

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ
ЗАРОБІТНОЇ ПЛАТИ
PAYROLL CALCULATION SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212сп
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

РУДЕНКО І.І.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

РУДЕНКА Іллі Ігоровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для розрахунку заробітної плати. Payroll Calculation Software

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області розрахунку заробітної плати. 2. Проєктування програмного забезпечення розрахунку заробітної плати. 3. Опис програмного забезпечення. 4. Рекомендації із впровадження програмного забезпечення розрахунку заробітної плати.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Ілля РУДЕНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Євгеній ГОФМАН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 102 с., 17 табл., 18 рис., 3 дод., 23 джерела.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЗАРОБІТНА ПЛАТА, АВТОМАТИЗАЦІЯ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, PYTHON, MYSQL, ВЕБЗАСТОСУНОК.

Об'єкт дослідження — процес розроблення програмного забезпечення для автоматизації розрахунку заробітної плати.

Предмет дослідження — методи та інструменти розробки вебзастосунку для обліку робочого часу й нарахування заробітної плати.

Мета роботи — підвищення ефективності процесу розрахунку заробітної плати працівників шляхом розроблення та впровадження веборієнтованої автоматизованої системи.

Матеріали, методи та технічні засоби: клієнт-серверна архітектура, мова програмування Python, фреймворк Django, система управління базами даних MySQL.

Результати. Створено веборієнтований програмний застосунок для автоматизації розрахунку заробітної плати з урахуванням українського законодавства, який забезпечує облік відпрацьованого часу, нарахування заробітної плати і податків, зберігання історії розрахунків та управління користувачами.

Висновки. Розроблено програмне забезпечення для автоматизації розрахунку заробітної плати, яке підвищує точність, надійність і прозорість процесів нарахування заробітної плати в організаціях, спрощує облік робочого часу та забезпечує відповідність розрахунків законодавчим нормам.

Галузь використання — автоматизація процесів обліку та розрахунку заробітної плати на підприємствах різного розміру та форм власності.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor:
102 pages, 17 tables, 18 figures, 3 appendixes, 23 sources.

SOFTWARE, PAYROLL, AUTOMATION, CLIENT-SERVER
ARCHITECTURE, PYTHON, MYSQL, WEB APPLICATION.

Object of study is the process of developing software for payroll calculation automation.

Subject of study are methods and tools for developing a web application for time tracking and payroll calculation.

The purpose of the work is to improve the efficiency of the employee payroll calculation process by developing and implementing a web-oriented automated system.

Materials, methods, and tools: client-server architecture, Python programming language, Django framework, MySQL database management system.

Results. A web-oriented software application has been created for automating payroll calculations in accordance with Ukrainian legislation, providing time tracking, salary and tax calculations, storage of calculation history, and user management.

Conclusions. Software has been developed to automate payroll calculations, which increases the accuracy, reliability, and transparency of payroll processes in organizations, simplifies time tracking, and ensures compliance of calculations with legislative standards.

The field of use is the automation of accounting and payroll calculation processes at enterprises of various sizes and forms of ownership.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області розрахунку заробітної плати.....	10
1.1 Актуальність розробки	10
1.2 Визначення заробітної плати	11
1.3 Форми розрахунку заробітної плати	13
1.4 Аналіз існуючих систем розрахунку заробітної плати	14
1.5 Висновки за розділом	20
2 Проектування програмного забезпечення розрахунку заробітної плати	22
2.1 Вимоги до розробки	22
2.3 Архітектура системи	27
2.4 Вибір технологій розробки	29
2.5 Висновки за розділом	37
3 Опис програмного забезпечення розрахунку заробітної плати	39
3.1 СУБД для розрахунку заробітної плати	39
3.2 Програмна частина.....	46
3.3 Висновки за розділом	58
4 Рекомендації із впровадження програмного забезпечення розрахунку заробітної плати.....	60
4.1 Інструкції користування системою	60
4.2 Тестування програмного забезпечення.....	65
4.3 Підвищення ефективності	68
4.4 Висновки за розділом	69
Висновки	71
Перелік джерел посилання	73
Додаток А Технічне завдання	75
Додаток Б Текст програми	79
Додаток В Слайди презентації.....	96

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	- Artificial Intelligence;
API	- Application Programming Interface;
CTE	- Common Table Expression;
GUID	- Globally Unique Identifier;
MVC	- Model-View-Controller;
БД	- база даних;
ЗП	- заробітна плата;
ІТ	- інформаційні технології;
ПЗ	- програмне забезпечення;
СУБД	- система управління базами даних.

ВСТУП

У сучасних умовах розвитку інформаційних технологій автоматизація бізнес-процесів стає ключовим чинником підвищення продуктивності підприємств і оптимізації їх витрат. Одним із важливих напрямів автоматизації є процес обліку й розрахунку заробітної плати, який потребує високої точності, оперативності й відповідності до чинних норм законодавства [1].

Розвиток ІТ-сфери відкриває широкі можливості для створення автоматизованих інформаційних систем, що забезпечують ефективне управління процесами розрахунку заробітної плати. Особливої актуальності набувають вебтехнології, які дозволяють створювати доступні, масштабовані рішення, інтегровані у корпоративні середовища та доступні через інтернет із будь-якого пристрою. Використання хмарних сервісів, сучасних баз даних та гнучких методів розробки програмного забезпечення сприяє побудові адаптивних систем, що враховують особливості різних організацій [1].

Метою даної дипломної роботи є підвищення ефективності процесу розрахунку заробітної плати працівників шляхом розроблення та впровадження веборієнтованої автоматизованої системи. Система має забезпечувати облік відпрацьованого часу та розрахунок заробітної плати з урахуванням заданих параметрів.

Об'єкт дослідження — процес розроблення програмного забезпечення для автоматизації розрахунку заробітної плати.

Предмет дослідження — методи та інструменти розробки вебзастосунку для обліку робочого часу й нарахування заробітної плати.

Завдання дослідження:

- провести аналіз існуючих програмних рішень для автоматизації обліку заробітної плати;
- визначити функціональні та нефункціональні вимоги до розроблюваного вебзастосунку;

- спроектувати архітектуру системи та структуру бази даних;
- реалізувати програмне забезпечення на основі сучасних технологій веброзробки;
- здійснити тестування і перевірку коректності функціонування системи.

Таким чином, дана робота спрямована на створення сучасного інструменту, що забезпечить підвищення точності, надійності та прозорості процесів розрахунку заробітної плати в організаціях.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ

1.1 Актуальність розробки

Розроблення програмного забезпечення для автоматизації розрахунку заробітної плати має важливе значення для сучасних підприємств, які прагнуть підвищити ефективність управління своїми трудовими ресурсами. Впровадження таких рішень дає змогу суттєво спростити процес нарахування заробітної плати на основі обліку відпрацьованого часу працівників і позитивно вплинути на загальну операційну діяльність організації [1].

Існуючі програмні продукти для розрахунку заробітної плати часто не враховують специфіку бізнес-процесів окремих компаній, що може призводити до неточностей у виплатах. Багато з них недостатньо гнучкі для своєчасного внесення змін відповідно до оновлень трудового законодавства або внутрішніх змін у політиці оплати праці. Окрім того, деякі системи є складними для користувачів і вимагають значних витрат часу на їх вивчення та впровадження [2].

Використання автоматизованих програмних рішень для розрахунку заробітної плати надає низку переваг, серед яких зменшення кількості помилок при нарахуванні виплат, підвищення точності розрахунків та покращення загальної ефективності роботи компанії. Такі системи також забезпечують якісніше ведення обліку робочого часу, що спрощує проходження аудиторських перевірок та дотримання нормативних вимог. Це, у свою чергу, підвищує рівень довіри та задоволеності працівників завдяки своєчасним і правильним виплатам [3].

Необхідність розробки сучасного програмного забезпечення для автоматизації процесу розрахунку заробітної плати обумовлена низкою факторів. Серед них — постійні зміни в трудовому законодавстві, що вимагають від систем гнучкості та можливості швидкого оновлення, а також

потреба у мобільності працівників і зростання кількості віддалених співробітників, що висуває нові вимоги до доступності та масштабованості програмних рішень.

Розроблення нового програмного забезпечення є виправданим не лише через потребу адаптації до змін у законодавчій базі, але й через необхідність інтеграції із суміжними корпоративними системами, зокрема системами управління персоналом і бухгалтерськими обліковими модулями. Сучасні ІТ-технології, такі як хмарні обчислення, машинне навчання та аналітика даних, відкривають нові можливості для створення більш потужних, надійних і ефективних рішень [4].

Автоматизація розрахунку заробітної плати за допомогою спеціалізованого програмного забезпечення є одним із ключових чинників підвищення ефективності управління підприємством. Впровадження інноваційних систем дозволяє істотно вплинути на рівень продуктивності, поліпшити якість внутрішніх процесів та забезпечити стабільність і точність фінансових розрахунків. Тому інвестування у створення та впровадження сучасних програмних рішень у цій сфері є актуальним і стратегічно важливим завданням для бізнесу.

1.2 Визначення заробітної плати

Розрахунок винагороди за працю є невід'ємною частиною діяльності всіх економічних суб'єктів, незалежно від їхнього масштабу, форми власності, галузевої належності чи типу господарської діяльності. Лише фізичні особи-підприємці, які не мають найманих працівників, можуть не здійснювати такі розрахунки. Заробітна плата має ключове значення в управлінні персоналом, фінансовому плануванні та податковому обліку, оскільки відображає економічні відносини між працівниками, роботодавцями, державними структурами та суспільством загалом. Саме

тому питання автоматизації процесів нарахування заробітної плати є актуальним як для бізнесу, так і для державних установ.

У науковій літературі терміни "винагорода за працю" та "заробітна плата" часто вживаються як синоніми. Проте більшість дослідників схиляються до використання терміна "заробітна плата", який закріплений у міжнародних правових актах [1]. Саме поняття заробітної плати є багатограним і вимагає розгляду з різних точок зору - економічної теорії, ринкової економіки, економіки праці, а також через призму інтересів працівника та роботодавця. Відповідно до Закону України "Про оплату праці", заробітна плата визначається як грошова винагорода, що виплачується роботодавцем працівникові за виконану роботу згідно з укладеним трудовим договором. Це визначення є базовим і найбільш визнаним серед фахівців, які наголошують на грошовій природі розрахунків [2].

Розмір заробітної плати кожного працівника залежить від його особистого внеску у загальні результати роботи підприємства та підлягає регулюванню чинним податковим законодавством. При цьому максимальні розміри виплат не обмежуються, тоді як мінімальний рівень оплати праці гарантується державою [3].

Оплата праці виконує низку важливих функцій: забезпечує відтворення робочої сили, стимулює працівників до підвищення продуктивності, регулює розподіл праці залежно від її складності та важливості, сприяє соціальній рівності та підтримує купівельний попит населення. Відтворювальна функція гарантує можливість працівника підтримувати свою працездатність, стимулююча - мотивує до якіснішої роботи, регулююча - забезпечує диференціацію оплати залежно від кваліфікації, соціальна - сприяє недопущенню дискримінації у розрахунках, а функція формування попиту підтримує економічну рівновагу [1].

Заробітна плата поділяється на номінальну та реальну: номінальна виражається у грошовій формі, тоді як реальна визначається кількістю

товарів і послуг, які можна придбати за ці кошти. Реальна заробітна плата є ключовим показником купівельної спроможності населення [1].

Розуміння особливостей функціонування системи оплати праці має принципове значення для розроблення ефективного програмного забезпечення. Автоматизація процесів розрахунку заробітної плати дозволяє врахувати всі зазначені аспекти, забезпечуючи точність, прозорість і відповідність чинним законодавчим вимогам, що у свою чергу сприяє підвищенню ефективності роботи підприємств та поліпшенню добробуту працівників.

1.3 Форми розрахунку заробітної плати

Класифікація, методи та підходи до формування винагороди за працю є основою для побудови процесів нарахування заробітної плати кожному працівнику в організації. Типи виплат, що включаються до складу заробітної плати, визначені статтею 2 Закону України «Про оплату праці» [1].

Відповідно до чинного законодавства, структура заробітної плати охоплює такі основні елементи:

- основна заробітна плата - сума виплат за виконання встановлених норм праці (норм часу, виробітку, службових обов'язків тощо);
- додаткова заробітна плата - винагорода за результати, що перевищують встановлені норми, а також за особливі досягнення, інноваційні пропозиції чи роботу в особливих умовах; включає доплати, надбавки, премії та інші обов'язкові компенсаційні виплати;
- інші заохочувальні та компенсаційні виплати - матеріальне заохочення за підсумками року, спеціальні премії, грошові гранти та інші виплати, що виходять за межі базових нормативів.

Системи оплати праці визначають підходи до обліку кількості та якості виконаної роботи, тоді як форми оплати вказують на способи

нарахування заробітку з урахуванням особливостей виробничого процесу та фінансової стратегії підприємства [2].

Вибір відповідної системи оплати праці залежить від розміру підприємства, його організаційної структури, форми власності, специфіки виготовлюваної продукції та внутрішніх цінностей і стратегічних цілей компанії.

Розрізняють два основних типи систем оплати праці [3]:

- тарифна система - базується на класифікації робіт за складністю і встановленні відповідних тарифних ставок, посадових окладів та кваліфікаційних вимог;
- контрактна система - ґрунтується на індивідуальних договорах між роботодавцем і працівником, де визначаються умови праці, порядок нарахування винагороди та інші договірні положення.

Форми оплати праці можуть бути відрядними або погодинними, кожна з яких має кілька варіантів залежно від умов виробництва і внутрішніх стандартів організації.

Правильний вибір системи й форми оплати праці, що відповідає особливостям робочого процесу, кваліфікаційним вимогам та індивідуальним очікуванням працівників, є важливим чинником підвищення продуктивності праці й загальної ефективності роботи підприємства [4]-[6].

Розроблення сучасного програмного забезпечення для розрахунку заробітної плати має враховувати всю цю складну структуру, забезпечуючи точний облік усіх видів виплат та адаптивність до змін у законодавстві й внутрішній політиці підприємства.

1.4 Аналіз існуючих систем розрахунку заробітної плати

Зростаюча потреба у використанні комп'ютерної техніки й програмного забезпечення стала актуальною реальністю для малих і середніх підприємств. Це обумовлено розвитком різноманітних ІТ-рішень, які

забезпечують обробку внутрішньої бухгалтерської інформації, ведення обліку по окремих об'єктах та автоматизацію формування податкової і статистичної звітності для передання до відповідних державних органів. Суттєвою перевагою є також поширення електронного документообігу, що підвищує ефективність звітних процесів і оптимізує робочий час бухгалтерських служб [7].

Попри значні переваги, впровадження комп'ютерних технологій супроводжується певними труднощами. Серед основних переваг відзначаються спрощення бухгалтерських процедур, зменшення кількості помилок при складанні первинних документів, скорочення часових витрат на обробку інформації та оперативний доступ до великих обсягів даних. До недоліків належить висока вартість комп'ютерного обладнання та ліцензійного програмного забезпечення, що є особливо відчутним для малих підприємств [8].

Інтеграція комп'ютерних технологій та спеціалізованого програмного забезпечення радикально змінює підхід до організації бухгалтерського обліку, перетворюючи його з допоміжного інструменту на центральний елемент корпоративної інформаційної системи. В Україні представлений широкий вибір програмних рішень для обліку оплати праці, однак автоматизація цього напрямку залишається одним із найскладніших завдань через такі особливості:

- складність алгоритмів розрахунку та різноманіття структур заробітної плати;
- регулярні зміни у нормативно-правовій базі;
- неоднозначність трактування окремих норм законодавства;
- необхідність застосування різних допоміжних алгоритмів обрахунку;
- відмінності у порядку нарахування, утримання і виплати заробітної плати;
- специфіку розрахунку податків та соціальних внесків (табл. 1.1).

Таблиця 1.1 – Основні аспекти використання ІТ у розрахунку заробітної плати

Категорія	Деталі
Потреба в автоматизації	Необхідність для малих і середніх підприємств для підвищення ефективності обліку та звітності.
Переваги	- спрощення роботи бухгалтерів; - зменшення кількості помилок; - оптимізація витрат часу; - швидкий доступ до даних.
Недоліки	Висока вартість устаткування та програмних продуктів.
Зміни в організації обліку	Перехід від допоміжної функції до ключового елементу корпоративної інформаційної системи.
Проблеми автоматизації	- складні алгоритми розрахунків; - часті зміни законодавства; - неоднозначне трактування норм; - різні алгоритми допоміжних обрахунків.
Підходи до розробки ПЗ	- відкрита архітектура: модульні системи з адаптивним функціоналом; - закриті системи: спеціалізовані пакети для окремих облікових завдань.

