ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ

УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Проектування та створення автоматизованої системи обліку кадрів підприємства»

Студента 4 курсу, 10 групи, Радзівілова Максима Ігоровича

спеціальності
122 «Комп'ютерні науки» _____ *підпис студента*

Науковий керівник
кандидат фізико-математичних наук, доцент _____ *підпис керівника* Самойленко Ганна Тимофіївна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студента

Радзівілова Максима Ігоровича
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Проектування та створення автоматизованої системи обліку кадрів підприємства»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *26 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка та впровадження автоматизованої системи обліку кадрів для ефективного управління кадровими процесами на підприємстві.
Об'єкт дослідження: є засоби створення автоматизованої системи обліку кадрів підприємства.
Предмет дослідження: є процес проектування та створення автоматизованої системи обліку кадрів підприємства, включаючи аналіз, проектування, розробку та впровадження системи.
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Самойленко Г.Т.	15.12.2022 р.	15.12.2022 р.
2	Самойленко Г.Т.	15.12.2022 р.	15.12.2022 р.
3	Самойленко Г.Т.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ОБЛІКУ КАДРІВ ПІДПРИЄМСТВА

1.1. Значення автоматизованої системи обліку кадрів для підприємства

1.2. Основні завдання та функції автоматизованої системи обліку кадрів

1.3 Виявлення проблем та недоліків існуючих системи обліку кадрів

РОЗДІЛ 2. АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДО АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ

2.1 Аналіз потреб підприємства щодо автоматизованої системи обліку кадрів

2.2 Визначення функціональних вимог до системи обліку кадрів

2.3. Визначення нефункціональних вимог до системи обліку кадрів

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ

3.1 Вибір технологічних рішень та інструментів для розробки системи

3.2 Архітектура та структура системи обліку кадрів

3.3 Реалізація системи обліку кадрів

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ОБЛІКУ КАДРІВ ПІДПРИЄМСТВА*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДО АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ*	*12.05.2023*	*12.05.2023*
7	*Висновки та результати*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	*31.05.2023 -01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту) Самойленко А.Т.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Радзівілов М.І.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи (проекту)
_____ 22.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____Радзівілова М.І.
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)

«____»_____2023 р.

Анотація

Дана випускна кваліфікаційна робота присвячена розробці автоматизованої системи обліку кадрів на підприємстві. Проведений аналіз значення та процесів обліку кадрів, а також досліджено існуючі системи обліку на підприємстві, виявлено проблеми та недоліки. Проведений аналіз потреб підприємства щодо автоматизованої системи обліку кадрів, визначено функціональні та нефункціональні вимоги до системи. Описано вибір технологічних рішень та інструментів для розробки системи, а також розроблено архітектуру та структуру системи. Наведений приклад реалізації системи обліку кадрів. Результатом роботи є Web-додаток, який забезпечує ефективний облік працівників.

Ключові слова: облік кадрів, автоматизована система, Web-додаток.

Abstract

This graduation thesis focuses on the development of an automated personnel management system for an enterprise. An analysis of the importance and processes of personnel management has been conducted, along with an investigation of existing personnel management systems within the enterprise, identifying problems and shortcomings. The study analyzes the enterprise's needs for an automated personnel management system, defines the functional and non-functional requirements of the system. The selection of technological solutions and development tools for the system is described, along with the design of the system's architecture and structure. An implementation example of the personnel management system is provided. The outcome of this work is a web application that facilitates efficient employee management.

Keywords: personnel management, automated system, web application**.**

ЗМІСТ

ВСТУП.....8

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ОБЛІКУ КАДРІВ ПІДПРИЄМСТВА.....10

1.1 Значення та процеси обліку кадрів на підприємстві.....10

1.2 Аналіз засобів та існуючих типів систем обліку на підприємстві.....13

1.3 Виявлення проблем та недоліків існуючих системи обліку кадрів.....16

РОЗДІЛ 2. АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДО АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ.....20

2.1 Аналіз потреб підприємства щодо автоматизованої системи обліку кадрів 20

2.2 Визначення функціональних вимог до системи обліку кадрів 23

2.3 Визначення нефункціональних вимог до системи обліку кадрів 25

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ.....29

3.1 Вибір технологічних рішень та інструментів для розробки системи.....29

3.2 Архітектура та структура системи обліку кадрів 33

3.3 Реалізація системи обліку кадрів.....36

Висновки та результати.....51

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....52

ДОДАТОК.....54

ВСТУП

Облік кадрів є невід'ємною частиною ефективного управління персоналом на підприємстві. З метою покращення роботи з кадрами, оптимізації процесів та підвищення продуктивності праці, все більше підприємств звертаються до автоматизації обліку кадрів. Інформаційні технології дозволяють створити автоматизовану систему обліку кадрів, що спрощує рутинні процеси, забезпечує точність та доступність даних, покращує аналітику та планування персоналу. У зв'язку з цим, дослідження проектування та створення автоматизованої системи обліку кадрів підприємства має велику актуальність та практичне значення.

Метою даної роботи є розробка та реалізація автоматизованої системи обліку кадрів на підприємстві. Головним завданням є створення функціональної та ефективної системи, яка забезпечуватиме зручний доступ до інформації про працівників та розрахунок заробітної плати. При цьому, система повинна бути гнучкою та адаптованою до специфіки підприємства, враховувати його потреби та вимоги.

У рамках випускної роботи будуть розглянуті наступні загальні завдання:

1. Вивчення теоретичних аспектів обліку кадрів, актуальних підходів до автоматизації та систем обліку кадрів.
2. Аналіз потреб та вимог підприємства щодо обліку кадрів, виявлення основних проблем та недоліків існуючих систем.
3. Проектування архітектури та функціональних можливостей автоматизованої системи обліку кадрів.
4. Розробка та реалізація системи, включаючи розробку інтерфейсу та необхідних модулів.
5. Тестування та апробація розробленої системи на підприємстві.

6. Оцінка ефективності та виявлення практичного значення автоматизованої системи обліку кадрів.

Об'єктом дослідження є засоби створення автоматизованої системи обліку кадрів підприємства.

Предметом дослідження є процес проектування та створення автоматизованої системи обліку кадрів підприємства, включаючи аналіз, проектування, розробку та впровадження системи.

У процесі виконання роботи застосовані наступні **методи дослідження**: аналіз наукової літератури, вивчення документації підприємства, порівняння існуючих систем обліку кадрів, моделювання процесів, проектування архітектури системи, розробка програмного забезпечення, тестування та апробація системи.

Результати дослідження та реалізація автоматизованої системи обліку кадрів підприємства матимуть **практичне значення** для підприємств. Впровадження системи сприятиме покращенню ефективності управління персоналом, зменшенню рутинних завдань, підвищенню точності та доступності даних, забезпеченню аналітики та плануванню персоналу. Крім того, система дозволить зменшити витрати на ведення кадрової документації та сприятиме підвищенню загальної продуктивності праці на підприємстві.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 17 найменувань, додатків і містить 51 сторінки основного тексту, 16 рисунків і 1 таблиця.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ АСПЕКТИ ОБЛІКУ КАДРІВ ПІДПРИЄМСТВА

1.1 Значення та процеси обліку кадрів на підприємстві

У світі, де бізнес-середовище стає все більш конкурентним і швидкозмінним, ефективне управління людськими ресурсами, є ключовим та одним з найголовнішим фактором успіху підприємства.

Підприємство – це стабільна формальна соціальна структура, яка використовує ресурси з навколишнього середовища і переробляє їх у продукти своєї діяльності для отримання прибутку. З позиції системного підходу підприємство будь-якого рівня і масштабу необхідно розглядати як велику систему, яка складається із взаємопов’язаних і взаємодіючих елементів [1].

У процесі прийому працівника на роботу відділом кадрів (ВК) застосовується процедура складання особистої картки, що включає збір інформації з трудової книжки, що зберігається у ВК, а також з диплому та паспорту працівника. Після цього працівникові призначається посада та відділ відповідно до його професійних навичок та кваліфікації.

Наступним кроком є визначення розміру окладу на основі цих даних у штатному розкладі. У наказі про прийом на роботу зазначаються посада, відділ, оклад працівника, а також його прізвище, ім'я, по батькові, дата прийому на роботу та табельний номер, який присвоюється працівникові в ВК. Інспектор відділу кадрів повідомляє керівника відповідного відділу про назначення працівника на роботу.

На підприємстві рух кадрів відображається шляхом оформлення первинних документів, зокрема наказів про прийом, переміщення та звільнення працівників. (рис.1.1)

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«УСЕ БУДЕ ДОБРЕ»
(ТОВ «УСЕ БУДЕ ДОБРЕ»)

Код ЄДРПОУ 65465465

НАКАЗ

17.09.2021 Київ № 177/к/тр

Про переміщення
Світлани Лозової

У зв'язку зі створенням у юридичному відділі сектору претензійної роботи, керуючись частиною другою статті 32 КЗпП України,

ПЕРЕМІСТИТИ:

ЛОЗОВУ Світлану Вікторівну, оператора комп'ютерного набору юридичного відділу, з 21.09.2021 до претензійного сектору юридичного відділу з подальшим виконанням роботи, обумовленої трудовим договором, із дотриманням попереднього графіка і режиму роботи та раніше встановлених умов оплати праці.

Підстава: доповідна записка керівника юридичного відділу Поліни Мазур від 17.09.2021, зареєстрована за № 4.

Директор *Добродій* Костянтин ДОБРОДІЙ

Візи, відмітка про ознайомлення з наказом

Рис.1.1 Приклад наказу на переміщення

У практиці обліку кадрів також застосовуються різні типи складів, такі як списковий склад, середньосписковий склад та наявний склад.

Списковий склад відображає фактичну чисельність всіх працівників організації. До спискового складу включаються працівники різних категорій, включаючи постійних, сезонних та тимчасових працівників, які були прийняті на роботу на період тривалістю не менше 5 днів. Крім того, до спискового складу включаються також працівники, які працюють за основним напрямом діяльності організації тривалий час. Процес включення працівників до спискового складу розпочинається з дня їх прийому на роботу.

У списку працівники повинні бути відображені як ті, хто фактично працює на даний період, так і ті, хто відсутні на роботі з певних причин.

Кадри є штатний склад працівників підприємств, установ, організацій і поділяються на дві великі групи: кадри управління (службовці) і робочі кадри. До робітників належать працівники, які безпосередньо зайняті створенням матеріальних цінностей або роботами по наданню різноманітних виробничих послуг і переміщенням вантажів [2].

