація в контексті сучасного менеджменту персоналу	11
1.3 Аналіз сучасних веб-орієнтованих HRM-систем: переваги, недоліки, перспективи	13
1.4 Постановка задачі дослідження та розробки HRM-системи	20
1.5 Обґрунтування вибору засобів та вимоги до апаратного забезпечення....	23
Висновки до першого розділу	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-ОРІЄНТОВАНОЇ HRM-СИСТЕМИ.....	27
2.1 Функціональні та нефункціональні вимоги до системи.....	27
2.2 Архітектура та технологічний стек системи.....	31
2.3 Проєктування бази даних та модулів системи.....	33
2.4 Розробка користувацького інтерфейсу та логіки взаємодії	42
Висновки до другого розділу.....	53
РОЗДІЛ 3. РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ HRM-СИСТЕМИ.....	54
3.1 Реалізація ключових компонентів системи	54
3.2 Тестування та забезпечення надійності роботи.....	56
Висновки до третього розділу	69
ВИСНОВКИ	71
ДОДАТКИ	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

HRM - система менеджменту людських ресурсів;

UI - інтерфейс користувача;

ЦП - центральний процесор;

JWT - JSON Web Token;

ОЗП - оперативна пам'ять.

ERP - система планування ресурсів підприємства

CRM - система управління взаємовідносинами з клієнтом

API - інтерфейс програмування додатків

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає в необхідності автоматизації процесів управління персоналом у сучасних компаніях, що динамічно розвиваються. У зв'язку з високою конкуренцією на ринку праці та постійними змінами в законодавстві, компанії потребують ефективних інструментів для керування людськими ресурсами, найму працівників та організації внутрішніх процесів. HRM-системи (Human Resource Management Systems) дозволяють оптимізувати процеси рекрутингу, обліку працівників, ведення документації, оцінювання персоналу, а також забезпечують зручну комунікацію між усіма учасниками робочого процесу. Враховуючи тенденції цифровізації бізнесу, створення адаптивного вебсайту, що поєднує в собі як функціонал для зовнішніх користувачів (пошук та подання на вакансії), так і внутрішні інструменти для HR-відділу та керівництва, є актуальним та доцільним.

Метою роботи є розробка вебзастосунку HRM-системи, що охоплює повний цикл управління людськими ресурсами - від найму до внутрішньої діяльності працівників. Встановлена мета обумовлює виконання різних завдань, зокрема аналіз наявних рішень у сфері HRM-систем, проектування архітектури вебзастосунку з урахуванням різних ролей користувачів, обґрунтування вибору засобів реалізації, а також створення системи для менеджменту вакансій, кандидатів, персоналу та внутрішніх процесів компанії.

Об'єктом дослідження виступає процес управління персоналом із використанням цифрових інформаційних систем у межах корпоративного середовища. Зокрема, досліджується, яким чином сучасні технологічні рішення дозволяють автоматизувати ключові HR-процеси - від пошуку та найму кандидатів до обліку робочого часу, управління зарплатами, оцінювання ефективності працівників та забезпечення прозорості взаємодії між відділом кадрів, керівництвом і працівниками.

Предметом дослідження є програмна реалізація функціоналу HRM-системи, включаючи рекрутинг, облік робочого часу, графіки, оцінювання, документообіг та інші аспекти керування персоналом.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Методами дослідження є застосування об'єктно-орієнтованого програмування (ООР), проектування клієнт-серверної архітектури, побудови RESTful API, використання компонентного підходу у фронтенд-розробці, а також методи порівняльного аналізу програмного забезпечення, моделювання бізнес-процесів і тестування ПЗ.

Практичне значення роботи полягає в розробці HRM-системи, яка може бути використана компаніями для оптимізації кадрової роботи, зменшення адміністративного навантаження та підвищення ефективності внутрішньої взаємодії.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Поняття, завдання та роль HRM-систем в управлінні персоналом

У сучасному бізнес-середовищі, яке постійно змінюється під впливом цифрових технологій, ефективне управління персоналом стає не лише необхідністю, а й ключовою умовою стабільного розвитку будь-якої організації. Підвищена динаміка змін на ринку праці, розвиток віддаленої зайнятості, потреба в оперативному прийнятті рішень і зростаючі обсяги даних про працівників потребують впровадження новітніх інструментів. У цьому контексті особливої актуальності набувають HRM-системи (Human Resource Management Systems) - цифрові рішення, що забезпечують комплексний підхід до управління людськими ресурсами.

HRM-системи являють собою програмне забезпечення, призначене для автоматизації всіх основних етапів взаємодії з працівниками: від підбору персоналу та оформлення на роботу до супроводу їх професійного розвитку, моніторингу ефективності та завершення трудових відносин [36]. Вони дозволяють не лише зберігати та обробляти великі обсяги інформації, а й організовувати її в структурований вигляд, що суттєво полегшує доступ до даних для керівників, HR-фахівців та самих співробітників.

Однією з головних переваг HRM-систем є автоматизація рутинних і повторюваних завдань. Такі дії, як облік робочого часу, контроль за відпустками та лікарняними, ведення електронної особової справи, розрахунок заробітної плати з урахуванням премій і надбавок, відстеження змін у посаді або структурі підрозділу - можуть бути повністю або частково делеговані цифровій системі. Це значно знижує навантаження на кадрову службу, скорочує кількість помилок і підвищує точність розрахунків.

Крім технічної автоматизації, HRM-системи виконують стратегічну функцію. Вони створюють єдиний інформаційний простір, у якому

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

відображаються як поточні, так і історичні дані про кожного працівника. Це дозволяє швидко аналізувати динаміку розвитку персоналу, виявляти сильні та слабкі сторони команди, планувати кар'єрні маршрути та підвищення кваліфікації. Також система може надавати аналітичні звіти про плинність кадрів, рівень залученості, відповідність працівників цілям компанії, що є цінною інформацією для управлінських рішень.

Застосування HRM-систем відкриває нові можливості у сфері рекрутингу. Зокрема, автоматизовані інструменти для аналізу резюме, сортування кандидатів, планування співбесід, створення рейтингових списків кандидатів та інтеграція з популярними платформами для пошуку роботи дозволяють значно скоротити час на підбір персоналу та підвищити якість найму. HR-фахівці можуть створювати шаблони вакансій, формувати запити до системи згідно з заданими параметрами та миттєво отримувати інформацію про релевантних кандидатів.

Інтеграція HRM-систем з іншими корпоративними рішеннями (ERP, CRM, бухгалтерією тощо) робить ці платформи ще більш потужним інструментом. Вона дозволяє обмінюватися інформацією в реальному часі, синхронізувати дані між відділами, уникати дублювання записів та узгоджувати діяльність різних підрозділів. Наприклад, дані про відпустки можуть автоматично впливати на графіки робочих змін або облік заробітної плати.

Особливу увагу в сучасних HRM-системах приділяють зручності інтерфейсу та доступу. Працівники компанії можуть самостійно переглядати свій графік роботи, заробітну плату, залишок відпусток, надавати документи, оновлювати особисті дані, залишати запити до HR-служби чи переглядати оцінки своєї діяльності. Усе це сприяє прозорості управлінських процесів, покращенню комунікації між працівником і керівництвом, а також формуванню довіри всередині організації [37].

Окремо варто згадати питання безпеки даних. HRM-системи працюють із великою кількістю чутливої інформації: паспортні дані, фінансова інформація, історія зайнятості, медичні довідки тощо. Саме тому сучасні рішення повинні

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

дотримуватись високих стандартів безпеки: шифрування даних, багаторівнева автентифікація, контроль прав доступу та обов'язкове логування дій користувачів [33]. Такі механізми дозволяють уникнути витоків інформації, несанкціонованого доступу або зловживання службовими обов'язками.

У підсумку можна зазначити, що HRM-системи є не просто цифровим інструментом, а повноцінним стратегічним ресурсом підприємства. Їх впровадження дає змогу не лише підвищити ефективність внутрішніх процесів, а й створити умови для сталого зростання компанії, її адаптивності до змін і збереження конкурентоспроможності в умовах глобального ринку. У сучасних реаліях цифровізації бізнесу важко уявити ефективну роботу організації без системного підходу до управління людським капіталом.

1.2 Цифрова трансформація в контексті сучасного менеджменту персоналу

HRM-система являє собою потужний та цінний інструмент при виконанні автоматизованих операцій управління персоналом. Мета цієї роботи полягає у розробці веб-системи менеджменту людських ресурсів, що використовує різноманітні сучасні підходи та технології і здатна представити функціонал, що складе достатню конкуренцію існуючих застосунків на ринку.

На першому етапі було проведено аналіз наявних HRM-рішень. У ході цього дослідження порівнювались популярні продукти, їх архітектура, UX/UI-рішення, набір функцій і гнучкість адаптації під різні типи бізнесу. Визначення ключових потреб кожної групи користувачів — адміністратора, HR-фахівця, кандидата та штатного працівника - дозволило сформулювати вимоги до функціональності, яка має бути реалізована у рамках проєкту. У результаті було складено загальну технічну специфікацію, що охоплює логіку системи, основні модулі, вимоги до безпеки, розмежування прав доступу та інші критично важливі аспекти архітектури.

Одним із центральних блоків системи стала рекрутингова підсистема. Вона дозволяє HR-відділу створювати вакансії, розміщувати їх на публічній сторінці

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

платформи, керувати заявками від кандидатів, призначати співбесіди, формувати етапи відбору та виставляти рейтинг після проходження кожного етапу. Окремо передбачено можливість залишати зворотний зв'язок про результати співбесіди, що підвищує прозорість і професіоналізм у взаємодії з кандидатами. Платформа також підтримує інтеграцію з Google Meet для надсилання автоматизованих запрошень на онлайн-інтерв'ю, а також зберігання відеозаписів співбесід для подальшого аналізу.

Облік персоналу - ще один важливий функціональний блок. Він реалізований у вигляді системи електронних карток працівників, у якій відображається повна інформація про посаду, підрозділ, графік роботи, історію змін, переведення між відділами, відпустки та лікарняні. Система дозволяє HR-фахівцям швидко орієнтуватися в структурі компанії та відстежувати динаміку змін без потреби в ручному веденні документації.

Бухгалтерська підсистема охоплює весь процес управління заробітною платою, включаючи розрахунок основної ставки, надбавок, премій та штрафів. Історія виплат зберігається в базі даних і доступна як для адміністратора, так і для самого працівника. Передбачена можливість завантаження та перегляду пов'язаних фінансових документів, зокрема довідок, розрахункових листів і контрактів.

Система оцінювання працівників реалізована у вигляді модуля, що дозволяє керівникам створювати звіти про виконання завдань, аналізувати динаміку результативності та приймати обґрунтовані рішення щодо підвищення кваліфікації, кар'єрного зростання або змін у графіку роботи. Працівники мають доступ до власних звітів та оцінок, а також можуть ознайомлюватися з рекомендаціями щодо поліпшення результатів.

Важливою частиною системи є функціонал для кандидатів. Вони можуть створювати облікові записи, завантажувати резюме, сертифікати та інші документи, переглядати статус своїх заявок, отримувати повідомлення електронною поштою та залишати зворотний зв'язок про досвід проходження

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

процесу найму. Такий підхід не лише підвищує рівень прозорості, а й позитивно впливає на репутацію компанії.

Працівники компанії мають окремий функціональний блок, через який вони можуть переглядати й частково редагувати власні дані, слідкувати за графіком роботи, отримувати завдання, переглядати виплати та взаємодіяти з внутрішніми сервісами. Також передбачено функціонал для завантаження документів (наприклад, дипломів, сертифікатів), що можуть бути використані для формування особової справи.

Уся система базується на модульній архітектурі, що дозволяє легко додавати нові функції без потреби в глобальному переписуванні коду. Передбачено гнучке налаштування прав доступу залежно від ролі користувача, а безпека персональних даних забезпечується через сучасні засоби шифрування та захисту на рівні API.

Таким чином, розроблена система не лише відповідає ключовим вимогам сучасного HR-середовища, а й створює комфортне цифрове середовище як для внутрішніх користувачів, так і для зовнішніх кандидатів. Це забезпечує ефективне управління людськими ресурсами, підвищує продуктивність відділу кадрів та формує позитивний імідж компанії як роботодавця.

1.3 Аналіз сучасних веб-орієнтованих HRM-систем: переваги, недоліки, перспективи

У сучасному цифровому середовищі системи управління людськими ресурсами (HRM) відіграють ключову роль у забезпеченні ефективної роботи організацій. Розглянемо три провідні HRM-платформи, які мають значний вплив на ринку: SuccessFactors від SAP, Dayforce від Ceridian та Oracle Cloud HCM.

SuccessFactors є потужною хмарною платформою для управління людськими ресурсами, орієнтованою на повний цикл взаємодії з працівником – від рекрутингу до виходу на пенсію. У нашій HRM-системі не планується така глибока інтеграція з ERP-рішеннями, однак буде запозичено ідею

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

централізованого підходу до управління персоналом, з акцентом на оцінювання, розвиток та аналітику [1] (рис 1.1):

Посилання на платформу: <https://www.sap.com/ukraine/products/hcm.html>

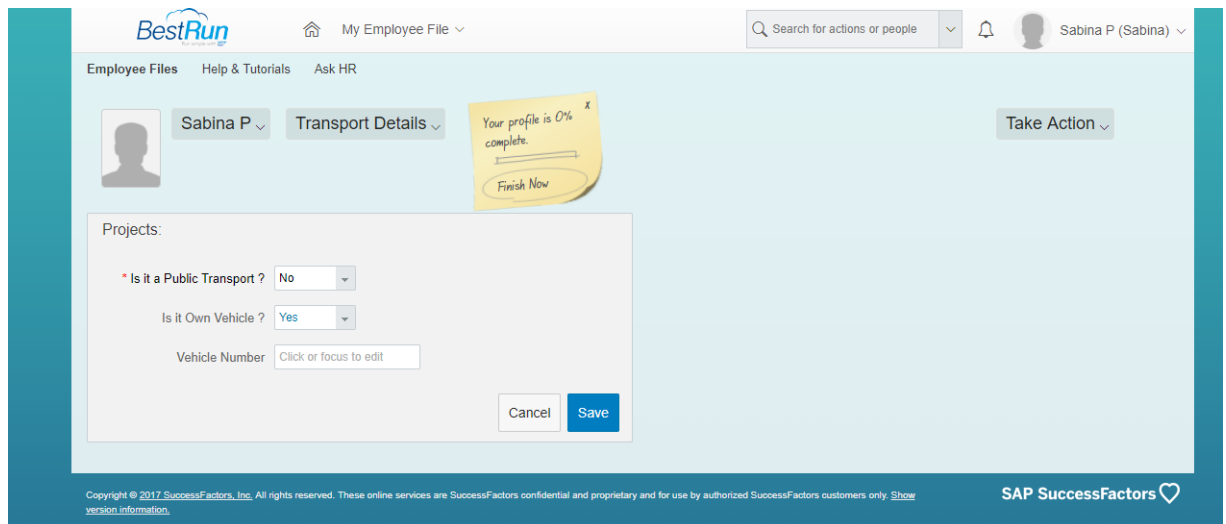


Рис. 1.1 - Інтерфейс профілю в SuccessFactors

Також SuccessFactors є хорошим прикладом гнучкої модульної структури, яку можна адаптувати під різні організаційні потреби, що є доцільним і для нашого рішення (рис.1.2):

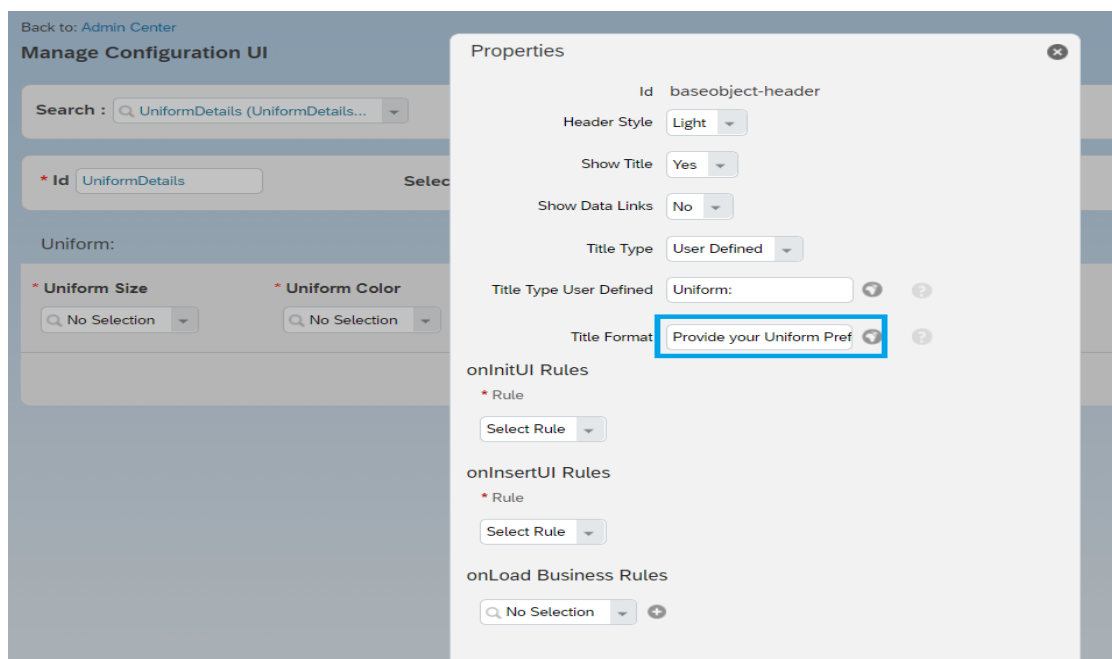


Рис. 1.2 – Налаштування інтерфейсу SuccessFactors

Одна з вагомих переваг SuccessFactors полягає в її тісній інтеграції з іншими продуктами екосистеми SAP. Завдяки цьому платформа сприяє підвищенню ефективності внутрішніх бізнес-процесів і забезпечує узгоджену

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

роботу різних відділів, особливо якщо в організації вже використовується ERP-рішення SAP. До того ж система має розгалужені можливості кастомізації — функціональні модулі можуть бути налаштовані з урахуванням особливостей конкретної організації, її структури та внутрішніх політик. Це дає змогу створити максимально персоналізоване HR-рішення.

Разом з тим SuccessFactors має і певні недоліки. Однією з основних проблем, яка часто згадується в оглядах користувачів, є висока вартість як впровадження, так і подальшого обслуговування системи. Це робить її менш доступною для малих і середніх компаній з обмеженим бюджетом. Крім того, початкове налаштування платформи може бути доволі складним і трудомістким процесом, що вимагає залучення технічних спеціалістів або консультантів.

Ще один виклик пов'язаний із залежністю від інфраструктури SAP. Щоб повністю розкрити потенціал SuccessFactors, компанія має використовувати й інші рішення цього виробника, що додатково збільшує загальні витрати. Також необхідно врахувати потребу в додатковому навчанні персоналу: інтерфейс і логіка платформи можуть виявитися незвичними для нових користувачів, тому повноцінне опанування функціоналу потребує часу і ресурсу.

Dayforce від Ceridian - це сучасна хмарна платформа для управління життєвим циклом працівників, яка поєднує функціонал HRM, розрахунку заробітної плати, кадрового обліку, тайм-менеджменту та планування змін у єдиному інтерфейсі [2]. У порівнянні з нашою HRM-системою, Dayforce вирізняється інтегрованим підходом до управління робочим часом та оплатою праці в режимі реального часу. Ми не плануємо реалізовувати настільки глибоку інтеграцію із фінансовим обліком, однак ідея об'єднання ключових HR-функцій в єдиному середовищі буде частково використана (рис.1.3):

Посилання на платформу: <https://www.dayforce.com/>

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

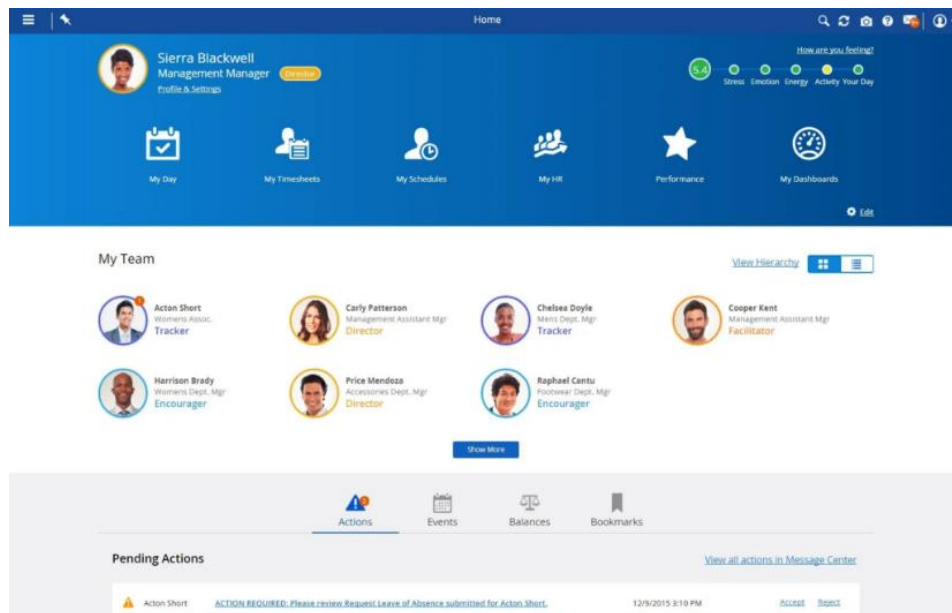


Рис. 1.3 – Інтерфейс профілю в DayForce

Крім того, гнучкий інтерфейс Dayforce та акцент на зручність користувача є орієнтиром для створення інтуїтивно зрозумілого UI у нашому додатку (рис. 1.4):

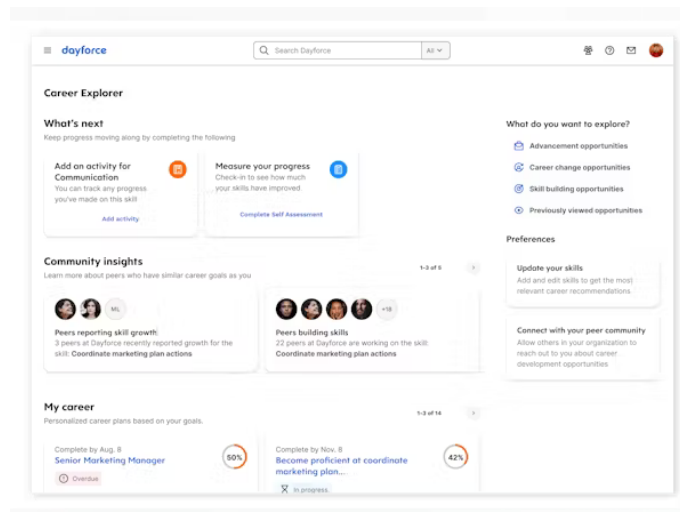


Рис. 1.4 – Інтерфейс одного із розділів в DayForce

Однією з важливих переваг Dayforce є можливість обробки та оновлення даних у режимі реального часу. Такий підхід дозволяє оперативно приймати управлінські рішення на основі актуальної інформації, а також миттєво реагувати на зміни у графіках роботи, відпустках, відсутностях або фінансових показниках. Високий рівень автоматизації в системі зменшує кількість ручної роботи HR-відділу та підвищує загальну продуктивність.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Аналітичні можливості Dayforce також заслуговують на увагу: платформа пропонує гнучкі дашборди, зручні для візуалізації важливих кадрових та фінансових метрик. Керівництво може отримувати уніфіковану статистику щодо навантаження персоналу, витрат на оплату праці, ефективності відділів тощо. Така звітність суттєво полегшує процес прийняття стратегічних HR-рішень.