Різноманіття підходів до розробки програмного забезпечення для автоматизації обліку дозволяє підприємствам обирати рішення, що найкраще відповідає їхнім потребам. Один із підходів полягає у створенні спеціалізованих мов програмування високого рівня, що полегшують розробку окремих функціональних модулів. Це дає змогу користувачам придбати базову програму-платформу і самостійно доповнювати її модулями відповідно до своїх вимог [9].

Концепція відкритої архітектури дозволяє гнучко налаштовувати і розширювати функціонал систем. Прикладом є система «X-DOOR» компанії SoftTAXI, що складається з окремих модулів для різних бухгалтерських напрямків. Альтернативний варіант — розробка закритих комплексних пакетів, спеціалізованих на певних облікових процесах.

Програмні продукти умовно поділяються на три категорії:

- універсальні бухгалтерські комплекси для автоматизації обліку на підприємствах будь-якого профілю. Вони складаються з модулів для обліку основних засобів, виробничих запасів, праці та заробітної плати, інтегрованих через головний модуль, наприклад, «Головну книгу» (серед прикладів — системи «R/3», «Platinum», «Scaly», «Sun», «Моноліт», «Галактика») [10];

- програмні пакети для малого та середнього бізнесу, такі як «Турбо-бухгалтер», «Колібри», «Фінанси без проблем», «Леді Фін», а також рішення для середніх і великих підприємств («БЕСТ», «Парус-Підприємство», «Офіс 2000 тм», «1С:Підприємство»);

- спеціалізовані галузеві рішення, що автоматизують окремі аспекти обліку, наприклад «Розрахунок заробітної плати», «Облік МШП».

Лідером серед постачальників бухгалтерських систем залишається компанія 1С, яка пропонує інформаційні комплекси, здатні охоплювати повний цикл бухгалтерського і податкового обліку — від обробки первинних документів до формування балансової та аналітичної звітності [11].

Таким чином, автоматизація процесів нарахування заробітної плати потребує використання сучасних програмних рішень, орієнтованих на точність розрахунків, гнучкість адаптації та відповідність змінюваним вимогам законодавства.

У сучасному бізнес-середовищі України якісне бухгалтерське програмне забезпечення є критично важливим інструментом для ефективного ведення фінансового обліку та виконання податкових зобов'язань. На ринку

бухгалтерських програм присутні різноманітні рішення, кожне з яких має свої особливості, переваги та недоліки.

У даному аналізі представлено порівняння чотирьох провідних конкурентів на ринку України: «Турбо-бухгалтер», «Колібри», «Фінанси без проблем» та «Леді Фін». Метою цього порівняння є визначення ключових відмінностей між програмами за найважливішими для користувачів критеріями, що допоможе підприємцям та організаціям зробити обґрунтований вибір, оптимальний для їхніх потреб та бюджету (табл. 1.2).

Таблиця 1.2 – Порівняння існуючих систем

Критерій	Турбо-бухгалтер	Колібри	Фінанси без проблем	Леді Фін
1	2	3	4	5
Вартість впровадження	Висока	Низька	Середня	Середня
Доступ з різних платформ	Windows, вебверсія	Windows, macOS, мобільні додатки, вебверсія	Windows, Android	Windows, macOS, iOS, вебверсія
Резервне копіювання	Автоматичне, хмарне зберігання	Ручне, локальне	Автоматичне, хмарне зберігання	Автоматичне, хмарне та локальне
Адаптивність до законодавчих змін	Висока, оновлення щомісяця	Середня, оновлення щокварталу	Висока, оновлення щотижня	Висока, оновлення щомісяця
Технічна підтримка	24/7, персональний менеджер	Тільки в робочі години	Робочі години + вихідні	24/7, чат-бот + оператор

Продовження таблиці 1.2

1	2	3	4	5
Інтеграція з іншими сервісами	Банки, ДПС, митниця	Лише банки	Банки, ДПС, HR-системи	Банки, ДПС, CRM, e-commerce платформи
Навчальні матеріали	Відеокурси, вебінари, документація	Базова документація	Вебінари, документація	Відеокурси, вебінари, документація, шаблони
Функціональність	Розширена	Базова	Стандартна	Розширена
Масштабованість	Висока (підходить для великих підприємств)	Низька (для ФОП та малого бізнесу)	Середня (для малого та середнього бізнесу)	Висока (гнучка система тарифів)

Аналіз конкурентного середовища на ринку бухгалтерського програмного забезпечення України демонструє різноманітність підходів та рішень, орієнтованих на різні сегменти користувачів.

"Турбо-бухгалтер" виділяється як преміальне рішення з найвищою вартістю впровадження, проте пропонує розширену функціональність і цілодобову технічну підтримку, що робить його оптимальним вибором для великих підприємств зі складною структурою обліку.

"Колібрі" позиціонується як економічне рішення з низькою вартістю впровадження та базовою функціональністю, що найкраще підходить для ФОП та малого бізнесу з простими обліковими операціями.

"Фінанси без проблем" пропонує збалансоване співвідношення ціни та якості з найчастішими оновленнями в контексті законодавчих змін, що важливо для бізнесу, який працює в галузях з динамічним регулюванням.

"Леді Фін" вирізняється найширшими можливостями інтеграції з різними платформами та сервісами, а також гнучкою системою масштабування, що дозволяє адаптувати рішення під зростаючі потреби бізнесу.

Порівняльний аналіз існуючих рішень демонструє, що незважаючи на різноманітність програмних продуктів на ринку, існує потреба в розробці нового, більш гнучкого та інноваційного рішення. Тому виникає необхідність розроблення системи розрахунку заробітної плати в рамках дипломної роботи, яка поєднуватиме в собі кращі практики існуючих рішень, враховуватиме сучасні тенденції розвитку інформаційних технологій та відповідатиме специфічним потребам українських підприємств. Створення такої системи дозволить запропонувати конкурентоспроможне програмне забезпечення, що матиме перевагами оптимальне співвідношення ціни та функціональності, високу адаптивність до змін законодавства та зручність використання.

1.5 Висновки за розділом

Проведений аналіз предметної області розрахунку заробітної плати виявив стратегічну важливість автоматизації цього процесу для сучасних підприємств. Існуючі програмні рішення часто не враховують специфіку окремих компаній та недостатньо гнучкі щодо адаптації до законодавчих змін.

Заробітна плата як економічна категорія виконує відтворювальну, стимулюючу, регулюючу, соціальну функції та функцію формування платоспроможного попиту. Згідно з українським законодавством, структура заробітної плати включає основну, додаткову та заохочувальні виплати. Різноманіття систем оплати (тарифна і контрактна) та форм нарахування (відрядна і погодинна) потребує універсального та адаптивного програмного рішення.

Аналіз існуючих автоматизованих систем на українському ринку ("Турбо-бухгалтер", "Колібри", "Фінанси без проблем", "Леді Фін") показав їх орієнтованість на різні сегменти ринку з відповідними відмінностями у функціоналі та вартості. Складність автоматизації в цій сфері зумовлена різноманіттям алгоритмів розрахунку, регулярними змінами законодавства та особливостями розрахунку податків і соціальних внесків.

Сучасні вебтехнології, хмарні сервіси та бази даних відкривають широкі можливості для створення гнучких, доступних і масштабованих рішень. Порівняльний аналіз існуючих продуктів свідчить про потребу в розробці нового, більш інноваційного підходу до автоматизації розрахунку заробітної плати.

Таким чином, незважаючи на різноманіття програмних продуктів на ринку, виникає необхідність розроблення системи в рамках дипломної роботи, яка поєднає кращі практики існуючих рішень, врахує сучасні технологічні тенденції та задовольнить специфічні потреби українських підприємств.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ

2.1 Вимоги до розробки

На початковому етапі розроблення програмного забезпечення для автоматизації розрахунку заробітної плати необхідно провести ретельний аналіз потреб цільової аудиторії та поточних вимог ринку. Дослідження існуючих процедур нарахування заробітної плати в компаніях дозволяє виявити слабкі місця, які потребують оптимізації або повної автоматизації (табл. 2.1).

Другим важливим кроком є формування чітких функціональних вимог до системи. Це включає деталізацію необхідних можливостей, таких як облік відпрацьованого часу, автоматичний розрахунок податків, оновлення ставок оплати праці, а також інтеграцію з іншими корпоративними системами обліку та звітності [12].

На третьому етапі здійснюється проєктування архітектури майбутньої системи. Визначаються взаємозв'язки між основними модулями, схеми обміну даними між ними та способи інтеграції з зовнішніми платформами.

Четвертий крок передбачає вибір технологічних рішень для реалізації проєкту. Вибір відповідного технологічного стеку критично важливий для забезпечення масштабованості, продуктивності, безпеки та майбутньої гнучкості системи [13].

П'ятим кроком є розроблення структури бази даних, яка повинна забезпечувати надійне зберігання інформації про працівників, їхню зайнятість, розрахунки заробітної плати та податкові відрахування, а також забезпечувати швидкий доступ до даних для звітності та аналізу.

Шостий крок — створення інтуїтивно зрозумілого інтерфейсу користувача. Інтерфейс повинен бути простим для сприйняття, доступним для користувачів з різним рівнем технічної підготовки та оптимізованим для швидкої навігації.

Сьомий етап — комплексне тестування системи для перевірки її стабільності, надійності, захисту даних і відповідності заданим функціональним вимогам. Тестування повинно включати як модульні, так і інтеграційні випробування, а також навантажувальні тести.

Таблиця 2.1 – Основні завдання розроблення програмного забезпечення розрахунку заробітної плати

Етап розробки	Основні завдання
Аналіз потреб	Дослідження поточних процедур розрахунку заробітної плати та виявлення потреб користувачів.
Визначення функціональних вимог	Специфікація функцій: розрахунок годин, податків, змін ставок, інтеграційні можливості.
Проектування системи	Створення архітектурної моделі взаємодії модулів і механізмів інтеграції з іншими системами.
Вибір технологій	Підбір технологічного стеку з урахуванням вимог до безпеки, масштабованості та продуктивності.
Розроблення бази даних	Створення бази даних для зберігання персональних даних, історії розрахунків і параметрів обліку.
Розробка інтерфейсу	Проектування простого й доступного користувацького інтерфейсу.
Тестування	Перевірка функціональної повноти, безпеки, витривалості та надійності програмного забезпечення.

Наступним критичним етапом є деталізація функціональних можливостей майбутньої системи. Нижче наведено ключові компоненти та процеси, які повинні бути реалізовані в межах проєкту.

Реєстрація та управління працівниками:

- зберігання персональної інформації (ПІБ, контактні дані, ПІН, дата народження, посада, відділ);
- облік історії зайнятості (дати прийому, переведення, історія змін заробітної плати, підвищення кваліфікації).

Облік робочого часу:

- класифікація типів годин (звичайні, лікарняні, відпусткові дні);
- інтеграція з системами обліку робочого часу або підтримка ручного введення;
- автоматичне коригування робочого часу у випадку святкових або вихідних днів.

Розрахунок податків:

- налаштування системи для обліку актуальних ставок ПДФО, ЄСВ, військового збору;
- можливість коригування розрахунків залежно від особливостей працівника (наприклад, пільги на дітей).

Внесення змін до ставок:

- забезпечення швидкої зміни ставок оплати праці та податкових ставок без збоїв у роботі системи;
- підтримка адаптації системи до нових нормативних вимог і змін внутрішніх політик компанії (рис. 2.1).

Окрім функціональних характеристик, розроблення автоматизованої системи розрахунку заробітної плати вимагає врахування низки нефункціональних вимог, які забезпечують загальну якість, безпеку та надійність рішення.

Безпека:

- захист персональних даних працівників відповідно до вимог законодавства про захист персональних даних;
- використання протоколів шифрування для передавання конфіденційної інформації;

- наявність системи автентифікації користувачів та управління правами доступу;
- реалізація механізмів резервного копіювання даних та їх відновлення у разі збоїв.



Рисунок 2.1 – Діаграма варіантів використання системи

Продуктивність:

- забезпечення мінімального часу обробки запитів незалежно від кількості одночасних користувачів;
- оптимізація запитів до бази даних для прискорення формування звітів і обробки великої кількості записів.

Масштабованість:

- можливість розширення функціональності системи без потреби в кардинальних змінах архітектури;
- підтримка збільшення кількості користувачів і обсягу оброблюваних даних із мінімальним впливом на продуктивність.

Надійність:

- гарантоване збереження даних у разі несподіваних відмов системи;
- стабільність роботи при високих навантаженнях та під час тривалого використання без необхідності перезавантаження.

Мобільність та доступність:

- забезпечення можливості доступу до системи через вебінтерфейс із різних пристроїв (ПК, ноутбуки, планшети);
- підтримка адаптивного дизайну для комфортної роботи на різних екранах.

Зручність використання:

- інтуїтивна навігація по інтерфейсу, зрозуміла логіка виконання дій;
- наявність підказок, повідомлень про помилки та довідкової інформації в системі.

Відповідність законодавчим нормам:

- оновлення алгоритмів розрахунків відповідно до змін у податковому законодавстві та вимог державних органів.

2.3 Архітектура системи

Архітектура системи відіграє ключову роль у процесі створення будь-якого програмного забезпечення. Вона визначає загальну будову системи, її основні компоненти та спосіб їх взаємодії між собою. Існує декілька основних типів архітектурних підходів, кожен із яких має власні особливості та області застосування:

- монолітна архітектура передбачає об'єднання всіх функціональних елементів - логіки обробки, користувацького інтерфейсу та доступу до даних - у межах одного програмного блоку або застосунку. Основними перевагами такого підходу є простота розгортання та тестування системи. Проте зі збільшенням обсягу функціональності монолітні системи стають дедалі складнішими в підтримці й масштабуванні [14];

- клієнт-серверна архітектура базується на розподілі системи на дві частини: клієнтську та серверну. Клієнт відповідає за взаємодію з користувачем, а сервер - за обробку запитів, управління бізнес-логікою та роботу з базою даних. Така структура забезпечує гнучкість, полегшує підтримку та спрощує масштабування програмного забезпечення [15];

- мікросервісна архітектура передбачає розбиття системи на набір дрібних, незалежних сервісів, кожен із яких виконує окрему функцію. Завдяки такому підходу можливе незалежне розгортання та масштабування окремих компонентів системи. Проте впровадження мікросервісної архітектури вимагає ретельного управління взаємодією між сервісами та складнішої інфраструктури [16];

- сервіс-орієнтована архітектура (SOA) будується на основі взаємодії незалежних сервісів, розташованих на різних серверах. Вона забезпечує гнучкість і можливість багаторазового використання сервісів, однак вимагає складних механізмів для управління обміном даними між компонентами.

Для розроблення програмного забезпечення розрахунку заробітної плати оптимальним вибором є клієнт-серверна архітектура (табл. 2.2).

Таблиця 2.2 – Переваги клієнт-серверної архітектури для системи розрахунку заробітної плати

Перевага	Опис
Розподіл навантаження	Дані обробляються як на клієнтській, так і на серверній стороні, що оптимізує ресурси.
Масштабованість	Система легко адаптується до зростаючих потреб шляхом додавання серверних потужностей.
Безпека	Централізоване управління безпекою даних зменшує ризики несанкціонованого доступу.
Легкість у підтримці	Завдяки розподілу функцій оновлення системи відбуваються без змін на клієнтській стороні.
Гнучкість у виборі технологій	Клієнтська та серверна частини можуть бути реалізовані з використанням різних технологій.
Централізоване управління даними	Єдина база даних забезпечує актуальність та цілісність інформації.

Вибір клієнт-серверної архітектури обґрунтовується наступними чинниками:

- розподіл навантаження. Клієнт-серверний підхід дозволяє розділити обробку даних між клієнтом та сервером, що забезпечує ефективніше використання ресурсів. Сервер бере на себе обробку складних обчислень та управління базами даних, тоді як клієнт відповідає за візуалізацію даних і забезпечення взаємодії з користувачем [17];

- масштабованість. У разі збільшення обсягу оброблюваної інформації система легко масштабується за рахунок додавання нових серверів без необхідності зміни клієнтської частини, що особливо актуально для підприємств, які розширюються [18];

- безпека. Централізоване зберігання важливої інформації на сервері дозволяє ефективно контролювати доступ до даних і захищати їх від зовнішніх загроз;

- легкість у підтримці. Оскільки логіка обробки даних і управління зосереджені на сервері, внесення змін або оновлень можливе без потреби оновлювати клієнтське програмне забезпечення [19];

- гнучкість у виборі технологій. Для розробки клієнтської та серверної частин можуть бути обрані найоптимальніші технології відповідно до потреб проєкту, наприклад: вебдодаток для клієнта та потужна серверна платформа із сучасною СУБД;

- централізоване керування даними. Вся інформація зберігається в єдиній базі даних, що дозволяє забезпечити її узгодженість, цілісність та актуальність, що є особливо критичним у процесах розрахунку заробітної плати [20].

Отже, клієнт-серверна архітектура є найдоцільнішим варіантом для системи автоматизації розрахунку заробітної плати, оскільки поєднує у собі високу продуктивність, надійність, безпеку, зручність у підтримці та гнучкість у розвитку.