У контексті виробництва робочих працівників можна умовно класифікувати на основних і допоміжних. Це має значення як важливий аналітичний показник ефективності виробництва. Зростання механізації та автоматизації виробничих процесів призводить до зміни ролі допоміжних працівників і виявлення їх значущості.

Облік кадрів або кадрове діловодство – це діяльність, яка передбачає створення та організування роботи з документами особового складу з питань оформлення, прийняття, переведення, звільнення, обліку, атестації, навчання, підвищення кваліфікації, стажування, пенсійного забезпечення працівників [3].

Облік кадрів виступає як складова цієї системи управління персоналом, де збираються, аналізуються та використовуються дані про працівників з метою оптимізації їхньої роботи та розвитку організації в цілому.

Він також охоплює широкий спектр процесів і дій, пов'язаних не тільки з управлінням персоналом та прийняття на роботу.

А також включаючє збір інформації про працівників, навчання, оцінку продуктивності, кар'єрне розвиток, оплату праці, управління відпустками та інше. Ці дані надають підприємству можливість зробити обґрунтовані рішення щодо використання своїх ресурсів та розвитку персоналу. Слід

зазначити що саме використання паперового обліку не дає підприємству можливості оперативно приймати рішення.

1.2 Аналіз засобів та існуючих типів систем обліку на підприємстві

Існує декілька типів систем обліку, що використовуються для ведення обліку кадрів на підприємстві.

Основні типи систем обліку включають ручний, електроний, інтигрований та хмарний облік.

Ручний облік передбачає використання паперової документації, такої як трудові книжки, реєстри та інші документи, для реєстрації та зберігання інформації про працівників. В цьому випадку облік здійснюється вручну, адміністративним персоналом підприємства. Використання такого типу обліку обмежує ефективність управління та являє собою складну систему з зберіганням паперового архіву данних.

На відміну від ручного, електронний облік (Рис.1.2) використовує комп'ютерні програми для здійснення процесу обліку кадрів. Одні з найвідоміших програм є:

- Microsoft Excel. Це одна з найбільш відомих програм для обробки даних, включаючи облік кадрів. Вона пропонує широкий набір функцій для створення, оформлення і аналізу таблиць.

№ з/п	Прізвище, ініціали	Дата прийому на роботу	Розряд	Ставка	Виконання плану, %
1	Антонов М.І.	02.03.1980	5	1750,00 грн	120,00
2	Громов А.М.	12.12.1994	4	1500,00 грн	115,00
3	Іванченко І.С.	21.05.2002	3	1100,00 грн	98,00
4	Кравчук К.І.	25.07.2000	4	1200,00 грн	100,00
5	Козаченко І.Д.	04.04.1996	6	2350,00 грн	100,00
6	Науменко В.І.	23.03.2006	2	1050,00 грн	105,00
7	Потапова Н.В.	24.03.2006	2	1050,00 грн	96,00
8	Прокопенко А.Б.	01.02.2002	4	1308,00 грн	110,00
			Максимальне:		
			Середнє:		

Рис.1.2 Зразок електроного обліку в Microsoft Excel

- Google Sheets. Це онлайн-аркуш, що надається безкоштовно Google. Він має схожий інтерфейс з Excel і дозволяє зберігати дані в хмарі та спільно працювати з ними.
- LibreOffice Calc. Це електронна таблиця, яка є частиною пакета офісних програм LibreOffice. LibreOffice є безкоштовною і відкритою альтернативою популярному пакету Microsoft Office.
- Numbers (для Mac). Це програма для обробки даних, що поставляється з комп'ютерами Mac. Вона пропонує різні функції для створення таблиць, графіків і діаграм.

Інформація про працівників зберігається в електронному вигляді і може бути легко доступною та оброблятися з використанням комп'ютерних інструментів. Електронний облік забезпечує більшу ефективність та точність обробки даних, а також спрощує пошук та аналіз інформації про працівників.

Інтегрована система кадрового обліку (ІСКО) важається одним з найєфективнішою системою, яка об'єднує облік кадрів з іншими функціями підприємства, такими як облік оплати праці, відомості про відпустки,

навчання та розвиток, атестація тощо. З представлених на ринку слід виділити такі системи:

- SAP SuccessFactors: Ця система надає широкий спектр функцій, включаючи кадровий облік, управління персоналом, управління талантами, адміністрування заробітної плати та багато іншого.
- Oracle HCM Cloud: Ця система управління людськими ресурсами надає модулі для кадрового обліку, управління трудовими відносинами, заробітної плати, рекрутингу, навчання та інших процесів.
- Workday HCM: Це програмне забезпечення, яке включає модулі для кадрового обліку, заробітної плати, управління талантами, рекрутингу та інших функцій. Інтегрована система кадрового обліку дозволяє забезпечити комплексний підхід до управління персоналом та забезпечити взаємодію між різними аспектами управління людськими ресурсами.

Хмарні системи обліку: Це відносно новий тип системи, який базується на зберіганні та обробці даних в хмарних обчислювальних сервісах. Він дозволяє зберігати та обробляти інформацію про працівників в хмарних серверах, що забезпечує більшу доступність, масштабованість та зручність використання. Такі системи обліку можуть бути електронними або інтегрованими та надають багатофункціональність для управління кадрами.

Ці типи систем обліку можуть використовуватися окремо або в поєднанні один з одним для забезпечення ефективного та точного обліку кадрів на підприємстві. Слід відзначити, що інтеграція останніх типів систем обліку кадрів у існуючий ВК може бути найскладнішою задачею через їх комплексність та потребу в інтеграції з різними програмами. Це особливо важливо враховувати, якщо наявний час на вдосконалення підприємства та роботу з персоналом є обмеженим.

1.3 Виявлення проблем та недоліків існуючих системи обліку кадрів

Виявлення проблем існуючих систем обліку кадрів вимагає аналізу функціональних, організаційних та технічних аспектів системи. Дослідження здійснюються за допомогою детального огляду процесів, процедур та взаємодії, які відбуваються в системі обліку кадрів, з метою ідентифікації проблемних зон та недоліків.

Однак більшість відділів кадрів на сьогодні не виконують функції методичного і координуючого центру з кадрової роботи в організації, оскільки вони структурно роз'єднані з підрозділами [4]:

- відділом заробітної плати;
- відділом охорони праці;
- відділом соціального забезпечення працівників

У багатьох організаціях відділи кадрів виявляються недостатньо компетентними у професійному відношенні, що призводить до невиконання наступних завдань управління людськими ресурсами:

- Здійснення соціально-психологічної діагностики персоналу.
- Керування виробничими та соціальними конфліктами в організації.
- Проведення аналізу кадрового потенціалу персоналу.
- Планування і контроль ділової кар'єри.
- Забезпечення соціально-психологічної адаптації.

Зазвичай нові кадрові служби в організації формуються на основі відділу кадрів, відділу організації праці і заробітної плати, відділу охорони праці та інших відповідних служб (Рис.1.3).

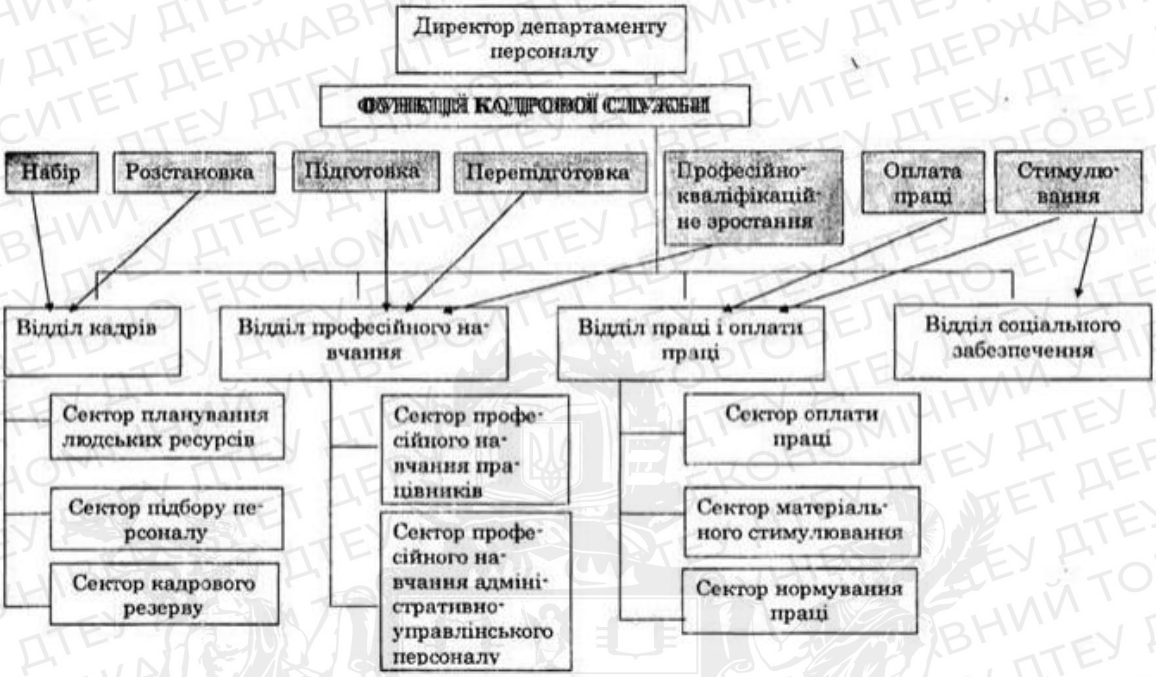


Рис.1.3 Організаційна структури кадрової служби

Один з аспектів, які варто розглянути, це архітектура та організація системи обліку кадрів. Як висвітлюються дані про працівників, яка інформація зберігається, як здійснюється доступ до неї та як вона оновлюється. Основна проблема всіх ВК, які використовують виключно ручний облік є дублювання даних, неповна або неточна інформація, труднощі з пошуком та фільтрацією даних.

Крім того, якщо процеси виконуються вручну декількома різними підрозділами, це може спричиняти помилки та займати багато часу. Важливо, які дані збираються від працівників та як вони переносяться до системи, якщо цей процес неефективний або недостатньо автоматизований, це може призводити до втрати даних або затримки у їх обробці.

Значною проблемою також є недостатня інтеграція системи обліку кадрів з іншими системами в організації. Наприклад, система обліку кадрів

повинна взаємодіяти з системами планування ресурсів. Недостатня або неправильна інтеграція може призводити до невідповідностей даних, втрати інформації та ускладнень у процесі управління підприємством.