Втім, як і в будь-якій складній системі, у Dayforce є свої недоліки. Зокрема, процес впровадження платформи у великій організації з багаторівневою структурою може бути довготривалим і ресурсоємним. До того ж, не всі функціональні можливості входять у базову комплектацію - деякі з них вимагають окремої оплати, що може стати критичним чинником для компаній з обмеженим бюджетом.

Oracle Cloud HCM - це комплексна хмарна платформа для управління людським капіталом, яка охоплює всі ключові аспекти HR: рекрутинг, адаптацію, управління талантами, компенсації, навчання, аналітику та прогнозування [3]. Система глибоко інтегрована з іншими модулями Oracle Cloud (ERP, Finance, CX), що забезпечує єдиний інформаційний простір для всього підприємства (рис.1.5):

Посилання на платформу: <https://www.oracle.com/human-capital-management/>



Рис. 1.5 – Інтерфейс меню OracleCloud HCM

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Oracle Cloud HCM пропонує глибоку автоматизацію процесів і широкі аналітичні можливості з використанням вбудованого штучного інтелекту (рис. 1.6):

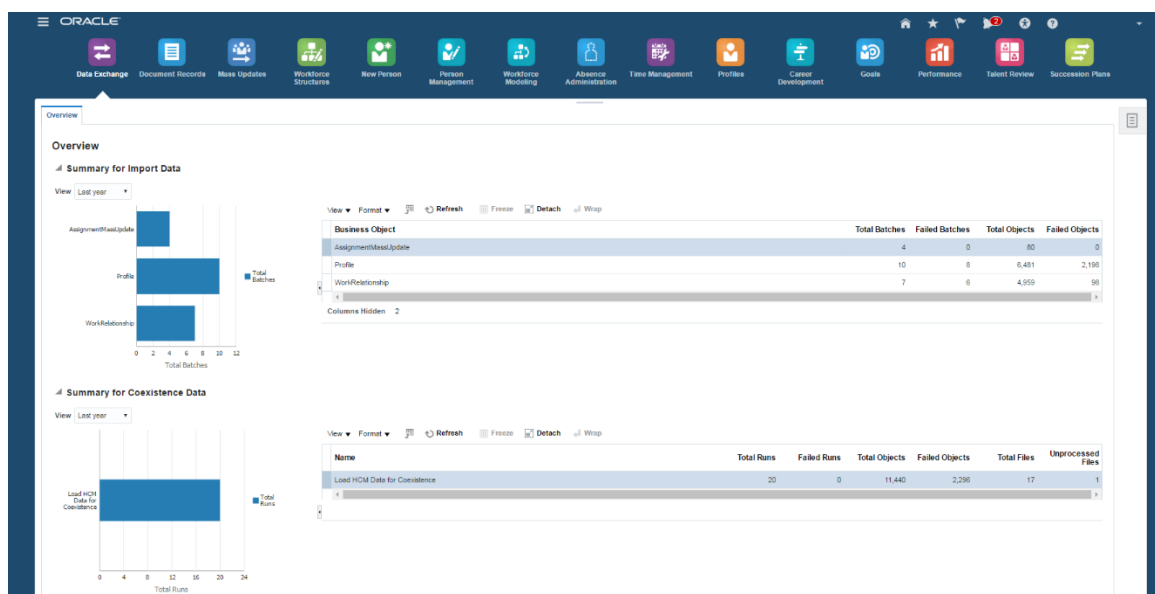


Рис. 1.6 – Інтерфейс статистичного аналізу OracleCloud HCM

Особливу увагу Oracle Cloud HCM приділяє використанню аналітики, зокрема на основі технологій штучного інтелекту. Інтелектуальні підказки, автоматизовані аналітичні моделі та алгоритми прогнозування дозволяють не лише аналізувати історичні дані, а й передбачати можливі сценарії розвитку HR-процесів. Це сприяє прийняттю обґрунтованих рішень щодо найму, утримання персоналу, формування команд та оптимізації ресурсів.

Гнучка модульна структура системи надає змогу адаптувати платформу під потреби конкретної організації, незалежно від її масштабу або галузі. Кожен модуль може бути впроваджений незалежно, що робить систему зручною для поетапної інтеграції [35]. Крім того, Oracle Cloud HCM відповідає найвищим стандартам безпеки, включно з вимогами міжнародного законодавства про захист персональних даних, що особливо важливо для компаній, які працюють у глобальному середовищі.

Але ця потужність платформи має і свої недоліки. Насамперед вартість володіння Oracle Cloud HCM є досить високою, що може бути критичним бар'єром для малого та середнього бізнесу. Більше того, впровадження системи

є складним процесом, який вимагає залучення кваліфікованих спеціалістів і може затягнутися у часі. Також варто зазначити, що для компаній з відносно простими кадровими процесами така система може виявитися надмірно функціональною - її потужний арсенал інструментів не завжди виправданий на практиці в невеликих командах.

Порівняння було здійснене у таблиці 1.1:

Таблиця 1.1

Характеристика	SuccessFactors (SAP)	Dayforce (Ceridian)	Oracle Cloud HCM
Оцінювання та розвиток персоналу	Так	Так	Так
Інтеграція з ERP	Тісна інтеграція з SAP ERP	Можлива інтеграція з іншими системами	Сумісність з продуктами Oracle
Гнучкість модульної структури	Так	Так	Так
Масштабованість	Висока	Висока	Висока
Індивідуальне налаштування	Так	Так	Так
Складність впровадження	Складна	Середня	Складна
Вартість	Висока	Середня	Висока
Додаткове навчання персоналу	Ні	Так	Так
Середовище	Хмарне	Хмарне	Хмарне

Жодна з розглянутих HRM-систем не покриває всіх необхідних потреб одночасно. Крім того, всі платформи є платними, складними у впровадженні та потребують додаткового навчання персоналу. У зв'язку з цим доцільно

розробити власне безкоштовне рішення, орієнтоване на потреби користувача з підтримкою базових функцій управління персоналом, оцінювання та аналітики. Вибравши найкращі характеристики кожної з систем, ми можемо отримати новий додаток, що знайде свою частку на ринку [5].

1.4 Постановка задачі дослідження та розробки HRM-системи

У сучасному веб-програмуванні, де зростають вимоги до масштабованості, гнучкості та швидкості розробки, архітектурні підходи відіграють ключову роль у проєктуванні ефективних систем. Один із найбільш поширених та традиційних підходів - це монолітна архітектура, яка й сьогодні залишається актуальною для ряду проєктів, зокрема у сфері веб-додатків.

Під терміном «моноліт» у веб-розробці розуміється програмне забезпечення, де всі функціональні частини (бекенд, фронтенд, база даних, API, логіка авторизації, управління сесіями тощо) об'єднані в єдину кодову базу та працюють як цілісна система. Веб-додаток із монолітною архітектурою зазвичай виконується на одному сервері або у вигляді єдиного процесу, де всі запити обробляються централізовано.

Цей підхід підходить для невеликих або середніх проєктів, які не потребують складної інфраструктури, або для стартапів, що прагнуть якнайшвидше вивести продукт на ринок. Завдяки єдиній структурі, налаштування середовища, відлагодження і розгортання значно спрощуються.

До основних переваг монолітної архітектури у веб-додатках належить, насамперед, можливість швидкого запуску проєкту. Розробку можна починати відразу, без необхідності налаштовувати окремі сервіси чи компоненти. Це значно знижує поріг входу для команди й дозволяє швидко протестувати базову функціональність. Також важливою перевагою є відносна простота у процесі розгортання: вся система розгортається як єдиний блок, тому відсутні труднощі, пов'язані з синхронізацією версій мікросервісів або з конфігурацією взаємозв'язків між ними. Централізований підхід до реалізації логіки системи полегшує процес налагодження і тестування, а також пришвидшує внесення змін

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

і оновлень. Ще однією суттєвою перевагою є спрощена взаємодія між модулями: оскільки всі компоненти працюють у спільному контексті виконання, вони можуть напряду викликати функції один одного, без складної мережевої комунікації чи обміну повідомленнями через API.

Водночас, монолітна архітектура має низку обмежень. Одним із ключових недоліків є складність масштабування - при зростанні навантаження на одну частину системи доводиться масштабувати весь застосунок, що не є ефективним з точки зору ресурсів. Крім того, зі збільшенням кількості функцій структура додатку ускладнюється, і це може призводити до втрати гнучкості й значного підвищення вартості підтримки. Ще однією проблемою є ризик впливу змін у одному з модулів на інші частини системи. Це ускладнює тестування, підвищує ймовірність помилок і може створювати непередбачувані ефекти при оновленні програмного забезпечення.

Нарешті, централізована структура ускладнює командну роботу: ведення паралельної розробки різними фахівцями над окремими модулями стає проблематичним, особливо якщо проєкт не має чітко визначеної модульної архітектури. З іншого боку, коли йдеться про гнучкість, масштабованість та модульність, усе частіше застосовують мікросервісну архітектуру [11]. Цей підхід є альтернативою традиційним монолітним системам і дозволяє будувати складні, динамічні проєкти, які легко адаптуються до змін.

Під мікросервісною архітектурою розуміють структуру, у якій додаток поділений на незалежні сервіси, кожен із яких виконує чітко визначену функцію - наприклад, обробку платежів, управління користувачами або облік товарів. Кожен сервіс має власну кодову базу, може розгортатися окремо, взаємодіє з іншими через API або черги повідомлень [28]. Це дозволяє командам працювати паралельно над різними частинами системи та оновлювати окремі модулі без впливу на всю програму.

Мікросервісна архітектура зазвичай застосовується у великих системах, де критично важливими є масштабованість, стійкість до збоїв та здатність швидко реагувати на зміни ринку. Її ключовою перевагою є можливість масштабувати

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

окремі компоненти системи незалежно: наприклад, якщо лише один із сервісів потребує більше ресурсів, можна розширити саме його, не торкаючись інших частин системи. Це дозволяє оптимально розподіляти навантаження та ефективніше використовувати інфраструктуру.

Ще однією перевагою мікросервісів є гнучкість у виборі технологій. Кожен сервіс може бути реалізований окремо, використовуючи різні мови програмування, фреймворки чи бази даних, що дає можливість адаптувати рішення до специфіки задачі. Крім того, така архітектура сприяє незалежній розробці й розгортанню: окремі команди можуть працювати автономно над своїми сервісами, пришвидшуючи процес розробки, спрощуючи оновлення й зменшуючи ризик конфліктів між модулями. Важливою перевагою є також підвищена стійкість до збоїв: якщо один сервіс виходить з ладу, інші залишаються працездатними, що мінімізує наслідки помилок.

Однак попри свої переваги, мікросервісна архітектура має і низку недоліків. Вона значно ускладнює інфраструктуру проєкту, оскільки потребує впровадження систем оркестрації, балансування навантаження, централізованого моніторингу, логування та автоматизованого керування версіями. Виникають також труднощі з налагодженням: взаємодія між сервісами вимагає чітко визначених API, захисту каналів передачі даних, а також врахування можливих мережових затримок. Тестування мікросервісів теж ускладнюється, адже кожен сервіс може мати залежності від інших, що ускладнює модульне та інтеграційне тестування. Нарешті, підтримка такої архітектури є витратною: зростає кількість середовищ, конфігурацій і розгортань, які потрібно адмініструвати, оновлювати та підтримувати в актуальному стані.

З огляду на вимоги до масштабованості, зручності підтримки та перспективи розвитку системи, було прийнято рішення використовувати мікросервісну архітектуру при розробці веб-додатку [10]. Такий підхід дозволяє структурувати систему на окремі, незалежні компоненти, що спрощує як розробку, так і супровід у майбутньому.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

У рамках обраної архітектури програмний продукт буде умовно поділено на три основні частини:

Фронтенд-сервіс - відповідатиме за інтерфейс користувача. Він реалізовуватиметься окремо та взаємодіятиме з іншими сервісами через API.

Бекенд-сервіс - оброблятиме бізнес-логіку, авторизацію, роботу з базою даних, а також забезпечуватиме взаємодію з іншими модулями.

Сховище файлів - окремий мікросервіс, призначений для збереження та обробки мультимедійних та службових файлів.

1.5 Обґрунтування вибору засобів та вимоги до апаратного забезпечення

У процесі вибору засобів реалізації бекенд-частини вебзастосунку було проаналізовано низку мов програмування. Враховуючи власний досвід, доступність документації та широку підтримку спільноти, перевагу було надано мові PHP [8]. Вона стабільно займає провідні позиції на ринку веброзробки, що свідчить про її актуальність та надійність у практичному застосуванні [23].

Серед фреймворків PHP першочергово було розглянуто Yii2 [24]. Його основною перевагою є швидкість створення стандартної адміністративної частини проєкту завдяки зручним інструментам генерації коду. Однак через жорстку структуру зв'язків між компонентами та оперування даними у вигляді масивів, Yii2 виявився недостатньо гнучким для реалізації складної API-логіки та подальшого масштабування.

Натомість фреймворк Symfony демонструє вищий рівень гнучкості, безпеки та масштабованості [7]. Його архітектура побудована на основі незалежних компонентів, що забезпечує зручну модульність і повторне використання коду. Symfony активно використовується у великих проєктах, зокрема в API-розробці, завдяки своїй стабільності, підтримці сучасних стандартів і можливості легко адаптуватися під потреби проєкту. Хоча він вимагає більше зусиль для вивчення, у довгостроковій перспективі це забезпечує

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

високу якість розробки та ефективне управління складними системами. Саме тому Symfony було обрано як основний фреймворк для реалізації серверної частини проєкту.

Під час розробки фронтенд-частини вебзастосунку постало питання вибору сучасного фреймворку, який би відповідав вимогам щодо продуктивності, зручності інтеграції з бекендом та простоти підтримки. Основними кандидатами для реалізації клієнтської логіки стали React та Vue 3 - найбільш популярні Javascript-фреймворки з активними спільнотами та розвиненою екосистемою.

React - бібліотека для створення інтерфейсів, яка забезпечує гнучкість та контроль над структурою додатка [15]. Завдяки широкій спільноті, великій кількості навчальних матеріалів і підтримці з боку компанії Meta, React давно зарекомендував себе як стабільне рішення для великих комерційних проєктів. Проте використання React передбачає роботу з додатковими інструментами (такими як Redux, React Router, тощо), що дещо ускладнює вхід для нових розробників і вимагає більше конфігурацій.

Vue 3, у свою чергу, є фреймворком з більш цілісною архітектурою [6]. Його основна перевага - простота синтаксису та легкість початку роботи. Він дозволяє будувати масштабовані та гнучкі компоненти, що значно спрощує підтримку складних інтерфейсів. Крім того, Vue забезпечує хорошу інтеграцію з серверними фреймворками, включно з Symfony, що стало додатковим аргументом на його користь. Тому, з огляду на поставлені завдання, Vue 3 було обрано як основний інструмент для реалізації фронтенд-частини.

Для реалізації мікросервісу, який відповідає за зберігання та обробку файлів у проєкті, було обрано використання Amazon S3 (Simple Storage Service) [25]. Цей сервіс дозволяє надійно зберігати великі обсяги даних у хмарі, забезпечуючи високу доступність, безпеку та масштабованість.

Вибір S3 був обумовлений кількома факторами. Amazon S3 є одним із найбільш надійних і перевірених рішень для зберігання даних на хмарних платформах, що надає можливість швидко масштабувати сховище відповідно до

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

зростаючих вимог проєкту. Також S3 підтримує великий обсяг даних, зручний механізм для управління доступом до файлів через використання політик безпеки та підтримку різних методів аутентифікації. Це дозволяє забезпечити надійний захист файлів і гарантувати доступ лише авторизованим користувачам або сервісам.

Для зберігання та обробки даних було обрано систему керування базами даних MySQL [9]. Цей вибір обумовлений високою популярністю та надійністю MySQL у веб-розробці, а також її здатністю ефективно працювати з великими обсягами даних при високих навантаженнях. MySQL є одним із найпоширеніших рішень для реляційних баз даних, забезпечуючи високу швидкість виконання запитів і простоту в адмініструванні. Крім того, MySQL підтримує масштабування, що дозволяє адаптувати систему до зростаючих вимог проєкту. Інтеграція з іншими компонентами системи також є важливим фактором на користь цього вибору.

Для спільного розгортання всіх компонентів проєкту буде використано Docker, що дозволить забезпечити ізольовану та стабільну середовище для кожного сервісу [21]. Використання контейнеризації дозволяє легко управляти залежностями, а також значно спрощує процес налаштування та розгортання проєкту на різних етапах його життєвого циклу. Docker забезпечить високу мобільність та портативність компонентів, дозволяючи зберігати їхню сумісність та швидко розгортати як у локальних середовищах розробки, так і в продуктивних системах.

Для коректної роботи HRM-системи на сервері користувача необхідно забезпечити відповідність мінімальним технічним вимогам. Сервер повинен мати щонайменше 20 GB вільного місця на SSD-диску, працювати під управлінням операційної системи Linux (наприклад, Ubuntu або CentOS), мати щонайменше 4 GB оперативної пам'яті (рекомендовано — 8 GB і більше) та процесор із мінімум чотирма ядрами, кожне з яких має частоту не нижче 2.0 GHz. Крім того, важливою є наявність стабільного інтернет-з'єднання зі швидкістю не менше 100 Mbps, а також статичної IP-адреси або підтримки динамічного DNS.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Серед необхідного програмного забезпечення слід зазначити веб-сервер Nginx, встановлену СУБД MySQL із підтримкою реплікації та бекапу, а також платформу Docker для контейнеризації застосунку. Для безпечної роботи в мережі необхідна підтримка захищених з'єднань за допомогою SSL-сертифікатів. Система має включати налаштовані механізми резервного копіювання даних і підтримувати інтеграцію з Amazon S3 для зберігання файлів.

Висновки до першого розділу

У першому розділі було проаналізовано завдання та існуючі системи управління персоналом (HRM). Порівнявши такі продукти, як SuccessFactors від SAP, Dayforce від Ceridian та Oracle Cloud HCM, було виявлено їхні ключові переваги та недоліки. SuccessFactors від SAP відзначається широкими можливостями для інтеграції з іншими бізнес-процесами, але має складність у налаштуванні та високі витрати на впровадження. Dayforce від Ceridian є потужним інструментом для автоматизації процесів управління персоналом, але його функціонал обмежений деякими специфічними ринками. Oracle Cloud HCM пропонує високу масштабованість і гнучкість, проте є менш зручним для малих та середніх підприємств через свою складність і високу ціну.

Під час вибору інструментів для реалізації нашої системи HRM було обрано декілька технологій, які забезпечать високу ефективність та масштабованість рішення. Для бекенд-розробки вибрано PHP з фреймворком Symfony, що дозволяє створювати надійні та безпечні веб-додатки, здатні обробляти великі обсяги даних. Для фронтенду обрано Vue 3, що забезпечить швидку розробку інтерактивного інтерфейсу користувача з гнучкими можливостями масштабування. Для зберігання файлів буде використовуватися Amazon S3, а для даних - MySQL завдяки її поширеності та достатньому функціоналу. Для контейнеризації додатків буде використано Docker, що дозволить забезпечити зручне розгортання системи на різних середовищах та оптимізує процеси оновлення і підтримки.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-ОРІЄНТОВАНОЇ HRM-СИСТЕМИ

2.1 Функціональні та нефункціональні вимоги до системи

Для забезпечення повного функціонування програмного продукту, відповідності сучасним стандартам, а також зручності користування, нижче наведено перелік ключових бізнес-, користувацьких та технічних вимог до HRM-системи [40]. Дані вимоги дозволяють досягти високого рівня продуктивності, масштабованості та зручності у використанні.

Бізнес вимоги

Основні цілі включають в себе створення функціональну систему управління людськими ресурсами для малого та середнього бізнесу, яка міститиме повний цикл рекрутингу, облік персоналу, оцінювання ефективності роботи, нарахування зарплати, а також зручний доступ до даних для співробітників [29].

Система реалізується у вигляді веб-додатку з адаптивним інтерфейсом, що дозволяє зручно працювати з нею через сучасні браузері на персональних комп'ютерах, ноутбуках і планшетах [12].

Передбачено, що архітектура рішення буде масштабованою, тобто такою, що легко адаптується до зростання кількості користувачів і обсягу даних без зниження швидкодії чи стабільності роботи.

Безпека при цьому відіграє ключову роль, тому уся інформація, зокрема персональні дані, має зберігатися у зашифрованому вигляді із застосуванням сучасних протоколів шифрування [38].

У системі передбачено три основні категорії користувачів, кожна з яких має свій набір вимог та рівень доступу. Адміністратор (Admin) володіє найвищими правами в системі: він має повний доступ до всіх модулів, може керувати налаштуваннями, створювати і редагувати ролі користувачів, переглядати логи активності, а також відповідати за параметри безпеки. Такий

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

рівень доступу дозволяє адміністратору підтримувати стабільність роботи системи та контролювати її захищеність.

Користувач із роллю HR відповідає за повний цикл управління персоналом. До його обов'язків входить створення та адміністрування вакансій, ведення бази кандидатів, організація етапів співбесід, виставлення рейтингів кандидатам, а також фіксація причин відмови. HR також має доступ до кадрових даних, може формувати графіки роботи, здійснювати облік заробітної плати, оцінювати ефективність працівників і створювати відповідні звіти.

Працівник (Worker) має доступ до власного профілю, де може переглядати та частково редагувати свої особисті дані. Йому також надається інформація про графік роботи, історію виплат заробітної плати, періоди лікарняних і відпусток. Крім того, працівник може ознайомитись з результатами оцінювання його роботи та завантажувати документи, такі як сертифікати, дипломи або інші підтверджуючі файли.

Власники (Stakeholders) або керівництво компанії потребують стратегічної інформації. Вони мають доступ до аналітики, яка включає перегляд статистики доходів, отриманих з внутрішніх покупок та реклами, а також графіки активності користувачів. Їм важливо бачити розподіл користувачів за країнами, пристроями, періодами активності, що дозволяє приймати управлінські рішення на основі реальних даних.

Характеристика об'єкта комп'ютеризації

Система HRM орієнтована на компанії, що прагнуть автоматизувати процеси управління персоналом, рекрутингу та кадрового обліку [19]. Основною цільовою аудиторією є HR-відділи, управлінці середньої ланки, а також працівники компанії, які повинні мати прозорий доступ до своїх даних, графіку, заробітної плати тощо.

Функціональні вимоги

Серед функціональних вимог до HRM-системи важливою є підтримка реєстрації та автентифікації користувачів із розмежуванням прав доступу відповідно до ролі - адміністратора, HR-фахівця або працівника. Система

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

повинна забезпечувати можливість створення і редагування вакансій, додавання кандидатів та побудову повноцінного процесу співбесід, що включає різні етапи відбору.