2.4 Вибір технологій розробки

Розробка ефективної системи розрахунку заробітної плати вимагає ретельного підбору технологій для кожного рівня архітектури. У даному підрозділі проаналізовано найбільш оптимальні технології для frontend і backend розробки, а також систем керування базами даних. Важливо обрати такі інструменти, які забезпечать продуктивність, масштабованість та надійність майбутньої системи.

2.4.1 Технології для розробки серверної частини

Python - багатоцільова мова програмування високого рівня з широкими можливостями застосування у веброзробці. Ця технологія має низку ключових переваг для серверної розробки. Насамперед, Python

відзначається простотою та зрозумілістю синтаксису, що значно прискорює процес розробки та полегшує подальшу підтримку коду [7]. Наявність великої кількості бібліотек та фреймворків для веброзробки, таких як Django або Flask, суттєво спрощує створення складних вебсистем. Важливим аспектом є активна спільнота розробників, яка постійно вдосконалює мову та створює нові бібліотеки для різноманітних задач.

Серед недоліків Python можна відзначити відносно нижчу продуктивність у порівнянні з компільованими мовами при обробці дуже великих обсягів даних. Також, динамічна типізація мови може призводити до помилок, які важче виявити на етапі компіляції, ніж у статично типізованих мовах [21].

Node.js - середовище виконання JavaScript на серверній стороні, яке дозволяє розробникам використовувати одну мову як для клієнтської, так і для серверної розробки. Ця уніфікація стеку технологій є значною перевагою, оскільки спрощує комунікацію між різними командами розробників та полегшує підтримку проєкту. Node.js базується на високопродуктивному рушії V8 від Google, що забезпечує високу швидкість обробки запитів [8].

Асинхронна модель обробки введення-виведення в Node.js дозволяє ефективно обробляти велику кількість одночасних з'єднань, що робить її особливо корисною для розробки масштабованих вебзастосунків. Проте, при виконанні складних обчислень Node.js може демонструвати нижчу продуктивність через свою однопоточну природу. Також, складність асинхронного програмування може ускладнювати розробку та налагодження, особливо для початківців [22].

Java Spring Boot - потужний фреймворк для розробки вебзастосунків на мові Java, який відрізняється високою стабільністю та продуктивністю, що робить його популярним вибором для корпоративних рішень. Статична типізація Java допомагає запобігти багатьом помилкам на етапі компіляції, що підвищує надійність коду [9].

Spring Boot надає широкі можливості для інтеграції з різними корпоративними системами та базами даних, а також має потужні засоби для забезпечення безпеки, що критично важливо для фінансових систем. До недоліків можна віднести складність налаштування та початкового вивчення платформи, а також більш тривалий цикл розробки порівняно з Python чи Node.js [23]. Вищі вимоги до обчислювальних ресурсів також можуть обмежувати використання Java в певних середовищах (табл. 2.3).

Таблиця 2.3 - Порівняння технологій серверної розробки

Критерій	Python	Node.js	Java Spring Boot
Продуктивність розробки	Висока	Середня	Низька
Продуктивність виконання	Середня	Висока для I/O операцій	Висока
Безпека	Хороша	Середня	Відмінна
Масштабованість	Хороша	Відмінна	Відмінна
Підтримка інтеграцій	Широка	Широка	Надзвичайно широка
Складність освоєння	Низька	Середня	Висока
Вимоги до ресурсів	Низькі	Низькі	Високі

На основі аналізу представлених технологій для реалізації серверної частини системи розрахунку заробітної плати було обрано Python з фреймворком Django. Ця комбінація забезпечує оптимальний баланс між швидкістю розробки, надійністю та легкістю підтримки. Наявність ORM Django спрощує роботу з базами даних, а система аутентифікації забезпечує базовий рівень захисту.

2.4.2 Технології для розробки клієнтської частини

React.js являє собою бібліотеку JavaScript для побудови інтерактивних користувацьких інтерфейсів, розроблену компанією Facebook. Основною перевагою React є компонентний підхід до розробки, який дозволяє створювати інтерфейси з багаторазово використовуваних блоків [10]. Віртуальний DOM забезпечує високу продуктивність навіть при складних оновленнях інтерфейсу, що важливо для систем з динамічними даними.

Реакт має велику екосистему додаткових бібліотек, які розширюють його функціональність, включаючи рішення для управління станом, маршрутизації та інтеграції з API. Значною перевагою також є велика спільнота розробників та ґрунтовна документація. До недоліків слід віднести відносно високу складність для новачків і необхідність використання додаткових бібліотек для повноцінного функціонування застосунку.

Vue.js - прогресивний фреймворк для побудови користувацьких інтерфейсів, який поєднує в собі найкращі ідеї React та Angular. Ключовою перевагою Vue є його інкрементальна природа, що дозволяє поступово впроваджувати фреймворк у проєкти [11]. Vue має інтуїтивно зрозумілий шаблонний синтаксис, який поєднує HTML з JavaScript-директивами, що робить його більш доступним для розробників з досвідом традиційного вебпрограмування.

Цей фреймворк забезпечує відмінну продуктивність завдяки оптимізованому рендерингу та реактивній системі даних. Однак, Vue має менш розвинену екосистему порівняно з React, особливо для складних корпоративних рішень. Також, меншою є кількість спеціалістів на ринку праці, що може ускладнити пошук розробників для підтримки проєкту.

Angular - потужний фреймворк, розроблений компанією Google, що забезпечує комплексне рішення для створення складних вебзастосунків. Він відрізняється від React та Vue тим, що пропонує повноцінний набір

інструментів "з коробки", включаючи маршрутизацію, управління формами, HTTP-клієнт та інструменти для тестування [12].

Використання TypeScript у Angular забезпечує сильну типізацію, що допомагає запобігти багатьом помилкам під час розробки. Фреймворк також підтримує двосторонній зв'язок даних, що спрощує розробку форм. Серед недоліків – висока складність освоєння, більша вага кінцевого продукту та загальна громіздкість для малих і середніх проєктів. Крім того, Angular має найбільш крутий поріг входження серед розглянутих фреймворків (табл. 2.4).

Таблиця 2.4 - Порівняння технологій клієнтської розробки

Критерій	React.js	Vue.js	Angular
Розмір бібліотеки	Малий	Дуже малий	Великий
Складність вивчення	Середня	Низька	Висока
Продуктивність	Висока	Висока	Середня
Екосистема	Дуже розвинена	Розвинена	Добре інтегрована
Інструменти розробки	Численні	Достатні	Вбудовані
Підтримка спільноти	Дуже активна	Активна	Активна
Підтримка TypeScript	Хороша	Прийнятна	Нативна

Для розробки клієнтської частини системи розрахунку заробітної плати обрано React.js через його гнучкість, високу продуктивність та великий вибір допоміжних бібліотек. React дозволяє створювати складні інтерфейси користувача з численними інтерактивними елементами, які необхідні для ефективної роботи з фінансовими даними та формами введення. Компонентний підхід також спрощує подальший розвиток і масштабування системи, а зрілість технології забезпечує стабільність і надійність рішення.

2.4.3 Системи керування базами даних

MySQL є однією з найпопулярніших реляційних систем керування базами даних з відкритим кодом. Її основними перевагами є висока продуктивність, надійність та відносна простота у використанні [13]. MySQL добре інтегрується з більшістю сучасних технологій веброзробки, включаючи Python, PHP, Java та інші.

Система забезпечує повну підтримку ACID-транзакцій, що критично важливо для фінансових застосунків, де кожна операція повинна бути або повністю виконана, або повністю скасована. MySQL також має ефективні механізми для забезпечення цілісності даних через використання зовнішніх ключів, обмежень та тригерів.

Серед недоліків MySQL можна виділити менш розвинені можливості для аналітичних запитів порівняно з деякими іншими СУБД та складність налаштування для роботи з дуже великими обсягами даних. Варто зазначити, що деякі передові функції доступні лише в комерційній версії.

PostgreSQL - це об'єктно-реляційна система керування базами даних з відкритим кодом, яка відрізняється від MySQL більш широким спектром функціональних можливостей [14]. PostgreSQL забезпечує повну відповідність стандартам SQL та підтримує складні типи даних, включаючи JSON, XML та геопросторові дані.

Ця СУБД відома своєю розширюваністю - користувачі можуть створювати власні типи даних, агрегатні функції та процедурні мови. PostgreSQL також забезпечує надійні механізми для паралельної обробки даних та управління транзакціями. Водночас, PostgreSQL вимагає більше системних ресурсів для ефективної роботи, має складнішу систему адміністрування та може демонструвати нижчу продуктивність для простих операцій зчитування даних.

MongoDB - це документно-орієнтована NoSQL база даних, яка зберігає дані у форматі, подібному до JSON (BSON) [15]. На відміну від

реляційних СУБД, MongoDB не вимагає попереднього визначення схеми даних, що забезпечує більшу гнучкість при роботі з даними, структура яких часто змінюється.

Горизонтальна масштабованість та висока продуктивність для операцій зчитування є значними перевагами MongoDB. Система також добре працює з великими обсягами даних та забезпечує автоматичне розподілення навантаження між серверами. Однак, MongoDB має обмежені можливості для складних транзакцій та аналітичних запитів, що може бути критичним недоліком для фінансових систем, де потрібна висока точність та цілісність даних.

У таблиці 2.5 наведено порівняння основних характеристик розглянутих СУБД.

Таблиця 2.5 - Порівняння систем керування базами даних

Критерій	MySQL	PostgreSQL	MongoDB
Тип	Реляційна	Об'єктно-реляційна	Документно-орієнтована
Продуктивність для простих запитів	Висока	Середня	Дуже висока
Продуктивність для складних запитів	Середня	Висока	Низька
Підтримка транзакцій	Повна	Повна	Обмежена
Масштабованість	Хороша	Хороша	Відмінна
Гнучкість схеми даних	Низька	Середня	Висока
Вимоги до ресурсів	Низькі	Середні	Низькі

Для системи розрахунку заробітної плати обрано MySQL як оптимальне рішення для зберігання даних. Ця СУБД забезпечує необхідну надійність, продуктивність та відповідність вимогам ACID, які критично важливі для фінансових операцій. MySQL також добре інтегрується з Django

через його ORM-систему, що спрощує роботу з базою даних та зменшує ризик помилок. Додатковою перевагою є широка поширеність MySQL, що полегшує розгортання системи в різних середовищах та забезпечує доступність кваліфікованих спеціалістів для підтримки та розвитку системи.

2.4.4 Загальний стек технологій системи

На основі проведеного аналізу технологій для розробки системи розрахунку заробітної плати було сформовано оптимальний технологічний стек, представлений на рисунку 2.2.

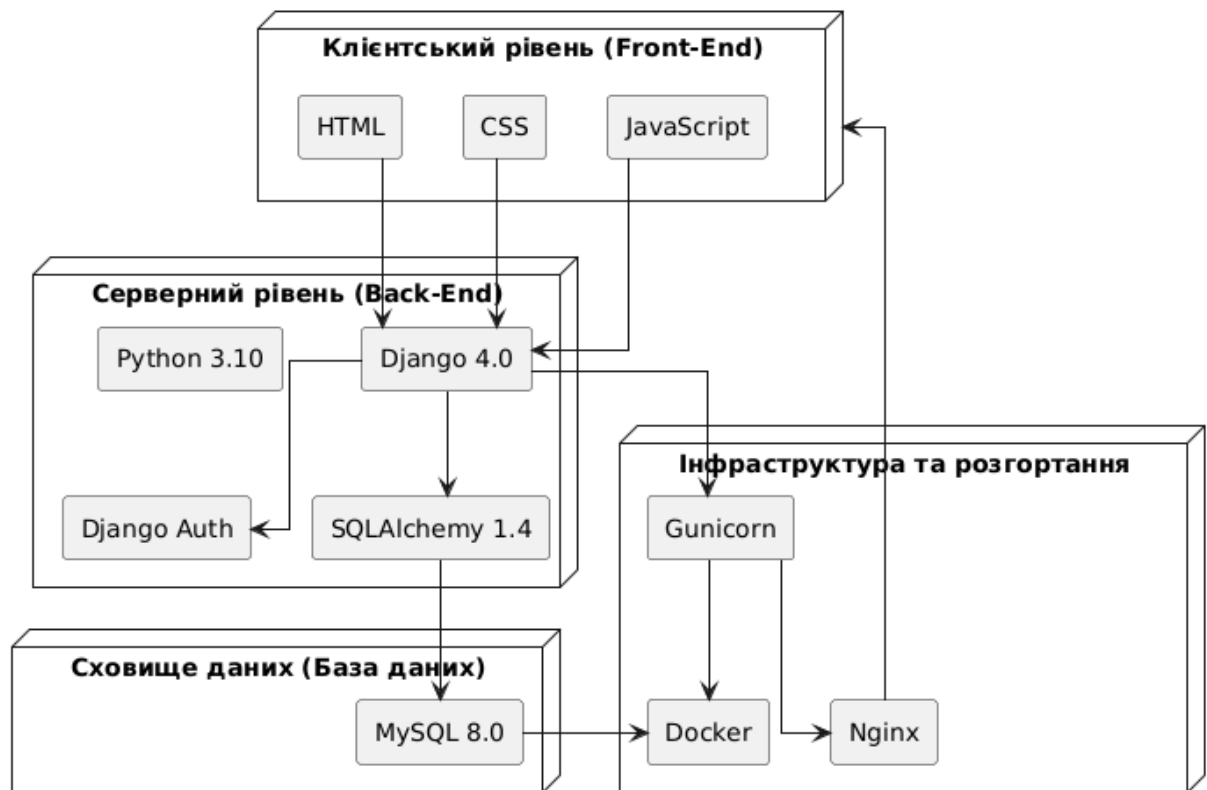


Рисунок 2.2 - Технологічний стек системи розрахунку заробітної плати

Цей стек технологій забезпечує необхідний баланс між продуктивністю розробки, надійністю функціонування та масштабованістю системи. Python з фреймворком Django на серверній стороні забезпечує швидкий розвиток та легкість підтримки, React.js на клієнтській стороні

дозволяє створити гнучкий та інтерактивний інтерфейс користувача, а MySQL на рівні даних гарантує надійне зберігання та обробку фінансової інформації.

В таблиці 2.6 підсумовано основні компоненти технологічного стеку та їх призначення в системі.

Таблиця 2.6 - Компоненти технологічного стеку та їх функції

Компонент	Версія	Призначення
Python	3.10	Мова програмування для серверної розробки
Django	4.0	Вебфреймворк для серверної частини
React.js	18.0	Бібліотека для розробки інтерфейсу користувача
MySQL	8.0	Система керування базами даних
SQLAlchemy	1.4	ORM для роботи з базою даних
Docker	20.10	Платформа для контейнеризації
Nginx	1.20	Вебсервер
Gunicorn	20.1	WSGI-сервер для Python

Обраний технологічний стек відповідає всім вимогам до системи розрахунку заробітної плати, забезпечуючи необхідну функціональність, безпеку, масштабованість та зручність використання. Використання сучасних технологій також гарантує довгострокову підтримку та можливість подальшого розвитку системи.

2.5 Висновки за розділом

У другому розділі було розглянуто проектування автоматизованої системи розрахунку заробітної плати. При розробці нової системи необхідно визначити точні функціональні вимоги, такі як реєстрація працівників, розрахунок годин, обрахунок податків, внесення змін до ставок тощо.

Система повинна бути гнучкою, адаптивною до законодавчих змін та зручною у використанні.

Для архітектури системи обрано клієнт-серверну модель, яка забезпечує розподіл навантаження, масштабованість, безпеку, легкість у підтримці, гнучкість у виборі технологій та централізоване керування даними.

В якості мови програмування для вебзастосунку обрано Python завдяки його простоті, великій кількості бібліотек, активній спільноті розробників та переносимості. Для бази даних обрано MySQL через надійність, масштабованість, розширюваність та підтримку складних запитів. Комбінація Python та MySQL дозволяє швидко розробляти надійну та масштабовану систему.

Таким чином, проведений аналіз дозволив визначити основні вимоги, архітектуру та технології для реалізації автоматизованої системи розрахунку заробітної плати, яка буде відповідати сучасним потребам бізнесу та забезпечить ефективну обробку даних.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ

3.1 СУБД для розрахунку заробітної плати

Для розробки системи розрахунку заробітної плати необхідно створити кілька таблиць у базі даних, які забезпечать зберігання та управління різноманітною інформацією про користувачів, ролі, співробітників, заробітну плату та інші компоненти системи. Нижче наведено описи цих таблиць.

Таблиця користувачів призначена для зберігання інформації про користувачів системи, включаючи їхні облікові дані та дату створення облікового запису (табл. 3.1).

