БАКАЛАВРСЬКА РОБОТА

КБР.ІСТ-14.00.00.000 ПЗ

Група ІСТ-22-1

Ніка ПАНЕВА

2026

Івано-Франківський Національний Технічний Університет Нафти і Газу
Факультет Інформаційних Технологій
Кафедра інформаційно-телекомунікаційних технологій і систем

Панева Ніка Сергіївна

УДК _____.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Розроблення інформаційної системи обліку робочого часу здобувачів освіти під час самостійної роботи

Інформаційні системи та технології
(назва освітньої програми)

126 - Інформаційні системи та технології
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____*Панева Н. С.*_____
(підпис, прізвище та ініціали здобувача)

Науковий керівник _____*Воропаєва А. О.*, доцент_____
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

В. о. завідувача кафедри

_____ Заміховська О. Л.
(посада) (підпис) (дата) (прізвище та ініціали)

м. Івано-Франківськ – 2026 рік

Івано-Франківський національний технічний університет нафти і газу

Факультет *Інформаційних технологій*

Кафедра *Інформаційно-телекомунікаційних технологій і систем*

Освітньо-кваліфікаційний рівень *бакалавр*

Спеціальність *126 – Інформаційні системи та технології*

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІТТС

_____ Заміховська О.Л.

«_» _____ *2026*року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Паневій Ніці Сергіївній

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи*) Розроблення інформаційної системи обліку робочого часу здобувачів освіти під час самостійної роботи*

керівник проекту (роботи) *Воропаєва А. О., доцент* ,

затверджені наказом вищого навчального закладу від

2. Строк подання студентом проекту (роботи) *16 червня 2026р.*

3. Вихідні дані до роботи *Методичні вказівки, технічна література* .

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1 Огляд аналогів та постановка завдання; 2 Розробка структури бази даних, розробка UML діаграми; 3 Розробка вигляду та функціоналу веб-орієнтованої інформаційної системи* .

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. UML діаграма

2. Інтерфейси та їх реалізація

6. Консультатни розділів роботи

Розділ	**Консультант**	**Підпис, дата**
1-3	*Воропаєва А. О.*	

7. Дата видачі завдання – *22 грудня 2025 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка завдання	Березень, 2026	виконано
2	Розробка структури інформаційної системи	Квітень, 2026	виконано
3	Розробка програмного забезпечення	Травень, 2026	виконано
4	Оформлення пояснювальної записки	Червень, 2026	виконано

Студент _____ Панева Н. С.

Керівник роботи _____ Воропаєва А. О.

АНОТАЦІЯ

В рамках випускної кваліфікаційної роботи розроблено веб-орієнтовану інформаційну систему для обліку робочого часу здобувачів освіти під час самостійної роботи на базі фреймворку Flask та ORM SQLAlchemy.

В результаті було створено функціональний вебдодаток із рольовим доступом для адміністраторів, викладачів та студентів. Система дозволяє студентам у реальному часі фіксувати тривалість сесій самостійної роботи за допомогою таймера, а викладачам — здійснювати централізований моніторинг прогресу та переглядати статистику закріплених академічних груп. Інтерфейс користувача є інтуїтивно зрозумілим та адаптивним для різних пристроїв.

Для реалізації back-end частини використовувалися мова Python та архітектура зв'язків «багато-до-багатьох» для дисциплін і груп, що забезпечує надійність збереження даних. Front-end частина реалізована за допомогою шаблонізатора Jinja2 та CSS-фреймворку Bootstrap, що дозволило створити гнучкий та кросплатформний інтерфейс.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК РОБОЧОГО ЧАСУ, САМОСТІЙНА РОБОТА, ВЕБДОДАТОК, FLASK, PYTHON, SQLALCHEMY, BOOTSTRAP.

ANNOTATION

Within the framework of the final qualifying work, a web-oriented information system for tracking the working time of education seekers during independent work was developed using the Flask framework and SQLAlchemy ORM.

As a result, a functional web application with role-based access for administrators, teachers, and students was created. The system allows students to record independent work sessions in real-time via a timer, while enabling teachers to centrally monitor progress and view statistics of assigned academic groups. The user interface is intuitive and responsive across various devices.

The back-end part was implemented using Python and a Many-to-Many relationship architecture for subjects and groups, ensuring data reliability and security. The front-end part was developed using the Jinja2 template engine and the Bootstrap CSS framework, which allowed for a flexible and cross-platform interface.

Keywords: INFORMATION SYSTEM, TIME TRACKING, INDEPENDENT WORK, WEB APPLICATION, FLASK, PYTHON, SQLALCHEMY, BOOTSTRAP.

ЗМІСТ

ВСТУП **7**

1 АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ОБЛІКУ РОБОЧОГО ЧАСУ ЗДОБУВАЧІВ ОСВІТИ 9

1.1 Опис предметної області та аналіз академічного контексту обліку часу 9

1.2 Комплексний порівняльний аналіз програмних аналогів систем тайм-трекінгу 12

1.3 Постановка завдання 18

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ 20

2.1 Загальна структура веб-орієнтованої інформаційної системи 20

2.2 UML-діаграма варіантів використання 21

2.3 Обґрунтування та розробка логічної структури таблиць бази даних 23

2.4 Структура бази даних 30

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ «EDUTIMETRACKER» 31

3.1 Технології розроблення серверної частини (Back-end) системи 31

3.2 Технології розроблення клієнтської частини (Front-end) системи 33

3.3 Програмна реалізація бізнес-логіки, маршрутизації та розмежування прав доступу 34

3.4 Програмна реалізація модулів аналітики та кабінетів користувачів 37

3.5 Тестування працездатності та оптимізація компонентів вебдодатка 44

ВИСНОВКИ 47

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 49

ДОДАТКИ 51

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

БД – База даних.

ІС – Інформаційна система.

СРС – Самостійна робота студентів.

API – Application Programming Interface (програмний інтерфейс додатка).

Bootstrap – CSS-фреймворк для розробки адаптивних веб-інтерфейсів.

Flask – мікрофреймворк мовою Python, що використовується для розробки вебдодатків.

ORM – Object-Relational Mapping (об'єктно-реляційне відображення даних).

SQLAlchemy – інструментарій для роботи з базами даних та ORM-система для мови Python.

ВСТУП

У сучасному світі самостійна робота студентів (СРС) займає значну частину навчального процесу у вищій освіті. Проте відстеження реальних часових витрат на вивчення дисциплін та фіксація прогресу без спеціалізованих інструментів є складними завданнями. Традиційні методи, такі як паперові журнали або електронні таблиці, є незручними, застарілими та вимагають багато часу для адміністрування.

У зв'язку з цим процес управління та обліку навчального часу за допомогою вебдодатків стає все більш актуальним. Інформаційні системи обліку часу (тайм-трекери) дозволяють користувачам легко фіксувати тривалість сесій у реальному часі, аналізувати продуктивність та планувати роботу. Вони надають можливості для структурування завдань за дисциплінами та автоматичного збору статистики.

Проте існуючі комерційні рішення для обліку часу, такі як Toggl Track або Clockify, мають свої недоліки при адаптації до освітнього процесу. Серед них — орієнтація суто на бізнес-сегмент, відсутність специфічної академічної рольової моделі (Адміністратор–Викладач–Студент) та складність налаштування гнучких зв’язків між навчальними групами й дисциплінами.

Таким чином, розробка нової веб-орієнтованої інформаційної системи обліку робочого часу здобувачів освіти, яка б враховувала недоліки існуючих рішень та забезпечувала зручний, інтуїтивно зрозумілий інтерфейс із функцією інтерактивного таймера, є актуальною задачею.

Метою роботи є розробка веб-орієнтованої інформаційної системи обліку робочого часу здобувачів освіти під час самостійної роботи («EduTimeTracker») на базі фреймворку Flask та ORM SQLAlchemy, яка дозволить користувачам ефективно фіксувати навчальний час та забезпечить викладачів інструментами централізованого моніторингу СРС.

Об’єктом дослідження є процеси автоматизації обліку, моніторингу та аналізу робочого часу здобувачів освіти за допомогою веб-орієнтованих інформаційних технологій.

Предметом дослідження є методи та засоби розробки вебдодатка для управління та обліку навчального часу, зокрема використання Python, Flask, ORM SQLAlchemy для роботи з базою даних та фреймворку Bootstrap для створення зручного інтерфейсу користувача.

1 АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ОБЛІКУ РОБОЧОГО ЧАСУ ЗДОБУВАЧІВ ОСВІТИ

1.1 Опис предметної області та аналіз академічного контексту обліку часу

Предметна галузь даного дослідження охоплює сферу цифрової трансформації вищої освіти, методологію управління навчальним процесом та впровадження спеціалізованих інформаційно-комунікаційних технологій для моніторингу самостійної роботи здобувачів освіти. В умовах сучасного розвитку вищої школи в Україні та її інтеграції до європейського освітнього простору відповідно до стандартів Болонського процесу, структура освітніх програм зазнала суттєвих трансформацій. Однією з ключових особливостей сучасної системи кредитно-модульного навчання є перенесення значної частини навчального навантаження на самостійну роботу студентів (СРС). Залежно від специфіки дисципліни та навчального плану, обсяг СРС може складати від 40% до 60% від загальної кількості годин, передбачених Європейською кредитно-трансферною системою (ECTS). Такий підхід спрямований на формування у майбутніх фахівців навичок безперервного навчання, критичного мислення та високого рівня професійної автономії [1].

Проте практична реалізація цієї концепції стикається із серйозними організаційними та методологічними бар'єрами, серед яких ключовою проблемою залишається високий рівень інформаційної асиметрії та децентралізації даних у трикутнику взаємодії суб'єктів освітнього процесу. З боку безпосередніх здобувачів освіти головним деструктивним чинником виступає відсутність розвинених навичок академічного тайм-менеджменту та самоорганізації. Студенти часто стикаються з проблемою нераціонального розподілу часових ресурсів, прокрастинацією та невмінням об'єктивно оцінити реальну трудомісткість виданих навчальних завдань, таких як лабораторні роботи, курсові проєкти або розрахунково-графічні роботи. Внаслідок цього

виникає хронічний дефіцит часу наприкінці семестру, що призводить до накопичення академічних заборгованостей та суттєвого зниження загальної якості засвоєння знань. Для нівелювання цих факторів студенти як перша ключова група зацікавлених сторін проєкту потребують зручного, інтуїтивно зрозумілого та кросплатформеного інструментарію для ведення персонального хронометражу навчальної діяльності, який забезпечить можливість запускати інтерактивний таймер, гнучко структурувати записи за конкретними дисциплінами й додавати коментарі до підзадач для подальшого аналізу власної продуктивності.

Водночас ефективна організація дидактичного процесу вимагає врахування інтересів професорсько-викладацького складу, який на сучасному етапі позбавлений інструментів об'єктивного, оперативного та прозорого контролю за ходом виконання самостійної роботи. Традиційні методи моніторингу, що обмежуються періодичними усними опитуваннями, перевіркою проміжних звітів або фіксацією виконання завдань безпосередньо під час практичних занять, є дискретними та суб'єктивними. Вони відображають лише кінцевий результат на певний момент часу, але не надають інформації про безпосередній процес роботи студента, сумарний час, витрачений на вивчення теоретичного матеріалу чи написання програмного коду, та динаміку залученості всієї академічної групи. Це створює суттєве бюрократичне навантаження на лекторів та асистентів, які змушені витрачати значний часовий ресурс на рутинні процеси збору та верифікації проміжної звітності [2]. Впровадження автоматизованої системи дозволяє викладачам як другого важливого стейкхолдера у реальному часі переглядати зведені відомості про сумарні витрати часу, фільтрувати дані за дисциплінами й оперативно виявляти студентів із критично низьким рівнем залученості для надання своєчасної консультаційної підтримки.

Окрім безпосередніх учасників аудиторної та позааудиторної взаємодії, результати моніторингу навчального часу мають стратегічне значення для адміністрації закладу вищої освіти, включаючи деканати, керівництво

факультетів, гарантів освітніх програм та навчально-методичні відділи. На цьому найвищому рівні стратегічного управління та забезпечення якості освітньої діяльності виникає об'єктивна потреба в інструментах глобального аналізу для оцінки збалансованості студентського навантаження протягом усього семестру. Відсутність консолідованих даних унеможливлює точний аудит реальної трудомісткості навчальних планів, що нерідко призводить до нерівномірного розподілу завдань між тижнями та критичного перевантаження здобувачів освіти. Доступ до узагальненої аналітики та автоматично згенерованих звітів дозволяє адміністративним користувачам оптимізувати співвідношення між аудиторною та самостійною роботою, підвищити прозорість освітнього процесу, а також оперувати об'єктивними метриками під час внутрішнього аудиту та підготовки до процедур акредитації освітніх програм.

Впровадження спеціалізованих веб-орієнтованих інформаційних технологій обліку часу в академічне середовище покликане об'єднати інформаційні потоки всіх зазначених сторін у межах єдиного цифрового простору. Проте аналіз ринку сучасного програмного забезпечення показує, що переважна більшість існуючих систем тайм-трекінгу орієнтована виключно на бізнес-сегмент, корпоративне управління, IT-індустрію або фріланс. Комерційні платформи оперують суто бізнесовими сутностями, такими як проєкти, клієнти, рейти оплати, бюджети та білінг, повністю ігноруючи специфіку та складну ієрархічну структуру закладів вищої освіти. В університетському середовищі базовими структурними одиницями є факультети, кафедри, академічні групи, навчальні курси та конкретні види самостійної роботи, що потребує гнучкої архітектури зв'язків «багато-до-багатьох». Наслідком цього є високий поріг входу, неможливість повноцінної адаптації комерційного софту до умов вищої школи та фінансові обмеження, що робить розробку спеціалізованого, безкоштовного та повністю інтегрованого рішення «EduTimeTracker» високоактуальним науково-практичним завданням для вітчизняної вищої школи.

1.2 Комплексний порівняльний аналіз програмних аналогів систем тайм-трекінгу

Для формування об'єктивного та вичерпного переліку вимог до проєктуємої інформаційної системи «EduTimeTracker» виникає необхідність у проведенні детального аналізу існуючих програмних аналогів, які є індустріальними стандартами у сфері обліку робочого часу та моніторингу продуктивності. Проведення компаративного дослідження дозволяє вивчити архітектурні рішення, ергономічні особливості інтерфейсів та функціональні можливості провідних платформ, що дає змогу запозичити найкращі практики проєктування та уникнути повторення критичних помилок чи деструктивних інженерних рішень. На сучасному етапі розвитку IT-індустрії ринок програмного забезпечення пропонує широкий спектр систем хронометражу, проте для детального аналізу було відібрано три найбільш репрезентативні платформи, що базуються на принципово різних підходах до фіксації даних: Toggl Track, Clockify та RescueTime.

Першим об'єктом дослідження виступає хмарна платформа Toggl Track, яка позиціонується як один із найпопулярніших та найфункціональніших інструментів для індивідуального та командного тайм-трекінгу у світі [3]. З погляду архітектури дана система є високонавантаженим кросплатформеним Web-додатком, що функціонує за моделлю SaaS (програмне забезпечення як послуга) з інтенсивним використанням хмарних баз даних для забезпечення миттєвої синхронізації між клієнтськими застосунками для настільних і мобільних операційних систем. Головний екран інтерфейсу Toggl Track орієнтований на максимальну ергономічність та швидкий доступ до базової функції — запуску інтерактивного таймера в реальному часі. Користувач має можливість у верхній панелі ввести текстовий опис поточної активності, прив'язати її до конкретного проєкту, клієнта чи задати унікальні теги для подальшої фільтрації, після чого запустити відлік часу одним кліком. Нижче під інтерактивною панеллю розташовується хронологічний список попередніх

записів (тайм-логів), згрупованих за днями тижня, що дозволяє користувачу наочно оцінювати ретроспективу своєї зайнятості та оперативно редагувати тривалість чи опис будь-якого минулого запису, як це продемонстровано на рисунку 1.1.