Також передбачено збереження оцінок кандидатів, що дозволяє ефективно проводити відбір і формувати обґрунтовані рішення щодо прийому на роботу. Для внутрішнього обліку реалізується ведення електронних карток співробітників, які містять інформацію про посаду, графік роботи та заробітну плату. Система повинна мати модуль розрахунку зарплати з урахуванням премій, надбавок, лікарняних та відпусток, що автоматизує значну частину бухгалтерських операцій.

Окремо впроваджується функціонал для оцінювання ефективності працівників з можливістю створення звітів. Не менш важливою є інтеграція з зовнішніми сервісами, зокрема Google Meet для організації онлайн-співбесід, а також можливість надсилати електронні повідомлення. Користувачі мають змогу завантажувати та переглядати документи, такі як сертифікати або дипломи. Завершальним елементом є система нотифікацій і зворотного зв'язку, яка дозволяє підтримувати ефективну комунікацію між кандидатами та HR-відділом.

Нефункціональні вимоги:

Серед нефункціональних вимог до розроблюваної HRM-системи важливе місце займає сприйняття користувачем. Система повинна бути інтуїтивно зрозумілою, із адаптивним інтерфейсом, що дозволить користувачу освоїти її за час не більше 20 хвилин. Для покращення взаємодії з користувачем усі основні дії повинні супроводжуватись підказками, нотифікаціями та повідомленнями про успішне чи некоректне виконання операцій.

Щодо надійності, система має гарантувати середній час безперервної роботи не менше 30 днів, а максимальний час простою під час оновлень не повинен перевищувати однієї години. У випадку виявлення критичних помилок, вони мають бути усунені протягом 24 годин з моменту виявлення.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

В аспекті продуктивності система повинна забезпечувати середній час відгуку на запити не більше 2 секунд, а також підтримувати стабільну роботу не менше ніж 500 одночасно активних користувачів. Усі дані, що зберігаються, мають регулярно копіюватися згідно з політикою щоденного резервного збереження, щоб уникнути втрати інформації у разі збою.

У плані експлуатаційної гнучкості HRM-система повинна легко масштабуватись - за рахунок можливості додавання нових модулів без необхідності повної перебудови архітектури. Крім того, проєкт передбачає обов'язкову наявність повноцінної користувацької документації та технічного опису для подальшої підтримки й розширення програмного забезпечення.

Аналіз вимог користувачів дав підстави для визначення варіантів використання майбутнього додатку. На рис. 2.1 наведено діаграму використання програмної системи:

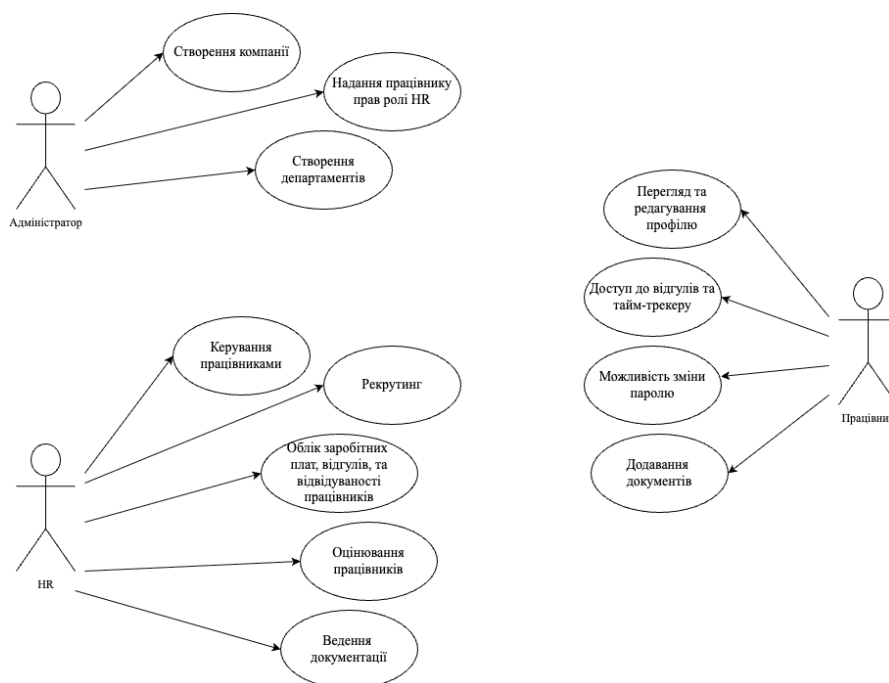


Рисунок 2.1 - Діаграма використання системи

На діаграмі [32] використання відображено основні сценарії взаємодії користувачів з системою відповідно до їх ролей: Адміністратор, HR, працівник. Кожна роль має доступ до специфічного набору функцій, що відповідає її рівню відповідальності та повноважень у системі. Такий розподіл ролей забезпечує гнучкість, контроль доступу та безпечну взаємодію з системою.

2.2 Архітектура та технологічний стек системи

Розроблена HRM-система побудована за принципами багаторівневої клієнт-серверної архітектури, що забезпечує логічне розділення презентаційного рівня, рівня бізнес-логіки та рівня зберігання даних. Такий підхід сприяє гнучкості, масштабованості та полегщує супровід системи.

Архітектурна модель

Архітектурна модель HRM-системи побудована за принципом чіткої розділеності обов'язків між її основними компонентами, що дозволяє досягти високої керованості, гнучкості та масштабованості.

Клієнтська частина реалізована за допомогою сучасного JavaScript-фреймворку Vue.js 3, що дає змогу створювати динамічний інтерфейс користувача у вигляді багатосторінкового застосунку. Уся взаємодія з сервером здійснюється через HTTP-запити до REST API, причому дані передаються у форматі JSON, що є загальноприйнятим стандартом у сучасній веб-розробці [13].

Серверна частина побудована на основі Symfony Framework і відповідає за обробку логіки програми. Саме тут реалізовано обробку всіх бізнес-процесів, управління даними та контроль доступу. Сервер приймає запити з клієнтської частини, обробляє їх згідно з бізнес-логікою та надсилає відповіді назад.

Для зберігання структурованої інформації (співробітники, департаменти, відпустки, оцінки, ролі тощо) використовується СУБД MySQL. Зв'язки між таблицями налаштовано відповідно до принципів реляційної моделі, що забезпечує цілісність і логічну узгодженість даних.

Система автентифікації реалізована на основі токенів JWT [27], що дозволяє забезпечити безпечну та масштабовану модель авторизації. Додатково підтримується OAuth2-авторизація через інтеграцію з Google API [26], завдяки чому користувачі можуть входити до системи, використовуючи облікові записи Google.

Для реалізації періодичних операцій, таких як автоматичне нарахування зарплати, використовуються фонові задачі (cron jobs), які налаштовуються у вигляді розкладу в межах серверного середовища.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

Розгортання системи здійснюється у контейнеризованому середовищі Docker, що значно полегшує процес деплоюменту, забезпечує однаковість середовищ на різних платформах і дозволяє гнучко масштабувати систему у випадку зростання навантаження.

Технологічний стек

У процесі розробки HRM-системи було використано сучасний набір інструментів і технологій, що забезпечують високу продуктивність, масштабованість і безпеку вебзастосунку. Клієнтська частина (frontend) реалізована за допомогою фреймворку Vue.js 3, що дозволяє створювати реактивні інтерфейси з динамічним оновленням даних. У поєднанні з HTML5, CSS3 та JavaScript це забезпечує адаптивну верстку та зручну взаємодію користувача з інтерфейсом.

Серверна частина (backend) побудована на мові програмування PHP версії 8.2 із використанням фреймворку Symfony, який відомий своєю гнучкістю, високою продуктивністю та зручною підтримкою REST-архітектури. Для обміну даними між клієнтською та серверною частинами використовується REST API, що дозволяє масштабувати систему та підключати зовнішні сервіси.

У якості системи керування базами даних обрано MySQL, що забезпечує швидке зберігання, пошук та обробку структурованих даних, необхідних для функціонування HRM-системи.

Для реалізації аутентифікації користувачів застосовуються два основні механізми: JWT (JSON Web Token) для авторизації на основі токенів та OAuth2 із підтримкою авторизації через Google API, що підвищує безпеку та зручність входу до системи.

Обробка фонових задач, таких як надсилання нотифікацій, архівація даних або регулярні звіти, реалізована через cron jobs і Symfony Schedule, що дозволяє запускати необхідні процеси відповідно до заданого розкладу.

Для зберігання документів, сертифікатів та інших файлів, які завантажують користувачі, використовується зовнішнє хмарне сховище AWS S3, що забезпечує високу доступність і надійність зберігання даних.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

З метою полегшення розгортання та перенесення проєкту між середовищами розробки, тестування й продакшн-інфраструктурою було використано Docker. Це забезпечило ізоляцію компонентів та стабільну роботу системи в різних середовищах.

Таким чином, обрана архітектура та використані технології забезпечують гнучкість, масштабованість і зручність підтримки розробленої HRM-системи. Чітке розділення на клієнтську та серверну частини, використання REST API [20] для взаємодії, а також застосування сучасних інструментів розгортання дозволяють ефективно реалізувати вимоги до функціональності, безпеки, та продуктивності.

2.3 Проєктування бази даних та модулів системи

У процесі створення HRM-системи було здійснено детальне проєктування бази даних та логічної модульної структури системи. Основною метою цього етапу є забезпечення надійного зберігання інформації, її структурованості, цілісності, та підтримки взаємозв'язків між даними.

Модульна структура системи

Модульна структура розробленої HRM-системи побудована на основі чітко визначених функціональних блоків, кожен з яких відповідає за реалізацію конкретного напрямку роботи з персоналом. Основу складає модуль користувачів, який забезпечує функції реєстрації, авторизації, а також управління ролями і правами доступу, формуючи гнучку модель взаємодії з системою. Модуль працівників відповідає за зберігання персональних даних співробітників, таких як ПІБ, посада, дата прийому на роботу, а також забезпечує прив'язку до відповідного підрозділу.

Функціональність управління організаційною структурою реалізована через модуль департаментів, де фіксується ієрархічна структура підприємства. Подача та обробка заявок на відпустки, лікарняні та інші форми відсутності реалізована у модулі відгулів, який забезпечує зручний механізм погодження та обліку. Облік робочого часу працівників, зокрема фіксація часу входу і виходу,

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

здійснюється через модуль відвідуваності, що є критично важливим для подальших розрахунків.

Розрахунок заробітної плати автоматизовано в модулі зарплатної відомості, який враховує оклади, надбавки, дні відсутності та фактичну відвідуваність. Робота з внутрішніми та зовнішніми файлами забезпечується за допомогою модуля документів, який дозволяє завантажувати і зберігати документи в хмарному сховищі Amazon S3. Загальна конфігурація системи, включаючи параметри безпеки, розклади фонових задач та інші системні налаштування, централізовано керується в межах адміністративного модуля.

Представлена модульна структура відображає логічне розділення системи на незалежні функціональні блоки, що дозволяє спростити розробку, тестування та супровід програмного забезпечення. Кожен модуль відповідає за окрему підсистему HRM-функціональності та взаємодіє з іншими модулями через чітко визначені точки входу й структури даних.

Структура бази даних

База даних HRM-системи спроектована таким чином, щоб забезпечити повне охоплення всіх функціональних напрямів управління персоналом. Основу її структури становить набір взаємопов'язаних сутностей, кожна з яких відповідає окремому аспекту кадрової роботи.

Інформація про організацію зберігається у структурі, що включає ключові атрибути компанії - її назву, адресу та сферу діяльності. Дані про користувачів, включно з автентифікаційними параметрами, ролями доступу та обліковими записами, організовані в окремому сегменті, що дозволяє реалізувати механізми контролю доступу до функціоналу відповідно до обов'язків.

Відомості про співробітників, включно з особистими даними, датою прийому на роботу, посадою та прив'язкою до структурного підрозділу, підтримуються у вигляді централізованої таблиці, що забезпечує зручну організацію обліку персоналу. Ієрархія та склад департаментів представлена в окремій сутності, яка дозволяє вибудувати чітку модель внутрішньої структури підприємства.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

Рекрутингова складова реалізована через зв'язку між інформаційними блоками про відкриті вакансії (із зазначенням назви посади, вимог і статусу) та кандидатів, що подають заявки. Для останніх фіксуються такі параметри, як ПІБ, контактні дані, прикріплені резюме та поточний етап розгляду.

Облік робочого часу реалізовано у вигляді окремого сховища, де реєструються записи про відвідуваність, запізнення, понаднормову роботу та графіки змін. Аналогічно, дані про відпустки, лікарняні й інші типи відсутності зберігаються у вигляді погоджених або запланованих відгулів із відповідними статусами розгляду.

Усі документи, що стосуються співробітників або кандидатів (сертифікати, договори, медичні довідки тощо), зберігаються у вигляді посилань на об'єкти в зовнішньому сховищі AWS S3. Це дозволяє оптимізувати обсяг основної бази та забезпечити безпечний доступ до файлів.

Загалом, структура бази даних орієнтована на підтримку стабільної та масштабованої роботи системи, дозволяючи ефективно організувати кадровий облік, документообіг, управління вакансіями та внутрішніми процесами в межах єдиного цифрового середовища. Таблиці бази даних охоплюють основні сутності предметної області та формують цілісну модель, що дозволяє ефективно зберігати, обробляти й аналізувати дані про персонал. Ретельно продумана структура забезпечує логічну послідовність у побудові запитів, підтримку цілісності зв'язків і можливість масштабування.

Діаграма структури бази даних

Нижче на рисунках зображено логічну модель бази даних у вигляді ER-діаграми (або UML-діаграми класів), яка відображає основні сутності (рис. 2.2), другорядні таблиці (рис. 2.3), а також їх атрибути та зв'язки між ними:

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

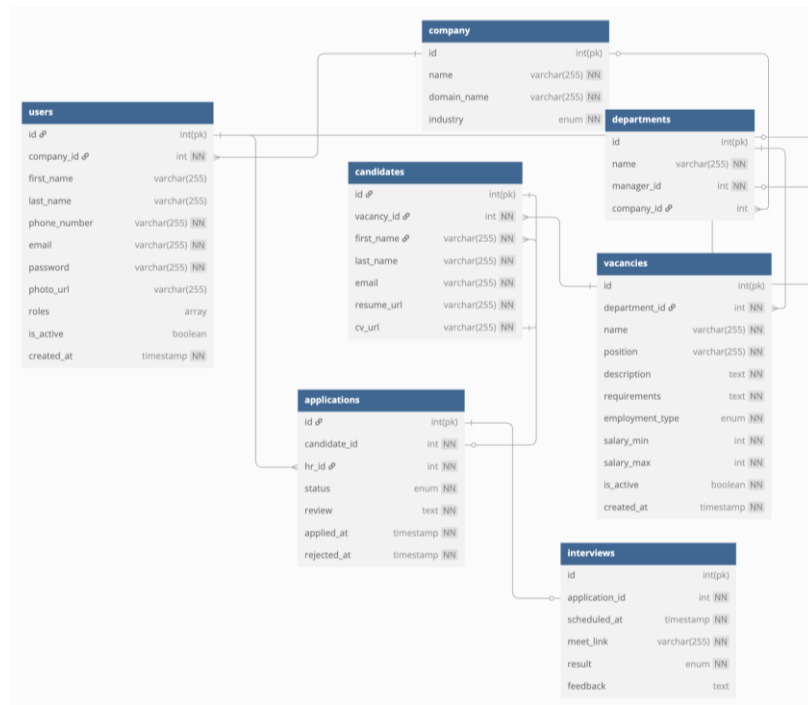


Рис. 2.2 - Діаграма структури базових таблиць бази даних

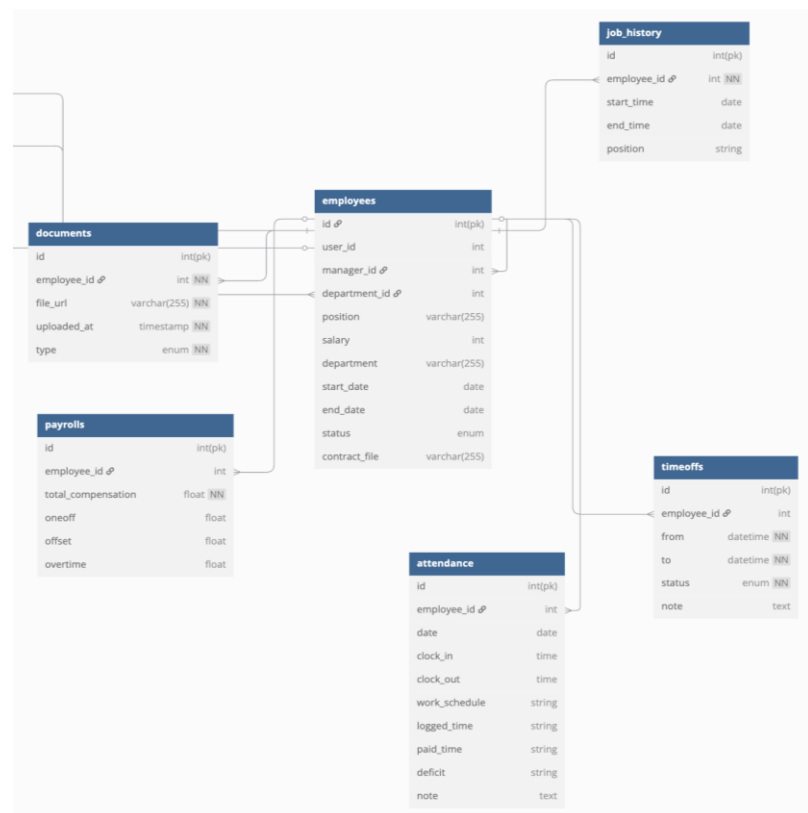


Рис. 2.3 - Діаграма структури допоміжних таблиць бази даних

Діаграма наочно ілюструє взаємозв'язки між таблицями бази даних, дозволяючи оцінити логіку побудови структури системи [16]. Вона є основою для реалізації бізнес-логіки на рівні програмного продукту та визначає спосіб взаємодії модулів із даними на практиці.

Діаграма наочно ілюструє взаємозв'язки між таблицями бази даних, дозволяючи оцінити логіку побудови структури системи [16]. Вона є основою для реалізації бізнес-логіки на рівні програмного продукту та визначає спосіб взаємодії модулів із даними на практиці.

У табл. 2.1 наведено опис таблиці users в базі даних, яка містить дані про користувачів системи: ім'я, прізвище, електронна пошта, номер телефону, пароль тощо.

Таблиця 2.1

Поле	Тип	Призначення
id	int	Первинний ключ
company_id	int	Унікальний ідентифікатор компанії
first_name	varchar(255)	Ім'я користувача
last_name	varchar(255)	Прізвище користувача
phone_number	varchar(255)	Номер телефону
email	varchar(255)	Електронна пошта
password	varchar(255)	Хеш паролю
photo_url	varchar(255)	URL до фотографії
roles	array	Ролі користувача
is_active	boolean	Активність акаунта
created_at	timestamp	Дата створення

У табл. 2.2 представлено таблицю candidates, що зберігає інформацію про кандидатів на вакансії, включаючи їхні імена, електронну пошту, резюме та CV.

Таблиця 2.2

Поле	Тип	Призначення
id	int	Первинний ключ
vacancy_id	int	Зв'язок з вакансією
first_name	varchar(255)	Ім'я кандидата
last_name	varchar(255)	Прізвище кандидата
email	varchar(255)	Електронна пошта
resume_url	varchar(255)	Резюме кандидата
cv_url	varchar(255)	CV кандидата

У табл. 2.3 описано структуру таблиці vacancies, яка містить дані про відкриті вакансії, їх опис, вимоги, рівень зарплати, тип зайнятості та статус.

Таблиця 2.3

Поле	Тип	Призначення
id	int	Первинний ключ
department_id	int	Зв'язок з департаментом
name	varchar(255)	Назва вакансії
position	varchar(255)	Посада
description	text	Опис вакансії
requirements	text	Вимоги
employment_type	enum	Тип зайнятості
salary_min	int	Мінімальна зарплата
salary_max	int	Максимальна зарплата
is_active	boolean	Чи активна вакансія
created_at	timestamp	Дата створення

У табл. 2.4 подано структуру applications, де зберігаються заявки кандидатів на вакансії, статуси заявок, відгуки HR та час подачі чи відхилення.

Таблиця 2.4

Поле	Тип	Призначення
id	int	Первинний ключ
candidate_id	int	Зв'язок з кандидатом
hr_id	int	HR, що обробляє заявку
status	enum	Статус заявки
review	text	Коментар
applied_at	timestamp	Дата подачі
rejected_at	timestamp	Дата відхилення

У табл. 2.5 представлено таблицю interviews, яка містить інформацію про інтерв'ю: дату, посилання на зустріч, результат та зворотній зв'язок.

Таблиця 2.5

Поле	Тип	Призначення
id	int	Первинний ключ
application_id	int	Зв'язок із заявкою
scheduled_at	timestamp	Час проведення
meet_link	varchar(255)	Посилання на зустріч
result	enum	Результат
feedback	text	Відгук

У табл. 2.6 подано опис таблиці documents, яка використовується для зберігання необхідних документів працівників: файли, типи документів і час завантаження.

Таблиця 2.6

Поле	Тип	Призначення
id	int	ІД запису (первинний ключ)
employee_id	int	ІД працівника
file_url	varchar(255)	Посилання на файл
uploaded_at	timestamp	Час завантаження
type	enum	Тип документу

У табл. 2.7 наведено структуру employees, де міститься розширена інформація про персонал: посада, зарплата, статус, відділ, менеджер, посилання на файл контракту, тощо.

Таблиця 2.7

Поле	Тип	Призначення
id	int	ІД запису (первинний ключ)
user_id	int	ІД користувача
manager_id	int	ІД менеджера
department_id	int	ІД відділу
position	varchar(255)	Посада
salary	int	Зарплата
department	varchar(255)	Назва відділу
start_date	date	Дата початку роботи
end_date	date	Дата завершення роботи
status	enum	Статус
contract_file	varchar(255)	Файл контракту

У табл. 2.8 описано таблицю job_history, яка фіксує історію змін посад працівників: попередні ролі, дати початку та завершення перебування на ній.

Таблиця 2.8

Поле	Тип	Призначення
id	int	ІД запису (первинний ключ)
employee_id	int	ІД працівника (унікальний)
start_time	date	Дата початку
end_time	date	Дата завершення
position	string	Посада

У табл. 2.9 подано структуру departments, що включає назви відділів, їхніх менеджерів та зв'язок із компаніями.

Таблиця 2.9

Поле	Тип	Призначення
id	int	ІД запису (первинний ключ)
name	varchar(255)	Назва відділу
manager_id	int	ІД менеджера (унікальний)
company_id	int	ІД компанії

У табл. 2.10 наведено опис таблиці company, де зберігаються назви компаній, їхні домени та сфера діяльності (індустрія).

Таблиця 2.10

Поле	Тип	Призначення
id	int	ІД запису (первинний ключ)
name	varchar(255)	Назва компанії
domain_name	varchar(255)	Домен компанії
industry	enum	Індустрія

У табл. 2.11 представлено таблицю timeoffs, яка фіксує відпустки, лікарняні, їхні дати, статуси й примітки.

