МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,
освітньо-професійної програми «Інформаційні технології проектування»
на тему: Веборієнтована інформаційна система підтримки діяльності онлайн-школи іноземних мов STAYinTouch

Здобувачки групи ІТ.мз-32с Сєрової Надії Геннадіївни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Надія СЄРОВА
(підпис)

Керівник доцент, к.т.н., доцент Анна НЕНЯ _____
(підпис)

Суми – 2025

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

Світлана ВАЩЕНКО

«____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Сєрової Надії Геннадіївни

1 Тема кваліфікаційної роботи Веборієнтована інформаційна система підтримки діяльності онлайн-школи іноземних мов STAYinTouch
затверджена наказом по університету від «15» жовтня 2024 р. №1053-VI

2 Термін здачі студентом кваліфікаційної роботи « ___» _____ 2024 р.

3 Вхідні дані до кваліфікаційної роботи технічне завдання

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування вебдодатку, розробка та тестування вебдодатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) Актуальність розробки, постановка задачі, аналіз додатків-аналогів, порівняльна таблиця, функціональні вимоги, структурно-функціональне моделювання, реалізація та тестування вебдодатку

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.
Керівник _____
(підпис)
Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/пп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	19.10.2024	
2	Оформлення технічного завдання	19.10.2024	
3	Огляд останніх досліджень та аналогів	19.10.2024	
4	Вибір засобів реалізації	19.10.2024	
5	Моделювання вебдодатку	10.11.2024	
6	Проектування вебдодатку	10.11.2024	
7	Структурно-функціональний аналіз	18.12.2024	
8	Розробка вебдодатку	18.12.2024	
9	Тестування вебдодатку	15.01.2025	
10	Оформлення звіту	15.01.2025	

Магістрант _____ Надія СЄРОВА

Керівник роботи _____ к.т.н., доц. Анна НЕНЯ

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Веборієнтована інформаційна система підтримки діяльності онлайн-школи іноземних мов STAYinTouch».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи – 80 сторінок, у тому числі 71 сторінка основного тексту, 3 сторінки списку використаних джерел, 7 сторінок додатків.

Актуальність роботи полягає у зростанні попиту на інноваційні інструменти для ефективного управління та організації діяльності онлайн-освітніх закладів. Сучасні виклики дистанційного навчання вимагають впровадження масштабованих і функціональних систем для підтримки операцій, комунікації та управління.

Мета: створення веборієнтованої інформаційної системи для автоматизації рутинних завдань і підвищення ефективності діяльності онлайн-школи.

Методи дослідження: системний аналіз, моделювання баз даних, веброзробка та UX/UI-дизайн. У проєкті використано сучасні технології: HTML, CSS, JavaScript, PHP, MySQL.

Основні результати: розроблено інформаційну систему, яка інтегрує інструменти для управління даними студентів, координації викладачів, відстеження платежів і автоматизації сповіщень. Ключові особливості – зручні панелі для користувачів різних ролей, інтеграція з платіжними системами та синхронізація даних між базами.

Система сприяє зниженню навантаження на адміністрацію, підвищенню ефективності роботи викладачів і задоволеності користувачів. Запроваджено механізми безпеки для захисту персональних даних.

Перспективи розвитку: розширення функціональності системи для багатомовних середовищ, інтеграція з платформами дистанційного навчання, додавання модулів персоналізації та аналітики.

Ключові слова: веборієнтована інформаційна система, онлайн-школа, управління базами даних, освітні технології, дистанційне навчання, автоматизація.

ЗМІСТ

ВСТУП

З розвитком технологій та зростанням попиту на дистанційну освіту, онлайн-школи іноземних мов стають все більш популярними. Система STAYinTouch, що представляє собою веборієнтовану інформаційну систему підтримки діяльності онлайн-школи іноземних мов, надасть можливості для управління даними про студентів, викладачів, курси та фінансові операції. Першочерговим етапом у розробці системи було проведення аналізу стану дослідження в області дистанційної освіти та інформаційних систем для підтримки навчальних процесів. Мета розробки системи полягає у створенні ефективної платформи для організації навчального процесу, підвищення взаємодії між учасниками та автоматизації багатьох рутинних завдань [7,12].

Основна функція STAYinTouch − забезпечення інтегрованого середовища для управління діяльністю онлайн-школи, яка поєднує в собі управління академічними, адміністративними та фінансовими аспектами. Платформа дозволяє студентам легко реєструватися на курси, відстежувати свій навчальний прогрес і мати доступ до матеріалів, а викладачам − керувати розкладом занять, відстежувати успішність студентів і оптимізувати робочі процеси [13].

У процесі розробки було поставлено такі задачі:

- Аналіз існуючих рішень у сфері онлайн-освіти для визначення оптимального підходу до реалізації платформи.
- Формулювання основних функцій системи на основі результатів аналізу.
- Розробка архітектури платформи з урахуванням етапів навчального процесу.
- Проєктування та реалізація бази даних для управління інформацією про студентів, викладачів, курси та фінансові операції.

На основі аналізу було проведено моделювання системи STAYinTouch, яке включає в себе розробку структурно-функціональної моделі. Основні компоненти

системи, такі як управління студентами, викладачами, курсами та фінансами, були визначені, і їх взаємозв’язки проілюстровані за допомогою діаграм варіантів використання.

Система STAYinTouch включає функціонал для обліку платежів та автоматичного нагадування про них, що спрощує процес управління фінансовими операціями. Крім того, платформа підтримує інтеграцію з базою даних, що дозволяє адміністраторам отримувати доступ до актуальної інформації, забезпечуючи прозорість і зручність управління інформацією про всіх учасників навчального процесу.

Таким чином, STAYinTouch сприяє підвищенню ефективності роботи онлайн-школи та створює сприятливі умови для розвитку освіти в дистанційному форматі, підтримуючи інноваційний підхід до навчання й управління в сучасному цифровому середовищі[14].

Перелік основних завдань, необхідних для реалізації проєкту, включає:

- Проведення аналізу існуючих систем для управління діяльністю онлайн-шкіл.
- Розробку структурно-функціональної моделі системи STAYinTouch.
- Опис основних компонентів системи, таких як управління студентами, викладачами, курсами та фінансами.
- Створення діаграм варіантів використання для ілюстрації взаємодії між компонентами.
- Опис механізмів автоматизації процесів, зокрема обліку фінансових операцій і нагадувань про платежі.
- Розробку пропозицій щодо інтеграції бази даних для забезпечення актуальності та зручності роботи з інформацією.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Дистанційна освіта має свої особливості та вимоги до інформаційних систем, які використовуються для підтримки навчального процесу. Дистанційна освіта має свої особливості та вимоги до інформаційних систем, які використовуються для підтримки навчального процесу. Для забезпечення ефективної роботи онлайн-школи іноземних мов необхідно враховувати наступні аспекти [3,4,5]:

- Облік студентів та викладачів: система повинна мати можливість зберігати інформацію про студентів, викладачів, їхній розклад, прогрес у навчанні та успішність.
- Управління курсами: зберігання детальної інформації про курси, такі як тривалість, вартість, навчальні матеріали та розклад занять.
- Фінансова підтримка: врахування платежів, нарахування стипендій або знижок, відстеження фінансових операцій.
- Автоматизація комунікацій: інтеграція функцій відправки електронних листів для нагадувань про платежі, нові курси або зміни в розкладі.

Використання онлайн-платформ відкриває широкі можливості для навчання, але водночас ставить перед освітніми закладами нові виклики щодо організації процесу. Для забезпечення ефективності дистанційного навчання інформаційні системи повинні бути добре спроєктовані, зручні у використанні та відповідати високим вимогам.

Однією з ключових особливостей дистанційного навчання є гнучкість, що дозволяє студентам обирати власний темп навчання. Це створює необхідність у системах, які забезпечують управління прогресом кожного студента. Іншою важливою рисою є високий рівень персоналізації, адже кожен студент має свої унікальні потреби. Сучасні платформи повинні враховувати це, пропонуючи індивідуалізовані навчальні траєкторії та адаптивні інструменти. Крім того, ефективність дистанційного навчання значно підвищується завдяки автоматизації

рутинних завдань, таких як реєстрація студентів, відстеження фінансових транзакцій чи надсилання нагадувань.

Інформаційні системи для підтримки дистанційної освіти повинні мати надійний функціонал для обробки даних про студентів і викладачів, управління навчальними курсами, інтеграцію фінансових операцій і забезпечення безпечної взаємодії між усіма учасниками процесу. Вони мають бути надійними, гнучкими та зручними, щоб відповідати вимогам сучасного освітнього середовища.

Попри численні переваги, дистанційна освіта стикається з певними викликами. Перш за все, ефективність такого навчання значною мірою залежить від якості технологічної інфраструктури. Інформаційні системи повинні забезпечувати не лише стабільну роботу, але й підтримку основних функцій, таких як:

- Управління навчальними даними: облік інформації про студентів, викладачів, результати тестувань, прогрес у навчанні тощо.
- Організація навчальних курсів: створення розкладу, зберігання матеріалів, управління завданнями та тестами.
- Інтеграція фінансових процесів: відстеження платежів, автоматизація фінансових операцій, розрахунок знижок для певних категорій студентів.
- Комунікація: автоматизовані нагадування про заплановані заняття, фінансові транзакції чи важливі оновлення.

Автоматизація рутинних процесів є ще однією важливою складовою, яка значно підвищує ефективність системи. Наприклад, завдяки автоматизованій реєстрації студентів або інтеграції фінансових модулів адміністративні витрати часу та ресурсів можна суттєво скоротити.

Технології дозволяють створювати високоякісні платформи для дистанційного навчання, які є безпечними, зручними та функціональними. Використання інноваційних рішень, таких як адаптивне навчання, інтерактивні середовища, аналіз даних про успішність студентів, дозволяє суттєво покращити освітній процес.

Зокрема, хмарні сервіси надають можливість доступу до матеріалів у будь-який час і з будь-якого пристрою, а мобільні додатки – забезпечують зручність навчання навіть для користувачів, які перебувають у русі. Інтеграція інструментів

штучного інтелекту, таких як персоналізовані рекомендації, допомагає студентам краще орієнтуватися в навчальних програмах і підвищує їхню залученість до процесу.

Сучасний ринок пропонує кілька популярних платформ для дистанційного навчання, кожна з яких має свої переваги та недоліки. Наприклад, Moodle є потужним і безкоштовним інструментом для управління курсами, що активно використовується у всьому світі. Вона пропонує широкий функціонал, однак її налаштування може бути складним для непідготовлених користувачів. Blackboard, яка орієнтована на корпоративний сегмент та університети, відрізняється високою масштабованістю, але її використання пов’язане з високими витратами. Edmodo пропонує зручний інтерфейс і простоту взаємодії, але має обмежений функціонал для управління складними курсами. Canvas LMS є сучасною платформою з акцентом на мобільність та зручність використання, але її можливості кастомізації також обмежені [1].

Moodle – це одна з найпоширеніших платформ з відкритим вихідним кодом, яка пропонує потужний функціонал для управління навчальними курсами. Серед її переваг – безкоштовність, велика спільнота користувачів, широкий спектр модулів і розширень, які дозволяють адаптувати платформу під потреби різних закладів. Однак, складність налаштування та потреба в технічній підтримці можуть стати перешкодою для тих, хто не має достатнього досвіду роботи з програмним забезпеченням.

Blackboard орієнтована на корпоративний сегмент і вищу освіту. Вона відзначається високою масштабованістю та стабільністю роботи навіть при значному навантаженні. Її функціонал включає інтеграцію зі сторонніми системами, розширені інструменти для аналітики та управління курсами. Основним недоліком цієї платформи є її висока вартість, яка може бути недоступною для невеликих організацій.

Edmodo є простішою у використанні платформою, орієнтованою переважно на середню освіту. Її переваги – інтуїтивний інтерфейс, легкість взаємодії між викладачами та студентами, можливість швидкої організації групових обговорень.

Проте, функціонал Edmodo обмежений, особливо якщо мова йде про управління складними навчальними програмами чи фінансовими аспектами.

Canvas LMS пропонує сучасний дизайн, мобільну доступність та зручність використання. Вона надає користувачам інструменти для організації курсів, тестування, управління прогресом студентів. Проте її можливості кастомізації обмежені, що може бути важливим недоліком для закладів з унікальними потребами.

Порівнюючи ці рішення, можна зробити висновок, що жодна з існуючих платформ не враховує повною мірою специфічні потреби мовних шкіл, які потребують акценту на автоматизацію фінансових аспектів та персоналізацію навчального процесу. Це підкреслює необхідність розробки нової платформи, орієнтованої саме на ці завдання. Результати порівняльного аналізу представлені в таблиці 1.1.