Таблиця 3.1 - Користувачі

Атрибут	Тип даних	Опис
user_id	INT	Ідентифікатор користувача, автоінкремент
username	VARCHAR(50)	Логін користувача, унікальний
password	VARCHAR(255)	Пароль користувача
email	VARCHAR(100)	Електронна адреса користувача, унікальна
created_at	TIMESTAMP	Дата та час створення облікового запису

Таблиця ролей зберігає інформацію про ролі користувачів у системі, включаючи їхні назви та описи (табл. 3.2).

Таблиця 3.2 - Ролі

Атрибут	Тип даних	Опис
role_id	INT	Ідентифікатор ролі, автоінкремент
role_name	VARCHAR(50)	Назва ролі, унікальна
description	VARCHAR(255)	Опис ролі

Таблиця ролей користувачів зберігає зв'язок між користувачами та їхніми ролями у системі (табл. 3.3).

Таблиця 3.3 - Ролі користувачів

Атрибут	Тип даних	Опис
user_id	INT	Ідентифікатор користувача
role_id	INT	Ідентифікатор ролі

Таблиця співробітників містить інформацію про працівників, включаючи їхні особисті дані, контактну інформацію, посаду та заробітну плату (табл. 3.4).

Таблиця 3.4 - Співробітники

Атрибут	Тип даних	Опис
employee_id	INT	Ідентифікатор співробітника, автоінкремент
first_name	VARCHAR(50)	Ім'я співробітника
last_name	VARCHAR(50)	Прізвище співробітника
email	VARCHAR(100)	Електронна адреса, унікальна
phone	VARCHAR(20)	Телефонний номер
address	VARCHAR(255)	Адреса
position	VARCHAR(100)	Посада
hire_date	DATE	Дата прийняття на роботу
basic_salary	DECIMAL(10, 2)	Основна заробітна плата
created_at	TIMESTAMP	Дата та час створення запису

Таблиця заробітної плати зберігає інформацію про заробітну плату співробітників, включаючи кількість відпрацьованих годин, загальну заробітну плату та дату нарахування (табл. 3.5).

Таблиця 3.5 - Заробітна плата

Атрибут	Тип даних	Опис
salary_id	INT	Ідентифікатор запису про заробітну плату, автоінкремент
employee_id	INT	Ідентифікатор співробітника
total_hours_worked	DECIMAL(5, 2)	Загальна кількість відпрацьованих годин
total_salary	DECIMAL(10, 2)	Загальна сума заробітної плати
salary_date	DATE	Дата нарахування заробітної плати
created_at	TIMESTAMP	Дата та час створення запису

Таблиця компонентів заробітної плати зберігає інформацію про різні складові заробітної плати, такі як основна частина, надбавки, податки та відрахування (табл. 3.6).

Таблиця 3.6 - Компоненти заробітної плати

Атрибут	Тип даних	Опис
component_id	INT	Ідентифікатор компонента, автоінкремент
salary_id	INT	Ідентифікатор запису про заробітну плату
component_type	ENUM	Тип компонента (basic, allowance, tax, deduction)
amount	DECIMAL(10, 2)	Сума компонента
description	VARCHAR(255)	Опис компонента
created_at	TIMESTAMP	Дата та час створення запису

Таблиця відпрацьованих годин зберігає інформацію про кількість годин, відпрацьованих співробітниками, у конкретні дні (табл. 3.7).

Таблиця 3.7 - Відпрацьовані години

Атрибут	Тип даних	Опис
work_hours_id	INT	Ідентифікатор запису, автоінкремент
employee_id	INT	Ідентифікатор співробітника
date_worked	DATE	Дата відпрацювання
hours_worked	DECIMAL(5, 2)	Кількість відпрацьованих годин
created_at	TIMESTAMP	Дата та час створення запису

Таблиця податків зберігає інформацію про податки, що застосовуються до заробітної плати співробітників (табл. 3.8).

Таблиця 3.8 - Податки

Атрибут	Тип даних	Опис
tax_id	INT	Ідентифікатор податку, автоінкремент
tax_name	VARCHAR(50)	Назва податку
tax_rate	DECIMAL(5, 2)	Ставка податку
description	VARCHAR(255)	Опис податку
created_at	TIMESTAMP	Дата та час створення запису

Таблиця відрахувань зберігає інформацію про різні відрахування із заробітної плати співробітників (табл. 3.9).

Таблиця 3.9 - Відрахування

Атрибут	Тип даних	Опис
deduction_id	INT	Ідентифікатор відрахування, автоінкремент
deduction_name	VARCHAR(50)	Назва відрахування
amount	DECIMAL(10, 2)	Сума відрахування
description	VARCHAR(255)	Опис відрахування
created_at	TIMESTAMP	Дата та час створення запису

Таблиця аудиту зберігає інформацію про дії користувачів у системі для забезпечення безпеки та відстеження змін (табл. 3.10).

Таблиця 3.10 - Аудит

Атрибут	Тип даних	Опис
audit_id	INT	Ідентифікатор запису аудиту, автоінкремент
user_id	INT	Ідентифікатор користувача
action	VARCHAR(100)	Опис дії
table_name	VARCHAR(50)	Назва таблиці, до якої було внесено зміни
record_id	INT	Ідентифікатор запису, до якого було внесено зміни

Таблиця 3.11 з первинними (PK) та зовнішніми (FK) ключами всіх сутностей бази даних для вебсистеми розрахунку заробітної плати, а також поясненням кожного зв'язку:

Таблиця 3.11 – Зв'язки та ключі сутностей бази даних

№	Таблиця	Ключ	Тип ключа	Зв'язок (посилання на таблицю)	Опис зв'язку
1	2	3	4	5	6
1	users	user_id	PK	–	Унікальний ідентифікатор користувача
2	roles	role_id	PK	–	Унікальний ідентифікатор ролі
3	user_roles	user_id	FK	→ user_id	Зв'язок користувача з роллю
4	user_roles	role_id	FK	→ role_id	Зв'язок ролі з користувачем
5	employees	employee_id	PK	–	Унікальний ідентифікатор співробітника
6	salaries	salary_id	PK	–	Ідентифікатор запису про зарплату
7	salaries	employee_id	FK	→ employee_id	Зарплата належить конкретному співробітнику
8	salary_components	component_id	PK	–	Унікальний ідентифікатор компонента зарплати
9	salary_components	salary_id	FK	→ salary_id	Компонент належить конкретному запису зарплати

Продовження таблиці 3.11

1	2	3	4	5	6
10	salary_components	tax_id	FK (опц.)	→ tax_id	Якщо компонент — це податок, то посиляється на податкову ставку
11	salary_components	deduction_id	FK (опц.)	→ deduction_id	Якщо компонент — відрахування, то пов'язано з відрахуванням
12	work_hours	work_hours_id	PK	—	Унікальний запис про відпрацьовані години
13	work_hours	employee_id	FK	→ employee_id	Запис годин належить певному працівнику
14	taxes	tax_id	PK	—	Унікальний податок, який можна прив'язати до компонента зарплати
15	deductions	deduction_id	PK	—	Унікальне відрахування, яке можна використати в компонентах зарплати
16	audit	audit_id	PK	—	Унікальний запис аудиту
17	audit	user_id	FK	→ user_id	Користувач, що здійснив дію
18	audit	salary_id	FK (опц.)	→.salary_id	Додатковий зв'язок із зарплатою (контекст аудиту)
19	audit	employee_id	FK (опц.)	→.employee_id	Додатковий зв'язок із співробітником (контекст аудиту)

На рисунку 3.1 показана схема бази даних.

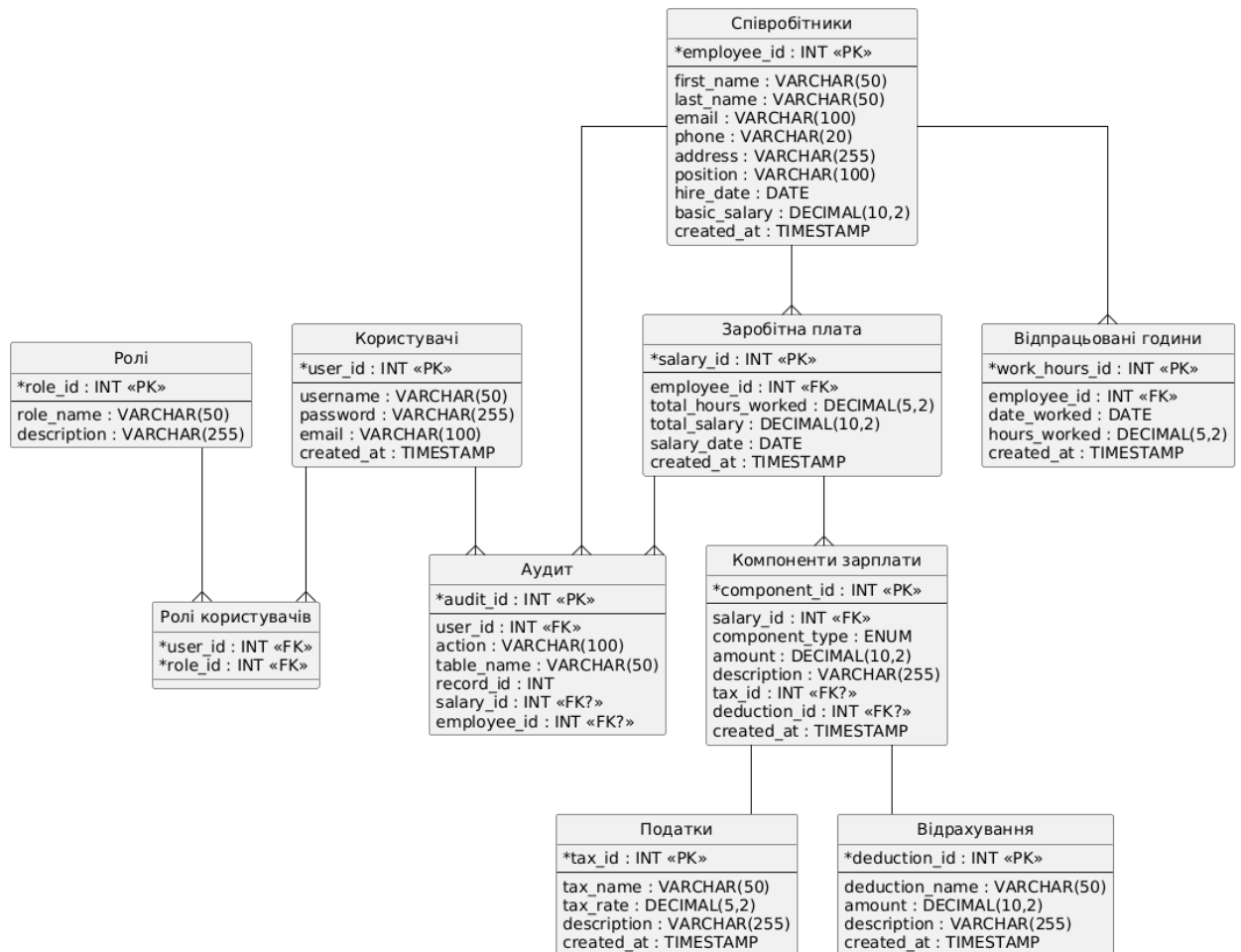


Рисунок 3.1 – Схема бази даних

3.2 Програмна частина

Програмна частина системи розрахунку заробітної плати складається з набору взаємопов'язаних модулів, що забезпечують повний функціонал для автоматизації процесів обчислення заробітної плати. Головна сторінка (index) є вхідною точкою до вебзастосунку та містить загальну інформацію про систему, її можливості та переваги, а також навігацію до інших розділів. Сторінка входу (login) забезпечує користувачам можливість авторизації через введення облікових даних, тоді як сторінка реєстрації (register) дозволяє створювати нові облікові записи.

Вебзастосунок передбачає декілька рівнів доступу з різними панелями: адміністратор отримує доступ до панелі адміністратора

(admin_dashboard) для керування користувачами та налаштуваннями системи; звичайний користувач має доступ до панелі користувача (user_dashboard) для перегляду особистих даних та історії заробітної плати. Сторінка управління співробітниками (manage_employees) дозволяє додавати, редагувати та видаляти інформацію про працівників, а сторінка розрахунку заробітної плати (calculate_salary) забезпечує автоматизацію обчислень на основі введених даних. Також передбачена сторінка налаштувань профілю (profile_settings) для редагування персональних даних та сторінка виходу (logout) для безпечного завершення сесії.

Для реалізації системи розрахунку заробітної плати обрано Python версії 3.10, MySQL версії 8.0 для управління базами даних, та Django версії 4.0 як вебфреймворк. Додатково використовуються SQLAlchemy версії 1.4 для ORM, Django Authentication для управління користувачами. Для забезпечення зручності деплою застосовується Docker, Nginx як вебсервер та Gunicorn як WSGI сервер, а розробка ведеться в середовищі PyCharm.

Ключова особливість реалізованої системи полягає в тому, що всі ставки податків, коефіцієнти та інші параметри розрахунку заробітної плати зберігаються в базі даних, що дозволяє гнучко налаштовувати систему через вебінтерфейс без необхідності зміни програмного коду. Розглянемо основні функції Python для розрахунку заробітної плати з використанням даних, що зберігаються в базі даних (лістинг 3.1).

Лістинг 3.1 - Функція отримання актуальних ставок податків

```
def get_current_tax_rates():
    """
    Отримання актуальних ставок податків з бази даних

    Returns:
        dict: Словник зі ставками податків
    """
    # Отримання актуальних ставок податків з бази даних
```

```

taxes = Taxes.objects.all()

# Формування словника з актуальними ставками
tax_rates = {}
for tax in taxes:
    tax_rates[tax.tax_name] = tax.tax_rate / 100 #
    Конвертація відсотків у десяткове число

return tax_rates

```

Функції розрахунку базової заробітної плати, які використовують дані про відпрацьовані години з бази даних (лістинг 3.2, лістинг 3.3).

Лістинг 3.2 - Функція розрахунку базової заробітної плати

```

def calculate_basic_salary(employee_id, month, year):
    """
    Розрахунок базової заробітної плати працівника з
    використанням даних з БД

    Args:
        employee_id (int): Ідентифікатор працівника
        month (int): Місяць розрахункового періоду
        year (int): Рік розрахункового періоду

    Returns:
        float: Базова заробітна плата
    """
    # Отримання даних про працівника з бази даних
    employee = Employee.objects.get(employee_id=employee_id)

    # Отримання відпрацьованих годин за місяць
    start_date = date(year, month, 1)
    if month == 12:
        end_date = date(year + 1, 1, 1) - timedelta(days=1)
    else:
        end_date = date(year, month + 1, 1) - timedelta(days=1)

```

```

work_hours_records = WorkHours.objects.filter(
    employee_id=employee_id,
    date_worked__gte=start_date,
    date_worked__lte=end_date
)

# Підрахунок загальної кількості відпрацьованих годин
total_hours = 0
for record in work_hours_records:
    total_hours += record.hours_worked

# Розрахунок базової заробітної плати
if hasattr(employee, 'hourly_rate'):
    # Для працівників з погодинною оплатою
    basic_salary = total_hours * employee.hourly_rate
else:
    # Для працівників з фіксованою місячною ставкою
    standard_monthly_hours = 160 # Стандартна кількість
робочих годин на місяць
    basic_salary = employee.basic_salary * (total_hours /
standard_monthly_hours)

return round(basic_salary, 2)

```

Лістинг 3.3 - Функція розрахунку податків

```

def calculate_taxes(gross_salary):
    """
    Розрахунок податків згідно українського законодавства
    з використанням ставок з бази даних

    Args:
        gross_salary (float): Нарахована заробітна плата до
вирахування податків

    Returns:
        dict: Словник з сумами податків та зборів
    """
    # Отримання актуальних ставок податків з бази даних

```

```

tax_rates = get_current_tax_rates()

# Розрахунок податків
taxes = {}
total_tax = 0

for tax_name, tax_rate in tax_rates.items():
    tax_amount = gross_salary * tax_rate
    taxes[tax_name] = round(tax_amount, 2)
    total_tax += tax_amount

taxes['total_tax'] = round(total_tax, 2)

return taxes

```

Для обчислення відрахувань із заробітної плати лістинг 3.4.

Лістинг 3.4 - Функція розрахунку відрахувань

```

def calculate_deductions(employee_id, gross_salary):
    """
    Розрахунок відрахувань із заробітної плати

    Args:
        employee_id (int): Ідентифікатор працівника
        gross_salary (float): Нарахована заробітна плата

    Returns:
        dict: Словник з відрахуваннями
    """
    # Отримання відрахувань з бази даних
    deductions_list = Deductions.objects.all()

    # Розрахунок відрахувань
    deductions = {}
    total_deduction = 0

    for deduction in deductions_list:
        # Визначення суми відрахування

```

```

        if deduction.amount > 0:
            deduction_amount = deduction.amount      # Фіксована
сума
        else:
            # Відрахування у відсотках від зарплати
            deduction_amount = gross_salary *
(round(deduction.percentage / 100))

            deductions[deduction.deduction_name] =
round(deduction_amount, 2)
            total_deduction += deduction_amount

        deductions['total_deduction'] = round(total_deduction, 2)

    return deductions

```

Для розрахунку додаткових компонентів заробітної плати реалізована функція лістинг 3.5.

