НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету інформаційних **технологій** _____	**Завідувач кафедри** _____комп'ютерних наук_____
(назва факультету)	(назва кафедри)
_____ Болбот І.М._____	_____ Голуб Б.Л.__
(підпис) (ім'я ПРІЗВИЩЕ)	(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 20_ р.	"___" _____ 20_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Програмне забезпечення аналізу, контролю та обліку роботи співробітників компанії

Спеціальність 121 «Інженерія програмного забезпечення»
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
_к.фіз.-мат.н., доцент_____ _____ ____Віктор КИРИЧЕНКО __
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
__к.т.н., доцент_____ _____ _____Ганна ВАЙГАНГ _____
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ _____ Володимир ОКУЛІЧ _____
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
___комп'ютерних наук_____
к.т.н., доцент _____ Белла ГОЛУБ
(наук. ступінь, вч. звання) (підпис) ((ім'я ПРІЗВИЩЕ))
"____" _____20_____року

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ
_____Окуліч Володимир Анатолійович_____
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія програмного забезпечення»
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

Тема магістерської кваліфікаційної роботи **Програмне забезпечення аналізу, контролю та обліку роботи співробітників компанії**
затверджена наказом ректора НУБіП України від "_1_" _листопада__2024р. №1963 «С»
Термін подання завершеної роботи на кафедру __14 листопада 2025_____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: корпоративні датасети, статистичні показники, нормативно-правова база та довідкові системи; методи інженерії ПЗ для аналізу й обліку роботи персоналу; моделювання бізнес-процесів, проєктування архітектури та алгоритми обробки даних для підвищення ефективності й надійності інформаційної системи..

Перелік питань, що підлягають дослідженню:
1. Аналіз предметної області та сучасних методів HR-аналітики
2. Інформаційне, математичне й архітектурне проєктування системи
3. Реалізація програмного забезпечення
4. Експериментальне дослідження та оцінка ефективності системи

Перелік графічного матеріалу (за потреби) презентація, постер

Дата видачі завдання "__1__" __грудня_____2024_ р.

Керівник магістерської кваліфікаційної роботи _____ ___Вайганг Г. О._____
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Окуліч В. А._____
(ім'я ПРІЗВИЩЕ)

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

AI (Artificial Intelligence) – штучний інтелект.

API (Application Programming Interface) – програмний інтерфейс застосунку.

DBMS / СУБД (Database Management System) – система управління базами даних.

DLP (Data Loss Prevention) – система запобігання витоку даних.

DWH (Data Warehouse) – сховище даних.

ETL (Extract–Transform–Load) – процес вилучення, трансформації та завантаження даних у сховище.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

HR (Human Resources) – людські ресурси, персонал організації.

HRM (Human Resource Management) – управління персоналом (HR-менеджмент).

HR-аналітика – аналітика даних про персонал для підтримки управлінських рішень.

JSON (JavaScript Object Notation) – текстовий формат представлення структурованих даних.

KPI (Key Performance Indicators) – ключові показники ефективності.

ML (Machine Learning) – машинне навчання.

MLP (Multilayer Perceptron) – багатошаровий перцептрон.

NLP (Natural Language Processing) – обробка природної мови.

REST (Representational State Transfer) – стиль проєктування веб-сервісів.

SaaS (Software as a Service) – модель постачання програмного забезпечення як сервісу.

UI (User Interface) – інтерфейс користувача.

UX (User Experience) – користувацький досвід.

Telegram-бот – програмний агент, що працює в месенджері Telegram і взаємодіє з користувачами через чат-інтерфейс.

ВСТУП

Цифрова трансформація бізнес-процесів зумовлює зростання ролі інформаційних систем у сфері управління персоналом. Підвищення мобільності працівників, поширення дистанційних форм зайнятості та зростання вимог до прозорості обліку робочого часу формують нові виклики для компаній різного масштабу. За статистичними даними міжнародних аналітичних звітів, частка віддалених працівників у світі у 2020–2023 рр. зросла більш ніж удвічі [2], що актуалізує потребу у програмних рішеннях, здатних забезпечити об’єктивний контроль трудової діяльності. Паралельно з цим у наукових дослідженнях та практиці управління персоналом спостерігається зростання інтересу до методів інтелектуального аналізу даних, зокрема до моделей обробки природної мови, статистичних методів і технологій аналітики поведінкових патернів [1; 3; 4]. Вітчизняні публікації також підкреслюють важливість комплексної цифровізації облікових процесів та оптимізації взаємодії між роботодавцем і працівником [5].

Актуальність дослідження полягає у необхідності розроблення сучасного прикладного програмного забезпечення, здатного автоматизувати процеси аналізу, контролю та обліку робочого часу співробітників компанії, формувати достовірні аналітичні звіти, а також підтримувати прийняття управлінських рішень на основі фактологічних даних. Умови розподіленої роботи та зростання обсягів цифрової взаємодії між учасниками бізнес-процесів потребують ефективних інструментів моніторингу, які поєднують аналітику, зручність використання та адаптивність до різних організаційних структур.

Об’єкт дослідження – процес управління трудовою діяльністю співробітників у корпоративному середовищі.

Предмет дослідження – методи, моделі та програмні засоби аналізу, контролю та обліку роботи співробітників компанії.

Мета дослідження полягає у створенні програмного забезпечення, яке

забезпечує автоматизоване відстеження робочого часу, контроль виконання завдань, аналіз трудової активності та формування аналітичних звітів для підвищення ефективності управління персоналом.

Для досягнення поставленої мети у роботі визначено такі **завдання**:

- здійснити системний аналіз предметної області обліку та контролю трудової діяльності;
- сформувати вимоги до функціональних і нефункціональних характеристик інформаційної системи;
- розробити структуру бази даних та архітектуру програмного забезпечення;
- реалізувати програмний застосунок відповідно до сформованих вимог;
- провести тестування програмного забезпечення та оцінити його ефективність.

Методи дослідження. У роботі використано методи системного аналізу, методи моделювання бізнес-процесів, технології проєктування програмних систем, алгоритми обробки даних, статистичні методи аналізу та методи інтелектуального опрацювання інформації. Для побудови аналітичних моделей враховано сучасні підходи до обробки текстових і поведінкових даних, висвітлені у працях J. Devlin, M. Chang, K. Toutanova, D. Jurafsky та ін. [3; 4].

Наукова новизна полягає у розробленні комплексної програмної моделі системи контролю трудової діяльності, яка поєднує механізми обліку часу, класифікації робочої активності, аналізу поведінкових патернів та формування узагальнених аналітичних показників у єдиному інформаційному просторі. Запропонований підхід забезпечує сумісність з вимогами сучасних корпоративних систем та підтримує розширення функціональності за рахунок модульної архітектури.

Апробація результатів дослідження здійснювалася шляхом презентації напрацювань на XVI Міжнародній науково-практичній конференції «Інформаційні технології в освіті, техніці та економіці», де представлено доповідь «Програмне забезпечення аналізу, контролю та обліку роботи

співробітників компанії» [6].

Практичне значення роботи полягає у можливості впровадження розробленого програмного забезпечення в компаніях різної величини для підвищення прозорості обліку робочого часу, удосконалення процесів управління персоналом, зменшення ризиків неефективного розподілу ресурсів та забезпечення доказової бази при ухваленні управлінських рішень. Система може бути інтегрована у корпоративні інформаційні середовища та адаптована до специфіки організації.

Структура роботи. Магістерська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Перший розділ присвячено аналізу предметної області та сучасних методів HR-аналітики. Другий розділ містить інформаційне, математичне та архітектурне проєктування системи, включаючи модель даних, інформаційні потоки, архітектуру програмного забезпечення й модель безпеки. У третьому розділі наведено реалізацію програмного забезпечення, організацію серверної та клієнтської частин, інтеграцію з Telegram-ботом і результати первинного функціонального тестування. Четвертий розділ присвячено експериментальному дослідженню та оцінці ефективності системи на основі моделей машинного навчання та кластеризації показників продуктивності персоналу. Роботу завершують узагальнюючі висновки, список використаних джерел та додатки з допоміжними матеріалами, фрагментами коду й графічними ілюстраціями.

Роботу доповнюють висновки, список джерел і додатки з допоміжними матеріалами та графічними ілюстраціями. Загальний обсяг роботи становить 95 сторінку, містить 52 рисунки, 21 таблиці та 34 джерел.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ МЕТОДІВ HR-АНАЛІТИКИ

1.1 Актуальність цифрових рішень для аналізу та контролю роботи персоналу

Цифрова трансформація бізнесу та поширення гнучких форм зайнятості суттєво змінюють підходи до управління персоналом. Умови конкуренції на глобальному ринку вимагають від компаній не лише точного обліку робочого часу, а й глибокого аналізу продуктивності співробітників, прозорості процедур оплати праці та дотримання трудового законодавства. За даними статистичних звітів щодо віддаленої зайнятості, частка дистанційної роботи у світі після 2020 року стабільно зростає, що підсилює потребу в автоматизованих засобах контролю та моніторингу діяльності працівників поза офісом [2].

Традиційні системи обліку робочого часу, основані на перфокартах або простих механічних реєстраторах, сьогодні переважно втратили актуальність через низьку гнучкість, складність інтеграції з іншими інформаційними системами та нездатність підтримувати віддалені режими роботи. Біометричні системи, що ґрунтуються на відбитках пальців, розпізнаванні обличчя чи долоні, забезпечують надійну ідентифікацію, однак орієнтовані переважно на стаціонарні робочі місця й потребують спеціалізованого обладнання.

Метою системи обліку робочих годин є електронний контроль і реєстрація відвідуваності для точності нарахування заробітної плати. За допомогою системи обліку робочого часу співробітників компанія створює докладні звіти з інформацією про години роботи співробітників, відгули, понаднормову роботу, відсутність тощо. Це допомагає автоматизувати розрахунок зарплати та забезпечити дотримання організацією трудового законодавства. Система реєструє робочі години та понаднормовий час, щоб ефективно контролювати діяльність співробітників. На даний момент такі системи можна поділити (рис. 1.1):

Рис. 1.1 Системи обліку робочих годин

- із використанням перфокарти;
- із використання біометричної ідентифікації;
- цифрові системи;
- веб системи;
- мобільні системи.

Цифрові, веб- та мобільні системи обліку часу істотно розширюють можливості роботодавця: вони підтримують різні сценарії фіксації присутності, інтегруються з HRM/ERP-платформами, дозволяють формувати детальні звіти щодо годин роботи, перерв, відпусток і понаднормової зайнятості (рис. 1.2).

Рис. 1.2 Система обліку робочих годин

Сучасні дослідження вказують, що цифрові інструменти управління персоналом перетворюються з допоміжних сервісів на ключовий елемент системи корпоративного управління, оскільки забезпечують доступ до великомасштабних масивів HR-даних та підтримують аналітичні рішення на їх основі [5]. Такі системи дають змогу:

– автоматизувати нарахування заробітної плати з урахуванням реальних відпрацьованих годин, понаднормової роботи та різних типів відгулів;

– гарантувати прозорість взаєморозрахунків між роботодавцем і працівником, зменшуючи кількість конфліктних ситуацій;

– підвищити дисципліну та відповідальність персоналу завдяки фіксації часу початку й завершення робочого дня, а також тривалості перерв;

– формувати аналітичні звіти для оцінювання ефективності роботи окремих співробітників, підрозділів і компанії загалом.

Однак простий облік часу вже не задовольняє потреби організацій, які працюють у висококонкурентному середовищі та активно використовують дистанційні моделі співпраці, зокрема фриланс-платформи на кшталт Upwork [2]. Значна частина взаємодії між роботодавцем і працівником, а також між працівником та інформаційною інфраструктурою компанії реалізується через цифрові канали, що породжує великі обсяги структурованих і неструктурованих даних: журнали активності, запити користувачів, комунікацію в месенджерах, текстові звіти тощо. Дослідження користувацьких запитів у веб-середовищі показують, що ці дані мають складну структуру, містять неформальну лексику, доменні терміни й можуть слугувати цінним джерелом знань про поведінку та інформаційні потреби користувачів [1].

Розвиток методів обробки природної мови та глибинного навчання створює передумови для переходу від суто реєстраційних систем до інтелектуальних платформ, здатних виявляти приховані закономірності у поведінці персоналу, оцінювати ризики вигорання чи зниження продуктивності, прогнозувати виконання планових показників. Сучасні мовні

моделі, зокрема BERT, відкривають можливості змістовного аналізу текстових даних – запитів, коментарів, внутрішньої переписки – та формування семантичних ознак для подальшого моделювання [3]. Комплексні підходи до обробки мовних та поведінкових даних детально описані в сучасних курсах з обробки природної мови, де підкреслюється важливість використання статистичних і нейромережевих методів для аналізу людської комунікації [4].

Водночас вітчизняні дослідження у сфері цифровізації управління персоналом акцентують на потребі адаптації міжнародних практик HR-аналітики до українського правового поля та особливостей локальних ринків праці [5]. Серед ключових викликів – забезпечення відповідності вимогам захисту персональних даних, інтеграція з наявними HRM-системами, підтримка гнучких режимів роботи та дистанційної співпраці.

Попередні роботи автора, присвячені розробленню програмного забезпечення аналізу, контролю та обліку роботи співробітників компанії, продемонстрували можливість побудови веб-орієнтованих систем, що поєднують тайм-трекер, візуалізацію показників і базові аналітичні звіти [6]. Проте такі рішення потребують подальшого розвитку в напрямі використання методів машинного навчання для прогнозування продуктивності, виявлення аномальної поведінки, формування персоналізованих рекомендацій для HR-аналітиків і керівників підрозділів.

Отже, актуальність цифрових рішень для аналізу та контролю роботи персоналу визначається:

– переходом від традиційних систем обліку часу до інтегрованих платформ, орієнтованих на аналітику й підтримку управлінських рішень;

– масовим поширенням віддалених і гібридних форм зайнятості, що потребують надійних засобів моніторингу та оцінювання ефективності працівників [2; 5];

– появою технологічних можливостей для використання сучасних методів обробки природної мови й глибинного навчання при роботі з HR-даними [3; 4];

– необхідністю створення спеціалізованих рішень, адаптованих до українського контексту та орієнтованих на поєднання обліку, контролю та інтелектуального аналізу діяльності співробітників [5; 6].

У цьому контексті розроблення програмного забезпечення аналізу, контролю та обліку роботи співробітників компанії, яке об’єднує цифрові механізми фіксації робочого часу, засоби аналітики та інструменти прогнозування продуктивності, є важливою науковою та практично значущою задачею.

1.2 Огляд існуючих підходів та рішень

Сучасні системи цифрового контролю та аналітики діяльності персоналу формують окремий сегмент ринку HR-технологій, що швидко розвивається внаслідок поширення дистанційної зайнятості та гібридних моделей роботи. Зростання попиту на інструменти відстеження часу, автоматизованого збору доказів виконання завдань і формування достовірної статистики зумовлене як економічними факторами, так і змінами в організації трудових процесів після пандемії, коли частка працівників, що виконують роботу поза офісом, суттєво збільшилась [2]. У цих умовах цифрові системи обліку робочого часу стали ключовим елементом забезпечення прозорості, підзвітності та справедливого нарахування заробітної плати, а також інструментом управління ризиками у розподілених командах [5].

Наявні рішення можна класифікувати на п’ять категорій: системи на основі перфокарт, біометричні модулі, цифрові системи з магнітними картками, веб-системи та мобільні платформи з GPS-контролем. Перші три групи здебільшого використовуються для фіксації входу та виходу з робочого місця, тоді як веб- і мобільні системи охоплюють повний цикл: облік часу, моніторинг активності, фіксацію доказів роботи (скриншоти, лог активностей), формування звітів і аналітики. Класифікацію таких систем наведено на рис. 1.3, який відображає еволюцію підходів від механічних технологій до веб-орієнтованих розподілених рішень.

Рис. 1.3 Класифікація систем обліку робочого часу

Коли працівник перебуває в період максимальної продуктивності, єдине, що потрібно, - це не забути відслідковувати свій робочий час. Ось тут-то і приходить на допомогу додаток для відстеження часу. Замість того, щоб переривати творчий процес для відстеження годин або покладатися на пам'ять, додаток для відстеження часу автоматично виконає всю роботу за вас [5].

Системи перфокарт, що були домінуючими у попередні десятиліття, сьогодні розглядаються як застарілі, оскільки не дозволяють працювати з віддаленими або мобільними командами [2]. Біометричні системи забезпечують високий рівень точності, проте вимагають фізичної присутності працівника та відповідного обладнання. Цифрові системи з магнітними картками характеризуються простотою, але не містять інструментів для детального аналізу активності. Найбільш поширеними стали веб- та мобільні рішення, що дозволяють фіксувати час роботи через браузер або мобільний застосунок, інтегруються з GPS-модулями та повідомляють керівництво про відхилення чи порушення режиму роботи [3]. Вони забезпечують автоматичну фіксацію перерв, понаднормових робіт та інших параметрів, що підвищує об’єктивність обліку та мінімізує людський фактор.

Поширення віддаленої роботи привело до появи систем, які поєднують відстеження робочого часу із розширеним моніторингом: фіксацією активних вікон, відвідуваних веб-сторінок, динаміки продуктивності, а подекуди – поведінкових патернів із використанням алгоритмів штучного інтелекту. Такі системи, як Teramind та ActivTrak, орієнтовані на поглиблений аналіз активності та застосовують DLP-модулі, патернову аналітику й прогнозування

ризикових дій співробітників [4]. У корпоративному секторі це дозволяє не лише контролювати дисципліну, а й своєчасно виявляти аномальні сценарії поведінки, що можуть свідчити про кіберзагрози чи порушення політик доступу. У свою чергу хмарні інструменти, як Hubstaff чи Time Doctor, пропонують мобільність, простоту розгортання та інтеграції з популярними платформами менеджменту завдань, проте мають нижчий рівень контролю порівняно з локальними DLP-рішеннями.

UpWork – один з найпопулярніших сервісів подібного формату. Переваги: для відстеження часу та оплати клієнтів Upwork пропонує надійний додаток, який можна використовувати для відстеження часуроботи, виконаної фрілансерами. Додаток Upwork робить періодичні знімки екрана та виставляє рахунки клієнту. Погодинна ставка вже визначена між клієнтами та фрілансерами. Фрілансери також можуть увімкнути зображення веб-камери зі скріншотами, що в деякихвипадкахможе бути вимогою клієнта контролювати фрілансерів, щоб переконатися, що вони працюють протягом зареєстрованого часу [6].

Хоча Upwork чудово підходить для фрілансерів та людей, які хочуть найняти працівників віддалено, деякі скаржиться на суму, яку стягує Upwork (рис. 1.4). Для початківців Upwork бере плату, яка зменшується лише тоді, коли ви відпрацювали певну кількість годин. Проблема в тому, що для досягнення цього рівня у вас може знадобитися кілька місяців, а то й рік чи більше. Через це багато клієнтів обрали альтернативи [6].

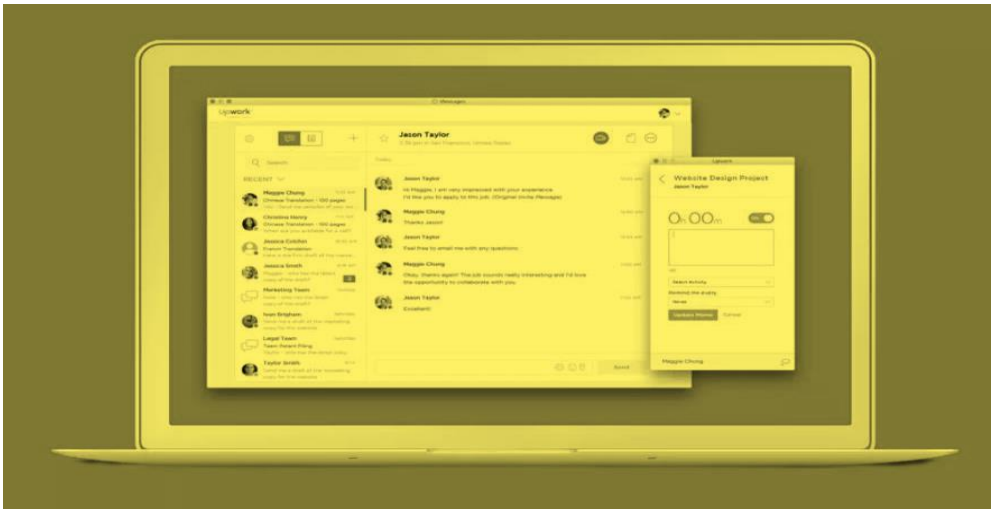

Рис. 1.4 Візуалізація додатку UpWork

Tick тайм трекер підходить для команд, яких лякає цінова політика інших додатків – «за кожного користувача» (рис. 1.5). Tick бере плату не «за користувача», а за кожен конкретний проєкт, кількість користувачів не обмежена. Іншими словами, можна значно заощадити, в залежності від проєкту, і особливо, якщо команда співробітників досить чисельна. Недоліки цього трекера в том, що інтерфейс не настільки простий, як у більшості додатків. Завдання доводиться створювати в інтерфейсі кожного окремого проєкту. Тому на створення нових задач витрачається додатковий час [4]

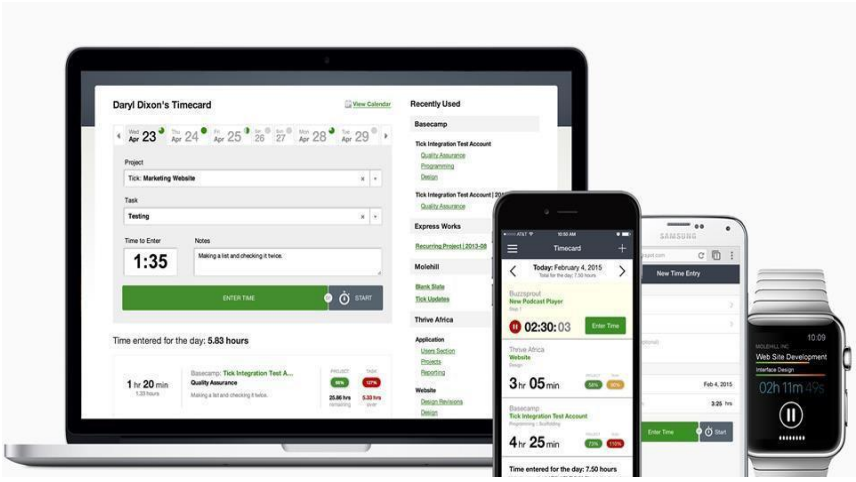

Рис. 1.5 Візуалізація додатку Tick

Порівняння найбільш значущих систем наведено в таблиці 1.1, де відображено ключові відмінності щодо архітектури, функцій, рівня аналітики та можливостей моніторингу. Аналіз показує, що жодне із наявних рішень не забезпечує водночас гнучкої інтеграції з месенджерами, автоматизованої відправки скриншотів без встановлення додаткового ПЗ на пристрої замовника, розширеної статистики, прозорого білінгу й мінімальних вимог до інфраструктури користувача. Це відкриває можливості для створення веб-системи нового типу, яка пропонує відправлення скриншотів через Telegram-бот, не вимагає встановлення програмного забезпечення на комп’ютері замовника та забезпечує високу швидкість комунікації, що особливо важливо для динамічних проєктів і фриланс-платформ.

Таблиця 1.1

Порівняльна характеристика сучасних систем контролю робочого часу

Система	Основні функції	Рівень аналітики	Скриншоти	GPS	Архітектура	Сильні сторони	Обмеження
Hubstaff	Облік часу, GPS	базовий	так	так	хмарна	простота, мобільність	мінімальна безпека
Teramind	моніторинг + DLP	високий	так	–	локальна/ хмара	AI-аналітика, безпека	висока вартість
Time Doctor	скринкасти, звіти	середній	так	–	хмарна	інтеграції (Jira, Asana)	обмежені можливості контролю
Kickidler	відеомоніторинг	високий	так	–	локальна	глибокий контроль	потребує сервера
ActivTrak	поведінкові патерни	високий	–	–	хмарна	автоматичні інсайти	низька кастомізація
Controlio	моніторинг + безпека	середній	–	–	хмарна	баланс контроль/ приватність	немає офлайн-збору

У літературі та конференційних працях відзначається, що інтеграція HR-аналітики з веб-системами контролю часу стає одним із ключових напрямів розвитку цифрового менеджменту персоналу [5; 6]. Використання методів машинного навчання, зокрема моделей BERT для обробки текстових описів активності, дозволяє системам переходити від пасивного моніторингу до інтерпретації даних, виявлення відхилень і формування рекомендацій [3; 4]. У роботах також підкреслюється необхідність створення рішень, що враховують

конфіденційність інформації працівників, забезпечують захист персональних даних і дозволяють налаштовувати політики доступу залежно від ролі актора.

На основі проведеного аналізу сформовано порівняльну матрицю (табл. 1.2), де розроблюване у межах магістерської роботи рішення зіставлено з найбільш поширеними системами.

Таблиця 1.2

Порівняння розробленої системи з існуючими сервісами

Функція	Toggl	Tick	Kickidler	UpWork	Monitask	MyTime
Облік часу	+	+	+	+	+	+
Скриншоти	–	–	+	+	+	+ через Telegram-бот
Веб-інтерфейс	+	+	+	–	+	+
Мобільність (Telegram)	–	–	–	–	–	+
Автоматичний білінг	+	+	–	+	–	+
Відеомоніторинг	–	–	+	–	–	–
AI-аналітика	–	–	+	–	–	можливість інтеграції

Порівняння підтверджує, що запропонований продукт займає проміжну позицію між простими SaaS-трекинговими рішеннями та комплексними корпоративними платформами моніторингу: він успадковує від перших мобільність і простоту використання, а від других – можливості розширення функціоналу за рахунок аналітики та автоматизації. Головними перевагами розробленої системи є інтеграція з Telegram-ботом, мінімальні вимоги до інфраструктури, можливість оперативного отримання доказів виконання роботи, швидка комунікація та потенціал до впровадження ML-модулів у майбутньому.

Таким чином, огляд існуючих підходів показує, що ринок цифрових систем контролю робочого часу є високодинамічним, проте має структурні прогалини, які можуть бути ефективно заповнені розроблюваним у роботі веб-орієнтованим рішенням. Його концепція відповідає сучасним тенденціям діджиталізації управління персоналом, потребам віддалених команд і вимогам до прозорого контролю виконання завдань, що свідчить про доцільність і актуальність обраної розробки.

1.3 Аналіз проблематики управління персоналом

Управління персоналом у сучасних організаціях характеризується високим рівнем складності, що зумовлено цифровою трансформацією бізнес-процесів, розширенням віддалених форм зайнятості, динамічністю ринку праці та підвищенням вимог до прозорості виконання службових обов’язків. Поширення гнучких та розподілених моделей роботи, що прискорилося після 2020 року, призвело до суттєвого зростання кількості співробітників, які виконують завдання поза офісом, що одночасно створило нові виклики для роботодавців у сферах контролю, аналітики та документування робочих подій [2].

Однією з ключових проблем є відсутність повної інформаційної визначеності щодо фактичної діяльності співробітників. Традиційні інструменти (ручний облік, перфокартні чи карткові системи) не забезпечують можливості фіксації активності віддалених або мобільних працівників, а також не дають змоги побачити динаміку робочих патернів. За умов гнучких графіків підвищується ризик викривлення даних щодо реального часу виконання завдань, тривалості перерв та рівня залученості, що ускладнює об’єктивну оцінку продуктивності та формування коректної звітності. За даними професійних оглядів ринку HRM-рішень, до 30 % компаній стикаються з неточністю або неповнотою даних, якщо облік виконання робіт здійснюється вручну або в напівавтоматичному режимі [7]. Узагальнення проблем представлено в таблиці 1.3.

Таблиця 1.3

Ключові проблеми управління персоналом та їх наслідки

Група проблем	Сутність	Можливі наслідки
Недостатність даних	Неповні, несвоєчасні або ручні дані про діяльність працівників	Помилки в оцінці продуктивності, неточна зарплатна звітність
Відсутність оперативного контролю	Нездатність отримувати дані про роботу в реальному часі	Втрати часу, ризики зриву проєктів
Фрагментація інструментів	Використання різних систем без інтеграції	Дублювання даних, помилки, збільшення витрат
Порушення приватності	Надмірний контроль або відсутність політик доступу	Конфлікти, юридичні ризики (GDPR/PDPA)
Недостатність аналітики	Відсутні ML-прогнози, непрозора динаміка	Неможливість стратегічного планування

Другим суттєвим блоком проблем є брак механізмів своєчасного контролю. У розподілених командах роботодавець часто отримує відомості про хід виконання роботи із затримкою, що позбавляє можливості оперативного реагування на відхилення, порушення графіка або зниження продуктивності. Системи, що не фіксують доказові дані (наприклад, скриншоти, часові мітки, повідомлення в месенджері), не дозволяють підтвердити факт виконання завдання, що створює ризики як для замовника, так і для виконавця.

Не менш важливим є питання збалансування контролю і конфіденційності. Частина корпоративних рішень впроваджує надмірний моніторинг (запис екрана, кейлогери, DLP-рішення), який може порушувати принципи захисту приватних даних, встановлені GDPR, PDPA та національним законодавством. Водночас надто спрощені інструменти (веб-трекери, ручні звіти) не забезпечують достатнього рівня доказовості та точності, що посилює управлінські ризики [5]. На рисунку 1.6 показано узагальнену структуру основних проблем HR-менеджменту. Проблематика також поглиблюється через відсутність уніфікованої аналітики

продуктивності.