Таблиця 1.1 − Порівняльний аналіз існуючих платформ

Платформа	Функціонал	Переваги	Недоліки
Moodle	Управління курсами, тестування, відстеження прогресу.	Безкоштовна, гнучка.	Складність кастомізації, потреба в технічній підтримці.
Blackboard	Повний спектр функцій для дистанційного навчання.	Масштабованість, надійність.	Висока вартість.
Edmodo	Простота взаємодії між викладачами та учнями.	Інтуїтивний інтерфейс.	Обмежений функціонал для управління курсами.
Canvas LMS	Управління курсами, мобільна доступність.	Простота використання, сучасний дизайн.	Обмежена підтримка кастомізації.

Джерело: розроблено автором.

Для створення ефективної платформи дистанційного навчання необхідно обрати сучасні технології, які забезпечать її функціональність, безпеку та зручність у використанні. Серверна частина системи може бути реалізована за допомогою таких інструментів, як Django або Flask, які є популярними фреймворками для Python. Django відзначається своєю надійністю та широким спектром можливостей для роботи з даними, тоді як Flask підходить для невеликих проєктів завдяки своїй простоті [3, 11].

Для користувацького інтерфейсу доцільно використовувати JavaScript-фреймворки, такі як React.js або Vue.js. React.js забезпечує створення інтерактивного інтерфейсу, що відповідає потребам сучасних користувачів, тоді як

Vue.js відомий своєю простотою у використанні. Для роботи з даними оптимальним вибором є PostgreSQL, яка пропонує потужний функціонал для обробки складних структур даних [4, 6].

Інтеграція різних компонентів системи може бути забезпечена через API, а автоматизація комунікації – завдяки SMTP для надсилання електронних повідомлень. Такий підхід дозволяє створити платформу, яка буде легко масштабуватися та підтримувати додаткові функції в майбутньому.

Аналіз предметної області показав, що дистанційне навчання є надзвичайно перспективним напрямком, однак існуючі рішення не завжди відповідають специфічним потребам мовних шкіл. Вони мають обмеження в управлінні фінансовими транзакціями, автоматизації рутинних завдань і персоналізації навчального процесу. Враховуючи це, розробка нової платформи, яка поєднає всі необхідні функції в єдиній системі, є актуальним і перспективним завданням.

Обрана концепція розробки передбачає використання Django для бекенду, React.js для фронтенду та PostgreSQL для роботи з даними. Такий підхід забезпечить високу продуктивність, масштабованість та безпеку платформи, що дозволить їй ефективно підтримувати діяльність мовних шкіл і відповідати сучасним викликам у сфері дистанційного навчання.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Мета цього проєкту – створення веборієнтованої системи, яка забезпечить повний цикл управління діяльністю онлайн-школи іноземних мов з можливістю інтеграції функцій обліку, планування, контролю за успішністю та підтримки фінансових процесів.

Задачі дослідження:

- Провести аналіз існуючих технологій для вирішення поставленої цілі проєкту та обрати оптимальний підхід для створення платформи онлайн-школи.
- Враховуючи результати аналізу існуючих рішень для освітніх платформ та визначені основні функціональні вимоги, сформулювати базові функції системи.
- Розробити архітектуру системи, що враховує особливості функціонування онлайн-школи, зокрема, передбачає компоненти та модулі, які реалізують функціонал для підтримки різних етапів навчального процесу.
- Спроектувати і реалізувати модель бази даних для зберігання інформації про студентів, викладачів, курси, оплату та комунікації.

2.2 Функціональні вимоги

У процесі розробки системи для онлайн-школи важливо визначити основні функціональні та нефункціональні вимоги, які забезпечать ефективне функціонування платформи та відповідність її потребам користувачів. Функціональні вимоги описують основні можливості системи, необхідні для підтримки навчального процесу, зокрема реєстрацію та облік студентів і викладачів, управління курсами, облік фінансів та автоматичні сповіщення.

Функціональні вимоги:

- Система реєстрації та обліку студентів і викладачів. Система повинна забезпечувати можливість реєстрації користувачів з різними рівнями доступу, включаючи студентів, викладачів та адміністративний персонал. Інформація про кожного користувача має зберігатися в базі даних, включаючи основні контактні дані, академічний статус, ролі, а також права доступу. Це забезпечить структуроване та зручне управління доступом до різних функцій платформи.
- Управління курсами та розкладом. Адміністраторам та викладачам надаються інструменти для створення та редагування курсів, додавання нових занять, а також управління розкладом занять. Система має забезпечувати гнучкість для планування занять та налаштування регулярних сесій. Цей функціонал включає можливість вносити зміни в розклад та сповіщати студентів про будь-які оновлення.
- Моніторинг академічної успішності. Система повинна мати інструменти для відстеження навчального прогресу студентів. Адміністратори та викладачі повинні мати можливість переглядати академічні результати, виконані завдання та відвідуваність занять. Це дозволить відстежувати динаміку навчання кожного студента та за необхідності коригувати навчальний процес.
- Облік фінансових операцій. Важливою частиною функціоналу є зберігання та обробка інформації про оплату за курси. Система повинна автоматично генерувати нагадування про майбутні платежі, а також формувати звіти для адміністрації щодо фінансових операцій. Це сприятиме прозорості та спрощенню управління оплатою за навчання.
- Автоматичні сповіщення. Система має включати автоматичні сповіщення для інформування студентів та викладачів про важливі події, такі як початок занять, зміни у розкладі, платежі та інші заходи. Це забезпечить зручність для користувачів і своєчасне інформування про будь-які зміни чи важливі події.
- Формування звітів. Система повинна дозволяти адміністраторам генерувати звіти з інформацією про навчальну діяльність, академічні показники студентів, активність викладачів та фінансові результати. Такий функціонал

дозволить мати доступ до статистичних даних і забезпечить можливість ефективного управління процесами.

2.3 Нефункціональні вимоги

Окрім функціональних можливостей, важливо враховувати нефункціональні аспекти системи, які впливають на її зручність, надійність та ефективність. Ці вимоги забезпечують відповідність платформи сучасним стандартам якості та створюють сприятливі умови для роботи користувачів. Нефункціональні вимоги висувають умови щодо зручності, продуктивності, безпеки, масштабованості, надійності та сумісності платформи, що дозволить забезпечити зручний, надійний та безпечний досвід для всіх учасників навчального процесу [16,19].

Нижче наведено основні нефункціональні вимоги до розробки системи:

- Зручність інтерфейсу. Інтерфейс системи має бути інтуїтивно зрозумілим і зручним у використанні як для студентів, так і для викладачів та адміністраторів. Навігація по функціях повинна бути простою та логічною, що забезпечить користувачам швидкий доступ до основних можливостей без потреби в додаткових інструкціях.
- Продуктивність. Система повинна забезпечувати швидкий доступ до інформації та здатність обробляти запити користувачів навіть при значному обсязі даних. Це особливо важливо для забезпечення ефективної роботи платформи при великій кількості користувачів, що дозволить мінімізувати час завантаження сторінок та пошуку інформації.
- Безпека даних. Конфіденційні дані про студентів, викладачів і фінансові операції повинні бути захищені відповідно до сучасних стандартів кібербезпеки. Масштабованість. Система має бути спроектована таким чином, щоб можна було легко додавати нові функції та модулі, не впливаючи на роботу існуючих

компонентів. Це забезпечить гнучкість та готовність до розширення функціоналу в майбутньому.

- Надійність. Висока стабільність роботи системи є обов’язковою умовою. Платформа повинна працювати без частих збоїв, а також включати механізми резервного копіювання та відновлення даних, що дозволить швидко повернутися до роботи після можливих технічних несправностей.
- Сумісність. Система повинна підтримувати сучасні браузери та адаптувати інтерфейс для різних типів пристроїв, таких як комп'ютери, планшети та смартфони. Це дозволить користувачам мати повний доступ до функцій платформи незалежно від типу пристрою.

3. ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Структурно-функціональне моделювання

Для успішної реалізації системи підтримки діяльності онлайн-школи STAYinTouch було проведено структурно-функціональне моделювання. Це моделювання включає опис основних компонентів системи та їх взаємозв'язків. Результати моделювання демонструють, як різні модулі, такі як управління студентами, викладачами, курсами та фінансами, взаємодіють між собою для забезпечення безперебійної роботи платформи. Основою цього підходу є методи UML для діаграм варіантів використання [2], концептуальне моделювання інформаційних систем [8], а також принципи функціонального моделювання, що сприяють декомпозиції задач на нульовому рівні [6].

Нижче представлені діаграми.

На рис. 2.1 представлена діаграма структурно-функціонального проектування на контексному рівні. Автоматизація процесів онлайн-школи STAYinTouch представлена у вигляді контекстної діаграми, яка демонструє ключові елементи, потоки даних і технології, що забезпечують функціонування системи. Діаграма включає центральний блок, вхідні й вихідні потоки, використовувані технології та керуючі фактори.

Центральним елементом є блок "Підтримка бізнес-процесів онлайн-школи STAYinTouch", який відображає основні функції системи. Цей блок слугує ядром, що приймає вхідні дані, обробляє їх і формує вихідну інформацію.

Вхідні дані включають:

- Дані авторизації, що забезпечують захищений доступ до системи для користувачів та адміністраторів, та використовуються на етапах авторизації або реєстрації.
- Умови пошуку курсів, що визначають критерії, за якими користувачі можуть знайти відповідні освітні програми.

- Заявки на навчання або викладання, які надсилають студенти або викладачі для реєстрації у системі.

Обмежуючими факторами є:

- Політика конфіденційності, що регламентує використання персональних даних користувачів.
- Фінансові обмеження, які визначають ліміти на оплату курсів чи інші фінансові умови.
- Правила користування платформою, які встановлюють загальні вимоги для роботи з системою.

На виході система генерує:

- Дані про статус оплати, що містять інформацію про фінансові операції студентів.
- Збережені записи, які зберігають дані про навчальний процес, включаючи виконані завдання.
- Сформовані розклади, що автоматично генеруються для студентів і викладачів.
- Надіслані повідомлення, які включають сповіщення користувачів про важливі події чи зміни.
- Сформовані звіти перевірки, що використовуються для аналізу прогресу студентів або перевірки фінансових транзакцій.

Для реалізації системи використовується низка сучасних технологій. Зокрема:

- Flask і SQLAlchemy, які забезпечують побудову серверної частини додатку та взаємодію з базою даних.
- HTML, CSS і JavaScript, що використовуються для створення інтуїтивно зрозумілого інтерфейсу.
- Stripe і Flask-Mail, які автоматизують фінансові операції та надсилання електронних повідомлень.

Таким чином, система STAYinTouch забезпечує ефективну автоматизацію основних процесів онлайн-школи, дозволяючи інтегрувати сучасні технології для управління курсами, оплати й комунікації з користувачами.

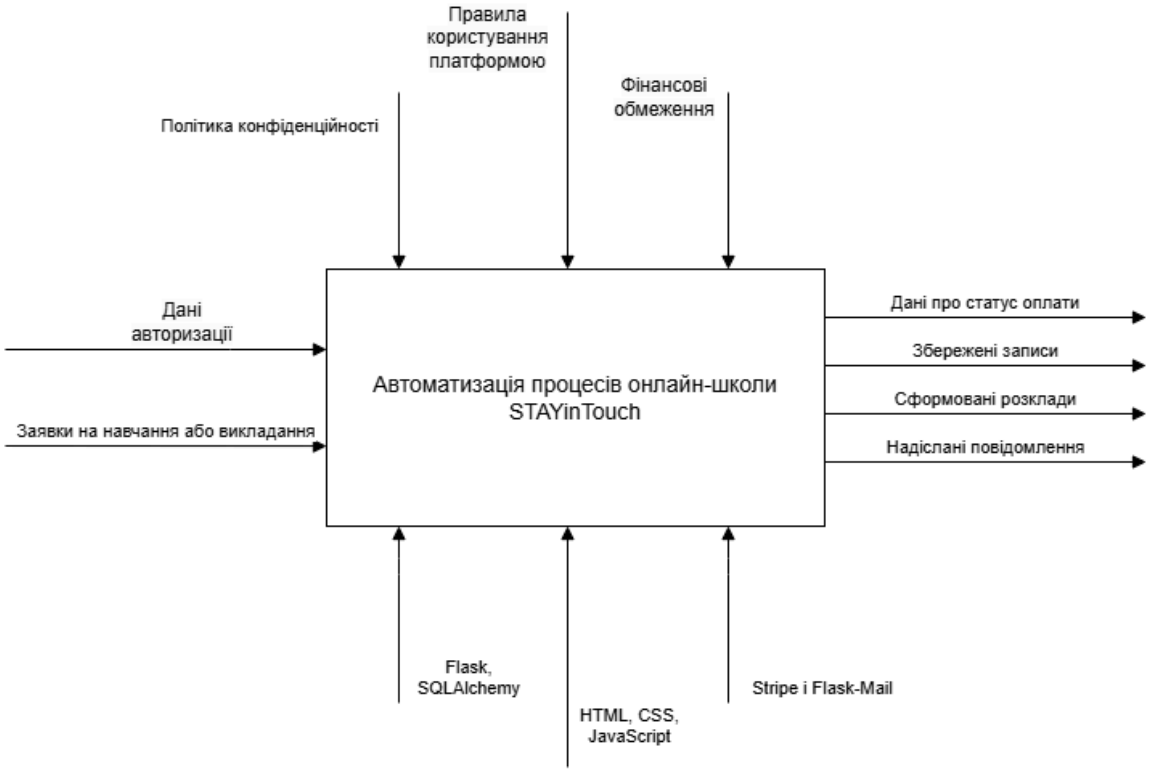

Рисунок 2.1 – Контекстна діаграма автоматизації процесів онлайн-школи STAYinTouch з точки зору адміністратора.