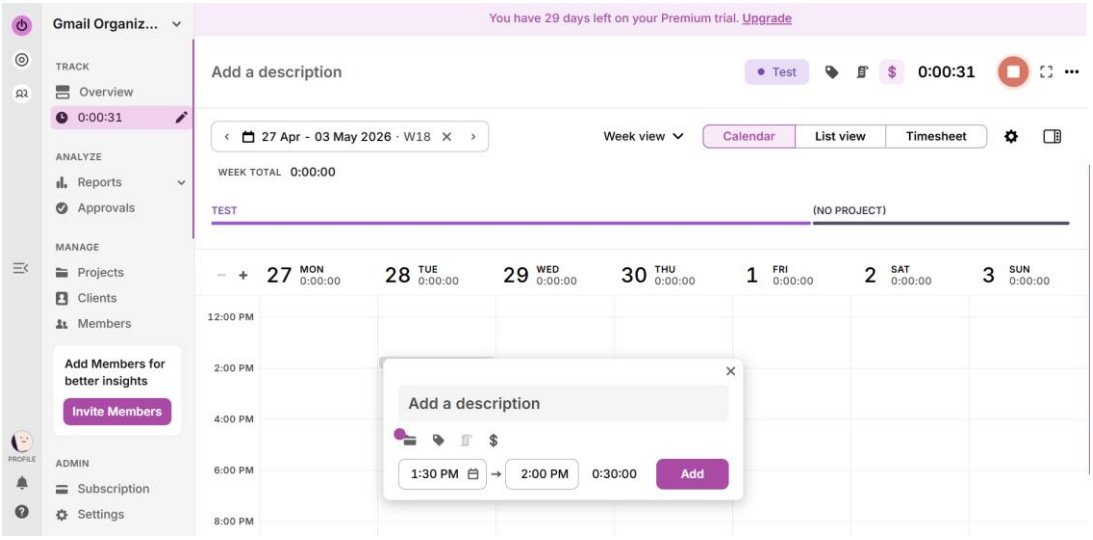

Рисунок 1.1 – Головний екран системи Toggl Track з інтерактивним таймером

Окрім безпосередньої фіксації часу, важливим архітектурним модулем Toggl Track є розгалужена підсистема графічних та аналітичних звітів, призначена для візуалізації накопичених масивів інформації. Цей розділ інтерфейсу надає користувачам та керівникам команд потужні інструменти аналізу продуктивності за допомогою динамічних кругових, стовпчикових та лінійних діаграм. Система автоматично розраховує сумарні витрати часу за визначені періоди, демонструє відсоткове співвідношення між різними проєктами чи тегами, а також дозволяє експортувати сформовані відомості у форматах CSV або PDF для зовнішнього аудиту, приклад логічної структури якого наведено на рисунку 1.2.

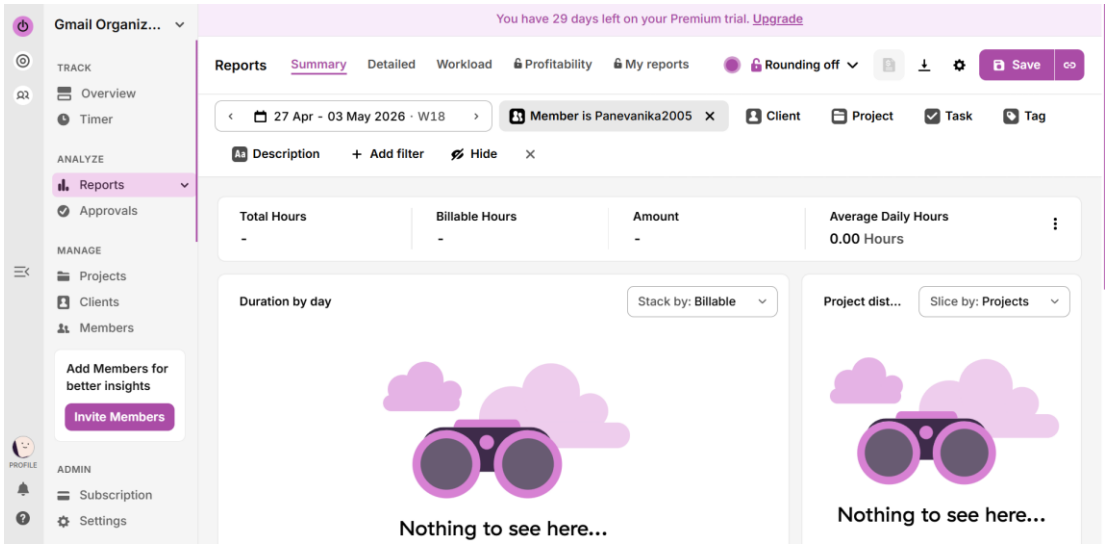

Рисунок 1.2 – Розділ аналітичних звітів у системі Toggl Track

Аналіз сильних сторін Toggl Track дозволяє виділити високу швидкість відгуку інтерфейсу, бездоганну оптимізацію UX-дизайну, наявність зручних розширень для браузерів та високий рівень автоматизації синхронізації даних. Проте в контексті адаптації платформи до освітнього процесу закладу вищої освіти виявляються суттєві архітектурні та функціональні недоліки. Платформа орієнтована суто на комерційний сектор, фріланс та бізнес-команди, через що вся її логіка побудована навколо білінгу, фінансових рейтів та бюджетів проєктів. У системі повністю відсутня специфічна академічна рольова модель, яка б дозволяла зв'язувати студентів, викладачів та адміністрацію в єдину ієрархію. Викладач позбавлений можливості централізовано створювати навчальні курси, закріплювати їх за конкретними академічними групами та автоматично збирати звіти про самостійну роботу, а розширені інструменти командного моніторингу заблоковані за дорогою платною підпискою.

Другим аналогом, обраним для дослідження, є система Clockify, яка за своєю функціональною структурою є головним конкурентом Toggl Track, проте має суттєву відмінність у політиці монетизації, пропонуючи значно більший обсяг можливостей у безкоштовній версії [4]. Архітектурно Clockify побудований як сучасна веб-платформа, що використовує потужний API-first підхід, де клієнтська частина базується на реактивних JavaScript-фреймворках

для забезпечення високої інтерактивності, а серверна частина оперує великими сховищами даних із підтримкою гнучких інструментів адміністрування прав користувачів. Інтерфейс системи пропонує два альтернативні режими обліку часу: класичний інтерактивний таймер та так званий тайм-шит (таблицю зайнятості), де користувач може вносити сумарну кількість відпрацьованих годин за весь день або тиждень у розрізі проєктів без необхідності постійного запуску секундоміра. Панель моніторингу Clockify фокусується на відображенні активності команди в реальному часі, дозволяючи менеджеру бачити, над якими саме завданнями працюють учасники проєкту в поточну секунду, що ілюструє рисунок 1.3.

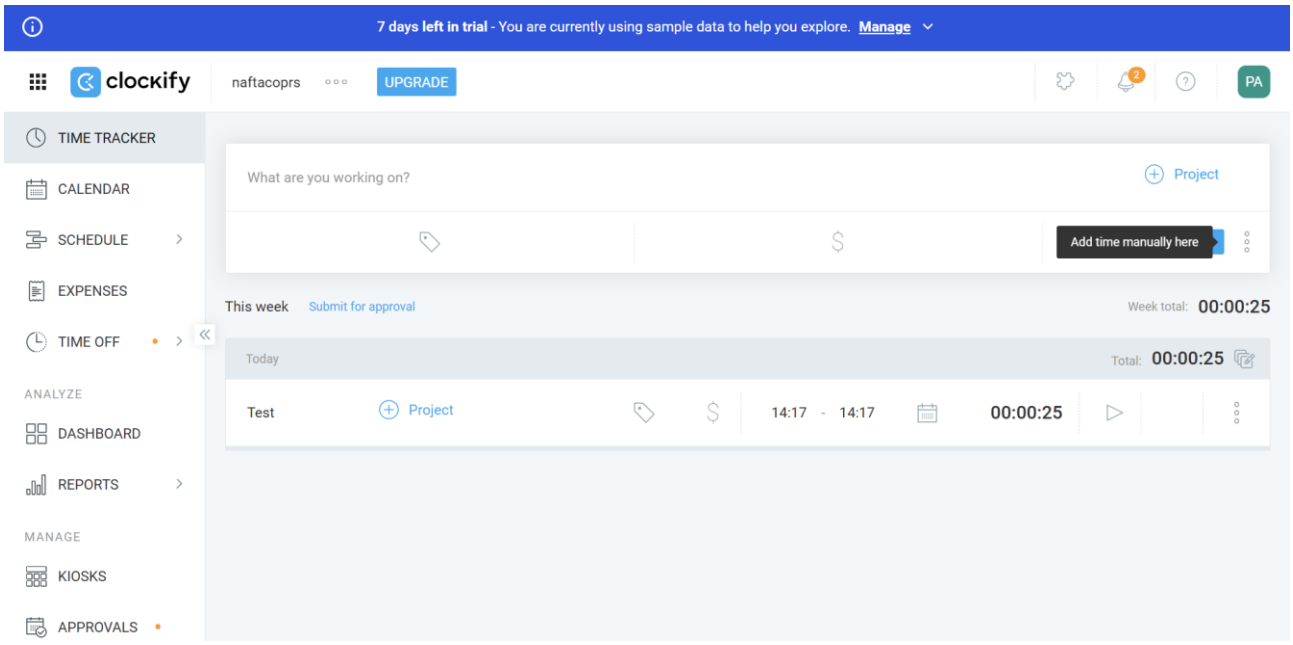

Рисунок 1.3 – Панель моніторингу та тайм-шит у системі Clockify

Серед очевидних переваг системи Clockify варто відзначити її безкоштовність для необмеженої кількості користувачів та базових проєктів, що робить її теоретично доступною для великих колективів, а також наявність розвиненого модуля управління командами, де можна створювати групи користувачів та призначати їм різні рівні доступу. Незважаючи на ці переваги, спроба впровадження Clockify як університетського інструменту контролю самостійної роботи студентів стикається із жорсткими обмеженнями. Інтерфейс

та термінологія системи залишаються суто корпоративними, де замість навчальних дисциплін фігурують «проєкти», а замість кафедр чи факультетів — «клієнти». Оскільки система не володіє знаннями про специфіку університетського розкладу та академічних зв'язків «багато-до-багатьох», процес налаштування взаємодії між викладачем і кількома потоками студентів перетворюється на складну ручну роботу з адміністрування, де кожен студент змушений самостійно створювати робочі простори та вручну запрошувати туди керівників, що призводить до плутанини, людських помилок та повної відсутності автоматизації освітньої звітності.

Третім об'єктом аналізу є система RescueTime, яка представляє кардинально інший, автоматизований підхід до обліку часу та оцінки продуктивності [5]. На відміну від попередніх аналогів, які вимагають від користувача усвідомленої ручної дії для запуску чи зупинки таймера, RescueTime функціонує як фоновий системний агент (трекер активності), що встановлюється безпосередньо в операційну систему персонального комп'ютера або мобільного пристрою. Архітектурне рішення платформи базується на безперервному перехопленні системних подій, моніторингу фокусних вікон активних додатків та аналізі URL-адрес відвідуваних веб-сайтів у браузерах. Отримані масиви даних передаються на сервер, де за допомогою інтелектуальних алгоритмів класифікації автоматично розподіляються за категоріями продуктивності, такими як «Розробка програмного забезпечення», «Соціальні мережі», «Комунікації» чи «Розваги». Інтерфейс RescueTime надає користувачу щоденний або щотижневий індекс продуктивності (Pulse Score), наочно демонструючи відсоткове співвідношення між корисною роботою та відволікаючими факторами, як це зображено на рисунку 1.4.

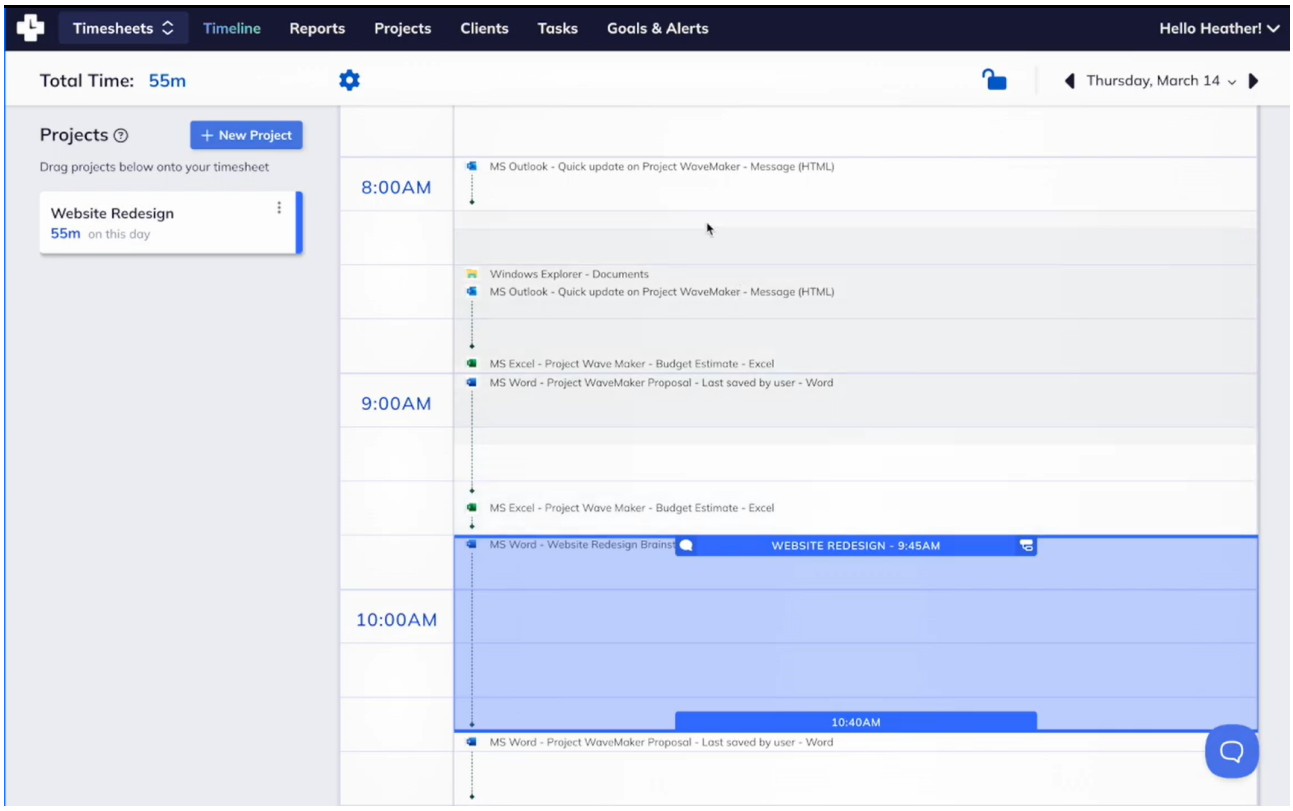

Рисунок 1.4 – Панель автоматичного моніторингу та аналізу продуктивності RescueTime

Головною сильною стороною RescueTime є повне нівелювання людського фактора, оскільки система працює автономно і не дозволяє користувачу забути увімкнути таймер або сфальсифікувати записи ручним введенням. Проте автоматизована концепція RescueTime створює низку критичних недоліків, які роблять її неприйнятною для використання в закладах вищої освіти. По-перше, виникає серйозна проблема порушення приватності особистого життя: фоновий моніторинг абсолютно всіх процесів на комп'ютері студента викликає сильний психологічний спротив та є неприпустимим з етичної та юридичної точок зору. По-друге, автоматичні алгоритми класифікації не здатні зрозуміти контекст подій. Наприклад, якщо студент проводить три години на відеохостингу YouTube, система RescueTime автоматично маркує цей час як розваги та знижує індекс продуктивності, хоча в цей момент студент міг переглядати лекційний матеріал або туторіал з Flask та SQLAlchemy для виконання лабораторної роботи. Крім того, RescueTime фіксує лише загальну активність в ОС, але не має

жодних механізмів для прив'язки цієї активності до конкретної навчальної дисципліни університету, що робить неможливим формування предметної відомості для викладача.