Крім того, забезпечення безпеки та конфіденційності даних є критично важливим аспектом у системі обліку кадрів. Проблеми можуть виникати внаслідок недостатнього контролю доступу до даних, так як організації розділяють облік кадрів на різні служби це додає вразливостей системі, а також недостатніх заходів безпеки. У таких випадках, зловмисники можуть отримати доступ до конфіденційної інформації про працівників, що може призвести до серйозних наслідків для організації та її співробітників.

Якщо підприємство вже використовує систему обліку варто приділити увагу на її ефективність та продуктивність. Погано розроблені алгоритми та процеси будуть призводити до надмірного завантаження системи, що буде проявлятись у повільному відгуку та затримках. Також слід зазначити що важливо зберегти можливість масштабованості системи так як її обмежування не зможуть в подальшому враховувати зростаючі потреби організації щодо кадрового обліку.

Також багато проблем є в точності та достовірність даних в системі обліку. Недостатня або неточна інформація про працівників може призвести до неправильного призначення завдань, некоректного розрахунку заробітної плати, помилок у звітах та аналітичних даних. Важливо щоб система змогла виявити можливі джерела помилок, такі як некоректне введення даних, недостатня перевірка на достовірність та мати механізми автоматичного виявлення та виправлення помилок.

Аналітика та звітність у системі обліку кадрів на більшості підприємств недостатня. Не коректна аналітична функціональність може ускладнювати процеси прийняття рішень з управління персоналом. Відсутність звітів та

аналітичних даних може заважати оцінці продуктивності працівників, ідентифікації потреб у навчанні та розвитку, а також управлінню талантами.

Аналізуючи типові системи обліку слід зазначити що є проблеми, пов'язані з інтерфейсом користувача та користувацьким досвідом. Складність та неінтуїтивність інтерфейсу ускладнюють роботу з системою та призводити до помилок вводу даних. Недостатня підтримка користувачів, недоступність допомоги та навчання призводить до незадоволеності та опору з боку користувачів системи.

Якщо система не відповідає потребам організації або не інтегрується зі стратегічними процесами, це може призвести до неефективного управління персоналом та втрати конкурентних переваг.

РОЗДІЛ 2.
АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДО АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ

2.1 Аналіз потреб підприємства щодо автоматизованої системи обліку кадрів

Потреба у впровадженні автоматизованої системи обліку кадрів виникає і зростає з кожним роком. З попередньої інформації в роботі ми знаємо що багато підприємств займались кадровим обліком вручну, використовуючи паперові документи і таблиці Excel, Однак, використання паперових документів та таблиць Excel може мати низьку вартість впровадження, але при збільшенні робочої сили та складності процесів, такий підхід може стати недоцільним та приводити до витрат часу і помилок. Слід зазначити що важливо розуміти для якого підприємства створюється система обліку, і за який час її потрібно впровадити.

Основні переваги автоматизованої системи обліку кадрів полягають у забезпеченні точності і надійності даних, швидкому доступі до інформації, зручному аналізі і звітності, автоматичному розрахунку заробітної плати, відпусток і інших кадрових показників. Така система також допомагає уникнути порушень трудового законодавства, забезпечує збереження історичних даних про співробітників, а також полегшує процеси рекрутингу та планування кадрів.

Для проведення аналізу потреб підприємства щодо АСОК слід враховувати декілька етапів. Перш за все, необхідно дослідити особливості діяльності самого підприємства. Кожне підприємство має свою структуру, розмір, склад персоналу та інші особливості, які можуть впливати на потреби в системі обліку кадрів. Прикладом може бути автоматизована система для

великого, та малого підприємства впроваджена в облік, вони суттєво відрізняються за своїми характеристиками і вимогами. (Таб.2.1)

Далі, слід вивчити законодавчу базу, що регулює кадровий облік, відповідні нормативні акти, положення та вимоги. Це дозволить встановити, наскільки система обліку кадрів повинна відповідати правовим вимогам і забезпечувати належний контроль та звітність.

Для ретельного аналізу потреб підприємства необхідно взяти до уваги його розмір і обсяги діяльності. Великі підприємства зі значною кількістю працівників можуть мати складніші потреби в автоматизованій системі обліку кадрів, яка забезпечить швидку і точну обробку великого обсягу даних. У невеликих підприємств можуть бути інші пріоритети, наприклад, простота використання та економічна ефективність системи.

Також слід враховувати наявність інших систем управління на підприємстві. Наприклад, якщо підприємство вже використовує систему управління виробництвом або систему управління відносинами з клієнтами, то бажаною буде інтеграція системи обліку кадрів з існуючими системами. Це дозволить уникнути дублювання даних, підвищити ефективність обміну даними і покращити загальний управлінський процес.

Таблиця 2.1 Порівняння потреб підприємств в АСОК

Характеристика	Велике-підприємство	Мале-підприємство
Масштаби діяльності	Операції з великим обсягом даних, багатофункціональність	Обмежений обсяг операцій, простота використання
Облікова система	Складна, інтегрована система з багатьма модулями	Проста, окремі програми для кожної функції

Таблиця 2.1 Продовження

Витрати на впровадження	Високі витрати на впровадження та навчання персоналу	Низькі витрати на впровадження та навчання персоналу
Бюджет	Великий бюджет на придбання та підтримку системи	Обмежений бюджет, низькі витрати на систему
Функціональність	Великий набір функцій, здатний задовольнити потреби різних відділів	Базовий набір функцій для основних потреб підприємства
Інтеграція	Інтеграція з іншими системами, включаючи CRM, ERP тощо	Мінімальна інтеграція з іншими системами
Звітність	Деталізовані звіти для менеджменту та зовнішніх зацікавлених сторін	Стандартні звіти для внутрішнього використання
Система контролю	Складна система контролю внутрішніх процесів та фінансів	Проста система контролю без складних вимог
Технічна підтримка	Спеціалізована команда для підтримки та розвитку системи	Основна технічна підтримка, можливо, зовнішня

Підсумовуючи, автоматизована система обліку кадрів є необхідним інструментом для сучасних підприємств. Аналіз потреб підприємства в такій системі вимагає вивчення особливостей його діяльності, вимог законодавства,

розміру та обсягів діяльності, а також інтеграційних можливостей з іншими системами. Шляхом глибокого аналізу можна визначити оптимальний варіант автоматизованої системи обліку кадрів, що задовольнятиме потреби підприємства та сприятиме його успішному розвитку.

2.2 Визначення функціональних вимог до системи обліку кадрів

Функціональні вимоги (functional requirements) визначають функціональність ПЗ, яку розробники повинні забезпечити, щоб користувачі змогли виконати свої завдання в межах бізнес-вимог [5].

Конкретні функції, операції та задачі, які повинна виконувати автоматизована система обліку кадрів. Вони описують, які можливості та функції має мати система для ефективного управління персоналом.

Функціональні вимоги виконують кілька важливих ролей у процесі розробки автоматизованої системи обліку кадрів такі як:

Визначення потреб бізнесу: Функціональні вимоги допомагають ідентифікувати потреби організації щодо управління персоналом. Вони встановлюють, які конкретні функції та можливості має мати система для задоволення цих потреб.

Орієнтир для розробки: функціональні вимоги стають орієнтиром для розробників під час створення системи. Вони чітко визначають, які задачі система повинна виконувати і які функції вона повинна мати. Це сприяє структурованості та ефективності процесу розробки.

Валідація системи: функціональні вимоги служать основою для валідації системи. Вони дозволяють перевірити, чи відповідає розроблена система вимогам та чи виконує необхідні функції. Це допомагає переконатися, що система відповідає потребам організації перед її впровадженням.

Розробка функціональних вимог відбувається на кількох етапах процесу розробки автоматизованої системи обліку кадрів:

- Аналіз потреб: вивчаються потреби організації щодо управління персоналом. Вимоги збираються через спілкування зі зацікавленими сторонами, аналізування діючих процесів та врахування рекомендацій і стандартів.
- Визначення вимог: на цьому етапі функціональні вимоги формалізуються та документуються. Вони визначають, які конкретні функції та можливості має мати система для вирішення проблем та задач управління персоналом.
- Аналіз та специфікація: Функціональні вимоги аналізуються та уточнюються на цьому етапі. Вони розбиваються на більш детальні компоненти та документуються у вигляді специфікацій, які слугують основою для розробки системи.

Детальний опис функціональних вимог розподіляється на різні підрозділи, які визначають конкретні функції та операції системи. Ці підрозділи можуть включати реєстрацію та управління персональними даними, облік робочого часу та графіків роботи, управління персоналом та розвиток кар'єри, забезпечення безпеки та конфіденційності даних.

1. Додавання нового працівника до бази даних.

Система повинна мати можливість додавати нових працівників до бази даних. Ця функція включає в себе заповнення важливої інформації про працівника, такої як його особисті дані, контактна інформація, позиція, дата прийняття на роботу та інші відомості, необхідні для обліку персоналу.

2. Назначення та зміна заробітної плати працівникам.

Система повинна дозволяти назначати та змінювати заробітну плату працівникам. Ця функція дозволяє керівникам встановлювати та оновлювати заробітну плату на основі факторів, таких як посадовий оклад, рівень досвіду, результативність роботи тощо.

3. Надання статусу працівника для можливості підвищення.

Система повинна мати можливість надавати працівникам певний статус, що показує їхню готовність до підвищення. Це дозволяє визначити потенційних кандидатів на підвищення та надає управлінцям зручний інструмент для визначення подальшого розвитку кар'єри працівників.

4. Фільтрація та пошук працівників.

Система повинна мати можливість фільтрувати та здійснювати пошук працівників за різними критеріями, такими як посада, відділ, досвід роботи, кваліфікація тощо. Це спрощує процес пошуку необхідної інформації про працівників та допомагає управлінцям при прийнятті рішень щодо управління персоналом.

2.3 Визначення нефункціональних вимог до системи обліку кадрів

Нефункціональні вимоги (англ. Non-Functional Requirements) – це вимоги до програмного забезпечення, які задають критерії оцінювання якості його роботи [6].

Нефункціональні вимоги визначають властивості та характеристики системи, які не відносяться безпосередньо до її функціоналу, але мають велику значимість для ефективної роботи системи та задоволення потреб користувачів.

Вони описують якості системи, такі як надійність, продуктивність, безпека, масштабованість та інші аспекти, які впливають на її функціонування та використання.