Джерело: розроблено автором.

Контексна діаграма демонструє ключові елементи системи автоматизації процесів онлайн-школи STAYinTouch. Діаграма відображає основні вхідні та вихідні потоки даних, а також технології, що забезпечують її функціонування.

Діаграма нульового рівня декомпозиції (рис. 2.2) відображає загальний огляд основних функціональних блоків системи STAYinTouch з точки зору адміністратора. Вона включає такі основні процеси, як реєстрація студентів, призначення навчальних планів, перевірка завдань та перевірка оплати. Кожен з цих блоків поділяється на більш дрібні функціональні елементи, що відповідають за конкретні завдання, такі як обробка заявок, генерування розкладів, обробка результатів перевірки завдань та контроль за фінансовими транзакціями.

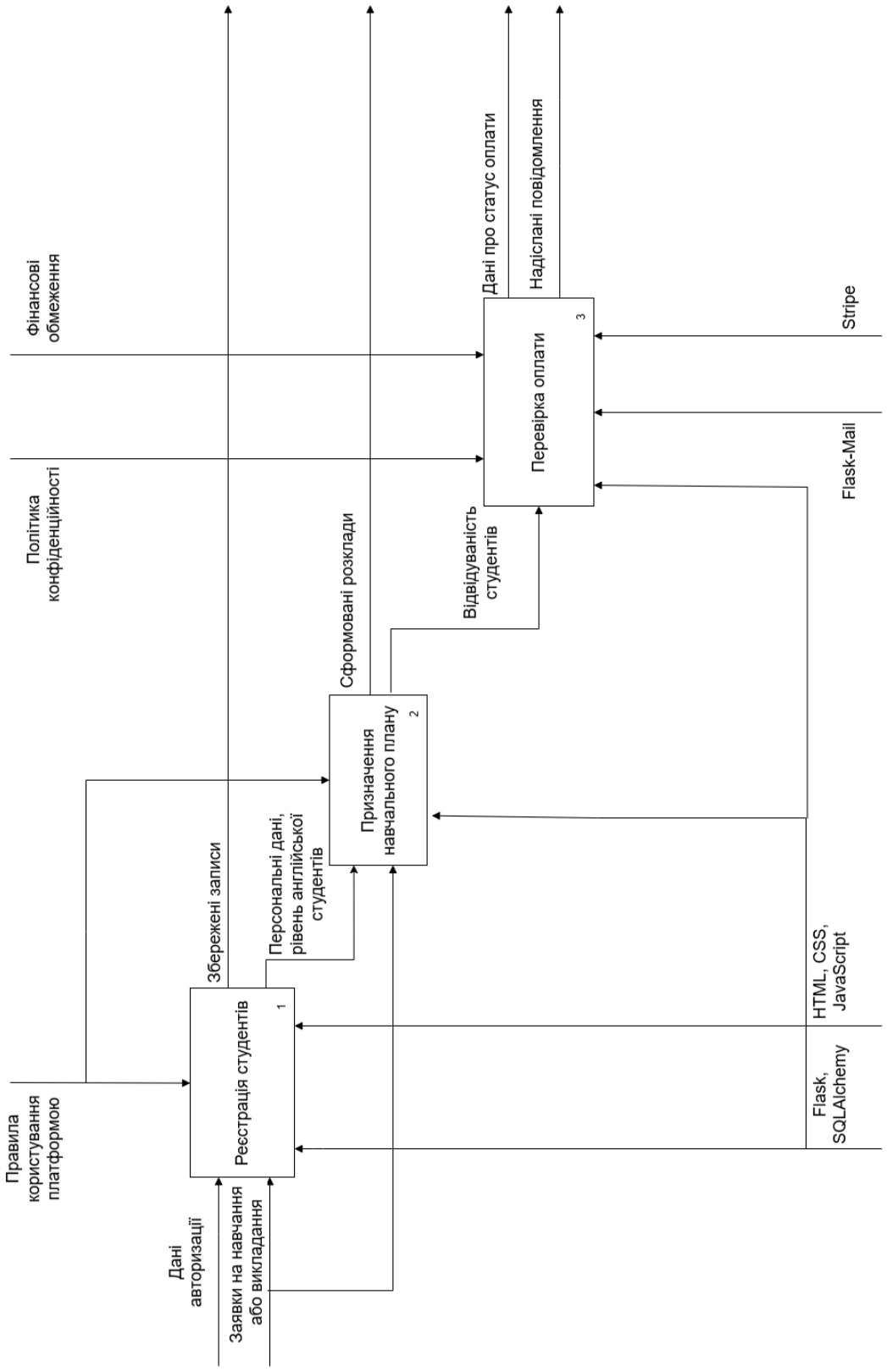

Рисунок 2.2 – Діаграма нульового рівня декомпозиції з точки зору адміністратора.

Джерело: розроблено автором.

На рис. 2.3 представлена контекстна діаграма автоматизації процесів для викладачів у системі STAYinTouch. Ця діаграма демонструє основні потоки даних, що забезпечують взаємодію викладача із системою, а також технології, які реалізують ці процеси.

Вхідні дані включають:

- Персональна інформація: включає особисті дані викладача, необхідні для створення облікового запису в системі.
- Дані авторизації: забезпечують безпечний доступ викладача до системи.
- Документи для підтвердження кваліфікації: використовуються для перевірки та підтвердження професійних навичок викладача.

Керуючі фактори:

- Політика конфіденційності: регламентує використання персональних даних викладачів та гарантує їхню безпеку.
- Правила користування платформою: визначають основні принципи та обмеження роботи викладачів у системі.

Для реалізації системи використовуються такі сучасні інструменти:

- Flask і SQLAlchemy — для обробки запитів викладачів, збереження та організації даних у базі.
- HTML, CSS, JavaScript — для забезпечення зручного та функціонального інтерфейсу користувача.
- Flask-Mail — для автоматизації надсилання повідомлень викладачам.

Вихідні дані:

- Збережені записи: відомості про виконану роботу, результати перевірок і комунікацію зі студентами.
- Сформований розклад занять: автоматично генерується для організації навчального процесу.
- Призначені групи студентів: забезпечують розподіл учнів між викладачами.

- Записані результати занять: включають звіти про успішність занять і взаємодію з учнями.
- Призначені домашні завдання: автоматично додаються до профілів учнів.
- Надіслані повідомлення: інформують учасників навчального процесу про зміни чи події.
- Оцінки та коментарі: дозволяють викладачам давати зворотний зв'язок студентам.
- Перевірені роботи студентів: результати оцінювання завдань викладачем.

Таким чином, система STAYinTouch забезпечує ефективну автоматизацію викладацької діяльності, використовуючи сучасні технології для спрощення взаємодії з учнями, організації навчального процесу та надання якісного зворотного зв'язку.

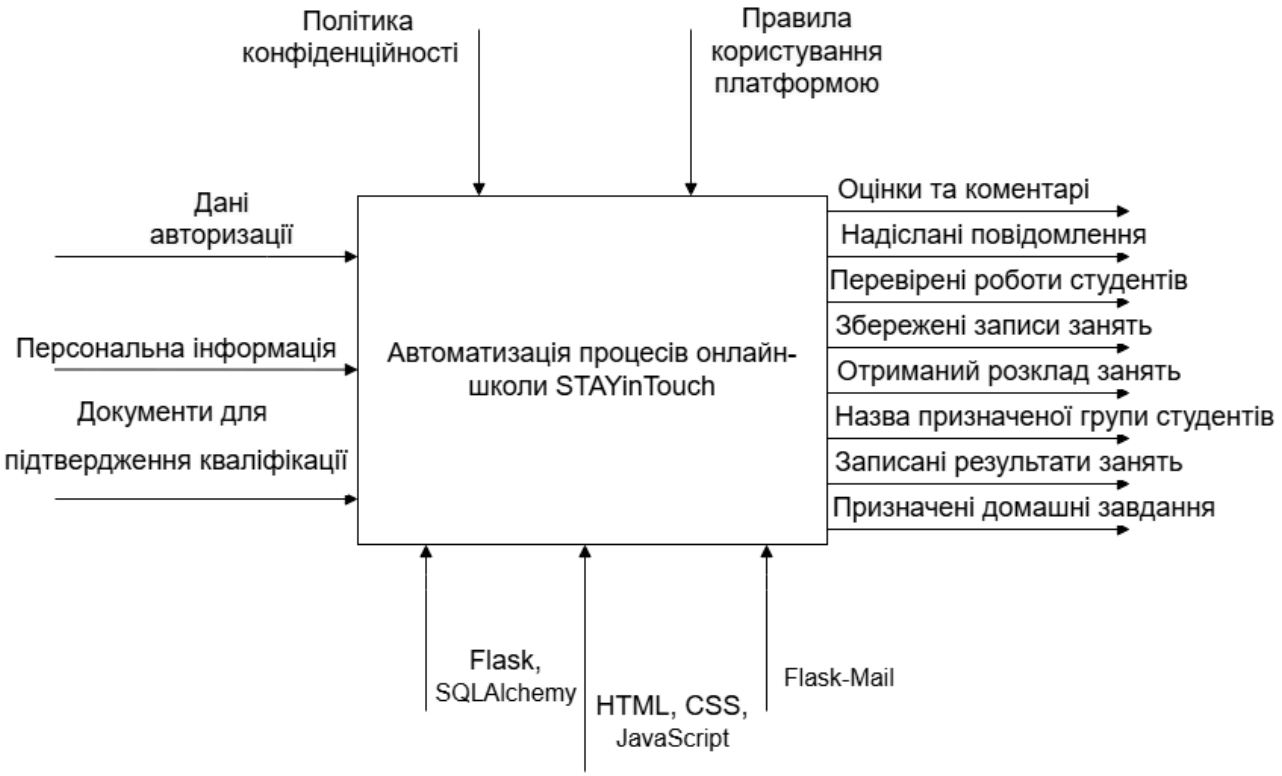

Рисунок 2.3 – Контекстна діаграма автоматизації процесів онлайн-школи STAYinTouch з точки зору викладача.

Джерело: розроблено автором.

З точки зору викладача система STAYinTouch має інші ключові процеси, що відображені на відповідній діаграмі нульового рівня декомпозиції з точки зору викладача (рис. 2.4). Основні функціональні блоки включають реєстрацію викладачів, призначення уроків, проведення занять, перевірку завдань. Реєстрація викладачів охоплює подання заявки та перевірку кваліфікації. Призначення уроків включає розподіл викладачів по групах та формування розкладів. Проведення занять передбачає інтерактивну взаємодію з учнями, надання навчальних матеріалів та перевірку відвідуваності. Перевірка завдань охоплює оцінювання робіт, надання зворотного зв’язку та спілкування зі студентами.