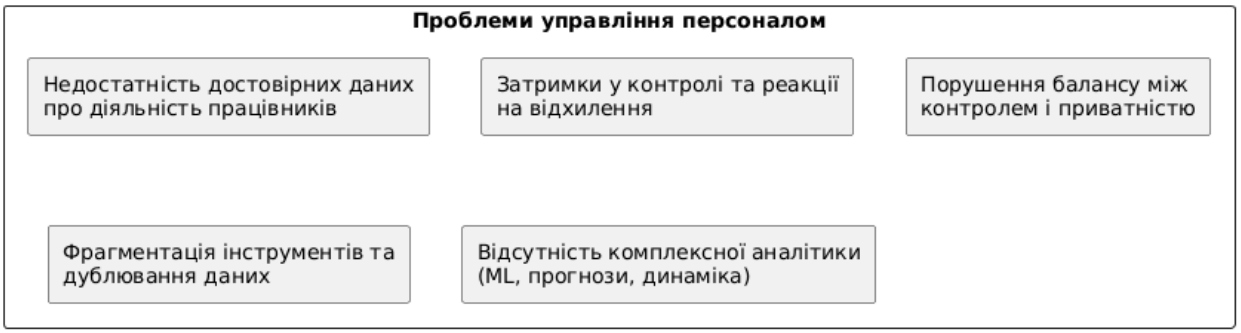

Рис. 1.6 Основні проблеми управління персоналом у розподілених командах

Більшість рішень обмежуються базовими показниками – кількістю відпрацьованих годин або відвідуваністю, тоді як роботодавці потребують комплексної оцінки темпу, стабільності активності, відхилень від норми, прогнозів можливих ризиків та інсайтів на основі історичних даних. У наукових публікаціях та сучасних методологіях цифровізації HR-систем відзначається тенденція інтегрувати алгоритми машинного навчання для моделювання поведінки працівників та підвищення прозорості управлінських процесів [3; 4].

Важливою проблемою залишається фрагментація інструментів HR-екосистеми, коли компанії змушені застосовувати різні сервіси для обліку часу, постановки завдань, чат-комунікацій, контролю виконання, зарплатної звітності й аналітики. Відсутність інтегрованого середовища підвищує ризики помилок, ускладнює аудит та збільшує операційні витрати. Порівняння основних підходів до контролю персоналу наведено в таблиці 1.4.

Таблиця 1.4

Порівняння підходів до контролю персоналу

Тип підходу	Переваги	Обмеження
Ручний контроль	Прості інструменти, мінімальні витрати	Схильність до помилок, суб’єктивність, неефективність у віддаленій роботі
Системи відстеження часу	Автоматизація, базові метрики	Не фіксують контекст завдань, відсутність ML-аналітики
DLP/моніторинг активності	Високий рівень деталізації	Порушення приватності, складність впровадження

Інтегровані HR-платформи	Комплексний підхід, аналітика	Висока вартість, громіздкість конфігурації
ML-орієнтовані системи	Прогнозування, автоматичні інсайти	Потребують коректних даних і налаштувань

На рисунку 1.7 представлено концептуальну модель взаємодії джерел даних та аналітичних модулів, характерну для сучасних HR-аналітичних рішень.

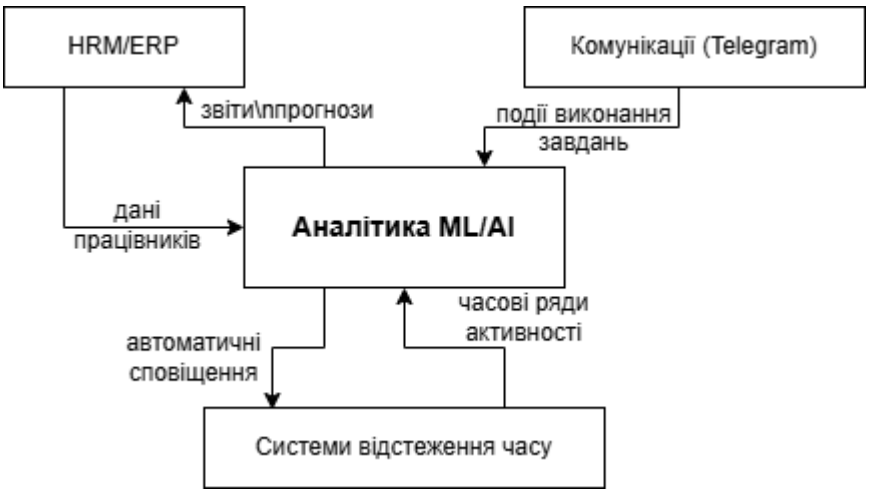

Рис. 1.7 Необхідність інтегрованої системи підтримки HR-аналітики

Узагальнюючи наведене, можна зазначити, що механічний облік робочого часу не забезпечує розуміння якості виконання завдань, а надмірні засоби моніторингу створюють юридичні та етичні ризики. Таким чином, ключовою стає концепція оптимального контролю, що ґрунтується на принципах мінімальної достатності даних, релевантності індикаторів, автоматизації збору доказів та оперативного інформування менеджера [8; 9].

Аналіз виявлених проблем дозволяє сформулювати вимоги до нової генерації систем управління персоналом, яка має забезпечувати мобільність, автоматизований збір доказів діяльності, інтеграцію з месенджерами, можливість масштабування та підтримку методів машинного навчання. Саме на таких засадах побудовано рішення, розроблене у межах магістерської роботи, що спрямоване на усунення ключових недоліків наявних інструментів та підвищення прозорості виконання завдань у віддаленому режимі.

1.4 Постановка задачі та вимоги до системи

З огляду на виявлені проблеми організації обліку робочого часу та контролю діяльності віддалених виконавців сформульовано задачу розроблення інформаційної системи, що забезпечує прозорий облік робочого часу, оперативний моніторинг виконання завдань і формування звітності для замовника. Особливістю запропонованого рішення є інтеграція веб-інтерфейсу виконавця з телеграм-ботом замовника, що дає змогу автоматизувати надсилання знімків екрана, фіксувати фактичний час роботи над проєктом та отримувати обґрунтовані дані для розрахунку оплати праці, продовжуючи ідеї, апробовані автором у попередніх дослідженнях [6].

Метою розроблення системи є автоматизація процесів відстеження використання робочого часу виконавцями, формування достовірної аналітики їхньої діяльності та підтримка розрахунків заробітної плати й виставлення рахунків клієнтам на основі об’єктивних даних. Система повинна забезпечувати не лише фіксацію часових витрат, а й можливість аналізу структури робочого дня, виявлення трудомістких завдань та оперативне інформування замовника про хід виконання проєкту.

Для досягнення поставленої мети необхідно розв’язати такі основні завдання:

– спроєктувати концепцію та інтерфейс веб-застосунку тайм-трекера з урахуванням сценаріїв роботи виконавця та замовника;

– розробити архітектуру клієнт-серверного рішення з виділенням веб-інтерфейсу, серверної логіки, підсистеми зберігання даних та модуля інтеграції з телеграм-ботом;

– реалізувати програмні модулі реєстрації та авторизації користувачів, створення і супроводу проєктів, запуску та зупинки трекінгу часу, формування та надсилання скріншотів, побудови статистичної звітності;

– виконати тестування розробленого програмного забезпечення з метою

виявлення логічних та програмних помилок і перевірки коректності обробки типових сценаріїв використання;

– оцінити отримані результати з погляду зручності використання, повноти функцій контролю та можливостей подальшого масштабування системи.

Постановка задачі передбачає визначення ролей користувачів та їхніх прав у системі. Виділяються два основні актори: виконавець (розробник, фрілансер), який безпосередньо працює із веб-інтерфейсом тайм-трекера, та замовник, який отримує аналітичну інформацію та знімки екрана через телеграм-бот. Програмний продукт реалізується у вигляді веб-застосунку, розміщеного на сервері; доступ виконавця здійснюється через браузер, тоді як замовник взаємодіє з системою переважно в середовищі месенджера Telegram, що знижує технічні бар’єри для його участі у процесі контролю.

Структуру варіантів використання системи подано на рисунку 1.8. У цій діаграмі відображено основні сценарії роботи виконавця (реєстрація, авторизація, створення та вибір проєкту, керування трекером, перегляд статистики, коментування включення трекера) та замовника (приєднання до телеграм-бота, перегляд знімків екрана, отримання форми оплати).

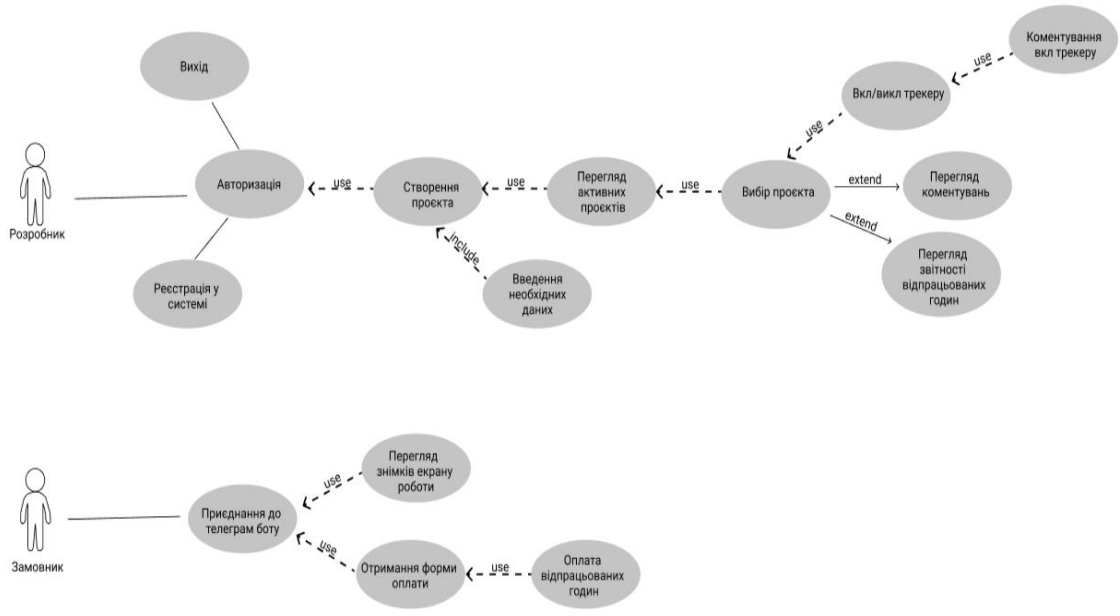

Рис. 1.8 Модель варіантів використання системи обліку робочого часу

На основі діаграми варіантів використання сформульовано функціональні вимоги до системи, які згруповано в табл. 1.5. У таблиці наведено відповідність між акторами, сценаріями використання та вимогами до інтерфейсу й поведінки системи, а також визначено пріоритет реалізації кожного варіанта.

Таблиця 1.5

Вимоги до варіантів використання

Актор	Варіант використання	Опис вимог варіанта використання	Пріоритет
1	2	3	4
Виконавець	Реєстрація	RQ 001 Користувач отримує екранну форму, що містить поля: ім’я користувача, пошту, пароль. RQ 001.1 Користувач отримує пусту екранну форму для її заповнення. RQ 001.2 Користувач заповнює усі необхідні поля валідними даними.	Високий
Виконавець	Авторизація	RQ 002 Користувач отримує екранну форму, що містить поля: пошту, пароль. RQ 002.1 Користувач отримує пусту екранну форму для її заповнення. RQ 002.2 Користувач заповнює усі необхідні поля валідними даними.	Високий
Виконавець	Створення проєкту	RQ 003 Користувач отримує екранну форму створення проєкта. RQ 003.1 Користувач отримує пусту екранну форму проєкта для заповнення необхідних полів. RQ 003.2 Користувач заповнює усі необхідні поля.	Високий
Виконавець	Вибір проєкту	RQ 004 Користувач отримує екранну форму, що містить список створених проєктів. RQ 004.1 Користувач вибирає потрібний проєкт із списку.	Високий
Виконавець	Включення трекеру	RQ 005 Користувач отримує екранну форму, що містить кнопку включення трекеру. RQ 005.1 Користувач включає трекер.	Високий
Виконавець	Виключення трекеру	RQ 006 Користувач отримує екранну форму, що містить кнопку виключення трекеру.	Високий

Актор	Варіант використання	Опис вимог варіанта використання	Пріоритет
1	2	3	4
		RQ 006.1 Користувач виключає трекер.	

Продовження табл. 1.5

1	2	3	4
Виконавець	Перегляд статистики відпрацьованих годин	RQ 007 Користувач отримує екранну форму, що містить звіт за потрібний період (день, тиждень, весь період).	Високий
Виконавець	Коментування включення трекеру	RQ 008 Користувач отримує екранну форму включення трекеру, що містить поле для коментування. RQ 008.1 Користувач заповнює поле для коментування включення трекеру.	–
Замовник	Перегляд знімків екрану виконавця	RQ 009 Користувач приєднується до телеграм-бота. RQ 009.1 Користувач отримує повідомлення в телеграм-боті, що містять знімки екрана виконавця в період включення трекеру.	Високий
Замовник	Отримання форми оплати	RQ 010 Користувач отримує в телеграм-боті посилання на форму оплати виконаного проєкта. RQ 010.1 Користувач проводить оплату.	Високий

Сукупність наведених вимог визначає функціональні межі системи та слугує основою для подальшого проєктування архітектури, моделювання бізнес-процесів і розробки програмних модулів.

Узагальнюючи наведене, постановка задачі та сформовані вимоги окреслюють концепцію цілісної системи обліку робочого часу, здатної інтегрувати процеси моніторингу, аналітики та взаємодії між виконавцем і замовником. Визначені функціональні ролі, сценарії використання та структуровані вимоги створюють фундамент для подальшого архітектурного проєктування й програмної реалізації. У межах цього підходу програмне рішення має забезпечити не тільки технічну коректність роботи, а й прозорість, доступність та можливість масштабування в умовах сучасних моделей дистанційної та проєктної зайнятості.

1.5 Висновки до 1 розділу

Проведений аналіз сучасного стану цифрових рішень у сфері управління персоналом засвідчив, що системи контролю робочого часу переходять від фіксації базових подій до комплексних аналітичних платформ, орієнтованих на підтримку управлінських рішень. В умовах зростання частки віддаленої та гнучкої зайнятості ключовими стають оперативність отримання даних, точність фіксації активності, відповідність законодавчим вимогам та збереження конфіденційності. Виявлено низку проблем – фрагментацію інструментів, недостатність достовірних даних, неможливість оперативного контролю та обмеженість доказової бази – що зумовлює необхідність створення інтегрованих систем нового покоління. Огляд рішень, представлених на ринку, показав суттєві розбіжності між потребами користувачів і наявними можливостями, особливо у частині мобільності, автоматизованого моніторингу та адаптивної аналітики.

Порівняння функціоналу існуючих тайм-трекерів підтвердило, що більшість систем фокусуються на окремих аспектах контролю і не здатні забезпечити повний цикл підтримки взаємодії між виконавцем і замовником: оперативних сповіщень, знімків екрану, формування звітності та виставлення рахунків. Це створює нішу для розробки рішень, орієнтованих на комплексну автоматизацію та простоту використання. Таким чином, результати першого розділу формують обґрунтовану основу для розробки програмного забезпечення, здатного подолати виявлені недоліки, забезпечити прозорість робочих процесів і підвищити ефективність управління персоналом у сучасних умовах цифрової економіки.

2. ІНФОРМАЦІЙНЕ, МАТЕМАТИЧНЕ Й АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Модель даних та інформаційні потоки системи

Інформаційна система контролю, аналізу та обліку робочого часу функціонує як веборієнтований тайм-трекер, орієнтований на підтримку віддалених моделей зайнятості та забезпечення прозорої взаємодії між виконавцем і замовником. Модель даних системи ґрунтується на строгій логічній організації інформаційних потоків, що проходять між користувачами, серверною частиною, Telegram-ботом та механізмами формування звітів. В основі концептуального подання лежать сутності, що описують користувачів, проєкти та події роботи трекера, доповнені атрибутами коментарів, статусів і часових параметрів, що дозволяє забезпечувати відтворюваність, узгодженість і доказовість даних [10; 11].

Початковий рівень інформаційних потоків формується під час взаємодії користувача з вебінтерфейсом. На головному екрані застосунку (рис. 2.1) доступні елементи навігації до режимів реєстрації, авторизації, створення проєктів і перегляду активних завдань.

Рис. 2.1 Приклад екранної форми головної сторінки веб-застосунку

Введення вхідних даних відбувається через діалогові форми, для яких реалізовано багаторівневі механізми контролю коректності: синтаксичні перевірки (формат електронної адреси, узгодженість паролів, заповнення обов’язкових полів) та семантичні обмеження (перевірка унікальності облікового запису, коректність асоціації користувача з проєктом) [12].

Форма реєстрації (рис. 2.2) забезпечує створення сутності User, якій призначається унікальний ідентифікатор і набір атрибутів, необхідних для подальшої автентифікації. Процес авторизації (рис. 2.3) формує сесію користувача та запускає перший інформаційний потік – ініціалізацію робочого простору відповідно до ролі (виконавець або замовник).

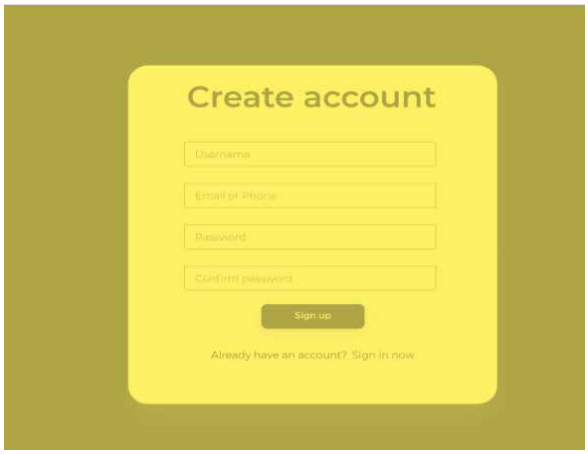

Рис. 2.2 Приклад екранної форми реєстрації

Рис. 2.3 Приклад екранної форми авторизації

У другому інформаційному потоці формується структура проєктів, що визначають контекст виконання робіт. У вікні створення проєкту (рис. 2.4) користувач вводить назву завдання, ім’я замовника та короткий коментар до включення трекера.

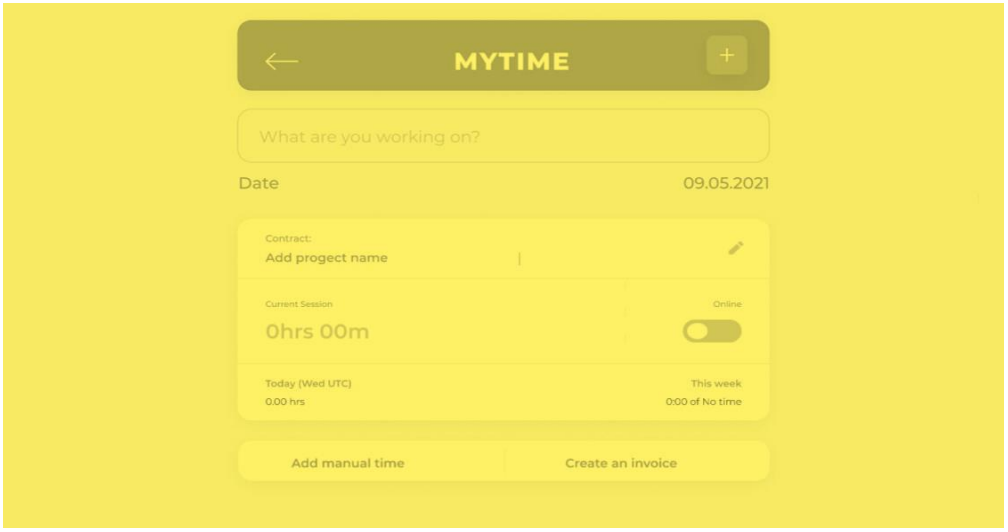

Рис. 2.4 Приклад екранної форми авторизації

На основі цих даних формується сутність Task, яка містить зв’язки з обліковим записом виконавця та сутністю замовника. На цьому етапі задаються первинні параметри організації робіт: часові межі, очікувані результати, інформаційний контекст взаємодії. Для цього рівня характерна структурована передача даних до серверної частини, де забезпечується їх валідація та фіксація.

Подальший перебіг інформаційних потоків пов’язаний із реєстрацією фактичної активності користувача. Під час запуску або зупинки трекера система створює відповідні події зміни статусу, які описуються атрибутами режиму роботи, часу та дати активації або деактивації. У цей же момент активується механізм формування знімків екрана, що генерує візуальні метадані й передає їх до Telegram-бота замовника. Такий підхід поєднує класичну хронологію подій із візуальною доказовістю робочого процесу, що відповідає сучасним вимогам до забезпечення прозорості віддаленої зайнятості [11; 13].

Таблиця 2.1 узагальнює логічні етапи обробки даних при автентифікації користувачів і демонструє структуру цього потоку.

Таблиця 2.1

Структура потоку автентифікації

Етап обробки	Вхідні дані	Перевірки	Вихід
Реєстрація	username, email, password	формальні, унікальність	створення User
Авторизація	email, password	відповідність БД	ініціалізація сесії
Розподіл ролей	ID користувача	тип користувача	інтерфейс виконавця/замовника

Після створення проєктів формується другий структурний блок, який відображено у таблиці 2.2.

Таблиця 2.2

Атрибутивна модель проєктів

Атрибут	Опис

ProjectName	назва проєкту
CustomerName	ім’я замовника
StartDate	дата початку роботи
StatusHistory	зміни станів трекера
Comment	коментар до запуску трекера

Третій блок інформаційних потоків охоплює процеси відстеження активності, передачі знімків, збору часових показників та формування статистичної звітності. На рис. 2.5 наведено приклад перегляду конкретного проєкту: система відображає тривалість робочих сесій, історію включень і вимкнень трекера, а також ключові коментарі виконавця.

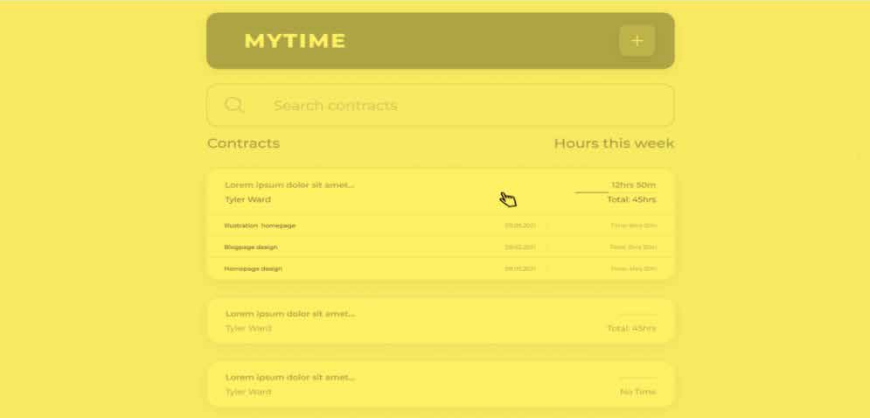

Рис. 2.5 Приклад екранної форми при натисканні на конкретний проєкт

На рис. 2.6 подано агрегований перелік активних проєктів, що відображає загальну завантаженість користувача та дозволяє здійснювати швидкий перехід до детальної інформації.

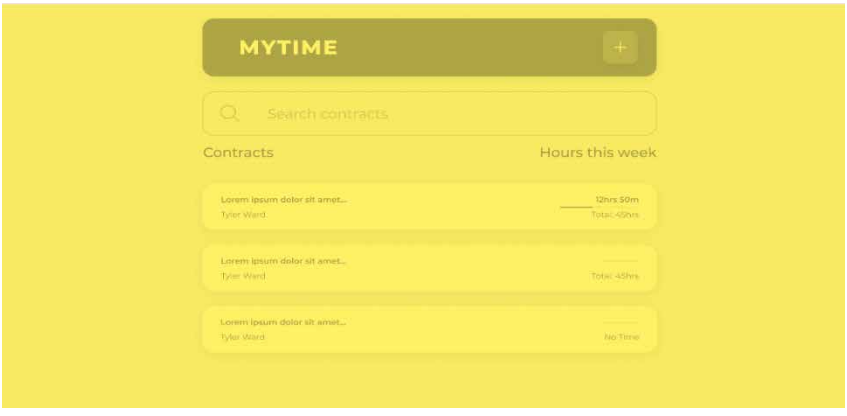

Рис. 2.6 Приклад екранної форми списку активних проєктів

Для формалізації логічної взаємодії між компонентами було побудовано узагальнену схему інформаційних потоків системи (рис. 2.7), яка відображає рух даних від користувача до серверних модулів, Telegram-бота та блоків формування звітності.

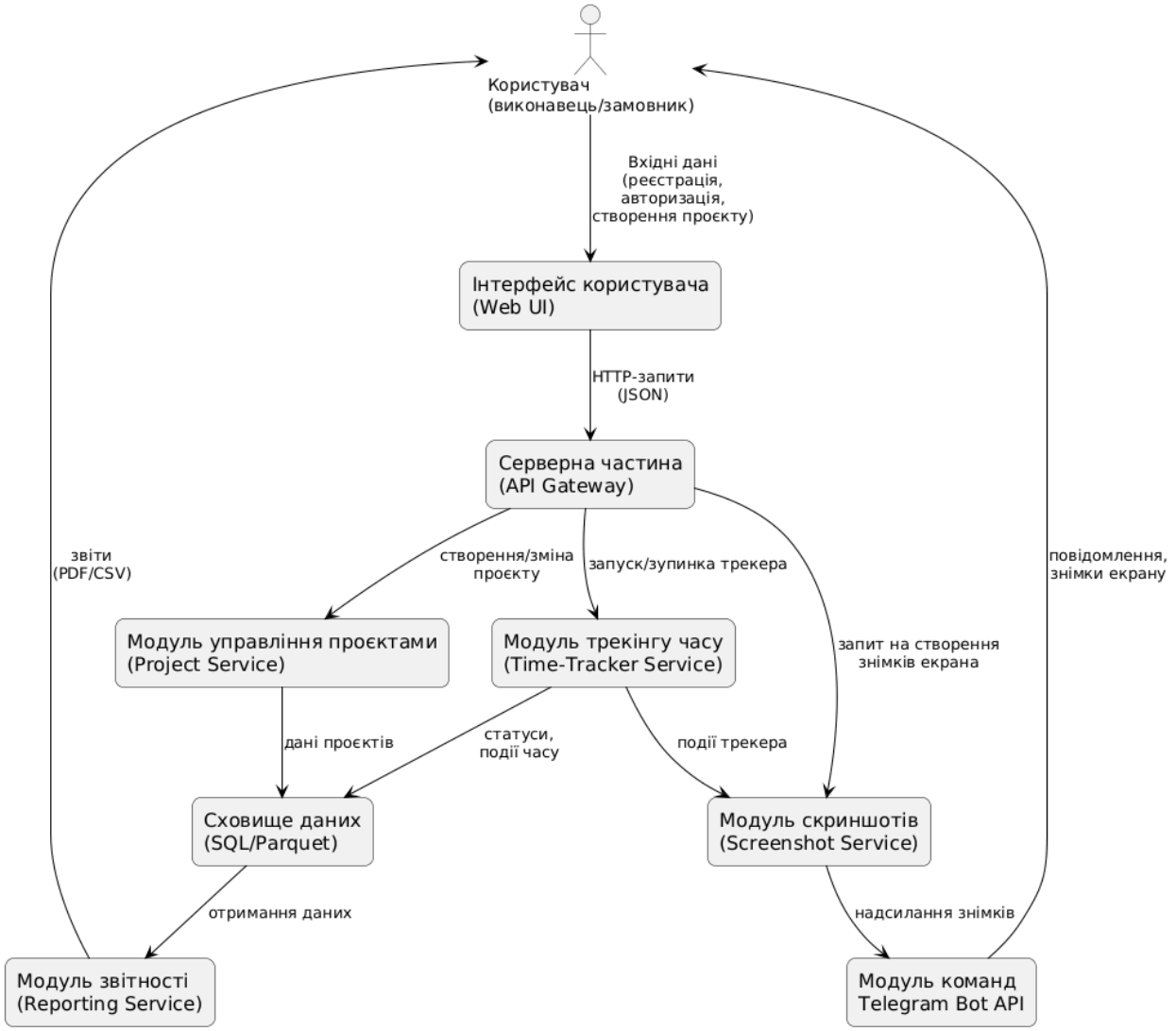

Рис. 2.7 Узагальнена схема інформаційних потоків системи

Схема демонструє, що інформаційні потоки поділяються на транзакційні (реєстрація, авторизація, створення проєктів), операційні (фіксація робочих подій і знімків екрана) та аналітичні (агрегація, формування звітів, експорт даних). Такий багаторівневий підхід забезпечує наскрізну обробку даних, що є необхідною умовою автоматизації HR-процесів і стійкого контролю робочої активності в розподілених командах.