1.3 Постановка завдання

З урахуванням проведеного аналізу предметної області та існуючих програмних рішень, завдання на розробку інформаційної системи обліку навчального часу здобувачів освіти «EduTimeTracker» передбачає проєктування, програмну реалізацію та комплексне тестування веб-орієнтованого додатку з трирівневою архітектурою [6]. Програмний продукт має базуватися на сучасному та надійному технологічному стеку, де як основний інструмент побудови серверної логіки виступає фреймворк Flask на мові програмування Python [7, 8], а управління реляційною базою даних здійснюється за допомогою об'єктно-реляційного відображення SQLAlchemy [9, 10]. Клієнтська частина повинна мати адаптивний та ергономічний інтерфейс, реалізований за допомогою технологій HTML5, CSS3 та компонентів інструментарію Bootstrap [11], що забезпечить коректне відображення на мобільних пристроях та настільних комп'ютерах.

В межах архітектури додатку необхідно спроєктувати та реалізувати підсистему безпеки та автентифікації користувачів, що базується на жорсткій ієрархічній рольовій моделі із розмежуванням прав доступу та управлінні сесіями користувачів [12]. Для забезпечення високого рівня захисту персональних даних та запобігання несанкціонованому доступу до внутрішнього контуру системи, паролі користувачів у базі даних у жодному разі не повинні зберігатися у відкритому вигляді, що вимагає впровадження алгоритмів криптографічного незворотного хешування на етапі реєстрації. У системі мають функціонувати три чітко розмежовані ролі користувачів, кожна з яких отримує унікальний набір повноважень та специфічний інтерфейс.

Першим рівнем доступу володіє адміністратор системи, до обов'язків якого належить глобальне управління конфігурацією платформи, створення та модерація облікових записів викладачів, формування глобального довідника факультетів, кафедр, академічних груп та налаштування зв'язків між ними. Другим суб'єктом взаємодії виступає викладач, який володіє правами на створення та адміністрування підконтрольних навчальних дисциплін, закріплення їх за конкретними групами студентів, перегляд розширеної консолідованої аналітики, фільтрацію даних за часовими проміжками та експорт зведених відомостей про роботу студентів. Кінцевим користувачем системи є студент, який отримує доступ до персонального кабінету, де реалізовано функціонал запуску та зупинки інтерактивного таймера, вибору поточної навчальної дисципліни із динамічного списку закріплених за його групою курсів, ручного створення й редагування тайм-логів, додавання детальних текстових коментарів до виконаних завдань та перегляду особистих графіків продуктивності.

Для забезпечення можливості проведення повноцінного навантажувального, функціонального та інтерфейсного тестування інформаційної системи на етапі розробки, обов'язковим завданням є створення спеціалізованого програмного модуля автоматичного заповнення бази даних тестовою інформацією. Логіка сидера має базуватися на використанні бібліотеки Faker для генерації унікальних, реалістичних масивів даних, включаючи випадкові імена студентів, назви дисциплін, текстові описи лабораторних робіт та відповідні часові інтервали. При цьому скрипт сидингу повинен гарантувати коректність зв'язків між таблицями бази даних, автоматично генерувати та вносити захешовані паролі для тестових профілів усіх трьох ролей, забезпечуючи розробника можливістю миттєвого розгортання повністю працездатного середовища з демонстраційними даними для перевірки аналітичних звітів та графіків.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальна структура веб-орієнтованої інформаційної системи

Інформаційна система обліку робочого часу здобувачів освіти «EduTimeTracker» проєктується та реалізується у вигляді вебдодатка на базі класичної клієнт-серверної трирівневої архітектури з використанням високоефективної мови програмування Python та мікрофреймворку Flask. Організаційну структуру розроблюваної системи можна розділити на три основні взаємопов'язані рівні, що забезпечують сувору ізоляцію логічних процесів: інтерфейс користувача (Front-end), серверну логіку додатка (Back-end) та реляційний рівень збереження даних (Data layer), побудований на базі компактної та надійної СУБД SQLite [13]. Такий підхід дозволяє відокремити логіку представлення даних від безпосередньої обробки бізнес-правил та збереження інформації, що суттєво підвищує масштабованість, гнучкість та безпеку програмного продукту.

Інтерфейс користувача відповідає за візуальне відображення інформації, збір метрик та безпосередню інтерактивну взаємодію з користувачем через сучасні веббраузери. Клієнтська частина реалізована з використанням адаптивного CSS-фреймворку Bootstrap 5, що гарантує коректне та ергономічне відображення робочих просторів під мобільні пристрої, планшети та настільні комп'ютери [11]. Генерація динамічних вебсторінок та безпосередня інтеграція обчислених даних із серверної частини виконується за допомогою вбудованого двигуна шаблонів Jinja2, який дозволяє безпечно та гнучко підставляти об'єкти з бази даних безпосередньо в HTML-макети на етапі рендерингу сторінки сервером [14].

Серверна логіка додатка виступає центральним ядром системи, забезпечуючи обробку вхідних HTTP-запитів, валідацію форм користувача за допомогою розширення Flask-WTF [15] та виконання основних операцій бізнес-

логики. Бекенд-компонент, написаний на мові Python, повністю координує процеси безпечної автентифікації користувачів, розмежування прав доступу на основі захищених сесій за допомогою інструменту Flask-Login [12], точну фіксацію сигналів інтерактивного таймера, логування навчальної активності студентів та агрегацію розширених звітів для професорсько-викладацького складу.

Основний процес взаємодії з веб-орієнтованою інформаційною системою можна описати наступним чином:

1) Користувач проходить процес автентифікації в системі та отримує доступ до робочого простору відповідно до своєї ролі (Адміністратор, Викладач або Студент).
2) Студент обирає навчальну дисципліну та запускає інтерактивний таймер для фіксації часу самостійної роботи, додаючи за потреби опис виконаного завдання.
3) Викладач здійснює централізований моніторинг часових витрат, переглядає статистику залученості та аналізує звіти закріплених за ним академічних груп.
4) Адміністратор здійснює керування обліковими записами, додає нові дисципліни й групи та налаштовує взаємозв'язки між ними в системі.

Таким чином, веб-орієнтована інформаційна система забезпечує зручний та адаптивний інтерфейс для моніторингу СРС, надає гнучкі інструменти для аналізу даних викладачами та гарантує надійне збереження академічної інформації.

2.2 UML-діаграма варіантів використання

Для розробки веб-орієнтованої інформаційної системи обліку робочого часу здобувачів освіти було створено UML-діаграму варіантів використання (Рисунок 2.1), яка описує основні сценарії взаємодії користувачів із системою. Ця діаграма дозволяє зрозуміти, які функції повинні бути реалізовані в

інформаційній системі та які дії можуть виконувати різні типи користувачів.

Основними акторами системи є Студент (Student), Викладач (Teacher) та Адміністратор (Administrator). Кожен із них має унікальний набір прав та можливостей для взаємодії з платформою.

Основні варіанти використання інформаційної системи:

- Фіксація та облік часу навчання: Користувач з роллю «Студент» обирає поточну навчальну дисципліну та запускає інтерактивний таймер; система фіксує тривалість сесії самостійної роботи та зберігає детальний логін-запис у базі даних.
- Перегляд особистої аналітики СРС: Користувач з роллю «Студент» переходить до розділу статистики; система динамічно зчитує історію його запусків таймера і генерує графіки та діаграми розподілу часу за дисциплінами чи вибраними періодами.
- Моніторинг самостійної роботи груп: Користувач з роллю «Викладач» отримує доступ до аналітичних таблиць; система підтягує записи студентів та формує консолідовані звіти з сумарним часом виконання СРС для контролю навчального процесу.
- Адміністрування користувачів, груп та дисциплін: Користувач з роллю «Адміністратор» здійснює додавання, редагування та видалення облікових записів студентів і викладачів, а також налаштовує взаємозв'язки між академічними групами та предметами.
- Автентифікація в системі: Базовий сервіс безпеки, який обов'язково включається (<<include>>) у виконання будь-якого з вищеперерахованих функціональних сценаріїв. Користувач вводить логін і пароль, а система перевіряє права доступу для створення захищеної сесії.

Діаграма варіантів використання (Рисунок 2.1) відображає ключові функції, які необхідні для ефективного обліку робочого часу здобувачів освіти

за допомогою веб-орієнтованої інформаційної системи.

Нижче зображено UML-діаграму варіантів використання:

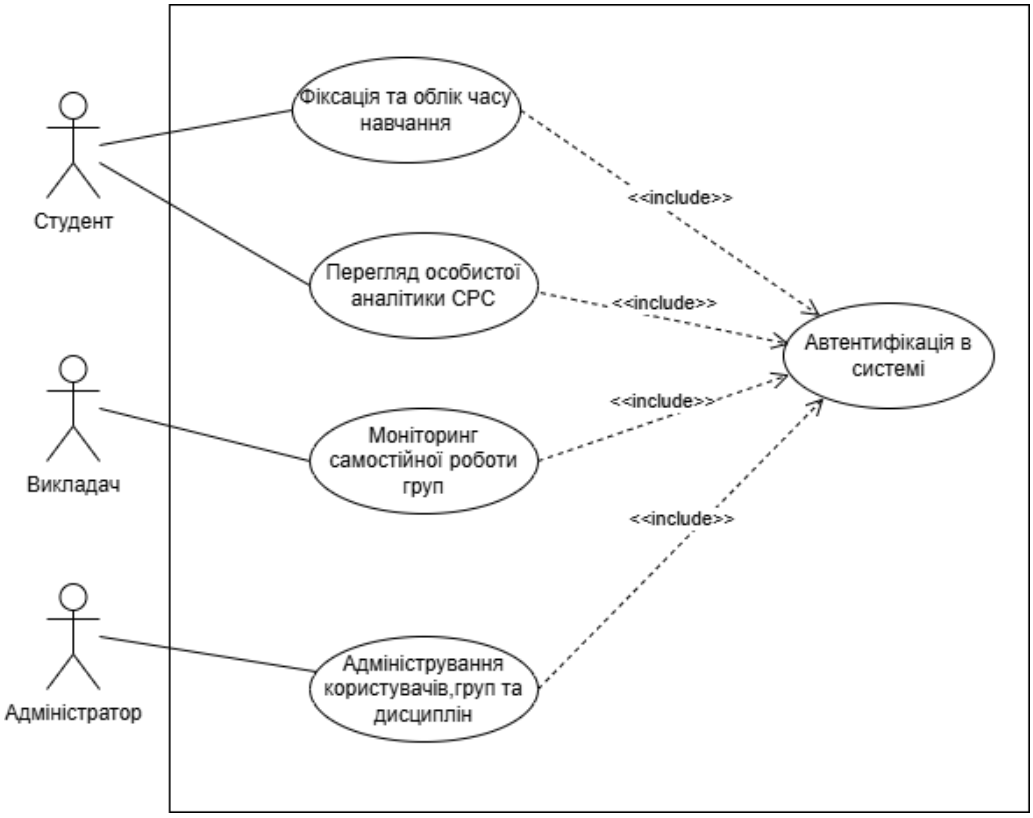

Рисунок 2.1 – UML-діаграма варіантів використання інформаційної системи обліку робочого часу здобувачів освіти

Дана діаграма допомагає зрозуміти, як різні типи користувачів взаємодіють з інформаційною системою, і які основні процеси мають бути реалізовані для забезпечення зручного та інтуїтивного користувацького досвіду. Цей підхід дозволяє планувати подальшу розробку системи, враховуючи всі необхідні аспекти, обмеження рольової моделі та функціональні вимоги.

2.3 Обґрунтування та розробка логічної структури таблиць бази даних

Для забезпечення надійного збереження, цілісності та несуперечності інформації у веб-орієнтованій системі «EduTimeTracker» було обрано реляційну модель бази даних, управління якою здійснюється за допомогою системи керування базами даних SQLite.

Особливістю архітектури даного проєкту є використання технології ORM (Object-Relational Mapping) за допомогою бібліотеки Flask-SQLAlchemy. Це дозволяє відмовитися від ручного створення таблиць через SQL-запити. Натомість структура бази даних проєктується безпосередньо у головному програмному файлі app.py у вигляді Python-класів (моделей), які автоматично транслюються ORM-компонентом у фізичні таблиці бази даних. Розглянемо детальніше розробку кожної таблиці бази даних інформаційної системи.

Структуру моделі User подано в таблиці 2.1.

Таблиця 2.1 – Структура таблиці User

Ключове поле	**Назва стовпця**	**Тип даних**	**Довжина**	**Дозволяє NULL**	**Вміст**
☑	id	INTEGER			Містить унікальний ідентифікатор користувача
	email	VARCHAR	120		Електронна пошта користувача
	password_hash	VARCHAR	255		Хешований пароль користувача для безпечної автентифікації
	first_name	VARCHAR	50		Ім'я користувача
	last_name	VARCHAR	50		Прізвище користувача
	role	VARCHAR	20		Роль користувача в системі (admin, teacher, student)
	group_id	INTEGER		☑	Зовнішній ключ для зв'язку з групою (тільки для студентів)

User – модель користувачів. Ця таблиця описує облікові записи всіх суб'єктів системи та забезпечує функціонування рольової політики (Адміністратор, Викладач, Студент) й механізм безпечної автентифікації. Вона використовує поле id як первинний ключ, а також має зовнішній ключ group_id,

який пов'язує її з таблицею академічних груп (тільки для користувачів із роллю студента). Її структура відповідає принципам третьої нормальної форми (3НФ), адже всі неключові атрибути чітко залежать від первинного ключа, кожен стовпець містить лише одне атомарне значення, а неключові атрибути не мають транзитивних залежностей один від одного.

Програмний опис моделі в архітектурі веб-застосунку:

```
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True,
    nullable=False)
    password = db.Column(db.String(255), nullable=False)
    full_name = db.Column(db.String(100), nullable=False)
    role = db.Column(db.String(20), nullable=False)
    group_id = db.Column(db.Integer, db.ForeignKey('group.id'),
    nullable=True)

    group = db.relationship('Group', back_populates='students')
```

Структуру моделі Group подано в таблиці 2.2.

Таблиця 2.2 – Структура таблиці Group

Ключове поле	Назва стовпця	Тип даних	Довжина	Дозволяє NULL	Вміст
☑	id	INTEGER			Унікальний ідентифікатор академічної групи
	name	VARCHAR	20		Унікальна назва академічної групи

Group – модель академічних груп. Ця таблиця призначена для опису студентських колективів з метою спрощення логіки моніторингу СРС та формування консолідованої звітності викладачами. Вона використовує поле id як первинний ключ і не містить безпосередніх зовнішніх ключів, оскільки зв'язок зі студентами реалізовано на рівні моделі User. Структура таблиці повністю задовольняє вимоги 3НФ, оскільки назва групи однозначно визначається її

ідентифікатором і не залежить від інших потенційних параметрів.

Програмний опис моделі в архітектурі веб-застосунку:

```
class Group(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(20), unique=True, nullable=False)
    students = db.relationship('User', back_populates='group',
lazy=True)
```

Subject_group – допоміжна таблиця зв'язку. Ця технічна таблиця необхідна для декомпозиції та забезпечення реляційного зв'язку типу «багато-до-багатьох» (Many-to-Many) між навчальними дисциплінами та академічними групами, оскільки одна група вивчає багато предметів, а один предмет може читатися для багатьох груп. Таблиця не має власного автоінкрементного ключа, замість цього вона використовує складений первинний ключ, який одночасно складається з двох зовнішніх ключів: subject_id (посилання на дисципліну) та group_id (посилання на групу). Така структура є повністю нормалізованою. Структуру таблиці зв'язку subject_group подано в таблиці 2.3.