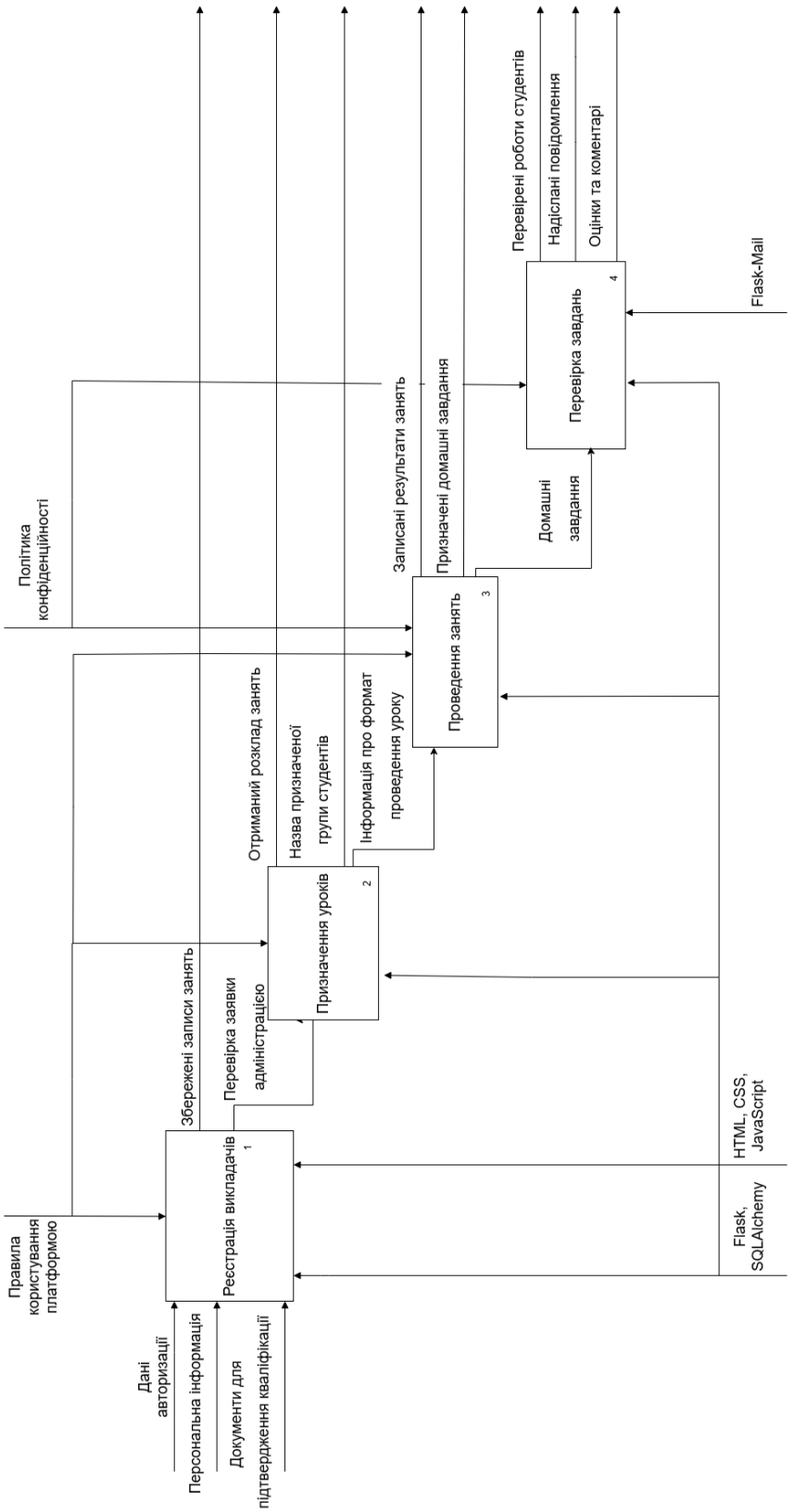

Рисунок 2.4 – Діаграма нульового рівня декомпозиції з точки зору викладача.

Джерело: розроблено автором.

На рис. 2.5 представлена контекстна діаграма автоматизації процесів для студентів у системі STAYinTouch. Ця діаграма відображає основні потоки даних, що забезпечують взаємодію студентів із системою, а також технології, які використовуються для реалізації цих процесів.

Вхідні дані включають:

- Параметри пошуку курсів: студенти вводять критерії пошуку, такі як предмети, викладачі або теми.
- Фільтри (рівень, формат занять): студенти можуть застосовувати додаткові фільтри для відбору курсів за рівнем складності чи форматом (онлайн/офлайн).
- Дані авторизації: забезпечують доступ студентів до їхніх персоналізованих профілів у системі.

Керуючі фактори:

- Політика конфіденційності: регламентує обробку персональних даних студентів і гарантує їхню безпеку.
- Правила користування платформою: визначають основні умови та обмеження для використання системи.

Технології:

- Flask і SQLAlchemy — для обробки запитів студентів і взаємодії з базою даних.
- HTML, CSS, JavaScript — для створення зручного інтерфейсу користувача.
- Flask-Mail — для автоматизації надсилання повідомлень студентам.
- Stripe — для обробки фінансових транзакцій, пов’язаних із оплатою курсів.

Вихідні дані:

- Перелік доступних курсів: відображає курси, які відповідають заданим параметрам пошуку та фільтрам.
- Статус реєстрації: інформація про те, чи зареєстрований студент на обраний курс.

- Виконані завдання: відображаються у профілі студента після їх завершення та подання на перевірку.
- Статус оплати: включає інформацію про успішність фінансових транзакцій.
- Квитанція про оплату: автоматично генерується після завершення оплати курсу.

Таким чином, система STAYinTouch забезпечує студентам зручний доступ до курсів, організовує їхню реєстрацію, виконання завдань і оплати, використовуючи сучасні технології для автоматизації та покращення навчального процесу.

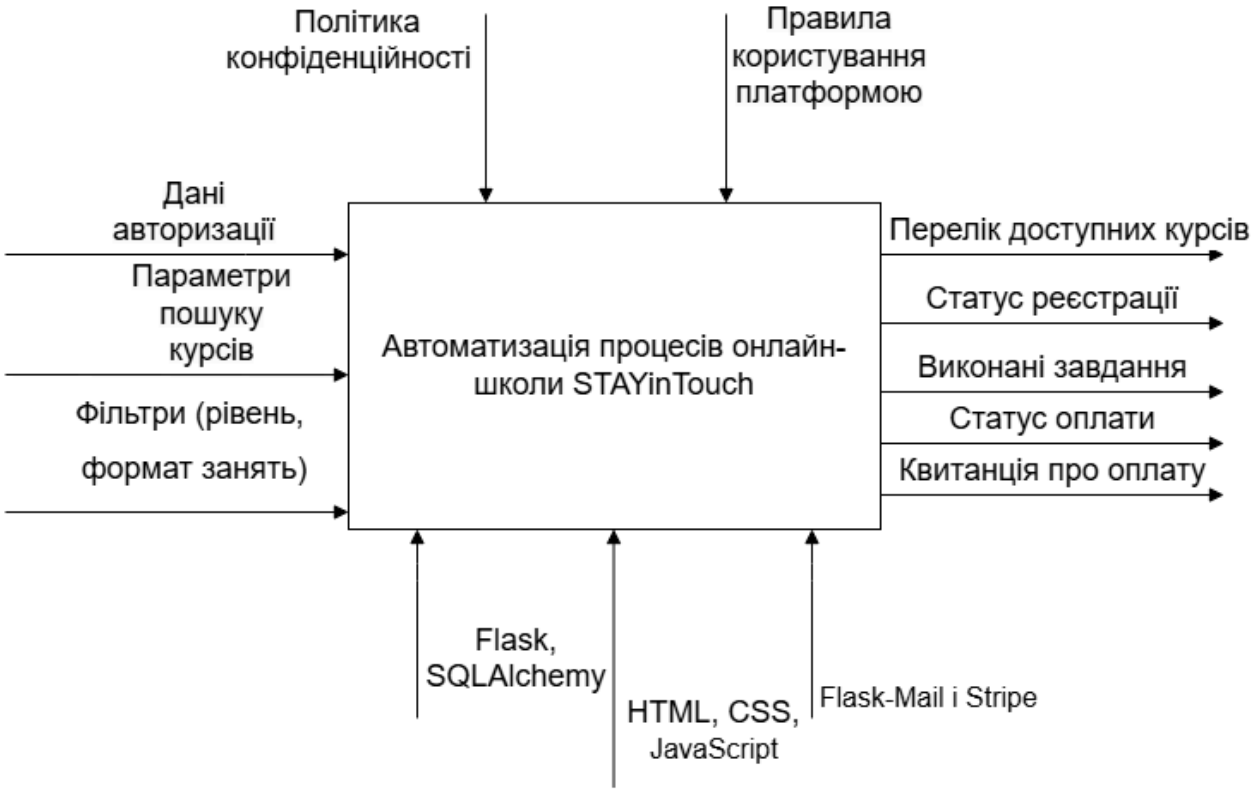

Рисунок 2.5 – Контекстна діаграма автоматизації процесів онлайн-школи STAYinTouch з точки зору студента.

Джерело: розроблено автором.

Діаграма нульового рівня декомпозиції (рис. 2.6) відображає загальний огляд основних функціональних блоків системи STAYinTouch з точки зору студента. Вона

включає такі основні процеси, як пошук та вибір курсу, реєстрація на курс, навчальний процес та оплата курсу.

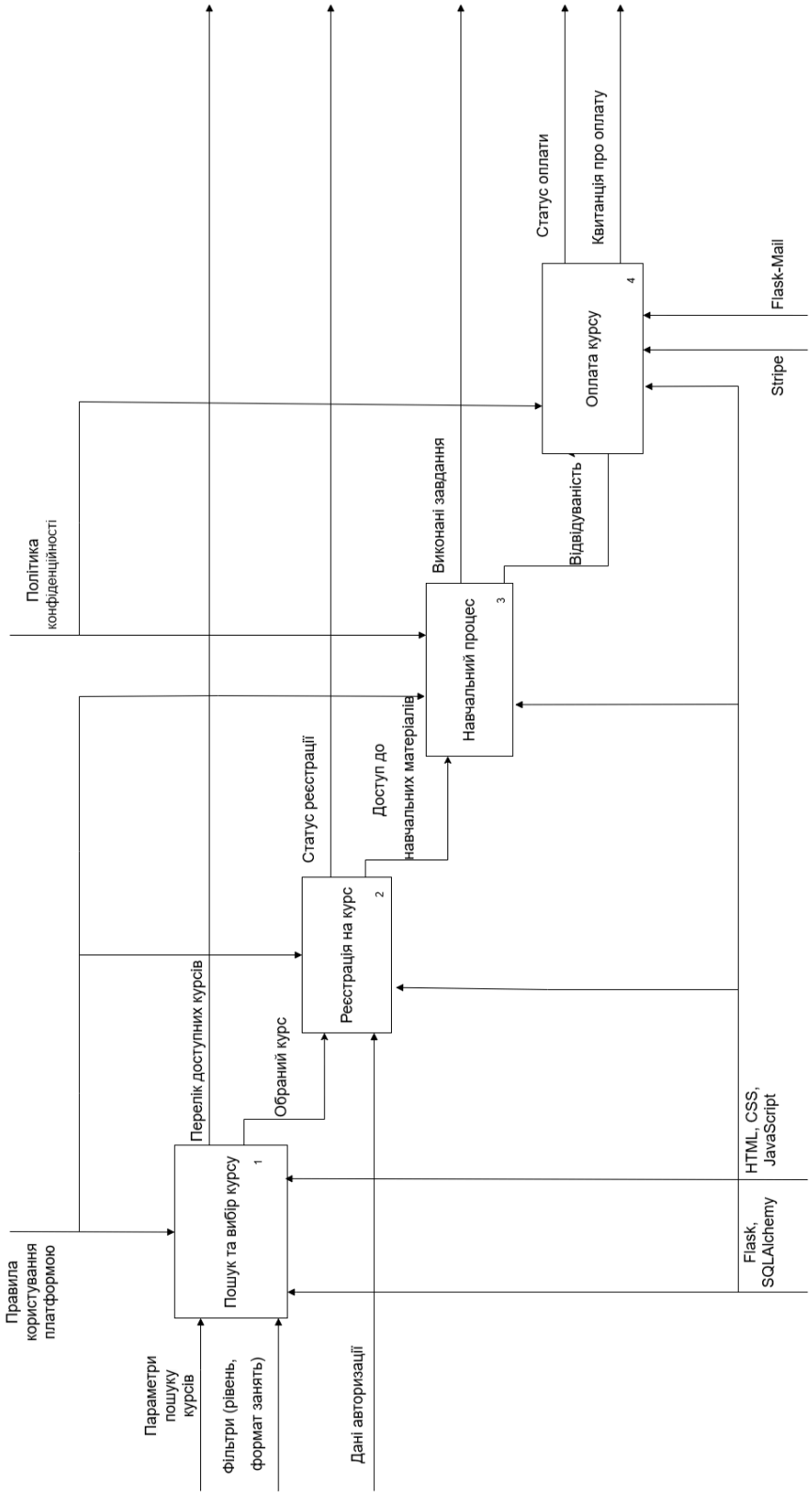

Рисунок 2.6 – Діаграма нульового рівня декомпозиції з точки зору студента.

Джерело: розроблено автором.

3.2 Діаграми варіантів використання

Діаграми варіантів використання (Use Case Diagrams) ілюструють основні сценарії взаємодії користувачів з системою STAYinTouch. Ці діаграми демонструють, як різні типи користувачів – адміністратори, слухачі та викладачі – взаємодіють з основними функціями платформи, такими як реєстрація, оплата, управління розкладом та комунікації.