У таблиці 2.3 узагальнено мапу інформаційних потоків, що відображає

взаємозв’язок між користувачем, серверними модулями та вихідними інформаційними продуктами.

Таблиця 2.3

Мапа інформаційних потоків системи

Потік	Джерело	Модуль обробки	Вихідні дані	Призначення
П-1	Веб-форма	Auth-API	Session Token	доступ
П-2	Створення проєкту	Project-API	Task	планування
П-3	Трекер	Time-Tracker Service	статуси, події	облік часу
П-4	Скріншоти	Screenshot-Service	зображення, метадані	Telegram бот
П-5	Агрегація часу	Reporting-API	PDF/CSV	звітність
П-6	Виконані години	Invoice-Service	Invoice	оплата

Загалом модель даних та інформаційні потоки системи забезпечують формування послідовної, узгодженої та масштабованої архітектури, що гарантує достовірність результатів, оперативність реакції на робочі події та прозорість комунікації між замовником і виконавцем. Побудована логічна схема створює підґрунтя для подальшого математичного моделювання, оптимізації бізнес-процесів, застосування методів прогнозної аналітики та переходу до детального архітектурного проєктування системи [12–14].

2.2 Обґрунтування вибору технологій

Проєктована система аналізу, контролю та обліку робочого часу реалізується як веборієнтований застосунок з інтеграцією у месенджер Telegram, тому вибір технологічного стеку має забезпечувати кросплатформеність, швидку розробку, простоту супроводу, можливість масштабування та подальшого розширення функціональності аналітичними модулями. Світові тенденції свідчать, що для таких рішень переважно застосовують мови загального призначення з розвиненою екосистемою вебфреймворків, бібліотек інтеграції та інструментів автоматизації. На основі

статистики GitHub та опитувань розробників лідируючі позиції за популярністю стабільно займають JavaScript, Python і Java, тоді як C#, PHP та інші мови поступаються часткою використання (рис. 2.8). Це підтверджує доцільність опори на відкриті технології із широкою спільнотою та якісною документацією [15].

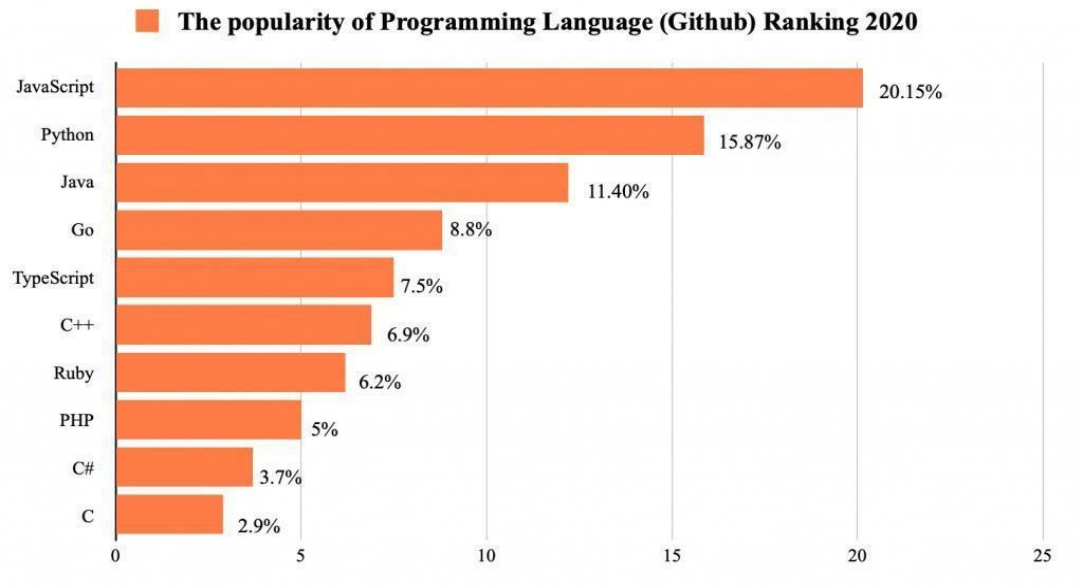

Рис. 2.8 Популярність мов програмування за даними GitHub, 2020 р.

З урахуванням предметної області та архітектури системи як клієнт-серверного вебзастосунку для серверної частини обрано мову програмування Python. Такий вибір зумовлений кількома групами факторів. По-перше, Python має розвинену екосистему бібліотек для реалізації вебсервісів, інтеграції з базами даних, взаємодії з API зовнішніх сервісів, зокрема Telegram Bot API [17]. По-друге, ця мова підтримує велике коло бібліотек для аналізу даних та машинного навчання, що є важливим з погляду подальшого розвитку системи в напрямі побудови моделей прогнозування продуктивності та виявлення аномалій у поведінці співробітників [23]. По-третє, низький поріг входу та лаконічний синтаксис позитивно впливають на швидкість розробки й зменшують ймовірність помилок у коді, що критично для невеликих команд і академічних проєктів [21].

Як серверний вебфреймворк використано Flask, який належить до класу мікрофреймворків. На відміну від важких платформ на зразок Django або Spring він не нав’язує жорсткої структуризації проєкту й дозволяє гнучко формувати REST API, орієнтований саме на ті бізнес-процеси, які закладено у тайм-трекер. Flask забезпечує маршрутизацію HTTP-запитів, роботу з шаблонами HTML, інтеграцію з ORM та механізми авторизації, залишаючи можливість поступового нарощування функціональності без значних переробок архітектури [16].

Клієнтська частина системи реалізується як вебінтерфейс, доступний через браузер, із використанням стандартних технологій HTML, CSS та JavaScript. Такий підхід дозволяє забезпечити підтримку різних платформ (настільні операційні системи, ноутбуки, тонкі клієнти) без встановлення додаткового програмного забезпечення на стороні користувача. Дизайн інтерфейсу орієнтовано на мінімалістичну концепцію, де основну увагу приділено швидкому доступу до операцій реєстрації, авторизації, створення проєктів, запуску трекера та перегляду звітів (рис. 2.1 - 2.6). Підхід підтверджує сучасна практика побудови HRIS та систем цифрового контролю персоналу [5; 7; 10].

Важливою особливістю розроблюваної системи є інтеграція зі службою обміну повідомленнями Telegram для доставки скриншотів робочого столу та сповіщень замовнику. Для цього у серверній частині застосовано бібліотеку PyTelegramBotAPI, яка спрощує роботу з Telegram Bot API, інкапсулюючи низькорівневі виклики у зручні методи [18]. Генерація скриншотів реалізується за допомогою бібліотеки типу pyautogui, що забезпечує захоплення зображення екрану в задані моменти часу та передачу результату на сервер для подальшої відправки. Така комбінація дозволяє уникнути встановлення додаткових настільних програм на комп’ютер замовника й використати Telegram як вже наявний канал комунікації [17].

Для зберігання даних обрано реляційну систему управління базами даних MySQL (або сумісну MariaDB), що забезпечує підтримку транзакцій,

механізмів цілісності, індексування та оптимізації запитів [19; 22]. Вона добре підходить для структурованої інформації про користувачів, проєкти, події трекера, коментарі і замовників. У поєднанні з SQL-модулем mysql-connector-python це дає можливість гнучко формувати запити, будувати агреговані звіти та зберігати історичні дані без втрати продуктивності. Крім того, MySQL є відкритим та широко підтримуваним рішенням, що спрощує розгортання системи на різних хостинг-платформах.

До початкового етапу проєктування було розглянуто декілька альтернативних стеків технологій. Узагальнене порівняння варіантів серверної частини наведено в таблиці 2.4, де оцінено відповідність основним критеріям: час розробки, можливості інтеграції з Telegram, підтримка аналітичних бібліотек, складність розгортання та ресурсні вимоги. Методологічні підходи до порівняння підтверджуються сучасними дослідженнями в області цифрових HR-систем [11; 13].

Таблиця 2.4

Порівняння варіантів технологій серверної частини

Варіант технології	Час розробки	Інтеграція з Telegram	Підтримка аналізу даних	Складність розгортання	Висновок
Java + Spring Boot	Середній	Потребує додаткових бібліотек	Висока, але зростає складність налаштування	Вища, потребує контейнеризації або серверів додатків	Належить для великих корпоративних систем

Node.js + Express	Невеликий	Добра підтримка через npm-пакети	Обмежена, потрібна інтеграція з окремими бібліотеками ML	Середня	Підходить для високонавантажених вебсервісів
Python + Flask (обраний стек)	Невеликий	Пряма підтримка через PyTelegramBotAPI	Висока, широка екосистема бібліотек для аналізу даних	Низька, можливе розгортання на типових VPS	Оптимальний варіант для дослідницького та малих продукційних рішень

Порівняльний аналіз підтвердив доцільність використання Python та Flask для реалізації прототипу і подальшої еволюції системи. Вони забезпечують баланс між швидкістю розробки, гнучкістю архітектури та можливостями інтеграції з інструментами аналітики [23].

Деталізацію вибору технологій для кожного компонента системи наведено в таблиці 2.5. Підхід відповідає сучасним рекомендаціям щодо побудови веборієнтованих інформаційних систем [24].

Таблиця 2.5

Вибір технологій для компонентів системи

Компонент системи	Обрана технологія	Призначення
Вебінтерфейс користувача	HTML, CSS, JavaScript	Формування сторінок реєстрації, авторизації, перегляду проєктів і звітів
Серверна логіка і REST API	Python, Flask	Обробка HTTP-запитів, авторизація, керування проєктами, запуск трекера
Модуль трекінгу часу та скриншотів	Python, pyautogui, планувальник завдань	Реєстрація подій часу, генерація та збереження скриншотів
Інтеграція з Telegram	PyTelegramBotAPI, Telegram Bot API	Надсилання скриншотів і сповіщень замовнику, отримання команд бота

Підсистема зберігання даних	MySQL, mysql-connector-python	Зберігання даних користувачів, проєктів, статусів трекера та журналів подій
Середовище розробки	PyCharm Community Edition	Розробка, налагодження та тестування програмного коду

Розробка системи здійснюється в середовищі PyCharm Community Edition, що забезпечує інтеграцію з системами контролю версій, інструментами тестування та статичного аналізу коду [20]. Використання єдиного середовища для клієнтських і серверних компонентів знижує трудомісткість супроводу та полегшує навчання студентів, які залучаються до проєкту.

Окремо визначено мінімальні вимоги до технічного забезпечення, що гарантують коректну роботу вебзастосунку. Для клієнтської сторони достатньо персонального комп’ютера або ноутбука з операційною системою класу Windows 10, процесором із тактовою частотою від 2 ГГц, не менше ніж 1 ГБ оперативної пам’яті, доступом до мережі Інтернет та сучасним браузером (Google Chrome, Mozilla Firefox, Meta тощо). З боку замовника необхідною умовою є наявність встановленого застосунку Telegram, через який отримуються повідомлення та скриншоти. Вимоги відповідають сучасним рекомендаціям у сфері проєктування корпоративних інформаційних систем [12]

Таким чином, обраний технологічний стек поєднує сучасні відкриті інструменти веброзробки, перевірені реляційні механізми зберігання даних і зручні засоби інтеграції з Telegram. Це створює надійну основу для реалізації та подальшого розвитку інформаційної системи аналізу, контролю та обліку роботи співробітників відповідно до вимог.

2.3 Архітектурна модель системи

Архітектурна модель інформаційної системи аналізу, контролю та обліку роботи співробітників ґрунтується на класичній трирівневій схемі

"клієнт – сервер застосунків – сервер бази даних", що дає змогу фізично розділити інтерфейс користувача, бізнес-логіку та підсистему зберігання даних. Узагальнену структуру такої взаємодії подано на рисунку 2.9. На рівні вебклієнта відбувається введення й візуалізація даних у браузері, сервер застосунків реалізує REST-інтерфейси та основні алгоритми роботи тайм-трекера, а сервер бази даних відповідає за надійне збереження інформації про користувачів, проєкти, події роботи трекера і коментарі.

Рис. 2.9 Схема архітектури інформаційної системи

Логіка роботи системи деталізується у вигляді сервісно орієнтованої структури (рис. 2.10), де центральне місце посідає серверна частина, що виконує роль шлюзу API.

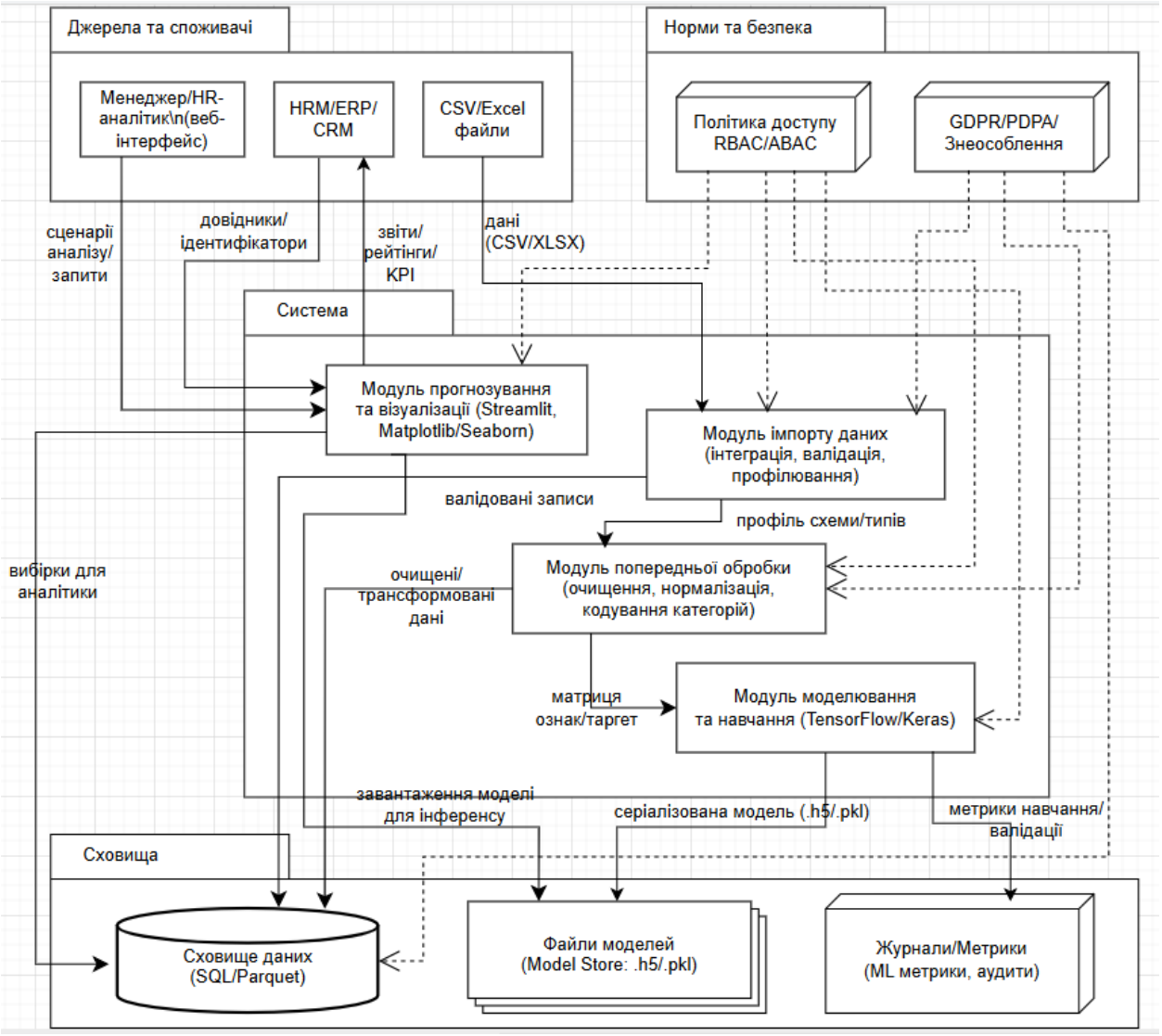

Рис. 2.10 Узагальнена схема інформаційних потоків системи та взаємодії сервісів

Користувач типу "виконавець" або "замовник" взаємодіє з вебінтерфейсом (Web UI), надсилаючи вхідні дані для реєстрації, авторизації, створення та налаштування проєктів. Інтерфейс користувача формує HTTP-запити у форматі JSON та передає їх до серверної частини. API-шлюз маршрутизує ці запити до відповідних доменних сервісів: сервісу управління проєктами (Project Service), сервісу трекінгу часу (Time-Tracker Service), сервісу скриншотів (Screenshot Service) та сервісу звітності (Reporting Service). Дані про проєкти, статуси трекера та результати обробки запитів зберігаються у сховищі даних на базі реляційної СУБД.

Сервіс скриншотів взаємодіє з модулем команд Telegram Bot API, який забезпечує відправку знімків екрана й текстових сповіщень замовнику. Таким чином реалізується наскрізний інформаційний потік від дій виконавця до отримання замовником валідаційних даних щодо витраченого часу та фактичної активності.

Для опису внутрішньої структури програмного забезпечення використано UML-діаграму класів, фрагмент якої наведено на рисунку 2.11.

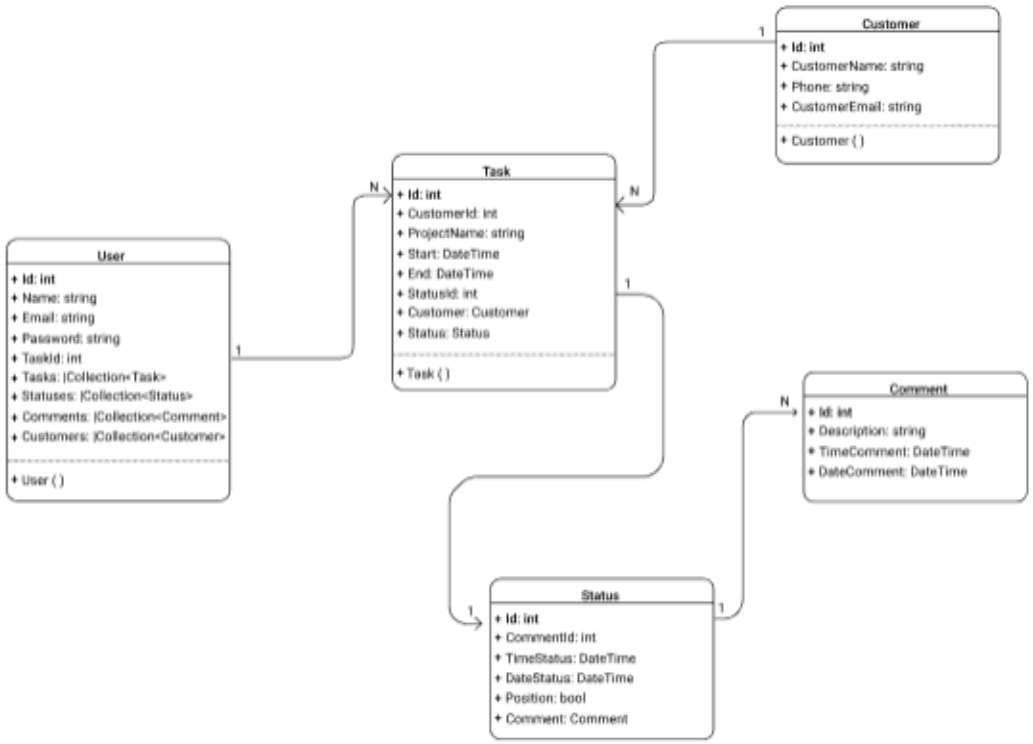

Рис. 2.11 Діаграма класів інформаційної системи

Модель даних реалізує основні сутності предметної області: User, Task, Customer, Status, Comment. Клас User описує виконавця, містить атрибути ідентифікатора, імені, електронної пошти, пароля та навігаційні властивості до колекцій завдань, статусів і коментарів. Клас Task відповідає окремому проєкту чи завданню, пов’язаний відношенням "один-до-багатьох" із замовником (Customer) та статусами трекера (Status), що дозволяє фіксувати історію зміни станів. Сутність Comment використовується для зберігання текстових пояснень виконавця й тимчасових міток, пов’язаних з окремими подіями трекінгу. Взаємозв’язки між класами забезпечують узгодженість

даних та підтримку цілісності на рівні ORM-моделі.

Специфікацію основних функцій доменної моделі узагальнено у таблиці 2.6. Вона демонструє відповідність між класами, їхнім призначенням та ключовими полями, які визначають логіку обліку часу і побудови звітів.

Таблиця 2.6

Специфікація основних функцій класів доменної моделі

Назва класу	Призначення
User	Описує користувача типу "виконавець" або "замовник", містить параметри для автентифікації (Email, Password), загальну інформацію (Name) та посилання на пов’язані завдання, статуси і коментарі.
Task	Моделює проєкт чи окреме завдання; включає ідентифікатор завдання, посилання на замовника, назву проєкта, дати початку і завершення, а також ідентифікатор поточного статусу трекера.
Customer	Відповідає замовнику послуг; містить ідентифікатор, ім’я, контактний номер телефону для зв’язку через Telegram та адресу електронної пошти.
Status	Фіксує стани тайм-трекера, зокрема параметр увімкнення/вимкнення, час і дату зміни статусу, посилання на коментар та завдання, до якого належить подія.
Comment	Зберігає текстові коментарі виконавця з атрибутами ідентифікатора, опису, часу і дати створення, що використовуються для пояснення контексту зафіксованих подій.

Динамічні аспекти архітектури відображено за допомогою діаграм діяльності та послідовності. На рисунку 2.12 подано діаграму діяльності, яка описує загальний життєвий цикл процесу обліку робочого часу. Розробник після авторизації обирає проєкт і вмикає трекер; система формує статистику витрат часу та ініціює створення скриншотів екрана. Замовник аналізує отримані знімки й статистичні показники, робить висновки щодо фактичних витрат часу, узгоджує вартість робіт і переходить до етапу оплати. Така модель демонструє роль кожного учасника процесу і візуалізує, на яких етапах задіяні програмні модулі системи.

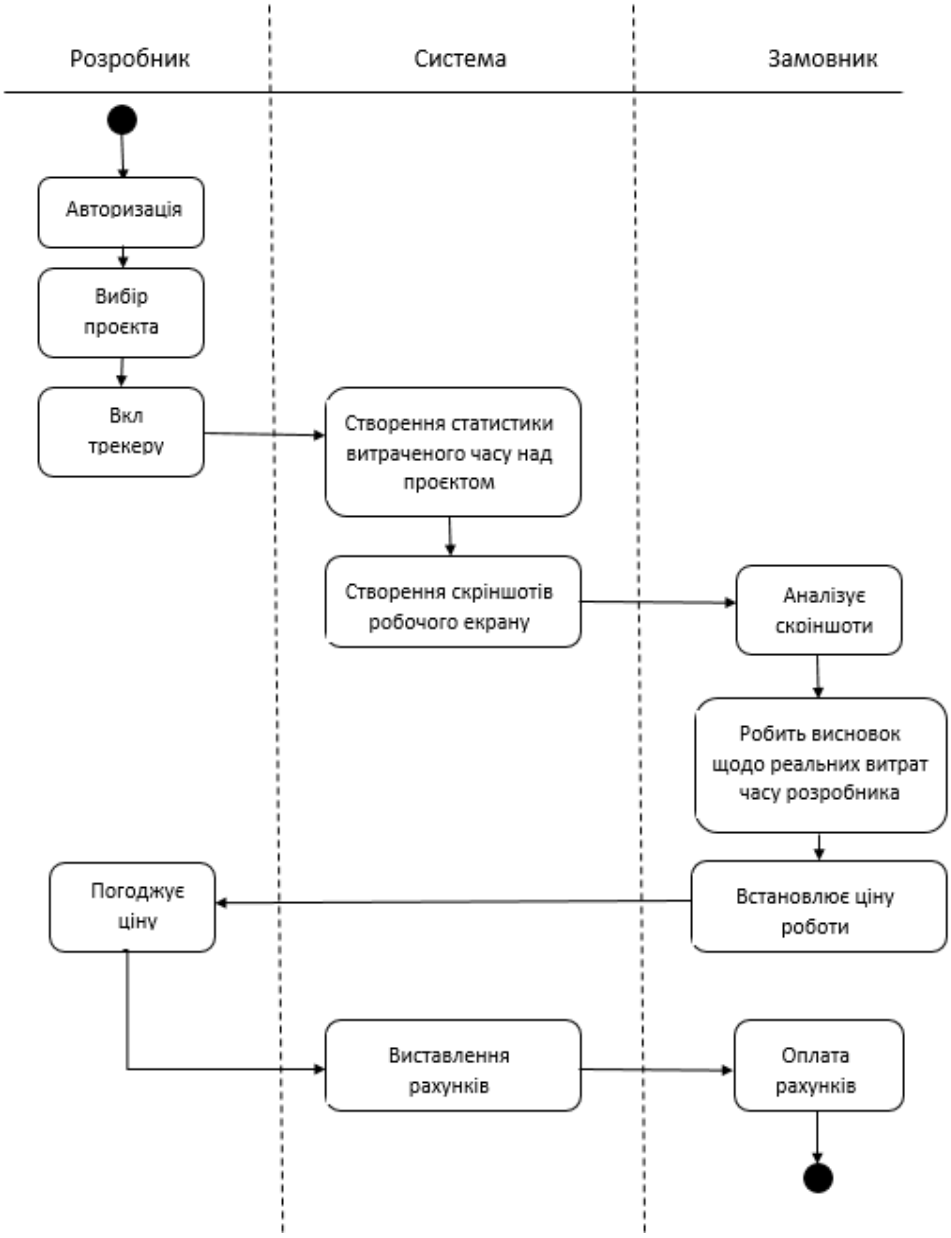

Рис. 2.12 Діаграма діяльності процесу обліку робочого часу

Послідовність обміну повідомленнями між клієнтським застосунком, сервером і базою даних під час авторизації користувача показано на рисунку 2.13. Користувач вводить облікові дані, клієнтська частина формує запит на авторизацію, система виконує верифікацію через звернення до сховища даних та повертає результат перевірки.

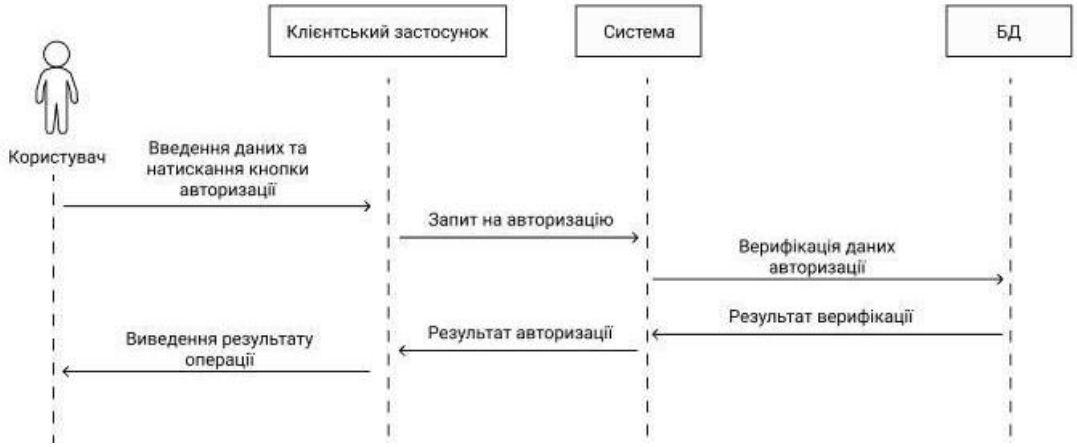

Рис. 2.13 Діаграма послідовності процесу авторизації користувача

Аналогічно на рисунку 2.14 відображено послідовність взаємодії під час створення проєкту, запуску трекера та передачі скриншотів: клієнтський застосунок надсилає дані для створення проєкту, система їх зберігає, далі у випадкову хвилину кожного десятого інтервалу часу ініціюється захоплення зображення екрана, після чого скриншот надсилається до бота, а замовник отримує підтвердження про успішне збереження та оплату виконаних робіт.