Таблиця 2.11

Поле	Тип	Призначення
id	int	ID запису (первинний ключ)
employee_id	int	ID працівника (унікальний)
from	datetime	Дата початку
to	datetime	Дата завершення
status	enum	Статус запиту
note	text	Примітка

У табл. 2.12 описано структуру payrolls, що містить інформацію про зарплати працівників, компенсації, бонуси, надбавки та коригування.

Таблиця 2.12

Поле	Тип	Призначення
id	int	ID запису (первинний ключ)
employee_id	int	ID працівника
total_compensation	float	Загальна компенсація
oneoff	float	Разова виплата
offset	float	Коригування
overtime	float	Оплата за понаднормові

У табл. 2.13 подано структуру attendance, яка відповідає за облік робочого часу: вхід, вихід, графіки, зареєстрований час, оплату й дефіцити.

Таблиця 2.13

Поле	Тип	Призначення
id	int	ID запису (первинний ключ)
employee_id	int	ID працівника
date	date	Дата
clock_in	time	Час входу
clock_out	time	Час виходу
work_schedule	string	Робочий графік
logged_time	string	Зареєстрований час
paid_time	string	Оплачений час
deficit	string	Дефіцит часу
note	text	Примітка

2.4 Розробка користувацького інтерфейсу та логіки взаємодії

Одним із важливих аспектів створення HRM-системи є розробка зручного та інтуїтивно зрозумілого користувацького інтерфейсу, що забезпечує ефективну взаємодію користувача з системою. Якість інтерфейсу безпосередньо впливає на швидкість виконання повсякденних операцій, зменшення навантаження на користувача та загальну задоволеність від роботи з програмним забезпеченням.

Однією з ключових задач у реалізації HRM-системи є організація взаємодії з хмарними сервісами, зокрема AWS S3 для зберігання файлів та Google API для аутентифікації й надсилання повідомлень електронною поштою [14]. Для вирішення цієї задачі було реалізовано сервіси для завантаження та отримання файлів через AWS S3 і відправлення листів через Gmail API, з обробкою access tokenів.

Для реалізації логіки завантаження файлу в об'єктне сховище в нас використовується метод `uploadFile()`, який приймає файл, перевіряє його коректність та унікалізує ім'я, після чого передає файл до S3-сховища. У разі успішного завантаження повертається інформація про файл, що далі зберігається в базі даних. Ось представлення коду, що реалізує цю логіку:

```
public function uploadFile($file, string $bucket): array {
    if ($file && $file->isValid() !== null && $file->getSize() > 0) {
        $fileName = $file->getClientOriginalName();
        $serverFileName = md5(uniqid(rand(), true)) . '.' . $file-
            >getClientOriginalExtension();

        $file = fopen($file->getPathName(), 'r');
        if ($file) {
            $this->s3Client->putObject([
                'Bucket' => $bucket,
                'Key' => $serverFileName,
                'Body' => $file,
            ]);

            fclose($file);

            return [
                'Title' => 'File with name' . $serverFileName . ' uploaded
                successfully!',
                'FileName' => $serverFileName,
                'IsTrue' => true,
            ];
        }

        return [
            'Title' => 'File did not opened!',
        ];
    }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```
'IsTrue' => false,
];
```

Реалізація цього коду представлена в додатку Б.

Далі, метод `getPresignedUrl()` дозволяє сформувати тимчасове підписане посилання, за яким користувач може отримати доступ до приватного файлу без авторизації. Це важливо для безпеки – прямий доступ надається лише обмежений час:

```
public function getPresignedUrl(string $filename, string $bucket): ?string
{
    try {
        $cmd = $this->s3Client->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $filename,
        ]);

        $request = $this->s3Client->createPresignedRequest($cmd, '+20 minutes');
        return (string)$request->getUri();
    } catch (Exception $exception) {
        return (string)$exception;
    }
}
```

Детальна реалізація наведена в додатку А.

Наступним було реалізовано інтеграцію з Google API, зокрема з Gmail API, що дозволяє HRM-системі надсилати службові повідомлення електронною поштою від імені підключеного акаунта. Це може використовуватись для підтвердження дій, запрошень, повідомлень про зміну статусу тощо.

Для ініціалізації клієнта Google API відбувається передавання конфігураційних файлів, зокрема `client_id`, `client_secret` та `redirect_uri`. Встановлюється режим `offline`-доступу, що дозволяє надсилати листи навіть після завершення сесії користувача:

```
public function createClient(): Google_Client
{
    $client = new Google_Client();
    $client->setClientId($_ENV['GOOGLE_CLIENT_ID']);
    $client->setClientSecret($_ENV['GOOGLE_CLIENT_SECRET']);
    $client->setRedirectUri($_ENV['GOOGLE_REDIRECT_URI']);
    $client->setAccessType('offline');
    $client->setPrompt('consent');
    $client->setScopes([
        'https://www.googleapis.com/auth/gmail.send',
        'https://www.googleapis.com/auth/calendar.events',
    ]);

    return $client;
}
```

Повний код наведено в додатку Г.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Для відправки повідомлення використовується метод `sendEmail()`. Відповідний код наведено в додатку В. Ця функція перевіряє актуальність access токена користувача. Якщо термін його дії вичерпано, він оновлюється за допомогою refresh токена, і новий токен зберігається в базі даних. Повідомлення кодується у форматі MIME та надсилається на зазначену електронну адресу через Gmail API.

Однією з базових функцій HRM-системи є можливість створення нового працівника з подальшим відображенням його у списку співробітників. Цей процес включає створення облікового запису користувача, збереження інформації про посаду, зарплату, дату початку роботи, а також прив'язку до існуючого відділу компанії. Додатково створюється запис у зарплатній таблиці, який фіксує стартову зарплату [39].

Запит на створення надсилається з клієнтської частини через форму додавання працівника, після чого обробка відбувається на сервері. Метод `invoke()` у класі `EmployeeCreateAction` реалізує цю логіку, оброблюючи отримані поля та зберігає нові сутності в базі даних:

```
$user = new User();
$user->setFirstName($userData['firstName'] ?? '');
$user->setLastName($userData['lastName'] ?? '');
$user->setPhoneNumber($userData['phoneNumber'] ?? '');
$user->setEmail($userData['email'] ?? '');
$plainPassword = $userData['password'] ?? 'defaultpass';
$user->setPassword(
    $this->passwordHasher->hashPassword($user, $plainPassword)
);
$user->setRoles(['ROLE_WORKER']);
$user->setActive(true);
$user->setCreatedAt((new DateTime())->format('Y-m-d H:i:s'));
```

Спочатку створюється новий користувач, де зберігаються основні поля, а пароль шифрується перед збереженням. Користувачу призначається базова роль `Worker`, і він одразу активується.

Після цього створюється сутність `Employee`, яка прив'язується до користувача. Тут вказується дата початку роботи, посада, зарплата та статус. Усі поля приходять з тіла запиту. Окремо виконується пошук відділу за переданою

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

змінною, і у випадку успішного знаходження, працівник прив'язується до відділу:

```
$employee = new Employee();
$employee->setUser($user);
$employee->setStartDate(new DateTime($data['startDate'] ?? 'now'));
$employee->setPosition($data['position'] ?? 'Default Position');
$employee->setSalary($data['salary'] ?? 0);
$employee->setStatus(EmployeeStatus::from(strtolower($data['status'] ??
'active')));

$departmentId = $data['departmentId'] ?? null;
$department = $this->departmentsRepository->find($departmentId);

if (!$department) {
    return new JsonResponse(['error' => 'Department not found'], 400);
}

$employee->setDepartment($department);
```

Останнім кроком створюється зарплатний запис, який відразу зберігає значення базової зарплати. Після цього всі об'єкти зберігаються в базу даних:

```
$payroll = new Payroll();
$payroll->setEmployee($employee);
$payroll->setTotalCompensation($employee->getSalary());

$this->entityManager->persist($user);
$this->entityManager->persist($employee);
$this->entityManager->persist($payroll);
$this->entityManager->flush();
```

Цей код подано в додатку Г.

Далі, нам потрібно вивести список працівників, які є у компанії. Для цього реалізовано кастомний провайдер EmployeeListProvider. Він отримує всіх працівників і залишає лише тих, які належать до компанії поточного авторизованого користувача:

```
public function provide(Operation $operation, array $uriVariables = [],
array $context = []): array
{
    $user = $this->security->getUser();

    if (!$user instanceof User || !$user->getCompany()) {
        return [];
    }

    $companyId = $user->getCompany()->getId();

    $allEmployees = $this->employeeRepo->findAll();

    $filteredMembers = array_filter($allEmployees, function (Employee
$employee) use ($companyId) {
        $department = $employee->getDepartment();
        $company = $department->getCompany();
        $roles = $employee->getUser()->getRoles() ?? [];
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return $company !== null
&& $company->getId() === $companyId
&& in_array('ROLE_HR', $roles, true);
});

return array_values($filteredMembers);
}

```

Реалізація цього провайдеру міститься в додатку Д.

На клієнтській стороні форма створення працівника представлена як модальне вікно `AddEmployeeModal.vue`. Код даного компоненту продемонстровано в додатку Е. Користувач заповнює поля форми, обирає відділ, дату початку роботи, і натискає «Create». Після цього надсилається запит до бекенду з усіма введеними даними:

```

const createEmployee = async () => {
  errorMessage.value = null
  try {
    await axios.post('/api/employee', {
      user: {
        firstName: firstName.value,
        lastName: lastName.value,
        email: email.value,
        phoneNumber: phoneNumber.value,
        password: password.value,
      },
      position: position.value,
      salary: salary.value,
      departmentId: extractId(selectedDepartment.value),
      joinDate: joinDate.value
    }, {
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json',
        'Accept': 'application/json'
      }
    })

    props.onCreated?.();
    closeForm()
  } catch (error) {
    console.error('Failed to create employee:', error)
    errorMessage.value = error.response?.data?.detail || 'Something went wrong.'
  }
}

```

Після успішного збереження співробітника, таблиця оновлюється автоматично. Виведення списку працівників реалізовано на окремій сторінці, де відображаються основні дані: ім'я, посада, пошта, відділ, та статус працівника. Інтерфейс реалізовано як інтерактивну таблицю з можливістю переходу до детального профілю кожного працівника.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожен рядок таблиці містить аватар користувача, повне ім'я, посаду, менеджера, відділ і статус, який можна змінити безпосередньо з таблиці. Для переходу на сторінку перегляду детальної інформації про співробітника використовується router-link:

```
<tr v-for="employee in filteredEmployees" :key="employee.id">
  <td class="avatar-td">
    
    <span class="user-avatar-name">{{ employee.user?.first_name }} {{
employee.user?.last_name }}</span>
  </td>
  <td>{{ employee.position }}</td>
  <td>{{ employee.managerEmail || '-' }}</td>
  <td>{{ employee.departmentName }}</td>
  <td class="status">
    <StatusSelect v-model="employee.status" :employee-id="employee.id" />
  </td>
  <td>
    <div class="account">
      <router-link :to="/dashboard/employee/detail/${employee.id}">
      <button class="account-action">
        
      </button>
      </router-link>
    </div>
  </td>
</tr>
```

Таблиця підтримує оновлення без перезавантаження: при зміні ролі, фільтра, або при створенні нового працівника – список автоматично оновлюється через виклик fetchEmployees(). Це дозволяє динамічно підтягувати лише актуальні дані:

```
const fetchEmployees = async () => {
  try {
    const queryParams = {
      page: currentPage.value,
      itemsPerPage: itemsPerPage.value,
      ...filtersAndSortingParams()
    }

    const response = await axios.get('/api/employees', {
      headers: {
        Authorization: `Bearer ${token}`,
        Accept: 'application/json',
      },
      params: queryParams
    })

    const data = response.data

    totalPages.value = totalItems.value > 0
      ? Math.ceil(totalItems.value / itemsPerPage.value)
      : 1
  }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (error) {
      console.error('Error during rendering employees:', error)
      employees.value = []
      totalItems.value = 0
      totalPages.value = 1
    }
  }
}

```

Далі реалізовано систему фільтрації, сортування та пошуку, що дає змогу швидко знаходити потрібного працівника навіть у великій базі. Для цього на сторінці доступні поля вибору статусу, відділу, посади, а також текстове поле для пошуку за ім'ям, прізвищем або email.

```

function filtersAndSortingParams() {
  const params = {}

  if (searchQuery.value && searchQuery.value.trim() !== '') {
    params.search = searchQuery.value.trim()
  }

  if (selectedStatus.value && selectedStatus.value !== 'all') {
    params.status = selectedStatus.value
  }

  if (selectedDepartment.value && selectedDepartment.value !== 'all') {
    params['department.name'] = selectedDepartment.value
  }

  if (selectedJobTitle.value && selectedJobTitle.value !== 'all') {
    params.position = selectedJobTitle.value
  }

  if (sortField.value) {
    params[`order[${sortField.value}]`] = sortDirection.value
  }

  return params
}

```

Код компоненту EmployeeDashboard.vue наведено в додатку Є.

Усе це дозволяє гнучко управляти виведенням працівників, легко знаходити потрібну людину за кілька кліків, і значно підвищує зручність роботи HR-менеджера в системі.

У HRM-системі реалізовано функціонал обліку робочого часу співробітника, який дає змогу самостійно відзначати початок і завершення робочого дня. Користувач активує таймер натисканням кнопки Check In, після чого час початку фіксується в базі даних, а інтерфейс починає зворотній відлік. Після завершення робочого дня натисканням Check Out таймер зупиняється, дані

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

надсилаються на сервер і обчислюється фактична тривалість, оплачений час та можливий дефіцит.

На клієнтській стороні це реалізовано у вигляді окремої сторінки MyAttendance.vue, де відображається загальна інформація по тижню, таблиця записів із фільтрами за датами, та інтерактивний таймер, що відображає відлік часу після початку сесії. Реалізація даного компоненту наведена в додатку І.

```
async function clockIn() {
  try {
    const token = localStorage.getItem('access_token');
    const { data } = await axios.post('/api/attendance/start', {}, {
      headers: {
        Authorization: `Bearer ${token}`,
        Accept: 'application/json',
      }
    });

    isClockedIn.value = true;
    startTs = Date.now();
    localStorage.setItem('clock_in_ts', String(startTs));
    timer.value = '00:00:00';
    startTimer();
  } catch (error) {
    console.error('Failed to start timer:', error);
  }
}
```

Після натискання Check In, клієнт надсилає POST-запит до маршруту /api/attendance/start. У відповіді система створює новий запис відвідуваності з поточним часом. Для безперервного зворотного відліку зберігається мітка часу clock_in_ts у localStorage, що дозволяє відновити таймер навіть після перезавантаження сторінки.

```
public function __invoke(Request $request): JsonResponse
{
    $user = $this->security->getUser();
    $employee = $this->employeeRepo->findOneBy(['user' => $user]);

    if (!$employee) {
        return new JsonResponse(['error' => 'Employee not found'], 404);
    }

    $timezoneString = 'Europe/Kyiv';
    $timezone = new DateTimeZone($timezoneString);
    $now = new DateTimeImmutable('now', $timezone);

    $attendance = new Attendance();
    $attendance->setEmployee($employee);
    $attendance->setDate($now);
    $attendance->setClockIn($now);
    $attendance->setTimezone($timezoneString);

    $this->em->persist($attendance);
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

$this->em->flush();

return new JsonResponse([
    'id' => $attendance->getId(),
    'clockIn' => $attendance->getClockIn()->format('H:i'),
    'date' => $attendance->getDate()->format('Y-m-d'),
], 201);
}

```

На сервері у класі StartTimerAction створюється новий об'єкт Attendance, який фіксує дату, час початку та часовий пояс користувача. Код даного контроллера наведений в додатку Ж.

Коли користувач завершує роботу, натискаючи Check Out, відображається модальне вікно з підсумком дня та полем для коментаря. Після підтвердження надсилається POST-запит до /api/attendance/stop.

```

$clockOut = new DateTimeImmutable('now', $timezone);

$attendance->setClockOut($clockOut);

$loggedSeconds = max(0, $clockOut->getTimestamp() - $clockIn->getTimestamp());
$loggedFormatted = gmdate('H:i:s', $loggedSeconds);

$paidSeconds = min($loggedSeconds, self::EXPECTED_WORK_SECONDS);
$paidFormatted = gmdate('H:i:s', $paidSeconds);

$deficitSeconds = max(0, $loggedSeconds - self::EXPECTED_WORK_SECONDS);
$deficitFormatted = gmdate('H:i:s', $deficitSeconds);

$attendance->setLoggedTime($loggedFormatted);
$attendance->setPaidTime($paidFormatted);
$attendance->setDeficit($deficitFormatted);

```

У StopTimerAction сервер фіксує час завершення та обчислює загальну тривалість сесії, оплачений час (до 8 годин), та дефіцит (перевищення часу роботи). Результати записуються до бази даних, а відповідь повертається клієнту для оновлення таблиці. Реалізація даного класу представлена в додатку З.

На сторінці присутня таблиця відвідуваності, де відображаються всі записи робітника з деталями: дата, час входу та виходу, тривалість, оплачений час, надлишки. Реалізований кастомний провайдер MyAttendanceProvider, який фільтрує дані за робітником, що дав запит на виведення:

```

public function provide(Operation $operation, array $uriVariables = [],
    array $context = []): object|array|null
{
    $user = $this->security->getUser();
    $employee = $user->getEmployee();

    if (!$employee) {
        return [];
    }
}

```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return $this->attendanceRepo->findBy(['employee' => $employee]);
}
```

Відповідний код даного провайдеру також наведено в додатку И.

Для отримання списку відвідуваності робітника на клієнтській стороні надсилається GET-запит на маршрут /api/my-attendance:

```
async function fetchAttendance() {
  isLoading.value = true;
  const token = localStorage.getItem('access_token');
  const { data } = await axios.get('/api/my-attendance', {
    headers: {
      Authorization: `Bearer ${token}`,
      Accept: 'application/json',
    }
  });

  records.value = data.map(item => {
    const loggedSec = durationInSeconds(item.clock_in, item.clock_out);
    const paidSec = Math.min(loggedSec, 8 * 3600);
    const deficitSec = Math.max(0, loggedSec - 8 * 3600);
    return {
      id: item.id,
      logged: formatDuration(loggedSec),
      paid: formatDuration(paidSec),
      deficit: deficitSec === 0 ? '0m' : formatDuration(deficitSec),
      note: item.note || '-',
      loggedSec,
      paidSec,
      deficitSec
    };
  });
}
```

У відповіді сервер повертає масив об'єктів, де кожен елемент містить дату, час входу, час виходу та супутні поля. Далі дані обробляються, обчислюється тривалість перебування, оплачений час, надлишки або дефіцит. Отримані дані форматуються у години та хвилини для зручності виводу. Також додаються службові поля, наприклад schedule, які виводяться у відповідній колонці. Це дозволяє зробити таблицю більш наочною, а користувачеві – швидко оцінити своє навантаження за день. У разі помилки запит обробляється в try-catch, і записи у таблицю не виведуться.

Для виведення історії відвідуваності робітника у вигляді таблиці використано стандартний HTML-табличний шаблон із прив'язкою до змінної paginatedRecords. У таблиці відображається дата, час входу і виходу, запланована тривалість, опрацьований час, опрацьований період, дефіцит та коментарі працівника:

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<table class="attendance-table">
<thead>
<tr>
<th>Date</th>
<th>Clock In</th>
<th>Clock Out</th>
<th>Work Schedule</th>
<th>Logged Time</th>
<th>Paid Time</th>
<th>Deficit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr v-if="filteredRecords.length === 0"><td colspan="8" class="no-records">No records found</td></tr>
<tr v-else v-for="r in paginatedRecords" :key="r.id">
<td>{{ r.date }}</td>
<td>{{ r.clockIn }}</td>
<td>{{ r.clockOut || '-' }}</td>
<td>{{ r.schedule }}</td>
<td>{{ r.logged }}</td>
<td>{{ r.paid }}</td>
<td>{{ r.deficit }}</td>
<td>{{ r.note }}</td>
</tr>
</tbody>
</table>

```

На сторінці My Attendance реалізовано клієнтську фільтрацію записів відвідуваності за двома параметрами: датою початку та датою завершення. Користувач може задати діапазон дат, і система автоматично обмежує записи, які потрапляють у вибраний період. Це дозволяє зручно переглядати, наприклад, лише записи за поточний місяць чи тиждень:

```

const filteredRecords = computed(() => {
  const start = filters.value.start ? new Date(filters.value.start) : null;
  const end = filters.value.end ? new Date(filters.value.end) : null;
  return records.value.filter(rec => {
    const recDate = new Date(rec.date);
    if (start && recDate < start) return false;
    if (end && recDate > end) return false;
    return true;
  });
});

```

Фільтрація реалізована за допомогою обчислювального значення (computed). Якщо користувач не вибрав жодної дати, то обмеження не застосовуються, і виводяться всі записи. У випадку, якщо вибрано хоча б одну межу, система порівнює кожен запис за датою і виключає ті, які не потрапляють у діапазон. Весь цей механізм працює на клієнті, без додаткових запитів до

сервера, що дає змогу працювати з уже завантаженими даними швидко й без затримок.

Таким чином, реалізований модуль відвідуваності забезпечує працівникам зручний інструмент для фіксації початку та завершення робочого дня, збереження особистих нотаток, а також перегляду історії присутності із деталізацією по днях.

Висновки до другого розділу

У другому розділі було проведено повноцінний етап проектування HRM-системи, було здійснено глибокий аналіз предметної області, сформовано чіткі бізнес-цілі проєкту, серед яких автоматизація повного циклу управління персоналом, забезпечення прозорості взаємодії між працівниками та HR-відділом, створення єдиної бази кадрових даних, а також підвищення ефективності процесів рекрутингу, адаптації та оцінювання. Цей функціонал є необхідним для працівників управління людським ресурсом і виділятиме наш додаток з-поміж конкурентів на ринку.

На основі зібраних вимог було складено функціональну специфікацію системи, у якій передбачено реалізацію критично важливих модулів: рекрутингового (з керуванням вакансіями, кандидатами та етапами відбору), кадрового (з обліком графіків, відпусток і лікарняних), бухгалтерського (для управління нарахуванням зарплати), аналітичного (з рейтингами, оцінками ефективності та генерацією звітів), а також окремого функціоналу для працівників і кандидатів. Система також враховує ролі власників, які мають доступ до перегляду статистики, графіків активності та фінансової аналітики.

Таким чином, проєктна документація, створена в цьому розділі, сформувала повну основу для реалізації системи: від технічної архітектури до логіки ролей та механізмів доступу. Уся структура системи, передбачена в ході проєктування, повністю відповідає поставленим завданням і забезпечує можливість подальшої гнучкої розробки, масштабування та підтримки.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ HRM-СИСТЕМИ

3.1 Реалізація ключових компонентів системи

Відповідно до розробленої архітектури та структури модулів, було реалізовано основні компоненти HRM-системи, що охоплюють клієнтську частину, серверну логіку, базу даних та взаємодію з зовнішніми сервісами.