Діаграма варіантів використання для адміністратора представлена на рисунку 2.7. Адміністратор є основним користувачем системи, який має доступ до ключових функцій платформи. Основні сценарії його взаємодії з системою включають:

- Реєстрація студентів − адміністратор додає нових студентів у систему.
- Управління викладачами – адміністратор керує викладачами, включаючи їх додавання, редагування та видалення.
- Формування розкладу – створення та редагування розкладу занять для викладачів і студентів.
- Призначення навчального плану – адміністратор закріплює студентів за відповідним навчальним планом.
- Перевірка оплати – перевірка статусу оплат, формування звітів про фінансові операції.

Ця діаграма відображає всі основні функції, доступні адміністратору, що забезпечують ефективне управління навчальним процесом, координацію роботи викладачів і студентів, а також фінансовий контроль у межах онлайн-школи.

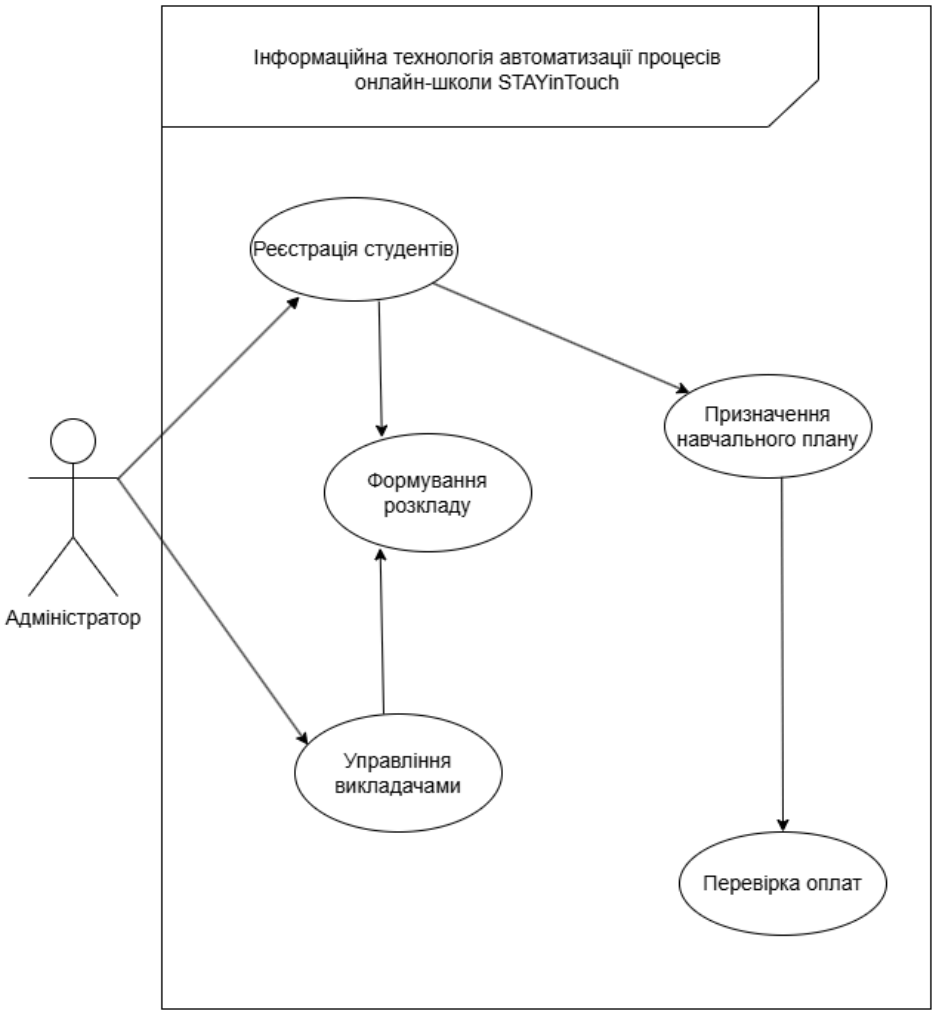

Рисунок 2.7 – Діаграма варіантів використання для адміністратора.

Джерело: розроблено автором.

Діаграма варіантів використання для слухача (рис. 2.8) ілюструє основні сценарії взаємодії студентів із системою STAYinTouch. Слухачі мають доступ до функцій, що забезпечують зручність навчання та організацію процесу проходження курсів. Основні сценарії взаємодії слухача з системою включають:

- Авторизація – вхід до системи для отримання доступу до особистого кабінету та навчальних матеріалів.
- Пошук та вибір курсу – перегляд доступних курсів та вибір відповідного навчального напряму.

- Реєстрація на курси – слухач обирає та реєструється на доступні курси.
- Перегляд розкладу занять – перегляд запланованих занять та оновлень у розкладі.
- Виконання завдань – доступ до навчальних матеріалів, виконання завдань та взаємодія з викладачем у межах курсу.
- Оплата курсу – перевірка статусу оплати та здійснення платежів за курси.

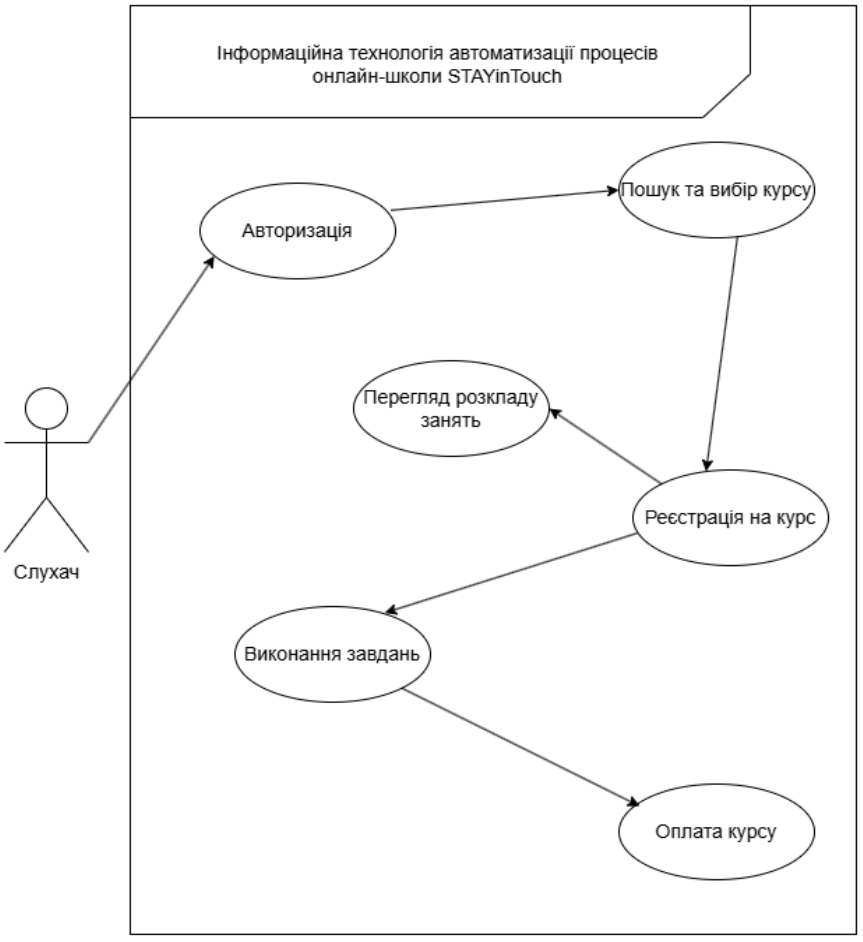

Рисунок 2.8 – Діаграма варіантів використання для слухача.

Джерело: розроблено автором.

Ця діаграма відображає всі основні функції, доступні слухачеві, що забезпечують повний цикл навчання – від вибору курсу до його проходження та оплати.

Діаграма варіантів використання для викладача (рис. 2.9) ілюструє основні сценарії взаємодії викладачів із системою STAYinTouch. Викладачі мають доступ до функцій, що дозволяють їм ефективно організовувати навчальний процес та взаємодіяти зі студентами. Основні сценарії взаємодії викладача з системою включають:

- Реєстрація викладачів – додавання нових викладачів до системи.
- Призначення уроків – отримання та управління розкладом занять.
- Проведення занять – взаємодія зі студентами під час уроків.
- Управління навчальними матеріалами – завантаження, редагування та видалення навчальних ресурсів.
- Перегляд розкладу – доступ до розкладу занять та оновлення інформації.
- Комунікація зі студентами – обмін повідомленнями, відповіді на запити студентів.
- Перевірка завдань – перевірка виконаних студентами завдань та надання зворотного зв’язку.

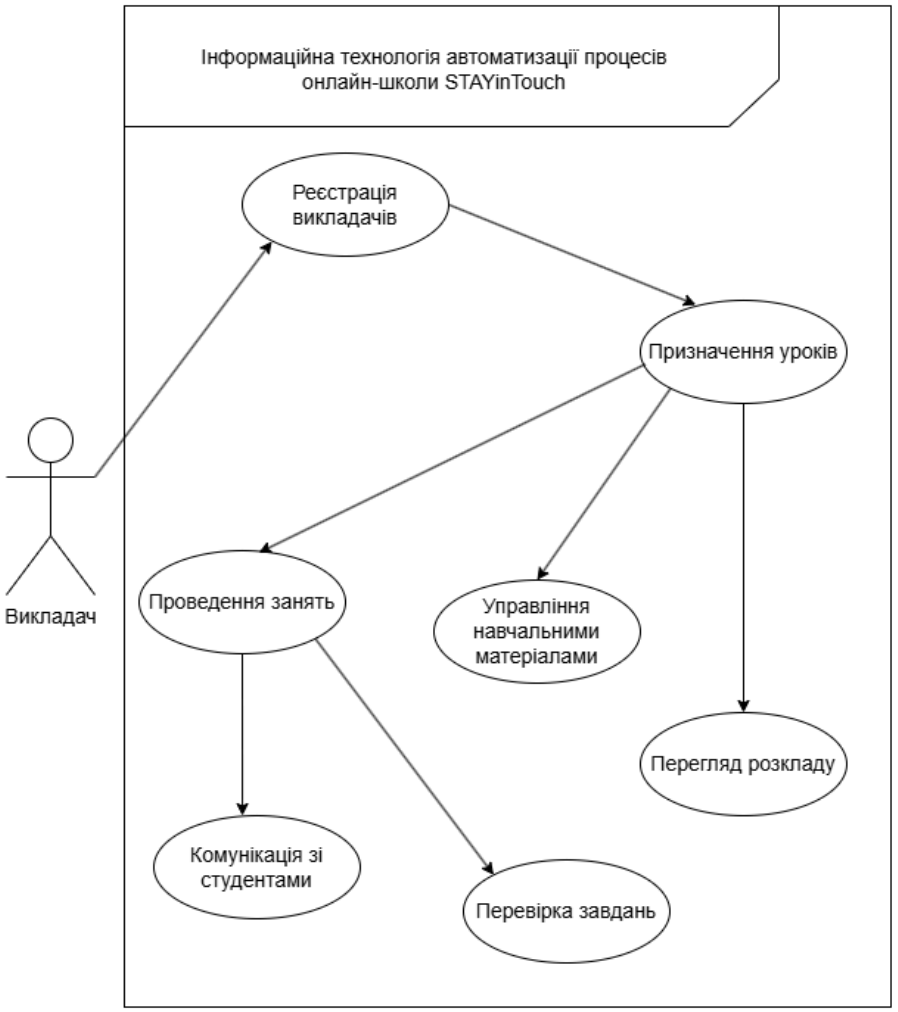

Рисунок 2.9 – Діаграма варіантів використання для викладача.

Джерело: розроблено автором.

Ця діаграма демонструє основні функції, доступні викладачеві, що забезпечують організацію навчального процесу, управління навчальними матеріалами та ефективну взаємодію зі студентами.

3.3 Архітектура системи

Архітектура інформаційної системи STAYinTouch базується на веб-технологіях і передбачає розподілену клієнт-серверну структуру. Вона складається з клієнтської частини, серверної частини та підсистеми зберігання даних. Основні компоненти системи та їх взаємодія наведені на рисунку 2.10.

Клієнтська частина реалізована у вигляді веб-додатка на основі React, що відповідає за відображення інтерфейсу користувача та взаємодію з сервером. Для анімаційного оформлення використовується бібліотека Framer Motion, що забезпечує плавність та інтерактивність інтерфейсу. Запити на сервер здійснюються за допомогою HTTP-запитів до Flask API, що забезпечує обробку користувацьких дій та передачу необхідних даних між клієнтом і сервером.

Серверна частина побудована на основі Flask, що виконує функції бізнес-логіки, обробки запитів від клієнта та взаємодії з базами даних. Flask API забезпечує обробку запитів і передачу відповідей, необхідних для коректного функціонування клієнтського додатка. Сервер також містить механізми аутентифікації користувачів, що реалізовані за допомогою Flask-Login. Це дозволяє забезпечити захищений доступ до особистих кабінетів користувачів, курсів і платіжних операцій. Крім того, серверна частина використовує Flask-Mail для надсилання повідомлень електронною поштою, а також SQLAlchemy та Alembic для управління базами даних і міграційної підтримки.