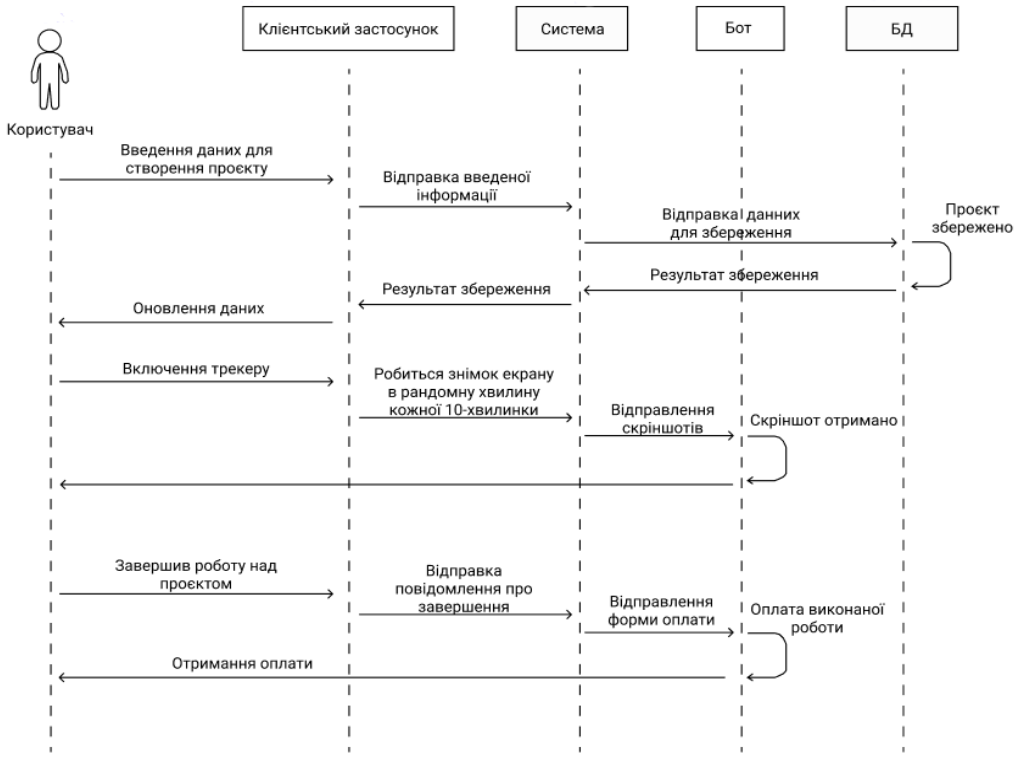

Рис. 2.14 Діаграма послідовності роботи трекера та взаємодії з Telegram-ботом

Статичну декомпозицію програмного продукту на інтерфейсну та серверну частини ілюструє діаграма компонентів (рис. 2.15). У складі серверної частини виділено модуль моделі, що реалізує доменні класи та логіку доступу до бази даних, і модуль конфігурації, який відповідає за налаштування підключення до СУБД, параметри авторизації, роботу з зовнішніми сервісами. Інтерфейсна частина містить компоненти вебсторінок, які звертаються до API серверної частини через HTTP-запити. Чітке розмежування відповідальності між компонентами зменшує зв’язність системи та полегшує її супровід, а також створює передумови для подальшого переходу до мікросервісного підходу.

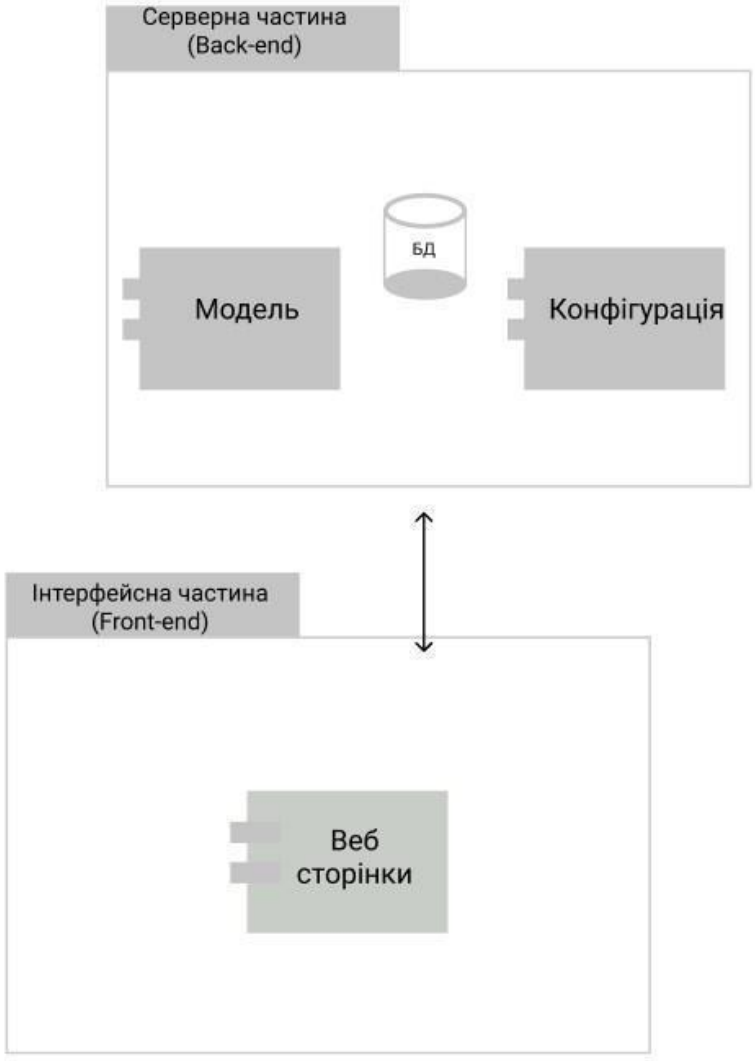

Рис. 2.15 Діаграма компонентів фронтенд- та бекенд-частин системи

Отже, запропонована архітектурна модель поєднує трирівневу клієнт-серверну організацію, сервісне розділення функцій, формальну UML-модель класів та діаграми поведінки, що описують ключові сценарії використання системи. Такий підхід забезпечує прозорий опис структури і механізмів роботи програмного забезпечення, підтримує можливість подальшого розширення функціональності, інтеграції аналітичних модулів і масштабування рішення на більшу кількість користувачів та проєктів.

2.4 Проєктування основних модулів

Проєктування основних модулів інформаційної системи аналізу, контролю та обліку роботи співробітників здійснюється на основі наперед визначеної архітектури клієнт-серверного застосунку та логічної моделі даних, описаної в попередніх підрозділах. Ключовою вимогою до структури програмного забезпечення є чітке розмежування відповідальності між модулями користувацького інтерфейсу, бізнес-логікою керування проєктами і трекером часу, підсистемою обробки скриншотів, модулем інтеграції з Telegram та підсистемою зберігання даних. Такий підхід забезпечує можливість незалежного розвитку окремих компонентів, спрощує супровід та дає змогу масштабувати систему без радикальної зміни програмного коду.

Структура бази даних відповідає вимогам до мінімізації надмірності та забезпечення цілісності інформації. Детальна структурна схема бази даних подана на рисунку 2.7. Вона містить п’ять взаємопов’язаних сутностей: User, Task, Status, Comment та Customer. Сутність User описує зареєстрованих користувачів системи та їх облікові дані; сутність Task фіксує проєкти та окремі завдання, пов’язані з певним замовником; сутність Status відображає хронологію станів трекера часу; сутність Comment містить текстові пояснення до подій, а сутність Customer акумулює контактну інформацію замовників. Зв’язки між таблицями реалізовано через зовнішні ключі типу "один до багатьох", що дозволяє для одного користувача реєструвати багато завдань, для одного завдання – декілька статусів і коментарів, а також пов’язувати

групу завдань з конкретним замовником.

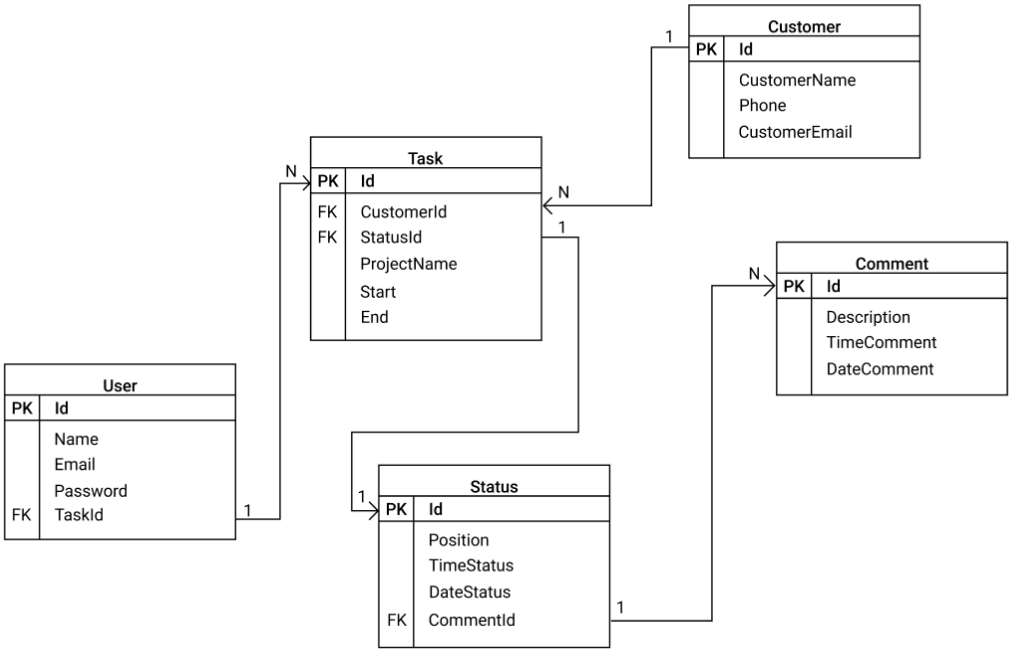

Рис. 2.16 Схема структурна бази даних інформаційної системи

Опис атрибутів таблиць бази даних, їх типів та призначення наведено в таблиці 2.7. Наведена структура забезпечує достатню деталізацію для відтворення повного життєвого циклу роботи над завданням: від створення запису про користувача та проєкт до реєстрації послідовних змін стану трекера й фіксації пояснювальних коментарів.

Таблиця 2.7

Структура таблиць розробленої бази даних

Назва стовпця	Тип даних	Опис
1	2	3
Таблиця User		
Id	int	Ідентифікатор користувача
Name	nvarchar(max)	Ім’я користувача
Email	nvarchar(max)	Пошта користувача
Password	nvarchar(max)	Пароль користувача
TaskId	int	Ідентифікатор завдання
Таблиця Task		
Id	int	Ідентифікатор завдання
CustomerId	int	Ідентифікатор замовника
StatusId	int	Ідентифікатор статусу
ProjectName	nvarchar(max)	Назва проєкту
Start	datetime	Створення завдання
End	datetime	Дедлайн виконання завдання

Продовження табл. 2.7

1	2	3
Таблиця Status		
Id	int	Ідентифікатор статусу
Position	bool	Режим трекеру
TimeStatus	datetime	Час включення
DateStatus	datetime	Дата включення
CommentId	nvarchar(max)	Ідентифікатор коментування
Таблиця Comment		
Id	int	Ідентифікатор коментування
Description	nvarchar(max)	Опис
TimeComment	datetime	Час коментування
DateComment	datetime	Дата коментування
Таблиця Customer		
Id	int	Ідентифікатор замовника
CustomerName	nvarchar(max)	Ім’я замовника
Phone	nvarchar(max)	Телефон замовника
CustomerEmail	nvarchar(max)	Пошта замовника

На основі наведеної схеми бази даних формуються основні програмні модулі системи. Кожен модуль оперує визначеним підмножинним набором таблиць і реалізує окремий фрагмент бізнес-логіки. Узагальнена відповідність між модулями та сутностями бази даних наведена у таблиці 2.8.

Таблиця 2.8

Основні модулі системи та їх взаємозв’язок із сутностями бази даних

Модуль системи	Основні функції	Сутності БД, що використовуються
Модуль керування користувачами (User Management)	реєстрація, авторизація, зміна профілю, деактивація облікових записів	User
Модуль керування проєктами (Project Service)	створення та редагування завдань, прив’язка до замовників, визначення дедлайнів	Task, Customer
Модуль трекінгу часу (Time-Tracker Service)	запуск і зупинка трекера, фіксація часових міток, обчислення тривалості сесій	Task, Status
Модуль коментарів (Comment Service)	збереження текстових пояснень, відображення історії подій	Comment, Status
Модуль скриншотів і Telegram-інтеграції (Screenshot & Telegram Service)	генерація скриншотів, передавання зображень через Telegram-бот, обробка команд бота	Task, Status, Customer
Модуль звітності (Reporting Service)	формування підсумкових звітів за проєктами та користувачами, експорт статистики	усі сутності

Функціональна взаємодія зазначених модулів відбувається в межах трирівневої архітектури вебзастосунку. Узагальнена схема розподілу відповідальності між вебклієнтом, сервером застосунків і сервером бази даних наведена на рисунку 2.17. Вебклієнт забезпечує введення та візуалізацію даних, сервер застосунків реалізує бізнес-логіку модулів User Management, Project Service, Time-Tracker Service, Screenshot & Telegram Service та Reporting Service, а сервер бази даних відповідає за надійне зберігання інформації й виконання запитів.

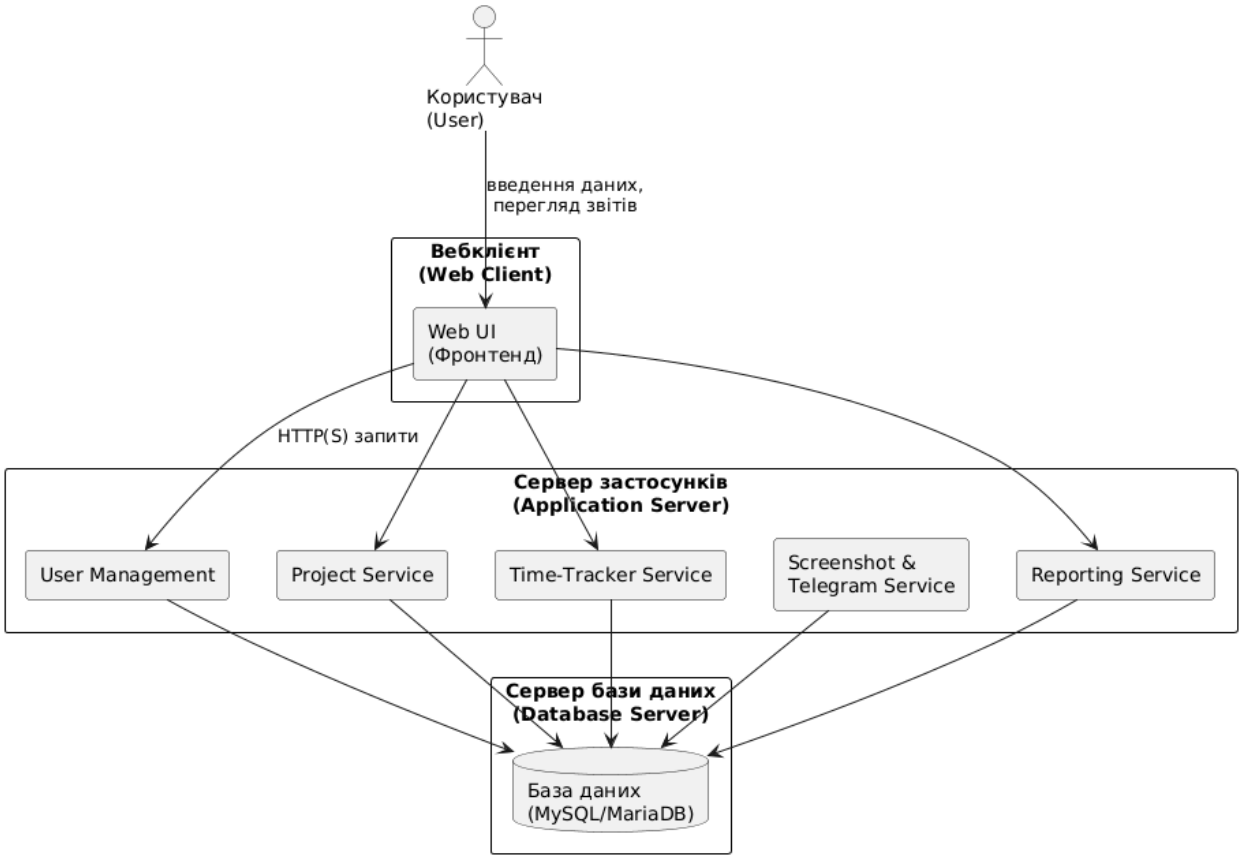

Рис. 2.17 Узагальнена трирівнева схема розміщення основних модулів системи

Більш детальне уявлення про інформаційні потоки між основними компонентами наведено на рисунку 2.18. Користувач взаємодіє з інтерфейсом Web UI, який надсилає HTTP-запити до серверної частини (API Gateway). Остання маршрутизує запити до окремих сервісів: модуля керування проєктами, модуля трекінгу часу та модуля скриншотів. Після обробки даних

сервер звертається до сховища, оновлює відповідні записи у таблицях і, за потреби, ініціює відправку скриншотів та сповіщень через Telegram Bot API. Результати обробки повертаються до Web UI у вигляді структурованих відповідей, що відображаються користувачеві у вигляді списків завдань, таймерів та звітів.

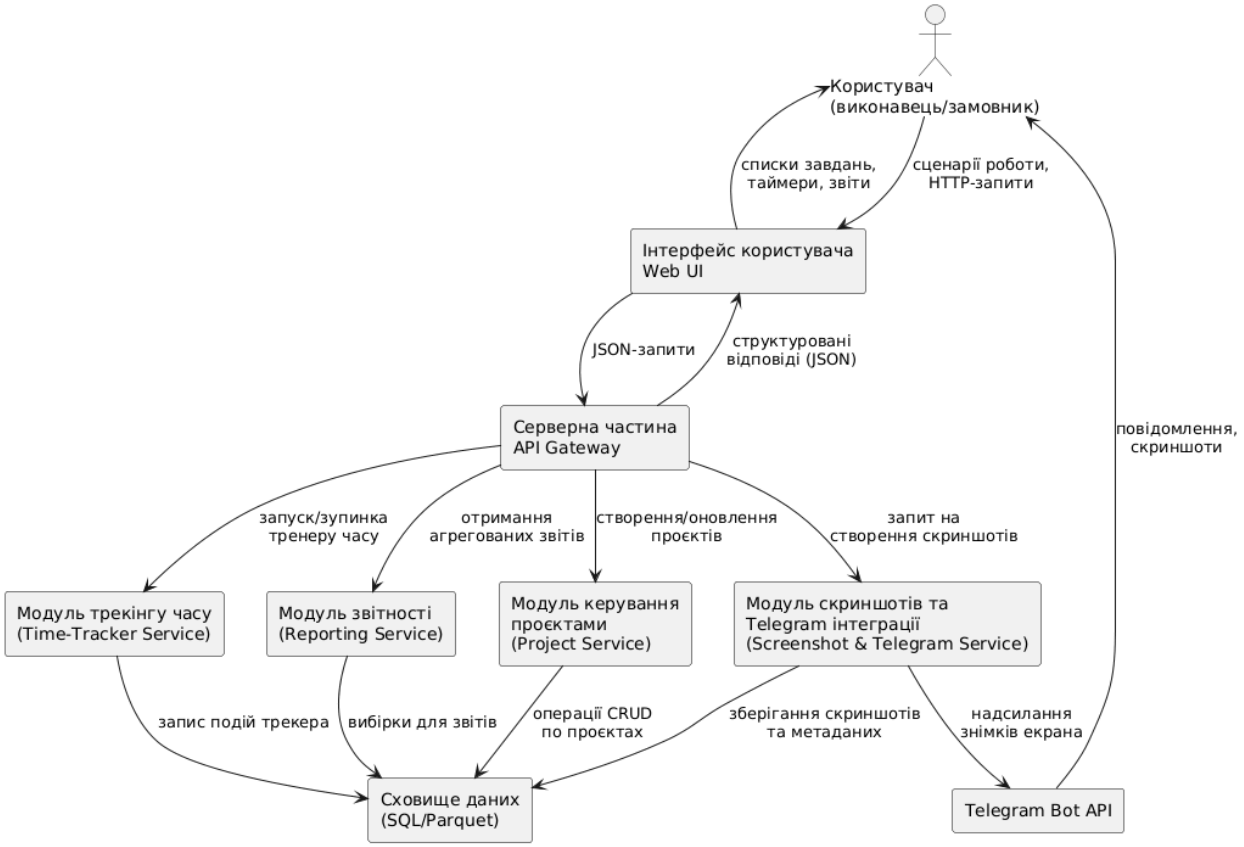

Рис. 2.18 Узагальнена схема інформаційних потоків між основними модулями системи

Проєктування модулів також спирається на формалізований опис бізнес-процесів, пов’язаних із використанням трекера. На рисунку 2.19 наведено діаграму діяльності, яка ілюструє взаємодію розробника, системи та замовника. Розробник авторизується в системі, обирає проєкт і вмикає трекер. Система в автоматичному режимі формує статистику витраченого часу та генерує скриншоти робочого екрана, що передаються замовнику. На підставі отриманої інформації замовник аналізує хід роботи, ухвалює рішення щодо вартості послуг, погоджує ціну та здійснює оплату. Така схема демонструє

місце кожного модуля в загальному процесі та слугує основою для подальшої алгоритмізації.

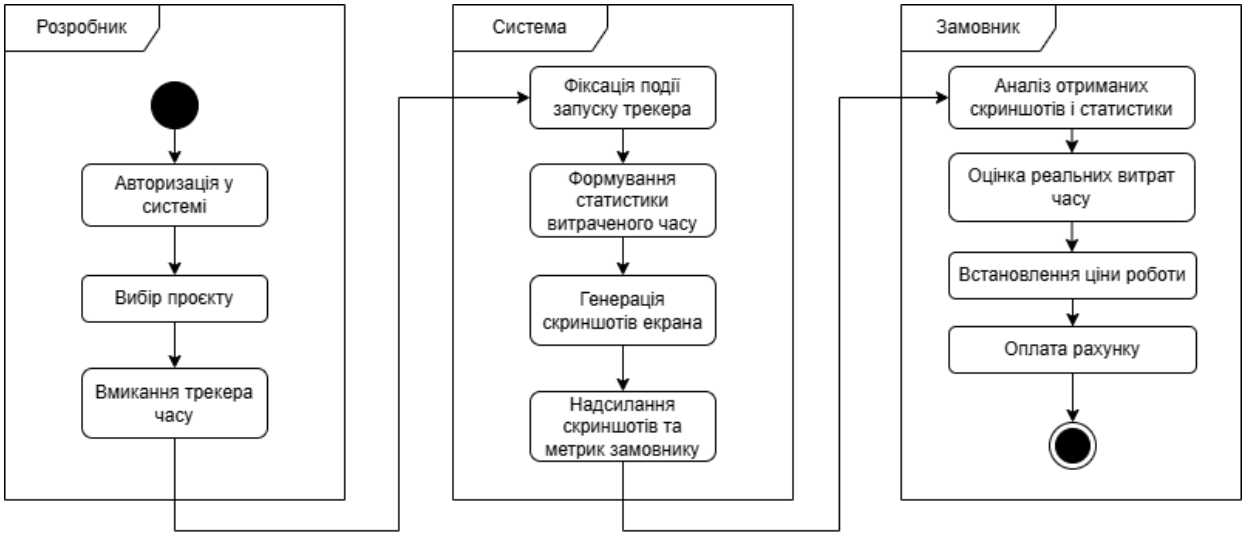

Рис. 2.19 Діаграма діяльності процесу використання трекера між виконавцем, системою та замовником

Послідовність взаємодії клієнтського застосунку, серверної частини, Telegram-бота та бази даних під час роботи з трекером подано на діаграмі послідовності (рис. 2.20). Структура цього процесу відображає повний життєвий цикл роботи користувача з системою – від створення нового проєкту до завершення виконання завдання й ініціювання фінальних процедур оплати. На початковому етапі користувач вводить параметри нового проєкту, після чого клієнтський застосунок передає їх серверній частині. API Gateway здійснює перевірку коректності вхідних даних, викликає бізнес-логіку сервісів та записує створені сутності (Project, Task, Customer, UserTask) у базу даних. Отримавши підтвердження успішного збереження, клієнтська частина інформує користувача про створення нового робочого середовища.

Під час активної роботи над завданням клієнтський застосунок керує статусами трекера, а сервер реєструє часові мітки включення та виключення, зберігаючи їх у таблиці Status. Запуск трекера активує періодичний цикл створення скриншотів, реалізований у модулі Screenshot Service.

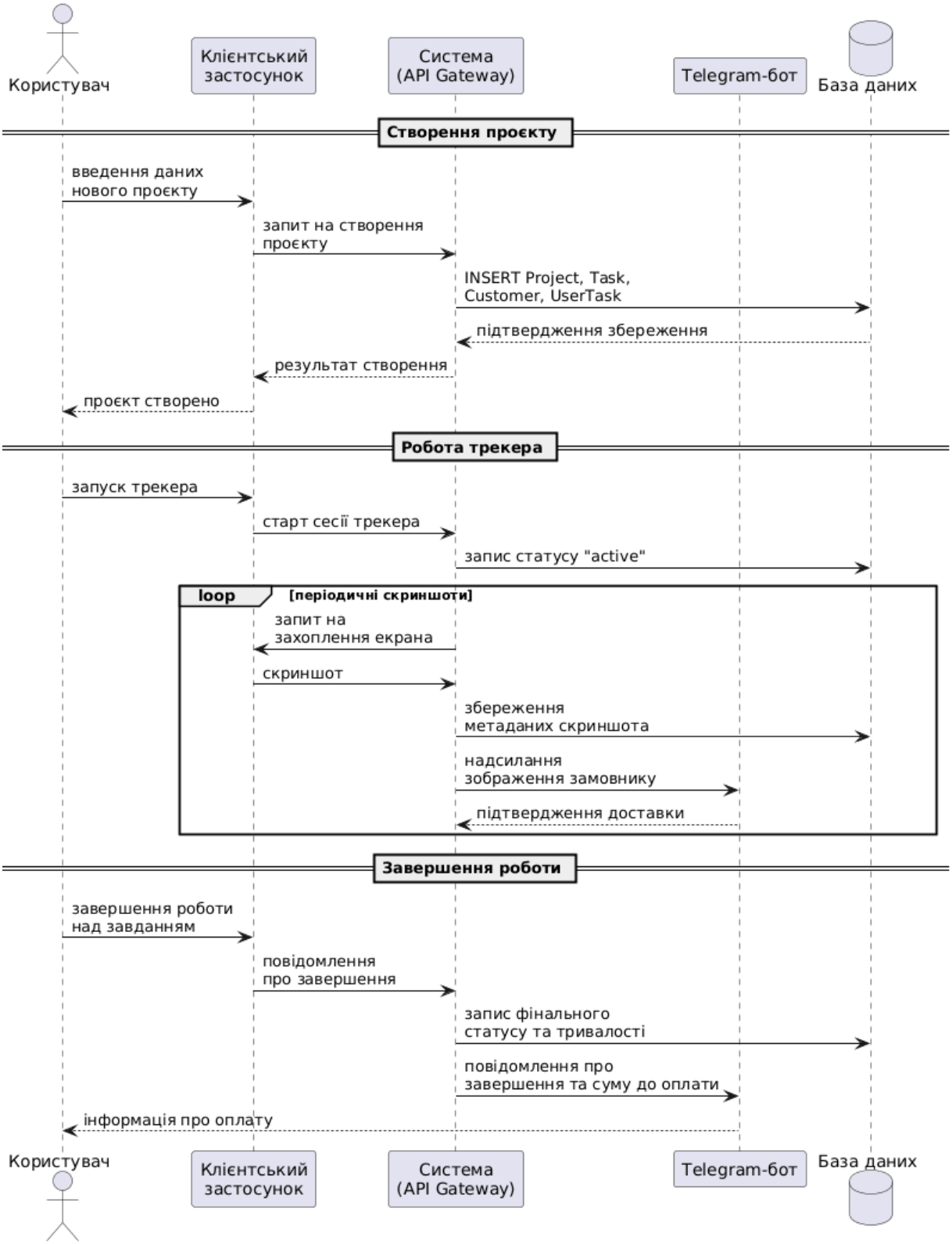

Рис. 2.20 Діаграма послідовності роботи трекера часу та взаємодії з Telegram-ботом

У межах визначеного інтервалу система надсилає запит на захоплення екрану, отримує сформоване зображення, зберігає його метадані у сховищі та

надсилає сам скриншот Telegram-боту замовника. Надалі бот передає користувачеві підтвердження доставки, що гарантує безперервність інформаційного контролю з боку замовника та прозорість виконання робіт.

Завершення роботи над проєктом супроводжується надсиланням клієнтом відповідного повідомлення на сервер. API Gateway реєструє фінальний статус у базі даних, формує підсумкові показники часу, підготовляє агреговану інформацію та ініціює надсилання замовнику сповіщення щодо завершення та необхідності здійснення оплати. Telegram-бот виконує роль каналу зворотного зв’язку, забезпечуючи оперативне інформування про фінансові операції та підтвердження їх виконання.

Запропонована структура модулів забезпечує узгодженість між логічною моделлю даних і реальними сценаріями експлуатації системи, а також дозволяє масштабувати рішення відповідно до потреб користувачів. Чітке розмежування функцій між клієнтською частиною, сервером застосунків, Telegram-ботом та базою даних спрощує впровадження нових компонентів, таких як модулі прогнозування продуктивності, інструменти аналітики ефективності роботи або інтеграція з зовнішніми платформами управління проєктами. Така архітектурна організація підвищує надійність, відмовостійкість і гнучкість системи, забезпечуючи її подальший розвиток і адаптацію до змін у вимогах користувачів.

2.5 Модель безпеки

Модель безпеки інформаційної системи аналізу, контролю та обліку робочого часу ґрунтується на принципах конфіденційності, цілісності та доступності даних, а також на вимогах регламентів захисту персональної інформації. Оскільки система оперує чутливими даними користувачів – обліковими записами, журналами активності, часовими статусами, скриншотами робочого екрана та контактною інформацією замовників – її архітектура повинна забезпечувати строгий розподіл прав доступу, аудит операцій та відповідність міжнародним нормам, зокрема GDPR. З цією метою

застосовано комбінацію моделей RBAC (Role-Based Access Control) і ABAC (Attribute-Based Access Control), що дозволяє гнучко регулювати доступ до ресурсів на основі ролей, атрибутів користувачів і контекстних умов роботи.