Інтерфейс системи реалізовано на основі фреймворку Vue.js з використанням компонентного підходу, що забезпечує високу модульність, масштабованість та зручність повторного використання елементів UI [31].

Серверна частина побудована на Symfony Framework, який реалізує обробку HTTP-запитів, аутентифікацію, авторизацію користувачів, взаємодію з базою даних MySQL, кешування та роботу з зовнішніми API (зокрема, Google API та AWS S3). Обмін даними між клієнтом і сервером відбувається через REST API, що дозволяє легко масштабувати систему, інтегрувати сторонні клієнти або мобільні додатки.

Усі сервіси системи контейнеризовані та розгортаються за допомогою Docker, що забезпечує стабільність середовища, спрощує деплоймент та підтримку інфраструктури. Для керування конфігураціями та мережею між контейнерами використовується Docker Compose.

На рис. 3.1 представлено діаграму розгортання HRM-системи, яка ілюструє структуру розміщення основних компонентів у середовищі виконання:

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

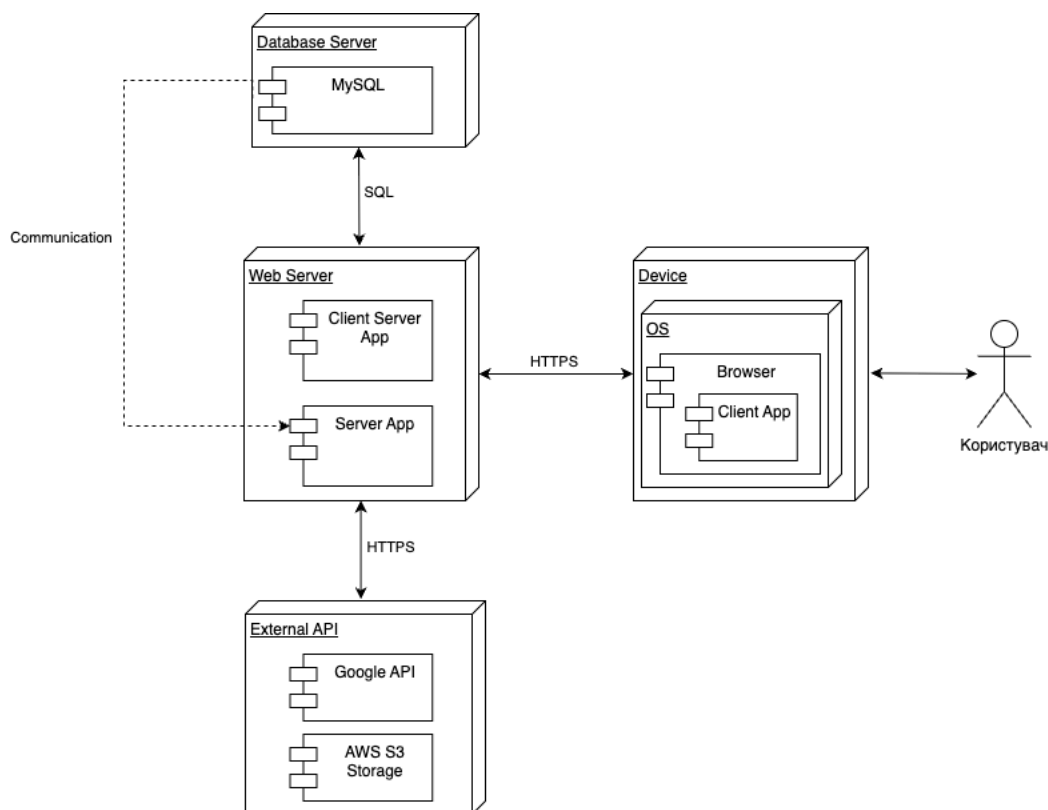


Рис. 3.1 - Діаграма розгортання

На діаграмі розгортання зображено взаємодію основних компонентів HRM-системи між собою та з користувачем [22]. Центральним елементом є вебсервер, на якому розгорнуто дві частини: клієнтський додаток та серверна частина, яка відповідає за обробку логіки, взаємодію з базою даних і зовнішніми сервісами.

Користувач отримує доступ до системи зі свого пристрою через браузер, де завантажується клієнтська частина застосунку. Усі запити до функціональних розділів - таких як перегляд працівників, запуск тайм-трекера чи робота з відгуками - надсилаються до бекенду через API-запити по HTTPS. Відповідно, вся взаємодія з даними та бізнес-логікою залишається на сервері, а клієнт лише відображає результати та ініціює дії.

Сервер додатку підключений до СУБД MySQL, яка зберігає всю інформацію про користувачів, співробітників, їхню присутність, розрахунки зарплат, структуру компанії тощо. Комунікація між сервером та базою даних відбувається за допомогою SQL-запитів, що забезпечує ефективну обробку запитів і збереження структурованих даних.

Окремо реалізовано модулі для взаємодії з зовнішніми сервісами. Зокрема, Google API використовується для авторизації користувачів через OAuth2, а також для надсилання повідомлень через Gmail. Крім того, інтеграція з AWS S3 забезпечує безпечне зберігання файлів користувачів, таких як документи, резюме або скановані довідки.

Усі зовнішні інтеграції реалізовано зі сторони серверної логіки, що дозволяє централізовано контролювати авторизацію, валідацію даних та забезпечувати високий рівень безпеки.

Крім основної логіки, було реалізовано систему ролей і прав доступу, яка дозволяє гнучко обмежувати функціональність відповідно до посадових обов'язків користувачів. Це забезпечує захист критичних даних та гарантує, що кожен працівник взаємодіє лише з тими модулями, що відповідають його рівню доступу.

Особливу увагу приділено масштабованості: завдяки контейнеризації, систему можна легко розгортати як в одному середовищі, так і в кластері з балансуванням навантаження. Такий підхід відкриває можливості для подальшого розширення системи без значної перебудови її архітектури.

У перспективі можлива інтеграція додаткових сервісів аналітики та BI-рішень для побудови дашбордів та звітів у режимі реального часу, що ще більше підвищить управлінську ефективність HRM-системи.

Загалом така архітектура дозволяє забезпечити чітке розділення відповідальностей, гнучку масштабованість та високу безпеку, що є критично важливим для корпоративної HRM-системи.

3.2 Тестування та забезпечення надійності роботи

У процесі розробки HRM-системи важливою частиною стала перевірка її стабільності, функціональності та відповідності вимогам. Тестування охоплювало як backend-логіку, так і інтерфейс користувача. Особливу увагу приділено перевірці коректної обробки API-запитів, валідації даних та контролю

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

доступу відповідно до ролей. Також реалізовано базові механізми захисту від збоїв, що забезпечують надійність роботи в реальному середовищі.

Інтерфейс HRM-системи є багатокomпонентним, однак завдяки компонентному підходу і розділенню за ролями, тестування кожної частини системи відбувалось окремо для ролей Admin, HR та Worker.

На головній сторінці HR бачить дашборд із ключовими метриками компанії, зокрема кількість співробітників, нових робітників, кандидатів на посади, та звільнених працівників (рис. 3.2).

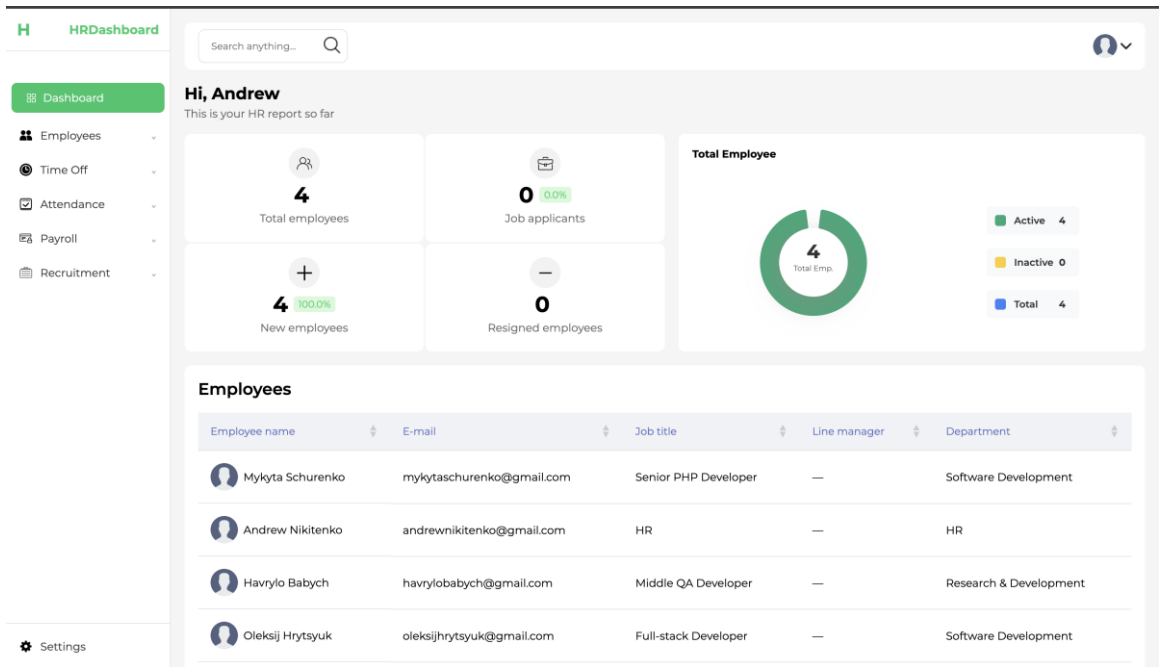


Рис. 3.2 - Головна сторінка

Переходячи на розділ Employee, HR може подивитись список робітників у таблиці, яку можна сортувати та фільтрувати за статусом, посадою та департаментом, або змінити статус працівника (рис. 3.3):

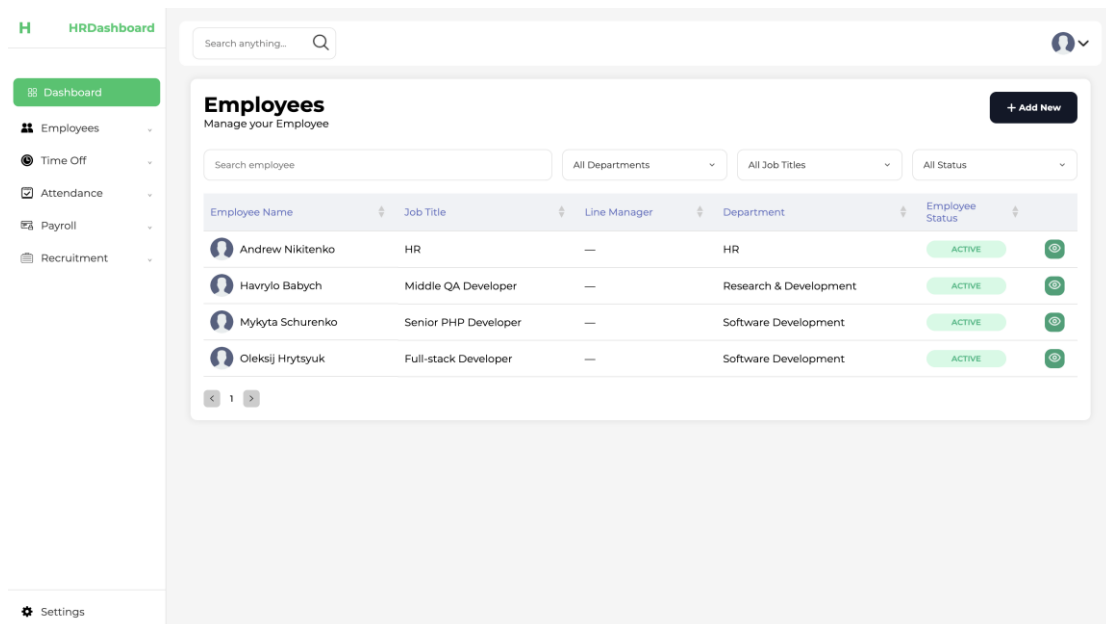


Рис. 3.3 - Таблиця зі співробітниками

Можна також переглянути детальні дані робітника, де можна побачити його ім'я та прізвище, пошту, номер телефону, дату народження, стать, національність, адресу тощо (рис. 3.4):

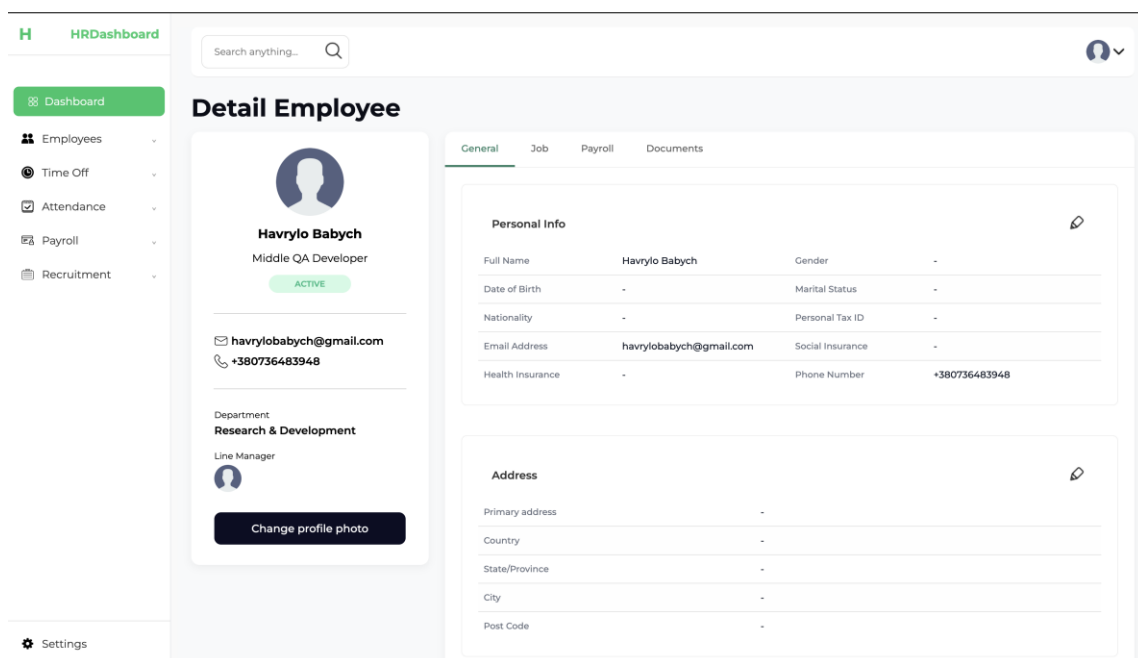


Рис. 3.4 - Деталізація робітника

Після натискання кнопки Add New відкривається модальне вікно для створення нового працівника. В цьому вікні реалізовано клієнтську валідацію всіх обов'язкових полів, а також передбачено перевірку на дублікати email-адрес (рис. 3.5):

Add New Profile

First Name *

Last Name *

Email Address *

Phone Number *

Password *

Position *

Salary *

Department *

▼

Start Date *



Рис. 3.5 - Додавання нового співробітника

В розділі My Time Off всі працівники можуть переглянути свої запити на додаткові вихідні дні, відфільтрувати їх за датами та статусом (рис. 3.6), а також подати новий запит на (рис. 3.7):

HRDashboard

Dashboard

Employees

Time Off

Attendance

Payroll

Recruitment

Search anything...

Request Time List

Manage your time off requests

+ Add New Request

Start date

-

End date

All Status

From	To	Total	Status	Action
26 Jul 2025	29 Jul 2025	4 Days	PENDING	View Delete

Рис. 3.6 - Таблиця з відображенням додаткових вихідних днів робітника

		Буланов О.П.			ИЗ.КР.Б – 121 – 25 – ИЗ	Арк.
		Толстой И.А.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

Add New TimeOff

Start Date *

DD MMM YYYY

End Date *

DD MMM YYYY

Note

Write description

Cancel

Create

Рис. 3.7 - Додавання нового запиту

В Employee Time Off HR може переглянути всі запити про відгул, відфільтрувати їх та змінити статус відгулу (рис. 3.8):

H

HRDashboard

Dashboard

Employees

Time Off

Attendance

Payroll

Recruitment

Search anything...

🔍

👤

▼

Employee Time Off

Search employee

🔍

Date Range

Start date

📅

–

End date

📅

All Status

All Status

▼

EMPLOYEE NAME	FROM	TO	TOTAL	NOTE	STATUS
Andrew Nikitenko andrewnikitenko@gmail.com	26 Jul 2025	29 Jul 2025	4 Days	Wedding	PENDING
Mykyta Schurenko mykytaschurenko@gmail.com	26 Jun 2025	28 Jun 2025	3 Days	Birthday	PENDING

Рис. 3.8 - Всі запити працівників

На сторінці My Attendance кожен працівник має змогу здійснити вхід у робочий день натисканням кнопки Check In. З цього моменту запускається таймер, який відображає час, проведений у системі (рис. 3.9):

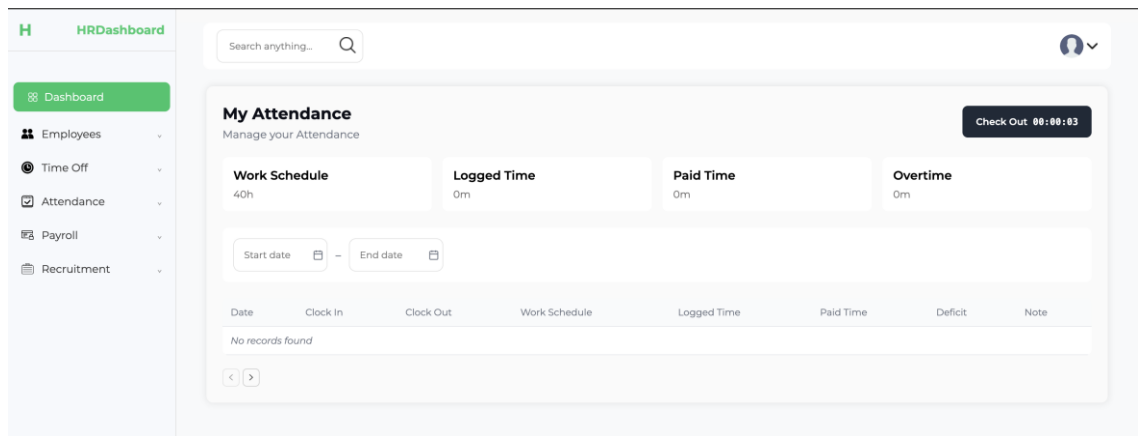


Рис. 3.9 - Початок роботи таймеру

При натисканні Checkout таймер зупиняється (рис. 3.10), і всі дані записуються в базу, після чого відображаються в табличному вигляді:

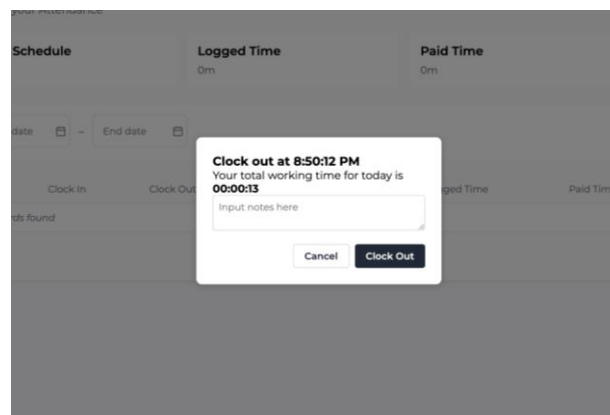


Рис. 3.10 - Закінчення роботи таймеру

У нижній частині сторінки відображається таблиця з усіма записами відвідуваності поточного працівника: дата, час входу, час виходу, загальний робочий час, оплачуваний час та дефіцит. Система також дозволяє фільтрувати записи за датою (рис. 3.11):

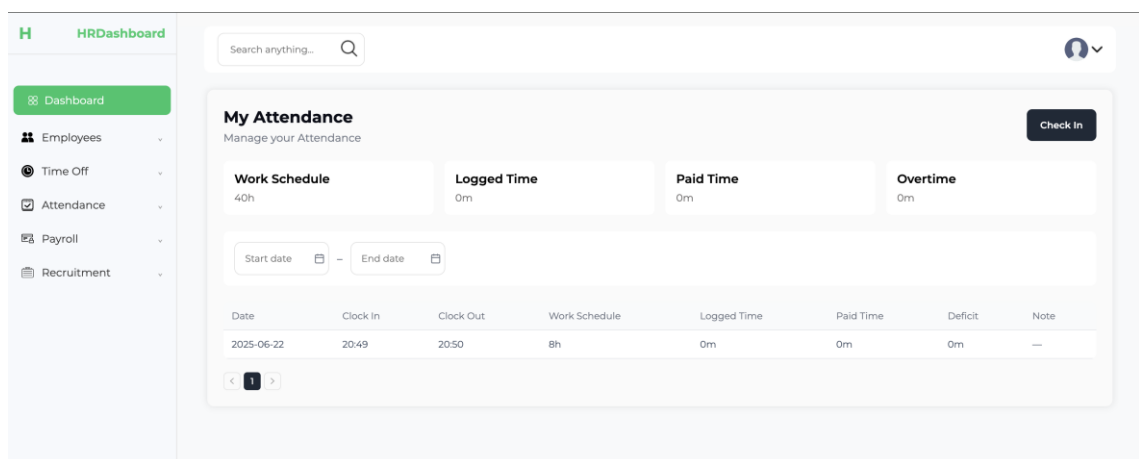


Рис. 3.11 - Усі записи відвідуваності працівника

На сторінці Employee Attendance HR може переглянути записи всіх присутностей працівників в таблиці. Доступна фільтрація за датами (рис. 3.12):

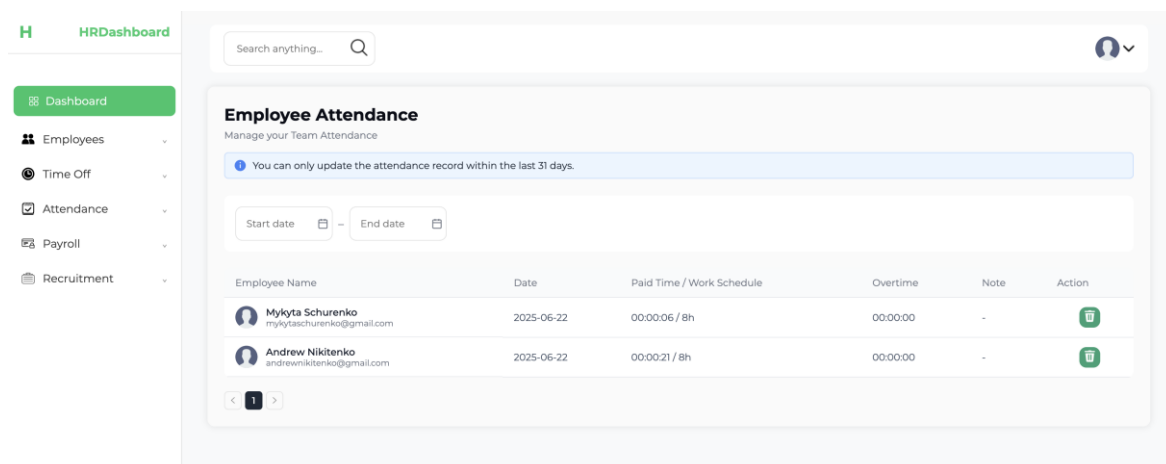


Рисунок 3.12 - Записи відвідуваності всіх працівників

Інтерфейс таблиці побудований у вигляді динамічного компонента, що автоматично підтягує актуальні дані через API. Також реалізовано відображення кількості годин за день, оплачуваних годин і нотаток, залишених працівником під час виходу.