Таблиця 2.3 – Структура таблиці subject_group

Ключове поле	**Назва стовпця**	**Тип даних**	**Довжина**	**Дозволяє NULL**	**Вміст**
☑	subject_id	INTEGER			Зовнішній ключ, що посилається на subject.id
☑	group_id	INTEGER			Зовнішній ключ, що посилається на group.id

Програмний опис моделі в архітектурі веб-застосунку:

```
subject_group = db.Table('subject_group',
    db.Column('subject_id',db.Integer,db.ForeignKey('subject.id')
    , primary_key=True),
    db.Column('group_id', db.Integer, db.ForeignKey('group.id'),
primary_key=True) )
```

Subject – модель навчальних дисциплін. Ця таблиця містить перелік

предметів, за якими студенти здійснюють фіксацію часу самостійної роботи. Первинним ключем виступає поле id. Модель містить зовнішній ключ teacher_id, який пов'язує кожну дисципліну з конкретним викладачем із таблиці User. Структура таблиці спроєктована у 3НФ, оскільки атрибут назви предмету залежить виключно від його ідентифікатора. Структуру моделі Subject подано в таблиці 2.4.

Таблиця 2.4 – Структура таблиці Subject

Ключове поле	**Назва стовпця**	**Тип даних**	**Довжина**	**Дозволяє NULL**	**Вміст**
☑	id	INTEGER			Унікальний ідентифікатор навчальної дисципліни
	name	VARCHAR	150		Назва навчального предмета / дисципліни
	teacher_id	INTEGER			Зовнішній ключ для зв'язку з викладачем (user.id)

Програмний опис моделі в архітектурі веб-застосунку:

```
class Subject(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    teacher_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
    teacher = db.relationship('User', backref='subjects')
    groups = db.relationship(
        'Group',
        secondary='subject_group',
        backref='subjects'
    )
```

SrsLog – модель логів самостійної роботи студентів. Це головна транзакційна таблиця системи, яка динамічно накопичує дані про кожну індивідуальну навчальну сесію, фіксує текстовий опис поточного завдання та забезпечує роботу інтерактивного таймера з урахуванням тривалості пауз. Первинним ключем є поле id. Модель містить два логічні зовнішні ключі:

student_id (ідентифікація студента) та subject_id (ідентифікація обраного предмета). Незважаючи на велику кількість атрибутів часу та стану пауз, всі вони характеризують виключно одну конкретну сесію таймера, що забезпечує повну відповідність вимогам третьої нормальної форми (3НФ). Структуру моделі SrsLog подано в таблиці 2.5.

Таблиця 2.5 – Структура таблиці SrsLog

Ключове поле	**Назва стовпця**	**Тип даних**	**Довжина**	**Дозволяє NULL**	**Вміст**
☑	id	INTEGER			Унікальний ідентифікатор логу сесії СРС
	student_id	INTEGER			Зовнішній ключ, що вказує на студента
	subject_id	INTEGER			Зовнішній ключ, що вказує на дисципліну
	task_description	TEXT			Короткий опис самостійного завдання
	start_time	DATETIME			Точний час запуска таймера
	end_time	DATETIME		☑	Час зупинки таймера
	is_paused	BOOLEAN			Поточний статус паузи таймера
	pause_start	DATETIME		☑	Час, коли користувач натиснув на паузу
	total_paused_seconds	INTEGER			Сумарна тривалість усіх пауз сесії у секундах

Програмний опис моделі в архітектурі веб-застосунку:

```
class SrsLog(db.Model):
     id = db.Column(db.Integer, primary_key=True)
     student_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
     subject_id = db.Column(db.Integer,
db.ForeignKey('subject.id'), nullable=False)
     task_description = db.Column(db.Text, nullable=False)
     start_time = db.Column(db.DateTime, default=datetime.now)
     end_time = db.Column(db.DateTime, nullable=True)
     is_paused = db.Column(db.Boolean, default=False)
     pause_start = db.Column(db.DateTime, nullable=True)
     total_paused_seconds = db.Column(db.Integer, default=0)
     student = db.relationship('User', backref='srs_logs')
     subject = db.relationship('Subject', backref='srs_logs')
```

Розроблені моделі та таблиці бази даних забезпечують ефективну організацію, структурування та надійне збереження інформації у веб-орієнтованій інформаційній системі обліку часу самостійної роботи студентів. Вони дозволяють чітко розмежовувати дані про користувачів та їхні рольові права, академічні групи, закріплені навчальні дисципліни, а також накопичувати детальні логи інтерактивного таймера СРС із точним врахуванням тривалості пауз. Завдяки дотриманню принципів третьої нормальної форми (3НФ) та використанню сучасного ORM-підходу на базі Flask-SQLAlchemy, створена структура мінімізує надлишковість даних, запобігає виникненню аномалій модифікації та забезпечує високу цілісність і несуперечність інформації. Це дозволяє сформувати стабільну, гнучку та оптимізовану під швидкі аналітичні запити архітектуру бази даних, яка повністю задовольняє вимоги до сучасних освітніх веб-застосунків.

2.4 Структура бази даних

База даних є ключовим компонентом веб-орієнтованої інформаційної системи обліку часу самостійної роботи студентів «EduTimeTracker», оскільки вона забезпечує збереження та доступ до облікових і транзакційних даних користувачів. Для реалізації бази даних було використано ORM-бібліотеку Flask-SQLAlchemy, яка інтегрується у фреймворк Flask і забезпечує зручний та безпечний спосіб взаємодії з реляційною СКБД SQLite у веб-застосунках. На рисунку 2.2 зображена ERD-діаграма бази даних інформаційної системи:

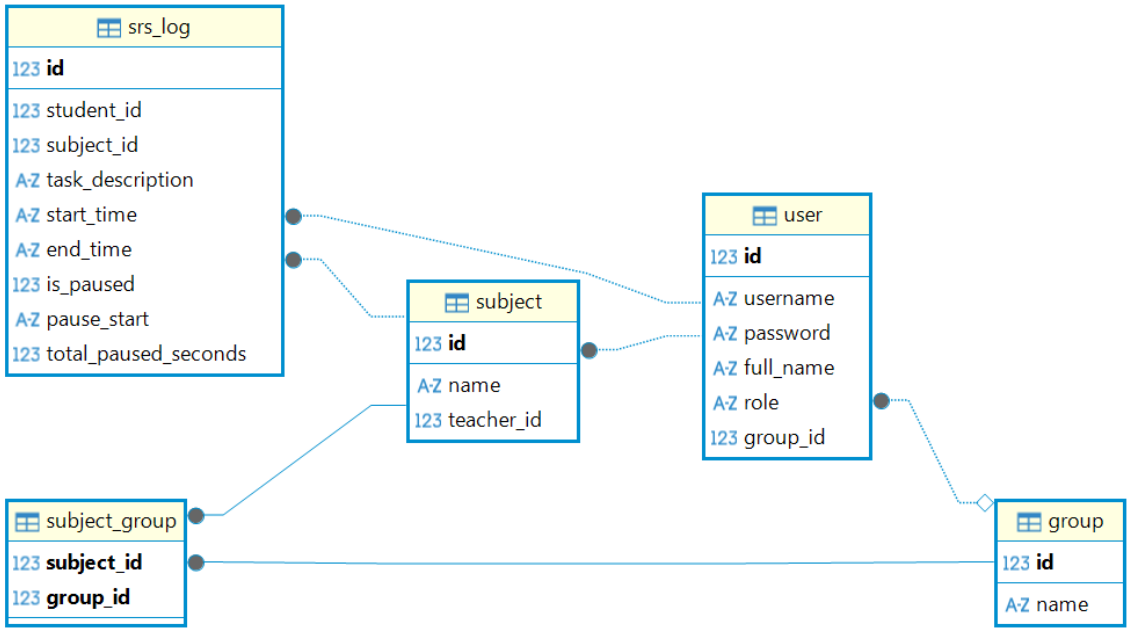

Рисунок 2.2 – ER-діаграма бази даних системи «EduTimeTracker»

База даних, побудована на основі СКБД SQLite та керована за допомогою Flask-SQLAlchemy, забезпечує надійне збереження даних та їхню цілісність. Взаємозв'язки між таблицями дозволяють ефективно організувати інформацію про користувачів, академічні групи й навчальні дисципліни та забезпечують зручний доступ до них для аналізу та відстеження часу самостійної роботи студентів. ER-діаграма відображає основні таблиці та зв'язки між ними, що допомагає краще зрозуміти структуру бази даних і спростити процес її розробки та підтримки.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ «EDUTIMETRACKER»

3.1 Технології розроблення серверної частини (Back-end) системи

Серверна частина (Back-end) інформаційної системи «EduTimeTracker» виступає її фундаментальним логічним ядром, яке забезпечує стабільне функціонування всього веб-застосунку. Вона відповідає за обробку HTTP-запитів від клієнтської сторони, динамічну маршрутизацію, автентифікацію та авторизацію користувачів, обчислення тривалості часових інтервалів самостійної роботи студентів, а також за безпосередню транзакційну взаємодію з рівнем збереження даних. Для програмної реалізації серверної логіки було обрано об'єктно-орієнтовану мову програмування високого рівня Python 3. Вибір цієї мови обґрунтований її високою читабельністю, швидкістю розробки, строгими стандартами безпеки та наявністю потужної екосистеми зрілих бібліотек для веб-програмування, що дозволяє створювати масштабовані та легкі в підтримці архітектурні рішення.

Платформою для створення веб-застосунку став мікрофреймворк Flask, який, на відміну від монолітних інструментів, не нав'язує розробнику жорсткої структури проєкту, є легковагим та дозволяє гнучко інтегрувати лише ті компоненти, які необхідні для вирішення конкретного завдання автоматизації обліку часу. Важливою складовою цього фреймворку є вбудований WSGI-сервер та набір утиліт Werkzeug, який відповідає за маршрутизацію URL та низькорівневу обробку запитів. Окрім базових функцій комунікації, Werkzeug виконує критичну роль у забезпеченні безпеки системи за допомогою вбудованого модуля криптографічного захисту. Модулі генерування та перевірки парольних хешів дозволяють уникати збереження паролів користувачів у відкритому вигляді, замінюючи їх незворотними криптографічними відбитками, що гарантує захист

облікових записів від компрометації навіть у разі потенційного витоку файлу бази даних.

Для ефективної взаємодії з реляційною базою даних було інтегровано шар об'єктно-реляційного відображення за допомогою розширення Flask-SQLAlchemy. Використання ORM дозволяє повністю абстрагуватися від написання сирого SQL-коду, трансформуючи таблиці бази даних у класи мови Python, а їхні рядки — в об'єкти. Це значно підвищує безпеку веб-застосунку, нейтралізуючи загрози ін'єкцій шкідливого коду через форми, та спрощує маніпулювання сутностями, такими як користувачі, академічні групи, навчальні дисципліни та логи часу. Вся інформація накопичується в локальній СКБД SQLite, яка функціонує у строгому транзакційному режимі ACID. Це гарантує цілісність даних: якщо під час фіксації тривалості самостійної роботи студента відбудеться збій на сервері, транзакція автоматично скасується, запобігаючи запису некоректного часу.

Приймання та обробка вхідних даних від клієнтського інтерфейсу реалізована за допомогою розширення Flask-WTF, яке автоматизує валідацію веб-форм і захищає ресурс від міжсайтової підробки запитів (CSRF-атак) за допомогою генерації та перевірки унікальних секретних токенів сесії. Розмежування прав доступу між користувачами різних рольових моделей — Студентами, Викладачами та Адміністраторами — базується на вбудованому механізмі сесій Flask. Сервер шифрує дані сесії за допомогою секретного ключа застосунку і зберігає їх у захищених cookie-файлах браузера. Такий підхід дозволяє серверній частині ідентифікувати користувача під час кожного звернення та гнучко контролювати доступ до маршрутів, надаючи студентам можливість керувати таймером, а викладачам — переглядати аналітику, що робить архітектуру сервера повністю захищеною та готовою до інтеграції з користувацьким інтерфейсом.

3.2 Технології розроблення клієнтської частини (Front-end) системи

Клієнтська частина (Front-end) інформаційної системи «EduTimeTracker» відповідає за безпосередню взаємодію з користувачем, візуалізацію аналітичних даних та забезпечення інтуїтивно зрозумілого інтерфейсу для контролю часу. Оскільки система орієнтована на різні категорії користувачів з відмінними технічними навичками, ключовими вимогами до фронтенд-складової стали висока швидкість завантаження сторінок, адаптивність дизайну під мобільні пристрої та динамічність інтерфейсу під час роботи з таймером. Технологічний стек клієнтської частини базується на класичному поєднанні мов розмітки та стилізації, інтегрованих із вбудованими інструментами серверного рендерингу фреймворку Flask.

Основою для формування динамічних веб-сторінок став рідний для Flask шаблонізатор Jinja2, який дозволяє створювати гнучку структуру користувацького інтерфейсу за допомогою концепції успадкування шаблонів. Використання базового макету та блочної структури унеможливлює дублювання ідентичного HTML-коду для навігаційних панелей та підвалів сайту на різних сторінках. Шаблонізатор забезпечує безпечне вбудовування даних, отриманих від серверної частини, автоматично екрануючи змінні, що запобігає поширенню вразливостей класу XSS (Cross-Site Scripting). Завдяки логічним конструкціям Jinja2 інтерфейс динамічно підлаштовується під роль поточного користувача, відображаючи картки запуску трекера для студентів або розгорнуті таблиці з графіками відвідуваності та статистики для викладачів.

Для стилізації компонентів та забезпечення кросбраузерності було використано сучасну версію компонентного фреймворку Bootstrap 5 разом із базовими стандартами HTML5 та CSS3. Вибір Bootstrap 5 обґрунтований наявністю гнучкої сітки (Grid System) на основі Flexbox, що дозволило реалізувати повну адаптивність системи. Студенти мають можливість комфортно взаємодіяти з таймером обліку часу самостійної роботи не лише з персональних комп'ютерів, а й з екранів смартфонів чи планшетів, без зміщення

елементів керування чи втрати читабельності тексту. Готові класи фреймворку використано для оформлення карток моніторингу, модальних вікон підтвердження дій, навігаційних меню та таблиць з інформацією про навчальні дисципліни, що дозволило зберегти єдину стилістичну концепцію та високу візуальну охайність інтерфейсу без перевантаження коду важкими графічними елементами.

Особливе місце в архітектурі клієнтської частини посідає мова програмування JavaScript, яка забезпечує інтерактивність веб-застосунку в реальному часі. Оскільки основним функціоналом системи є трекінг часу, реалізація візуального відліку секунд і хвилин виконана на стороні клієнта за допомогою асинхронних механізмів та таймерів JavaScript. Це дозволяє інтерфейсу динамічно оновлювати цифри на екрані без постійного надсилання запитів на сервер та без перезавантаження сторінки, суттєво знижуючи навантаження на мережу та забезпечуючи плавний користувацький досвід. JavaScript також керує станами елементів інтерфейсу, блокуючи або активуючи кнопки старту, паузи та зупинки таймера, та за допомогою асинхронних HTTP-запитів (API-інструментів) передає точні часові мітки на сервер для подальшої фіксації в базі даних, створюючи цілісну та чутливу взаємодію між користувачем і системою.

3.3 Програмна реалізація бізнес-логіки, маршрутизації та розмежування прав доступу

Практичний етап створення інформаційної системи «EduTimeTracker» полягав у безпосередньому програмуванні функціональних маршрутів веб-застосунку, побудові логіки автентифікації користувачів та розмежуванні їхніх прав доступу відповідно до визначених рольових моделей. Програмний комплекс реалізовано у межах єдиного координаційного файлу app.py, що дозволяє централізовано керувати обробкою запитів, взаємодією моделей даних та контролем сесій користувачів. Для забезпечення безпеки та менеджменту

станів вхідних з'єднань застосовано розширення Flask-Login, яке інтегрує в систему об'єкт LoginManager, призначений для відстеження активних сесій та автоматичного перенаправлення неавтентифікованих запитів на сторінку авторизації. Базовий процес перевірки автентичності реалізовано через маршрут /login, який приймає дані методом POST, виконує пошук відповідного запису в базі даних за унікальним іменем користувача (Email) та зіставляє введені облікові дані.