Система використовує дві окремі бази даних:

- PostgreSQL (Admin DB) – основна база даних адміністративної частини, яка зберігає інформацію про адміністраторів, викладачів, платежі та курси.
- SQLite (User DB) – окрема база даних для користувачів, яка зберігає реєстраційні дані студентів, історію уроків та інші користувацькі дані.

Для забезпечення надійності даних передбачена система резервного копіювання баз даних, що гарантує відновлення у разі втрати інформації або технічних збоїв.

Клієнтська частина взаємодіє з сервером через Flask API, передаючи та отримуючи дані, необхідні для відображення інтерфейсу та виконання основних функцій системи. Серверна частина обробляє ці запити, звертаючись до баз даних та інших модулів, а також виконує функції аутентифікації та управління даними.

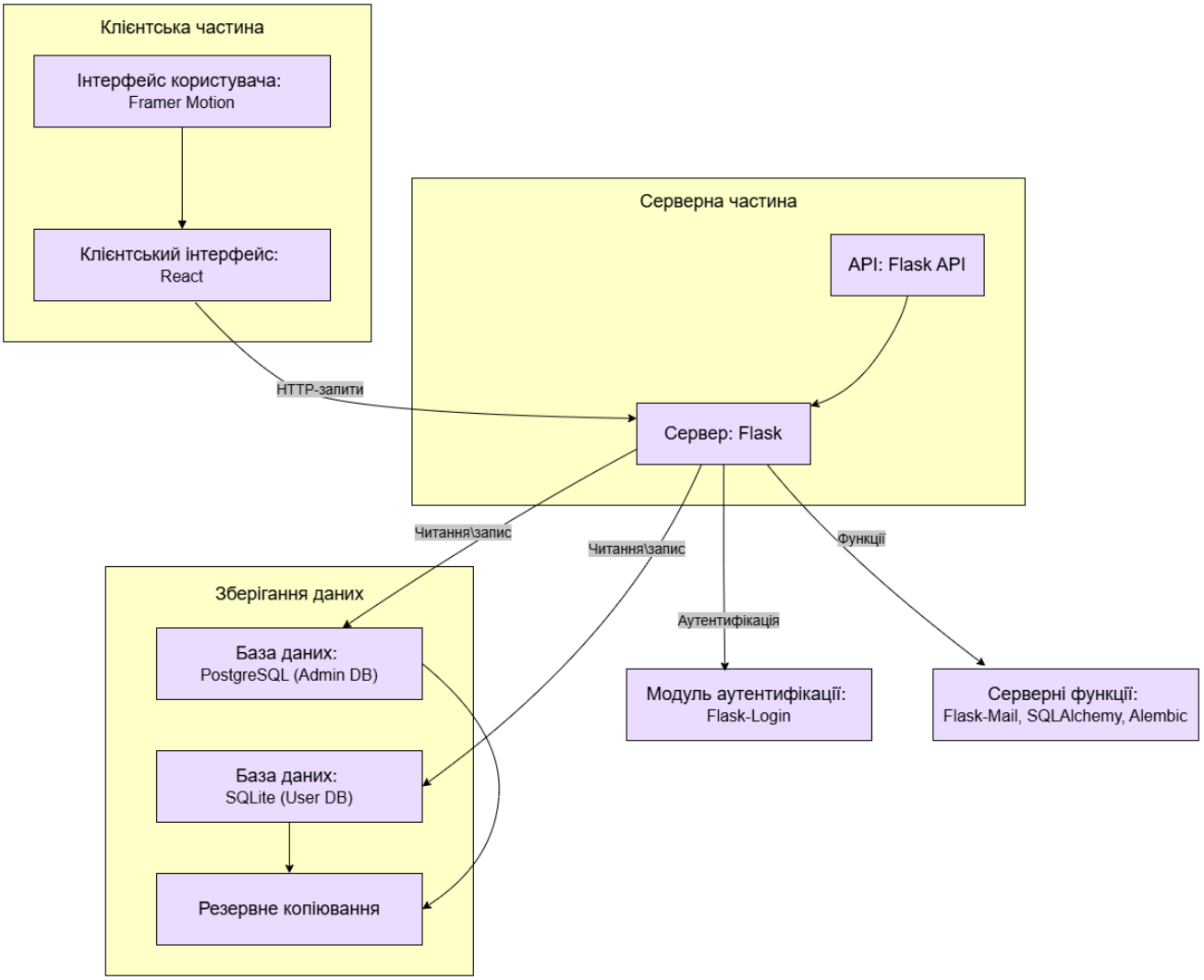

Рисунок 2.10 – Діаграма розгортання в нотації UML.

Джерело: розроблено автором.

Таким чином, архітектура системи STAYinTouch забезпечує ефективну взаємодію між клієнтом, сервером та базами даних, що дозволяє реалізувати ключові функціональні можливості платформи та гарантувати її стабільну роботу.

3.4 Проектування моделі бази даних

У цьому підрозділі представлені результати проектування бази даних, яка є основою для зберігання та обробки інформації про студентів, викладачів, курси та платіжні транзакції. Для досягнення оптимальної організації даних обрано технологію реляційних баз даних, яка дозволяє реалізувати зв'язки між сутностями.

Опис таблиць (рис. 2.11):

- Student: зберігає інформацію про студентів, зокрема ім'я, контактну інформацію та статус.
- Teacher: зберігає дані про викладачів, включаючи ім'я, спеціалізацію, контактні дані та доступність.
- Course: містить інформацію про курси, що пропонує школа, включаючи назву, опис, тривалість та рівень складності.
- Payment: зберігає платіжну інформацію, включаючи суму, дату оплати та інформацію про студента, до якого належить платіж.

Зв’язки між таблицями:

- Таблиця Student пов’язана з таблицею Teacher, що дозволяє відстежувати, хто з викладачів відповідає за навчання кожного студента.
- Таблиця Student також пов’язана з таблицею Payment через зовнішній ключ, що дозволяє контролювати платежі кожного студента, включаючи суму, дату та статус оплати.
- Зв'язок між таблицями Course та Student дозволяє фіксувати, які студенти записані на які курси, забезпечуючи належний розподіл навчальних матеріалів та ресурсів.

Структура бази даних забезпечує інтеграцію різних елементів, що дозволяє здійснювати аналіз даних і формувати звіти за різними параметрами [2].

Структура бази даних організована таким чином, щоб забезпечити збереження та обробку інформації про студентів, викладачів, курси та платежі в єдиній інтегрованій системі. Використання реляційної моделі бази даних дозволяє

здійснювати ефективне зберігання даних і підтримувати зв'язки між різними сутностями для подальшого аналізу та звітності. Зв'язки між таблицями дозволяють забезпечити цілісність даних і зручність доступу до інформації.

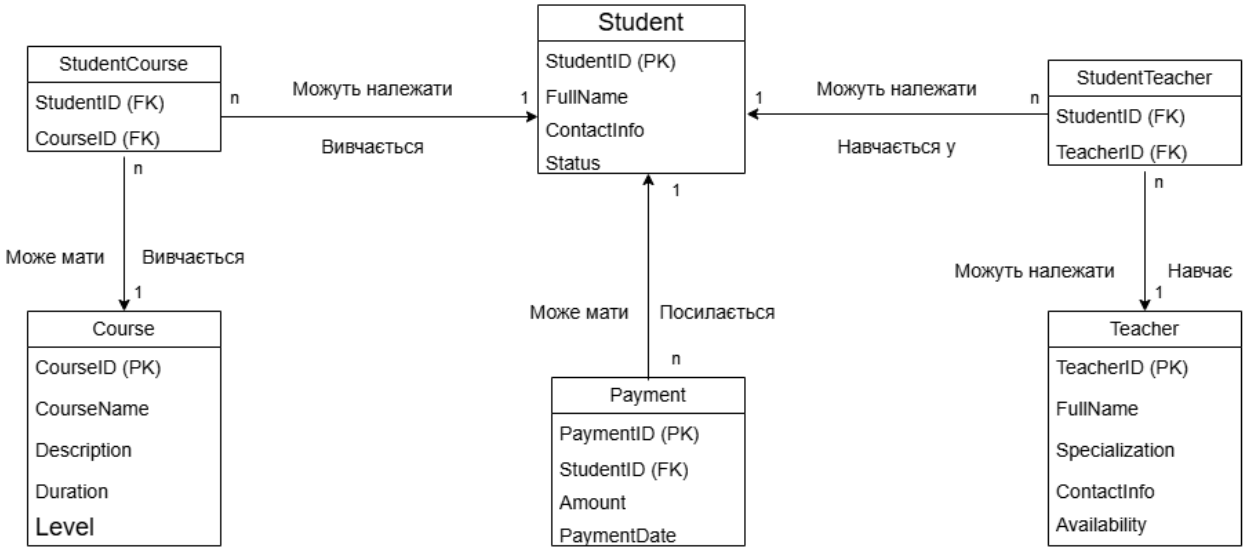

Рисунок 2.11 – Структура бази даних STAYinTouch.

Джерело: розроблено автором.

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1. Опис реалізації бази даних

У цьому розділі розглядається реалізація розробленої моделі бази даних вебзастосунку онлайн-школи STAYinTouch. Моделі бази даних, структура яких була представлена в попередньому розділі, реалізовані за допомогою SQLAlchemy і забезпечують зберігання ключових даних про систему, таких як користувачі, курси, платежі, розклад занять і історія уроків.

Таблиця реалізована класом User, структура якого наведена на рисунку 4.1. Вона зберігає основні дані користувача, включаючи ідентифікатор, ім'я, електронну пошту та пароль. Додаткові поля, такі як phone, course_type, level, payment_status, amount_due та due_date, дозволяють відстежувати контактну інформацію, тип курсу, рівень знань, статус оплати та фінансові зобов’язання користувача. Крім того, модель встановлює зв'язки між користувачами та викладачами, історією уроків і платежів, що забезпечує інтеграцію різних компонентів системи.

```python
class User(db.Model):
    __tablename__ = 'user'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    phone = db.Column(db.String(15), nullable=True)
    course_type = db.Column(SQLAlchemyEnum(CourseType, name="course_type_enum"), nullable=False)
    level = db.Column(db.String(50))  # e.g., 'Beginner', 'Intermediate'
    payment_status = db.Column(db.String(50))  # 'Paid', 'Unpaid', 'Pending'
    amount_due = db.Column(db.Float, nullable=True)  # Amount the student needs to pay
    due_date = db.Column(db.Date, nullable=True)  # Payment due date

    teacher_id = db.Column(db.Integer, db.ForeignKey('teacher.id'))

    # Relationships
    lesson_history = db.relationship('LessonHistory', backref='user', lazy=True)
    payment_history = db.relationship('Payment', back_populates='user', lazy=True)
```

Рисунок 4.1 – Модель бази даних для студентів.

Джерело: розроблено автором.

Таблиця реалізована класом Lesson, структура якого наведена на рисунку 4.2. Вона використовується для зберігання інформації про уроки, що проводяться в рамках навчального процесу онлайн-школи STAYinTouch. Таблиця включає основні атрибути уроку та встановлює зв'язки з іншими таблицями, такими як User (студенти) та Teacher (викладачі).

```python
class Lesson(db.Model):
    __tablename__ = 'lesson'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date_completed = db.Column(db.DateTime, nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    teacher_id = db.Column(db.Integer, db.ForeignKey('teacher.id'), nullable=True)
    approved = db.Column(db.Boolean, default=False)

    user = db.relationship('User', back_populates='lessons')
    teacher = db.relationship('Teacher', back_populates='lessons')
```

Рисунок 4.2 – Модель уроку для управління розкладом

Джерело: розроблено автором.

Ця модель дозволяє ефективно зберігати дані про уроки, відслідковувати їхній статус, призначення викладачів та учнів, а також управляти інформацією про завершення та затвердження занять.

Таблиця реалізована класом Teacher (рис. 4.3), структура якого наведена на відповідному рисунку. Вона використовується для зберігання інформації про викладачів онлайн-школи STAYinTouch. Таблиця містить основні поля для персональних даних викладача, контактної інформації, кваліфікацій та статусу активності. Крім того, вона забезпечує встановлення зв'язків з іншими таблицями, такими як User, Lesson та TeacherRecommendation.

```python
class Teacher(db.Model):
    __tablename__ = 'teacher'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    last_name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    phone = db.Column(db.String(20), nullable=True)
    qualifications = db.Column(db.Text, nullable=True)
    is_online = db.Column(db.Boolean, default=True)
    password = db.Column(db.String(200))

    # Existing relationships
    users = db.relationship('User', backref='teacher', lazy=True)

    # Updated relationship with Lesson
    lessons = db.relationship('Lesson', back_populates='teacher')

    # Updated recommendations relationship
    recommendations = db.relationship('TeacherRecommendation', back_populates='teacher')


    # Password methods
    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)
```

Рисунок 4.3 – Модель бази даних для вчителя

Джерело: розроблено автором.