Лістинг 3.5 - Функція розрахунку компонентів заробітної плати

```

def calculate_salary_components(employee_id, basic_salary,
taxes, deductions, month, year):
    """
    Розрахунок компонентів заробітної плати

    Args:
        employee_id (int): Ідентифікатор працівника
        basic_salary (float): Базова заробітна плата
        taxes (dict): Розраховані податки
        deductions (dict): Розраховані відрахування
        month (int): Місяць розрахунку
        year (int): Рік розрахунку

    Returns:
        list: Список компонентів заробітної плати
    """
    # Створення запису про заробітну плату

```

```

salary = Salaries(
    employee_id=employee_id,
    total_hours_worked=get_total_hours_worked(employee_id,
month, year),
    total_salary=basic_salary - taxes['total_tax'] -
deductions['total_deduction'],
    salary_date=date(year, month, 1),
    created_at=datetime.now()
)
salary.save()

# Створення компонентів заробітної плати
components = []

# Базова складова
basic_component = SalaryComponents(
    salary_id=salary.salary_id,
    component_type='basic',
    amount=basic_salary,
    description='Базова заробітна плата',
    created_at=datetime.now()
)
basic_component.save()
components.append(basic_component)

# Податкові компоненти
for tax_name, tax_amount in taxes.items():
    if tax_name != 'total_tax':
        tax = Taxes.objects.get(tax_name=tax_name)
        tax_component = SalaryComponents(
            salary_id=salary.salary_id,
            component_type='tax',
            amount=tax_amount,
            description=f'Податок: {tax_name}',
            created_at=datetime.now(),
            tax_id=tax.tax_id
        )
        tax_component.save()
        components.append(tax_component)

```

```

# Компоненти відрахувань
for deduction_name, deduction_amount in deductions.items():
    if deduction_name != 'total_deduction':
        deduction = Deductions.objects.get(deduction_name=deduction_name)
        deduction_component = SalaryComponents(
            salary_id=salary.salary_id,
            component_type='deduction',
            amount=deduction_amount,
            description=f'Відрахування: {deduction_name}',
            created_at=datetime.now(),
            deduction_id=deduction.deduction_id
        )
        deduction_component.save()
        components.append(deduction_component)

# Додаткові надбавки
allowances = calculate_allowances(employee_id, basic_salary,
month, year)
for allowance_name, allowance_amount in allowances.items():
    if allowance_amount > 0:
        allowance_component = SalaryComponents(
            salary_id=salary.salary_id,
            component_type='allowance',
            amount=allowance_amount,
            description=f'Надбавка: {allowance_name}',
            created_at=datetime.now()
        )
        allowance_component.save()
        components.append(allowance_component)

return components

```

Функція для розрахунку додаткових надбавок лістинг 3.6.

Лістинг 3.6 - Функція розрахунку надбавок

```

def calculate_allowances(employee_id, basic_salary, month,
year):
    """
    Розрахунок надбавок для працівника

    Args:
        employee_id (int): Ідентифікатор працівника
        basic_salary (float): Базова заробітна плата
        month (int): Місяць розрахунку
        year (int): Рік розрахунку

    Returns:
        dict: Словник з надбавками
    """
    # Отримання даних про працівника
    employee = Employee.objects.get(employee_id=employee_id)

    # Ініціалізація словника надбавок
    allowances = {}

    # Надбавка за стаж роботи
    hire_date = employee.hire_date
    current_date = date(year, month, 1)
    years_of_experience = (current_date.year - hire_date.year) -
(
    (current_date.month, current_date.day) <
(hire_date.month, hire_date.day)
)

    if years_of_experience >= 3:
        # 10% надбавки за кожні 3 роки стажу, максимум 50%
        experience_percent = min(10 * (years_of_experience //
3), 50) / 100
        allowances['experience_bonus'] = round(basic_salary *
experience_percent, 2)

    # Надбавка за понаднормову роботу
    start_date = date(year, month, 1)

```

```

if month == 12:
    end_date = date(year + 1, 1, 1) - timedelta(days=1)
else:
    end_date = date(year, month + 1, 1) - timedelta(days=1)

# Підрахунок понаднормових годин (понад 8 годин на день)
overtime_hours = 0
work_hours_records = WorkHours.objects.filter(
    employee_id=employee_id,
    date_worked__gte=start_date,
    date_worked__lte=end_date
)

daily_hours = {}
for record in work_hours_records:
    day = record.date_worked.day
    if day in daily_hours:
        daily_hours[day] += record.hours_worked
    else:
        daily_hours[day] = record.hours_worked

for day, hours in daily_hours.items():
    if hours > 8:
        overtime_hours += (hours - 8)

if overtime_hours > 0:
    # Надбавка за понаднормову роботу (подвійна ставка)
    hourly_rate = basic_salary /
get_standard_monthly_hours(month, year)
    allowances['overtime_bonus'] = round(overtime_hours *
hourly_rate * 2, 2)

# Надбавка за роботу у вихідні дні
weekend_hours = 0
for record in work_hours_records:
    if record.date_worked.weekday() >= 5: # 5 і 6 - субота
і неділя
        weekend_hours += record.hours_worked

```

```

    if weekend_hours > 0:
        # Надбавка за роботу у вихідні дні (подвійна ставка)
        hourly_rate = basic_salary /
get_standard_monthly_hours(month, year)
        allowances['weekend_bonus'] = round(weekend_hours *
hourly_rate * 2, 2)

    return allowances

```

Об'єднання всіх компонентів та розрахунку кінцевої заробітної плати, реалізована головна функція лістинг 3.7.

Лістинг 3.7 - Головна функція розрахунку заробітної плати

```

def calculate_final_salary(employee_id, month, year):
    """
    Головна функція для розрахунку кінцевої заробітної плати

    Args:
        employee_id (int): Ідентифікатор працівника
        month (int): Місяць розрахунку
        year (int): Рік розрахунку

    Returns:
        dict: Повний розрахунок заробітної плати
    """
    # Розрахунок базової заробітної плати
    basic_salary = calculate_basic_salary(employee_id, month,
year)

    # Розрахунок загальної нарахованої суми з урахуванням
надбавок
    allowances = calculate_allowances(employee_id, basic_salary,
month, year)
    total_allowances = sum(allowances.values())
    gross_salary = basic_salary + total_allowances

    # Розрахунок податків

```

```

taxes = calculate_taxes(gross_salary)

# Розрахунок відрахувань
deductions = calculate_deductions(employee_id, gross_salary)

# Розрахунок чистої заробітної плати
net_salary = gross_salary - taxes['total_tax'] -
deductions['total_deduction']

# Створення компонентів заробітної плати
salary_components = calculate_salary_components(
    employee_id, basic_salary, taxes, deductions, month,
year
)

# Отримання даних про працівника
employee = Employee.objects.get(employee_id=employee_id)

# Формування звіту про заробітну плату
salary_report = {
    'employee_id': employee_id,
    'employee_name': f"{employee.last_name}
{employee.first_name}",
    'position': employee.position,
    'period': f"{month}/{year}",
    'basic_salary': basic_salary,
    'allowances': allowances,
    'total_allowances': total_allowances,
    'gross_salary': gross_salary,
    'taxes': taxes,
    'deductions': deductions,
    'net_salary': round(net_salary, 2),
    'calculation_date': datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
}

# Запис в аудит
create_audit_record(
    user_id=1, # ID поточного користувача

```

```

        action=f"Розрахунок заробітної плати для співробітника
{employee_id}",
        table_name="salaries",
        record_id=salary_components[0].salary_id      #      ID
створеного запису про зарплату
    )

    return salary_report

```

Розроблені функції забезпечують комплексний розрахунок заробітної плати з урахуванням актуальних норм українського законодавства. Всі параметри розрахунку, такі як ставки податків, правила відрахувань та коефіцієнти для надбавок, зберігаються в базі даних і можуть бути змінені через вебінтерфейс адміністратором системи без необхідності модифікації програмного коду.

Система підтримує різні моделі оплати праці, враховує стаж роботи, понаднормові години та роботу у вихідні дні. Крім того, система зберігає детальну історію всіх розрахунків у вигляді компонентів заробітної плати, що дозволяє в будь-який момент отримати повну інформацію про заробітну плату співробітника за будь-який період

3.3 Висновки за розділом

У третьому розділі описано розробку автоматизованої системи розрахунку заробітної плати. Було детально розглянуто структуру бази даних MySQL, яка лежить в основі системи. База даних складається з таблиць для зберігання інформації про користувачів, ролі, співробітників, заробітну плату, податки, відрахування та інші компоненти. Кожна таблиця містить необхідні атрибути та зв'язки для ефективного управління даними.

Було описано основні сторінки вебзастосунку, такі як головна сторінка, сторінки входу і реєстрації, панелі адміністратора і користувача, сторінки управління співробітниками, розрахунку заробітної плати, ності та

налаштувань профілю. Кожна сторінка виконує свою функцію і забезпечує зручний інтерфейс для роботи з системою.

Розроблена система дозволяє автоматизувати процеси введення даних про відпрацьовані години, розрахунку заробітної плати з урахуванням податків і відрахувань, управління користувачами і співробітниками. Вона забезпечує точність обчислень, зручність використання та безпеку даних.

4 РЕКОМЕНДАЦІЇ ІЗ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ

4.1 Інструкції користування системою

Сторінка реєстрації користувача є першим компонентом вебзастосунку, який забезпечує новим користувачам можливість створення облікового запису. Дана сторінка містить форми для введення особистої інформації, такої як ім'я, прізвище, електронна пошта, ім'я користувача та пароль (рис. 4.1).

Сторінка реєстрації користувача має заголовок "Створіть свій обліковий запис". Вона містить наступні елементи:

- Форма "Ім'я" з текстовим полем.
- Форма "Прізвище" з текстовим полем.
- Форма "Електронна пошта" з текстовим полем.
- Форма "Ім'я користувача" з текстовим полем.
- Форма "Пароль" з текстовим полем.
- Підказка під полем пароля: "Використовуйте 8 або більше символів з комбінацією літер, цифр та символів".
- Галочка з текстом: "Я погоджуюсь з Умовами обслуговування та Політикою конфіденційності".
- Жовтий кнопка "Зареєструватися в Асте".

Рисунок 4.1 - Сторінка реєстрації користувача

Для забезпечення безпеки пароля, користувачам рекомендується використовувати комбінацію літер, цифр та символів, про що зазначено під полем введення пароля. Користувач також повинен погодитися з умовами обслуговування та політикою конфіденційності, поставивши відповідну галочку, перед тим як зможе завершити процес реєстрації.

На сторінці присутня кнопка "Зареєструватися", яка відправляє введені дані на сервер для подальшої обробки та зберігання в базі даних. У разі успішної реєстрації користувач перенаправляється на сторінку входу або до своєї панелі користувача.

Додатково, для користувачів, які вже мають обліковий запис, надається посилання для швидкого переходу на сторінку входу.

Сторінка табелювання призначена для відображення інформації про робочі години співробітників за певний період. Вона забезпечує зручний інтерфейс для перегляду даних про присутність та відпрацьовані години працівників (рис. 4.2).

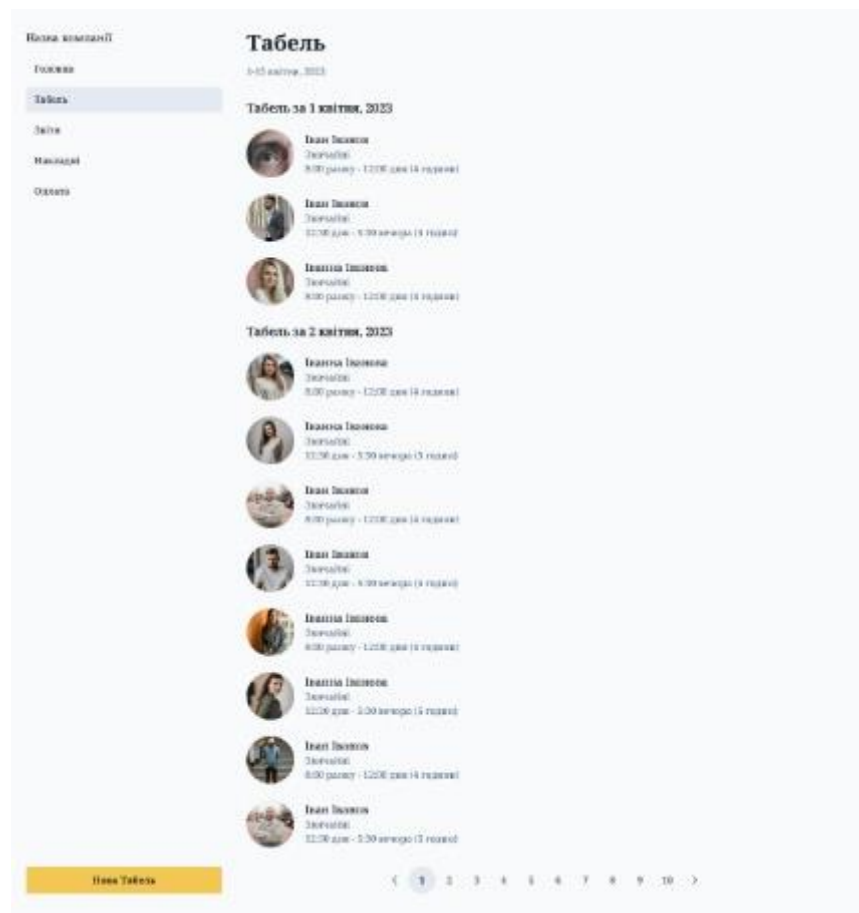


Рисунок 4.2 - Сторінка табелювання

На сторінці відображаються кілька розділів табелів, кожен з яких містить інформацію про співробітників за конкретну дату. Кожен запис у таблиці містить фотографію працівника, його ім'я, прізвище, та коротку

інформацію про відпрацьовані години та статус присутності (наприклад, "Присутній", "Відсутній").

Інтерфейс також включає навігаційне меню зліва, що дозволяє користувачам швидко перемикатися між різними розділами системи, такими як "Головна", "Табель", "Налаштування" та "Вихід".

У нижній частині сторінки знаходиться кнопка "Новий Табель", яка дозволяє адміністраторам або відповідальним особам створювати нові записи табелів для працівників. Навігаційні кнопки внизу сторінки дозволяють перемикатися між сторінками табелів, що особливо корисно для великих організацій з багатьма співробітниками.

Сторінка розрахунку заробітної плати призначена для автоматизованого обчислення заробітної плати співробітників на основі введених даних. Вона забезпечує користувачів інтерфейсом для введення різноманітних параметрів, таких як погодинна ставка, відпрацьовані години, премії та відрахування (рис. 4.3).

Обчислити вашу зарплату

Погодинна ставка
\$

Відпрацьовані години
40

Перевіряні години
5

Премії та компенсації

Опис	Сума
Премія	\$500

Відрахування

Опис	Сума
Медична страховка	\$150

Обчислити

Ваша виплатна відомість

Заробітна плата	\$1,000
Федеральний податок	\$100
Державний податок	\$50
Місцевий податок	\$25
Соціальне страхування	\$62
Медичне	\$14
Відрахування до податку	\$150
Відрахування після податку	\$90
Чистий дохід	\$549

Рисунок 4.3 - Сторінка розрахунку заробітної плати

На сторінці є кілька основних полів:

- погодинна ставка: поле для введення ставки заробітної плати за годину;
- відпрацьовані години: поле для введення загальної кількості відпрацьованих годин та перевірочних годин;
- премії та компенсації: поля для введення опису та суми премії або компенсації;
- відрахування: поля для введення опису та суми відрахувань, таких як медична страховка.

Після введення всіх необхідних даних користувач може натиснути кнопку "Обчислити", щоб система автоматично розрахувала загальну заробітну плату.

Сторінка налаштувань призначена для конфігурації різних параметрів вебзастосунку, надаючи користувачам можливість персоналізувати свій досвід використання системи. Вона поділена на кілька розділів, кожен з яких містить відповідні параметри для налаштування (рис. 4.4).

Налаштування

Основні налаштування

Тема
Вибір теми вигляду для всіх користувачів

Світла

Оновлення
Повідомлення про важливі оновлення

Оновлення

AI Data Usage
Дозволити використання функцій AI для покращення моделей

Оновлення

Мовні налаштування

Мова
Вибір мови інтерфейсу

Українська

Контакти для сповіщень

Електронна пошта
За замовчуванням використовується ваша адреса електронної пошти

admin@botco.com

Мова сповіщень
Вибір мови для сповіщень

Українська

Рисунок 4.4 – Сторінка налаштувань

Основні налаштування включають:

- тема: вибір теми вигляду для всіх користувачів, доступні опції "Світла" і "Темна";
- оновлення: вмикання або вимикання повідомлень про важливі оновлення;
- AI Data Usage: дозвіл на використання функцій штучного інтелекту для покращення моделей.