У межах моделі RBAC ролі визначають основні можливості взаємодії з системою: виконавець може керувати власними завданнями, запускати та зупиняти трекер, переглядати власні скриншоти; замовник отримує доступ лише до інформації, пов’язаної з його проєктами, включно з часовими логами та надісланими скриншотами; адмін здійснює технічне керування системою, налаштування проєктів і управління користувачами. При цьому ABAC розширює можливості RBAC за рахунок контекстних атрибутів, що охоплюють час доступу, тип операції, статус проєкту, айпі-адресу, а також рівень чутливості ресурсу. Такий підхід дає змогу формувати політики доступу, що автоматично блокують спроби перегляду або модифікації інформації поза межами дозволених сценаріїв.

Для системи визначено набір атрибутів, що беруть участь у прийнятті рішень щодо доступу, а саме: User.Role, User.ProjectId, Resource.Type, Resource.Owner, Action.Type, Context.Time, Context.IP тощо. Приклад відповідності типових політик безпеки наведено в таблиці 2.9, де для кожного сценарію визначено необхідні атрибути та прийняте рішення контролера доступу.

Таблиця 2.9

Приклади політик доступу в моделі ABAC

№	Ресурс	Атрибути	Умова	Рішення
1	Скриншоти виконавця	User.Role = "Owner" AND Resource.Owner = User.Id	Доступ лише до власних скриншотів	Permit
2	Скриншоти проєкту	User.Role = "Customer" AND User.ProjectId = Resource.ProjectId	Перегляд скриншотів лише замовником	Permit
3	Управління користувачами	User.Role = "Admin"	Операції створення, блокування та редагування користувачів	Permit
4	Доступ із невідомої IP	Context.IP ∉ TrustedList	Заборона доступу незалежно від ролі	Deny
5	Запуск трекера	User.Role = "Owner" AND Resource.Type = "Task"	Доступ лише до власних завдань	Permit

Взаємодія механізмів RBAC та ABAC у межах системи зображена на рисунку 2.21, де продемонстровано узагальнений підхід до визначення дозволів шляхом послідовного оцінювання ролей, атрибутів та політик доступу.

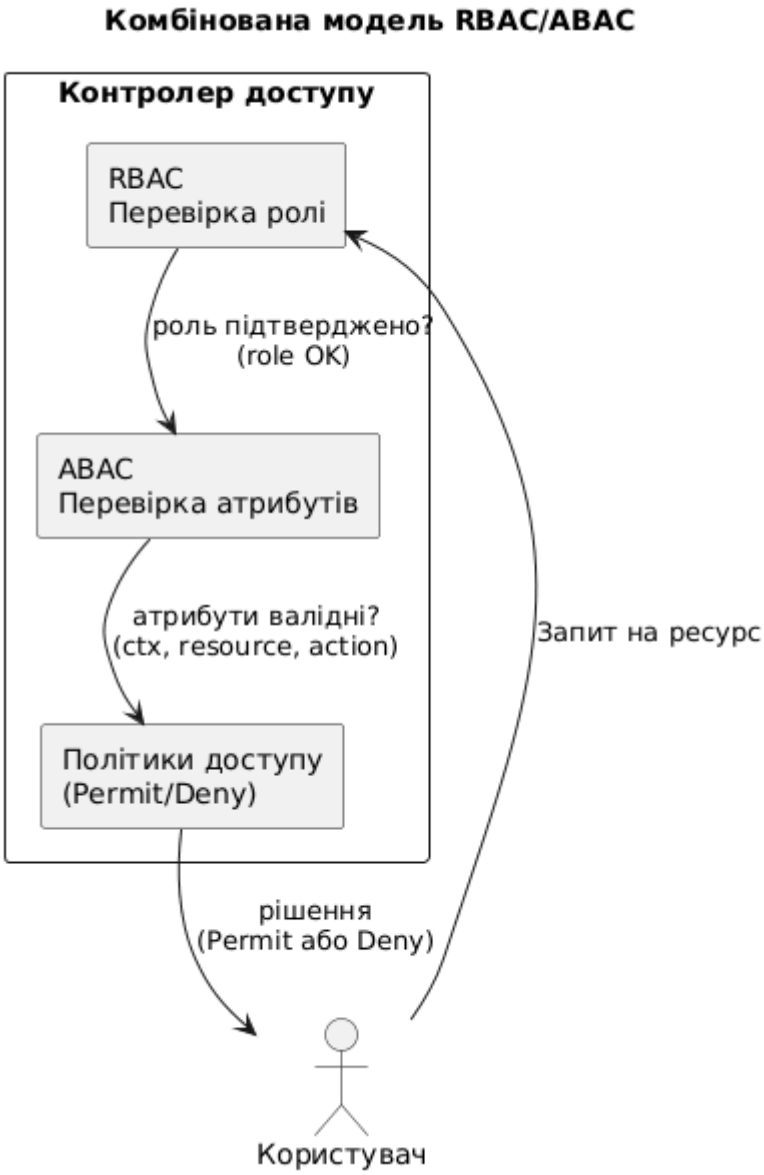

Рис. 2.21 Узагальнена схема поєднання RBAC та ABAC у системі

На нормативному рівні система повинна відповідати вимогам GDPR (General Data Protection Regulation), оскільки оперує персональними даними громадян ЄС або може використовуватися компаніями, що виконують міжнародні контракти. Основні принципи GDPR, які враховані під час проектування моделі безпеки, включають: мінімізацію даних, цільове

обмеження, точність і актуальність інформації, право на доступ та видалення, право на обмеження обробки, запобігання несанкціонованому розкриттю та аудит операцій з персональними даними. У таблиці 2.10 наведено ключові вимоги GDPR та відповідні механізми, реалізовані у системі.

Таблиця 2.10

Відповідність GDPR і реалізованих заходів системи

Принцип GDPR	Реалізований механізм
Мінімізація даних	Зберігаються лише необхідні поля: Email, ім’я, статуси, скриншоти проєкту
Прозорість обробки	Логування операцій у журналах (audit log)
Право на доступ	Функція експорту даних користувача
Право на видалення	Механізм деактивації та фізичного видалення профілю
Захист даних	Шифрування паролів (bcrypt), HTTPS, TLS
Обмеження доступу	RBAC + ABAC політики доступу
Повідомлення про інциденти	Система сповіщень адміністратору

Для узагальнення моделі захисту інформаційних потоків побудовано структурну схему взаємодії компонентів безпеки (рис. 2.22). На схемі відображено модуль автентифікації, підсистему авторизації, модуль контролю політик, сховище ключів, журнал аудиту та ізольовані канали передачі даних між сервісами.

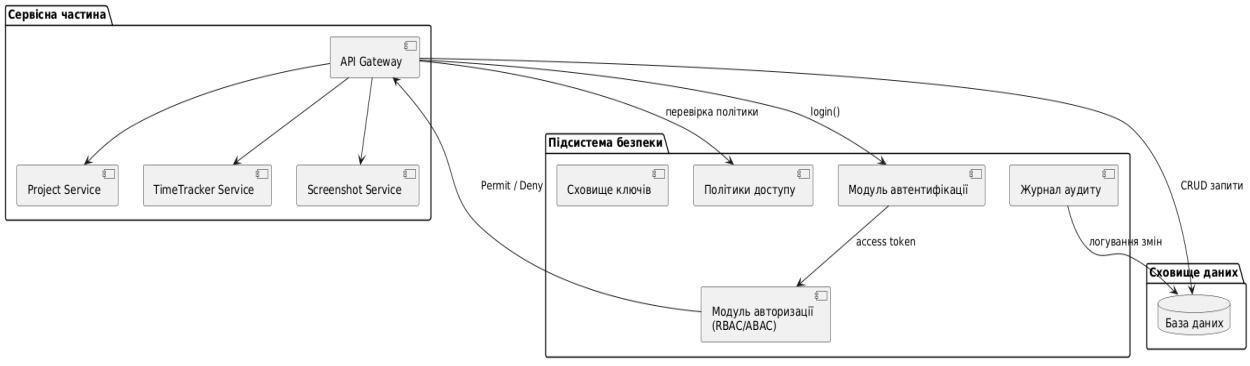

Рис. 2.22 Структурна схема підсистеми безпеки інформаційних потоків

Запроваджена модель безпеки формує цілісний комплекс механізмів, що забезпечують надійний захист даних і контроль дій користувачів у всіх модулях системи. Поєднання RBAC і ABAC дозволяє масштабувати політики

доступу залежно від зростання кількості проєктів, користувачів і типів ресурсів. Врахування вимог GDPR забезпечує юридичну коректність обробки персональної інформації, а вбудовані механізми аудиту та ізольовані канали зв’язку гарантують цілісність і простежуваність усіх операцій. Такий підхід створює підґрунтя для подальшого розширення функціональності системи та підтримує необхідний рівень інформаційної та правової безпеки під час її експлуатації.

2.6 Висновки до 2 розділу

У розділі було побудовано комплексну архітектурну модель веб-орієнтованої системи трекінгу часу, скриншот-моніторингу та генерації аналітичних звітів. Проведене проєктування охопило структуру функціональних модулів, логічну модель даних, механізми взаємодії клієнтського інтерфейсу із серверною частиною та інформаційні потоки між сервісами системи. Розроблені діаграми (компонентна, діяльності, послідовності, схема безпеки та моделі доступу RBAC/ABAC) продемонстрували узгодженість між користувацькими сценаріями, бізнес-логікою застосунку та інфраструктурними вимогами, що підтверджує коректність обраного підходу до побудови трирівневої архітектури.

Результати аналізу безпеки показали, що інтеграція механізмів автентифікації, ролевої та атрибутної моделей доступу, протоколів шифрування та відповідність GDPR формують необхідний рівень захисту персональних даних і забезпечують контроль доступу до критичних ресурсів системи. Узагальнення результатів дозволяє стверджувати, що розроблена архітектура є масштабованою, безпечною та придатною для подальшого розширення, зокрема за рахунок модулів прогнозної аналітики, інтелектуальних рекомендацій та інтеграції з зовнішніми сервісами управління проєктами.

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Логічна структура програмного забезпечення

Логічна структура програмного забезпечення системи аналізу, контролю та обліку робочого часу визначає взаємодію її функціональних компонентів, механізмів обробки даних та інтерфейсів користувача в межах трирівневої архітектури. Проєктована система включає клієнтський застосунок, серверну частину з модульною бізнес-логікою та централізоване сховище даних, що забезпечує узгодженість станів, обробку подій і формування звітів. Кожен компонент виконує чітко визначені функції, а взаємодія між ними реалізується через REST-інтерфейси та стандартизовані протоколи обміну даними.

У клієнтському рівні реалізовано браузерний інтерфейс, який забезпечує авторизацію, створення проєктів, управління сеансами трекера, перегляд статистики та отримання сповіщень щодо змін статусів завдань. Логічна модель клієнтського інтерфейсу базується на принципах мінімізації кількості кроків користувача, використанні стандартизованих форм введення та чіткого поділу екранів за функціональними призначеннями. На серверному рівні функціонує модульна система, що складається з API Gateway, сервісів керування користувачами, управління проєктами, модуля трекінгу часу, модуля обробки скриншотів і сервісу формування звітів. Кожен сервіс має власний набір REST-ендпоїнтів, політику доступу та механізми обробки даних, що забезпечує низьку зв’язність і високу масштабованість архітектури.

Центральне сховище даних структурується у вигляді набору реляційних таблиць, об’єднаних логічними зв’язками «один-до-багатьох» та «багато-до-одного». Таблиці Project, Task, TimeEntry, ScreenshotLog, User та Customer забезпечують зберігання об’єктів доменної моделі, тоді як службові таблиці AccessPolicy та AuditLog відповідають за фіксацію подій авторизації, операцій модифікації даних та перевірку цілісності системи. На логічному рівні створено модель, у якій бізнес-процеси на кшталт створення проєкту, запуску

трекера, надсилання скриншоту або завершення завдання формуються як транзакційні операції з чіткою послідовністю дій, що дозволяє виключити дублювання записів і забезпечити відмовостійкість системи.

На рисунку 3.1 подано узагальнену компонентну схему логічної структури ПЗ, що відображає основні модулі, їх призначення та взаємозв'язки. Вона демонструє, що система побудована як набір незалежних сервісів з чіткою границею відповідальності та єдиною точкою доступу – API Gateway.

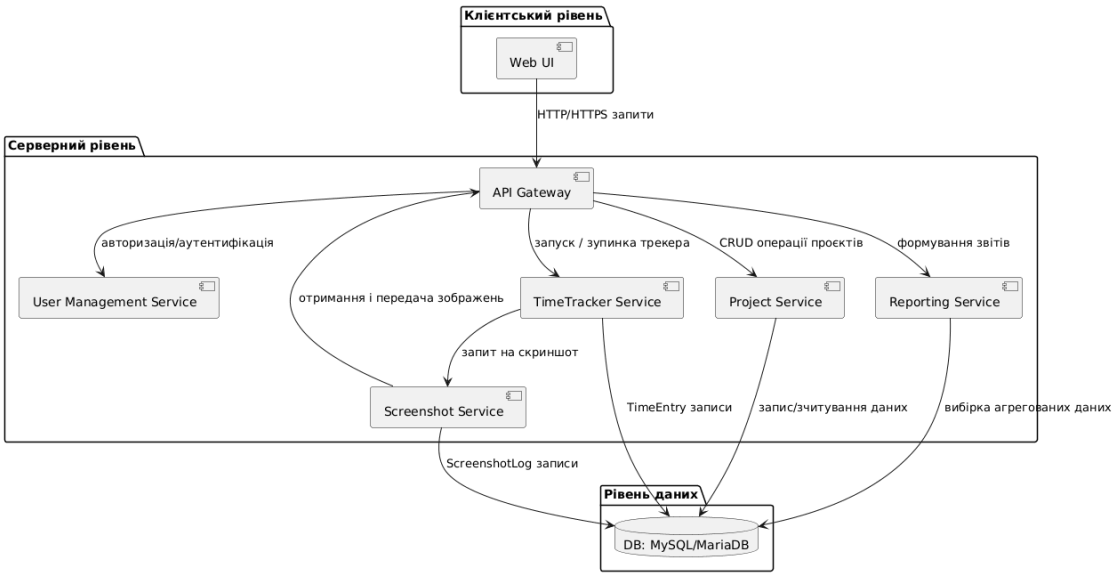

Рис. 3.1 Узагальнена логічна структура програмного забезпечення

Наступна частина логічної структури описує внутрішню організацію модулів серверної частини. Кожен модуль розглядається як функціональний блок, що обробляє власну групу запитів, забезпечує валідацію даних, застосовує політики доступу та виконує бізнес-правила. На рис. 3.2 представлено деталізовану схему модульної структури серверної частини, яка узагальнює функції кожного сервісу та напрямки інформаційних потоків.

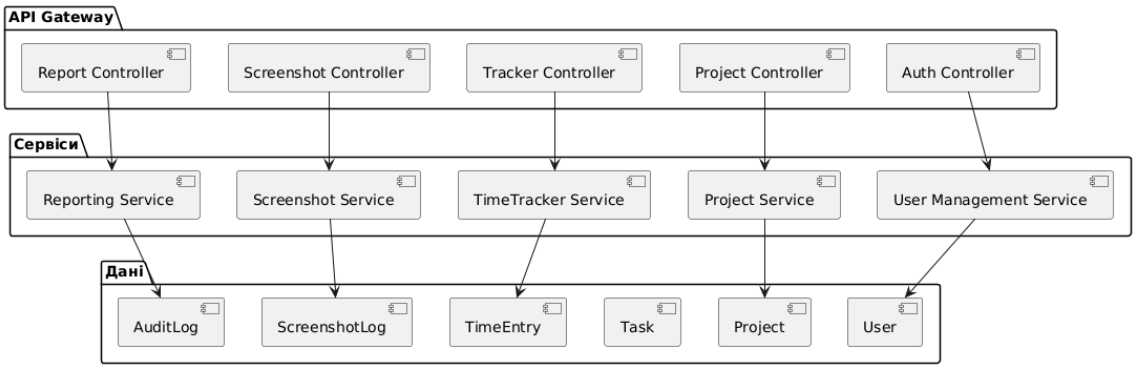

Рис. 3.2 Деталізована модульна структура серверної частини

Для забезпечення відтворюваності роботи системи, підтримки журналювання, аналізу активності користувачів та можливості віддаленого моніторингу застосовано узгоджену структуру таблиць. У таблиці 3.1 наведено ключові сутності логічної моделі та їх призначення.

Таблиця 3.1

Сутності логічної моделі даних

Сутність	Призначення
User	Зберігання облікових записів, ролей і атрибутів доступу
Project	Опис проєктів, пов’язаних замовників і структурних параметрів
Task	Зберігання переліку завдань і статусів виконання
TimeEntry	Фіксація періодів роботи, запуску або зупинки трекера
ScreenshotLog	Збереження метаданих і шляхів до створених скриншотів
Customer	Дані замовника, канали зв’язку, Telegram-ID
AuditLog	Протоколування операцій та безпекових подій
AccessPolicy	Визначення ролей, політик доступу та правил контролю

Інтеграція цих сутностей та серверних модулів формує єдину логічну конфігурацію, яка підтримує всі основні бізнес-процеси системи. На рис. 3.3 представлено узагальнену діаграму інформаційних потоків у логічній структурі ПЗ, що відображає напрямки передачі даних між рівнями.

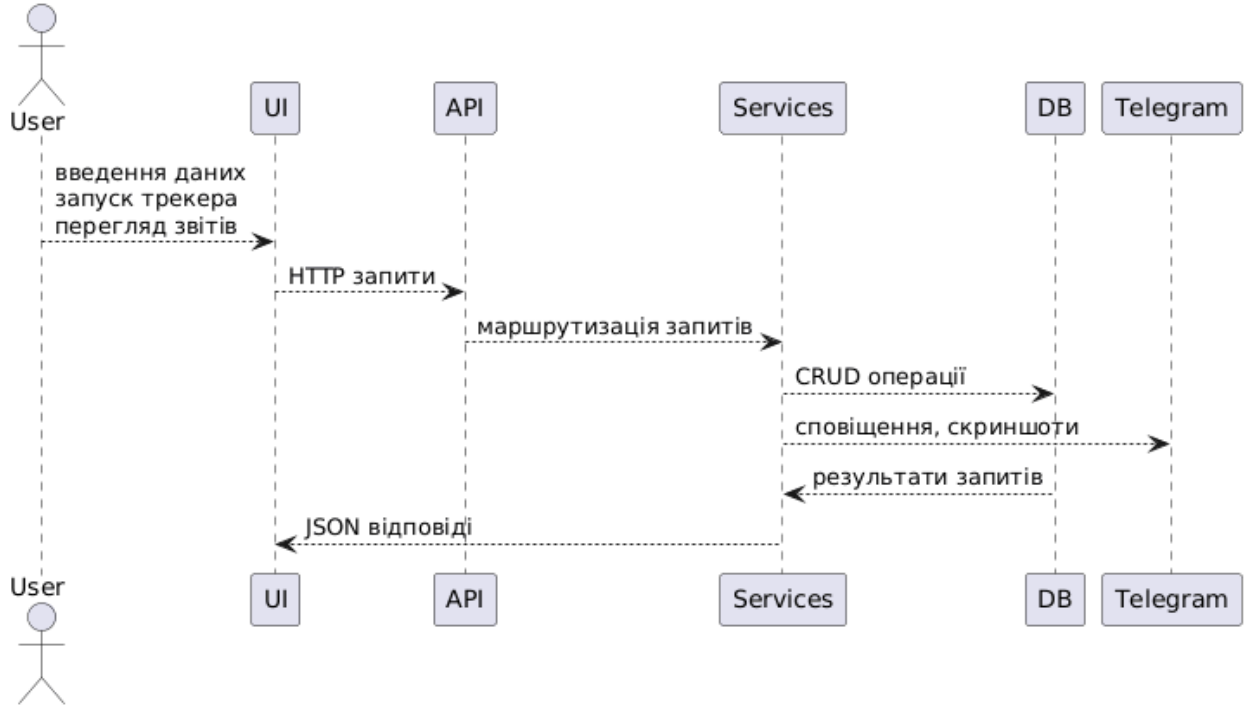

Рис. 3.3 Інформаційні потоки в логічній структурі ПЗ

Таким чином, логічна структура програмного забезпечення представлена як узгоджена система модулів, що реалізують функціональні сценарії через чітко визначені інтерфейси, використовують стандартизовані протоколи обробки даних та забезпечують збереження, відтворюваність і контрольованість усіх операцій.

3.2 Організація та реалізаціяпрограмних модулів системи

Організація програмних модулів системи ґрунтується на принципах модульності, слабкого зв’язку між компонентами та чіткої регламентації взаємодії між клієнтською частиною, серверними сервісами та системою зберігання даних. Логічна структура програмного забезпечення визначає сукупність функціональних модулів, що забезпечують реєстрацію користувачів, ведення обліку робочого часу, автоматичне формування скриншотів, обробку подій, аналітичні обчислення та комунікацію із замовником через Telegram-бот.

У межах серверної частини виділено декілька ключових сервісів, кожен з яких відповідає за окремий набір функцій. Модуль User Management реалізує операції реєстрації, авторизації, керування обліковими записами та перевірку прав доступу. Модуль Project Service забезпечує створення, перегляд і редагування проєктів, а також пов’язує їх із відповідними завданнями. Модуль Time-Tracker Service відповідає за запуск трекера, реєстрацію змін статусу, фіксацію часу активності та внесення записів до таблиць бази даних. Модуль Screenshot & Telegram Service призначений для періодичного формування скриншотів робочого екрана, передачі їх на сервер і подальшої відправки замовнику. Аналітична підсистема реалізує агрегування подій, побудову статистичних показників, формування звітів та підтримку алгоритмів інтелектуальної обробки даних. Узагальнена схема організації цих сервісів представлена на рисунку 3.4.

Організація модулів імпорту та очищення даних передбачає

використання механізму уніфікованого конвеєра, що складається з етапів: завантаження, валідації, виявлення аномалій, нормалізації та підготовки даних до подальшого аналізу.

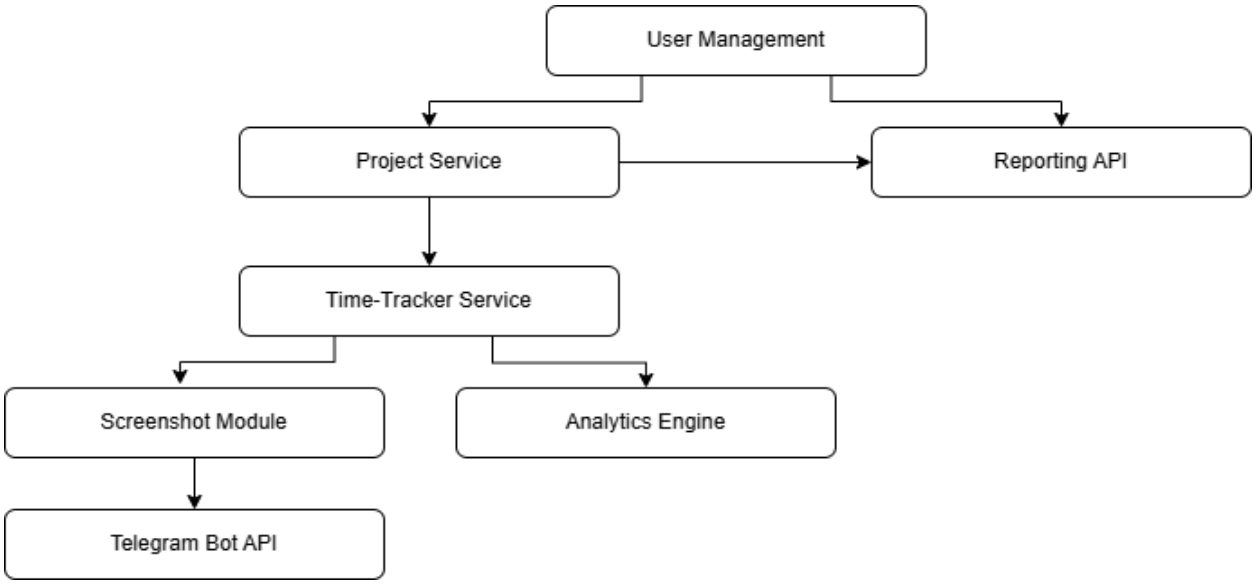

Рис. 3.4 Організаційна схема серверних та інтеграційних модулів

Модуль імпорту працює з різними форматами даних – структурованими (табличні журнали), напівструктурованими (JSON-повідомлення Telegram-бота) та автоматично згенерованими системою файлами логів.

Первинний аналіз включає виявлення дублювання подій, неузгодженостей часових міток та порушення хронології статусів. Модуль очищення застосовує методи трансформації форматів часу, усунення пропусків, стандартизації текстових полів та формування єдиного вектора ознак. Узагальнений процес роботи цього конвеєра відображено на рисунку 3.5.

Після проходження етапів очищення дані передаються до модуля інтелектуального аналізу, який забезпечує роботу моделей машинного та глибокого навчання. Реалізовані моделі застосовуються для прогнозування очікуваних трудовитрат, класифікації типів активності, виявлення аномальних патернів у роботі виконавця, а також аналізу вмісту скриншотів. У системі реалізовано як класичні моделі (логістична регресія, випадковий ліс, K-means), так і глибинні архітектури (MLP-мережа для часових рядів; CNN–модуль для

аналізу зображень). Основні характеристики моделей подано в таблиці 3.2.

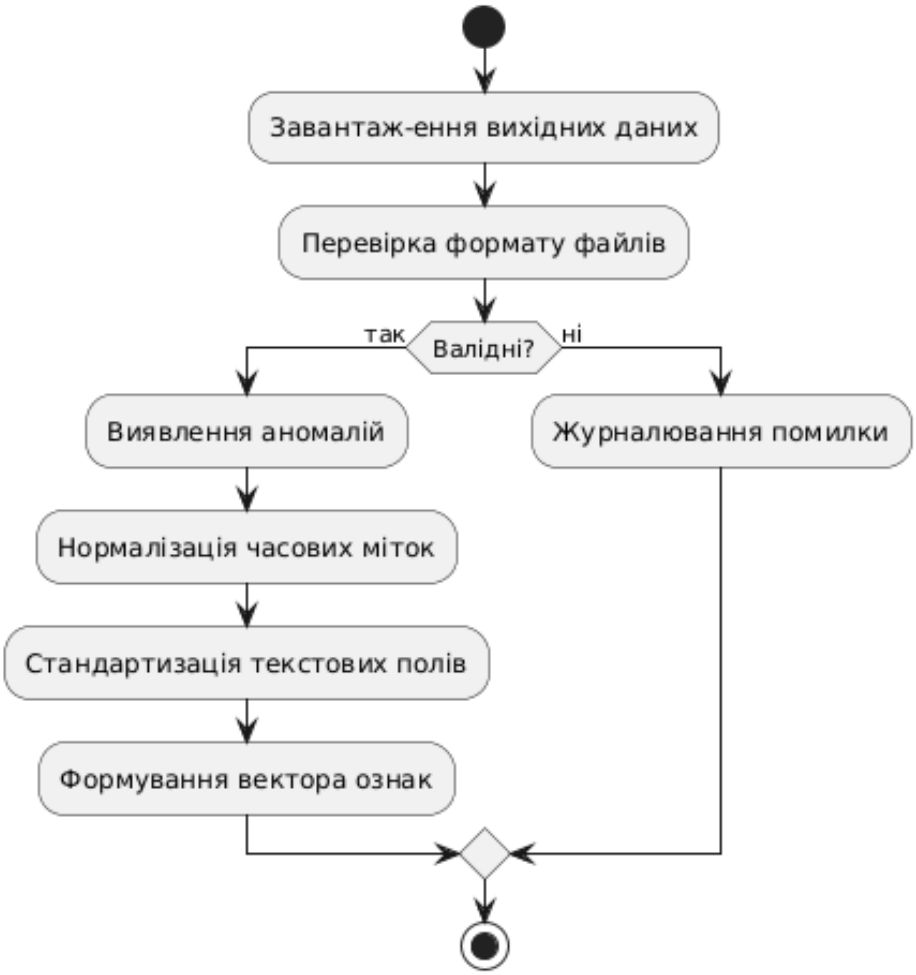

Рис. 3.5 Узагальнений конвеєр імпорту, очищення та нормалізації даних

Таблиця 3.2

Реалізовані моделі машинного та глибокого навчання

Модель	Призначення	Типи даних	Ключові параметри	Вихід
Logistic Regression	Класифікація режимів роботи	Часові та поведінкові ознаки	L2-регуляризація	Ймовірність та клас
Random Forest	Виявлення аномалій	Агреговані статуси	Кількість дерев, depth	Мітка аномальності
K-Means	Кластеризація робочих сесій	Нормалізовані ознаки	k-кластерів	Номер кластера
MLP	Прогноз завершення задачі	Часові ряди	3–5 шарів, batch size	Прогноз часу
CNN	Аналіз	Зображення	Conv2D +	Клас/скаляр

	скриншотів		MaxPooling	активності

Архітектура ML-ядра передбачає поділ на три логічні шари: Data Layer, Model Layer та Serving Layer. Data Layer відповідає за підготовку та трансформацію даних; Model Layer – за навчання, збереження та оновлення моделей; Serving Layer – за інтеграцію результатів із REST-сервісами та оновлення аналітичних панелей користувача. Структуру ML-ядра наведено на рисунку 3.6.