На сторінці Employee Payroll представлена таблиця з базовою інформацією: ім'я працівника, заробітна плата, загальна компенсація, одноразові виплати, зміщення та надбавки за понаднормовий час. Передбачено також сортування за різними параметрами (рис. 3.13):

Employee Name	Salary	Total Comp	One-off	Offset	OT
Andrew Nikitenko	\$1,000	\$1,000	\$0	\$0	\$0
Mykyta Schurenko	\$5,000	\$5,000	\$0	\$0	\$0
Havrylo Babych	\$2,000	\$2,000	\$0	\$0	\$0
Oleksij Hrytsyuk	\$3,000	\$3,000	\$0	\$0	\$0

Рис. 3.13 - Зарплати працівників

Сторінка Payroll Settings дозволяє HR налаштувати базові параметри для формування зарплат: валюта (рис. 3.14), дата вирахування зарплати, відповідальний тощо (рис. 3.15):

Рис. 3.14 - Налаштування розрахунку зарплат

Рис. 3.15 - Налаштування розрахунку зарплат

Усі зміни зберігаються через форму з валідацією. Це забезпечує гнучке керування фінансовими параметрами, яке легко адаптується до змін у компанії.

На сторінці Jobs в розділі Recruitment HR може додавати нові вакансії на посади в компанії. Це дає змогу робити облік вакансій, де можна змінити статус вакансії (рис. 3.16):

Рис. 3.16 - Всі вакансії компанії

HR також може перейти до таблиці з кандидатами на вакансію, переглянувши основні дані про кандидата, такі як прізвище та ім'я, резюме, CV, дата створення профілю кандидата, також присутня можливість зміни статусу,

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

за якою кандидату відправиться повідомлення на пошту, і є можливість видалення профілю кандидата (рис. 3.17):

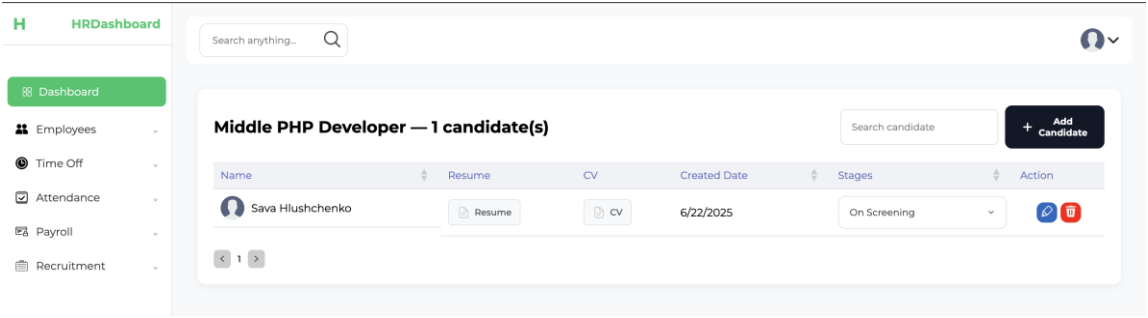


Рис. 3.17 - Кандидати на вакансію

Клікнувши на поле з кандидатом, HR може отримати детальну інформацію про кандидата, а також посилання на Google Meet з кандидатом, з зазначенням дати та часу (рис. 3.18):

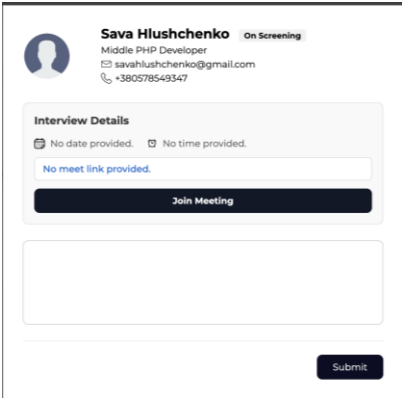


Рис. 3.18 - Детальна інформація про кандидата

В таблиці також можна віднайти кнопку для зміни даних про кандидата, при натисканні якої відкривається модальне вікно, в якому можна змінити ім'я, прізвище, номер телефону, пошту, CV, резюме, та задати дату інтерв'ю з кандидатом (рис. 3.19):

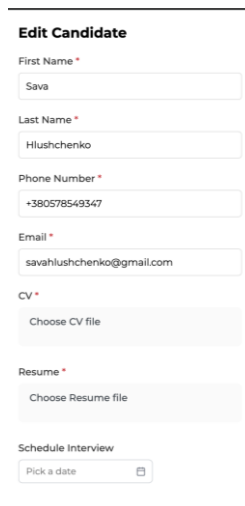


Рис. 3.19 - Зміна даних про кандидата

При натисканні кнопки Add New відкривається модальне вікно зі створенням нового кандидата, де можна вказати основні дані про нього: ім'я, прізвище, номер телефону, пошту, CV та резюме (рис. 3.20):

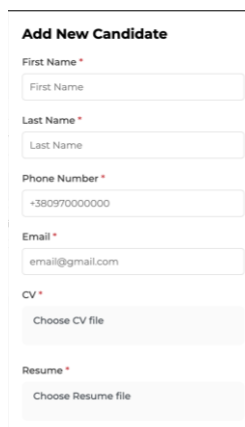


Рис. 3.20 - Додавання нового кандидата

На сторінці Candidates можна переглянути загальний список кандидатів, змінювати їхній статус, та видаляти їх (рис. 3.21):

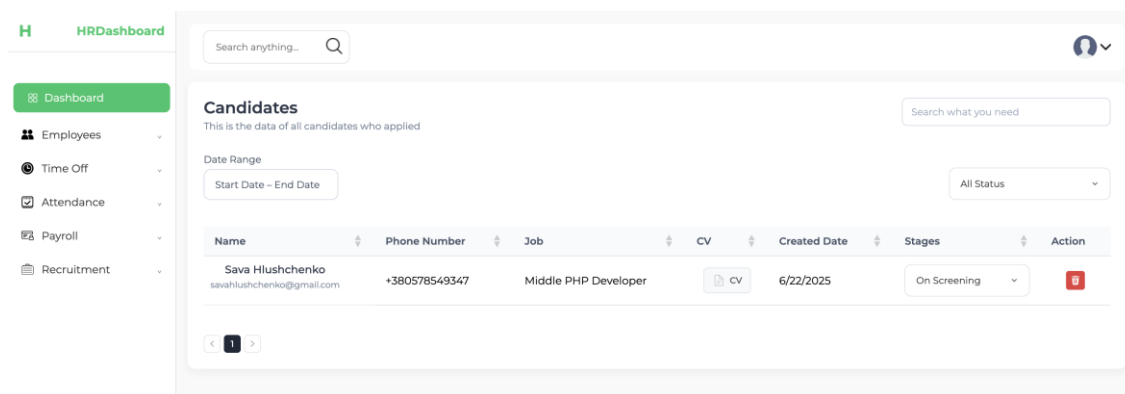


Рис. 3.21 - Усі кандидати в компанії

На сторінці Recruitment Settings можна створити нові e-mail templates, для того щоб відправляти повідомлення на пошти кандидатів автоматизовано, через зміну статусу (рис. 3.22):

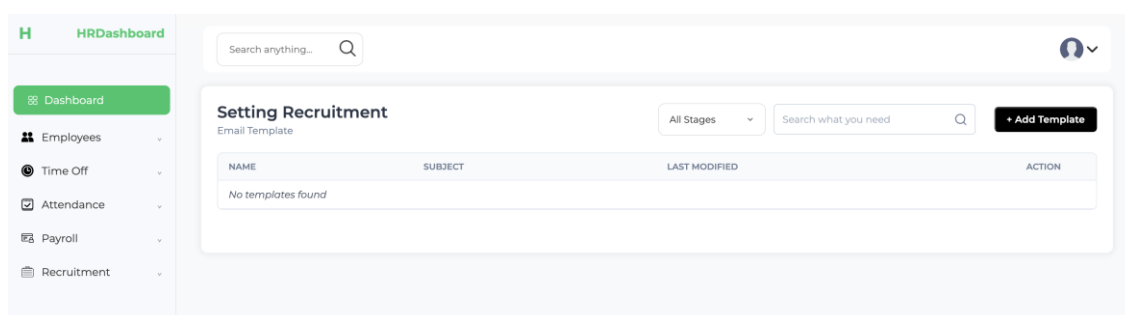


Рис. 3.22 - Налаштування рекрутингу

На сторінці Settings HR та працівник можуть змінити свої паролі до акаунтів (рис 3.23):

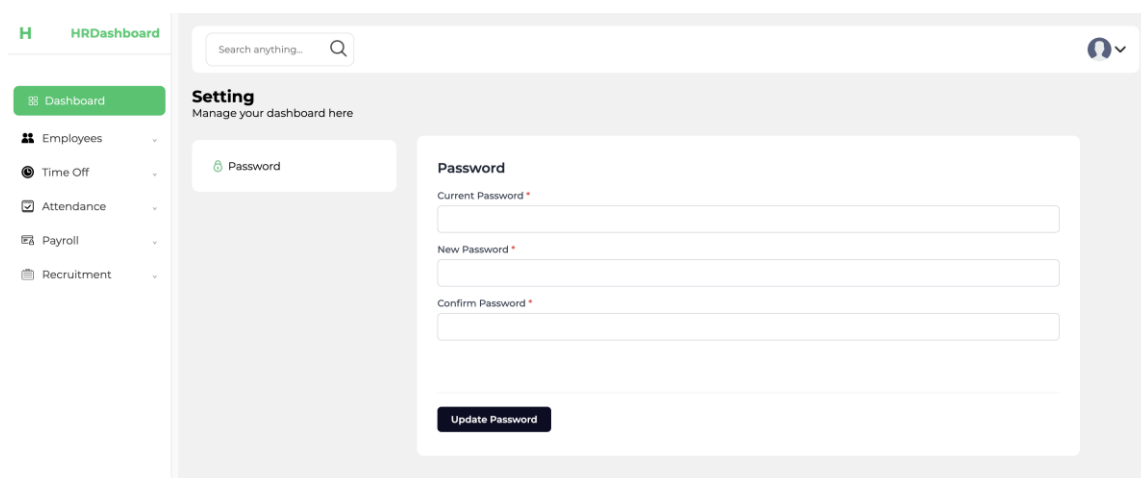


Рис. 3.23 - Зміна паролю від акаунту

Адміністратор, окрім зміни паролю від акаунту, може додавати нові департаменти для компанії (рис. 3.24):

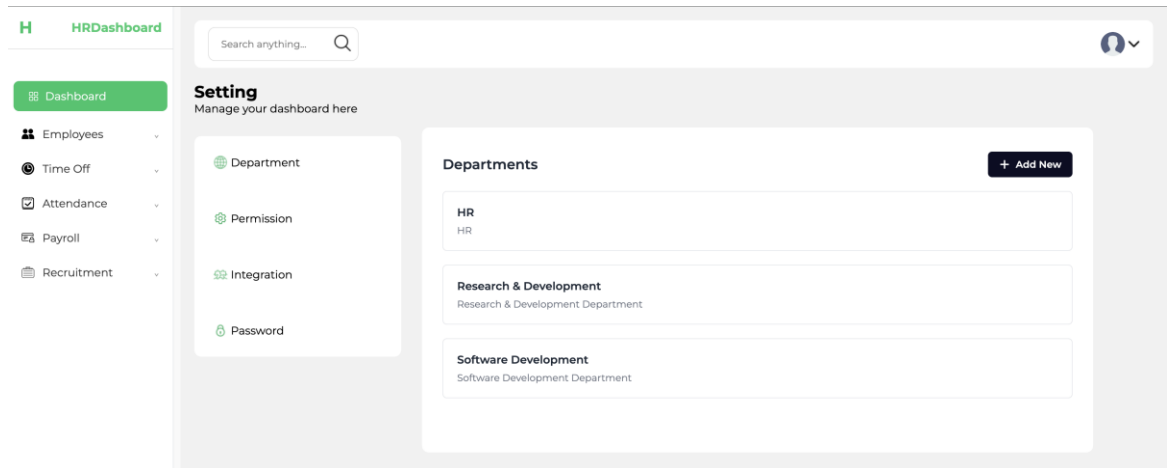


Рис. 3.24 - Додавання нових департаментів

Також адміністратор може додавати нового HR, просто вписавши ім'я, прізвище або пошту робітника в поле пошуку, та натиснувши на робітника для того, щоб надати йому права HR, або навпаки, видалити цього робітника із списку HR (рис. 3.25):

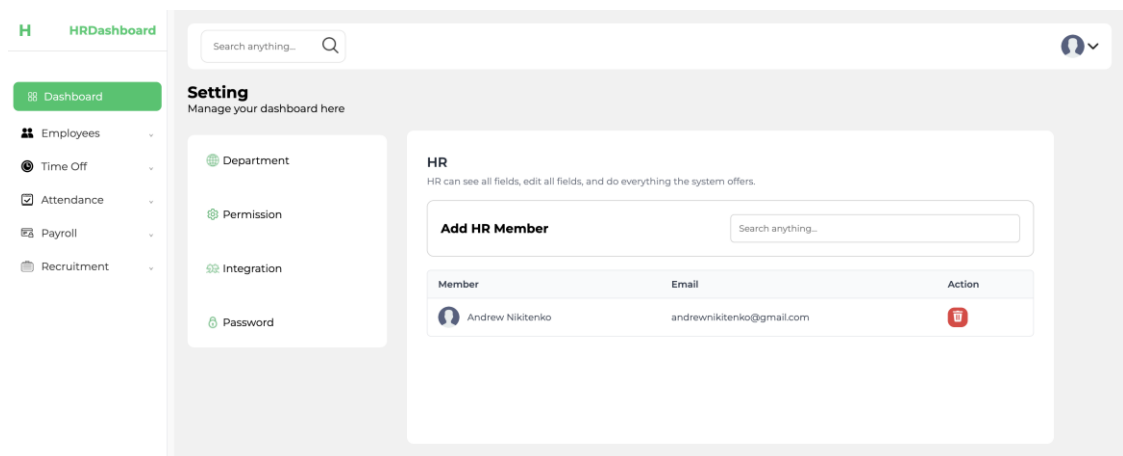


Рис. 3.25 - Додавання нового HR

Наостанок, адміністратор може прив'язати Google-акаунт компанії для певних дій, наприклад, для відправки повідомлень на пошту кандидата зі сторони акаунту компанії для автоматизації, або для створення Google Meet, а також для створення нових подій в Google Calendar (рис. 3.26):

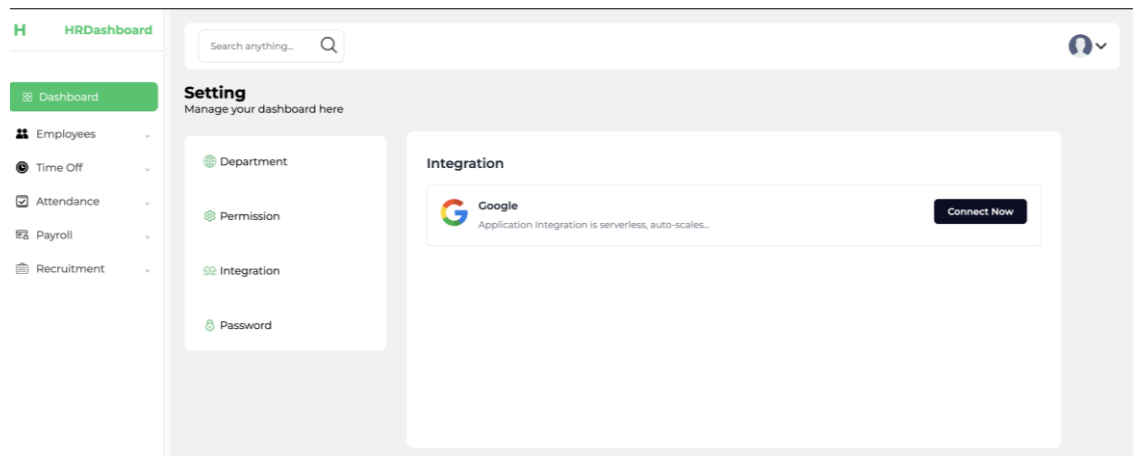


Рис. 3.26 - Інтеграція з Google

У процесі тестування були проведені перевірки ключових функцій системи, включаючи створення працівників, облік відвідуваності, нарахування зарплат, обробку фільтрів, а також інтерфейси рекрутингу та налаштувань. Основна мета - переконатися, що програма працює стабільно та передбачувано в типових сценаріях використання.

У процесі проведення функціонального тестування системи було виявлено низку критичних помилок [34], що мали значний вплив на безпеку, стабільність роботи та взаємодію з зовнішніми сервісами. Одна з найсерйозніших проблем стосувалася відсутності перевірки приналежності користувачів до конкретної компанії при виконанні запитів до бази даних. Це призводило до того, що HR-менеджер мав змогу переглядати інформацію про працівників інших організацій, що суперечить принципам ізоляції даних та створювало потенційний ризик витоку конфіденційної інформації.

Ще однією виявленою проблемою стала некоректна робота таймера на сторінці обліку відвідуваності. При переході користувача зі сторінки «My Attendance» у момент активної зміни, таймер зупинявся, хоча сама зміна продовжувалась. Після повернення таймер не відновлювався, що призводило до розбіжностей між очікуваним та фактичним часом зміни та могло вводити користувача в оману.

Крім того, виявлено критичну помилку в модулі зовнішньої інтеграції з Google API. У разі втрати чинності access token'а, система намагалась надіслати

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

електронне повідомлення без попереднього оновлення токєну, що завершувалося помилкою, яка, однак, не відображалась в інтерфейсі. Така поведінка ускладнювала діагностику несправностей і могла викликати непорозуміння серед користувачів. Внаслідок цього було прийнято рішення реалізувати механізм автоматичного оновлення токєнів перед виконанням запиту, а також повідомлення про помилки авторизації у користувацькому інтерфейсі. Загалом, під час тестування було виявлено низку критичних і незначних помилок, що стосувалися як логіки на сервері, так і обробки даних на клієнті. Після ідентифікації проблем кожен з дефектів було проаналізовано, зафіксовано та усунуто з відповідною перевіркою повторної появи.

Такий підхід до тестування дав змогу суттєво підвищити надійність роботи HRM-системи, покращити досвід користувача та забезпечити стабільну роботу основного функціоналу - від управління працівниками до роботи з Google API та контролю відвідуваності.

Висновки до третього розділу

У третьому розділі було реалізовано функціональність розробленої HRM-системи, а також продемонстровано її роботу з точки зору користувача. Особливу увагу було приділено інтерактивному інтерфейсу, який забезпечує зручність навігації та доступ до функцій відповідно до ролі користувача — адміністратора, HR-фахівця або працівника. Інтерфейс було реалізовано у вигляді вебзастосунку з адаптивною версткою, що гарантує сумісність із різними пристроями та дозволяє комфортно працювати як з комп'ютера, так і з планшета.

Особливу увагу в розділі приділено етапу тестування. Для оцінки стабільності та працездатності системи було застосовано ручне функціональне тестування відповідно до реальних сценаріїв використання. Виявлені на цьому етапі помилки — зокрема, некоректна обробка певних типів форм, дублювання записів або неправильна реакція на недійсні дані — були своєчасно усунені, що дало змогу забезпечити належну якість і стабільність

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

роботи застосунку. Тестування також підтвердило відповідність реалізованого функціоналу поставленим вимогам - як функціональним, так і нефункціональним.

Загалом, реалізація користувацької логіки, інтерфейсу та повноцінне тестування дозволили переконатися в готовності системи до використання. Сформовані модулі виконують свою функцію відповідно до специфікації, а гнучка структура забезпечує можливість подальшого розширення та масштабування рішення.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У межах кваліфікаційної роботи було реалізовано вебзастосунок HRM-системи, орієнтований на автоматизацію процесів управління персоналом у малих та середніх компаніях. Актуальність теми зумовлена необхідністю підвищення ефективності рекрутингу, кадрового обліку, оцінювання співробітників та організації внутрішньої взаємодії в умовах діджиталізації бізнесу та високої конкуренції на ринку праці.

У процесі виконання роботи було досягнуто поставленої мети - створено функціональну систему, що охоплює повний цикл керування людськими ресурсами - від пошуку кандидатів до моніторингу ефективності працівників. Система передбачає розмежування прав доступу для різних категорій користувачів (адміністратор, HR, працівник) і забезпечує інструменти для керування вакансіями, співбесідами, графіками, заробітними платами, внутрішніми документами та оцінками ефективності.

Розробка враховувала функціональні та нефункціональні вимоги: адаптивність інтерфейсу, масштабованість архітектури, надійність зберігання даних, безпеку персональної інформації, інтуїтивну зрозумілість та стабільну продуктивність. Вебзастосунок реалізовано із використанням сучасних підходів до розробки: об'єктно-орієнтованого програмування, REST API, компонентної архітектури, що забезпечує легкість підтримки й розширення функціоналу в майбутньому.

Таким чином, результати цієї роботи підтверджують доцільність застосування цифрових інструментів для оптимізації HR-процесів у корпоративному середовищі та мають практичну цінність для компаній, які прагнуть підвищити прозорість, ефективність і керованість роботи з персоналом.

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SuccessFactors. URL: <https://www.sap.com/ukraine/products/hcm.html>. (дата звернення: 17.05.2025).
2. DayForce. URL: <https://www.dayforce.com/>. (дата звернення: 13.05.2025).
3. Oracle Cloud HCM. URL: <https://www.oracle.com/human-capital-management/>. (дата звернення: 12.05.2025).
4. TechTarget: 18 must-have HR software features and system requirements. URL: <https://techtarget.com/searchhrsoftware/tip/15-must-have-HR-software-features-and-system-requirements>. (дата звернення: 06.05.2025).
5. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7th ed. Newtown Square, PA : Project Management Institute, 2021. 370 с.
6. Djirdeh H., Murray N., Lerner A. Fullstack Vue: The Complete Guide to Vue.js. Fullstack.io, 2021. 472 с.
7. Potencier F. Symfony 6: The Fast Track. Symfony SAS, 2022. 312 с.
8. Lockhart J. Modern PHP: New Features and Good Practices. O'Reilly Media, 2015. 270 с.
9. Seyed M. M. Learning MySQL. O'Reilly Media, 2021. 824 с.
10. Newman S. Building Microservices. O'Reilly Media, 2021. 600 с.
11. Newman S. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, 2020. 272 с.
12. Frain B. Responsive Web Design with HTML5 and CSS. Packt Publishing, 2022. 456 с.
13. Introducing JSON. URL: <https://www.json.org/json-en.html>. (дата звернення: 07.05.2025).
14. Wittig M., Wittig A. Amazon Web Services in Action. Manning Publications, 2023. 464 с.
15. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2023. 350 с.