Важливою архітектурною складовою захисту системи є механізм рольової маршрутизації, який виконує автоматичний розподіл користувачів після успішного входу в систему. Ця функціональність покладена на спеціалізовану допоміжну функцію redirect_by_role, яка аналізує строкове значення атрибута ролі поточного користувача, повертає відповідну адресу перенаправлення. Реалізація цієї функції програмним кодом виглядає наступним чином:

```
def redirect_by_role(role):
    if role == 'admin':
        return redirect(url_for('admin_dashboard'))
    elif role == 'teacher':
         return redirect(url_for('teacher_dashboard'))
    elif role == 'student':
         return redirect(url_for('student_dashboard'))
    return redirect(url_for('login'))
```

Розмежування прав доступу (авторизація) на стороні сервера реалізовано шляхом поєднання стандартного декоратора @login_required та внутрішніх умовних конструкцій, які перевіряють приналежність користувача до конкретної адміністративної чи викладацької групи. Якщо користувач із роллю студента спробує вручну ввести в адресному рядку браузера шлях до панелі адміністратора /admin/dashboard або викладача /teacher/dashboard, сервер перехопить цей запит, виконає умовну перевірку if current_user.role != 'admin' і миттєво припинить обробку, повернувши клієнту стандартну помилку доступу

HTTP 403 Forbidden. Такий підхід гарантує повну ізоляцію даних та унеможливлює несанкціоновану модифікацію інформації.

Центральною та найважливішою частиною бізнес-логіки всього проєкту є програмний модуль обліку часу самостійної роботи студентів, що складається з чотирьох взаємопов'язаних процесів: ініціалізації сесії трекінгу, фіксації паузи, відновлення відліку та повної зупинки таймера із збереженням результатів. Процес запуску таймера реалізовано за допомогою маршруту /add, який обробляє POST-запити від форми вибору навчальної дисципліни. Перед створенням нового запису в таблиці SrsLog сервер виконує обов'язкову валідацію на наявність уже відкритої часової сесії цього конкретного студента, для якої поле end_time залишається пустим (None). Якщо таку сесію виявлено, система блокує створення дублікату та сповіщає користувача про необхідність завершити поточну роботу, що запобігає викривленню статистичних даних.

Особливу складність при розробці становила реалізація функціоналу зупинки на паузу та розрахунку чистого часу самостійної роботи з виключенням періодів бездіяльності. Ця задача була успішно вирішена за рахунок впровадження у модель SrsLog додаткових полів: логічного прапорця is_paused, мітки початку паузи pause_start та лічильника акумульованих секунд паузи total_paused_seconds. Нижче наведено програмний код маршрутів, які відповідають за фіксацію станів паузи та відновлення роботи користувача.

```
@app.route('/pause/<int:log_id>')
@login_required
def pause_log(log_id):
     log = SrsLog.query.get_or_404(log_id)
     if log.student_id == current_user.id and not log.is_paused:
          log.is_paused = True
          log.pause_start = datetime.now()
          db.session.commit()
     return redirect(url_for('student_dashboard'))
```

```
@app.route('/resume/<int:log_id>')
@login_required
def resume_log(log_id):
     log = SrsLog.query.get_or_404(log_id)
     if log.student_id == current_user.id and log.is_paused:
         duration = (datetime.now() -
     log.pause_start).total_seconds()
         log.total_paused_seconds += int(duration)
         log.is_paused = False log.pause_start = None
         db.session.commit()
     return redirect(url_for('student_dashboard'))
```

Під час активації маршруту /pause поточний час фіксується у полі pause_start, а прапорець переводиться в положення істини. При відновленні роботи через маршрут /resume сервер обчислює різницю між поточним моментом часу та міткою pause_start у секундах, додає отримане значення до загальної суми пройдених секунд пауз total_paused_seconds та очищує тимчасові змінні для підготовки до можливих наступних зупинок.

Завершення сесії обліку відбувається через маршрут /stop, де полі end_time присвоюється фінальна мітка часу. Під час генерації звітів для викладачів чи самих студентів загальна тривалість самостійної роботи обчислюється як різниця між end_time та start_time за вирахуванням значення total_paused_seconds. Такий математично-програмний алгоритм забезпечує абсолютну точність моніторингу, нівелює вплив людського фактору (забутий увімкнений таймер) і формує прозору аналітичну базу для викладацького складу.

3.4 Програмна реалізація модулів аналітики та кабінетів користувачів

Для забезпечення повноцінної взаємодії користувачів із накопиченими даними часового трекінгу в інформаційній системі «EduTimeTracker» було спроєктовано та реалізовано розподілену систему персональних кабінетів

(інформаційних панелей/дашбордів). Презентаційний шар вебзастосунку побудовано за класичним шаблоном MVC (Model-View-Controller) за допомогою вбудованого в Flask двигуна шаблонізації Jinja2. Це дозволило відокремити важку обчислювальну логіку бекенду від інтерфейсного представлення, забезпечуючи динамічний рендеринг сторінок на основі контекстних змінних сервера. Кожен кабінет є ізольованим програмним модулем, доступ до якого суворо обмежується перевіркою сесійного об'єкта current_user.

Першим ключовим інтерфейсним компонентом є кабінет студента, реалізований через обробник маршруту /student/dashboard. Головне призначення цього модуля — надати здобувачу освіти зручний інструмент для керування таймером у реальному часі та відображення детальної статистики self-study активності. Програмний алгоритм функціонує таким чином: спочатку виконується верифікація ролі користувача, після чого через ORM-запити SQLAlchemy з бази даних вилучається повний перелік навчальних дисциплін для форми вибору, а також історія логів часу, відсортована від найновіших до найдавніших. Особливістю архітектури є виділення окремого об'єкта active_log, який шукає запис із порожнім значенням кінцевого часу (end_time=None), що свідчить про запущений у цей момент трекер.

Для виведення підсумкових блоків аналітики безпосередньо на бекенді ініціалізується порожній словник total_time_by_subject, який циклічно обходить усі завершені сесії студента, обчислює чисту тривалість кожної з них (із математичним вирахуванням накопичених секунд пауз) та акумулює фінальні змінні у розрізі кожної дисципліни. Реалізація аналітичного кабінету студента програмним кодом виглядає наступним чином:

```
@app.route('/student/dashboard')
@login_required
def student_dashboard():
     if current_user.role != 'student':
          return redirect(url_for('login'))
```

```
    subjects = Subject.query.all()
    logs =
SrsLog.query.filter_by(student_id=current_user.id).order_by(SrsLog
.start_time.desc()).all()
    active_log =
SrsLog.query.filter_by(student_id=current_user.id,
end_time=None).first()

    total_time_by_subject = {}
    for log in logs:
        if log.end_time:
            duration = (log.end_time -
        log.start_time).total_seconds() -
        log.total_paused_seconds
            total_time_by_subject[log.subject_id] =
        total_time_by_subject.get(log.subject_id, 0) + max(0,
        duration)
    return render_template('student_dashboard.html',
subjects=subjects, logs=logs, active_log=active_log,
total_time=total_time_by_subject)
```

Графічний інтерфейс кабінету студента (Рисунок 3.1) складається з верхнього навігаційного меню із відображенням профілю користувача та його академічної групи, інтерактивної форми ініціалізації нової сесії СРС, а також центральної таблиці моніторингу поточної та завершеної активності. На фронтенд-стороні шаблон student_dashboard.html використовує отримані дані для динамічної перебудови UI. Якщо active_log існує, інтерфейс блокує кнопку старту у формі вибору дисциплін і відображає у верхній частині таблиці окрему інтерактивну панель керування з кнопками «Пауза/Продовжити» та «Стоп». Залежно від логічного прапорця is_paused, рядок таблиці активного трекера за допомогою CSS-стилів підсвічується зеленим (код активного виконання завдання) або помаранчевим (стан очікування/паузи) кольором, що покращує користувацький досвід (UX).

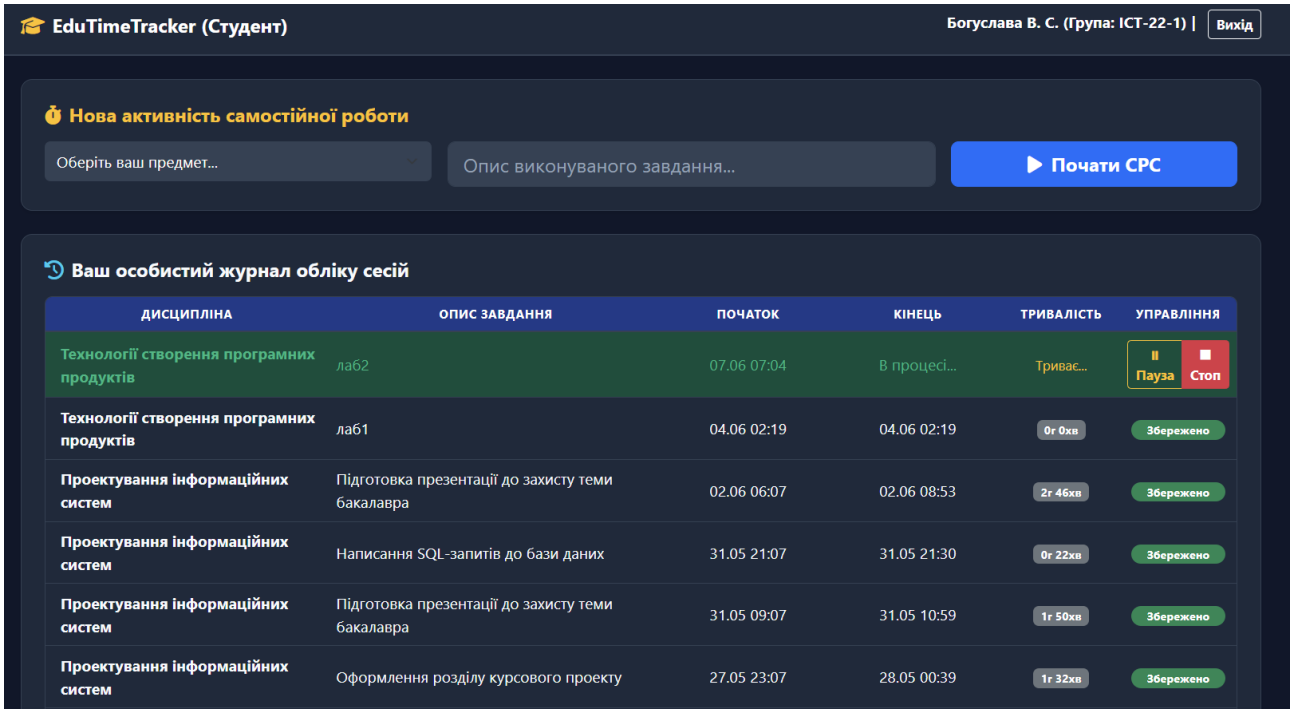

Рисунок 3.1 – Інтерфейс персонального кабінету студента з активним таймером

Другим за важливістю аналітичним ядром системи є кабінет викладача (маршрут /teacher/dashboard). Цей модуль вирішує складну задачу консолідації та фільтрації великих масивів інформації, що надходять від різних студентських груп. Головна вимога до цього кабінету полягала у забезпеченні прозорого контролю за виконанням нормативних годин самостійної роботи. Програмний контролер вилучає з бази даних списки академічних груп, профілі студентів та всі наявні часові логи. На серверній стороні формується багаторівневий масив (або словник) аналітики, який групує загальний чистий час роботи кожного студента. Реалізація логіки кабінету викладача програмним кодом виглядає наступним чином:

```
@app.route('/teacher/dashboard')
@login_required
def teacher_dashboard():
    if current_user.role != 'teacher':
        return redirect(url_for('login'))
```

```
    groups = Group.query.all()
    students = User.query.filter_by(role='student').all()
    all_logs =
    SrsLog.query.order_by(SrsLog.start_time.desc()).all()

    analytics = {}
    for log in all_logs:
        if log.end_time:
            duration = (log.end_time -
        log.start_time).total_seconds() -
        log.total_paused_seconds
            student = User.query.get(log.student_id)
            if student:
                analytics[student.id] =
            analytics.get(student.id, 0) + max(0, duration)

    return render_template('teacher_dashboard.html',
groups=groups, students=students, analytics=analytics)
```

Екранна форма кабінету викладача (Рисунок 3.2) розроблена у вигляді комплексної аналітичної панелі, що містить верхню навігаційну лінію, блок фільтрації та елементів керування, а також розгорнуту зведену таблицю результатів. Візуальне представлення кабінету викладача (teacher_dashboard.html) містить розширений набір елементів керування. Окрім стандартних таблиць виведення, сюди інтегровано інструменти вибору академічної групи або конкретної дисципліни, які взаємодіють із клієнтськими JavaScript-скриптами для миттєвої фільтрації даних без перезавантаження сторінки. Також у шаблоні реалізовано алгоритм підсвічування так званих «боржників» — студентів, чий сумарний час роботи нижчий за встановлений ліміт. Рядки таких студентів автоматично фарбуються у темно-пурпуровий відтінок. Додатково у верхній панелі керування розміщено функціональні

кнопки експорту відомості у форматі CSV та підготовки до друку, що формує завершену структуру вебсторінки.

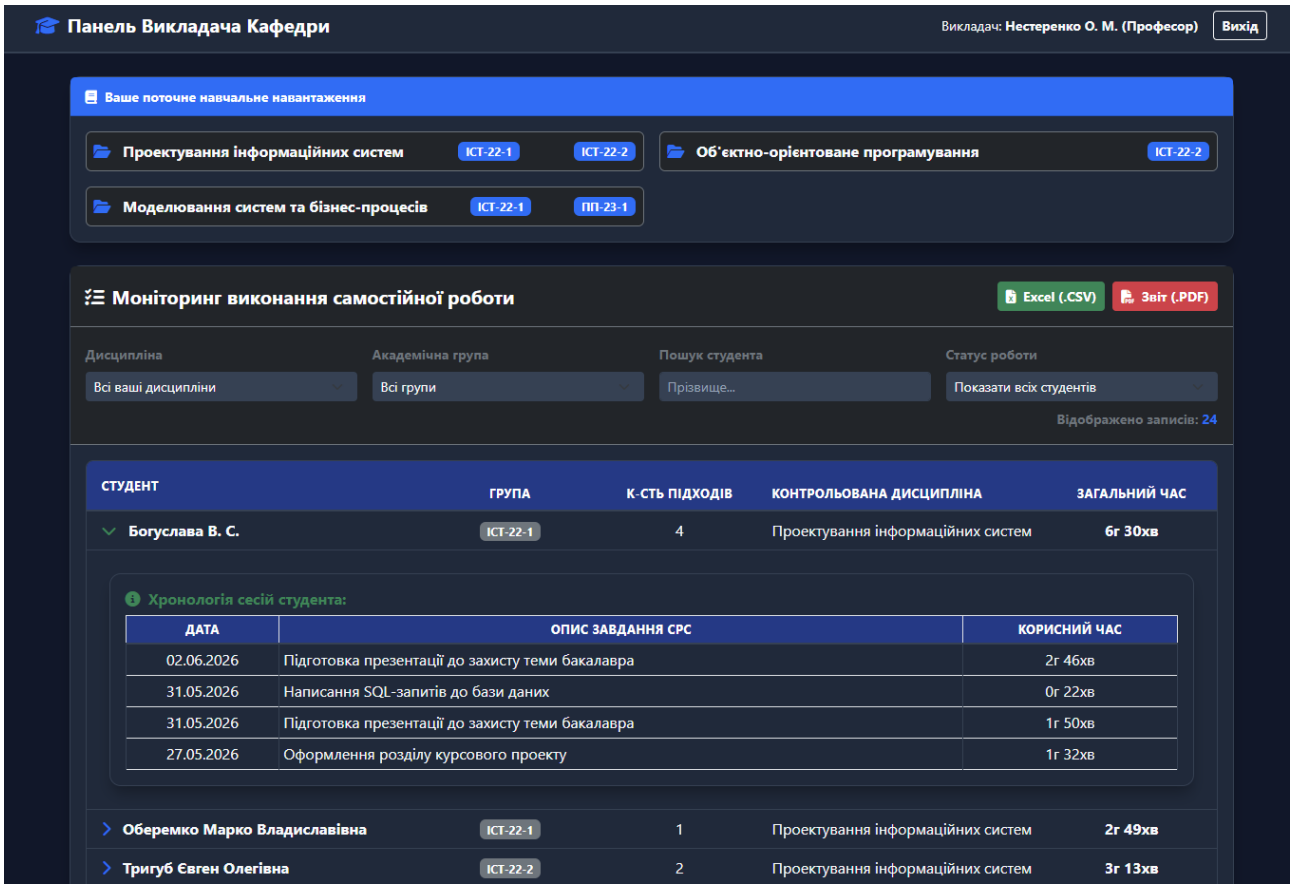

Рисунок 3.2 – Аналітична панель викладача з фільтрацією даних

Для адміністрування глобальних сутностей системи, таких як створення нових навчальних курсів, реєстрація користувачів та закріплення викладачів за дисциплінами, було розроблено кабінет адміністратора (деканат), який функціонує через маршрут /admin/dashboard або пряме керування моделями. Цей інтерфейс забезпечує цілісність даних на початкових етапах семестру. Адміністратор бачить зведену матрицю покриття курсів, де відображаються назви дисциплін, призначені лектори та списки підв'язаних академічних груп. Реалізація логіки виведення адміністративної панелі програмним кодом виглядає наступним чином:

```
@app.route('/admin/dashboard')
@login_required def admin_dashboard():
```

```
    if current_user.role != 'admin':
        return redirect(url_for('login'))
    subjects = Subject.query.all()
    groups = Group.query.all()
    teachers = User.query.filter_by(role='teacher').all()
    return render_template('dashboard_admin.html',
subjects=subjects, groups=groups, teachers=teachers)
```