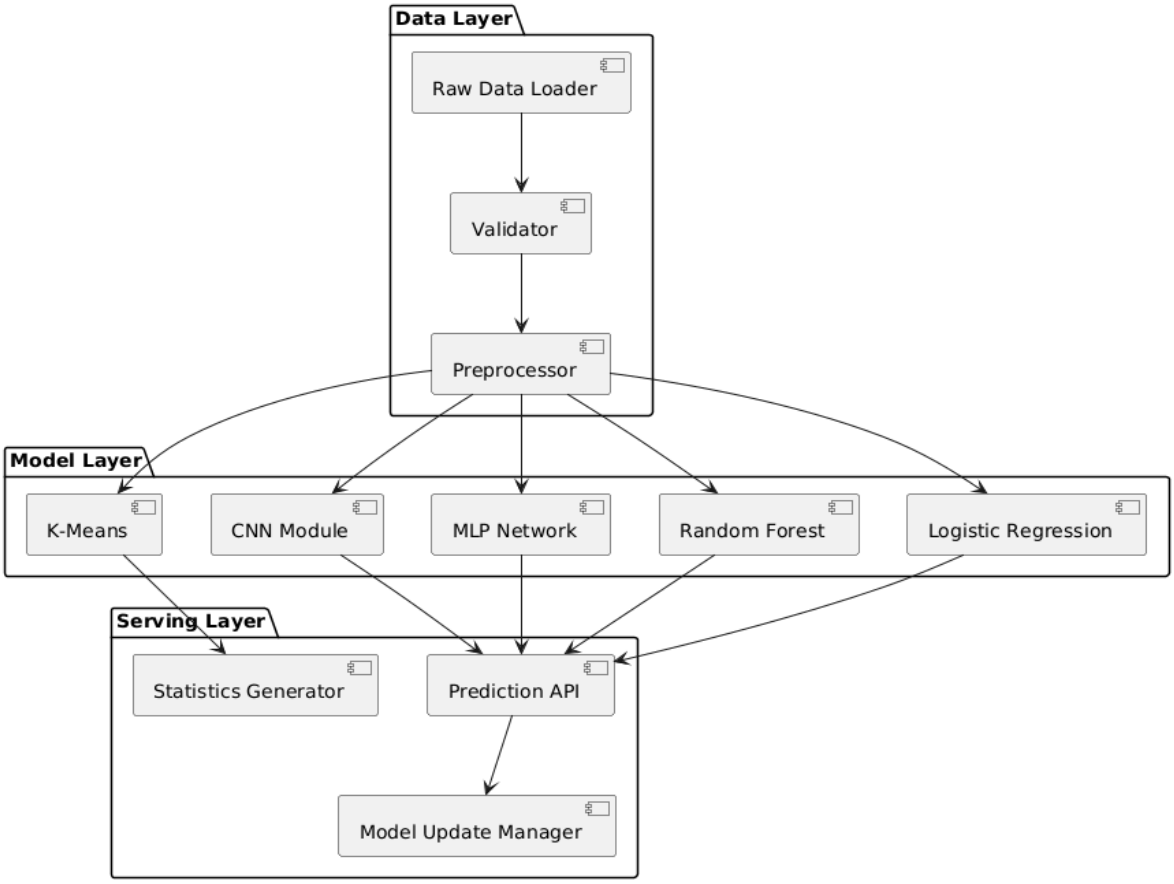

Рис. 3.6 Архітектура ML-ядра та модулів машинного і глибокого навчання

Послідовність взаємодії клієнтського застосунку, серверної частини, Telegram-бота, бази даних і модулів машинного навчання під час обробки подій трекера подано на рисунку 3.7. У процесі створення завдання інформація зберігається у базі даних, після чого модуль трекінгу ініціює роботу планувальника скриншотів, модуль очищення формує вектор ознак, а моделі класифікації визначають характер активності. Для замовника надсилаються як

скриншоти, так і агреговані статистичні показники, що формуються аналітичною підсистемою.

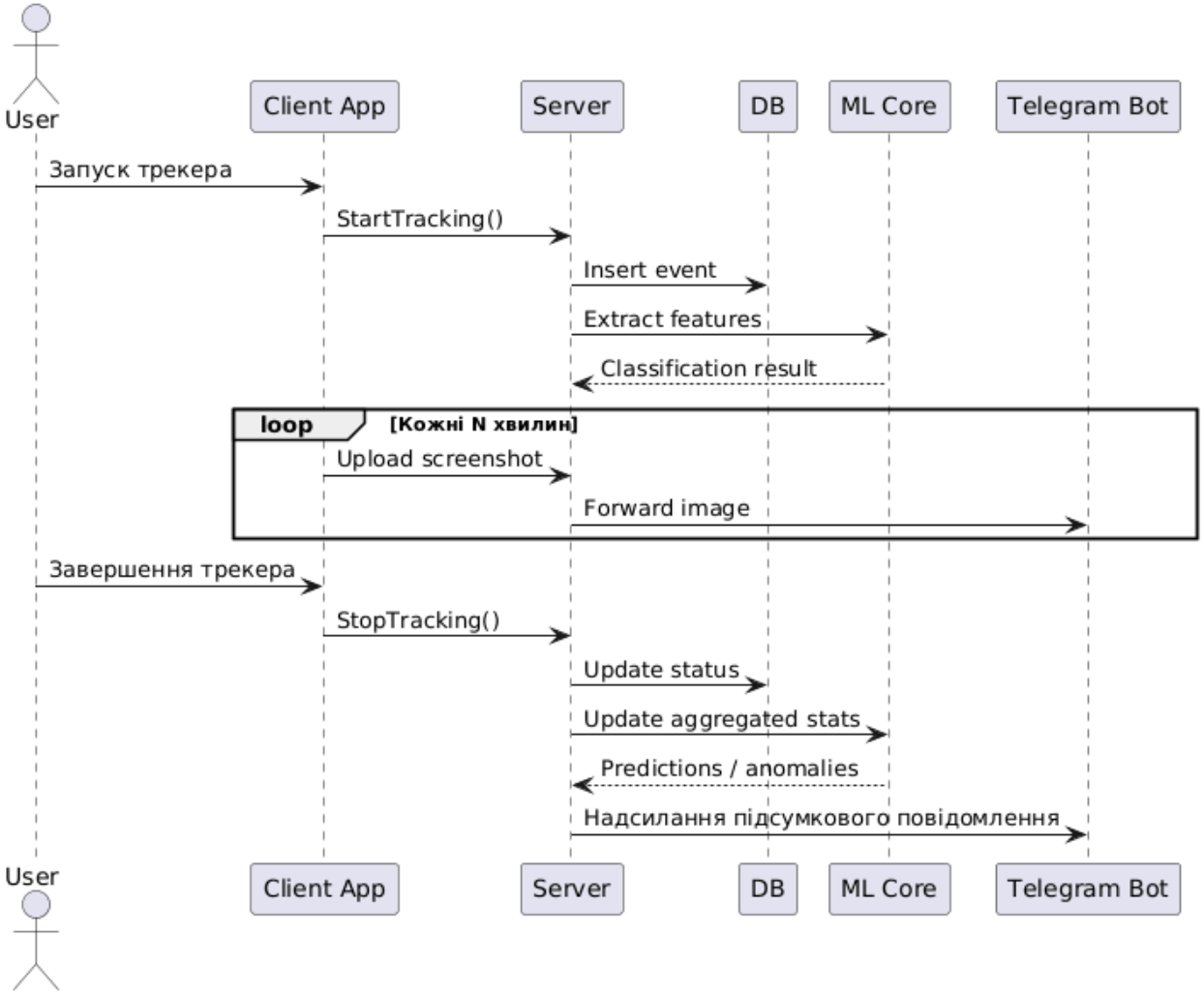

Рис. 3.7 Послідовність взаємодії модулів трекера, Telegram-бота, бази даних і ML-ядра

Завершальним елементом реалізації є організація централізованого механізму зберігання проміжних і фінальних моделей, їх оновлення та повторного навчання. Система підтримує інкрементальне навчання, що дозволяє безперервно удосконалювати якість прогнозів на основі нових даних, не порушуючи роботи користувачів. На рисунку 3.8 наведено узагальнений цикл оновлення моделей.

Таким чином, організація програмних модулів системи забезпечує цілісність обробки даних, узгодженість між сервісами, масштабованість і можливість подальшого розширення функціональності.

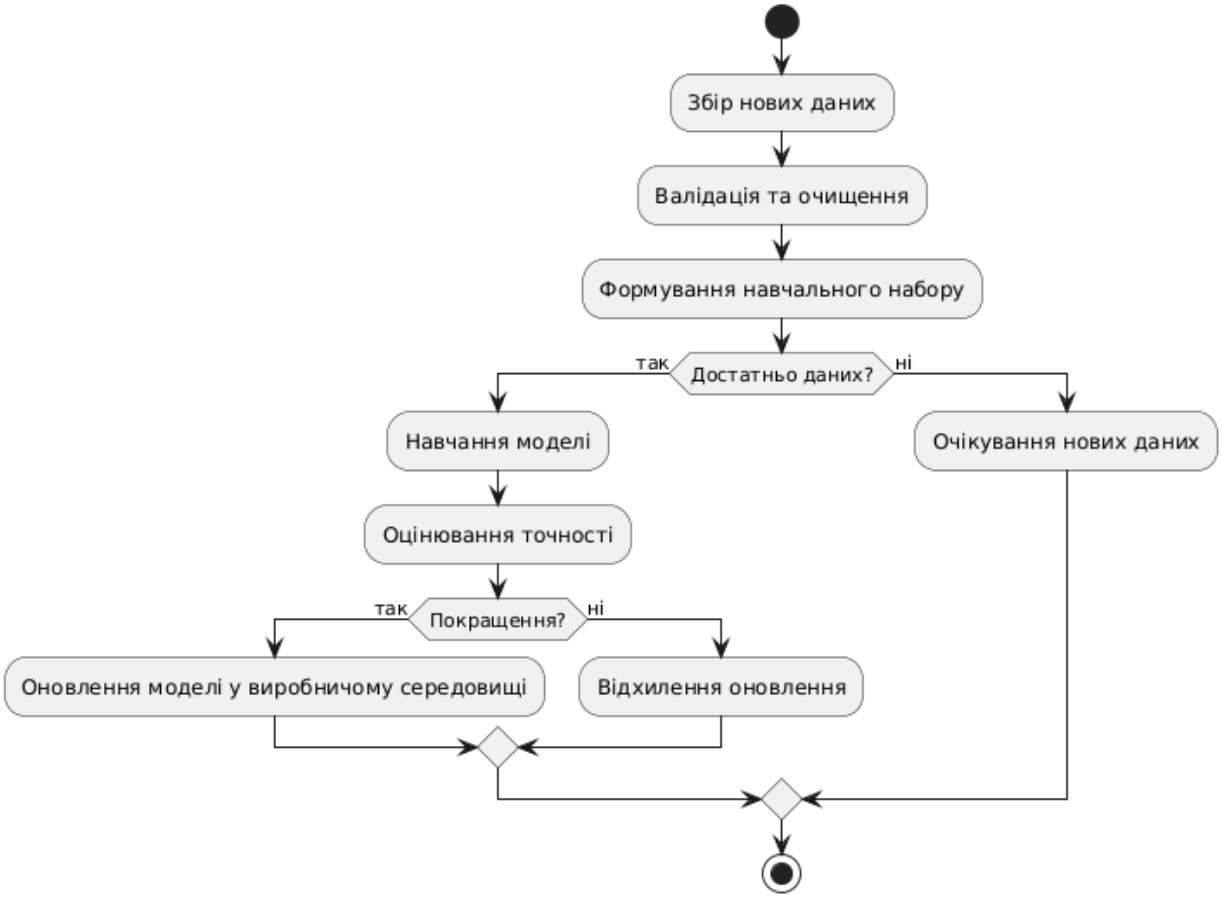

Рис. 3.8 Узагальнений алгоритм оновлення та повторного навчання моделей

Інтеграція аналітичних компонентів із модулем трекінгу та Telegram-ботом створює підґрунтя для формування інтелектуальної платформи підтримки управлінських рішень, орієнтованої на підвищення прозорості, ефективності та прогнозованості виконання проєктів.

3.3. Взаємодія з користувачем та робота з системою

Взаємодія користувача із системою здійснюється через інтуїтивно зрозумілий веб-інтерфейс, який забезпечує доступ до повного циклу операцій – від створення облікового запису та формування проєкту до запуску трекера, перегляду активних завдань і генерації інвойсів. Архітектура інтерфейсу орієнтована на мінімізацію кількості дій користувача, забезпечення прозорої навігації та швидкої обробки подій, що підтверджено структурою сторінок і

функціональними можливостями, відображеними на рисунках 3.9–3.16.

Головна сторінка веб-застосунку представлена на рисунку 3.9. У неавторизованому режимі користувачу доступні дві основні кнопки: «Sign in», що відкриває форму входу, та «Sign up», яка переводить на сторінку створення нового облікового запису.

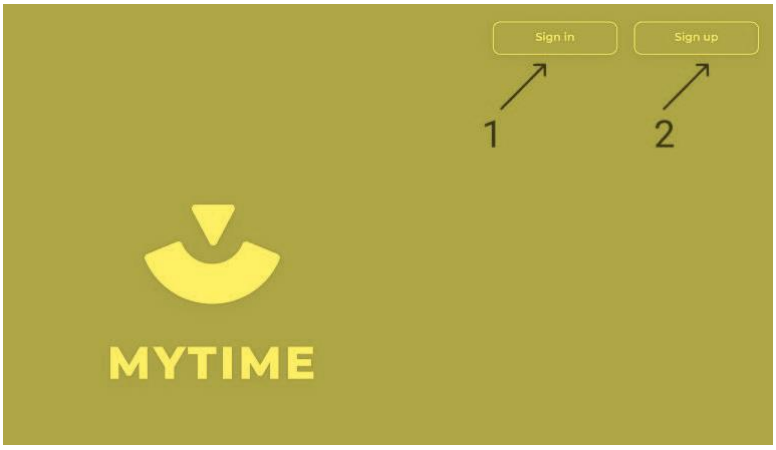

Рис. 3.9 Головна сторінка застосунку

Перехід до реєстрації ініціюється натисканням відповідної кнопки, після чого відкривається сторінка, структура якої зображена на рисунку 3.10. Реєстраційна форма містить поля для введення імені користувача, електронної пошти, пароля та підтвердження пароля, а також кнопку підтвердження реєстрації. У разі наявності облікового запису доступний перехід до сторінки авторизації.

Рис. 3.10 Сторінка реєстрації

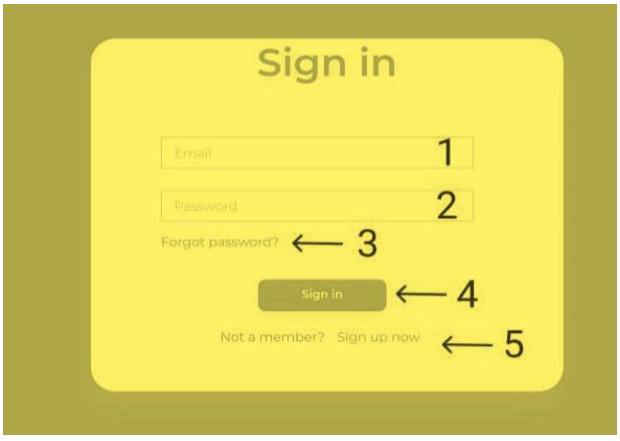

Рис. 3.11 Сторінка авторизації

На рисунку 3.11 представлено інтерфейс авторизації, що передбачає

введення електронної пошти та пароля. Додатково передбачено опцію «Forgot password», що забезпечує відновлення доступу, а також можливість переходу до форми реєстрації.

Після успішної авторизації користувач автоматично переводиться на сторінку створення першого проєкту (рис. 3.12), якщо активних проєктів ще не існує.

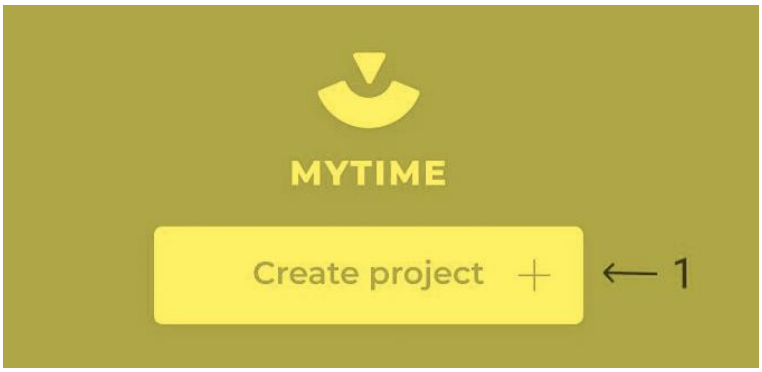

Рис. 3.12 Сторінка додавання проєкту

Інтерфейс створення проєкту (рис. 3.13) включає поле введення назви, текстовий опис поточної активності та елемент керування для вмикання трекера. Після збереження даних система формує повноцінну сторінку проєкту, що відображена на рисунку 3.14.

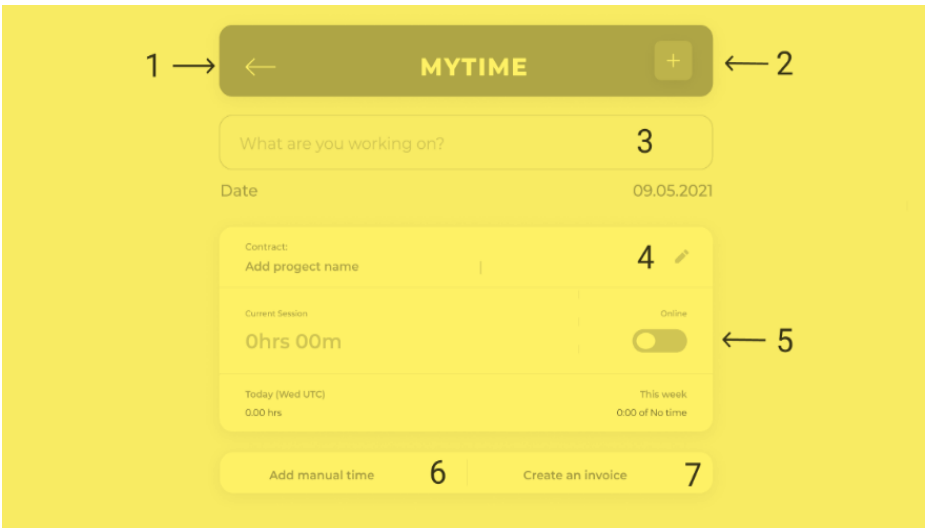

Рис. 3.13 Сторінка створення проєкту

Тут користувач має можливість запускати та зупиняти трекер, додавати час вручну, створювати інвойси для замовника, а також переглядати

статистику активності поточного дня та тижня.

Рис. 3.14 Сторінка створеного проєкту з включеним трекером

Сторінка з переліком активних проєктів (рис. 3.15) надає навігацію по всім проєктам користувача, можливість швидкого створення нового проєкту та пошуку за ім’ям клієнта чи назвою контракту.

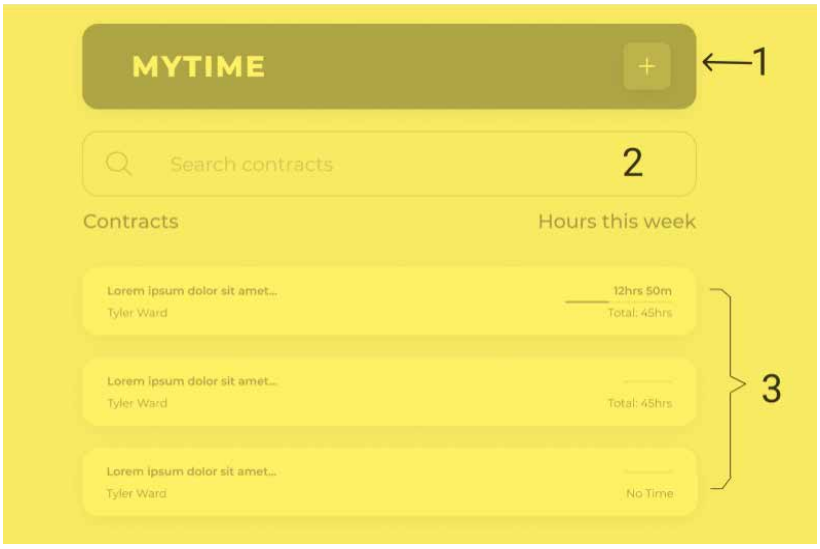

Рис. 3.15 Сторінка з переліком активних проєктів

Перехід до вибраного проєкту відкриває детальний список усіх виконаних завдань і коментарів (рис. 3.16). Тут зосереджені дані, які формуються кожного разу під час активації трекера: опис роботи, дата, тривалість та часові мітки сесій.

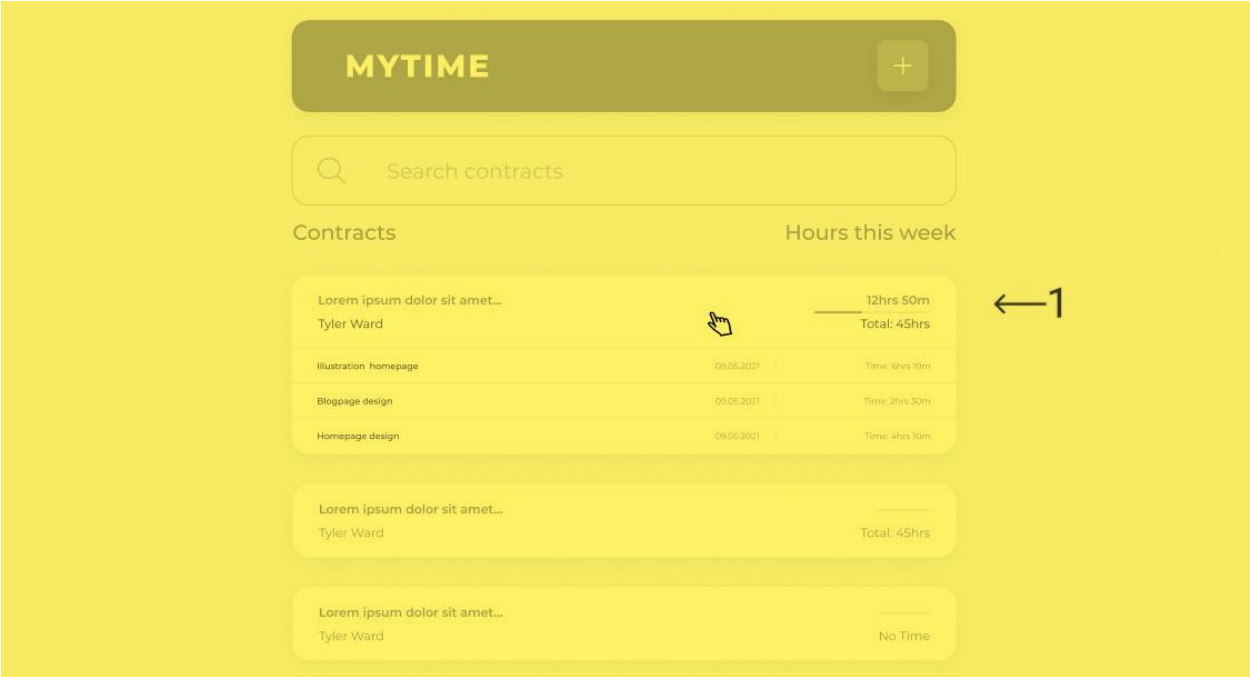

Рис. 3.16 Список коментування завдань вибраного проєкта

Для початку роботи трекера користувачу необхідно активувати перемикач на сторінці проєкту. Після цього Telegram-бот ініціює налаштування каналу комунікації, запитуючи адресу електронної пошти розробника та забезпечуючи пересилання скриншотів замовнику. Такий механізм забезпечує прозорість і контроль виконання робіт у реальному часі, доповнюючи статистичні дані автоматично сформованими візуальними доказами активності.

Оцінка коректності роботи інтерфейсу та функціоналу здійснювалась шляхом проведення комплексних випробувань відповідно до вимог стандартів ДСТУ 34.603-92 та ДСТУ 50-34.698-90. Перевірялися навігація, доступність основних функцій, коректність обробки подій, відповідність повідомлень сценаріям взаємодії, а також стабільність роботи трекера під час довготривалих сесій.

3.5. Тестування системи та завантаження моделей

Тестування розробленої інформаційної системи проводилося з метою перевірки відповідності реалізованого функціоналу вимогам технічного

завдання, оцінювання стабільності роботи сервісів та коректності обробки подій, що виникають у процесі взаємодії користувача з системою. Особливу увагу приділено механізмам авторизації й реєстрації, створенню та супроводженню проєктів, працездатності трекеру робочого часу, коректності формування статистики, роботі Telegram-бота, а також модулю інтеграції даних для моделей машинного навчання. Усі тести виконувалися відповідно до вимог стандартів ДСТУ 34.603-92 та ДСТУ 50-34.698-90, що регламентують порядок проведення випробувань автоматизованих систем та структуру супровідної документації.

Процес тестування охоплював як ручні, так і автоматизовані сценарії, спрямовані на перевірку поведінки системи у штатних, граничних і виняткових ситуаціях. Для цього сформовано послідовність тест-кейсів, що охоплюють усі ключові функціональні компоненти: інтерфейсне введення та відображення інформації, навігацію між сторінками, запуск і зупинку трекеру, формування звітності, передачу даних до Telegram-бота та обробку результатів. У таблиці дод.А наведено результати виконання тестів за узгодженою структурою, що включає назву тесту, передумови, схему його проведення, очікуваний результат та порівняння фактичної поведінки системи з очікуваною.

Окрім функціонального тестування, проведено перевірку коректності роботи модулів машинного навчання, що включала тестування завантаження моделей зі сховища, перевірку їх цілісності, відповідності версій, а також правильності десеріалізації. Під час тестів моделі завантажувалися у робоче середовище сервера (Serving Layer), виконували контрольні інференси на тестових вибірках і порівнювалися з еталонними результатами, сформованими у процесі навчання. Було підтверджено, що механізм завантаження моделей коректно обробляє ситуації заміни версії, відсутності файлу моделі чи пошкодження структури, а система логування фіксує всі критичні події.

Таким чином, результати тестування підтвердили коректність роботи інформаційної системи, стабільність її основних підсистем, а також

відповідність функціоналу вимогам технічного завдання. Реалізовані компоненти забезпечують надійну взаємодію між клієнтським застосунком, серверною частиною, Telegram-ботом і модулями машинного навчання, що свідчить про готовність системи до експлуатації в умовах реального використання.

3.6. Висновки до 3 розділу

У третьому розділі детально реалізовано технічну архітектуру інформаційної системи обліку та аналізу робочого часу. Сформовано логічну структуру програмних модулів, визначено їх функціональні ролі та описано технологічні засоби, що забезпечують роботу клієнтської частини, серверних сервісів, механізмів обміну даними та взаємодію з Telegram-ботом. Наведені UML-діаграми, структурні схеми та фрагменти інтерфейсу демонструють узгодженість обраних технічних рішень, а також підтримку принципів модульності, масштабованості та ефективної організації інформаційних потоків.

Розділ також містить опис процесів взаємодії користувача із системою, включно зі сценаріями реєстрації, авторизації, створення та ведення проєктів, запуском трекера й отриманням статистичної інформації. Проведене тестування, результати якого винесено до Додатків, підтвердило коректність реалізації основних функцій, стабільність роботи модулів і відповідність поведінки системи встановленим функціональним вимогам. Це засвідчує готовність програмного продукту до практичного використання та подальшого розширення його функціональних можливостей.

4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Методика формування вибірок, передобробки даних та валідації моделей

Експериментальне дослідження системи аналізу, контролю та обліку роботи співробітників базується на формалізованій процедурі підготовки даних, що забезпечує їх цілісність, однорідність та репрезентативність для побудови моделей машинного навчання. У межах роботи було використано комплекс корпоративних даних, сформований у процесі функціонування вебзастосунку та Telegram-бота. До складу вибірки входять часові сесії активності, журнали подій трекера, коментарі виконавців, метадані скріншотів, а також агреговані KPI-показники, що відображають інтенсивність і структуру робочої діяльності. Первинні записи описуються сутностями, визначеними архітектурою системи у розділі 2, зокрема User, Project, TrackerEvent, ScreenshotMeta та Session. Базовий елемент – робоча сесія – сформовано у вигляді кортежу

$$Session = \{t_{start}, t_{end}, ProjectID, UserID\}, \quad (4.1)$$

що забезпечує можливість відтворення часової динаміки роботи виконавця.