Нефункціональні вимоги важливі для забезпечення якості та ефективності системи обліку кадрів. Вони допомагають визначити вимоги до надійності, швидкодії, безпеки, зручності використання та інших аспектів, які можуть впливати на задоволення потреб користувачів і успішну реалізацію бізнес-процесів організації. Нефункціональні вимоги розробляються після визначення функціональних вимог та наступного аналізу потреб користувачів і бізнес-вимог. Вони враховуються на етапі проектування системи та встановлюють вимоги до якості, продуктивності, безпеки, зручності використання та інших параметрів системи.

Опис нефункціональних вимог включає в себе розширене формулювання кожної вимоги та її детальний опис. Нефункціональні вимоги можуть включати такі аспекти, як надійність, продуктивність, безпека, зручність використання, сумісність.

Система повинна працювати без відмов та збоїв, забезпечуючи стабільну роботу та збереження даних.

Повинна мати достатню швидкодію та масштабованість для ефективної обробки великого обсягу даних та операцій.

Безпека: система повинна мати механізми захисту даних, контролю доступу та запобігання несанкціонованому доступу до інформації.

Зручність використання: система повинна бути зрозумілою та зручною у використанні, з інтуїтивно зрозумілим інтерфейсом та функціональністю.

Сумісність: система повинна взаємодіяти з іншими системами та стандартами, забезпечуючи сумісність та інтеграцію даних.

Розглянемо нефункціональні вимоги до нашої майбутньої автоматизованої системи обліку кадрів:

Надійність: додаток повинен забезпечувати стабільну роботу без відмов та збоїв. Система повинна мати механізми резервного копіювання та відновлення даних для запобігання їх втраті.

Продуктивність: додаток повинен працювати швидко та ефективно, забезпечуючи швидкодію виконання операцій та завантаження даних.

Безпека: додаток повинен мати механізми автентифікації та авторизації для контролю доступу до системи та обмеження прав користувачів. Система повинна забезпечувати шифрування конфіденційної інформації та захист даних від несанкціонованого доступу. Якщо цей функціонал потрібен його можливо швидко встановити.

Зручність використання: Додаток буде мати інтуїтивно зрозумілий інтерфейс та простий у використанні. На далі система повинна мати можливість персоналізації налаштувань та відображення необхідної інформації.

Сумісність: Додаток повинен бути сумісний з різними операційними системами (наприклад, Windows, macOS, Linux) та браузерами (наприклад, Chrome, Firefox, Safari). Система повинна мати можливість інтеграції з іншими системами, такими як система управління персоналом або бухгалтерська система.

Масштабованість: Система повинна бути здатна масштабуватися для обробки зростаючого обсягу даних та забезпечення швидкості роботи навіть при збільшенні кількості користувачів та операцій.

Доступність: Додаток повинен бути доступний з будь-якого місця та будь-якого пристрою з підключенням до Інтернету.

Система повинна забезпечувати підтримку людей з різними особливими потребами, включаючи навігацію за допомогою клавіатури та екранного читача.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ КАДРІВ

3.1 Вибір технологічних рішень та інструментів для розробки системи

Аналіз попередніх розділів надає нам структуру, яка більш за все потрібна в кадрової політиці більшості підприємств, з головного це інформативний web-додаток, який буде допомагати виконувати організаційні та кадрові функції організацій, які не встигли додати автоматизовану систему, або хочуть інтегрувати нову АСОК для обліку кадрів з подальшим її вдосконаленням.

Тому з можливих найефективніших варіантів краще за все буде створення web-додатку, який буде оптимізовувати роботу працівників ВК.

Щоб розробити інтерфейс користувача (Рис.3.1)додатку я використовував дизайнерську програму Figma. Figma дозволяє легко створювати та змінювати інтерфейсні елементи. Завдяки миттєвому оновленню змін у режимі реального часу, розробники можуть швидко проглядати та тестувати різні варіанти дизайну, вносити зміни та вдосконалювати інтерфейс без необхідності перезавантажувати сторінки або зберігати локальні файли.

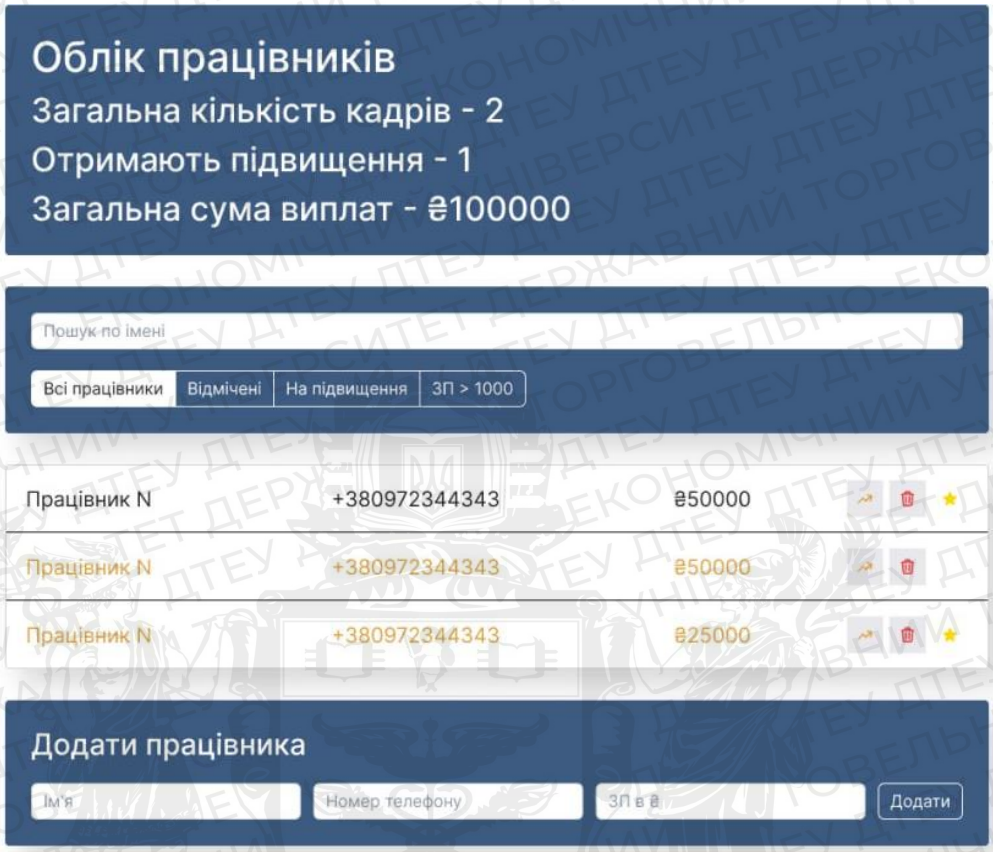


Рис3.1 Остаточний дизайн майбутнього додакту

Також Figma дозволяє створювати інтерактивні прототипи для тестування та оцінки користувацького досвіду. Завдяки вбудованим можливостям створення перехідних ефектів, анімації та взаємодії, розробники можуть візуалізувати та продемонструвати роботу додатку замовникам або потенційним користувачам, що допомагає виявити та виправити можливі проблеми ще на етапі проектування.

Вона надає можливість створювати повторно використовувані компоненти (Рис.3.2) та стилі для швидкої розробки та збереження єдиної системи дизайну. Це дозволяє забезпечити консистентність інтерфейсу.

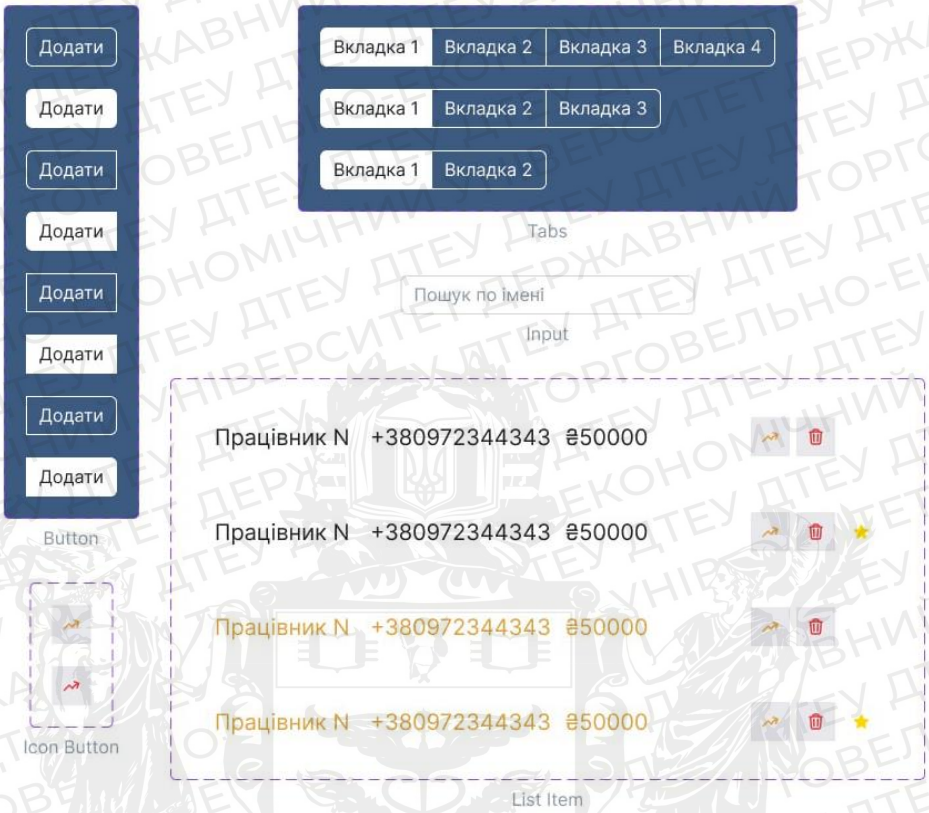


Рис.3.2 Компоненти в Figma

Для додатку з представлених на ринку бібліотек я вибрав React.

React - це JavaScript бібліотека для розробки інтерфейсів користувача, яка зарекомендувала себе як один з найпопулярніших інструментів для створення веб-додатків. Хоча важко сказати, що React є кращим за всі інші інструменти, оскільки кожен має свої сильні сторони, проте React має деякі вагомі переваги, які заслуговують уваги.

Віртуальний DOM: React використовує віртуальний DOM, що дозволяє ефективно оновлювати лише необхідні частини сторінки при зміні даних, а не повністю перебудовувати всю сторінку. Це забезпечує кращу продуктивність і швидкість реагування додатка.