Візуальний інтерфейс адміністратора (Рисунок 3.3) має лаконічну структуровану архітектуру, орієнтовану на швидкий аудит навчальних курсів. Шаблон dashboard_admin.html використовує компоненти Bootstrap 5 для формування центральної картки-панелі, де розміщено головну таблицю розподілу навантаження кафедри. У стовбцях цієї таблиці послідовно відображаються назва навчальної дисципліни, закріплений за нею викладач та перелік прив'язаних академічних груп. Інтерфейс дозволяє уникнути хаотичного розподілу навантаження завдяки наочному графічному відображенню зв'язків між сутностями бази даних у вигляді компактних кольорових тегів (badges). Якщо для певної дисципліни адміністратор ще не встиг призначити групи, система динамічно виводить у відповідній комірці яскраве червоне попереджувальне повідомлення «Груп не призначено», що відразу привертає увагу оператора та мінімізує людські помилки під час планування навчального процесу.

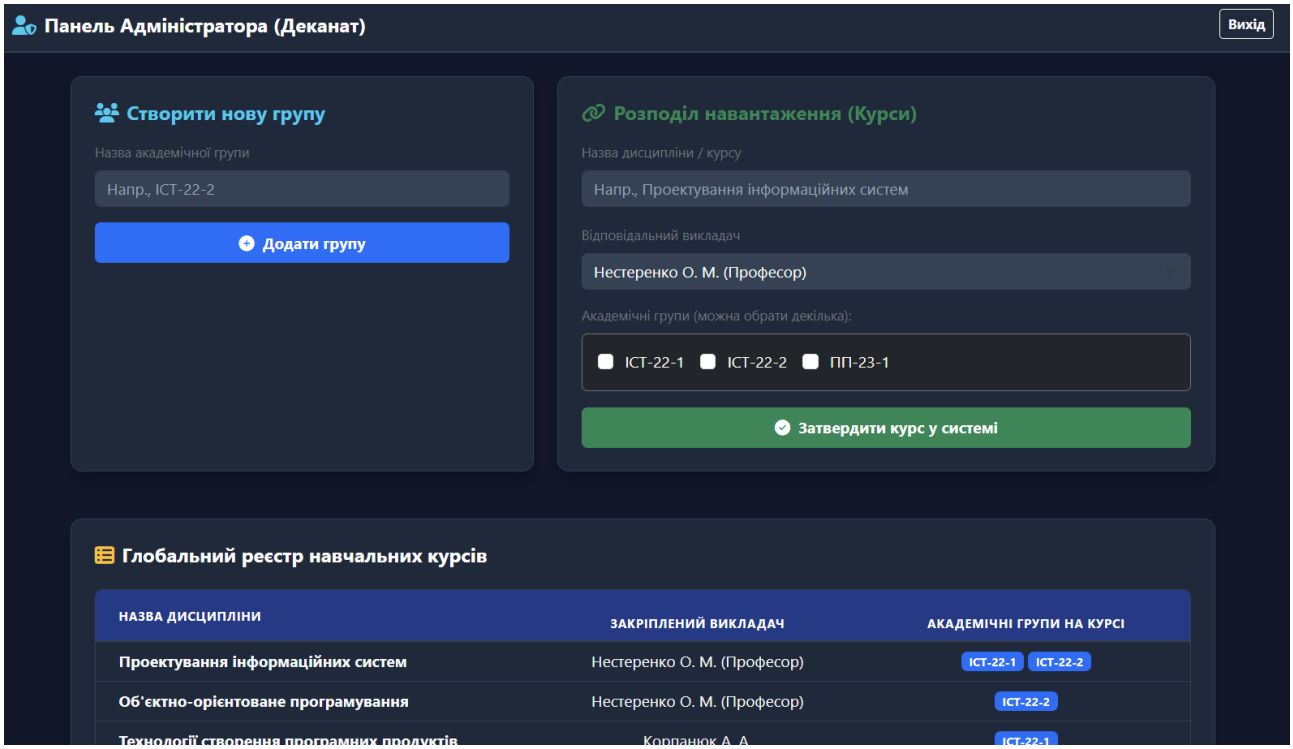

Рисунок 3.3 – Інтерфейс кабінету адміністратора інформаційної системи

Завдяки такій програмній організації кабінетів, інформаційна система «EduTimeTracker» забезпечує повний закритий цикл моніторингу: від моменту генерації логу часу студентом до моменту його глобального аналізу викладачем та контролю з боку адміністрації деканату.

3.5 Тестування працездатності та оптимізація компонентів вебдодатка

Для підтвердження коректності роботи, надійності та відповідності розробленої інформаційної системи «EduTimeTracker» поставленим вимогам було проведено функціональне тестування компонентів вебдодатка. Враховуючи архітектуру системи, основний акцент було зроблено на ручному функціональному тестуванні методом «чорної скриньки», що дозволило перевірити бізнес-логіку застосунку через інтерфейс користувача без аналізу внутрішньої структури коду під час виконання операцій.

Результати виконання функціональних тестів зведено в таблицю 3.1.

Таблиця 3.1 – Матриця результатів функціонального тестування компонентів системи

ID	Назва тест-кейсу	Опис кроків (Вхідні дані)	Очікуваний результат	Фактичний результат	Статус
TC-01	Модуль авторизації (Валідація)	Введення неіснуючого Email або некоректного пароля	Система блокує вхід, виводить флеш-повідомлення про помилку	Доступ заблоковано, відображено alert-повідомлення	**Успішно**
TC-02	Реєстрація користувача	Створення профілю студента із вибором академічної групи	Дані успішно записуються в БД, пароль хэшується, відбувається редірект на логін	Користувача створено, дані збережено в srs_tracker.db	**Успішно**
TC-03	Керування трекером (Студент)	Натискання кнопки «Старт» для обраної дисципліни	У базі створюється новий SrsLog, на UI блокується форма, з'являється панель керування	Запис створено (end_time=None), інтерфейс змінився динамічно	**Успішно**
TC-04	Фіксація пауз у трекері	Натискання кнопки «Пауза» під час активного таймера	Поле is_paused змінюється на True, фіксується час початку паузи, рядок стає помаранчевим	Статус змінено, колір рядка оновлено на помаранчевий	**Успішно**
TC-05	Агрегація даних (Викладач)	Перехід у кабінет викладача, застосування JS-фільтра групи	Таблиця миттєво приховує студентів інших груп без перезавантаження сторінки	Скрипт відфільтрував DOM-елементи таблиці за лічені мілісекунди	**Успішно**
TC-06	Моніторинг лімітів (Боржники)	Завантаження логів студентів із часом, нижчим за кафедральний норматив	Система ідентифікує невідповідність годин і підсвічує рядок темно-пурпуровим кольором	Спрацював CSS-клас .debtor-row, візуальний маркер активний	**Успішно**
TC-07	Цілісність даних (Адміністратор)	Відображення курсу, до якого не прив'язано жодної студентської групи	Система обробляє порожній масив відносин і виводить червоний бейдж «Груп не призначено»	У комірці відображено червоний badge-попередження	**Успішно**

Для типізації перевірок було розроблено та виконано серію тест-кейсів (тестових сценаріїв), які охоплюють критичні шляхи взаємодії користувачів усіх трьох рольових моделей із системою.

Аналіз результатів, наведених у таблиці 3.1, свідчить про те, що всі ключові модулі бізнес-логіки розробленої інформаційної системи «EduTimeTracker» працюють коректно й у повній відповідності до початкових технічних вимог. Завдяки реалізації обробників помилок на стороні сервера та динамічній валідації форм на стороні клієнта, система демонструє стійкість до некоректних дій користувача. Успішне проходження всіх запланованих тест-кейсів підтверджує високу надійність вебдодатка та його готовність до інтеграційних випробувань із підсистемою збереження даних.

ВИСНОВКИ

У в випускній кваліфікаційній роботі було розглянуто і реалізовано процес розробки інформаційної системи обліку часу самостійної роботи студентів «EduTimeTracker». Робота складалася з трьох основних розділів, кожен з яких виконував ключову роль у досягненні поставленої мети автоматизації та моніторингу позааудиторного навчального навантаження.

На початку роботи було проведено детальний аналіз предметної області, що дозволило оцінити особливості обліку часу в умовах кредитно-модульної системи навчання. Було виконано огляд аналогічних комерційних рішень, що допомогло визначити їхні слабкі сторони — відсутність академічної прив'язки до груп та високу вартість ліцензування — і сформулювати основні завдання для розробки нового вебзастосунку. Це включало визначення ключових функціональних вимог, таких як авторизація з розподілом ролей, динамічний запуск і пауза таймера, а також перегляд аналітичної звітності.

Другий розділ роботи був присвячений розробці загальної структури системи та її моделюванню. Було створено UML-діаграму варіантів використання, що допомогло візуалізувати взаємодію трьох рольових моделей (студента, викладача та адміністратора) із системою. Також була розроблена оптимальна структура реляційної бази даних, яка забезпечує надійне збереження та обробку даних користувачів. Загальна структура системи включала як back-end, так і front-end компоненти, обґрунтувавши вибір технологічного стеку на основі мови Python, фреймворку Flask та інструментів Bootstrap 5.

У третьому розділі було детально розглянуто безпосередній процес програмної реалізації та верифікації вебдодатка. Завдяки використанню ORM SQLAlchemy було забезпечено безпечну роботу з базою даних SQLite, а шаблонізатор Jinja2 дозволив створити інтуїтивно зрозумілі інтерфейси користувача. У ході роботи реалізовано кабінет студента з інтерактивним трекером, кабінет викладача з інструментами фільтрації та підсвічування

академічних боржників, а також панель адміністратора з маркерами контролю навчальних курсів, для кожного з яких надано детальний опис функціональності. Працездатність та надійність системи підтверджено успішним інтеграційним тестуванням через модуль заповнення бази даних seed.py та виконанням матриці ручних функціональних тест-кейсів за методом «чорної скриньки».

Значущість отриманих результатів полягає у створенні спеціалізованого та безкоштовного інструменту для цифровізації освітнього середовища ЗВО. Оцінка можливості застосування розробленої системи доводить, що її гнучка архітектура дозволяє легке впровадження на рівні окремих кафедр або факультетів університетів. Очікувана організаційно-технічна ефективність пропозиції виражається у суттєвому скороченні часу викладачів на збір та аналіз паперової чи усної звітності, а також у мінімізації людського фактора під час моніторингу заборгованостей студентів. Напрямками подальшого дослідження у цьому контексті є розробка мобільного застосунку для трекінгу часу на платформах Android та iOS, а також створення модулів прямої інтеграції «EduTimeTracker» із загальноуніверситетськими системами управління навчанням.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Захожай В. Б. Організація самостійної роботи студентів в умовах використання сучасних інформаційно-комунікаційних технологій. Інформаційні технології і засоби навчання, 2020. № 4. С. 112–125.
2. Співаковський О. В., Щедролосєв Д. Є. Інформаційні технології в управлінні навчальним процесом: Навчальний посібник. Херсон: Айлант, 2015. 210 с.
3. Toggl Track Official Website. Time Tracking Software. URL: https://toggl.com/track/ (дата звернення: 05.04.2026).
4. Clockify Official Website. Free Time Tracker for Teams. URL: https://clockify.me/ (дата звернення: 18.04.2026).
5. RescueTime Official Website. Automatic Time Tracking and Productivity Software. URL: https://www.rescuetime.com/ (дата звернення: 22.04.2026).
6. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. O'Reilly Media, 2018. 316 p.
7. Python 3.12.0 documentation. Офіційний сайт мови програмування Python. URL: https://docs.python.org/3/ (дата звернення: 12.05.2026).
8. Flask Documentation (3.0.x). Офіційна документація фреймворку Flask. URL: https://flask.palletsprojects.com/ (дата звернення: 14.05.2026).
9. SQLAlchemy 2.0 Documentation. Object Relational Tutorial. URL: https://docs.sqlalchemy.org/ (дата звернення: 15.05.2026).
10. Flask-SQLAlchemy Documentation. URL: https://flask-sqlalchemy.palletsprojects.com/ (дата звернення: 16.05.2026).
11. Bootstrap 5.3 Documentation. Офіційний сайт фреймворку Bootstrap. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 10.05.2026).
12. Flask-Login Documentation. User session management for Flask. URL: https://flask-login.readthedocs.io/ (дата звернення: 22.05.2026).

13. SQLite Documentation. Офіційний сайт бази даних SQLite. URL: https://www.sqlite.org/docs.html (дата звернення: 18.05.2026).

14. Jinja2 Documentation. Шаблонізатор Jinja. URL: https://jinja.palletsprojects.com/ (дата звернення: 14.05.2026).

15. Flask-WTF Documentation. Simple integration of Flask and WTForms. URL: https://flask-wtf.readthedocs.io/ (дата звернення: 25.05.2026).