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

16. Hernandez M. J. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. Addison-Wesley, 2020. 672 с.
17. Kavanagh M. J., Johnson R. D. Human Resource Information Systems: Basics, Applications, and Future Directions. SAGE Publications, 2020. 552 с.
18. Waddill D. Digital HR: A Guide to Technology-Enabled Human Resources. Routledge, 2018. 216 с.
19. What is a Human Resources Management System (HRMS)? URL: <https://www.techtarget.com/searchhrsoftware/definition/human-resource-management-system-HRMS>. (дата звернення: 19.05.2025).
20. Richardson L., Amundsen M., Ruby S. RESTful Web APIs. O'Reilly Media, 2013. 406 с.
21. Poulton N. Docker Deep Dive. TechWorld with Nigel, 2023. 400 с.
22. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2018. 208 с.
23. Lockhart J. Modern PHP: New Features and Good Practices. O'Reilly Media, 2015. 270 с.
24. Bogdanov A., Eliseev D. Yii2 Application Development Cookbook – Third Edition. Packt Publishing, 2016. 344 с.
25. Wittig M., Wittig A. Amazon Web Services in Action. Manning Publications, 2023. 464 с.
26. Richer J., Sanso A. OAuth 2 in Action. Manning Publications, 2017. 360 с.
27. Peyrott S. JWT Handbook: Build Secure APIs with JSON Web Tokens. Auth0, 2020. 112 с.
28. Geewax J. API Design Patterns. Manning Publications, 2022. 350 с.
29. Kavanagh M. J., Thite M. Human Resource Management Systems: Basics, Applications, and Future Directions for SMEs. Routledge, 2021. 368 с.
30. Oshero R. The Art of Unit Testing. Manning Publications, 2013. 320 с.
31. Rahy S., Bass J. M. Managing Non-Functional Requirements in Agile Software Development. IET Software, 2021. 18 с.

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

32. Wang H., Zhang B. Design and Implementation of Human Resource Information Systems Based on UML. International Journal of Computer Applications, 2020. 12 с.
33. García-Peñalvo M. T., Vázquez-Ingelmo E. Data Security and Privacy Concerns in Cloud-Based HRM Systems. Journal of Informatics Education and Research, 2024. 22 с.
34. Yu H. Design and Development of Human Resource Management Computer System. Journal of Healthcare Engineering, 2021. 15 с.
35. Sun Q., Wu T., Hua J. Design of Distributed Human Resource Management System of Spark Framework Based on Fuzzy Clustering. Journal of Sensors, 2022. 11 с.
36. Ahmad A. A.-S., Andras P. Scalability Analysis Comparisons of Cloud-Based Software Services. Journal of Cloud Computing, 2019. 14 с.
37. Hamdoon M. R., Ibrahim R. A. Securing Electronic Human Management System Using Compatible Information Security. Academy of Entrepreneurship Journal, 2021. 17 с.
38. Mastering Deployment Diagrams. URL: <https://numberanalytics.com/blog/uml-deployment-diagram/>. (дата звернення: 09.05.2025).
39. Deployment Diagram. URL: https://en.wikipedia.org/wiki/Deployment_diagram. (дата звернення: 09.05.2025).
40. Are “Non-Functional” Requirements Really Non-Functional? URL: <https://arxiv.org/abs/1607.06728>. (дата звернення: 23.05.2025).

ДОДАТКИ

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой І.А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А

Додаток А містить реалізацію класу FileGetter, що відповідає за генерацію тимчасового посилання (presigned URL) на файл, розміщений у хмарному сховищі Amazon S3. Клас є частиною сервісного шару та використовується для безпечного завантаження файлів через обмежений у часі URL.

```
<?php

namespace App\Service;

use Aws\S3\S3Client;
use Exception;

readonly class FileGetter
{
    public function __construct(private S3Client $s3Client)
    {
    }

    public function getPresignedUrl(string $filename, string $bucket): ?string
    {
        try {
            $cmd = $this->s3Client->getCommand('GetObject', [
                'Bucket' => $bucket,
                'Key' => $filename,
            ]);

            $request = $this->s3Client->createPresignedRequest($cmd, '+20 minutes');
            return (string)$request->getUri();
        } catch (Exception $exception) {
            return (string)$exception;
        }
    }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток Б

У Додатку Б наведено реалізацію класу FileUploader, що відповідає за обробку файлів, їх валідацію та завантаження до хмарного сховища Amazon S3. Завдяки цьому забезпечується інтеграція системи з файловою інфраструктурою.

```
<?php

namespace App\Service;

use Aws\S3\S3Client;

readonly class FileUploader
{
    public function __construct(private S3Client $s3Client)
    {
    }

    public function uploadFile($file, string $bucket): array {
        if ($file && $file->isValid() !== null && $file->getSize() > 0) {
            $fileName = $file->getClientOriginalName();
            $serverFileName = md5(uniqid(rand(), true)) . '.' . $file-
            >getClientOriginalExtension();

            $file = fopen($file->getPathName(), 'r');
            if ($file) {
                $this->s3Client->putObject([
                    'Bucket' => $bucket,
                    'Key' => $serverFileName,
                    'Body' => $file,
                ]);

                fclose($file);

                return [
                    'Title' => 'File with name' . $serverFileName . ' uploaded successfully!',
                    'FileName' => $serverFileName,
                    'IsTrue' => true,
                ];
            } else {
                return [
                    'Title' => 'File with name' . $fileName . ' is not uploaded!',
                    'IsTrue' => false,
                ];
            }
        }

        return [
            'Title' => 'File did not opened!',
            'IsTrue' => false,
        ];
    }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток В демонструє клас `GmailService`, який реалізує надсилення електронної пошти через Gmail API. Передбачено перевірку дії access token, його оновлення та формування повідомлення.

```
<?php

namespace App\Service;

use App\Entity\GoogleAccount;
use DateInterval;
use DateTime;
use DateTimeImmutable;
use Doctrine\ORM\EntityManagerInterface;
use Google_Service_Gmail;
use Google_Service_Gmail_Message;
use RuntimeException;

readonly class GmailService
{
    public function __construct(
        private GoogleClientFactory $googleClientFactory,
        private EntityManagerInterface $em,
    )
    {
    }

    public function sendEmail(GoogleAccount $account, string $to, string $subject,
        string $body): void
    {
        $client = $this->googleClientFactory->createClient();

        $now = new DateTime();
        $isExpired = $account->getExpiresAt() <= $now;

        if ($isExpired) {
            if (!$account->getRefreshToken()) {
                throw new RuntimeException('Token expired and no refresh token available');
            }

            $newToken = $client->refreshToken($account->getRefreshToken());

            if (!isset($newToken['access_token'])) {
                throw new RuntimeException('Failed to refresh access token');
            }

            $account->setAccessToken($newToken['access_token']);

            if (isset($newToken['refresh_token'])) {
                $account->setRefreshToken($newToken['refresh_token']);
            }

            $expiresIn = $newToken['expires_in'] ?? 3600;
            $account->setExpiresAt((new DateTimeImmutable())->add(new
                DateInterval("PT{$expiresIn}S")));

            $this->em->flush();
        } else {
            $client->setAccessToken([
                'access_token' => $account->getAccessToken(),
            ]
        )
    }
}
```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

'expires_in' => $account->getExpiresAt()->getTimestamp() - time(),
]);
}

$gmail = new Google_Service_Gmail($client);

$rawMessage = base64_encode(
    "To: $to\r\n" .
    "Subject: $subject\r\n" .
    "Content-Type: text/plain; charset=utf-8\r\n\r\n" .
    "$body"
);
$rawMessage = str_replace(['+', '/', '='], ['-', '_', ''], $rawMessage);

$message = new \Google_Service_Gmail_Message();
$message->setRaw($rawMessage);

$gmail->users_messages->send('me', $message);
}
}

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

У Додатку Г подано код класу GoogleClientFactory, який відповідає за створення та налаштування клієнтів Google API. Клас дозволяє працювати з поштою та календарем через авторизовані облікові записи.

```
<?php

namespace App\Service;

use App\Entity\GoogleAccount;
use DateInterval;
use DateTimeImmutable;
use Doctrine\ORM\EntityManagerInterface;
use Google_Client;
use RuntimeException;

readonly class GoogleClientFactory
{
    public function __construct(
        private EntityManagerInterface $em
    ) {}

    public function createClient(): Google_Client
    {
        $client = new Google_Client();
        $client->setClientId($_ENV['GOOGLE_CLIENT_ID']);
        $client->setClientSecret($_ENV['GOOGLE_CLIENT_SECRET']);
        $client->setRedirectUri($_ENV['GOOGLE_REDIRECT_URI']);
        $client->setAccessType('offline');
        $client->setPrompt('consent');
        $client->setScopes([
            'https://www.googleapis.com/auth/gmail.send',
            'https://www.googleapis.com/auth/calendar.events',
        ]);

        return $client;
    }

    public function createClientForAccount(GoogleAccount $account): Google_Client
    {
        $client = $this->createClient();

        if ($account->getExpiresAt() <= new \DateTimeImmutable()) {
            if (!$account->getRefreshToken()) {
                throw new RuntimeException('Token expired and no refresh token available');
            }

            $client->refreshToken($account->getRefreshToken());
            $newToken = $client->getAccessToken();

            $account->setAccessToken($newToken['access_token']);
            $account->setExpiresAt((new DateTimeImmutable())->add(new DateInterval('PT' .
                $newToken['expires_in'] . 'S')));
            $this->em->flush();
        } else {
            $client->setAccessToken([
                'access_token' => $account->getAccessToken(),
                'expires_in' => $account->getExpiresAt()->getTimestamp() - time(),
            ]);
        }
    }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return $client;
}
}
```

		Буланов О.П.			ИПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Толстой И.А.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

У Додатку Г реалізовано клас EmployeeCreateAction, що обробляє запити на створення працівника. У межах цього процесу створюється користувач, працівник, а також запис про заробітну плату.

```
<?php

namespace App\Action;

use ApiPlatform\Validator\ValidatorInterface;
use App\Entity\Employee;
use App\Entity\Payroll;
use App\Entity\User;
use App\Enum\EmployeeStatus;
use App\Repository\DepartmentsRepository;
use DateMalformedStringException;
use DateTime;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
use Symfony\Component\Serializer\Normalizer\AbstractNormalizer;
use Symfony\Component\Serializer\SerializerInterface;

readonly class EmployeeCreateAction
{
    /**
     * @param UserPasswordHasherInterface $passwordHasher
     * @param ValidatorInterface $validator
     * @param EntityManagerInterface $entityManager
     * @param SerializerInterface $serializer
     * @param DepartmentsRepository $departmentsRepository
     */
    public function __construct(
        private UserPasswordHasherInterface $passwordHasher,
        private ValidatorInterface $validator,
        private EntityManagerInterface $entityManager,
        private SerializerInterface $serializer,
        private DepartmentsRepository $departmentsRepository,
    )
    {
    }

    /**
     * @param Request $request
     * @return JsonResponse
     * @throws DateMalformedStringException
     */
    public function __invoke(Request $request): JsonResponse
    {
        $data = json_decode($request->getContent(), true);

        if (!is_array($data)) {
            return new JsonResponse(['error' => 'Invalid data'], 400);
        }

        $userData = $data['user'] ?? [];

        $user = new User();
        $user->setFirstName($userData['firstName'] ?? '');
    }
}
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

$user->setLastName($userData['lastName'] ?? '');
$user->setPhoneNumber($userData['phoneNumber'] ?? '');
$user->setEmail($userData['email'] ?? '');
$plainPassword = $userData['password'] ?? 'defaultpass';
$user->setPassword(
    $this->passwordHasher->hashPassword($user, $plainPassword)
);
$user->setRoles(['ROLE_WORKER']);
$user->setActive(true);
$user->setCreatedAt((new DateTime())->format('Y-m-d H:i:s'));

$this->validator->validate($user);

$employee = new Employee();
$employee->setUser($user);
$employee->setStartDate(new DateTime($data['startDate'] ?? 'now'));
$employee->setPosition($data['position'] ?? 'Default Position');
$employee->setSalary($data['salary'] ?? 0);
$employee->setStatus(EmployeeStatus::from(strtolower($data['status'] ??
'active')));

$departmentId = $data['departmentId'] ?? null;
$department = $this->departmentsRepository->find($departmentId);

if (!$department) {
    return new JsonResponse(['error' => 'Department not found'], 400);
}

$employee->setDepartment($department);

$payroll = new Payroll();
$payroll->setEmployee($employee);
$payroll->setTotalCompensation($employee->getSalary());

$this->entityManager->persist($user);
$this->entityManager->persist($employee);
$this->entityManager->persist($payroll);
$this->entityManager->flush();

return new JsonResponse([
    'id' => $employee->getId(),
    'firstName' => $user->getFirstName(),
    'lastName' => $user->getLastName(),
    'email' => $user->getEmail(),
    'phoneNumber' => $user->getPhoneNumber(),
    'salary' => $employee->getSalary(),
    'total_compensation' => $payroll->getTotalCompensation(),
], 201);
}
}

```

Додаток Д включає клас EmployeeListProvider, що забезпечує фільтрацію списку працівників за належністю до компанії поточного користувача. Клас використовується у провайдері API Platform.

```
<?php

namespace App\State;

use ApiPlatform\Metadata\Operation;
use ApiPlatform\State\ProviderInterface;
use App\Entity\Employee;
use App\Entity\User;
use App\Repository\EmployeesRepository;
use Symfony\Bundle\SecurityBundle\Security;

class EmployeeListProvider implements ProviderInterface
{
    public function __construct(
        private EmployeesRepository $employeeRepo,
        private Security $security
    ) {}

    /**
     * @param Operation $operation
     * @param array $uriVariables
     * @param array $context
     * @return array
     */
    public function provide(Operation $operation, array $uriVariables = [], array $context = []): array
    {
        $user = $this->security->getUser();

        if (!$user instanceof User || !$user->getCompany()) {
            return [];
        }

        $companyId = $user->getCompany()->getId();

        $allEmployees = $this->employeeRepo->findAll();

        $filteredMembers = array_filter($allEmployees, function (Employee $employee)
        use ($companyId) {
            $department = $employee->getDepartment();
            $company = $department->getCompany();
            $roles = $employee->getUser()->getRoles() ?? [];

            return $company !== null
                && $company->getId() === $companyId
                && in_array('ROLE_HR', $roles, true);
        });

        return array_values($filteredMembers);
    }
}
```

У Додатку Е міститься Vue-компонент `AddEmployeeForm.vue`, який відповідає за побудову форми додавання нового працівника. Компонент взаємодіє з API, виконує перевірки введених даних і відправляє їх на сервер.

```
<script setup>
import {defineEmits, onMounted, ref} from 'vue';
import CustomCalendar from '../smallComponents/CustomCalendar.vue';
import CustomSelect from '@components/smallComponents/CustomSelect.vue';
import axios from 'axios'

const props = defineProps({
  onCreated: Function
})

const emit = defineEmits(['close'])
const token = localStorage.getItem('access_token')

const joinDate = ref(null)
const firstName = ref('')
const lastName = ref('')
const email = ref('')
const phoneNumber = ref('')
const password = ref('')
const position = ref('')
const salary = ref(null)
const selectedDepartment = ref(null)

const departments = ref([])
const errorMessage = ref(null)

const closeForm = () => emit('close')

const fetchDepartments = async () => {
  try {
    const response = await axios.get(`/api/company-departments`, {
      headers: {
        Authorization: `Bearer ${token}`,
        Accept: 'application/json'
      }
    })
  } catch (error) {
    console.error('Error fetching departments:', error)
  }
}

departments.value = response.data.map(dep => ({
  label: dep.name,
  value: `/api/departments/${dep.id}`
}))

const extractId = iri => {
  const parts = iri.split('/');
  return parseInt(parts[parts.length - 1]);
};

const createEmployee = async () => {
  errorMessage.value = null
  try {
    await axios.post('/api/employee', {
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

user: {
  firstName: firstName.value,
  lastName: lastName.value,
  email: email.value,
  phoneNumber: phoneNumber.value,
  password: password.value,
}, {
  position: position.value,
  salary: salary.value,
  departmentId: extractId(selectedDepartment.value),
  joinDate: joinDate.value
}, {
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json',
    Accept: 'application/json'
  }
})

props.onCreate?.();
closeForm()
} catch (error) {
  console.error('Failed to create employee:', error)
  errorMessage.value = error.response?.data?.detail || 'Something went wrong.'
}
}

onMounted(fetchDepartments)
</script>

<template>
  <div class="new-employee-form">
    <button class="close-btn" @click="closeForm">
    <span class="arrow-back">
    
    </span>
    </button>
    <div class="form-header">
    <h2>Add New Profile</h2>
    </div>

    <div class="form-content">
    <div class="form-group">
    <label>First Name <span class="required">*</span></label>
    <input v-model="firstName" type="text" placeholder="First Name" class="form-control" />
    </div>

    <div class="form-group">
    <label>Last Name <span class="required">*</span></label>
    <input v-model="lastName" type="text" placeholder="Last Name" class="form-control" />
    </div>

    <div class="form-group">
    <label>Email Address <span class="required">*</span></label>
    <input v-model="email" type="email" placeholder="email@example.com" class="form-control" />
    </div>

    <div class="form-group">
    <label>Phone Number <span class="required">*</span></label>
    <input v-model="phoneNumber" type="text" placeholder="+380996669999" class="form-control" />

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</div>

<div class="form-group">
<label>Password <span class="required">*</span></label>
<input v-model="password" type="password" placeholder="Password" class="form-
control" />
</div>

<div class="form-group">
<label>Position <span class="required">*</span></label>
<input v-model="position" type="text" placeholder="Default Position"
class="form-control" />
</div>

<div class="form-group">
<label>Salary <span class="required">*</span></label>
<input v-model="salary" type="number" placeholder="1000" class="form-control"
/>
</div>

<div class="form-group">
<label>Department <span class="required">*</span></label>
<CustomSelect
:options="departments"
v-model="selectedDepartment"
placeholder="Select department"
/>
</div>

<div class="form-group">
<label>Start Date <span class="required">*</span></label>
<div class="date-input-wrapper">
<CustomCalendar
v-model="joinDate"
placeholder="DD MMM YYYY"
class="compact-calendar"
/>
</div>
</div>
</div>

<div class="form-actions">
<button class="cancel-btn button" @click="closeForm">Cancel</button>
<button class="create-btn button" @click="createEmployee">Create</button>
</div>
</div>
</template>

<style src="../../assets/css/components/AddEmployee.css" scoped></style>

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

У Додатку Є подано реалізацію компонента EmployeeDashboard.vue, що формує основний інтерфейс керування працівниками. Він охоплює таблицю зі списком, фільтрами, пагінацією та інструментами керування.

```
<script setup>
import StatusSelect from "@/components/smallComponents/StatusSelect.vue";
import CustomSelect from "@/components/smallComponents/CustomSelect.vue";
import {computed, onMounted, ref, watch} from 'vue'
import axios from 'axios'
import Dashboard from "@/components/Dashboard/Dashboard.vue"
import AddEmployeeModal from "@/components/ModalWindows/AddEmployeeModal.vue"
import HeaderComponent from "@/components/smallComponents/HeaderComponent.vue";

const isFormOpen = ref(false)
const employees = ref([])
const allEmployees = ref([])
const positionOptions = ref([
  { label: "All Job Titles", value: "all" }
])

const currentPage = ref(1)
const itemsPerPage = ref(8)
const totalItems = ref(0)
const totalPages = ref(1)

const openForm = () => isFormOpen.value = true
const closeForm = () => isFormOpen.value = false

const searchQuery = ref('')

const sortField = ref(null)
const sortDirection = ref('asc')

const toggleSort = (field) => {
  if (sortField.value === field) {
    sortDirection.value = sortDirection.value === 'asc' ? 'desc' : 'asc'
  } else {
    sortField.value = field
    sortDirection.value = 'asc'
  }
  fetchEmployees()
}

const positions = ref([])
const departments = ref([])
const selectedDepartment = ref('all')
const selectedJobTitle = ref('all')
const selectedStatus = ref('all')

const statusOptions = [
  { label: "All Status", value: "all" },
  { label: "active", value: "active" },
  { label: "inactive", value: "inactive" },
  { label: "terminated", value: "terminated" },
]

const token = localStorage.getItem('access_token')

const fetchDepartments = async () => {
  try {
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const response = await axios.get('/api/employee-departments', {
  headers: { Authorization: `Bearer ${token}` }
})
departments.value = [
  { label: 'All Departments', value: 'all' },
  ...response.data.map(dep => {
    return {
      label: dep,
      value: dep,
    };
  })
]
catch (error) {
  console.error('Error during rendering departments:', error)
}

const fetchPositions = async () => {
  try {
    const response = await axios.get('/api/employee-positions', {
      headers: { Authorization: `Bearer ${token}` }
    })
    positions.value = [{ label: 'All Job Titles', value: 'all' },
    ...response.data.map(pos => ({
      label: pos,
      value: pos,
    }))]
  } catch (error) {
    console.error('Error during rendering positions:', error)
  }
}

function filtersAndSortingParams() {
  const params = {}

  if (searchQuery.value && searchQuery.value.trim() !== '') {
    params.search = searchQuery.value.trim()
  }

  if (selectedStatus.value && selectedStatus.value !== 'all') {
    params.status = selectedStatus.value
  }

  if (selectedDepartment.value && selectedDepartment.value !== 'all') {
    params['department.name'] = selectedDepartment.value
  }

  if (selectedJobTitle.value && selectedJobTitle.value !== 'all') {
    params.position = selectedJobTitle.value
  }

  if (sortField.value) {
    params[`order[${sortField.value}]`] = sortDirection.value
  }

  return params
}

const fetchEmployees = async () => {
  try {
    const queryParams = {
      page: currentPage.value,
      itemsPerPage: itemsPerPage.value,

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

...filtersAndSortingParams()
}

const response = await axios.get('/api/employees', {
  headers: {
    Authorization: `Bearer ${token}`,
    Accept: 'application/json',
  },
  params: queryParams
})

const data = response.data

if (Array.isArray(data)) {
  employees.value = data
  totalItems.value = data.length
} else if (Array.isArray(data.items)) {
  employees.value = data.items
  totalItems.value = data.totalItems || 0
} else {
  console.warn('Unknown response format:', data)
  employees.value = []
  totalItems.value = 0
}

totalPages.value = totalItems.value > 0
? Math.ceil(totalItems.value / itemsPerPage.value)
: 1

} catch (error) {
  console.error('Error during rendering employees:', error)
  employees.value = []
  totalItems.value = 0
  totalPages.value = 1
}
}

const filteredEmployees = computed(() => {
  if (!Array.isArray(employees.value)) {
    return []
  }

  let result = [...employees.value];

  if (typeof searchQuery.value === 'string' && searchQuery.value.trim() !== '') {
    const query = searchQuery.value.toLowerCase();

    result = result.filter(e =>
      e.user?.first_name?.toLowerCase().includes(query) ||
      e.user?.last_name?.toLowerCase().includes(query) ||
      e.user?.email?.toLowerCase().includes(query)
    );
  }

  if (selectedJobTitle.value !== 'all') {
    result = result.filter(e => e.position === selectedJobTitle.value);
  }

  if (selectedDepartment.value !== 'all') {
    result = result.filter(e => e.departmentName === selectedDepartment.value);
  }

  if (selectedStatus.value !== 'all') {
    result = result.filter(e => e.status === selectedStatus.value);
  }