Компонентна архітектура: React пропонує компонентну модель розробки, де користувачі можуть будувати складні інтерфейси, розбиваючи їх

на малий набір перевикористовуваних компонентів. Це сприяє чистоті коду, полегшує його розуміння і підтримку.

Єдиний напрямок потоку даних: React рекомендує використовувати єдиний напрямок потоку даних (однонаправлене зв'язування даних), що спрощує відлагодження та прогнозованість взаємодії між компонентами.

Розширюваність: Багато сторонніх бібліотек і плагінів підтримують React, що дозволяє розширити його функціональність та можливості. Крім того, існує велика спільнота розробників React, що сприяє обміну знаннями та забезпечує розширену підтримку.

Також слід використовувати бібліотеки для зручності і швидкої зміни функціоналу для різної автоматизації задач.

Styled Components - це бібліотека для стилізації компонентів в React, яка дозволяє описувати стилі прямо в коді JavaScript з використанням синтаксису схожого на CSS. Хоча Styled Components не можна однозначно назвати "кращим" за всі інші аналоги, вона має деякі переваги, які зробили її дуже популярною серед розробників.

Причини чому ми вибираємо Styled Components:

Компонентно-орієнтований підхід: Styled Components пропонує підхід до стилізації, який пов'язує стилі безпосередньо з компонентами. Кожен компонент може мати свої унікальні стилі, що полегшує розробку і підтримку. Він дозволить нам визначати стилі для компонентів в одному місці, що зробить код більш організованим та зрозумілим.

Вбудована область видимості: Кожен стилізований компонент в Styled Components має свою вбудовану область видимості стилів. Це означає, що стилі, визначені для одного компонента, не конфліктують зі стилями інших компонентів. Це дозволяє легко управляти стилізацією компонентів і уникнути непотрібних конфліктів.

Динамічні стилізація: Styled Components дозволяє використовувати JavaScript для визначення динамічних стилів. Ви можете використовувати умовні оператори, змінні та вирази, щоб змінювати стилі в залежності від умов або даних. Це робить стилізацію більш гнучкою і потужною.

Автоматичне присвоєння класів: Styled Components автоматично генерує унікальні імена класів для стилізованих компонентів. Це дозволяє уникнути конфліктів імен класів та зберігає розмір стилів невеликим.

Інтеграція з React: Styled Components розроблено спеціально для використання з React і має гармонійну інтеграцію з цією бібліотекою. Ми зможемо безпосередньо використовувати стилізовані компоненти разом з компонентами React, що дозволяє легко стилізувати наш додаток.

3.2 Архітектура та структура системи обліку кадрів

Використання мови UML (Unified Modeling Language) для моделювання автоматизованої системи обліку кадрів має декілька обґрунтувань і переваг.

Стандартизація: UML є широко використовуваною мовою моделювання, яка має встановлені стандарти. Це означає, що команда розробників, що працює над системою, зможе легко спілкуватися і розуміти один одного завдяки спільному мовленню, яке надає UML.

Візуалізація: UML надає графічний спосіб відображення складних системних структур, включаючи компоненти системи, їх взаємозв'язки та поведінку. Це дозволяє команді проектувальників та зацікавленим сторонам легко розуміти структуру та функціональність системи обліку кадрів.Аналіз та проектування:

Використання UML дозволяє проводити аналіз та проектування системи перед її фактичною реалізацією. За допомогою діаграм UML, таких

як діаграми класів, діаграми послідовностей та діаграми станів, можна моделювати взаємодію між різними компонентами системи та описати потоки даних та операцій. Це дозволяє ідентифікувати потенційні проблеми та помилки ще до реалізації системи та забезпечити більш ефективний процес розробки.

Документація: UML дозволяє створювати докладні та структуровані діаграми, які можуть використовуватися як документація для системи. Це спрощує розуміння системи та полегшує майбутнє супроводження, підтримку та розширення системи обліку кадрів.

Діаграма Use Case (рис.3.3) дозволяє ідентифікувати та візуалізувати основні функції або операції, які система має виконувати. Вона допомагає зрозуміти, які актори (користувачі, системи або зовнішні сутності) взаємодіють з системою та які вимоги вони мають.

Крім того діаграма Use Case надає можливість описати різні сценарії використання системи. Кожен Use Case представляє окрему функцію або операцію, яку актор може виконати в системі. Це дозволяє детально описати послідовність подій та дій, які відбуваються між акторами та системою в кожному конкретному сценарії.

Діаграма Use Case допомагає визначити межі та обсяг системи обліку кадрів. Вона дозволяє виокремити, які функції та операції входять до системи та які зовнішні елементи взаємодіють з системою. Це полегшує розуміння того, як система інтегрується з іншими системами або процесами в організації.

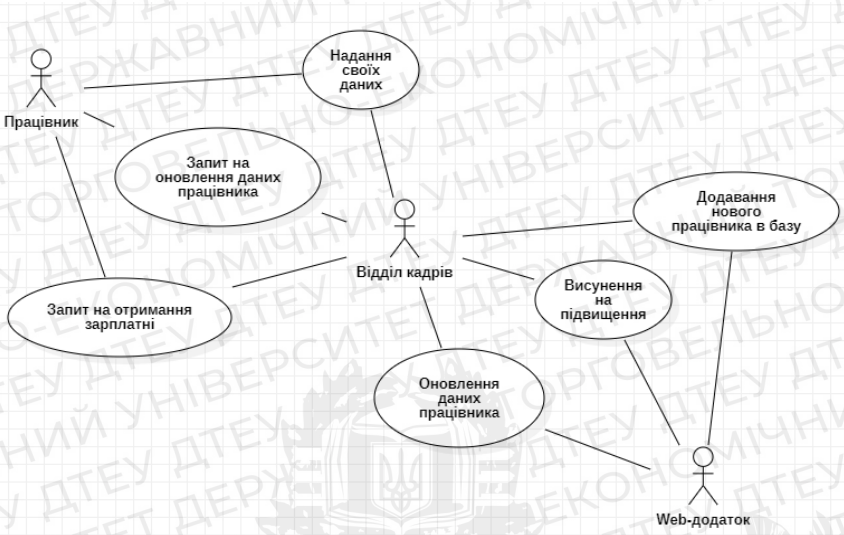


Рис.3.3 Use case діаграма

Діаграма станів (Рис.3.4)дозволяє візуалізувати різні стани, в яких може перебувати система обліку кадрів. Кожен стан представляє певну фазу, умову або поведінку системи. Це полегшує розуміння того, як система реагує на зміни у своєму оточенні або вхідні події.

Опис переходів між станами: Діаграма станів дозволяє описати переходи між різними станами системи та умови, які спричиняють ці переходи. Це допомагає зрозуміти, як система змінює свій стан відповідно до вхідних подій або змін у своєму середовищі.

Моделювання поведінки системи: Діаграма станів дозволяє моделювати поведінку системи обліку кадрів у різних станах. Вона може включати дії, події, умови та відповіді системи на ці події. Це дозволяє розробникам та зацікавленим сторонам зрозуміти, як система реагує на певні події та які дії вона виконує у кожному стані.

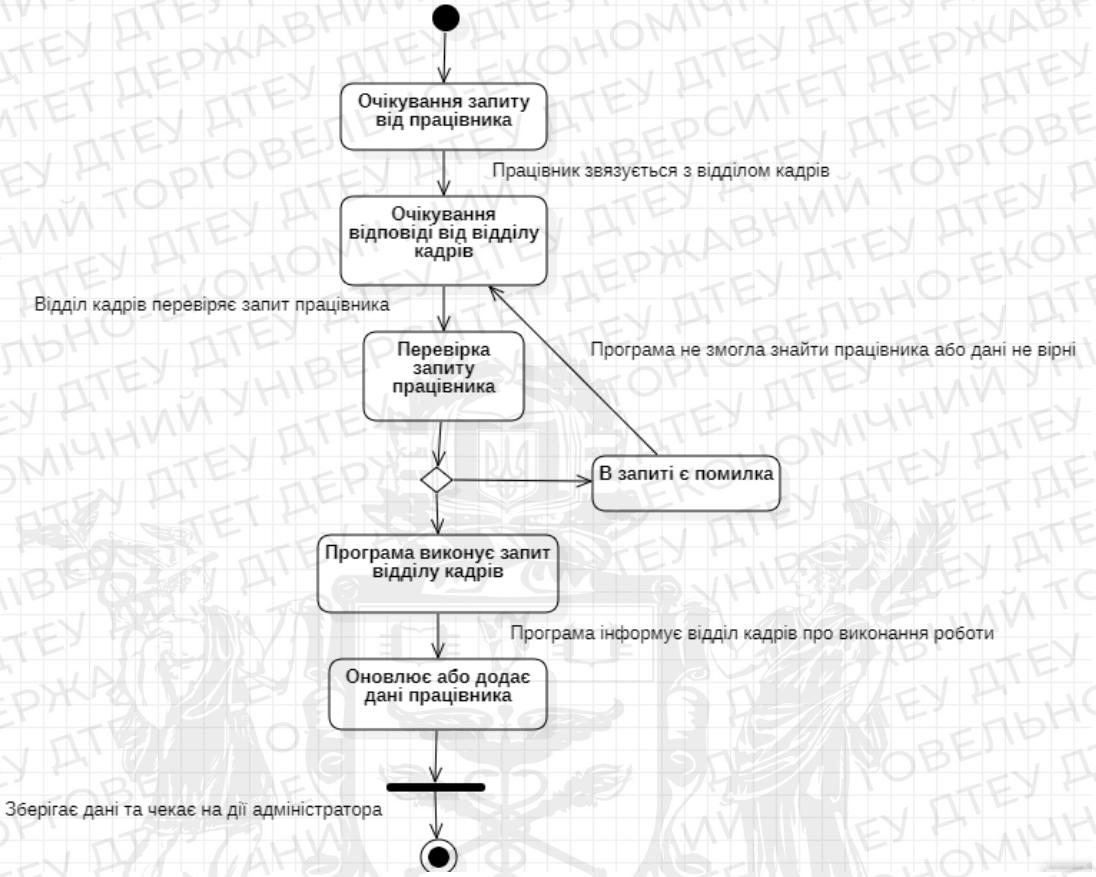


Рис.3.4 Діаграма станів

3.3 Реалізація системи обліку кадрів

На початку створення додатку використовується код, який відповідає за рендеринг головного компонента додатку у кореневому елементі з ідентифікатором 'root'. Для цього ми використовуємо React і ReactDOM з бібліотеки React.