ДОДАТКИ

Додаток А

Вміст файлу app.py

```
import os
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, logout_user,
login_required, current_user

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_super_secret_key_12345'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///srs_tracker.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)
    full_name = db.Column(db.String(100), nullable=False)
    role = db.Column(db.String(20), nullable=False)
    group_id = db.Column(db.Integer, db.ForeignKey('group.id'),
nullable=True)

    group = db.relationship('Group', back_populates='students')

class Group(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(20), unique=True, nullable=False)

    students = db.relationship('User', back_populates='group', lazy=True)

subject_group = db.Table('subject_group',
    db.Column('subject_id', db.Integer, db.ForeignKey('subject.id'),
primary_key=True),
    db.Column('group_id', db.Integer, db.ForeignKey('group.id'),
primary_key=True)
)

class Subject(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    teacher_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
```

```
    teacher = db.relationship('User', backref='subjects')

    groups = db.relationship(
        'Group',
        secondary='subject_group',
        backref='subjects'
    )

class SrsLog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    student_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
    subject_id = db.Column(db.Integer, db.ForeignKey('subject.id'),
nullable=False)
    task_description = db.Column(db.Text, nullable=False)
    start_time = db.Column(db.DateTime, default=datetime.now)
    end_time = db.Column(db.DateTime, nullable=True)

    is_paused = db.Column(db.Boolean, default=False)
    pause_start = db.Column(db.DateTime, nullable=True)
    total_paused_seconds = db.Column(db.Integer, default=0)

    student = db.relationship('User', backref='srs_logs')
    subject = db.relationship('Subject', backref='srs_logs')


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

@app.route('/', methods=['GET', 'POST'])
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect_by_role(current_user.role)

    if request.method == 'POST':
        email_input = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(username=email_input).first()

        if user and user.password == password:
            login_user(user)
            return redirect_by_role(user.role)
        else:
```

```
            flash('Неправильне ім\'я користувача або пароль!', 'danger')

    return render_template('login.html')

def redirect_by_role(role):
    if role == 'admin':
        return redirect(url_for('admin_dashboard'))
    elif role == 'teacher':
        return redirect(url_for('teacher_dashboard'))
    elif role == 'student':
        return redirect(url_for('student_dashboard'))
    return redirect(url_for('login'))

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

@app.route('/admin/dashboard')
@login_required
def admin_dashboard():
    if current_user.role != 'admin':
        return "Доступ заборонено", 403
    groups = Group.query.all()
    teachers = User.query.filter_by(role='teacher').all()
    subjects = Subject.query.all()
    return render_template('dashboard_admin.html', groups=groups,
teachers=teachers, subjects=subjects)

@app.route('/admin/create_group', methods=['POST'])
@login_required
def create_group():
    if current_user.role != 'admin':
        return "Доступ заборонено", 403
    group_name = request.form.get('group_name').strip()
    if group_name:
        existing = Group.query.filter_by(name=group_name).first()
        if not existing:
            new_group = Group(name=group_name)
            db.session.add(new_group)
            db.session.commit()
            flash(f'Групу {group_name} успішно створено!', 'success')
        else:
            flash('Така група вже існує!', 'warning')
    return redirect(url_for('admin_dashboard'))
```

```
@app.route('/admin/create_subject', methods=['POST'])
@login_required
def create_subject():
    if current_user.role != 'admin':
        return "Доступ заборонено", 403

    name = request.form.get('subject_name').strip()
    t_id = request.form.get('teacher_id')
    selected_group_ids = request.form.getlist('group_ids')

    if name and t_id and selected_group_ids:
        existing_sub = Subject.query.filter_by(name=name,
teacher_id=int(t_id)).first()

        if existing_sub:
            for g_id in selected_group_ids:
                group_obj = Group.query.get(int(g_id))
                if group_obj and group_obj not in existing_sub.groups:
                    existing_sub.groups.append(group_obj)
            flash(f'До курсу "{name}" додано нові академічні групи!',
'success')
        else:
            new_sub = Subject(name=name, teacher_id=int(t_id))
            for g_id in selected_group_ids:
                group_obj = Group.query.get(int(g_id))
                if group_obj:
                    new_sub.groups.append(group_obj)
            db.session.add(new_sub)
            flash('Новий курс успішно розподілено між групами!', 'success')

        db.session.commit()
    else:
        flash('Помилка: заповніть назву та обов’язково оберіть хоча б одну
групу!', 'danger')

    return redirect(url_for('admin_dashboard'))

@app.route('/teacher/dashboard')
@login_required
def teacher_dashboard():
    if current_user.role != 'teacher':
        return "Доступ заборонено", 403

    my_subjects = Subject.query.filter_by(teacher_id=current_user.id).all()

    my_groups = []
    for s in my_subjects:
```

```
    for g in s.groups:
            if g not in my_groups:
                my_groups.append(g)

    sub_ids = [s.id for s in my_subjects]
    my_logs =
SrsLog.query.filter(SrsLog.subject_id.in_(sub_ids)).order_by(SrsLog.start_ti
me.desc()).all()

    return render_template('dashboard_teacher.html', subjects=my_subjects,
groups=my_groups, logs=my_logs)

@app.route('/student/dashboard')
@login_required
def student_dashboard():
    if current_user.role != 'student':
        return "Доступ заборонено", 403

    student_subjects = []
    if current_user.group_id:
        student_group = Group.query.get(current_user.group_id)
        if student_group:
            student_subjects = student_group.subjects

    student_logs =
SrsLog.query.filter_by(student_id=current_user.id).order_by(SrsLog.start_tim
e.desc()).all()
    active_log = SrsLog.query.filter_by(student_id=current_user.id,
end_time=None).first()

    return render_template('dashboard_student.html',
subjects=student_subjects, logs=student_logs, active_log=active_log)

@app.route('/add', methods=['POST'])
@login_required
def add_log():
    if current_user.role != 'student':
        return "Доступ заборонено", 403
    sub_id = request.form.get('subject_id')
    task = request.form.get('task')
    active = SrsLog.query.filter_by(student_id=current_user.id,
end_time=None).first()
    if active:
        flash('У вас вже є активна сесія! Спочатку завершіть її.',
'warning')
        return redirect(url_for('student_dashboard'))
    if sub_id and task:
```

```
        new_log = SrsLog(student_id=current_user.id, subject_id=int(sub_id),
task_description=task)
        db.session.add(new_log)
        db.session.commit()
    return redirect(url_for('student_dashboard'))

@app.route('/stop/<int:log_id>')
@login_required
def stop_log(log_id):
    log = SrsLog.query.get_or_404(log_id)
    if log.student_id != current_user.id:
        return "Доступ заборонено", 403
    if not log.end_time:
        log.end_time = datetime.now()
        db.session.commit()
    return redirect(url_for('student_dashboard'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect_by_role(current_user.role)
    groups = Group.query.all()
    if request.method == 'POST':
        full_name = request.form.get('full_name')
        email = request.form.get('email')
        password = request.form.get('password')
        role = request.form.get('role')
        group_id = request.form.get('group_id')

        existing = User.query.filter_by(username=email).first()
        if existing:
            flash('Користувач з таким Email вже існує!', 'danger')
            return render_template('register.html', groups=groups)

        new_user = User(
            username=email,
            password=password,
            full_name=full_name,
            role=role,
            group_id=int(group_id) if (role == 'student' and group_id) else
None
        )
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація успішна! Увійдіть.', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', groups=groups)
```

```
@app.route('/pause/<int:log_id>')
@login_required
def pause_log(log_id):
    log = SrsLog.query.get_or_404(log_id)
    if log.student_id == current_user.id and not log.is_paused:
        log.is_paused = True
        log.pause_start = datetime.now()
        db.session.commit()
    return redirect(url_for('student_dashboard'))

@app.route('/resume/<int:log_id>')
@login_required
def resume_log(log_id):
    log = SrsLog.query.get_or_404(log_id)
    if log.student_id == current_user.id and log.is_paused:
        duration = (datetime.now() - log.pause_start).total_seconds()
        log.total_paused_seconds += int(duration)
        log.is_paused = False
        log.pause_start = None
        db.session.commit()
    return redirect(url_for('student_dashboard'))

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

Вміст файлу dashboard_admin.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Панель Адміністратора | Деканат</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>

    <nav class="navbar navbar-dark navbar-custom mb-4">
        <div class="container">
```

```
            <span class="navbar-brand fw-bold"><i class="fa-solid fa-user-
shield me-2 text-info"></i>Панель Адміністратора (Деканат)</span>
             <a href="/logout" class="btn btn-outline-light btn-sm fw-
bold">Вихід</a>
        </div>
    </nav>

    <div class="container" style="max-width: 1200px;">
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for cat, msg in messages %}
                    <div class="alert alert-{{cat}} alert-dismissible fade
show mb-4 small" role="alert">
                        {{ msg }}
                        <button type="button" class="btn-close" data-bs-
dismiss="alert" aria-label="Close"></button>
                    </div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        <div class="row g-4 mb-5">
            <div class="col-md-5">
                <div class="card card-custom h-100 p-4">
                    <h5 class="fw-bold mb-3 text-info"><i class="fa-solid
fa-users me-2"></i>Створити нову групу</h5>
                    <form action="/admin/create_group" method="POST">
                        <div class="mb-3">
                            <label class="form-label small text-
secondary">Назва академічної групи</label>
                            <input type="text" name="group_name"
class="form-control" placeholder="Напр., ICT-22-2" required>
                        </div>
                        <button type="submit" class="btn btn-primary w-100
py-2 fw-bold">
                            <i class="fa-solid fa-plus-circle me-1"></i>
Додати групу
                        </button>
                    </form>
                </div>
            </div>

            <div class="col-md-7">
    <div class="card card-custom h-100 p-4">
        <h5 class="fw-bold mb-3 text-success"><i class="fa-solid fa-link me-
2"></i>Розподіл навантаження (Курси)</h5>
        <form action="/admin/create_subject" method="POST">
```

```
            <div class="mb-3">
                <label class="form-label small text-secondary">Назва
дисципліни / курсу</label>
                <input type="text" name="subject_name" class="form-control"
placeholder="Напр., Проектування інформаційних систем" required>
            </div>
            <div class="mb-3">
                <label class="form-label small text-
secondary">Відповідальний викладач</label>
                <select name="teacher_id" class="form-select" required>
                    {% for t in teachers %}<option
value="{{t.id}}">{{t.full_name}}</option>{% endfor %}
                </select>
            </div>
            <div class="mb-3">
                <label class="form-label small text-secondary d-
block">Академічні групи (можна обрати декілька):</label>
                <div class="p-3 border border-secondary rounded bg-dark d-
flex flex-wrap gap-3">
                    {% for g in groups %}
                    <div class="form-check">
                        <input class="form-check-input" type="checkbox"
name="group_ids" value="{{g.id}}" id="group_{{g.id}}">
                        <label class="form-check-input-label text-white ms-
1" for="group_{{g.id}}">
                            {{g.name}}
                        </label>
                    </div>
                    {% endfor %}
                </div>
            </div>
            <button type="submit" class="btn btn-success w-100 py-2 fw-
bold">
                <i class="fa-solid fa-circle-check me-1"></i> Затвердити
курс у системі
            </button>
        </form>
    </div>
</div>
        </div>

        <div class="card card-custom p-4 mb-5">
    <h5 class="fw-bold mb-4 text-white"><i class="fa-solid fa-table-list me-
2 text-warning"></i>Глобальний реєстр навчальних курсів</h5>
    <div class="table-custom-container">
        <div class="table-responsive">
            <table class="table table-hover table-custom align-middle">
```

```
                <thead>
                    <tr class="text-center">
                        <th class="text-start ps-4 py-3" style="width:
40%;">Назва дисципліни</th>
                        <th style="width: 30%;">Закріплений викладач</th>
                        <th style="width: 30%;">Академічні групи на
курсі</th>
                    </tr>
                </thead>
                <tbody>
                    {% for sub in subjects %}
                    <tr class="text-center">
                        <td class="text-start ps-4 fw-bold text-white">{{
sub.name }}</td>
                        <td class="text-light">{{ sub.teacher.full_name
}}</td>
                        <td>
                            <div class="d-flex flex-wrap justify-content-
center gap-1">
                                {% for g in sub.groups %}
                                <span class="badge bg-primary px-2 py-1
text-uppercase fw-bold">{{ g.name }}</span>
                                {% else %}
                                <span class="badge bg-danger px-2 py-1 fw-
bold">Груп не призначено</span>
                                {% endfor %}
                            </div>
                        </td>
                    </tr>
                    {% else %}
                    <tr>
                        <td colspan="3" class="text-center py-4 text-
secondary">Жодного навчального курсу ще не розподілено.</td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>
        </div>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m
in.js"></script>
</body>
</html>
```

Вміст файлу dashboard_student.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Кабінет Студента | Облік часу</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar navbar-dark navbar-custom mb-4">
        <div class="container">
            <span class="navbar-brand fw-bold"><i class="fa-solid fa-
graduation-cap me-2 text-warning"></i>EduTimeTracker (Студент)</span>
            <span class="navbar-text text-white fw-bold">
                {{ current_user.full_name }} (Група: {{
current_user.group.name if current_user.group else 'Без групи' }}) |
                <a href="/logout" class="btn btn-outline-light btn-sm ms-2
fw-bold">Вихід</a>
            </span>
        </div>
    </nav>

    <div class="container">
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for cat, msg in messages %}
                    <div class="alert alert-{{cat}} mb-4 small py-
2">{{msg}}</div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        <div class="card card-custom shadow-sm mb-4">
            <div class="card-body p-4">
                <h5 class="fw-bold mb-3 text-warning"><i class="fa-solid fa-
stopwatch me-2"></i>Нова активність самостійної роботи</h5>
                <form action="/add" method="POST" class="row g-3">
                    <div class="col-md-4">
                        <select name="subject_id" class="form-select form-
control-lg" required>
```

```
                            <option value="" disabled selected>Оберіть ваш
предмет...</option>
                            {% for sub in subjects %}<option
value="{{sub.id}}">{{sub.name}}</option>{% endfor %}
                        </select>
                    </div>
                    <div class="col-md-5">
                        <input type="text" name="task" class="form-control
form-control-lg" placeholder="Опис виконуваного завдання..." required>
                    </div>
                    <div class="col-md-3">
                        <button type="submit" class="btn btn-primary btn-lg
w-100 fw-bold"><i class="fa-solid fa-play me-1"></i> Почати СРС</button>
                    </div>
                </form>
            </div>
        </div>

        <div class="card card-custom shadow-sm p-4">
            <h5 class="fw-bold mb-3"><i class="fa-solid fa-history me-2
text-info"></i>Ваш особистий журнал обліку сесій</h5>
            <div class="table-custom-container">
                <div class="table-responsive">
                    <table class="table table-hover table-custom mb-0 align-
middle text-center">
                        <thead>
                            <tr>
                                <th style="width: 25%;">Дисципліна</th>
                                <th style="width: 30%;">Опис завдання</th>
                                <th style="width: 15%;">Початок</th>
                                <th style="width: 15%;">Кінець</th>
                                <th style="width: 10%;">Тривалість</th>
                                <th style="width: 5%;">Управління</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for log in logs %}
                            {% set is_active_now = (active_log and
active_log.id == log.id) %}
                            <tr class="{% if is_active_now %}{% if
log.total_paused_seconds > 0 and log.end_time == None and request.endpoint
== 'pause' %}paused-row{% else %}active-row{% endif %}{% endif %}">
                                <td class="fw-bold text-start ps-3">{{
log.subject.name }}</td>
                                <td class="text-start">{{
log.task_description }}</td>
```

```
                                <td>{{ log.start_time.strftime('%d.%m
%H:%M') }}</td>
                                <td>{{ log.end_time.strftime('%d.%m %H:%M')
if log.end_time else 'В процесі...' }}</td>
                                <td>
    {% if log.end_time %}
        {% set duration = (log.end_time - log.start_time).total_seconds() -
log.total_paused_seconds %}
        <span class="badge bg-secondary">{{ (duration // 3600) | int }}г {{
((duration % 3600) // 60) | int }}хв</span>
    {% else %}
        <span class="text-warning small animate-pulse">Триває...</span>
    {% endif %}
</td>

<td>
    {% if not log.end_time %}
        <div class="btn-group btn-group-sm">
            {% if not log.is_paused %}
                <a href="/pause/{{log.id}}" class="btn btn-outline-warning
fw-bold">⏸ Пауза</a>
            {% else %}
                <a href="/resume/{{log.id}}" class="btn btn-outline-success
fw-bold">▶ Продовжити</a>
            {% endif %}
            <a href="/stop/{{log.id}}" class="btn btn-danger fw-bold">⏹
Стоп</a>
        </div>
    {% else %}
        <span class="badge rounded-pill bg-success px-3">Збережено</span>
    {% endif %}
</td>
                            </tr>
                            {% else %}
                            <tr><td colspan="6" class="text-center py-4
text-muted">Журнал вашої самостійної роботи поки що порожній.</td></tr>
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Вміст файлу dashboard_teacher.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Панель Викладача | Контроль СРС</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>

    <nav class="navbar navbar-dark navbar-custom mb-4">
        <div class="container">
            <span class="navbar-brand fw-bold"><i class="fa-solid fa-
graduation-cap me-2 text-primary"></i>Панель Викладача Кафедри</span>
            <div class="d-flex align-items-center">
                <span class="text-light small me-3">Викладач: <strong>{{
current_user.full_name }}</strong></span>
                <a href="/logout" class="btn btn-outline-light btn-sm fw-
bold">Вихід</a>
            </div>
        </div>
    </nav>