Мовні налаштування дозволяють користувачам змінювати мову інтерфейсу. У цьому розділі можна вибрати мову для системи, наприклад, "Українська".

Контакти для сповіщень включають:

- електронна пошта: поле для введення електронної адреси, яка використовується для отримання сповіщень;
- мова сповіщень: вибір мови, на якій надсилатимуться сповіщення.

Ця сторінка забезпечує гнучкість та зручність для користувачів, дозволяючи їм налаштовувати систему відповідно до своїх потреб та переваг.

Сторінка користувачів призначена для управління обліковими записами користувачів системи. Вона надає адміністраторам можливість переглядати, редагувати та додавати нових користувачів, а також управляти їхніми ролями та статусом активності (рис. 4.5).

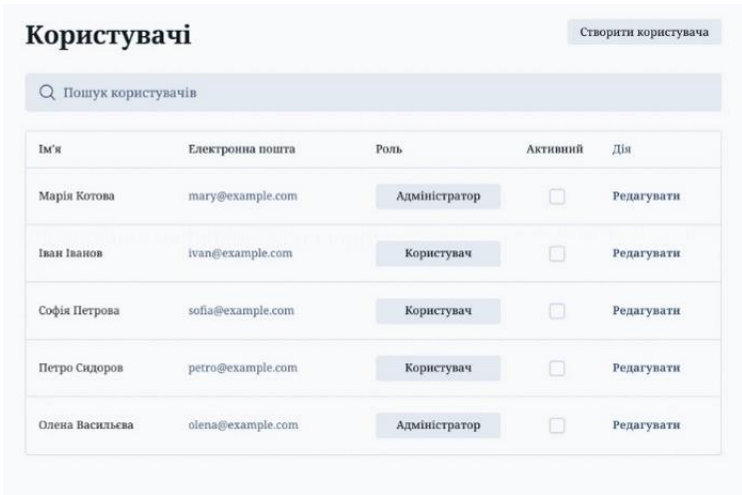


Рисунок 4.5 – Сторінка користувачі

Сторінка містить таблицю, яка включає такі колонки:

- ім'я: ім'я користувача;
- електронна пошта: адреса електронної пошти користувача;
- роль: роль користувача в системі (адміністратор або користувач);
- активний: стан активності облікового запису;
- дії: можливість редагування облікового запису користувача.

Кожен запис у таблиці користувачів містить кнопку "Редагувати", яка дозволяє адміністратору змінювати дані користувача, такі як його роль, електронну пошту або статус активності. Це забезпечує гнучке управління обліковими записами та дозволяє швидко вносити необхідні зміни.

Ця сторінка є важливим інструментом для адміністраторів системи, забезпечуючи ефективне управління користувачами та їхніми обліковими записами, що сприяє безпеці та організації роботи всередині системи.

4.2 Тестування програмного забезпечення

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, і автоматизована система розрахунку заробітної плати не є винятком. Одним з найбільш ефективних підходів до тестування такої системи є модульне тестування.

Модульне тестування - це метод тестування програмного забезпечення, при якому окремі модулі або компоненти системи перевіряються на коректність роботи незалежно від інших частин. Кожен модуль розглядається як самостійна одиниця, і для нього створюються спеціальні тестові випадки, які перевіряють його функціональність, обробку вхідних даних та генерацію очікуваних результатів.

Основні переваги модульного тестування для автоматизованої системи розрахунку заробітної плати наступні.

Рання ідентифікація помилок. Модульне тестування дозволяє виявляти помилки та недоліки в роботі окремих компонентів системи на ранніх етапах розробки. Це дає можливість швидко локалізувати проблемні місця і виправити їх до інтеграції модулів в єдину систему.

Спрощення процесу відлагодження. При виявленні помилок під час модульного тестування, набагато легше зрозуміти, в якому конкретно модулі знаходиться проблема. Це скорочує час на пошук і виправлення дефектів, оскільки розробники можуть зосередитися на конкретному модулі, а не шукати проблему у всій системі.

Для ефективного модульного тестування автоматизованої системи розрахунку заробітної плати необхідно визначити ключові модулі системи і розробити для них вичерпні тестові сценарії. Ці сценарії повинні покривати всі основні функції модуля, враховувати різні варіанти вхідних даних (коректні та некоректні) і перевіряти відповідність отриманих результатів очікуваним (табл. 4.1).

Тестування програмного забезпечення системи розрахунку заробітної плати проводилося відповідно до розробленого плану та сценаріїв тестування, які охоплюють усі ключові функціональні модулі системи. В процесі тестування були перевірені функції реєстрації та аутентифікації користувачів, включаючи обробку коректних та некоректних вхідних даних, що дозволило переконатися в надійності системи безпеки. Особлива увага приділялася перевірці модуля управління співробітниками, де були успішно протестовані процеси додавання, редагування та видалення інформації про співробітників, з метою забезпечення цілісності даних у базі. Центральний компонент системи – модуль розрахунку заробітної плати – пройшов детальне тестування з використанням різних сценаріїв, включаючи обчислення базової зарплати, надбавок, податків та відрахувань відповідно до українського законодавства, що підтвердило правильність реалізованих алгоритмів розрахунку.

Таблиця 4.1 – Сценарії тестування

Модуль	Сценарій	Очікуваний результат
Реєстрація та аутентифікація користувачів	Реєстрація нового користувача з коректними даними	Користувач успішно зареєстрований, дані збережені в базі даних
	Вхід в систему з правильними обліковими даними	Користувач успішно авторизований, перенаправлений на головну сторінку
	Вхід з неправильним паролем	Виведення повідомлення про помилку, вхід не виконаний
Управління співробітниками	Додавання нового співробітника з коректними даними	Співробітник доданий в базу даних, відображається в списку
	Редагування інформації про співробітника	Зміни збережені в базі даних, оновлена інформація відображається
	Видалення співробітника з системи	Запис про співробітника видалений з бази даних
Розрахунок заробітної плати	Введення коректних даних про відпрацьовані години	Розрахунок заробітної плати виконаний правильно, результат відображений
	Розрахунок заробітної плати з урахуванням надбавок і відрахувань	Сума заробітної плати обчислена коректно згідно з правилами
	Збереження результатів розрахунку в базі даних	Дані про заробітну плату збережені, доступні для подальшого використання
Налаштування користувача	Зміна персональних даних користувача	Внесені зміни збережені в базі даних, оновлена інформація відображається в профілі
	Зміна пароля з дотриманням вимог безпеки	Новий пароль збережений, старий пароль більше не дійсний

4.3 Підвищення ефективності

Розроблена система розрахунку заробітної плати демонструє значне підвищення ефективності порівняно з наявними на ринку програмними рішеннями. Цей висновок підтверджується результатами порівняльного аналізу ключових показників ефективності (табл. 4.2), який дозволяє об'єктивно оцінити переваги розробленої системи відносно конкурентних продуктів.

Таблиця 4.2 - Порівняння систем за ключовими показниками ефективності

Показник	Розроблена система	Турбо-бухгалтер	Колібрі	Фінанси без проблем	Леді Фін
Автоматичний розрахунок надбавок за стаж	Так	Частково	Ні	Так	Частково
Автоматичне визначення понаднормових годин	Так	Ні	Ні	Частково	Так
Автоматичне визначення роботи у вихідні	Так	Ні	Ні	Ні	Так
Параметризація без зміни коду	Так	Ні	Частково	Частково	Ні
Вебдоступ з різних пристроїв	Так	Частково	Так	Ні	Так
Інтеграція з іншими системами	Так	Так	Ні	Частково	Так

Аналізуючи дані з таблиці 4.3, можна відзначити, що основною перевагою розробленої системи є комплексна автоматизація процесів розрахунку різних компонентів заробітної плати. Система забезпечує повністю автоматичний розрахунок надбавок за стаж, що відсутнє у більшості конкурентів або реалізовано лише частково. Це досягається

завдяки впровадженню алгоритмів, описаних у розділі 3, які автоматично визначають тривалість роботи співробітника на підприємстві та застосовують відповідні коефіцієнти для розрахунку надбавок [14].

Впровадження розробленої системи дозволяє суттєво підвищити ефективність процесів нарахування заробітної плати, скоротити витрати на адміністрування за рахунок автоматизації рутинних операцій та забезпечити швидку адаптацію до змін законодавства завдяки гнучкій архітектурі та можливості параметризації без зміни програмного коду. Всі ці фактори роблять систему конкурентоспроможною на ринку програмного забезпечення для розрахунку заробітної плати та актуальною для впровадження на підприємствах різного розміру та форм власності [20].

4.4 Висновки за розділом

Четвертий розділ присвячений рекомендаціям із впровадження автоматизованої системи розрахунку заробітної плати. Були розроблені детальні інструкції для користувачів системи, які описують процеси реєстрації, табелювання, розрахунку заробітної плати, налаштувань профілю. Ці інструкції забезпечують зручність використання системи та дозволяють користувачам ефективно виконувати свої завдання.

Для забезпечення надійності та якості системи було проведено модульне тестування. Цей підхід дозволяє перевіряти коректність роботи окремих компонентів незалежно від інших частин системи. Були розроблені тестові сценарії для ключових модулів, таких як реєстрація та аутентифікація користувачів, управління співробітниками, розрахунок заробітної плати, налаштування профілю. Модульне тестування дозволяє рано виявляти помилки, спрощує процес відлагодження, забезпечує надійність системи, полегшує рефакторинг та слугує формою документації коду.

Таким чином, розроблені інструкції користувача та проведене модульне тестування забезпечують зручність використання та надійність

автоматизованої системи розрахунку заробітної плати. Вони є важливими складовими успішного впровадження системи на підприємстві та дозволяють максимально ефективно використовувати її можливості для оптимізації процесів нарахування заробітної плати.

ВИСНОВКИ

В бакалаврській роботі розглянуто програмне забезпечення розрахунку заробітної плати. В ході роботи було проаналізовано актуальність розробки такої системи, визначено основні вимоги до функціоналу та інтерфейсу, обрано оптимальну архітектуру та технології для реалізації.

Аналіз предметної області показав, що автоматизація процесів розрахунку заробітної плати є важливим завданням для сучасних підприємств, оскільки дозволяє підвищити ефективність управління людськими ресурсами, зменшити кількість помилок у нарахуваннях та забезпечити відповідність законодавчим нормам. При цьому необхідно враховувати складність структур заробітної плати, часті зміни в законодавстві та необхідність інтеграції з іншими корпоративними системами.

Для розробки програмного забезпечення було обрано клієнт-серверну архітектуру, яка забезпечує розподіл навантаження, масштабованість, безпеку та зручність підтримки. В якості мови програмування для вебзастосунку використано Python, а для бази даних - MySQL. Ці технології дозволяють швидко розробляти надійні та масштабовані рішення.

Розроблена система складається з бази даних, яка містить таблиці для зберігання інформації про користувачів, співробітників, заробітну плату, податки та відрахування, а також програмної частини, яка реалізує функціонал введення даних, розрахунку заробітної плати, управління користувачами. Система забезпечує точність обчислень, зручність використання та безпеку даних.

Для успішного впровадження програмного забезпечення розрахунку заробітної плати було розроблено детальні інструкції користувача та проведено модульне тестування. Інструкції описують процеси реєстрації, табелювання, розрахунку заробітної плати, налаштувань профілю, що забезпечує зручність використання системи. Модульне тестування дозволяє

перевіряти коректність роботи окремих компонентів, виявляти помилки на ранніх етапах та забезпечувати надійність системи.

Таким чином, розроблене програмне забезпечення розрахунку заробітної плати відповідає сучасним вимогам бізнесу та дозволяє оптимізувати процеси нарахування заробітної плати, зменшити кількість помилок та підвищити ефективність управління людськими ресурсами на підприємстві. Подальші дослідження можуть бути спрямовані на розширення функціоналу системи, інтеграцію з іншими корпоративними додатками та адаптацію до специфічних потреб різних галузей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Web-based payroll management system: design, implementation, and evaluation [Electronic resource] // Journal of Electrical Systems and Information Technology. – Access mode: <https://jesit.springeropen.com/articles/10.1186/s43067-020-0004-3>.
2. Automated Payroll System Development Guide and Case Study [Electronic resource] // Relevant Software. – Access mode: <https://relevant.software/blog/automated-payroll-system/>.
3. Complete Guide to Successful Payroll System Implementation [Electronic resource] // Jemini. – Access mode: <https://www.jemini.com/blog/successful-payroll-system-implementation>.
4. Payroll System Implementation: Challenges & Best Practices [Electronic resource] // Select Software Reviews. – Access mode: <https://www.selectsoftwarereviews.com/blog/payroll-system-implementation>.
5. Garcia-Molina H., Ullman J. D., Widom J. Database Systems: The Complete Book. – 2nd ed. – Upper Saddle River, NJ: Pearson, 2008. – 1248 p.
6. Hernandez M. J. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. – 3rd ed. – Boston, MA: Addison-Wesley, 2013. – 672 p.
7. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – 1st ed. – Sebastopol, CA: O'Reilly Media, 2017. – 616 p.
8. Silva R. MySQL Crash Course: A Hands-on Introduction to Database Development. – 1st ed. – Indianapolis, IN: Sams Publishing, 2005. – 480 p.
9. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. – 3rd ed. – San Francisco, CA: No Starch Press, 2018. – 472 p.
10. Simpson K. You Don't Know JS: Scope & Closures. – 1st ed. – Sebastopol, CA: O'Reilly Media, 2014. – 98 p.

11. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development. – 1st ed. – Indianapolis, IN: Wiley, 2014. – 640 p.
12. Freeman A. Pro ASP.NET Core 3. – 1st ed. – New York, NY: Apress, 2020. – 1024 p.
13. Database Design [Electronic resource]. – 2nd ed. – Open Textbook Library. – Access mode: <https://open.umn.edu/opentextbooks/textbooks/database-design-2nd-edition>.
14. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. – 2nd ed. – New York, NY: Apress, 2009. – 447 p.
15. Vincent B. Django for Professionals: Production websites with Python & Django. – 1st ed. – Charleston, SC: WelcomeToCode, 2020. – 320 p.
16. Grinberg M. Flask Web Development: Developing Web Applications with Python. – 2nd ed. – Sebastopol, CA: O'Reilly Media, 2018. – 258 p.
17. Bader D. Python Tricks: A Buffet of Awesome Python Features. – 1st ed. – San Francisco, CA: Dan Bader, 2017. – 301 p.
18. Beighley L., Morrison M. Head First SQL. – 1st ed. – Sebastopol, CA: O'Reilly Media, 2007. – 608 p.
19. Matplotlib: Visualization with Python [Electronic resource] – Access mode: <https://matplotlib.org/>.
20. Pandas Documentation [Electronic resource] – Access mode: <https://pandas.pydata.org/docs/>.
21. Docker Documentation [Electronic resource] – Access mode: <https://docs.docker.com/>.
22. Unicorn Documentation [Electronic resource] – Access mode: <https://docs.gunicorn.org/en/stable/>.
23. Django ORM Cookbook [Electronic resource] – Access mode: <https://books.agiliq.com/projects/django-orm-cookbook/en/latest/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення призначене для автоматизації процесу розрахунку заробітної плати в організаціях різного розміру та форм власності.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему "Програмне забезпечення для розрахунку заробітної плати", затверджене наказом Національного університету "Запорізька політехніка" № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для автоматизації процесу обліку робочого часу працівників, розрахунку заробітної плати, податків та відрахувань відповідно до чинного законодавства України.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

При розробці програмного забезпечення необхідно забезпечити:

- реєстрацію та управління обліковими записами користувачів системи;
- управління даними співробітників (додавання, редагування, видалення);
- облік відпрацьованого часу працівників;
- розрахунок базової заробітної плати відповідно до тарифних ставок;

- розрахунок надбавок за стаж, понаднормову роботу та роботу у вихідні дні;
- розрахунок податків та зборів згідно актуальних ставок законодавства України;
- розрахунок відрахувань із заробітної плати;
- формування та збереження результатів розрахунку в базі даних;
- можливість експорту результатів у формат PDF.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс користувача передбачається як вебзастосунок з адаптивним дизайном, доступний через браузер. Інтерфейс має бути інтуїтивно зрозумілим, з можливістю швидкої навігації між різними розділами системи.

A.3.3 Вимоги до надійності

Вимоги до надійності:

- забезпечення цілісності даних при їх зберіганні та обробці;
- захист від несанкціонованого доступу через систему авторизації та автентифікації;
- автоматичне резервне копіювання бази даних;
- забезпечення точності розрахунків відповідно до чинного законодавства.