Передобробка даних включала кілька послідовних етапів. На першому етапі виконувалося очищення вибірки від дублікатів, що визначалися за тотожністю часових меж сесії та ідентифікаторів користувача. Далі відбувалася фільтрація шумів: усувалися аномально короткі (менше 1 хвилини) та надмірно тривалі (понад 12 годин) сесії, а випадкові розриви, спричинені нестабільністю мережі, об’єднувалися у суцільні інтервали за умови $\Delta t < 2$ хв.

Після цього виконувалася нормалізація часових інтервалів на основі стандартизації

$$z = \frac{x-\mu}{\sigma}, \quad (4.2)$$

а також перевірка вибірки на наявність аномальних значень за міжквартильним критерієм $x \in [Q_1 - 1.5IQR,\ Q_3 + 1.5IQR]$.

Пропущені дані становили менше 1% і заповнювалися методом forward fill для часових ознак та найчастотнішим значенням для категорійних.

З метою подальшого моделювання сформовано систему ознак, що включає темпоральні параметри (тривалість сесії, час початку, день тижня), поведінкові характеристики (частота зміни активних вікон, частка активного часу), проєктні атрибути та векторизовані подієві коментарі виконавця, до яких застосовано TF-IDF перетворення

$$tfidf_{t,d} = tf_{t,d} \cdot log\frac{N}{df_t}. \quad (4.3)$$

Інтегрована статистична характеристика фінальної вибірки подана у таблиці 4.1.

Таблиця 4.1

Описові статистики сформованої вибірки

Ознака	Min	Max	Mean	Std	Коментар
Тривалість сесії, хв	3	238	57.3	29.4	Основний індикатор продуктивності
Кількість сесій/день	1	18	6.1	3.2	Характеризує ритм роботи
Частка активного часу	0.35	0.97	0.74	0.11	Оцінка інтенсивності діяльності
Коментарі TF-IDF (розмірність)	–	–	120	–	Векторизовані текстові описи

Статистичний аналіз продемонстрував, що розподіл тривалості сесій має виражену правосторонню асиметрію, притаманну для продуктивних фаз роботи. Оцінка асиметрії для цієї ознаки становить $\gamma_1 = 1.12$, що підтверджує наявність коротких стандартних сесій та довгих періодів зосередженої діяльності. Гістограмма тривалості сесій наведена на рис. 4.1, де спостерігається характерний спад частоти зі збільшенням тривалості, що відповідає гамма-подібному розподілу.

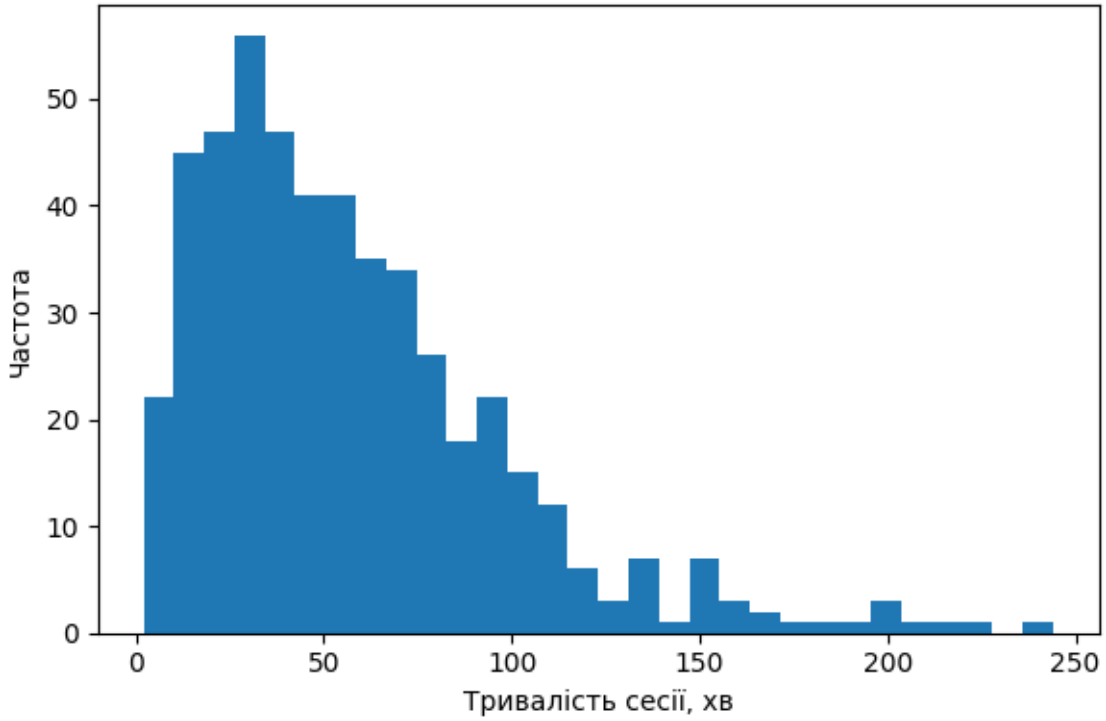

Рис. 4.1 Розподіл тривалості робочих сесій (гістограма)

Після завершення етапу інжинірингу ознак виконано формування вибірок для навчання та оцінювання моделей. Для забезпечення репрезентативності та запобігання витоку інформації дані поділено у співвідношенні 70/15/15 на навчальну, валідаційну та тестову вибірки відповідно. Стратифікація здійснювалася за користувачами, проєктами та категоріями коментарів, що дозволило уникнути змішування патернів роботи різних виконавців. Для часових даних використано rolling-validation, коли модель навчається на інтервалі (t_0, t_1)та перевіряється на сегменті (t_1, t_2), що забезпечує відповідність реальному сценарію розгортання системи.

Для узагальнення виконаних етапів формування вибірок, очищення та перетворення даних, а також процедури їх подальшої підготовки до моделювання було побудовано підсумкову схему методики. Вона відображає повний технологічний цикл обробки даних – від інтеграції подієвих журналів системи до формування узгодженої матриці ознак, що використовується під час навчання моделей. Така схема забезпечує наочність, підкреслює логічну послідовність виконаних операцій і дозволяє легко відтворити весь процес у

подальших експериментальних дослідженнях (рис. 4.2).

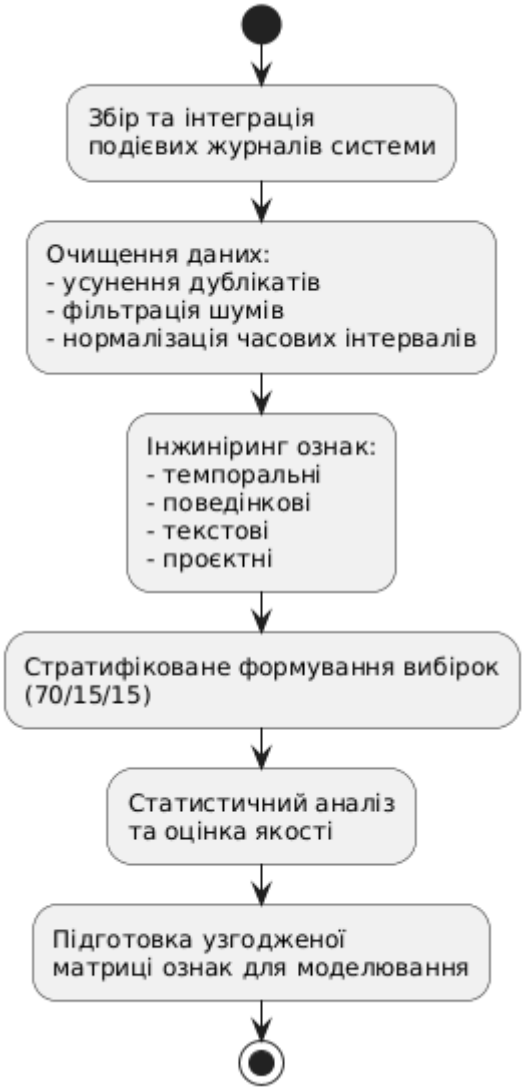

Рис. 4.2 Підсумкова схема методики підготовки даних

Узагальнюючи викладене, методика формування та передобробки даних забезпечує створення структурованої, узгодженої та статистично стабільної вибірки, придатної для побудови моделей оцінювання продуктивності, класифікації активності та виявлення закономірностей у поведінці користувачів.

4.2 Порівняльний аналіз моделей машинного навчання

Після формування та передобробки вибірок виникла потреба у виборі оптимальної моделі машинного навчання, здатної забезпечити стійке та точне

прогнозування показників продуктивності персоналу в умовах реальних коливань робочих сесій, різної інтенсивності активності та відмінностей у поведінці виконавців. Порівняльний аналіз моделей став центральним етапом експериментального дослідження, оскільки від правильного вибору алгоритму залежить як якість оцінювання ефективності персоналу, так і стабільність системи при масштабуванні.

Для забезпечення коректності порівняння всі моделі навчалися на одній і тій самій узгодженій вибірці, сформованій відповідно до методики, описаної вище. Такий підхід унеможливлює викривлення результатів і гарантує, що різниця у якості моделювання зумовлена виключно властивостями алгоритмів, а не структурою даних. До аналізу було включено набір класичних моделей машинного навчання, які не використовують глибинні нейронні мережі, проте забезпечують високу інтерпретованість та достатню точність: Decision Tree Classifier, Random Forest, k-Nearest Neighbours, Logistic Regression, Naive Bayes, а також градієнтний бустинг (XGBoost) як найпотужнішу модель у групі деревоподібних алгоритмів.

Вибір моделей обґрунтований різними підходами до обробки інформації: дерево рішень дозволяє інтерпретувати правила прийняття рішень; випадковий ліс та XGBoost формують ансамблі моделей, що знижує дисперсію; логістична регресія використовує лінійні гіперплощини; алгоритм k-NN ґрунтується на структурі простору ознак; наївний Байєс ефективний для текстових та категорійних ознак за умов припущення незалежності. Гіперпараметри моделей підбиралися з урахуванням рекомендацій літератури та характеристик вибірки: для Random Forest визначалися кількість дерев $n\_estimators$, максимальна глибина дерева max_depth; для k-NN оптимальне число сусідів kобиралося шляхом мінімізації середньої помилки; для XGBoost налаштовувалися параметри $learning\_rate$, max_depthта кількість бустингових ітерацій.

Оцінювання якості проводилося з використанням набору метрик, релевантних для задач як класифікації, так і регресії. Для класифікаційних

моделей розраховувалися

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN},$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

Для регресійних моделей використовувалися

$$MAE = \frac{1}{n}\sum | y_i - \hat{y}_i |, RMSE = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2},$$

$$MAPE = \frac{100\%}{n}\sum | \frac{y_i - \hat{y}_i}{y_i} |, R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

Метою аналізу було виявлення моделі, яка найточніше оцінює продуктивність користувачів і демонструє високу стабільність при зміні структури робочих сесій, інтенсивності активності та особливостей поведінкових ознак. Це критично важливо для бізнес-контексту системи, оскільки в реальних умовах показники роботи можуть коливатися, а отже модель повинна залишатися надійною навіть при нетипових патернах.

На основі тестування отримано результати, узагальнені в таблиці 4.2.

Таблиця 4.2

Порівняння точності моделей за метрикою Accuracy

Модель	Accuracy
Decision Tree	0.78
Random Forest	0.85
k-NN	0.74
Logistic Regression	0.81
Naive Bayes	0.69
XGBoost	0.88

Для наочного порівняння побудовано стовпчикову діаграму (рис. 4.3), яка показує відносну якість моделей. Найкращі показники продемонстрував алгоритм XGBoost (0.88), що очікувано завдяки його здатності працювати з неідеальними, шумними та змішаними за структурою даними.

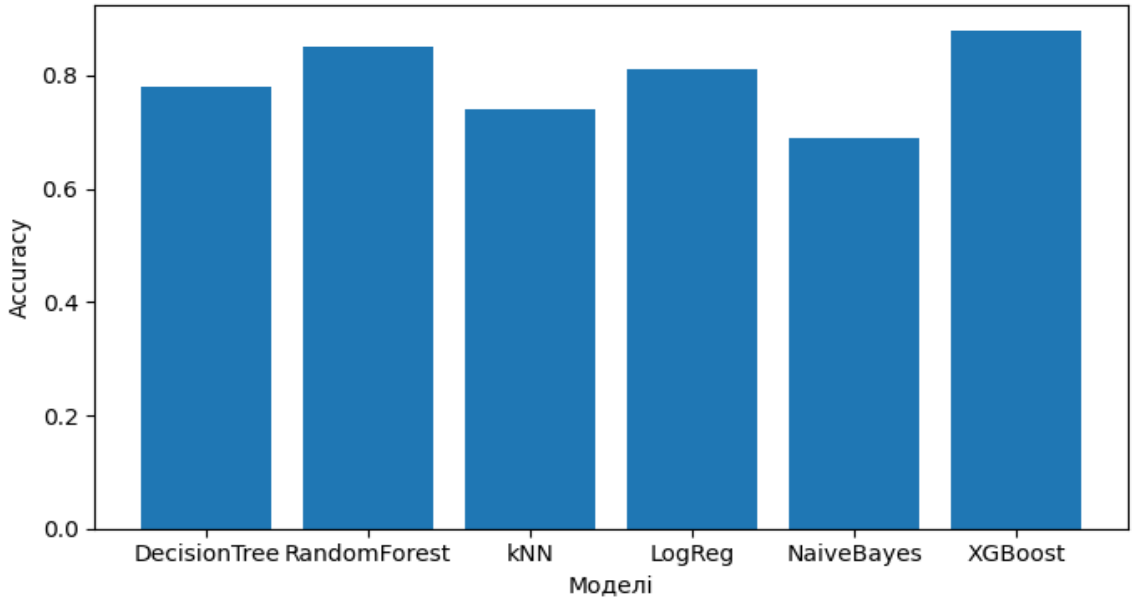

Рис. 4.3 Порівняльна діаграма якості моделей

Друге місце посів Random Forest з показником 0.85, що підтверджує ефективність ансамблевих методів. Моделі Logistic Regression та Decision Tree продемонстрували збалансоване співвідношення точності та інтерпретованості. Алгоритм Naive Bayes показав найнижчу якість, що пояснюється сильними залежностями між ознаками, які порушують його базові припущення.

Таким чином, результати показують, що ансамблеві алгоритми (Random Forest і XGBoost) найкраще підходять для моделювання продуктивності персоналу в умовах змішаних типів даних. Вони забезпечують високу точність, низьку чутливість до шумів, стабільність при зміні структури вибірки та ефективність при великій кількості ознак, побудованих у процесі інжинірингу. Моделі логістичної регресії та дерева рішень можуть використовуватися як інтерпретовані базові рішення, тоді як k-NN та Naive Bayes доцільно застосовувати лише для порівняння або у вузьких спеціалізованих сценаріях. Підсумкові висновки підтверджують, що XGBoost є найбільш надійним кандидатам для подальшого моделювання, зокрема в підрозділах аналізу та прогнозування продуктивності.

4.3 Кластеризаційний аналіз та виявлення структур у даних

Кластеризаційний аналіз у межах досліджуваної системи відіграє ключову роль, оскільки дозволяє автоматично виявляти приховані групи користувачів із подібними робочими патернами, рівнем продуктивності, стабільністю активності та особливостями поведінки під час виконання завдань. На відміну від моделей машинного навчання, спрямованих на прогнозування або класифікацію, кластеризація не потребує наявності розмічених даних, що забезпечує її ефективність у реальних робочих середовищах, де оцінка продуктивності часто є контекстною, а поведінка працівників змінюється залежно від типу проєкту, складності завдань чи графіку роботи. Використання кластеризаційних алгоритмів дозволяє побачити структуру даних, сформувати профілі працівників, виявити групи ризику та типові сценарії робочих сесій, що забезпечує вагому підтримку прийняття рішень у HR-аналітиці.

Для аналізу було сформовано набір ознак, що найповніше описують поведінку виконавців: середня тривалість сесії, варіативність ритму роботи, частка активного часу, інтенсивність подій, стабільність протягом дня та співвідношення активних і пасивних періодів. Дані були стандартизовані та приведені до узгодженого вигляду, що дозволило застосувати кілька типів кластеризаційних алгоритмів. Як базовий метод обрано k-Means завдяки його інтерпретованості та здатності формувати компактні та добре поділені групи. Для виявлення аномалій та нестандартних патернів роботи застосовано DBSCAN, що ефективний для виділення шумових спостережень, зокрема сесій із різкими провалами активності. Для аналізу ієрархічних відношень між групами використано Hierarchical Clustering, демонстрований через побудову дендрограм.

Кластеризаційний аналіз у межах досліджуваної системи відіграє ключову роль, оскільки дозволяє автоматично виявляти приховані групи користувачів із подібними робочими патернами, рівнем продуктивності,

стабільністю активності та особливостями поведінки під час виконання завдань. На відміну від моделей машинного навчання, спрямованих на прогнозування або класифікацію, кластеризація не потребує наявності розмічених даних, що забезпечує її ефективність у реальних робочих середовищах, де оцінка продуктивності часто є контекстною, а поведінка працівників змінюється залежно від типу проєкту, складності завдань чи графіку роботи. Використання кластеризаційних алгоритмів дозволяє побачити структуру даних, сформувати профілі працівників, виявити групи ризику та типові сценарії робочих сесій, що забезпечує вагому підтримку прийняття рішень у HR-аналітиці.

Для аналізу було сформовано набір ознак, що найповніше описують поведінку виконавців: середня тривалість сесії, варіативність ритму роботи, частка активного часу, інтенсивність подій, стабільність протягом дня та співвідношення активних і пасивних періодів. Дані були стандартизовані та приведені до узгодженого вигляду, що дозволило застосувати кілька типів кластеризаційних алгоритмів. Як базовий метод обрано *k-Means* завдяки його інтерпретованості та здатності формувати компактні та добре поділені групи. Для виявлення аномалій та нестандартних патернів роботи застосовано *DBSCAN*, що ефективний для виділення шумових спостережень, зокрема сесій із різкими провалами активності. Для аналізу ієрархічних відношень між групами використано Hierarchical Clustering, демонстрований через побудову дендрограм.

Оскільки алгоритм k-Means потребує встановлення кількості кластерів k, оптимальне значення визначалося двома методами: методом «лікоть» та індексом силуету. Метод «лікоть» аналізує криву зміни внутрішньокластерної інерції

$$W(k) = \sum_{i=1}^{k} \sum_{x \in C_i} \| x - \mu_i \|^2,$$

та знаходить точку, у якій покращення якості починає зменшуватися – ця точка відповідає оптимальному k. У межах вибірки мінімізація інерції

демонструє виражений «злам» при $k = 4$, що підтверджується значенням середнього коефіцієнта силуету

$$s = \frac{b - a}{max(a, b)},$$

де a – середня внутрішньокластерна відстань, а b – найменша відстань до іншого кластера. Отримане значення $s = 0.61$ свідчить про добре відокремлену структуру кластерів.

На рисунку 4.4 подано двовимірну PCA-візуалізацію результатів кластеризації, яка відображає чотири стабільні групи користувачів. Видимість чітких меж між групами підтверджує здатність алгоритму k-Means ефективно структурувати дані.

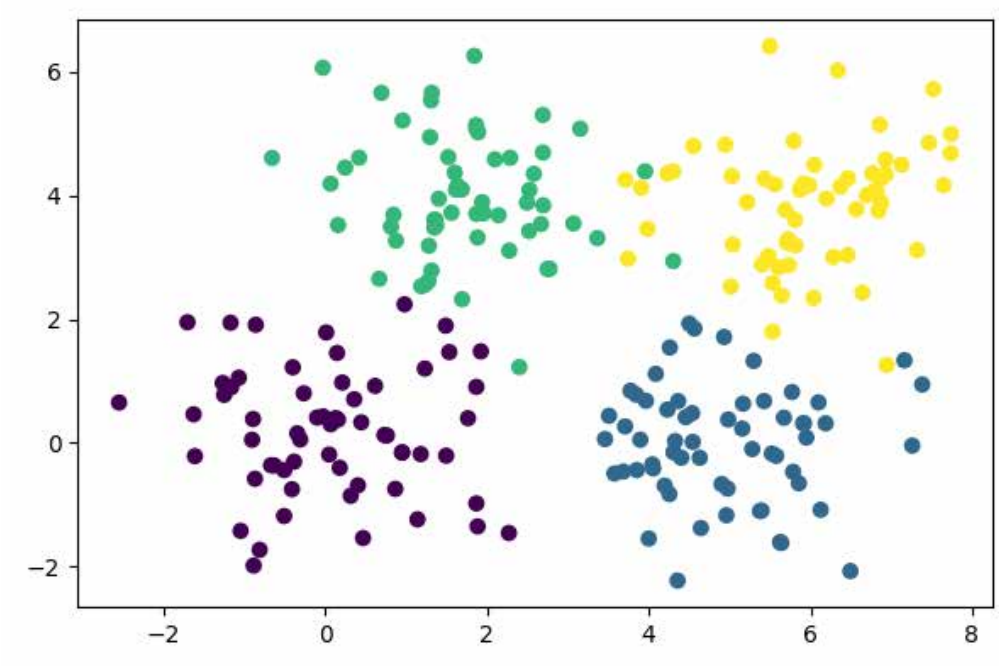

Рис. 4.4 Візуалізація кластеризації (PCA-простір)

Рисунок 4.5 демонструє криву «лікоть», що використана для визначення оптимальної кількості кластерів, і чітко вказує на $k = 4$ як найкраще значення. Після застосування кластеризації виконано інтерпретацію отриманих груп. Перший кластер представлений працівниками з високою продуктивністю, стабільним графіком і короткими перервами – це «високопродуктивні» виконавці.

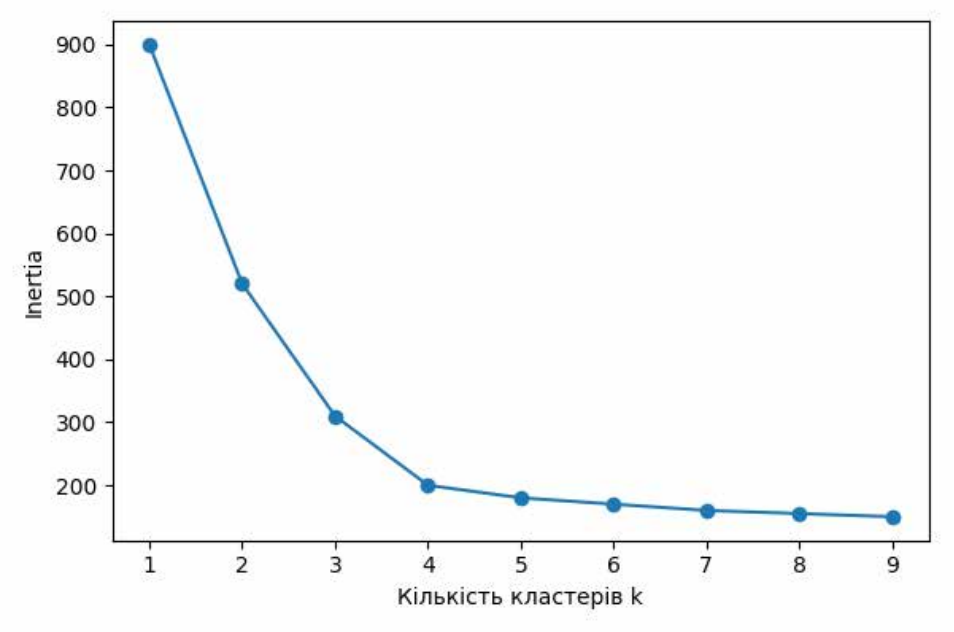

Рис. 4.5 Метод «лікоть» для визначення кількості кластерів

Другий кластер характеризується нерівномірним робочим ритмом і значними коливаннями тривалості сесій, що відповідає групі «нестабільних». Третій кластер складається з користувачів із частими періодами низької активності, довгими пасивними відрізками та вираженими провалами продуктивності; його можна інтерпретувати як «ризиковий».

Четвертий кластер об’єднує працівників із середньою стабільністю, типовими навантаженнями та відсутністю критичних аномалій – це група «регулярні середні». Узагальнені характеристики наведено в таблиці 4.3.

Таблиця 4.3

Характеристики виділених кластерів

Кластер	Середня тривалість сесії	Частка активного часу	Варіативність ритму	Інтерпретація
1	Низька	Висока	Низька	Високопродуктивні
2	Середня	Середня	Висока	Нестабільні
3	Низька	Низька	Висока	Ризикові
4	Середня	Середня	Низька	Регулярні середні

Таким чином, кластеризаційний аналіз дав змогу сформувати глибоке розуміння структури робочої поведінки користувачів системи. Отримані

групи можуть надалі використовуватися для персоналізованого моніторингу, прогнозування ризиків, підвищення якості HR-аналітики та адаптації моделей машинного навчання, описаних у попередньому підрозділі. Такий підхід є важливим замінником нейронних мереж у системах з обмеженою або нерівномірно структурованою вибіркою, забезпечуючи при цьому високу пояснюваність і практичну цінність результатів.

4.4 Інтерпретація результатів та аналіз продуктивності персоналу

Інтерпретація прогнозних значень продуктивності та поведінкових характеристик працівників є ключовим етапом оцінювання ефективності розробленої аналітичної системи. Поєднання моделювання, описової статистики та кластеризаційних методів дає змогу виділити закономірності, що безпосередньо впливають на виконання завдань, стабільність робочого ритму та загальну якість роботи персоналу. На основі інтегрованої матриці ознак було здійснено комплексний аналіз поведінкових патернів, що включає тривалість робочих сесій, частку активних інтервалів, кількість виконаних дій, результативність задач та прогнозовану продуктивність моделі.

Розподіл продуктивності, представлений на рисунку 4.6, демонструє декілька характерних груп співробітників. Виявлено, що значення в діапазоні 0.20–0.45 притаманні працівникам із нерегулярним режимом роботи, короткими або фрагментованими сесіями та низьким обсягом змістовних дій. Такі користувачі зазвичай роблять часті перерви, що призводить до значної різниці між тривалістю перебування у системі та фактично продуктивним часом. Друга зона, 0.45–0.75, відповідає працівникам зі стабільним графіком, однак коливання темпу роботи свідчать про можливі труднощі з ритмічністю або концентрацією. Найбільш продуктивні працівники демонструють значення у межах 0.75–0.95, що характеризуються довгими безперервними сесіями, регулярними діями, рівномірністю виконання задач та структурованими коментарями у системі.

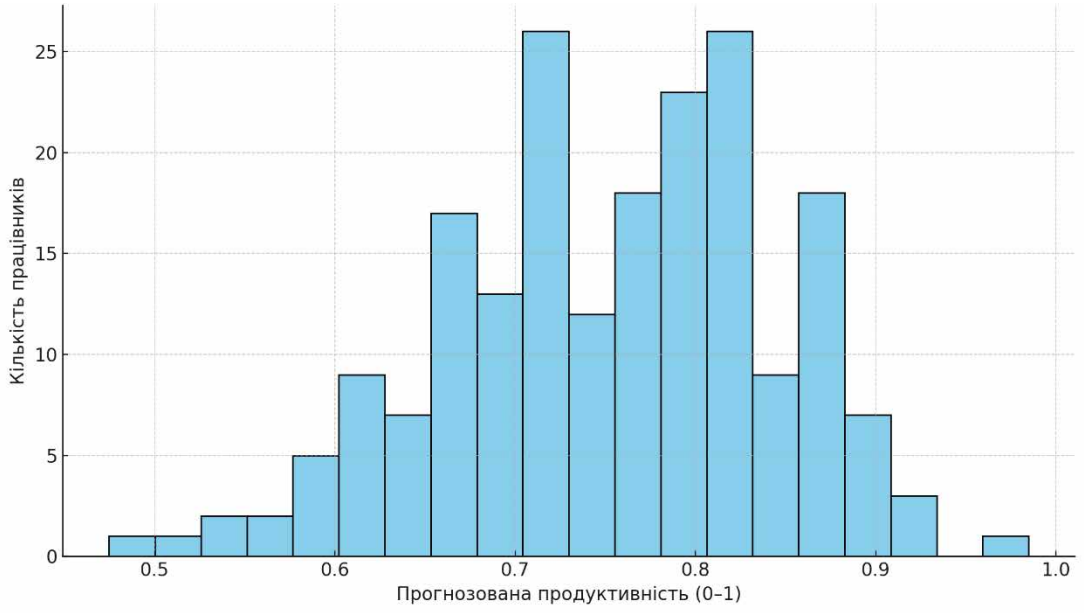

Рис. 4.6 Гістограма прогнозної продуктивності персоналу

Для поглибленого аналізу було виконано кореляційне дослідження взаємозв’язків між поведінковими та часовими характеристиками працівників. Отримана кореляційна матриця (рис. 4.6) засвідчує, що найтісніші кореляції спостерігаються між часткою активних інтервалів та кількістю виконаних дій, а також між довжиною безперервної сесії та загальною ефективністю виконання задач. Негативні залежності проявляються між інтенсивністю роботи та частотою тривалих перерв, що підтверджує припущення про вплив відпочинку не лише на продуктивність, а й на ритм залучення до робочого процесу.

Особливу увагу було приділено виявленню аномалій. За допомогою методів локальної щільності та кластеризації DBSCAN встановлено групу працівників із різкими падіннями темпу, хаотичною структурою сесій та великою часткою пасивних інтервалів. Такі профілі є важливими для HR-аналітики, оскільки дозволяють оперативно виявляти ймовірні «ризикові зони» та проєктні перевантаження або, навпаки, недозавантаження працівників.