Спочатку ми імпортуємо React із бібліотеки React, а також ReactDOM з бібліотеки react-dom/client. ReactDOM використовується для взаємодії з DOM і здійснення рендерингу React компонентів.Далі імпортується компонент App

з './App'. Цей файл містить визначення головного компонента додатку. Компонент App буде відповідальний за відображення всіх інших компонентів та управління логікою додатку.

Також ми імпортуємо стилізацію з './index.css'. У цьому файлі містяться стилі CSS, які будуть застосовані до компонентів додатку.

Далі ми створюємо кореневий вузол (root) за допомогою ReactDOM.createRoot(). Ми використовуємо createRoot() замість render() для підтримки Concurrent Mode, який дозволяє більш ефективно керувати рендерингом React компонентів.

Нарешті, ми викликаємо метод render() на кореневому вузлі (root) для рендерингу головного компонента App. Обгортка <React.StrictMode> навколо компонента App вмикає строгий режим React, що допомагає виявляти потенційні проблеми в додатку.

Цей код є основою для початку роботи з React і встановлює зв'язок між головним компонентом і кореневим елементом DOM.

```
import React from 'react'
import ReactDOM from 'react-dom/client'

import App from './App'

import './index.css'

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

Рис.3.5 Блок для початку роботи з React

У цьому кодовому блоку оголошується початковий стан (state) компонента.(Рис.3.6)

Стан має наступні властивості:

"employees": це масив об'єктів, що представляють працівників.

Кожен об'єкт має унікальний ідентифікатор ("id"), ім'я ("name"), номер телефону ("phone"), заробітну плату ("salary") та флаг ("like"), що вказує, чи сподобався працівник.

"queryString": це рядок запиту, який потенційно використовується для фільтрації працівників за певною умовою. Він може містити текст для пошуку або фільтрації даних.

"filters": це масив об'єктів, що представляють фільтри для працівників.

Кожен об'єкт має мітку ("label"), тип ("type"), флаг ("isActive"), який вказує, чи є фільтр активним, функцію-предикат ("predicate"), яка визначає, чи задовольняє працівник умову фільтру, і значення ("doFilter"), яке потенційно вказує на поточний вибраний фільтр.

Цей стан використовується для збереження даних про працівників, рядку запиту та фільтрах, які застосовуються до списку працівників.

Він може бути використаний для фільтрації, сортування або відображення даних у компоненті. Наприклад, ви можете використовувати цей стан для відображення списку працівників, фільтрування їх за рядком запиту, або застосування вибраних фільтрів до списку.

```
state = {
  employees: [
    {
      id: 1,
      name: 'Максим Р.',
      phone: '380972344343',
      salary: 50000,
      like: true,
    },
    {
      id: 2,
      name: 'Олександра Р.',
      phone: '380638732101',
      salary: 50000,
      increase: true,
    },
  ],
  queryString: '',
  filters: [
    { label: 'Всі працівники', type: 'all', isActive: true },
    {
      label: 'Відмічені',
      type: 'like',
      predicate: e => e.like,
    },
    {
      label: 'На підвищення',
      type: 'increase',
      predicate: e => e.increase,
    },
    {
      label: 'ЗП > 1000',
      type: 'moreThen',
      predicate: e => e.salary > 1000,
    },
  ],
  doFilter: null,
}
```

Рис.3.6 Розробка списку та фільтрації працівників

Далі розробляємо метод onPhoneChange, який виконується при зміні значення телефонного номеру працівника. (Рис.3.7)

Давайте розберемо його:

Спочатку створюється копія масиву employees зі стану (this.state.employees) за допомогою оператора ... (розширення масиву). Це зроблено для створення незалежної копії масиву.

Знаходиться індекс об'єкта item у масиві employees за допомогою методу indexOf(). item - це працівник, чий номер телефону був змінений.

Створюється копія об'єкта item за допомогою оператора .... Це робиться для створення незалежної копії об'єкта працівника.

Використовуючи подію e (зміна введеного значення), отримується посилання на вхідний елемент input за допомогою деконструкції об'єкту currentTarget з події.

Змінюється значення phone об'єкта працівника employees[index] на нове значення, яке отримане з input.value. Застосовується регулярний вираз /\\d+/ для отримання лише числових значень з рядка, якщо вони присутні, за допомогою методу match().

Якщо числове значення не знайдене, phone встановлюється на порожню рядок ('').

Змінюється стан employees на оновлений масив employees за допомогою setState(). Таким чином, змінюється значення телефонного номеру працівника, і компонент буде перерендерений з оновленими даними.

Цей метод використовується для оновлення значення телефонного номеру працівника в стані компонента, коли відбувається зміна відповідного вхідного поля.

```
onPhoneChange = (e, item) => {
  const employees = [...this.state.employees]
  const index = employees.indexOf(item)
  employees[index] = { ...item }

  const { currentTarget: input } = e
  employees[index].phone = +input.value.match(/\d+/)?.[0] || ''

  this.setState({ employees })
}
```

Рис.3.7 Блок редагування телефону працівника

Далі розробимо два методи: onDelete і onAdd, які використовуються для видалення та додавання працівників в стан компонента.(Рис.3.8)

Розглянемо їх детальніше:

Метод onDelete: Приймає параметр id, який відповідає ідентифікатору працівника, який потрібно видалити.

Створюється копія масиву employees зі стану, використовуючи метод filter(). Усі працівники, чий id не співпадає з переданим id, залишаються в масиві, а працівник з відповідним id видаляється.

Оновлений масив employees встановлюється в стані компонента за допомогою setState(). Таким чином, працівник з переданим id буде видалений зі списку працівників, і компонент буде перерендерений з оновленими даними.

Метод onAdd: Приймає параметри name, phone і salary, що відповідають імені, телефону та заробітній платі нового працівника.

Створюється новий об'єкт employee з властивостями id, name, phone і salary. Значення id отримується з збільшенням змінної this.maxId, яка потенційно зберігає останнє значення id працівника.

Створюється копія масиву employees зі стану за допомогою оператора .... Потім до цього масиву додається новий об'єкт employee.

Оновлений масив employees встановлюється в стані компонента за допомогою setState(). Таким чином, новий працівник буде доданий до списку працівників, і компонент буде перерендерений з оновленими даними.

Ці методи використовуються для взаємодії зі списком працівників, забезпечуючи можливість видалення та додавання нових працівників до стану компонента.

```
onDelete = id => {
    const employees = this.state.employees.filter(e => e.id !== id)
    this.setState({ employees })
  }

  onAdd = (name, phone, salary) => {
    const employee = { id: this.maxId++, name, phone, salary }

    const employees = [...this.state.employees]
    employees.push(employee)

    this.setState({ employees })
  }
```

Рис.3.8 Форма додавання нового працівника.

Цей код представляє метод onFilterChange, який викликається при зміні активного фільтру.(Рис.3.9)

Давайте розберемо його крок за кроком:

Створюється копія масиву filters зі стану (this.state.filters) за допомогою оператора .... Це робиться для збереження незалежного екземпляра масиву.

Знаходиться активний фільтр, шукаючи перший фільтр, для якого властивість isActive має значення true, за допомогою методу find().
Якщо активний фільтр співпадає з переданим фільтром (active === filter), то ніяких змін не вноситься, і метод повертається.

Знаходиться індекс активного фільтра в масиві filters за допомогою методу indexOf(). Створюється копія активного фільтра за допомогою оператора .... Це робиться для збереження незалежного екземпляра об'єкта фільтра.Знаходиться індекс переданого фільтру в масиві filters за допомогою методу indexOf().

Створюється копія переданого фільтру за допомогою оператора .... Це робиться для збереження незалежного екземпляра об'єкта фільтра.

Змінюється властивість isActive нового фільтра на протилежне значення від значення фільтра, який був натиснутий. Це означає, що новий фільтр буде активним, якщо він був неактивним, або навпаки.

Оновлений масив filters встановлюється в стані компонента за допомогою setState(). Таким чином, активний фільтр оновлюється, а фільтри у масиві filters оновлюються відповідно.

Властивість doFilter стану компонента встановлюється на предикат фільтра (filter.predicate). Це дозволяє використовувати функцію-предикат для фільтрації списку працівників згідно з вибраним фільтром.

Цей метод використовується для зміни активного фільтру та виконання відповідних дій при зміні фільтра.

```
onFilterChange = filter => {
    const filters = [...this.state.filters]
    const active = filters.find(f => f.isActive)

    if (active === filter) return

    const activeIndex = filters.indexOf(active)
    filters[activeIndex] = { ...active, isActive: false }

    const index = filters.indexOf(filter)
    filters[index] = { ...filter }
    filters[index].isActive = !filter.isActive

    this.setState({ filters, doFilter: filter.predicate })
  }
```

Рис.3.9 Форму додавання нового працівника

Цей код представляє компонент Header, який відображає заголовок і інформацію про облік працівників.(Рис.3.10)

Давайте розберемо його по частинах:

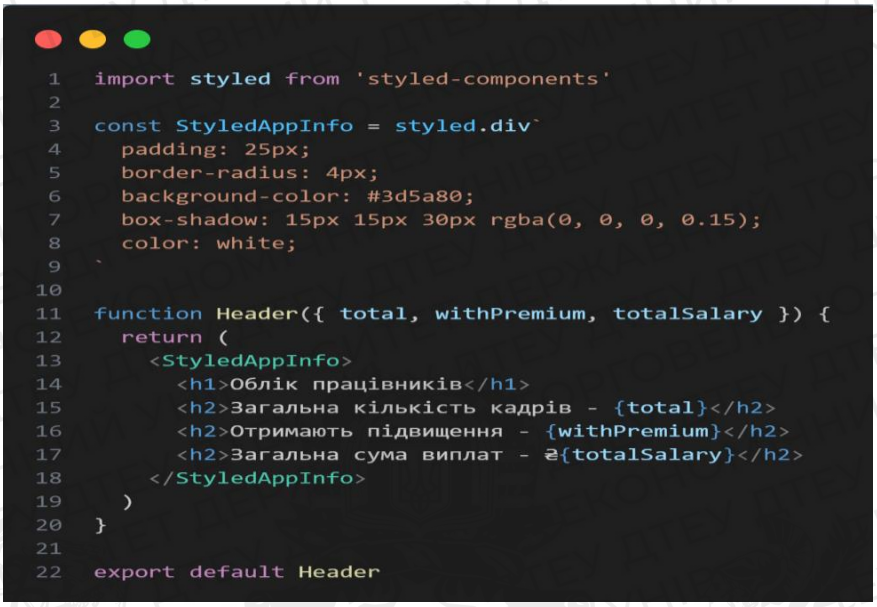
Імпортується бібліотека styled-components, яка дозволяє стилізувати компоненти за допомогою CSS в JavaScript. Створюється стилізований компонент StyledAppInfo за допомогою функції styled.div. Цей компонент відображає блок зі стилізованим виглядом. Використовуються CSS властивості, такі як padding, border-radius, background-color, box-shadow і color, для встановлення візуального вигляду блоку.

Оголошується функція Header, яка приймає об'єкт з трьома властивостями: total, withPremium і totalSalary. В тілі функції Header повертається JSX код, який представляє розмітку заголовка і інформації про облік працівників. Компонент StyledAppInfo використовується для обгортання цього JSX коду, надаючи йому стилізований вигляд.

Використовуються елементи <h1>, <h2> для відображення заголовків і текстової інформації. Значення змінних total, withPremium і totalSalary вставляються в JSX код за допомогою фігурних дужок {}.

Компонент Header експортується за допомогою export default, що дозволяє його використовувати в інших частинах додатку.

Цей компонент використовується для відображення заголовку і статистичної інформації про облік працівників.