A.3.4 Умови експлуатації

Для експлуатації потрібен сервер під управлінням ОС Linux або Windows Server, з встановленим Python 3.10+, Django 4.0+, MySQL 8.0, Nginx, Gunicorn. Для клієнтської частини потрібен будь-який сучасний веббраузер з підтримкою HTML5, CSS3 та JavaScript.

A.3.5 Вимоги до складу та параметрів технічних засобів

Сервер:

- процесор: Intel Xeon E5 або аналогічний, не менше 4 ядер;
- оперативна пам'ять: не менше 8 Гб;
- дисковий простір: не менше 100 Гб;
- мережеве підключення: не менше 100 Мбіт/с.

Клієнтська станція:

- процесор: Intel Core i3 або аналогічний;
- оперативна пам'ять: не менше 4 Гб;
- мережеве підключення: не менше 10 Мбіт/с.

A.3.6 Вимоги до маркування і пакування

Програмне забезпечення поширюється у вигляді Docker-контейнерів або вихідного коду з інструкцією щодо розгортання. На пакуванні має бути назва "Програмне забезпечення для розрахунку заробітної плати" та версія програмного продукту.

ДОДАТОК Б
Текст програми

```

// Frontend JavaScript для форми розрахунку заробітної плати
document.addEventListener('DOMContentLoaded', function() {
  // Отримуємо елементи форми
  const calculateForm = document.getElementById('calculateSalaryForm');
  const calculateButton = document.getElementById('calculateButton');

  // Обробник події для кнопки "Обчислити"
  calculateButton.addEventListener('click', function(e) {
    e.preventDefault();

    // Збираємо дані з форми
    const employeeId = document.getElementById('employeeSelect').value;
    const monthSelect = document.getElementById('monthSelect').value;
    const yearSelect = document.getElementById('yearSelect').value;

    // Валідація введених даних
    if (!employeeId || !monthSelect || !yearSelect) {
      showAlert('Будь ласка, заповніть всі обов'язкові поля', 'error');
      return;
    }

    // Відображення індикатора завантаження
    showLoadingIndicator();

    // Відправка даних на сервер для розрахунку
    fetch('/api/calculate-salary/', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'X-CSRFToken': getCsrfToken()
      },
      body: JSON.stringify({
        employee_id: employeeId,
        month: monthSelect,
        year: yearSelect
      })
    })
    .then(response => {
      if (!response.ok) {
        throw new Error('Помилка сервера при розрахунку заробітної плати');
      }
      return response.json();
    })
    .then(data => {

```

```

        // Виведення результатів розрахунку
        hideLoadingIndicator();
        displayCalculationResults(data);
    })
    .catch(error => {
        hideLoadingIndicator();
        console.error('Помилка:', error);
        showAlert(error.message, 'error');
    });
});

// Функція для отримання CSRF токєну з куки
function getCsrftoken() {
    const name = 'csrftoken';
    let cookieValue = null;
    if (document.cookie && document.cookie !== "") {
        const cookies = document.cookie.split(';');
        for (let i = 0; i < cookies.length; i++) {
            const cookie = cookies[i].trim();
            if (cookie.substring(0, name.length + 1) === (name + '=')) {
                cookieValue = decodeURIComponent(cookie.substring(name.length +
1));
                break;
            }
        }
    }
    return cookieValue;
}

// Функція для відображення результатів розрахунку
function displayCalculationResults(data) {
    const resultsContainer = document.getElementById('calculationResults');
    resultsContainer.innerHTML = "";

    // Створюємо заголовок результатів
    const header = document.createElement('div');
    header.className = 'results-header';
    header.innerHTML = `
        <h3>Результати розрахунку заробітної плати</h3>
        <p><strong>Співробітник:</strong> ${data.employee_name}</p>
        <p><strong>Посада:</strong> ${data.position}</p>
        <p><strong>Період:</strong> ${data.period}</p>
    `;
    resultsContainer.appendChild(header);
}

```

```

// Створюємо блок з основною інформацією
const mainInfo = document.createElement('div');
mainInfo.className = 'main-info';
mainInfo.innerHTML = `
  <div class="info-row">
    <div class="info-label">Базова заробітна плата:</div>
    <div class="info-value">${formatCurrency(data.basic_salary)}</div>
  </div>
  <div class="info-row">
    <div class="info-label">Загальні надбавки:</div>
    <div class="info-value">${formatCurrency(data.total_allowances)}</div>
  </div>
  <div class="info-row">
    <div class="info-label">Загальна нарахована сума:</div>
    <div class="info-value">${formatCurrency(data.gross_salary)}</div>
  </div>
`;
resultsContainer.appendChild(mainInfo);

// Створюємо блок з надбавками
if (Object.keys(data.allowances).length > 0) {
  const allowancesBlock = document.createElement('div');
  allowancesBlock.className = 'details-block';
  allowancesBlock.innerHTML = '<h4>Надбавки:</h4>';

  const allowancesList = document.createElement('ul');
  for (const [name, amount] of Object.entries(data.allowances)) {
    const item = document.createElement('li');
    item.innerHTML = `
      ${formatAllowanceName(name)}:
      ${formatCurrency(amount)}
    `;
    allowancesList.appendChild(item);
  }
  allowancesBlock.appendChild(allowancesList);
  resultsContainer.appendChild(allowancesBlock);
}

// Створюємо блок з податками
const taxesBlock = document.createElement('div');
taxesBlock.className = 'details-block';
taxesBlock.innerHTML = '<h4>Податки та збори:</h4>';

const taxesList = document.createElement('ul');
for (const [name, amount] of Object.entries(data.taxes)) {
  if (name !== 'total_tax') {

```

```

        const item = document.createElement('li');
        item.innerHTML = `${formatTaxName(name)}:
    ${formatCurrency(amount)}`;
        taxesList.appendChild(item);
    }
}
taxesBlock.appendChild(taxesList);
resultsContainer.appendChild(taxesBlock);

// Створюємо блок з відрахуваннями
if (Object.keys(data.deductions).length > 0) {
    const deductionsBlock = document.createElement('div');
    deductionsBlock.className = 'details-block';
    deductionsBlock.innerHTML = '<h4>Відрахування:</h4>';

    const deductionsList = document.createElement('ul');
    for (const [name, amount] of Object.entries(data.deductions)) {
        if (name !== 'total_deduction') {
            const item = document.createElement('li');
            item.innerHTML = `${formatDeductionName(name)}:
    ${formatCurrency(amount)}`;
            deductionsList.appendChild(item);
        }
    }
    deductionsBlock.appendChild(deductionsList);
    resultsContainer.appendChild(deductionsBlock);
}

// Створюємо підсумковий блок
const summary = document.createElement('div');
summary.className = 'summary';
summary.innerHTML = `
    <div class="summary-row">
        <div class="summary-label">Всього податків та зборів:</div>
        <div class="summary-
value">${formatCurrency(data.taxes.total_tax)}</div>
    </div>
    <div class="summary-row">
        <div class="summary-label">Всього відрахувань:</div>
        <div class="summary-
value">${formatCurrency(data.deductions.total_deduction)}</div>
    </div>
    <div class="summary-row total">
        <div class="summary-label">Сума до виплати:</div>

```

```

        <div
value">${formatCurrency(data.net_salary)}</div>
        </div>
        <div class="calculation-date">
            Дата розрахунку: ${data.calculation_date}
        </div>
    `;
    resultsContainer.appendChild(summary);

    // Додаємо кнопки для дій з результатами
    const actionsBlock = document.createElement('div');
    actionsBlock.className = 'actions-block';
    actionsBlock.innerHTML = `
        <button id="printResultsBtn" class="btn btn-
secondary">Друкувати</button>
        <button id="exportPdfBtn" class="btn btn-secondary">Експорт в
PDF</button>
        <button id="saveResultsBtn" class="btn btn-primary">Зберегти
розрахунок</button>
    `;
    resultsContainer.appendChild(actionsBlock);

    // Додаємо обробники подій для кнопок дій
    document.getElementById('printResultsBtn').addEventListener('click',
printResults);
    document.getElementById('exportPdfBtn').addEventListener('click',
exportToPdf);
    document.getElementById('saveResultsBtn').addEventListener('click',
saveCalculation);

    // Показуємо контейнер з результатами
    resultsContainer.style.display = 'block';

    // Прокручуємо до результатів
    resultsContainer.scrollIntoView({ behavior: 'smooth' });
}

// Допоміжні функції
function formatCurrency(amount) {
    return new Intl.NumberFormat('uk-UA', { style: 'currency', currency: 'UAH'
}).format(amount);
}

function formatAllowanceName(name) {
    const nameMap = {

```

```

    'experience_bonus': 'Надбавка за стаж',
    'overtime_bonus': 'Надбавка за понаднормову роботу',
    'weekend_bonus': 'Надбавка за роботу у вихідні дні'
  };
  return nameMap[name] || name;
}

function formatTaxName(name) {
  const nameMap = {
    'pdfo': 'Податок на доходи фізичних осіб',
    'military': 'Військовий збір',
    'esv': 'Єдиний соціальний внесок'
  };
  return nameMap[name] || name;
}

function formatDeductionName(name) {
  const nameMap = {
    'medical_insurance': 'Медичне страхування',
    'union_fee': 'Профспілковий внесок'
  };
  return nameMap[name] || name;
}

function showLoadingIndicator() {
  const loadingIndicator = document.getElementById('loadingIndicator');
  loadingIndicator.style.display = 'flex';
}

function hideLoadingIndicator() {
  const loadingIndicator = document.getElementById('loadingIndicator');
  loadingIndicator.style.display = 'none';
}

function showAlert(message, type) {
  const alertContainer = document.getElementById('alertContainer');
  const alert = document.createElement('div');
  alert.className = `alert alert-${type} alert-dismissible fade show`;
  alert.role = 'alert';
  alert.innerHTML = `
    ${message}
    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
  `;
  alertContainer.appendChild(alert);
}

```

```

// Автоматичне закриття повідомлення через 5 секунд
setTimeout(() => {
  alert.classList.remove('show');
  setTimeout(() => {
    alertContainer.removeChild(alert);
  }, 150);
}, 5000);
}

function printResults() {
  window.print();
}

function exportToPdf() {
  // Реалізація експорту в PDF
  showAlert('Експорт в PDF-файл...', 'info');

  const employeeId = document.getElementById('employeeSelect').value;
  const monthSelect = document.getElementById('monthSelect').value;
  const yearSelect = document.getElementById('yearSelect').value;

  fetch('/api/export-salary-pdf/', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'X-CSRFToken': getCsrfToken()
    },
    body: JSON.stringify({
      employee_id: employeeId,
      month: monthSelect,
      year: yearSelect
    })
  })
  .then(response => {
    if (!response.ok) {
      throw new Error('Помилка при експорті в PDF');
    }
    return response.blob();
  })
  .then(blob => {
    // Створення посилання для завантаження файлу
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.style.display = 'none';

```

```

    a.href = url;
    a.download = `Заробітна_плата_${monthSelect}_${yearSelect}.pdf`;
    document.body.appendChild(a);
    a.click();
    window.URL.revokeObjectURL(url);
    showAlert('PDF-файл успішно створено', 'success');
  })
  .catch(error => {
    console.error('Помилка:', error);
    showAlert(error.message, 'error');
  });
}

function saveCalculation() {
  // Реалізація збереження розрахунку в базі даних
  showAlert('Збереження розрахунку...', 'info');

  const employeeId = document.getElementById('employeeSelect').value;
  const monthSelect = document.getElementById('monthSelect').value;
  const yearSelect = document.getElementById('yearSelect').value;

  fetch('/api/save-salary-calculation/', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'X-CSRFToken': getCsrfToken()
    },
    body: JSON.stringify({
      employee_id: employeeId,
      month: monthSelect,
      year: yearSelect
    })
  })
  .then(response => {
    if (!response.ok) {
      throw new Error('Помилка при збереженні розрахунку');
    }
    return response.json();
  })
  .then(data => {
    showAlert('Розрахунок успішно збережено', 'success');
  })
  .catch(error => {
    console.error('Помилка:', error);
    showAlert(error.message, 'error');
  });
}

```

```

    });
}
});

// Python код (backend) для обробки запиту на розрахунок заробітної плати
"""
from django.http import JsonResponse
from django.views.decorators.http import require_http_methods
from django.views.decorators.csrf import csrf_protect
import json
from datetime import datetime, date, timedelta
from decimal import Decimal
import logging

from .models import Employee, WorkHours, Salaries, SalaryComponents, Taxes,
Deductions
from .utils import create_audit_record, get_standard_monthly_hours

logger = logging.getLogger(__name__)

@csrf_protect
@require_http_methods(["POST"])
def calculate_salary(request):
    """
    API-ендпоінт для розрахунку заробітної плати
    """
    try:
        # Парсинг JSON-даних з запиту
        data = json.loads(request.body)
        employee_id = data.get('employee_id')
        month = int(data.get('month'))
        year = int(data.get('year'))

        if not employee_id or not month or not year:
            return JsonResponse({'error': 'Відсутні обов'язкові параметри'},
                                status=400)

        # Розрахунок заробітної плати
        salary_report = calculate_final_salary(employee_id, month, year)

        # Логування успішного розрахунку
        logger.info(f"Успішний розрахунок заробітної плати для співробітника
{employee_id}")

        return JsonResponse(salary_report)

```

```

except json.JSONDecodeError:
    logger.error("Невірний формат JSON")
    return JsonResponse({'error': 'Невірний формат запиту'}, status=400)

except Employee.DoesNotExist:
    logger.error(f"Співробітника з ID {employee_id} не знайдено")
    return JsonResponse({'error': 'Співробітника не знайдено'}, status=404)

except Exception as e:
    logger.error(f"Помилка при розрахунку заробітної плати: {str(e)}")
    return JsonResponse({'error': str(e)}, status=500)

def calculate_final_salary(employee_id, month, year):
    """
    Головна функція для розрахунку кінцевої заробітної плати

    Args:
        employee_id (int): Ідентифікатор працівника
        month (int): Місяць розрахунку
        year (int): Рік розрахунку

    Returns:
        dict: Повний розрахунок заробітної плати
    """
    # Розрахунок базової заробітної плати
    basic_salary = calculate_basic_salary(employee_id, month, year)

    # Розрахунок загальної нарахованої суми з урахуванням надбавок
    allowances = calculate_allowances(employee_id, basic_salary, month, year)
    total_allowances = sum(allowances.values())
    gross_salary = basic_salary + total_allowances

    # Розрахунок податків
    taxes = calculate_taxes(gross_salary)

    # Розрахунок відрахувань
    deductions = calculate_deductions(employee_id, gross_salary)

    # Розрахунок чистої заробітної плати
    net_salary = gross_salary - taxes['total_tax'] - deductions['total_deduction']

    # Створення компонентів заробітної плати
    salary_components = calculate_salary_components(
        employee_id, basic_salary, taxes, deductions, month, year
    )

```

```

)

# Отримання даних про працівника
employee = Employee.objects.get(employee_id=employee_id)

# Формування звіту про заробітну плату
salary_report = {
    'employee_id': employee_id,
    'employee_name': f"{employee.last_name} {employee.first_name}",
    'position': employee.position,
    'period': f"{month}/{year}",
    'basic_salary': float(basic_salary),
    'allowances': {k: float(v) for k, v in allowances.items()},
    'total_allowances': float(total_allowances),
    'gross_salary': float(gross_salary),
    'taxes': {k: float(v) for k, v in taxes.items()},
    'deductions': {k: float(v) for k, v in deductions.items()},
    'net_salary': float(round(net_salary, 2)),
    'calculation_date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
}

# Запис в аудит
create_audit_record(
    user_id=request.user.id, # ID поточного користувача
    action=f"Розрахунок заробітної плати для співробітника {employee_id}",
    table_name="salaries",
    record_id=salary_components[0].salary_id # ID створеного запису про
зарплату
)

return salary_report

def calculate_basic_salary(employee_id, month, year):
    """
    Розрахунок базової заробітної плати працівника з використанням даних з
    БД

    Args:
        employee_id (int): Ідентифікатор працівника
        month (int): Місяць розрахункового періоду
        year (int): Рік розрахункового періоду

    Returns:
        float: Базова заробітна плата
    """