Ця модель і її функціонал дозволяють ефективно керувати інформацією про викладачів, забезпечувати їхню автентифікацію та інтеграцію з іншими аспектами навчального процесу, такими як управління уроками та рекомендаціями.

Таблиця реалізована класом AdminUser (рис. 4.4), структура якого наведена на відповідному рисунку. Вона використовується для зберігання інформації про адміністративних користувачів онлайн-школи STAYinTouch. Таблиця містить основні поля для ідентифікації адміністратора, контактної інформації, а також визначення його ролі та прав доступу. Крім того, вона підтримує автентифікацію та авторизацію адміністраторів через стандартні методи Flask-Login.

```python
class AdminUser(db_admin.Model, UserMixin):
    __tablename__ = 'admin_user'

    id = db_admin.Column(db_admin.Integer, primary_key=True)
    name = db_admin.Column(db_admin.String(100), nullable=False)
    email = db_admin.Column(db_admin.String(120), unique=True, nullable=False)
    password = db_admin.Column(db_admin.String(128), nullable=False)
    phone = db_admin.Column(db_admin.String(20))
    is_admin = db_admin.Column(db_admin.Boolean, default=False, nullable=False)
    role = db_admin.Column(db_admin.String(50), nullable=False, default='admin')

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)

    @property
    def is_active(self):
        return True

    @property
    def is_authenticated(self):
        return True

    @property
    def is_anonymous(self):
        return False
```

Рисунок 4.4 – Модель бази даних для адміністратора

Джерело: розроблено автором.

Ця модель забезпечує безпечне керування обліковими записами адміністративних користувачів, дозволяє гнучко налаштовувати ролі та права доступу, а також інтегрується з системами автентифікації Flask.

Таблиця реалізована класом TeacherRecommendation (рис. 4.5), структура якого наведена на відповідному рисунку. Вона використовується для зберігання відгуків і рекомендацій користувачів щодо викладачів у системі онлайн-школи STAYinTouch. Таблиця забезпечує зв’язок між користувачами та викладачами, дозволяючи відображати та аналізувати відгуки про викладачів.

```python
class TeacherRecommendation(db.Model):
    __tablename__ = 'teacher_recommendation'

    id = db.Column(db.Integer, primary_key=True)
    recommendation_text = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    teacher_id = db.Column(db.Integer, db.ForeignKey('teacher.id'), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.now, nullable=False)

    user = db.relationship('User', back_populates='teacher_recommendations')
    teacher = db.relationship('Teacher', back_populates='recommendations')
```

Рисунок 4.5 – Модель бази даних для відправлення рекомендацій

Джерело: розроблено автором.

У цьому підрозділі було проаналізовано структуру бази даних вебзастосунку онлайн-школи STAYinTouch, розроблену з використанням ORM-бібліотеки SQLAlchemy. Представлені моделі забезпечують комплексне управління даними про користувачів, викладачів, уроки, платежі та рекомендації. Встановлені між моделями зв’язки дозволяють інтегрувати навчальний процес із фінансовими операціями та адміністративними функціями, створюючи єдину цілісну систему. Така архітектура сприяє ефективному управлінню навчальним процесом, полегшує адміністрування та забезпечує гнучкість для подальшого розвитку і масштабування функціоналу системи.

4.2. Реалізація функціональних можливостей

У цьому підрозділі описані ключові функціональні можливості вебзастосунку STAYinTouch, реалізовані за допомогою мови Python та фреймворку Flask. Зокрема, розглядаються функції реєстрації, авторизації, управління розкладом, відправки електронних повідомлень і обробки платежів.

Функціонал реєстрації та авторизації користувачів у системі забезпечує надійний та безпечний доступ до її ресурсів. Реалізація передбачає використання сучасних технологій та інструментів, таких як Flask-WTF для ефективної обробки веб-форм і Flask-Bcrypt для хешування паролів, що підвищує рівень захисту персональних даних. Код функції реєстрації наведено нижче (рис. 4.6, 4.7).

```python
@main.route('/register_user', methods=['POST', 'GET'])
def register_user():
    from .forms import RegistrationForm
    form = RegistrationForm()

    try:
        if form.validate_on_submit():
            name = form.name.data
            email = form.email.data
            phone = form.phone.data
            password = form.password.data
            course_type = form.course_type.data

            # Log form submission
            current_app.logger.info("Form submitted with name: %s, email: %s", name, email)

            # Check if user already exists
            existing_user = User.query.filter_by(email=email).first()
            if existing_user:
                current_app.logger.warning("Registration attempt with already registered email: %s", email)
                flash("Email already registered. Please log in.", 'error')
                return redirect(url_for('main.home'))  # Redirect to home or relevant page

            # Create new user and hash password
            new_user = User(
                name=name,
                email=email,
                phone=phone,
                course_type=course_type,
            )
```

Рисунок 4.6 – Функції реєстрації користувачів

Джерело: розроблено автором.

```
            new_user.set_password(password)

            # Save to database
            db.session.add(new_user)
            db.session.commit()
            current_app.logger.info("New user registered: %s", email)

            # Store user ID in session
            session['user_id'] = new_user.id

            flash("Registration successful!", 'success')
            return redirect(url_for('main.profile'))  # Redirect to profile page after successful registration
        else:
            current_app.logger.warning("Form validation failed: %s", form.errors)
            flash("Please correct the errors in the form.", 'error')
            return render_template('index.html', form=form)  # Pass the form back to the template for corrections
    except Exception as e:
        # Log unexpected errors
        current_app.logger.error("Error during registration: %s", e, exc_info=True)
        flash("An unexpected error occurred. Please try again later.", 'error')

    return render_template('index.html', form=form)
```

Рисунок 4.7 – Продовження функції реєстрації користувачів

Джерело: розроблено автором.

Функція login дозволяє користувачам виконувати вхід у систему. Спочатку створюється форма авторизації LoginForm, яка обробляє введені дані. Перевірка введених даних здійснюється методом validate_on_submit(), який гарантує коректність інформації та забезпечує захист через CSRF-токен.

Далі з форми отримуються електронна пошта та пароль, після чого система шукає користувача в базі даних за електронною поштою. Якщо користувач не знайдений, виводиться відповідне повідомлення. У разі успішного знаходження користувача перевіряється правильність введеного пароля шляхом порівняння з хешем, збереженим у базі даних. Якщо паролі співпадають, користувач автентифікується, а в сесію зберігається його ідентифікатор. Після цього користувач перенаправляється на сторінку профілю. Якщо пароль невірний або є інші помилки, відображається повідомлення про помилку, а форма залишається доступною для повторного заповнення (рис. 4.8).

```python
@main.route('/login', methods=['GET', 'POST'])
def login():
    from project.forms import LoginForm
    login_form = LoginForm()
    # Debugging: Print CSRF token to verify
    print(f"Login Form CSRF: {login_form.csrf_token.data}")
    print(f"CSRF Token Exists: {login_form.csrf_token}")

    if login_form.validate_on_submit():
        email = login_form.email.data
        password = login_form.password.data
        # Debugging statements
        print(f"Login attempt: Email={email}, Password Provided={password}")

        user = User.query.filter_by(email=email).first()

        if not user:
            flash("User not found. Please check your email.", 'error')
            print("No user found with the provided email.")
            return redirect(url_for('main.login'))

        print(f"User password hash: {user.password_hash}")
        if user.check_password(password):
            login_user(user)
            session['user_id'] = user.id
            flash(f"Welcome back, {user.name}!", 'success')
            print("User authenticated successfully.")
            return redirect(url_for('main.profile'))
        else:
            flash("Invalid email or password. Please try again.", 'error')
            print("Password mismatch.")

    return render_template('base.html', login_form=login_form)
```

Рисунок 4.8 – Функції авторизації користувачів

Джерело: розроблено автором.

Маршрути забезпечують взаємодію користувачів і викладачів із системою. Вони реалізують логіку для обробки HTTP-запитів, виконання операцій у базі даних та відображення відповідей. Наприклад, маршрут для відправки повідомлення студенту продемонстрований на рисунку 4.9.

```python
# Send messages
@admin.route('/admin/send_message', methods=['POST'])
def send_message():

    attach_user_db()

    from project.models import User

    # Check if user_db_conn exists before querying
    if g.user_db_conn is None:
        flash('User database connection not established', 'error')
        return redirect(url_for('admin.student_management'))

    student_id = request.form.get('student_id')  # Get the student ID from the form
    user = User.query.get(student_id)  # Query from the 'User' model
    message_content = request.form.get('message')  # Get the message content from the form

    if user:  # Proceed only if the user exists
        try:
            msg = Message("Important Update", recipients=[user.email])
            msg.body = f"Dear {user.name},\n\n{message_content}"
            mail.send(msg)
            flash(f'Message sent successfully to {user.name}', 'success')
        except Exception as e:
            flash(f"An error occurred while sending the message: {e}", 'error')
    else:
        flash('Student not found.', 'danger')

    return redirect(url_for('admin.student_management'))
```

Рисунок 4.9 – Маршрут для відправки повідомлення студенту

Джерело: розроблено автором.

Ця модель зберігає такі основні дані про користувача, як ідентифікатор, ім'я, електронну пошту та пароль. Поля phone, course_type, level, payment_status, amount_due та due_date використовуються для відстеження контактної інформації, типу курсу, рівня знань, статусу оплати та фінансових зобов'язань користувача. Крім цього, модель встановлює зв'язки між користувачами і викладачами, а також зберігає історію уроків та платежів. Це забезпечує інтеграцію системи для управління розкладом занять, відправки повідомлень студентам і контролю фінансових операцій.

Маршрут для додавання нового уроку продемонстрований на рисунку 4.10.

```python
@admin.route('/add_lesson', methods=['POST'])
def add_lesson():
    from project.models import LessonHistory
    course_id = request.form.get('course_id')
    topic = request.form.get('topic')
    date = request.form.get('date')

    new_lesson = LessonHistory(course_id=course_id, topic=topic, date=date)
    db_admin.session.add(new_lesson)
    db_admin.session.commit()

    flash('Урок успішно додано!', 'success')
    return redirect(url_for('teacher.view_schedule'))
```

Рисунок 4.10 – Маршрут для додавання нового уроку

Джерело: розроблено автором.

Цей маршрут дозволяє викладачу створювати нові заняття, передаючи дані через форму.

Конфігурація SMTP-сервера для надсилання повідомлень через бібліотеку Flask-Mail наведена на рисунку 4.11.

```python
# Flask-Mail configuration
app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USERNAME'] = os.getenv('MAIL_USERNAME')
app.config['MAIL_PASSWORD'] = os.getenv('MAIL_PASSWORD')
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False
app.config['MAIL_DEFAULT_SENDER'] = os.getenv('MAIL_USERNAME')
```

Рисунок 4.11 – Конфігурація SMTP-сервера

Джерело: розроблено автором.

Функція для відправки електронних листів на рисунку 4.12.

```python
@admin.route('/admin/send_payment_reminder/<int:user_id>')
def send_payment_reminder(user_id):
    # Attach the user database
    attach_user_db()

    # Check if user_db_conn exists
    if g.user_db_conn is None:
        flash('User database connection not established', 'error')
        return redirect(url_for('admin.student_management'))  # Redirect to a safe route

    try:
        # Use SQLAlchemy to query the user
        from project.models import User, Payment

        user = User.query.get(user_id)

        if not user:
            flash('User not found.', 'danger')
            return redirect(url_for('admin.student_management'))

        # Get the most recent payment record
        latest_payment = Payment.query.filter_by(student_id=user_id).order_by(Payment.date.desc()).first()

        if latest_payment and latest_payment.status == 'Unpaid':
            # Send email reminder
            msg = Message("Payment Reminder", recipients=[user.email])
            msg.body = f"Dear {user.name},\n\nYour payment of ${latest_payment.amount_due} is due."
            mail.send(msg)
            flash('Payment reminder sent successfully', 'success')
        else:
            flash('No unpaid payment found for this user.', 'info')

    except Exception as e:
        flash(f"An error occurred while sending the payment reminder: {e}", 'error')
    return redirect(url_for('admin.student_management'))
```

Рисунок 4.12 – Функція для відправки електронних листів

Джерело: розроблено автором.

Ця функція використовується в різних частинах системи, наприклад, для відправки підтвердження реєстрації або нагадувань про оплату.