```
import styled from 'styled-components'

const StyledAppInfo = styled.div`
  padding: 25px;
  border-radius: 4px;
  background-color: #3d5a80;
  box-shadow: 15px 15px 30px rgba(0, 0, 0, 0.15);
  color: white;
`

function Header({ total, withPremium, totalSalary }) {
  return (
    <StyledAppInfo>
      <h1>Облік працівників</h1>
      <h2>Загальна кількість кадрів - {total}</h2>
      <h2>Отримають підвищення - {withPremium}</h2>
      <h2>Загальна сума виплат - ₴{totalSalary}</h2>
    </StyledAppInfo>
  )
}

export default Header
```

Рис.3.10 Функція відображення заголовку

Цей код представляє компонент EmployeesFilter, який відображає фільтри для працівників.(Рис.3.11)

Давайте розберемо його по частинах:

Імпорт компонентів Filter та Button з шляху ../common. Ці компоненти, ймовірно, визначені в іншому файлі і експортуються для використання в даному компоненті.

Оголошення функції EmployeesFilter, яка приймає два аргументи: filters - масив фільтрів і onFilterChange - функція, яка викликається при зміні фільтра.

В тілі функції EmployeesFilter повертається JSX код, який представляє фільтри працівників. Компонент Filter використовується для обгортання цього JSX коду, створюючи контейнер для фільтрів.За допомогою методу map() масив filters перебирається, і для кожного фільтра створюється компонент Button. Кожен фільтр представлений об'єктом з властивостями, такими як label і isActive.

Компонент Button отримує наступні властивості: key - унікальний ключ для елемента, що дозволяє React ефективно оновлювати список елементів.

active - встановлюється значення true, якщо фільтр активний, або false, якщо фільтр неактивний. Це дозволяє відображати активний стан фільтра за допомогою відповідного стилю.

onClick - задається функція, яка викликається при кліці на кнопку фільтра.

Виклик функції onFilterChange(filter) передає вибраний фільтр для обробки вищим рівнем компонентів.

Вміст кнопки - {filter.label} - відображається мітка фільтра.

Компонент EmployeesFilter експортується за допомогою export default, що дозволяє його використовувати в інших частинах додатку.

Цей компонент відображає набір фільтрів для працівників і надає можливість вибору фільтра за допомогою кнопок. При виборі фільтра викликається функція onFilterChange, яка повідомляє вищим рівнем компонентів про зміну фільтрації.

```jsx
import { Filter, Button } from '../common'

function EmployeesFilter({ filters, onFilterChange }) {
  return (
    <Filter>
      {filters.map((filter, i) => (
        <Button
          key={i}
          active={filter.isActive}
          onClick={() => onFilterChange(filter)}
        >
          {filter.label}
        </Button>
      ))}
    </Filter>
  )
}

export default EmployeesFilter
```

Рис.3.11 Фільтр працівників

Цей код представляє класовий компонент EmployeesAddForm, який відповідає за форму додавання нового працівника.(Рис.3.12)

Давайте розберемо його по частинах:

Оголошення класу EmployeesAddForm та розширення його від класу AppForm. Це означає, що EmployeesAddForm успадковує функціональність з AppForm та може використовувати його методи та властивості.

Визначення стану компонента state, який містить об'єкт data з властивостями name, phone та salary. Початкові значення цих властивостей встановлюються на пусті рядки.

Оголошення методу validate, який перевіряє, чи всі поля форми заповнені. Він перевіряє, чи є хоча б одне поле з пустим значенням у властивості data. Якщо є пусте поле, повертається значення false, інакше - true.

Оголошення методу doSubmit, який виконується при надсиланні форми. Спочатку перевіряється валідність форми за допомогою методу validate. Якщо форма не валідна, функція просто повертається. У протилежному випадку, дані про ім'я, номер телефону та заробітну плату отримуються зі стану data, і викликається функція onAdd з передачею цих даних як аргументів. Після цього стан data обнуляється, встановлюючи всі властивості на пусті рядки.

Реалізація методу render, який відповідає за відображення компонента. Він повертає JSX код, який представляє форму додавання працівника.
Використовується компонент StyledAddForm, який, ймовірно, визначений в іншому файлі та має відповідні стилі. Заголовок <h3> відображає текст "Додати працівника". Форма має клас "add-form d-flex" і оброблюється методом onSubmit, який, ймовірно, визначений в AppForm.

Використовуються методи renderInput та renderSubmitButton з успадкованого класу AppForm, які генерують поля вводу та кнопку відповідно. Кожен метод отримує властивості, такі як name, label, className та type, які використовуються для відображення відповідних полів та налаштувань.

Компонент EmployeesAddForm експортується за допомогою export default, що дозволяє його використовувати в інших частинах додатку.

Отже, цей компонент представляє форму для додавання нового працівника, яка містить поля для введення його імені, номера телефону та заробітної плати. При надсиланні форми дані про працівника передаються вищим рівнем компонентів за допомогою функції onAdd.

```
class EmployeesAddForm extends AppForm {
  state = {
    data: {
      name: '',
      phone: '',
      salary: '',
    },
  }

  validate = () => {
    const { data } = this.state
    const hasEmpty = !!Object.keys(data).find(
      key => data[key].trim().length === 0
    )

    return hasEmpty ? false : true
  }

  doSubmit = () => {
    if (!this.validate()) return

    const { name, phone, salary } = this.state.data
    this.props.onAdd(name, phone, salary)

    this.setState({ data: { name: '', phone: '', salary: '' } })
  }

  render() {
    return (
      <StyledAddForm>
        <h3>Додати працівника</h3>
        <form className="add-form d-flex" onSubmit={this.onSubmit}>
          {this.renderInput('name', "Ім'я", 'new-post-label')}
          {this.renderInput(
            'phone',
            'Номер телефону',
            'new-post-label',
            'number'
          )}
          {this.renderInput('salary', 'ЗП в ₴', 'new-post-label', 'number')}
          {this.renderSubmitButton('Додати')}
        </form>
      </StyledAddForm>
    )
  }
}