Додатковим джерелом інтерпретації виступив аналіз важливості ознак за моделлю Random Forest. Найвпливовішими предикторами продуктивності

виявилися: частка активного часу, середня довжина безперервної сесії, кількість змістовних дій та стабільність робочого ритму. Це узгоджується з практичними спостереженнями у системах оцінювання ефективності, де ключовим чинником є не лише кількість виконаної роботи, а й її ритмічність і рівномірність.

Результати кластеризації, описані у попередньому підрозділі, стали основою для формування профілів працівників. Поєднання прогнозних значень продуктивності та кластерних характеристик дозволило виділити чотири типові групи: «високопродуктивні», «регулярні середні», «нестабільні» та «ризикові». Відмінності у поведінці та результативності між групами наведено у таблиці 4.4. Зокрема, «нестабільні» працівники демонструють подібний рівень активності з «регулярними», проте їхній ритм значно менш передбачуваний, що призводить до меншої продуктивності у довгостроковій перспективі. «Ризикові» співробітники формують невелику, але критично важливу групу для HR-втручання.

Таблиця 4.4

Порівняльні характеристики груп працівників

Група	Середня продуктивність	Стабільність ритму	Середня довжина сесії	Інтерпретація
Високо-продуктивні	0.80–0.95	Висока	Довгі сесії	Стійка робота, рівномірний темп
Регулярні середні	0.55–0.75	Середня	Середні сесії	Нормована продуктивність, рівномірний графік
Нестабільні	0.45–0.65	Низька	Нерівні сесії	Коливання активності, епізодичні провали
Ризикові	0.20–0.45	Низька	Короткі сесії	Часті перерви, низький темп роботи

Проведений аналіз демонструє, що продуктивність працівників

формується під впливом комплексної взаємодії часових, поведінкових і діяльнісних факторів. Запропонована система дає змогу не тільки прогнозувати показники ефективності, але й інтерпретувати причини їх погіршення, виявляти аномальні патерни та сегментувати персонал для подальшого прийняття управлінських рішень. Отримані результати створюють основу для підвищення прозорості HR-аналітики, вдосконалення управління командами та розвитку персоніфікованих підходів до оптимізації робочих процесів.

4.5 Висновки до 4 розділу

У четвертому розділі проведено комплексне експериментальне дослідження, що охоплює підготовку даних, порівняння моделей машинного навчання та кластеризаційний аналіз поведінкових характеристик працівників. Результати моделювання продемонстрували, що найбільш точні та стабільні прогнози продуктивності забезпечують ансамблеві методи, тоді як кластеризація дозволила виокремити типові групи співробітників із характерними патернами активності, продуктивності та стабільності робочого ритму.

Отримані експериментальні дані підтверджують здатність системи не лише прогнозувати ефективність роботи, але й інтерпретувати причини її зниження, виявляти аномалії та структурувати персонал за профілями. Поєднання моделювання та кластеризації створює практичні можливості для HR-аналітики: своєчасне виявлення ризикових груп, оптимізацію навантаження, підтримку управлінських рішень і покращення організації робочих процесів.

ВИСНОВКИ

У магістерській роботі розроблено й досліджено інформаційно-аналітичну систему для оцінювання продуктивності персоналу на основі аналізу подієвих журналів, поведінкових характеристик та часових метрик робочих сесій. На основі детального огляду предметної області встановлено потребу у системі, здатній об’єднати дані різної природи та надати об’єктивні, репрезентативні показники, що відображають реальний рівень ефективності працівників. Запропонована архітектура забезпечує інтеграцію з джерелами даних, формування аналітичних ознак та можливість масштабування для великих робочих середовищ.

У роботі реалізовано повний цикл підготовки даних: очищення, узгодження часових інтервалів, формування поведінкових і темпоральних ознак, розподіл вибірок для навчання моделей. Система моделювання включає застосування низки алгоритмів машинного навчання, що дозволило отримати точні прогностичні оцінки продуктивності та визначити набір ознак, які найбільше впливають на результативність. Особливої цінності набули результати кластеризації, що показали наявність структурованих груп працівників з різними робочими стратегіями й рівнем стабільності. Це дає змогу інтерпретувати поведінку персоналу не лише на рівні індивідуальних характеристик, але й у формі групових закономірностей.

Розроблена система поєднує прогностичну аналітику та поведінковий аналіз, формуючи основу для комплексного підходу до управління командою. Вона дозволяє виявляти ризики, оптимізувати навантаження, визначати потреби у підтримці чи додатковому контролі, а також розпізнавати сильні сторони співробітників. Отримані результати підтверджують, що така система може бути інтегрована у корпоративне середовище як інструмент підвищення ефективності роботи персоналу та удосконалення процесів управління.

Отже, узагальнюючи все вище зазначене, можна зробити наступні висновки по роботі:

1. Розроблено архітектуру інформаційно-аналітичної системи, яка забезпечує інтеграцію подієвих журналів, обробку поведінкових даних працівників та формування узгоджених показників продуктивності, що створює основу для побудови надійних аналітичних модулів.
2. Створено повний конвеєр підготовки даних, що включає очищення, нормалізацію, відновлення часових інтервалів, генерацію поведінкових і темпоральних ознак, формування вибірок для тренування та тестування моделей, забезпечуючи високу якість вхідних даних для подальшого аналізу.
3. Проведено порівняльне моделювання методами машинного навчання, у межах якого протестовано кілька алгоритмів; найвищу точність і стабільність продемонстрували ансамблеві моделі (зокрема Random Forest), що довело їх ефективність у задачах прогнозування продуктивності персоналу.
4. Визначено ключові фактори впливу на продуктивність, ідентифіковані через аналіз важливості ознак: стабільність робочого ритму, частка активного часу, довжина безперервних сесій та кількість змістовних дій, що дозволяє обґрунтовано інтерпретувати результати моделювання.
5. Виконано кластеризаційний аналіз, за результатами якого виокремлено чотири стійкі групи працівників з різними рівнями ефективності та поведінковими моделями; на цій основі сформовано профілі співробітників, що підсилюють HR-аналітику, дозволяючи виявляти ризикові та високопродуктивні сегменти.
6. Підтверджено практичну цінність системи для підтримки управлінських рішень: виявлено аномальні робочі патерни, визначено ризикові ситуації зниження ефективності та окреслено перспективні напрями розвитку – включно з упровадженням рекомендаційних механізмів і прогнозування динаміки продуктивності в реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jansen B.J., Spink A., Saracevic T. Real life, real users, and real needs: a study and analysis of user queries on the Web. Information Processing & Management., 2000.
2. Upwork, BusinessWire. Statistical reports on remote employment. 2020.
3. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805, 2019.
4. Jurafsky D., Martin J.H. Speech and Language Processing. Stanford University, 2023. URL: https://web.stanford.edu/~jurafsky/slp3/
5. Руденко С. П., Кузьмінець М. П. Цифровізація управління персоналом: сучасні тенденції та інструменти. Економіка та держава, 2021.
6. Окуліч В.А. Програмне забезпечення аналізу, контролю та обліку роботи співробітників компанії. XVI Міжнародна науково-практична конференція «Інформаційні технології в освіті, техніці та економіці». http://econference.nubip.edu.ua/index.php/itete/XVI/paper/view/4108
7. Rudenko S., Kozhukhivska O. Digital HR technologies: trends of intelligent automation. Management and Entrepreneurship in Ukraine, 2022.
8. Rutkauskas A. IoT-based monitoring systems for distributed teams. Information Systems Research, 2021.
9. Patel H., Kumar R. Evaluation of employee monitoring platforms in distributed environments. Journal of Digital Management, 2020.
10. Чаркіна Т. Ю. Інформаційні системи та технології в управлінні персоналом: сучасний стан та перспективи розвитку. Вісник економіки, 2022.
11. Kavanagh M., Thite M., Johnson R. Human Resource Information Systems: Basics, Applications, and Future Directions. SAGE, 2021.
12. Laudon K., Laudon J. Management Information Systems: Managing the Digital Firm. 17th ed. Pearson, 2022.
13. Dery K., Tansley C. Digital workforce and HR analytics in the age of remote

work. Human Resource Management Journal, 2021.

14. Rahman H. Time tracking and productivity analytics in distributed software teams. Journal of Systems and Software, 2020.
15. GitHub Octoverse Report 2020. The State of the Octoverse. GitHub Inc., 2020. URL: https://octoverse.github.com/
16. Grinberg M. Flask Web Development: Developing Web Applications with Python. O'Reilly Media, 2018.
17. Telegram Bot API Documentation. Telegram Messenger LLP. URL: https://core.telegram.org/bots/api
18. PyTelegramBotAPI Documentation. URL: https://pypi.org/project/pyTelegramBotAPI/
19. MySQL Reference Manual. Oracle Corporation. URL: https://dev.mysql.com/doc/
20. The PyCharm Guide. JetBrains Documentation. URL: https://www.jetbrains.com/pycharm/
21. Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/
22. MariaDB Documentation. MariaDB Foundation. URL: https://mariadb.com/kb/
23. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter. O'Reilly Media, 2022.
24. Гончаренко О. М., Коваль С. В. Сучасні вебтехнології та платформи для розробки інформаційних систем. Вісник Хмельницького національного університету, 2020. URL: http://journals.khnu.km.ua
25. Могилова М., Голосенко Д. Технологічна трансформація HR-менеджменту: цифрові рішення і можливості. *Empirio*. 2024. Т. 1, № 2. С. 107–120.
26. Збрицька Т. П., Сорока О. В. Управління персоналом в умовах цифрової економіки. *Економіка та суспільство*. 2021. № 31. DOI: 10.32782/2524-0072/2021-31-20.
27. Длугопольська Т. І., Гук Ю. В. Цифрова трансформація у сфері HR: напрями, проблеми та можливості. *Причорноморські економічні студії*.

2021. Вип. 62. С. 13–18.

28. Власенко Т. А., Пілінська В. В. Організаційно-економічне забезпечення управління персоналом підприємства в умовах цифрової економіки. *Ukrainian Journal of Applied Economics and Technology*. 2024. Т. 9, № 3. С. 268–276.
29. Григор'єв О. В. Інноваційні технології HR-менеджменту в умовах цифрової економіки. *Інтелект XXI*. 2023. № 2. С. 45–52. intellect21.nuft.org.ua
30. Pokojski Z., Kister A., Lipowski M. Remote Work Efficiency from the Employers' Perspective—What's Next? *Sustainability*. 2022. Vol. 14, No. 7. Art. 4220. DOI: 10.3390/su14074220.
31. Stachová K., Stacho Z., Šamalík P., Sekan F. The Impact of E-HRM Tools on Employee Engagement. *Administrative Sciences*. 2024. Vol. 14, No. 11. Art. 303. DOI: 10.3390/admsci14110303.
32. Sandoval-Reyes J., Revuelto-Taboada L., Duque-Oliva E. J. Analyzing the Impact of the Shift to Remote Work Mode on Middle Managers' Well-Being in the Pandemic. *European Research on Management and Business Economics*. 2023. Vol. 29, No. 2. Art. 100217. DOI: 10.1016/j.iedeen.2023.100217.
33. Manikam K. M. Optimizing Remote Workforce Productivity: A Non-Invasive Monitoring Framework. *International Journal of Computing and Engineering*. 2024. Vol. 6, No. 7. P. 40–51. DOI: 10.47941/ijce.2432.
34. Maghlaperidze E., Kharadze N., Kuspliak H. Development of Remote Jobs as a Factor to Increase Labor Efficiency. *Journal of Eastern European and Central Asian Research (JEECAR)*. 2021. Vol. 8, No. 3. P. 337–348. DOI: 10.15549/jeecar.v8i3.669.

ДОДАТКИ

Додаток А

А.1 Результати тестування системи

Поле	Значення
Тест-кейс: Показ головної сторінки	
Назва тесту:	Показ головної сторінки
Передумови:	Запущено веб-браузер
Схема проведення тесту:	1. В адресний рядок введено http://localhost:8080/ 2. Натиснуто кнопку для переходу на сторінку
Очікуваний результат:	Показано головну сторінку веб-застосунку, як для неавторизованого користувача
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Перехід на сторінку реєстрації	
Назва тесту:	Перехід на сторінку реєстрації
Передумови:	Показано головну сторінку веб-застосунку
Схема проведення тесту:	1. Натиснута кнопка «Sign up»
Очікуваний результат:	Показано сторінку реєстрації застосунку з полями: 1. Поле вводу «Username» для введення імені користувача 2. Поле вводу «Email» для введення поштової адреси користувача
Тест-кейс: Перехід на сторінку авторизації	
Назва тесту:	Перехід на сторінку авторизації
Передумови:	Показано головну сторінку веб-застосунку
Схема проведення тесту:	1. Натиснуто кнопку «Sign in»
Очікуваний результат:	Показано сторінку авторизації застосунку з полями: 1. Поле вводу «Email» для введення поштової адреси користувача 2. Поле вводу «Password» для введення паролю
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Реєстрація користувача	
Назва тесту:	Реєстрація користувача
Передумови:	Показано сторінку реєстрації
Схема проведення тесту:	1. Введені валідні дані в необхідні поля 2. Натиснути кнопку реєстрації «Sign up»
Очікуваний результат:	Нового користувача додано. Перехід на сторінку авторизації.
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Авторизація користувача	
Назва тесту:	Авторизація користувача
Передумови:	Показано сторінку авторизації
Схема проведення тесту:	1. Введені валідні дані в необхідні поля

	2. Натиснути кнопку авторизації «Sign in»
Очікуваний результат:	Авторизацію пройдено. Показано сторінку створення нового проєкта.
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Створення проєкта	
Назва тесту:	Створення проєкта
Передумови:	Показано сторінку створення проєкта
Схема проведення тесту:	1. Введені дані: назва проєкта, ім’я замовника 2. Натиснути кнопку «Створити»
Очікуваний результат:	Проєкт успішно створений. Показано сторінку зі списком активних проєктів.
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Вибір проєкта	
Назва тесту:	Вибір проєкта
Передумови:	Показано сторінку з активними проєктами
Схема проведення тесту:	Користувач вибирає активний проєкт зі списку
Очікуваний результат:	Показано сторінку з обраним проєктом
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Включення трекеру	
Назва тесту:	Включення трекеру
Передумови:	Показано сторінку проєкта
Схема проведення тесту:	Користувач натискає на кнопку ввімкнення трекеру
Очікуваний результат:	Трекер почав відлік часу
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Виключення трекеру	
Назва тесту:	Виключення трекеру
Передумови:	Показано сторінку проєкта
Схема проведення тесту:	Користувач натискає на кнопку вимкнення трекеру
Очікуваний результат:	Трекер припинив відлік часу
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Перегляд статистики відпрацьованих годин	
Назва тесту:	Перегляд статистики відпрацьованих годин
Передумови:	Показано сторінку проєкта
Схема проведення тесту:	Користувач натискає на кнопку звітності
Очікуваний результат:	Користувач спостерігає правильність підрахування відпрацьованих годин
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Коментування включення трекеру	

Назва тесту:	Коментування включення трекеру
Передумови:	Показано сторінку проєкта з включеним трекером
Схема проведення тесту:	Користувач заповнює поле для коментування включення трекеру та зберігає дані
Очікуваний результат:	Користувач спостерігає збережене коментування
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Перегляд знімків екрану	
Назва тесту:	Перегляд знімків екрану
Передумови:	Показано телеграм-бот програми
Схема проведення тесту:	Користувач вводить валідні дані для під’єднання до проєкта
Очікуваний результат:	Під’єднання успішне. Отримання повідомлень зі знімками екрану
Чи відповідає спостережуваний результат очікуваному?	Так
Тест-кейс: Оплата проєкту	
Назва тесту:	Оплата проєкту
Передумови:	Користувач отримав всі знімки екрану. Показане посилання на оплату проєкту
Схема проведення тесту:	Користувач вводить валідні дані для оплати проєкту
Очікуваний результат:	Оплата успішна. Підтвердження виконано
Чи відповідає спостережуваний результат очікуваному?	Так

Додаток Б

Б.1 Тексти програмного коду

```
{% extends 'base.html'%}

{%
block title %}
Страница
авторизации
{% endblock
%}

{% block body %}

<h1>Страница авторизации </h1>

<main class="container mt-5">

 <form method="post">

<h1 class="h3 mb-3 fw-normal">Please sign in</h1>

<input type="email" name="email" class="form-control" id="email" placeholder="name@example.com"><br>
<input type="password" name="password" class="form-control" id="password" placeholder="Password"><br>
<button class="w-100 btn btn-lg btn-primary" type="submit" formaction="/create">Sign in</button>
</form> </main>

{% endblock %}

<!doctype html> <html lang="en"> <head>
<meta charset="UTF-8"> <meta name="viewport"
content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}"> <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css">
<title>{% block title %}{% endblock %}</title>

</head> <body>
{% block body %}{% endblock %} </body>
</html>


{% extends 'base.html'%}


{% block title %} Создание проекта {% endblock %}


{% block body %}
```

```
<div class="conteiner mt-5"> <h1>Создание проекта  </h1>
<header class="bd-header bg-dark py-3 d-flex align-items-stretch border-bottom border-blue">
<div class="container-fluid d-flex align-items-center"> <h1 class="h3 mb-3 fw-normal">My time</h1>
<button class="ms-auto link-light" type="submit" formaction="/">+</button> </div>
</header>

<div class="conteiner mt-5">

<input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
</div><br>

<div class="row gy-3"> <div class="col-md-6">
<label class="form-label">Name on card</label> </div>
<div class="col-md-6">

<label class="form-label">Credit card number</label>
</div> </div>
<div class="styles217493542__root Grid2890349840__columnRoot _1gSj_" style="grid-area: span 1 / span 8 / auto / auto;">
<div class="_-8w2N _6fWpo"> <div>
<div data-hook="step" class="Box2645419772__root Box2645419772---direction-10-horizontal Box2645419772---justifyContent-6-middle" style="--Box2645419772-gap:0;">
<div class="Box2645419772__root Box2645419772---direction-8-vertical _1QxH3


_3Ekjt


{% endblock %}

{% extends 'base.html'%}


{% block title %} Главная страница {% endblock %}


{% block body %}

<header class="container mt-5"> <form>
<a class="btn btn-sm btn-outline-secondary" href="/autorisation">Sign in</a> <a class="btn btn-sm btn-outline-secondary" href="/registration">Sign up</a> </form>
</header>

<h1>Главная страница </h1> {% endblock %}
{% extends 'base.html'%}


{% block title %} Страница регистрации {% endblock %}
{% block body %}

<h1>Страница регистрации </h1> <!--
<div class="container mt-5"> <form method="post">
<h1 class="h3 mb-3 fw-normal">Please sign in</h1>
```

```
<input type="email" name="email" class="form-control" id="floatin_gemail" placeholder="name@example.com"><br>
<input type="password" name="password" class="form-control" id="floatingPassword" placeholder="Password"><br>
<button class="w-100 btn btn-lg btn-primary" type="submit" formaction="/create">Sign in</button>
</form> </div>
-->

<main class="container mt-5"> <form method="post">
<h1 class="h3 mb-3 fw-normal">Please sign up</h1>

<input type="text" name="user" class="form-control" id="user" placeholder="User"><br>
<input type="email" name="email" class="form-control" id="email"
placeholder="name@example.com"><br>

<input type="password" name="password" class="form-control" id="password" placeholder="Password"><br>
<input type="password" name="confirmpassword" class="form-control" id="confirmpassword" placeholder="Confirm Password"><br>
<button class="w-100 btn btn-lg btn-primary" type="submit" formaction="/autorisation">Sign up</button>
</form> </main>
{% endblock %}


import telebot

from telebot import types


TOKEN = '1765799695:AAFyY7YtsANfI10yJD4fdFPKLKEliSOOVRY'
bot = telebot.TeleBot(TOKEN)


@bot.message_handler(commands=['start']) def email(message):
bot.send_message(message.chat.id, " Введи, пожалуйста, email разработчика")


@bot.message_handler(content_types=['text']) def auto(message):


if message.text == "marisha73@gmail.com":

bot.send_message(message.chat.id, " Вы подключились, к проекту Lorem ipsum dolor sit amet")
bot.send_message(message.chat.id, " Каждые 10 минут вам будут приходить скриншоты работы разработчика")


p1 = open("p1.png", 'rb')
```

```
p2 = open("p2.png", 'rb') p3 = open("p3.png", 'rb') p4 = open("p4.png", 'rb') p5 = open("p5.png", 'rb')


bot.send_photo(message.chat.id, p1) bot.send_photo(message.chat.id, p2) bot.send_photo(message.chat.id, p3) bot.send_photo(message.chat.id, p4) bot.send_photo(message.chat.id, p5)


markup = types.InlineKeyboardMarkup()

item = types.InlineKeyboardButton("Оплатить проект", callback_data='hi') markup.add(item)
bot.send_message(message.chat.id, " Разработчик закончил работу над проэктом", reply_markup=markup)
else:

bot.send_message(message.chat.id, " Вашего емайла нет в базе")



bot.polling()


if __name__ == "__main__": app.run(debug=True)# ukazivaet na oshibki
import random2

from datetime import datetime import numpy as np
import pyautogui import imutils import cv2


def random():

minute_random = int(random2.uniform(1, 11)) print(minute_random)
return minute_random


def time_now(time_hour, time_minute):

if time_hour == 1 and time_minute == 0: time_hour = datetime.now().hour return time_hour
elif time_hour == 0 and time_minute == 1: time_minute = datetime.now().minute return time_minute
else:

print('veli ne pravelnie znachenia')


def only_minute(hour, minute): only_minute = hour*60+minute return only_minute


def randomnie_minute(minute_ten, ran):
```

```
        next_minute = (minute_ten + 1) * 10 + ran return next_minute


        def future_time(): ran=random()
        minute_ten = int(time_now(0,1)/10)

        next_minute = randomnie_minute(minute_ten, ran)


        if next_minute >= 60:

        new_time_now_hour = time_now(1, 0) + 1 new_time_now_minute = next_minute - 60
        alone_minute = only_minute(new_time_now_hour,new_time_now_minute)
print(alone_minute)
        return alone_minute else:
        alone_minute = only_minute(time_now(1, 0), next_minute) print(alone_minute)
        return alone_minute


        def main():

        if only_minute(time_now(1, 0), time_now(0, 1)) == future_time(): while True:
        #if only_minute(time_now(1,0),time_now(0, 1)) == future_time(): image =
pyautogui.screenshot(region=(0, 0, 300, 400))
        image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
cv2.imshow('Screenshot', imutils.resize(image, width=600)) cv2.imwrite('pic.png', image)
        break


        main()

        " style="--Box2645419772-gap:0;"><div><div class="Box2645419772__root
Box2645419772---direction-10-horizontal Box2645419772---alignItems-13-space-between
Box2645419772---justifyContent-6-middle" style="--Box2645419772-gap:0;"><div
class="Box2645419772__root Box2645419772---direction-10-horizontal Box2645419772---
        justifyContent-6-middle" style="min-height: 24px; --Box2645419772-gap:0;"><svg
viewBox="0 0 18 18" fill="currentColor" width="18" height="18" class="_3TSQe _2YSSl" data-
hook="step-incomplete"><circle cx="9" cy="9" r="2"></circle></svg><div
class="Box2645419772__root Box2645419772---direction-10-horizontal" style="padding-right:
12px; padding-left: 8px; --Box2645419772-gap:0;"><span class="Text4265422782__root
Text4265422782---size-4-tiny Text4265422782---skin-8-standard Text4265422782---weight-4-
thin Text4265422782---list-style-9-checkmark" data-size="tiny" data-secondary="false" data-
skin="standard" data-light="false" data-weight="thin" data-list-style="checkmark"><span data-
hook="step-title" class="Text4265422782__root Text4265422782---size-4-tiny
Text4265422782---skin-8-standard Text4265422782---weight-6-normal Text4265422782---list-
style-9-checkmark awxcV" data-size="tiny" data-secondary="false" data-skin="standard" data-
light="false" data-weight="normal" data-list-style="checkmark"> </span><span data-
hook="step-time-to-complete" class="Text4265422782__root Text4265422782---size-4-tiny
Text4265422782--secondary Text4265422782---skin-8-standard Text4265422782--light
Text4265422782---weight-4-thin Text4265422782---list-style-9-checkmark _1gbdK" data-
size="tiny" data-secondary="true" data-skin="standard" data-light="true" data-weight="thin"
data-list-style="checkmark">(7&amp;nbsp;мин.)</span></span></div></div><div
```

class="Box2645419772__root Box2645419772---direction-10-horizontal" style="--Box2645419772-gap:0;"> <button data-size="tiny" data-skin="standard" data-underline="none" data-weight="thin" class="buttonnext967279998__root Focusable1296630989__root TextButton834966688__root TextButton834966688---skin-8-standard TextButton834966688---underline-4-none TextButton834966688---weight-4-thin TextButton834966688---size-4-tiny" data-hook="step-skip" type="button" tabindex="0" aria-disabled="false">Пропустить</button> <button data-madefor="true" data-size="small" data-priority="primary" data-skin="standard" class="buttonnext967279998__root Focusable1296630989__root Button2487934312__root Button2487934312---skin-8-standard Button2487934312---priority-7-primary Button2487934312---size-5-small Button2487934312--ellipsis _3XmQO" data-hook="step-cta" type="button" tabindex="0" aria-disabled="false">Добавить товар</button> <div class="Box2645419772__root Box2645419772---direction-8-vertical _1QxH3

" style="--Box2645419772-gap:0;"> <svg viewBox="0 0 18 18" fill="currentColor" width="18" height="18" class="_3TSQe _2YSSl" data-hook="step-incomplete"><circle cx="9" cy="9" r="2"></circle></svg> <span class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782---skin-8-standard Text4265422782---weight-4-thin Text4265422782---list-style-9-checkmark" data-size="tiny" data-secondary="false" data-skin="standard" data-light="false" data-weight="thin" data-list-style="checkmark"><span data-hook="step-title" class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782---skin-8-standard Text4265422782---weight-6-normal Text4265422782---list-style-9-checkmark awxcV" data-size="tiny" data-secondary="false" data-skin="standard" data-light="false" data-weight="normal" data-list-style="checkmark">Настройте способы оплаты<span data-hook="step-time-to-complete" class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782--secondary Text4265422782---skin-8-standard Text4265422782--light Text4265422782---weight-4-thin Text4265422782---list-style-9-checkmark _1gbdK" data-size="tiny" data-secondary="true" data-skin="standard" data-light="true" data-weight="thin" data-list-style="checkmark">(4 мин.) <button data-size="tiny" data-skin="standard" data-underline="none" data-weight="thin" class="buttonnext967279998__root Focusable1296630989__root TextButton834966688__root TextButton834966688---skin-8-standard TextButton834966688---underline-4-none TextButton834966688---weight-4-thin TextButton834966688---size-4-tiny"

data-hook="step-skip" type="button" tabindex="0" aria-disabled="false">Пропустить</button> <button data-madefor="true" data-size="small" data-priority="secondary" data-skin="standard" class="buttonnext967279998__root Focusable1296630989__root Button2487934312__root Button2487934312---skin-8-standard Button2487934312---priority-9-secondary Button2487934312---size-5-small Button2487934312--ellipsis _3XmQO" data-hook="step-cta" type="button" tabindex="0" aria-disabled="false">Start</button> <svg viewBox="0 0 18 18" fill="currentColor" width="18" height="18" class="_3TSQe _2YSSl" data-hook="step-incomplete"><circle cx="9" cy="9" r="2"></circle></svg> <span class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782---skin-8-standard Text4265422782---weight-4-thin Text4265422782---list-style-9-checkmark" data-size="tiny" data-secondary="false" data-skin="standard" data-light="false" data-weight="thin" data-list-style="checkmark"><span data-hook="step-title" class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782---skin-8-standard Text4265422782---weight-6-normal Text4265422782---list-style-9-checkmark awxcV" data-size="tiny" data-secondary="false" data-skin="standard" data-light="false" data-weight="normal" data-list-style="checkmark">/span><span data-hook="step-time-to-complete" class="Text4265422782__root Text4265422782---size-4-tiny Text4265422782--secondary Text4265422782---skin-8-standard Text4265422782--light Text4265422782---weight-4-thin Text4265422782---list-style-9-checkmark _1gbdK" data-size="tiny" data-secondary="true" data-skin="standard" data-light="true" data-weight="thin" data-list-style="checkmark">(3 мин.) <button data-size="tiny" data-skin="standard" data-underline="none" data-weight="thin" class="buttonnext967279998__root Focusable1296630989__root TextButton834966688__root TextButton834966688---skin-8-standard TextButton834966688---underline-4-none TextButton834966688---weight-4-thin TextButton834966688---size-4-tiny" data-hook="step-skip" type="button" tabindex="0" aria-disabled="false"> </button> <button data-madefor="true" data-size="small" data-priority="secondary" data-skin="standard" class="buttonnext967279998__root Focusable1296630989__root Button2487934312__root Button2487934312---skin-8-standard Button2487934312---priority-9-secondary Button2487934312---size-5-small Button2487934312-

-ellipsis _3XmQO" data-hook="step-cta" type="button" tabindex="0" aria-disabled="false">
</button>