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    if (sortField.value) {
      const getSortValue = (item) => {
        const fieldMap = {
          name: () => `${item.user?.first_name ?? ''} ${item.user?.last_name ?? ''}`.toLowerCase(),
          jobTitle: () => item.position ?? '',
          lineManager: () => item.managerEmail ?? '',
          department: () => item.departmentName ?? '',
          status: () => item.status ?? '',
        };

        return fieldMap[sortField.value]?.() ?? '';
      };

      employees.value.sort((a, b) => {
        const aValue = getSortValue(a);
        const bValue = getSortValue(b);

        if (aValue < bValue) return sortDirection.value === 'asc' ? -1 : 1;
        if (aValue > bValue) return sortDirection.value === 'asc' ? 1 : -1;
        return 0;
      });
    }

    return result;
  });

  onMounted(() => {
    fetchEmployees()
    fetchDepartments()
    fetchPositions()
  })

  watch([selectedDepartment, selectedJobTitle, selectedStatus], fetchEmployees)
</script>

<template>
  <div class="employee-general-container">
    <aside class="sidebar">
      <Dashboard />
    </aside>
    <div v-if="isFormOpen" class="overlay-background"></div>
    <main class="dashboard-main-content">
      <header class="payroll-header">
        <HeaderComponent />
      </header>
      <div class="main-content">
        <div class="main-content-manage">
          <div class="main-content-text">
            <h1>Employees</h1>
            <p>Manage your Employee</p>
          </div>
          <div class="employee-actions">
            <button class="add-new" @click="openForm">
              
              Add New
            </button>
          </div>
          <div class="filters">

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<input
class="table-search"
type="text"
v-model="searchQuery"
@input="fetchEmployees"
placeholder="Search employee"
/>

<CustomSelect
:options="departments"
v-model="selectedDepartment"
placeholder="All Departments"
/>

<CustomSelect
:options="positions"
v-model="selectedJobTitle"
placeholder="All Job Titles"
/>

<CustomSelect
:options="statusOptions"
v-model="selectedStatus"
placeholder="All Status"
/>
</div>
<table class="table">
<thead class="table-head">
<tr>
<th @click="toggleSort('name')">
<div class="header-content">
Employee Name
<div class="sort-arrows">
<span :class="{ active: sortField === 'name' && sortDirection === 'asc'
}">▲</span>
<span :class="{ active: sortField === 'name' && sortDirection === 'desc'
}">▼</span>
</div>
</div>
</th>
<th @click="toggleSort('jobTitle')">
<div class="header-content">
Job Title
<div class="sort-arrows">
<span :class="{ active: sortField === 'jobTitle' && sortDirection === 'asc'
}">▲</span>
<span :class="{ active: sortField === 'jobTitle' && sortDirection === 'desc'
}">▼</span>
</div>
</div>
</th>
<th @click="toggleSort('lineManager')">
<div class="header-content">
Line Manager
<div class="sort-arrows">
<span :class="{ active: sortField === 'lineManager' && sortDirection === 'asc'
}">▲</span>
<span :class="{ active: sortField === 'lineManager' && sortDirection === 'desc'
}">▼</span>
</div>
</div>
</th>
<th @click="toggleSort('department')">
<div class="header-content">

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Department
<div class="sort-arrows">
  <span :class="{ active: sortField === 'department' && sortDirection === 'asc'
}">▲</span>
  <span :class="{ active: sortField === 'department' && sortDirection === 'desc'
}">▼</span>
</div>
</div>
</th>
<th @click="toggleSort('status')">
  <div class="header-content">
    Employee Status
    <div class="sort-arrows">
      <span :class="{ active: sortField === 'status' && sortDirection === 'asc'
}">▲</span>
      <span :class="{ active: sortField === 'status' && sortDirection === 'desc'
}">▼</span>
    </div>
  </div>
</th>
<th>
  <div class="header-content">
    <div class="sort-arrows"></div>
  </div>
</th>
</tr>
</thead>
<tbody>
<tr v-for="employee in filteredEmployees" :key="employee.id">
<td class="avatar-td">
  
  <span class="user-avatar-name">{{ employee.user?.first_name }} {{
employee.user?.last_name }}</span>
</td>
<td>{{ employee.position }}</td>
<td>{{ employee.managerEmail || '-' }}</td>
<td>{{ employee.departmentName }}</td>
<td class="status">
  <StatusSelect v-model="employee.status" :employee-id="employee.id" />
</td>
<td>
  <div class="account">
    <router-link :to="/dashboard/employee/detail/${employee.id}">
    <button class="account-action">
      
    </button>
    </router-link>
  </div>
</td>
</tr>
</tbody>
</table>
<div class="pagination">
  <button
    class="pagination-arrow-btn"
    :disabled="currentPage === 1"
    @click="currentPage-- && fetchEmployees()"
  >
    
  </button>

  <button
    v-for="page in totalPages"
    :key="page"

```

```

: class="['pagination-btn', { active: currentPage === page }]"
@ click="() => { currentPage = page; fetchEmployees() }"
>
{{ page }}
</button>

<button
class="pagination-arrow-btn"
: disabled="currentPage === totalPages"
@ click="currentPage++ && fetchEmployees()"
>

</button>
</div>
</div>
</main>
<transition name="slide">
<AddEmployeeModal v-if="isFormOpen" @close="closeForm"
: onCreate="fetchEmployees"/>
</transition>
</div>
</template>

<style scoped>
.sidebar {
width: 240px;
background-color: #fff;
border-right: 1px solid #e0e0e0;
height: 100vh;
position: sticky;
top: 0;
}

.dashboard-main-content {
flex: 1;
padding: 20px;
background-color: #f5f5f5;
}

th {
cursor: pointer;
position: relative;
}

.header-content {
display: flex;
align-items: center;
justify-content: space-between;
}

.sort-arrows {
display: flex;
flex-direction: column;
margin-left: 8px;
}

.sort-arrows span {
color: #ccc;
font-size: 10px;
line-height: 0.8;
}

.sort-arrows span.active {
color: #5c6bc0;
}

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.sort-arrows span:first-child {
  margin-bottom: 2px;
}

.filters {
  display: flex;
  gap: 15px;
  margin-top: 20px;
  align-items: center;
}

.table-search {
  padding: 12px 16px;
  border: 1px solid #e0e0e0;
  border-radius: 10px;
  font-size: 14px;
  min-height: 48px;
}

</style>

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток Ж ілюструє клас `StartTimerAction`, який фіксує початок робочого дня працівника. Відповідна логіка зберігає дані про час входу та часовий пояс у базі даних.

```
<?php

namespace App\Action;

use App\Entity\Attendance;
use App\Repository\EmployeesRepository;
use DateTime;
use DateTimeImmutable;
use DateTimeZone;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\SecurityBundle\Security;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;

readonly class StartTimerAction
{
    /**
     * @param Security $security
     * @param EmployeesRepository $employeeRepo
     * @param EntityManagerInterface $em
     */
    public function __construct(
        private Security $security,
        private EmployeesRepository $employeeRepo,
        private EntityManagerInterface $em,
    ) {}

    /**
     * @param Request $request
     * @return JsonResponse
     * @throws \DateInvalidTimeZoneException
     * @throws \DateMalformedStringException
     */
    public function __invoke(Request $request): JsonResponse
    {
        $user = $this->security->getUser();
        $employee = $this->employeeRepo->findOneBy(['user' => $user]);

        if (!$employee) {
            return new JsonResponse(['error' => 'Employee not found'], 404);
        }

        $timezoneString = 'Europe/Kyiv';
        $timezone = new DateTimeZone($timezoneString);
        $now = new DateTimeImmutable('now', $timezone);

        $attendance = new Attendance();
        $attendance->setEmployee($employee);
        $attendance->setDate($now);
        $attendance->setClockIn($now);
        $attendance->setTimezone($timezoneString);

        $this->em->persist($attendance);
        $this->em->flush();
    }
}
```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return new JsonResponse([
    'id' => $attendance->getId(),
    'clockIn' => $attendance->getClockIn()->format('H:i'),
    'date' => $attendance->getDate()->format('Y-m-d'),
], 201);
}
}

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

У Додатку 3 подано клас StopTimerAction, що завершує запис присутності працівника. Окрім збереження часу виходу, виконується обчислення зареєстрованого, оплаченого часу й дефіциту.

```
<?php

namespace App\Action;

use App\Repository\AttendanceRepository;
use App\Repository\EmployeesRepository;
use DateInvalidTimeZoneException;
use DateMalformedStringException;
use DateTimeImmutable;
use DateTimeZone;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\SecurityBundle\Security;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;

readonly class StopTimerAction
{
    /**
     *
     */
    private const EXPECTED_WORK_SECONDS = 8 * 3600;

    /**
     * @param Security $security
     * @param EmployeesRepository $employeeRepo
     * @param AttendanceRepository $attendanceRepo
     * @param EntityManagerInterface $em
     */
    public function __construct(
        private Security $security,
        private EmployeesRepository $employeeRepo,
        private AttendanceRepository $attendanceRepo,
        private EntityManagerInterface $em,
    ) {}

    /**
     * @param Request $request
     * @return JsonResponse
     * @throws DateInvalidTimeZoneException
     * @throws DateMalformedStringException
     */
    public function __invoke(Request $request): JsonResponse
    {
        $user = $this->security->getUser();
        $employee = $this->employeeRepo->findOneBy(['user' => $user]);

        if (!$employee) {
            return new JsonResponse(['error' => 'Employee not found'], 404);
        }

        $attendance = $this->attendanceRepo->findOneBy([
            'employee' => $employee,
            'clock_out' => null,
        ], ['clock_in' => 'DESC']);
```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (!$attendance) {
return new JsonResponse(['error' => 'No active attendance record found'], 404);
}

$timezoneString = $attendance->getTimezone() ?? 'UTC';
$timezone = new DateTimeZone($timezoneString);

$clockIn = DateTimeImmutable::createFromFormat(
'Y-m-d H:i:s',
$attendance->getClockIn()->format('Y-m-d H:i:s'),
$timezone
);

$clockOut = new DateTimeImmutable('now', $timezone);

$attendance->setClockOut($clockOut);

$loggedSeconds = max(0, $clockOut->getTimestamp() - $clockIn->getTimestamp());
$loggedFormatted = gmdate('H:i:s', $loggedSeconds);

$paidSeconds = min($loggedSeconds, self::EXPECTED_WORK_SECONDS);
$paidFormatted = gmdate('H:i:s', $paidSeconds);

$deficitSeconds = max(0, $loggedSeconds - self::EXPECTED_WORK_SECONDS);
$deficitFormatted = gmdate('H:i:s', $deficitSeconds);

$attendance->setLoggedTime($loggedFormatted);
$attendance->setPaidTime($paidFormatted);
$attendance->setDeficit($deficitFormatted);

$content = json_decode($request->getContent(), true);
if (isset($content['note'])) {
$attendance->setNote($content['note']);
}

$this->em->flush();

return new JsonResponse([
'id' => $attendance->getId(),
'clockIn' => $clockIn->format('H:i'),
'clockOut' => $clockOut->format('H:i'),
'loggedTime' => $loggedFormatted,
'deficit' => $deficitFormatted,
'note' => $attendance->getNote(),
]);
}
}

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток И

У Додатку И наведено реалізацію класу MyAttendanceProvider, який повертає список відвідувань конкретного авторизованого працівника. Це дозволяє працівнику переглядати лише власні записи.

```
<?php

namespace App\State;

use ApiPlatform\Metadata\Operation;
use ApiPlatform\State\Pagination\PartialPaginatorInterface;
use ApiPlatform\State\ProviderInterface;
use App\Repository\AttendanceRepository;
use Symfony\Bundle\SecurityBundle\Security;

readonly class MyAttendanceProvider implements ProviderInterface
{
    public function __construct(
        private Security $security,
        private AttendanceRepository $attendanceRepo,
    ) {}

    /**
     * @param Operation $operation
     * @param array $uriVariables
     * @param array $context
     * @return object|array|null
     */
    public function provide(Operation $operation, array $uriVariables = [], array $context = []): object|array|null
    {
        $user = $this->security->getUser();
        $employee = $user?->getEmployee();

        if (!$employee) {
            return [];
        }

        return $this->attendanceRepo->findBy(['employee' => $employee]);
    }
}
```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

У Додатку І наведено реалізацію компонента `MyAttendance.vue`, який забезпечує інтерфейс для обліку часу працівника. Користувач може здійснити вхід та вихід із робочої сесії, переглядати історію відвідувань, загальний робочий час, оплачений час та надлишкові години. Компонент взаємодіє з API системи, зберігає проміжні дані у `localStorage` та використовує клієнтський таймер для відображення поточного стану.

```
<script setup>
import { ref, computed, onMounted } from 'vue';
import Dashboard from '@components/Dashboard/Dashboard.vue';
import CustomCalendar from '@components/smallComponents/CustomCalendar.vue';
import HeaderComponent from '@components/smallComponents/HeaderComponent.vue';
import PaginationComponent from
'@components/smallComponents/PaginationComponent.vue';
import leftArrow from '@assets/images/left-arrow.png';
import rightArrow from '@assets/images/right-arrow.png';
import axios from "axios";

const isClockedIn = ref(false);
const timer = ref('00:00:00');
const totalWorkTime = ref('00:00:00');
let intervalId = null;
let startTs = 0;

function startTimer() {
  clearInterval(intervalId);
  intervalId = setInterval(() => {
    const elapsedSeconds = Math.floor((Date.now() - startTs) / 1000);
    const h = String(Math.floor(elapsedSeconds / 3600)).padStart(2, '0');
    const m = String(Math.floor((elapsedSeconds % 3600) / 60)).padStart(2, '0');
    const s = String(elapsedSeconds % 60).padStart(2, '0');
    timer.value = `${h}:${m}:${s}`;
  }, 1000);
}

async function clockIn() {
  try {
    const token = localStorage.getItem('access_token');
    const { data } = await axios.post('/api/attendance/start', {}, {
      headers: {
        Authorization: `Bearer ${token}`,
        Accept: 'application/json',
      }
    });

    isClockedIn.value = true;
    startTs = Date.now();
    localStorage.setItem('clock_in_ts', String(startTs));
    timer.value = '00:00:00';
    startTimer();
  } catch (error) {
    console.error('Failed to start timer:', error);
  }
}

function openClockOutModal() {
```

		Буланов О.П.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Толстой І.А.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

clearInterval(intervalId);
intervalId = null;
clockModalTime.value = new Date().toLocaleTimeString();
totalWorkTime.value = timer.value;
showClockModal.value = true;
}

function cancelClock() {
  if (isClockedin.value) startTimer();
  showClockModal.value = false;
}

async function confirmClock() {
  try {
    const token = localStorage.getItem('access_token');
    await axios.post('/api/attendance/stop', {
      note: notes.value,
    }, {
      headers: {
        Authorization: `Bearer ${token}`,
        'Content-Type': 'application/json',
        Accept: 'application/json',
      }
    });

    isClockedin.value = false;
    showClockModal.value = false;
    localStorage.removeItem('clock_in_ts');
    clearInterval(intervalId);
    timer.value = '00:00:00';
    await fetchAttendance();
    notes.value = '';
  } catch (error) {
    console.error('Failed to stop timer:', error);
  }
}

function formatDate(iso) {
  return iso ? iso.split('T')[0] : '';
}

function formatTime(iso) {
  return iso ? iso.split('T')[1]?.slice(0, 5) : '';
}

function formatDuration(seconds) {
  if (!seconds || isNaN(seconds)) return '0m';
  const h = Math.floor(seconds / 3600);
  const m = Math.floor((seconds % 3600) / 60);
  return `${h > 0 ? `${h}h` : ''}${m}m`.trim();
}

function durationInSeconds(from, to) {
  if (!from || !to) return 0;
  return Math.max(0, Math.floor((new Date(to) - new Date(from)) / 1000));
}

const records = ref([]);
const isLoading = ref(false);

async function fetchAttendance() {
  isLoading.value = true;
  try {
    const token = localStorage.getItem('access_token');
    const { data } = await axios.get('/api/my-attendance', {
      headers: {
        Authorization: `Bearer ${token}`,

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Accept: 'application/json',
    }
});

records.value = data.map(item => {
    const loggedSec = durationInSeconds(item.clock_in, item.clock_out);
    const paidSec = Math.min(loggedSec, 8 * 3600);
    const deficitSec = Math.max(0, loggedSec - 8 * 3600);
    return {
        id: item.id,
        date: formatDate(item.date),
        clockIn: formatTime(item.clock_in),
        clockOut: formatTime(item.clock_out),
        schedule: '8h',
        logged: formatDuration(loggedSec),
        paid: formatDuration(paidSec),
        deficit: deficitSec === 0 ? '0m' : formatDuration(deficitSec),
        note: item.note || '-',
        loggedSec,
        paidSec,
        deficitSec
    };
});
} catch (err) {
    console.error('Error loading attendance records:', err);
} finally {
    isLoading.value = false;
}
}

const filters = ref({ start: '', end: '' });
const page = ref(1);
const perPage = 8;

const filteredRecords = computed(() => {
    const start = filters.value.start ? new Date(filters.value.start) : null;
    const end = filters.value.end ? new Date(filters.value.end) : null;
    return records.value.filter(rec => {
        const recDate = new Date(rec.date);
        if (start && recDate < start) return false;
        if (end && recDate > end) return false;
        return true;
    });
});

const paginatedRecords = computed(() => {
    const startIdx = (page.value - 1) * perPage;
    return filteredRecords.value.slice(startIdx, startIdx + perPage);
});

const totalPages = computed(() => Math.ceil(filteredRecords.value.length / perPage));

const totals = computed(() => {
    const startOfWeek = new Date();
    startOfWeek.setDate(startOfWeek.getDate() - startOfWeek.getDay());
    const weekRecords = records.value.filter(r => new Date(r.date) >= startOfWeek);
    const totalLogged = weekRecords.reduce((sum, r) => sum + r.loggedSec, 0);
    const totalPaid = weekRecords.reduce((sum, r) => sum + r.paidSec, 0);
    const totalOvertime = weekRecords.reduce((sum, r) => sum + r.deficitSec, 0);
    return {
        schedule: '40h',
        logged: formatDuration(totalLogged),

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		103

```

    paid: formatDuration(totalPaid),
    overtime: formatDuration(totalOvertime)
  });
});

const showClockModal = ref(false);
const clockModalTime = ref('');
const notes = ref('');
const showEditModal = ref(false);
const edit = ref({ date: '', time: '' });

function cancelEdit() { showEditModal.value = false; }
function confirmEdit() { showEditModal.value = false; }

onMounted(async () => {
  await fetchAttendance();
  const storedTs = localStorage.getItem('clock_in_ts');
  if (storedTs) {
    const parsedTs = parseInt(storedTs);
    const clockInDate = new Date(parsedTs);
    const today = new Date();

    const isSameDay = clockInDate.toDateString() === today.toDateString();

    if (isSameDay && !isNaN(parsedTs)) {
      isClockIn.value = true;
      startTs = parsedTs;
      startTimer();
    } else {
      localStorage.removeItem('clock_in_ts');
    }
  }
});
</script>

<template>
  <div class="dashboard-layout">
    <aside class="sidebar"><Dashboard /></aside>
  <!-- <div v-if="isClockIn" class="overlay-background"></div>-->
  <div class="main-wrapper">
    <div class="payroll-container">
      <header class="payroll-header my-attendance-header"><HeaderComponent
    /></header>
    <main class="main-content">
      <div class="header">
        <div><h1 class="title">My Attendance</h1><p class="subtitle">Manage
your Attendance</p></div>
        <button class="btn btn-primary timer-btn" @click="isClockIn ?
openClockOutModal() : clockIn()">
          {{ isClockIn ? 'Check Out' : 'Check In' }}<span v-
if="isClockIn" class="timer-text">{{ timer }}</span>
        </button>
      </div>

      <div class="summary-cards">
        <div class="attendance-card card-schedule">
          <div class="card-label">Work Schedule</div>
          <div class="card-value">{{ totals.schedule }}</div>
        </div>
        <div class="attendance-card card-logged">
          <div class="card-label">Logged Time</div>
          <div class="card-value">{{ totals.logged }}</div>
        </div>
        <div class="attendance-card card-paid">

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <div class="card-label">Paid Time</div>
        <div class="card-value">{{ totals.paid }}</div>
    </div>
    <div class="attendance-card card-overtime">
        <div class="card-label">Overtime</div>
        <div class="card-value">{{ totals.overtime }}</div>
    </div>
</div>

<div class="filters">
    <div class="filter-item date-range">
        <CustomCalendar v-model="filters.start" placeholder="Start date"
        />
        <span>-</span>
        <CustomCalendar v-model="filters.end" placeholder="End date" />
    </div>
</div>

<div v-if="isLoading" class="loading">Loading records...</div>
<div class="table-container" v-else>
    <table class="attendance-table">
        <thead>
            <tr>
                <th>Date</th>
                <th>Clock In</th>
                <th>Clock Out</th>
                <th>Work Schedule</th>
                <th>Logged Time</th>
                <th>Paid Time</th>
                <th>Deficit</th>
                <th>Note</th>
            </tr>
        </thead>
        <tbody>
            <tr v-if="filteredRecords.length === 0"><td colspan="8" class="no-
records">No records found</td></tr>
            <tr v-else v-for="r in paginatedRecords" :key="r.id">
                <td>{{ r.date }}</td>
                <td>{{ r.clockIn }}</td>
                <td>{{ r.clockOut || '-' }}</td>
                <td>{{ r.schedule }}</td>
                <td>{{ r.logged }}</td>
                <td>{{ r.paid }}</td>
                <td>{{ r.deficit }}</td>
                <td>{{ r.note }}</td>
            </tr>
        </tbody>
    </table>
</div>

<PaginationComponent v-model:page="page" :total-pages="totalPages"
:prev-arrow-src="leftArrow" :next-arrow-src="rightArrow" />

<!-- Clock Out Modal -->
<div v-if="showClockModal" class="modal-overlay">
    <div class="modal">
        <h2>Clock out at {{ clockModalTime }}</h2>
        <p>Your total working time for today is <strong>{{ totalWorkTime
}}</strong></p>
        <textarea v-model="notes" placeholder="Input notes
here"></textarea>
        <div class="modal-actions">
            <button class="btn btn-secondary"
@click="cancelClock">Cancel</button>

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <button class="btn btn-primary" @click="confirmClock">Clock
Out</button>
    </div>
</div>
</div>

<!-- Edit Modal -->
<div v-if="showEditModal" class="modal-overlay">
    <div class="modal">
        <h2>Edit Paid Time</h2>
        <label>Date</label><input type="date" v-model="edit.date" />
        <label>Time</label><input type="time" v-model="edit.time" />
        <div class="modal-actions">
            <button class="btn btn-secondary"
@click="cancelEdit">Cancel</button>
            <button class="btn btn-primary"
@click="confirmEdit">Save</button>
        </div>
    </div>
</div>

</main>
</div>
</div>
</div>
</template>

<style src="../../../assets/css/components/Attendance.css" scoped></style>

```

		Буланов О.П.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Толстой І.А.				106
Змн.	Арк.	№ докум.	Підпис	Дата		