export default EmployeesAddForm
```

Рис.3.12 Форма додавання нового працівника

На останок отримаємо додаток для ведення обліку кадрів є веб-додатком, побудованим за допомогою бібліотеки React та пакету create-react-app. Додаток використовує модульний підхід до розробки, де кожен функціональний блок представлений відповідним модулем.(Рис.3.13)

Додаток має декілька ключових компонентів. Компонент App є головним компонентом додатку, в якому відбувається відображення і керування станом додатку. Компонент Header відповідає за відображення заголовку та інформації про кадри, такі як загальна кількість, кількість отримуючих підвищення та загальна сума виплат.

Компонент EmployeesList відображає список працівників з можливістю фільтрації. Компонент EmployeesFilter відповідає за відображення фільтрів та обробку їх вибору. Компонент EmployeesAddForm представляє форму для додавання нового працівника.

Стан додатку зберігається в компоненті App і містить список працівників, рядок запиту, список фільтрів та функцію фільтрації. Додаток також має різні функції для редагування списку працівників, такі як додавання нового працівника та видалення працівника за його ідентифікатором. Для стилізації компонентів використовується пакет styled-components, який дозволяє використовувати CSS в стилі JavaScript для створення красивих та реагуючих компонентів.

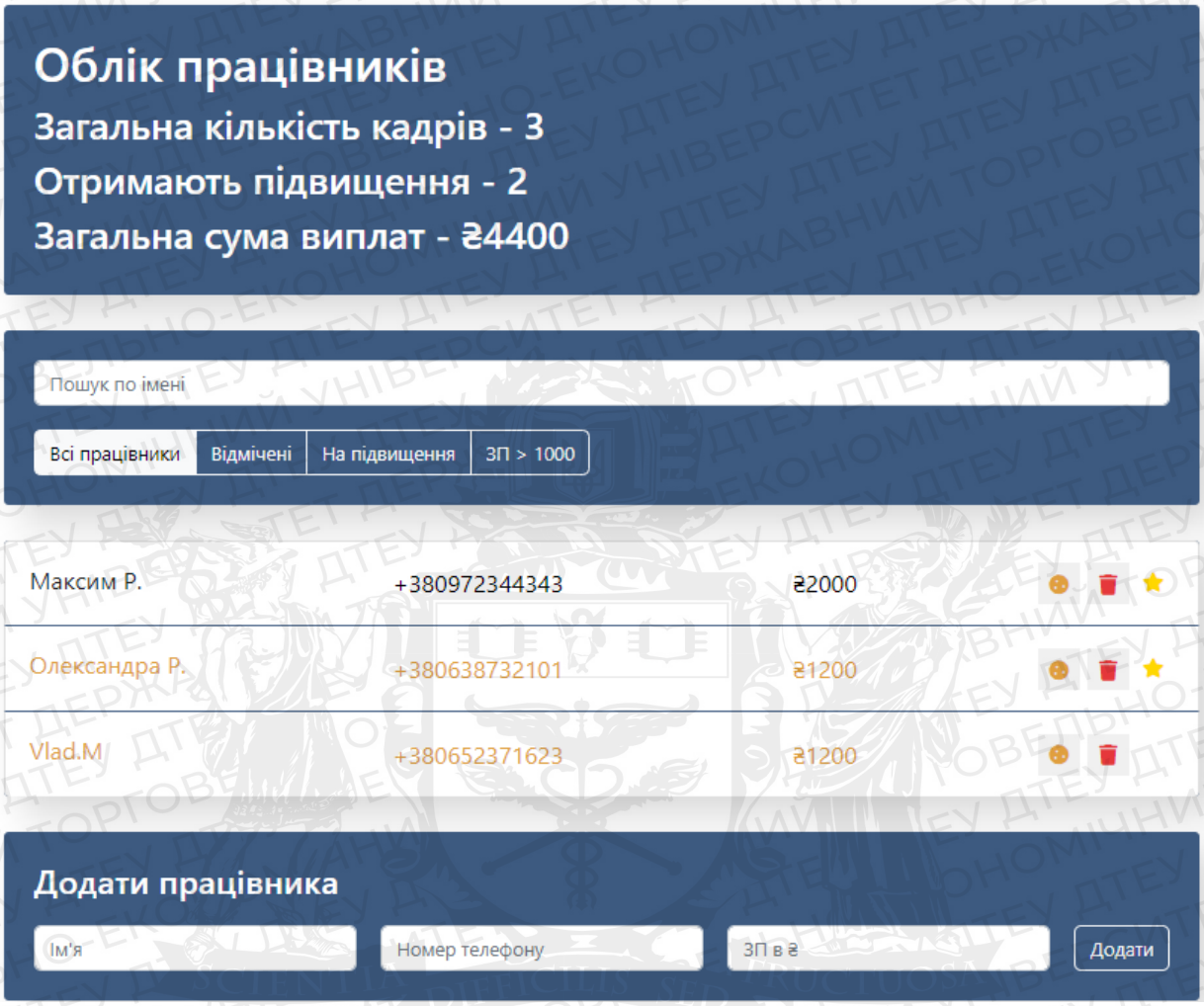


Рис.3.13 Готовий додаток для обліку кадрів

Узагальнюючи, цей додаток на React забезпечує ведення обліку кадрів, де користувач може переглядати список працівників, вибирати та застосовувати фільтри, додавати нових працівників і видаляти існуючих. Він має зручний інтерфейс та забезпечує ефективне управління кадрами у компанії.

Висновки та результати

У випускній кваліфікаційній роботі представлено результати теоретичних і прикладних досліджень, Web-додаток для обліку кадрів, розроблений в рамках даного проекту, є результатом аналізу та удосконалення існуючих систем обліку кадрів на підприємстві. Розглянуті теоретичні основи автоматизованої системи обліку кадрів, її значення для підприємства та основні завдання та функції.

Аналіз показав проблеми та недоліки існуючих систем обліку кадрів, а також виявлено проблематику, з якою стикається підприємство. В результаті встановлені вимоги до нової системи обліку кадрів.

Для проектування та розробки системи були вибрані певні технологічні рішення та інструменти. Web-додаток розроблений з використанням бібліотеки React та пакету create-react-app. Для стилізації компонентів використовується пакет styled-components.

Архітектура та структура системи обліку кадрів були розроблені з урахуванням потреб підприємства. Додаток складається з різних компонентів, таких як заголовок, фільтри, список працівників та форма додавання працівника. Кожен компонент виконує певну функціональність для забезпечення ефективного обліку кадрів.

Узагальнюючи, розроблений Web-додаток для обліку кадрів забезпечує автоматизовану систему обліку кадрів, що відповідає вимогам підприємства. Він дозволяє вести облік працівників, також їх фільтрацію за різними критеріями, додавати та видаляти працівників. Застосування сучасних технологій та інструментів дозволяє забезпечити зручний та ефективний інтерфейс користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизація технологічних процесів і систем автоматичного керування / Барало О.В., Самойленко П.Г., Гранат С.Є., Ковальов В.О. К.: Аграрна освіта, 2010. – 557 с

2. Батюк А.Е. та ін.. Інформаційні системи в менеджменті: Навчальний посібник. – Львів: НУ «Львівська політехніка», 2004.

3. Виноградський М.Д., Бєляєва С.В., Виноградська А.М., Шканова О.М. Управління персоналом. Навч. пос. для ВНЗ.– К.:ЦУЛ.- 2006.- 500 с.

4. Гавриш, І. В., & Карпенко, О. В. (2019). Значення обліку кадрів у сучасних умовах. Науковий вісник Полісся, 1(13), 81-85 [2].

5. Журавльов, В.М. (2019). Інформаційні системи в управлінні персоналом та економіки праці. Методичні вказівки до вивчення курсу для студентів спеціальності 051 «Економіка», спеціалізації «Управління персоналом і економіка праці». Кропивницький: ЦНТУ, с. 136. [1].

6. Козак О.Л. Опорний конспект лекцій з курсу ―Аналіз вимог до програмного забезпечення‖ для студентів напрямку підготовки ―Програмна інженерія‖ / О.Л. Козак. – Тернопіль, 2011. – 56 с. [5].

7. Конспект лекцій з кредитного модуля «Сучасні методи проектування» / Уклад.: І.О. Казак. – К.: НТУУ «КПІ ім. Ігоря Сікорського», 2017. – 65 с

8. Конспект лекцій з курсу "Організація кадрової роботи" для підготовки фахівців освітньо-кваліфікаційного рівня "бакалавр", денної та заочної форми навчання. Тернопіль, 2013 с. 136 [4].

9. Літинська, В.А. & Шевчук, А.В. "Кадрове діловодство на підприємстві: Personnel Records at the Enterprise." Хмельницький національний університет [3].

10. Новітні тенденції розвитку управління підприємствами : монографія / [Федонін О.С., Швиданенко Г.О., Лаврененко В.В. та ін.]. — К. : КНЕУ, 2011. — 257, [7] с.
11. Основи програмної інженерії: навчальний посібник / Є. О. Зайцев – К.: КНТЕУ, 2017. – с. [6].
12. Попович М. Г.,Ковальчук О. В. Теорія фатоматичного керування: Підручник. -2-ге вид., перероб. І доп. - К,:Либідь,2007.- 656 с.
13. Посібник: знайомство з React [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/tutorial/tutorial.html.
14. Самсонов В. Методи та засоби Інтернет-технологій: навч. посібник/ В. Самсонов, А. Єрохін, Х.: Компанія СМІТ, 2008. – 264 с
15. Тяннікова К.П. Соціально-економічна сутність персоналу підприємства / К.П. Тяннікова, С.В. Березюк // Економічні проблеми розвитку аграрного виробництва в регіоні. — 2010. – Вип. 6. – С. 78–82.
16. Чикуркова А. Д. Стратегія управління / А. Д. Чикуркова. – Кам’янець-Подільський: Видавець ПП Зволейко Д. Г., 2010. – 428 с.
17. Ядворська О. Оцінка персоналу в системі ефективного управління // Економіка та держава. – 2009. - №1., с.60-63.

ДОДАТОК

Програмний код реалізації Web-додатку

```
import { Component } from 'react'
import styled from 'styled-components'

import Header from '../components/Header'
import FiltersPanel from '../components/FiltersPanel'
import EmployeeListItem from '../components/EmployeeListItem'
import EmployeesAddForm from '../components/EmployeesAddForm'

import ListGroup from '../components/common/ListGroup'

const StyledApp = styled.div`
 margin: 50px auto;
 max-width: 1000px;
`

export class App extends Component {
 state = {
  employees: [
   { id: 1, name: 'Maximka R.', salary: 1, like: true },
   { id: 2, name: 'Alexandra R.', salary: 100000, increase: true },
  ],
  queryString: '',
  filters: [
   { label: 'All employees', type: 'all', isActive: true },
   {
    label: 'Favorites',
    type: 'like',
    predicate: e => e.like,
   },
   {
```

```
      label: 'For rise',
      type: 'increase',
      predicate: e => e.increase,
    },
    {
      label: 'Salary > 1000',
      type: 'moreThen',
      predicate: e => e.salary > 1000,
    },
  ],
  doFilter: null,
}

maxId = 4

onSalaryChange = (e, item) => {
  const employees = [...this.state.employees]
  const index = employees.indexOf(item)
  employees[index] = { ...item }

  const { currentTarget: input } = e
  employees[index].salary = +input.value.match(/\d+/)?.[0] || ''

  this.setState({ employees })
}

#toggleEmployeeProp = (item, propName) => {
  const employees = [...this.state.employees]
  const index = employees.indexOf(item)
  employees[index] = { ...item }
  employees[index][propName] = !item[propName]

  this.setState({ employees })
}
```

```
onIncrease = item => {
 this.#toggleEmployeeProp(item, 'increase')
}

onLike = item => {
 this.#toggleEmployeeProp(item, 'like')
}

onDelete = id => {
 const employees = this.state.employees.filter(e => e.id !== id)
 this.setState({ employees })
}

onAdd = (name, salary) => {
 const employee = { id: this.maxId++, name, salary }

 const employees = [...this.state.employees]
 employees.push(employee)

 this.setState({ employees })
}

onSearch = e => {
 const { currentTarget: input } = e
 this.setState({ queryString: input.value })
}

onFilterChange = filter => {
 const filters = [...this.state.filters]
 const active = filters.find(f => f.isActive)

 if (active === filter) return
```

```
  const activeIndex = filters.indexOf(active)
  filters[activeIndex] = { ...active, isActive: false }

  const index = filters.indexOf(filter)
  filters[index] = { ...filter }
  filters[index].isActive = !filter.isActive

  this.setState({ filters, doFilter: filter.predicate })
 }

 #getFilteredData = () => {
  const { employees, queryString, doFilter } = this.state

  const searched = queryString
   ? employees.filter(e =>
     e.name.toLowerCase().includes(queryString.toLowerCase())
    )
   : employees

  const filtered = doFilter ? searched.filter(doFilter) : searched

  const withPremium = filtered.filter(e => e.increase).length

  return {
   data: filtered,
   total: filtered.length,
   withPremium,
  }
 }

 render() {
  const { total, data: employees, withPremium } = this.#getFilteredData()
  const { queryString, filters } = this.state
```

```
    return (
      <StyledApp>
        <Header total={total} withPremium={withPremium} />
        <FiltersPanel
          queryString={queryString}
          onSearch={this.onSearch}
          filters={filters}
          onFilterChange={this.onFilterChange}
        />
        <ListGroup
          data={employees}
          component={EmployeeListItem}
          onSalaryChange={this.onSalaryChange}
          onIncrease={this.onIncrease}
          onLike={this.onLike}
          onDelete={this.onDelete}
        />
        <EmployeesAddForm onAdd={this.onAdd} />
      </StyledApp>
    )
  }
}

export default App
```