    <div class="container" style="max-width: 1250px;">

        <div class="card card-custom no-print mb-4">
            <div class="card-header bg-primary text-white py-2">
                <span class="small fw-bold"><i class="fa-solid fa-book me-
2"></i>Ваше поточне навчальне навантаження</span>
            </div>
            <div class="p-3">
                <div class="row g-3">
                    {% for sub in subjects %}
                    <div class="col-md-6">
                        <div class="p-2 border border-secondary rounded d-
flex justify-content-between align-items-center bg-dark">
                            <div>
                                <i class="fa-solid fa-folder-open text-
primary me-2"></i>
                                <strong>{{ sub.name }}</strong>
```

```
                            </div>
                            {% for g in sub.groups %}
                                <span class="badge bg-primary text-
uppercase">{{ g.name }}</span>
                            {% else %}
                                <span class="badge bg-danger text-
uppercase">Без групи</span>
                            {% endfor %}
                        </div>
                    </div>
                    {% endfor %}
                </div>
            </div>
        </div>

        <div class="card card-custom p-0 mb-5">
            <div class="card-header bg-dark p-3 d-flex justify-content-
between align-items-center border-bottom border-secondary">
                <h5 class="fw-bold mb-0 text-white"><i class="fa-solid fa-
list-check me-2"></i>Моніторинг виконання самостійної роботи</h5>
                <div class="d-flex gap-2 no-print">
                    <button onclick="exportReport()" class="btn btn-sm btn-
success fw-bold"><i class="fa-solid fa-file-excel me-1"></i> Excel
(.CSV)</button>
                    <button onclick="exportPDF()" class="btn btn-sm btn-
danger fw-bold"><i class="fa-solid fa-file-pdf me-1"></i> Звіт
(.PDF)</button>
                </div>
            </div>

            <div class="filter-panel p-3 no-print bg-dark border-bottom
border-secondary">
                <div class="row g-3">
                    <div class="col-md-3">
                        <label class="form-label small fw-bold text-
secondary">Дисципліна</label>
                        <select id="subjectFilter" onchange="filterTable()"
class="form-select form-select-sm">
                            <option value="">Всі ваші дисципліни</option>
                            {% for sub in subjects %}
                                <option value="{{ sub.name }}">{{ sub.name
}}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div class="col-md-3">
```

```
                        <label class="form-label small fw-bold text-
secondary">Академічна група</label>
                        <select id="groupFilter" onchange="filterTable()"
class="form-select form-select-sm">
                            <option value="">Всі групи</option>
                            {% for g in groups %}
                                <option value="{{ g.name }}">{{ g.name
}}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div class="col-md-3">
                        <label class="form-label small fw-bold text-
secondary">Пошук студента</label>
                        <input type="text" id="searchInp"
onkeyup="filterTable()" class="form-control form-control-sm"
placeholder="Прізвище...">
                    </div>
                    <div class="col-md-3">
                        <label class="form-label small fw-bold text-
secondary">Статус роботи</label>
                        <select id="statusFilter" onchange="filterTable()"
class="form-select form-select-sm">
                            <option value="all">Показати всіх
студентів</option>
                            <option value="active">Тільки ті, хто
працював</option>
                            <option value="passive">Боржники (0 годин
роботи)</option>
                        </select>
                    </div>
                </div>
                <div class="text-end mt-2 text-secondary small fw-bold">
                    Відображено записів: <span id="rowCount" class="text-
primary">0</span>
                </div>
            </div>

            <div class="p-0">
                <div class="d-none print-header p-4 text-center border-
bottom mb-4">
                    <h3 class="fw-bold mb-1 text-black">ВІДОМІСТЬ ОБЛІКУ
САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ</h3>
                    <p class="text-muted mb-0">Викладач: {{
current_user.full_name }} | Дата друку: <span id="pdf-date"></span></p>
                </div>
```

```
                <div class="table-custom-container m-3">
                    <div class="table-responsive">
                        <table class="table table-hover table-custom mb-0
align-middle" id="srsTable">
                            <thead>
                                <tr>
                                    <th class="ps-3 py-3" style="width:
30%;">Студент</th>
                                    <th class="text-center" style="width:
15%;">Група</th>
                                    <th class="text-center" style="width:
15%;">К-сть підходів</th>
                                    <th class="text-start" style="width:
25%;">Контрольована дисципліна</th>
                                    <th class="text-center" style="width:
15%;">Загальний час</th>
                                </tr>
                            </thead>
                            <tbody id="aggregated-rows"></tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>

        <script>
            const rawLogs = [
                {% for log in logs %}
                {
                    studentName: "{{ log.student.full_name }}",
                    group: "{{ log.student.group.name if log.student.group else
'-' }}",
                    subject: "{{ log.subject.name }}",
                    task: "{{ log.task_description | replace('"', '\\"') }}",
                    date: "{{ log.start_time.strftime('%d.%m.%Y') }}",
                    seconds: {% if log.end_time %}{{ (log.end_time -
log.start_time).total_seconds() - log.total_paused_seconds }}{% else %}0{%
endif %}
                },
                {% endfor %}
            ];

            const allStudents = [
                {% for sub in subjects %}
                    {% for group in sub.groups %}
                        {% for student in group.students %}
```

```
                    {
                        name: "{{ student.full_name }}",
                        group: "{{ group.name }}",
                        subject: "{{ sub.name }}"
                    },
                    {% endfor %}
                {% endfor %}
            {% endfor %}
        ];

        function initDashboard() {
            const tbody = document.getElementById("aggregated-rows");
            tbody.innerHTML = "";
            const targetMap = {};

            allStudents.forEach(st => {
                const key = st.name + "_" + st.subject;
                targetMap[key] = { name: st.name, group: st.group, subject:
st.subject, totalSeconds: 0, count: 0, details: [] };
            });

            rawLogs.forEach(log => {
                const key = log.studentName + "_" + log.subject;
                if (!targetMap[key]) {
                    targetMap[key] = { name: log.studentName, group:
log.group, subject: log.subject, totalSeconds: 0, count: 0, details: [] };
                }
                targetMap[key].totalSeconds += log.seconds;
                targetMap[key].count += 1;
                targetMap[key].details.push(log);
            });

            let globalIdx = 0;
            for (let k in targetMap) {
                const entry = targetMap[k];
                const hours = Math.floor(entry.totalSeconds / 3600);
                const minutes = Math.floor((entry.totalSeconds % 3600) /
60);
                const timeStr = `${hours}г ${minutes}хв`;

                const isDebtor = entry.totalSeconds === 0;
                const rowClass = isDebtor ? "debtor-row" : "table-row-data";

                const mainRow = document.createElement("tr");
                mainRow.className = `${rowClass} align-middle`;
                mainRow.setAttribute("data-student",
entry.name.toLowerCase());
```

```
                mainRow.setAttribute("data-group", entry.group);
                mainRow.setAttribute("data-subject", entry.subject);
                mainRow.setAttribute("data-status", isDebtor ? "passive" :
"active");
                mainRow.style.cursor = isDebtor ? "default" : "pointer";

                let toggleAction = isDebtor ? "" :
`onclick="toggleDetails('details_${globalIdx}', this)"`;

                mainRow.innerHTML = `
                    <td class="ps-3 fw-bold" ${toggleAction}>
                        ${!isDebtor ? '<i class="fa-solid fa-chevron-right
me-2 text-primary small-icon"></i>' : '<i class="fa-solid fa-user-xmark me-2
text-danger"></i>'}
                        ${entry.name}
                    </td>
                    <td class="text-center" ${toggleAction}><span
class="badge bg-secondary text-uppercase">${entry.group}</span></td>
                    <td class="text-center"
${toggleAction}>${entry.count}</td>
                    <td class="text-start text-info"
${toggleAction}>${entry.subject}</td>
                    <td class="text-center fw-bold"
${toggleAction}>${timeStr}</td>
                `;
                tbody.appendChild(mainRow);

                if (!isDebtor) {
                    const detailRow = document.createElement("tr");
                    detailRow.id = `details_${globalIdx}`;
                    detailRow.className = "nested-log";

                    let innerTable = `
                        <td colspan="5" class="p-3 bg-dark">
                            <div class="card card-custom p-3 m-2">
                                <h6 class="fw-bold mb-2 text-success"><i
class="fa-solid fa-info-circle me-1"></i> Хронологія сесій студента:</h6>
                                <table class="table table-sm table-bordered
text-white mb-0">
                                    <thead>
                                        <tr class="bg-secondary text-
center">
                                            <th>Дата</th><th>Опис завдання
CPC</th><th>Корисний час</th>
                                        </tr>
                                    </thead>
                                    <tbody>
```

```
                    `;
                    entry.details.forEach(d => {
                        const h = Math.floor(d.seconds / 3600);
                        const m = Math.floor((d.seconds % 3600) / 60);
                        innerTable += `
                            <tr>
                                <td class="text-center">${d.date}</td>
                                <td>${d.task}</td>
                                <td class="text-center">${h}г ${m}хв</td>
                            </tr>
                        `;
                    });
                    innerTable += `</tbody></table></div></td>`;
                    detailRow.innerHTML = innerTable;
                    tbody.appendChild(detailRow);
                }
                globalIdx++;
            }
            filterTable();
        }

        function toggleDetails(rowId, el) {
            const row = document.getElementById(rowId);
            const icon = el.querySelector(".small-icon");
            if (row.style.display === "table-row") {
                row.style.display = "none";
                if(icon) icon.className = "fa-solid fa-chevron-right me-2
text-primary small-icon";
            } else {
                row.style.display = "table-row";
                if(icon) icon.className = "fa-solid fa-chevron-down me-2
text-success small-icon";
            }
        }

        function filterTable() {
            const subVal = document.getElementById("subjectFilter").value;
            const groupVal = document.getElementById("groupFilter").value;
            const searchVal =
document.getElementById("searchInp").value.toLowerCase();
            const statusVal = document.getElementById("statusFilter").value;

            const rows = document.getElementsByClassName("table-row-data");
            const debtorRows = document.getElementsByClassName("debtor-
row");
            let visibleCount = 0;
```

```
            function processRows(rowList) {
                for (let i = 0; i < rowList.length; i++) {
                    const r = rowList[i];
                    const sName = r.getAttribute("data-student");
                    const sGroup = r.getAttribute("data-group");
                    const sSub = r.getAttribute("data-subject");
                    const sStat = r.getAttribute("data-status");

                    let match = true;
                    if (subVal && sSub !== subVal) match = false;
                    if (groupVal && sGroup !== groupVal) match = false;
                    if (searchVal && !sName.includes(searchVal)) match =
false;
                    if (statusVal !== "all" && sStat !== statusVal) match =
false;

                    if (match) {
                        r.style.display = "table-row";
                        visibleCount++;
                    } else {
                        r.style.display = "none";
                        const detailId = r.nextElementSibling?.id;
                        if (detailId && detailId.startsWith("details_")) {
                            document.getElementById(detailId).style.display
= "none";
                            const icon = r.querySelector(".small-icon");
                            if(icon) icon.className = "fa-solid fa-chevron-
right me-2 text-primary small-icon";
                        }
                    }
                }
            }
            processRows(rows);
            processRows(debtorRows);
            document.getElementById("rowCount").textContent = visibleCount;
        }

        function exportReport() {
            let csvContent = "\uFEFFСтудент;Група;Дисципліна;Загальний
час\n";
            const rows = document.querySelectorAll(".table-row-data,
.debtor-row");
            rows.forEach(r => {
                if (r.style.display !== "none") {
                    const name = r.cells[0].textContent.replace(/[\n\r]/g,
'').replace(/\s+/g, ' ').trim();
                    const group = r.cells[1].textContent.trim();
```

```
                    const sub = r.cells[3].textContent.trim();
                    const time = r.cells[4].textContent.trim();
                    csvContent += `${name};${group};${sub};${time}\n`;
                }
            });
            let blob = new Blob([csvContent], { type: 'text/csv;charset=utf-
8;' });
            let link = document.createElement("a");
            link.href = URL.createObjectURL(blob);
            link.setAttribute("download", `СРС_Відомість.csv`);
            link.click();
        }

        function exportPDF() {
            document.getElementById('pdf-date').textContent = new
Date().toLocaleDateString('uk-UA');
            window.print();
        }

        window.onload = initDashboard;
    </script>
</body>
</html>
```

Вміст файлу login.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Вхід до системи</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>
    <div class="container py-5">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card card-custom shadow mt-5">
                    <div class="card-body p-4">
                        <h3 class="text-center mb-4 fw-bold text-
primary">Авторизація</h3>
```

```
                        {% with messages =
get_flashed_messages(with_categories=true) %}
                            {% if messages %}
                                {% for cat, msg in messages %}
                                    <div class="alert alert-{{cat}} small
py-2">{{msg}}</div>
                                {% endfor %}
                            {% endif %}
                        {% endwith %}
                        <form method="POST">
                            <div class="mb-3">
                                <label class="form-label small fw-
bold">Email</label>
                                <input type="email" name="email"
class="form-control" placeholder="admin@gmail.com" required>
                            </div>
                            <div class="mb-4">
                                <label class="form-label small fw-
bold">Пароль</label>
                                <input type="password" name="password"
class="form-control" placeholder="••••••••" required>
                            </div>
                            <button type="submit" class="btn btn-primary w-
100 fw-bold mb-3">Увійти</button>
                            <div class="text-center small">
                                <a href="/register" class="text-info text-
decoration-none">Немає акаунту? Зареєструватися</a>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Вміст файлу register.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Реєстрація у системі</title>
```

```
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>
    <div class="container py-5">
        <div class="row justify-content-center">
            <div class="col-md-5">
                <div class="card card-custom shadow">
                    <div class="card-body p-4">
                        <h3 class="text-center mb-4 fw-bold text-
success">Створення акаунту</h3>
                        {% with messages =
get_flashed_messages(with_categories=true) %}
                            {% if messages %}
                                {% for cat, msg in messages %}
                                    <div class="alert alert-{{cat}} small
py-2">{{msg}}</div>
                                {% endfor %}
                            {% endif %}
                        {% endwith %}
                        <form method="POST">
                            <div class="mb-3">
                                <label class="form-label small fw-bold">ПІБ
(Повне ім'я)</label>
                                <input type="text" name="full_name"
class="form-control" placeholder="Напр., Панева Н. С." required>
                            </div>
                            <div class="mb-3">
                                <label class="form-label small fw-
bold">Email</label>
                                <input type="email" name="email"
class="form-control" placeholder="yourname@gmail.com" required>
                            </div>
                            <div class="mb-3">
                                <label class="form-label small fw-
bold">Пароль</label>
                                <input type="password" name="password"
class="form-control" placeholder="••••••••" required>
                            </div>
                            <div class="mb-3">
                                <label class="form-label small fw-bold">Хто
ви?</label>
                                <select name="role" id="roleSelect"
class="form-select" onchange="toggleGroupField()" required>
```

```
                                    <option value="student">Студент
(Здобувач освіти)</option>
                                    <option value="teacher">Викладач /
Керівник</option>
                                </select>
                            </div>
                            <div class="mb-4" id="groupField">
                                <label class="form-label small fw-
bold">Академічна група</label>
                                <select name="group_id" class="form-select">
                                    {% for g in groups %}<option
value="{{g.id}}">{{g.name}}</option>{% endfor %}
                                </select>
                            </div>
                            <button type="submit" class="btn btn-success w-
100 fw-bold mb-3">Зареєструватися</button>
                            <div class="text-center small">
                                <a href="/login" class="text-info text-
decoration-none">Вже є акаунт? Увійти</a>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script>
        function toggleGroupField() {
            var role = document.getElementById('roleSelect').value;
            document.getElementById('groupField').style.display = (role ===
'student') ? 'block' : 'none';
        }
        toggleGroupField();
    </script>
</body>
</html>
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи:
«Розроблення інформаційної системи обліку робочого часу здобувачів освіти під час самостійної роботи»

Обсяг пояснювальної записки 76 аркушів.

6 таблиці;

9 рисунки;

1 додаток.

Дата завершення роботи: 15 *червня 2026 р*.

Підпис студента-дипломника _____ Ніка ПАНЕВА.