```

```

# Отримання даних про працівника з бази даних
employee = Employee.objects.get(employee_id=employee_id)

# Отримання відпрацьованих годин за місяць
start_date = date(year, month, 1)
if month == 12:
    end_date = date(year + 1, 1, 1) - timedelta(days=1)
else:
    end_date = date(year, month + 1, 1) - timedelta(days=1)

work_hours_records = WorkHours.objects.filter(
    employee_id=employee_id,
    date_worked__gte=start_date,
    date_worked__lte=end_date
)

# Підрахунок загальної кількості відпрацьованих годин
total_hours = 0
for record in work_hours_records:
    total_hours += record.hours_worked

# Розрахунок базової заробітної плати
if hasattr(employee, 'hourly_rate'):
    # Для працівників з погодинною оплатою
    basic_salary = total_hours * employee.hourly_rate
else:
    # Для працівників з фіксованою місячною ставкою
    standard_monthly_hours = get_standard_monthly_hours(month, year) #
    Стандартна кількість робочих годин на місяць
    basic_salary = employee.basic_salary * (total_hours /
    standard_monthly_hours)

    return round(basic_salary, 2)

def calculate_taxes(gross_salary):
    """
    Розрахунок податків згідно українського законодавства
    з використанням ставок з бази даних

    Args:
        gross_salary (float): Нарахована заробітна плата до вирахування податків

    Returns:
        dict: Словник з сумами податків та зборів
    """

```

```

# Отримання актуальних ставок податків з бази даних
taxes = Taxes.objects.all()

# Формування словника з актуальними ставками
tax_rates = {}
for tax in taxes:
    tax_rates[tax.tax_name] = tax.tax_rate / 100 # Конвертація відсотків у
десятькове число

# Розрахунок податків
taxes_result = {}
total_tax = 0

for tax_name, tax_rate in tax_rates.items():
    tax_amount = gross_salary * tax_rate
    taxes_result[tax_name] = round(tax_amount, 2)
    total_tax += tax_amount

taxes_result['total_tax'] = round(total_tax, 2)

return taxes_result

def calculate_deductions(employee_id, gross_salary):
    """
    Розрахунок відрахувань із заробітної плати

    Args:
        employee_id (int): Ідентифікатор працівника
        gross_salary (float): Нарахована заробітна плата

    Returns:
        dict: Словник з відрахуваннями
    """
    # Отримання відрахувань з бази даних
    deductions_list = Deductions.objects.all()

    # Розрахунок відрахувань
    deductions = {}
    total_deduction = 0

    for deduction in deductions_list:
        # Визначення суми відрахування
        if deduction.amount > 0:
            deduction_amount = deduction.amount # Фіксована сума
        else:

```

```

    # Відрахування у відсотках від зарплати
    deduction_amount = gross_salary * (deduction.percentage / 100)

    deductions[deduction.deduction_name] = round(deduction_amount, 2)
    total_deduction += deduction_amount

deductions['total_deduction'] = round(total_deduction, 2)

return deductions

def calculate_allowances(employee_id, basic_salary, month, year):
    """
    Розрахунок надбавок для працівника

    Args:
        employee_id (int): Ідентифікатор працівника
        basic_salary (float): Базова заробітна плата
        month (int): Місяць розрахунку
        year (int): Рік розрахунку

    Returns:
        dict: Словник з надбавками
    """
    # Отримання даних про працівника
    employee = Employee.objects.get(employee_id=employee_id)

    # Ініціалізація словника надбавок
    allowances = {}

    # Надбавка за стаж роботи
    hire_date = employee.hire_date
    current_date = date(year, month, 1)
    years_of_experience = (current_date.year - hire_date.year) - (
        (current_date.month, current_date.day) < (hire_date.month, hire_date.day)
    )

    if years_of_experience >= 3:
        # 10% надбавки за кожні 3 роки стажу, максимум 50%
        experience_percent = min(10 * (years_of_experience // 3), 50) / 100
        allowances['experience_bonus'] = round(basic_salary * experience_percent,
2)

    # Надбавка за понаднормову роботу
    start_date = date(year, month, 1)
    if month == 12:

```

```

    end_date = date(year + 1, 1, 1) - timedelta(days=1)
else:
    end_date = date(year, month + 1, 1) - timedelta(days=1)

# Підрахунок понаднормових годин (понад 8 годин на день)
overtime_hours = 0
work_hours_records = WorkHours.objects.filter(
    employee_id=employee_id,
    date_worked__gte=start_date,
    date_worked__lte=end_date
)

daily_hours = {}
for record in work_hours_records:
    day = record.date_worked.day
    if day in daily_hours:
        daily_hours[day] += record.hours_worked
    else:
        daily_hours[day] = record.hours_worked

for day, hours in daily_hours.items():
    if hours > 8:
        overtime_hours += (hours - 8)

if overtime_hours > 0:
    # Надбавка за понаднормову роботу (подвійна ставка)
    hourly_rate = basic_salary / get_standard_monthly_hours(month, year)
    allowances['overtime_bonus'] = round(overtime_hours * hourly_rate * 2, 2)

# Надбавка за роботу у вихідні дні
weekend_hours = 0
for record in work_hours_records:
    if record.date_worked.weekday() >= 5: # 5 і 6 - субота і неділя
        weekend_hours += record.hours_worked

if weekend_hours > 0:
    # Надбавка за роботу у вихідні дні (подвійна ставка)
    hourly_rate = basic_salary / get_standard_monthly_hours(month, year)
    allowances['weekend_bonus'] = round(weekend_hours * hourly_rate * 2, 2)

return allowances

def calculate_salary_components(employee_id, basic_salary, taxes, deductions,
month, year):
    """

```

Розрахунок компонентів заробітної плати

Args:

employee_id (int): Ідентифікатор працівника
 basic_salary (float): Базова заробітна плата
 taxes (dict): Розраховані податки
 deductions (dict): Розраховані відрахування
 month (int): Місяць розрахунку
 year (int): Рік розрахунку

Returns:

list: Список компонентів заробітної плати
 """

Отримання загальної кількості відпрацьованих годин

total_hours = get_total_hours_worked(employee_id, month, year)

Розрахунок чистої зарплати

net_salary = basic_salary - taxes['total_tax'] - deductions['total_deduction']

Створення запису про заробітну плату

```
salary = Salaries(
    employee_id=employee_id,
    total_hours_worked=total_hours,
    total_salary=net_salary,
    salary_date=date(year, month, 1),
    created_at=datetime.now()
)
```

salary.save()

Створення компонентів заробітної плати

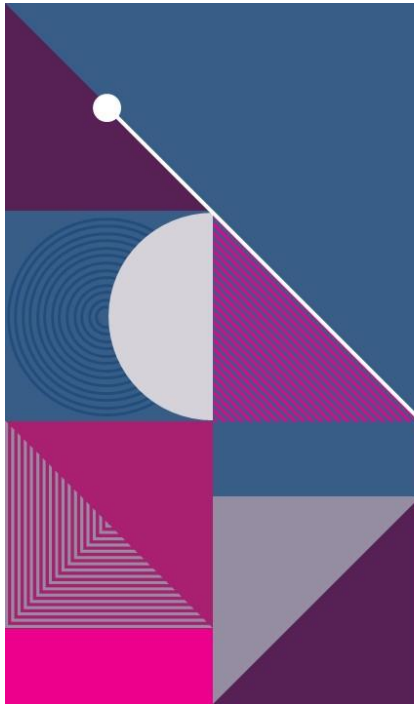
components = []

Базова складова

```
basic_component = SalaryComponents(
    salary_id=salary.salary_id,
    component_type='basic',
    amount=basic_salary,
    description='Базова заробітна плата',
    created_at=datetime.now()
)
```

```
basic_component.save()
components.append(basic_component)
```

ДОДАТОК В
Слайди презентації

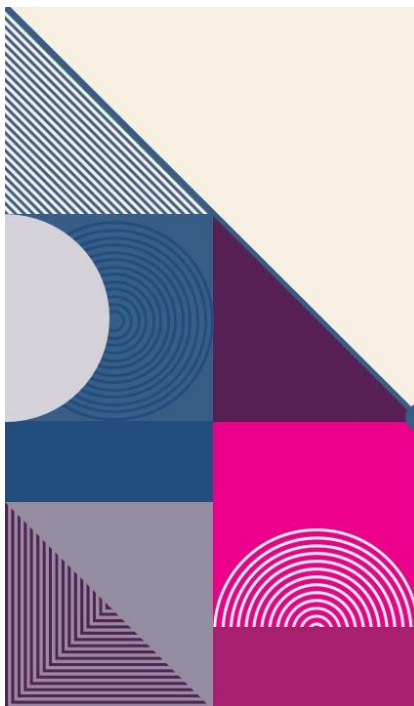


Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ.

- Виконав студент групи КНТ-212сп Ілля РУДЕНКО
- Керівник проєкту, к.т.н. Євген ГОФМАН

Рисунок В.1 – Слайд 1



МЕТА ТА ЗАВДАННЯ

Метою даної дипломної роботи є підвищення ефективності процесу розрахунку заробітної плати працівників шляхом розроблення та впровадження веборієнтованої автоматизованої системи. Система має забезпечувати облік відпрацьованого часу та розрахунок заробітної плати з урахуванням заданих параметрів.

Об'єкт — процес розроблення програмного забезпечення для автоматизації розрахунку заробітної плати.

Предмет — методи та інструменти розробки вебзастосунку для обліку робочого часу й нарахування заробітної плати.

Завдання:

- провести аналіз існуючих програмних рішень для автоматизації обліку заробітної плати;
- визначити функціональні та нефункціональні вимоги до розроблюваного вебзастосунку;
- спроектувати архітектуру системи та структуру бази даних;
- реалізувати програмне забезпечення на основі сучасних технологій веброзробки;
- здійснити тестування і перевірку коректності функціонування системи.

2

Рисунок В.2 – Слайд 2

ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБЛЕННЯ СИСТЕМИ

Критерій	Показник
Потреба в автоматизації	Необхідність для малих і середніх підприємств для підвищення ефективності обліку та звітності.
Переваги	- Спрощення роботи бухгалтерів; - Зменшення кількості помилок; - Оптимізація витрат часу; - Швидкий доступ до даних.
Недоліки	Висока вартість устаткування та програмних продуктів.
Зміни в організації обліку	Перехід від допоміжної функції до ключового елементу корпоративної інформаційної системи.
Проблеми автоматизації	- Складні алгоритми розрахунків; - Часті зміни законодавства; - Неоднозначне трактування норм; - ізнi алгоритми допоміжних обрахунків.
Підходи до розробки ПЗ	- Відкрита архітектура: модульні системи з адаптивним функціоналом; - Закриті системи: спеціалізовані пакети для окремих облікових завдань.

Рисунок В.3 – Слайд 3

ПЛАН РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРАХУНКУ ЗАРОБІТНОЇ ПЛАТИ

Етап розроблення	Основні завдання
Аналіз потреб	Дослідження поточних процедур розрахунку заробітної плати та виявлення потреб користувачів.
Визначення функціональних вимог	Специфікація функцій: розрахунок годин, податків, змін ставок, інтеграційні можливості.
Проектування системи	Створення архітектурної моделі взаємодії модулів і механізмів інтеграції з іншими системами.
Вибір технологій	Підбір технологічного стеку з урахуванням вимог до безпеки, масштабованості та продуктивності.
Розроблення бази даних	Створення бази даних для зберігання персональних даних, історії розрахунків і параметрів обліку.
Розробка інтерфейсу	Проектування простого й доступного користувацького інтерфейсу.
Тестування	Перевірка функціональної повноти, безпеки, витривалості та надійності програмного забезпечення.

Рисунок В.4 – Слайд 4



Рисунок В.5 – Слайд 5



Рисунок В.6 – Слайд 6

СХЕМА БАЗИ ДАНИХ

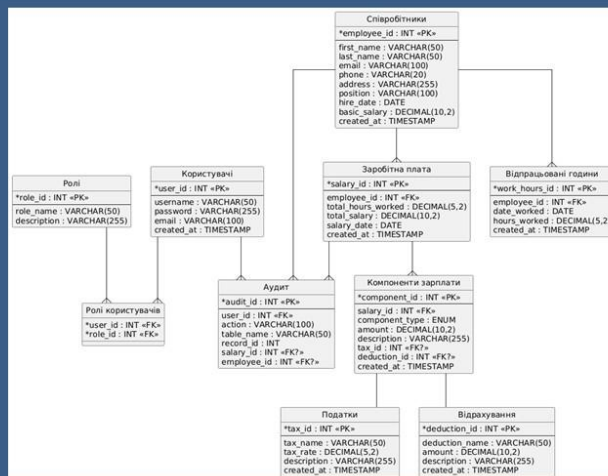


Рисунок В.7 – Слайд 7

ІНСТРУКЦІЇ КОРИСТУВАННЯ

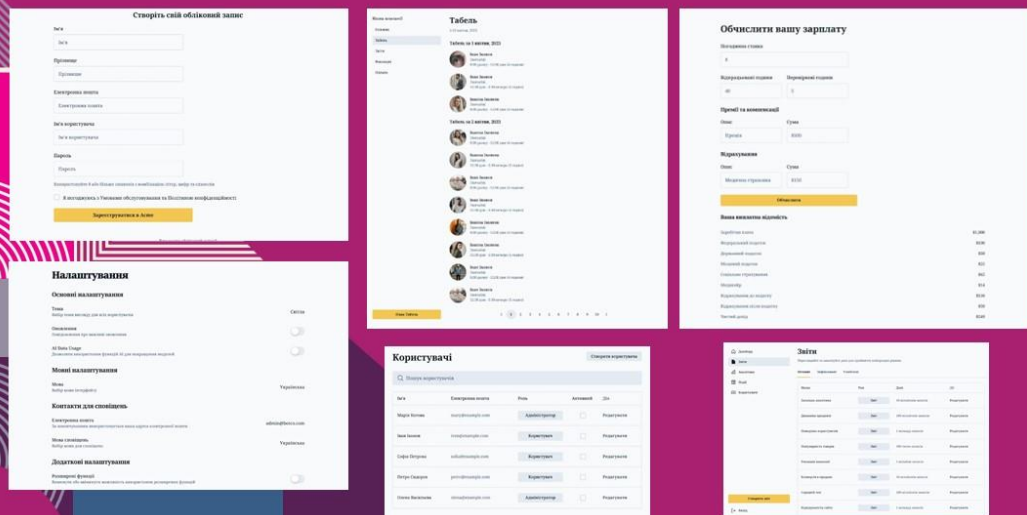


Рисунок В.8 – Слайд 8

СЦЕНАРІЇ ТЕСТУВАННЯ

Модуль	Сценарій	Очікуваний результат
Реєстрація та аутентифікація користувачів	Реєстрація нового користувача з коректними даними	Користувач успішно зареєстрований, дані збережені в базі даних
	Вхід в систему з правильними обліковими даними	Користувач успішно авторизований, перенаправлений на головну сторінку
	Вхід з неправильним паролем	Виведення повідомлення про помилку, вхід не виконаний
Управління співробітниками	Додавання нового співробітника з коректними даними	Співробітник доданий в базу даних, відображається в списку
	Редагування інформації про співробітника	Зміни збережені в базі даних, оновлена інформація відображається
	Видалення співробітника з системи	Запис про співробітника видалений з бази даних
Розрахунок заробітної плати	Введення коректних даних про відпрацьовані години	Розрахунок заробітної плати виконаний правильно, результат відображений
	Розрахунок заробітної плати з урахуванням надбавок і відрахувань	Сума заробітної плати обчислена коректно згідно з правилами
	Збереження результатів розрахунку в базі даних	Дані про заробітну плату збережені, доступні для подальшого використання
Налаштування користувача	Зміна персональних даних користувача	Внесені зміни збережені в базі даних, оновлена інформація відображається в профілі
	Зміна пароля з дотриманням вимог безпеки	Новий пароль збережений, старий пароль більше не дійсний

Рисунок В.9 – Слайд 9

ПОРІВНЯННЯ СИСТЕМ ЗА КЛЮЧОВИМИ ПОКАЗНИКАМИ ЕФЕКТИВНОСТІ

Показник	Розроблена система	Турбо-бухгалтер	Колібри	Фінанси без проблем	Леді Фін
Автоматичний розрахунок надбавок за стаж	Так	Частково	Ні	Так	Частково
Автоматичне визначення понаднормових годин	Так	Ні	Ні	Частково	Так
Автоматичне визначення роботи у вихідні	Так	Ні	Ні	Ні	Так
Параметризація без зміни коду	Так	Ні	Частково	Частково	Ні
Вебдоступ з різних пристроїв	Так	Частково	Так	Ні	Так
Інтеграція з іншими системами	Так	Так	Ні	Частково	Так

Рисунок В.10 – Слайд 10



ВИСНОВКИ

- Розроблено вебзастосунок для автоматизованого розрахунку заробітної плати з урахуванням податків та відрахувань відповідно до українського законодавства;
- Реалізовано зручний інтерфейс для адміністраторів та користувачів, включаючи облік працівників, перегляд історії нарахувань та генерацію звітів;
- В основі системи – сучасний технологічний стек: Python, Django, MySQL, Docker, що забезпечує масштабованість і надійність;
- Архітектура бази даних розроблена з урахуванням принципів нормалізації та взаємозв'язків між сутностями;
- Система дозволяє ефективно керувати обліковими даними, спрощує обчислення заробітної плати та підвищує прозорість розрахунків.

11

Рисунок В.11 – Слайд 11