Функція `payment_gateway` перевіряє авторизацію користувача, шукає його в базі даних і перевіряє наявність заборгованості. Якщо заборгованість є, формується запит до платіжної системи Fondy для створення платіжної сесії. Якщо запит успішний, користувача перенаправляють на сторінку платежу, інакше

відображається помилка. Функція може бути адаптована для інших платіжних систем, таких як Stripe чи PayPal (рис. 4.13).

```python
@main.route('/payment_gateway', methods=['GET', 'POST'])
@login_required
def payment_gateway():
    import requests
    import time
    if 'user_id' not in session:
        flash("Please log in to proceed with payment.", 'warning')
        return redirect(url_for('main.home'))

    user_id = session['user_id']
    user = User.query.get_or_404(user_id)
    if not user or user.amount_due <= 0:
        flash("No pending payments or user not found.", 'info')
        return redirect(url_for('main.payment'))
    # Fondy payment integration
    fondy_url = "https://pay.fondy.eu/api/checkout/url/"
    data = {
        "merchant_id": "<YOUR_MERCHANT_ID>", "order_id": f"order_{user_id}_{int(time.time())}",
        "order_desc": "Payment for course subscription", "amount": int(user.amount_due * 100),
        "currency": "UAH",
        "server_callback_url": url_for('main.payment_callback', _external=True),
        "response_url": url_for('main.payment', _external=True),
    }
    headers = {"Content-Type": "application/json"}
    try:
        response = requests.post(fondy_url, json=data, headers=headers)
        response_data = response.json()
        if response_data.get("response").get("status") == "success":
            return redirect(response_data["response"]["checkout_url"])
        else:
            flash("Error connecting to payment gateway. Please try again.", "danger")
    except Exception as e:
        current_app.logger.error(f"Fondy API error: {e}")
        flash("Payment gateway error. Please try again.", "danger")
        return redirect(url_for('main.payment'))
    return render_template('payment_gateway.html', user=user)
```

Рисунок 4.13 – Код реалізації платіжної системи

Джерело: розроблено автором.

Програмна реалізація системи STAYinTouch забезпечує ефективну взаємодію між адміністраторами, викладачами та студентами. Використання Flask, SQLAlchemy та інших бібліотек дозволило створити гнучкий і функціональний вебзастосунок, що відповідає потребам сучасного онлайн-навчання.

4.3. Настанови з використання вебдодатку

4.3.1 Настанова з використання додатка в профілі студента

У цьому розділі представлено порядок використання вебдодатку підтримки діяльності онлайн школи STAYinTouch.

На головній сторінці вебдодатку (рисунок 4.14) розміщено основну інформацію про школу та її послуги. Головна сторінка створена для ознайомлення користувачів із платформою та включає наступні елементи:

- Меню навігації, яке дозволяє переходити до розділів "Про нас", "Курси", "Контакти" та інших сторінок.
- Блок привітання, що містить короткий опис школи STAYinTouch та її місії, а також кнопку для швидкого переходу до реєстрації або авторизації.

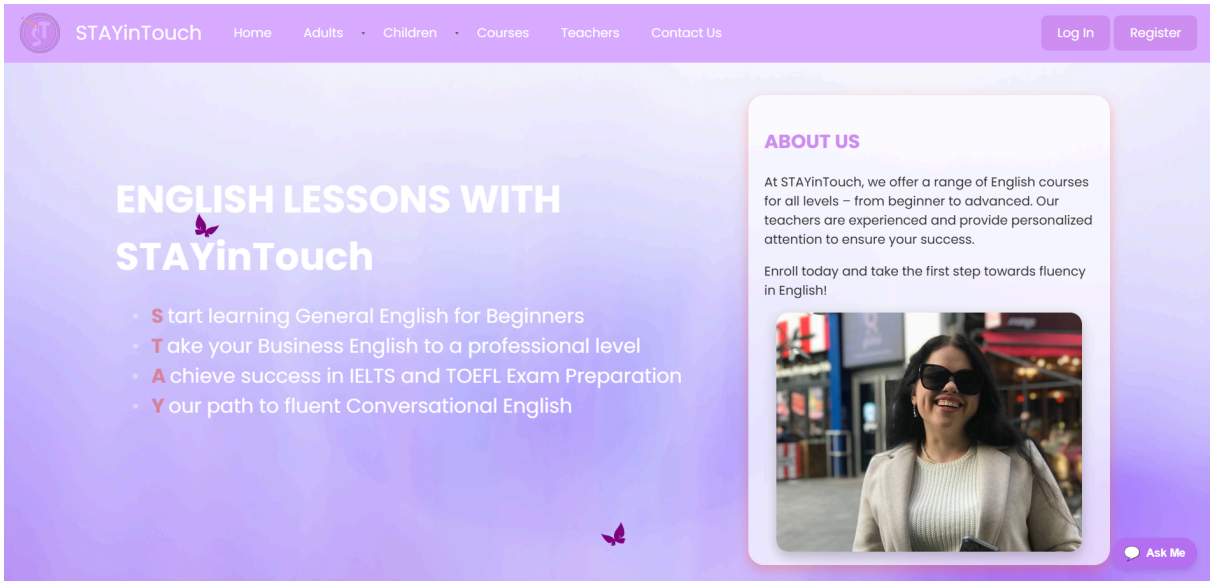

Рисунок 4.14 – Головна сторінка вебдодатку

Джерело: розроблено автором.

- Розділ з популярними курсами, де представлено кілька основних програм навчання з коротким описом та посиланнями на детальну інформацію.
- Контактна інформація, включаючи адресу електронної пошти, телефон, а також форму для надсилання запиту.

Ця сторінка забезпечує інтуїтивну взаємодію користувачів із платформою та дає змогу швидко зорієнтуватися в основних можливостях вебдодатку.

На рисунку 4.15 зображено перший етап взаємодії з вебдодатком – авторизацію. Користувач має можливість зареєструватися, натиснувши кнопку "Register", та заповнити реєстраційну форму із зазначенням особистих даних, таких як електронна пошта, пароль, ім’я та інші необхідні поля.

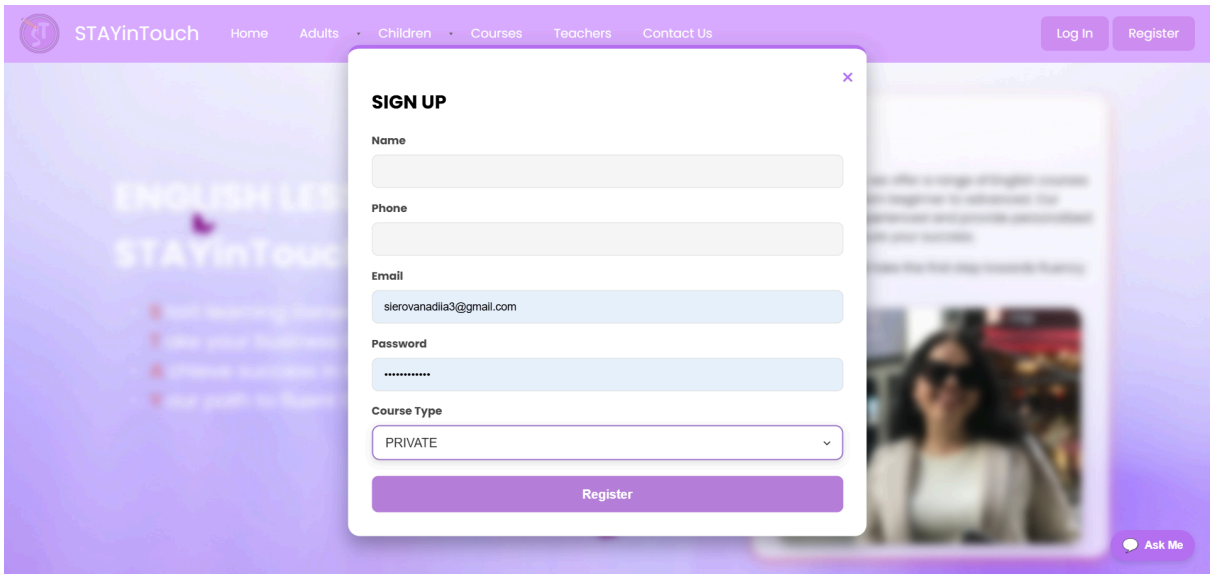

Рисунок 4.15 – Авторизація користувача. Реєстрація

Джерело: розроблено автором.

Після успішної реєстрації користувач може виконати авторизацію, ввівши електронну пошту та пароль, а потім натиснути кнопку "Login", щоб отримати доступ до особистого кабінету (рис. 4.16).

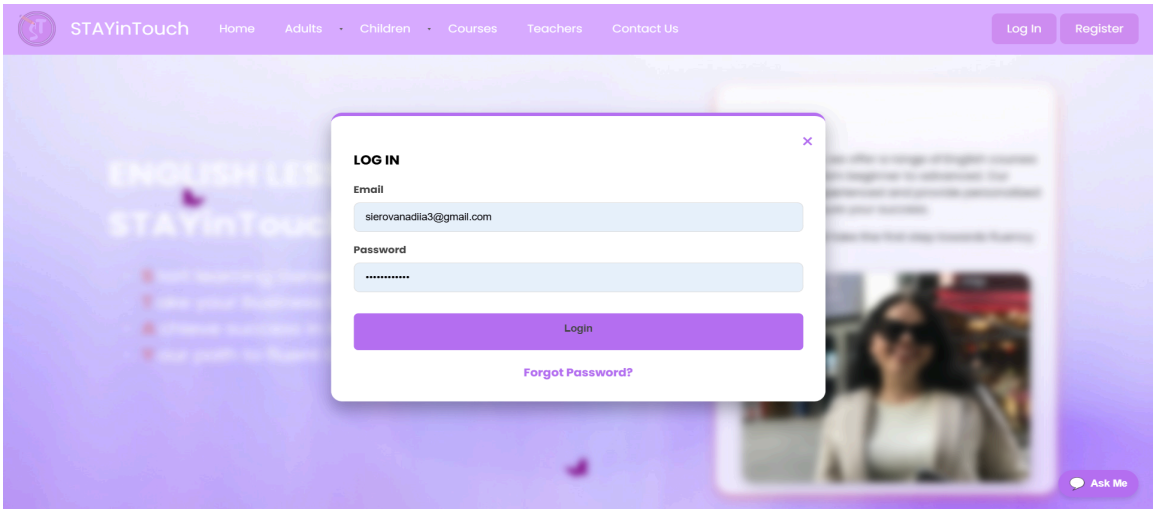

Рисунок 4.16 – Авторизація користувача. Логін

Джерело: розроблено автором.

Після входу в систему користувач потрапляє до свого особистого кабінету, де може переглянути інформацію про свій прогрес, оплату та розклад занять (рис. 4.17).

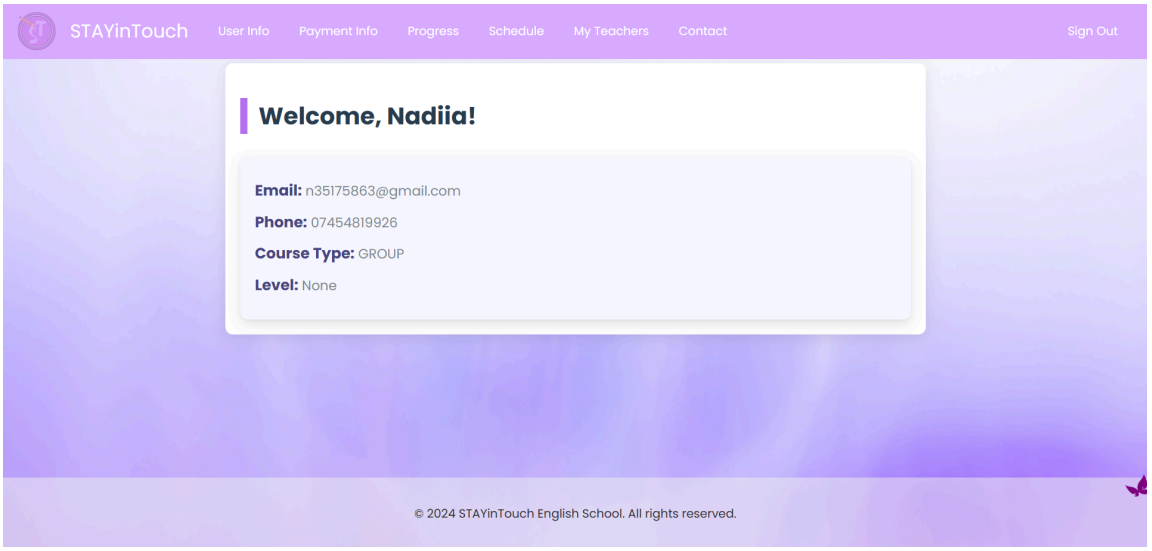

Рисунок 4.17 – Особистий кабінет користувача вебдодатку

Джерело: розроблено автором.

У вкладці "Мій прогрес" відображено інформацію про пройдені уроки, результати тестів та рекомендації викладача (рис. 4.18).

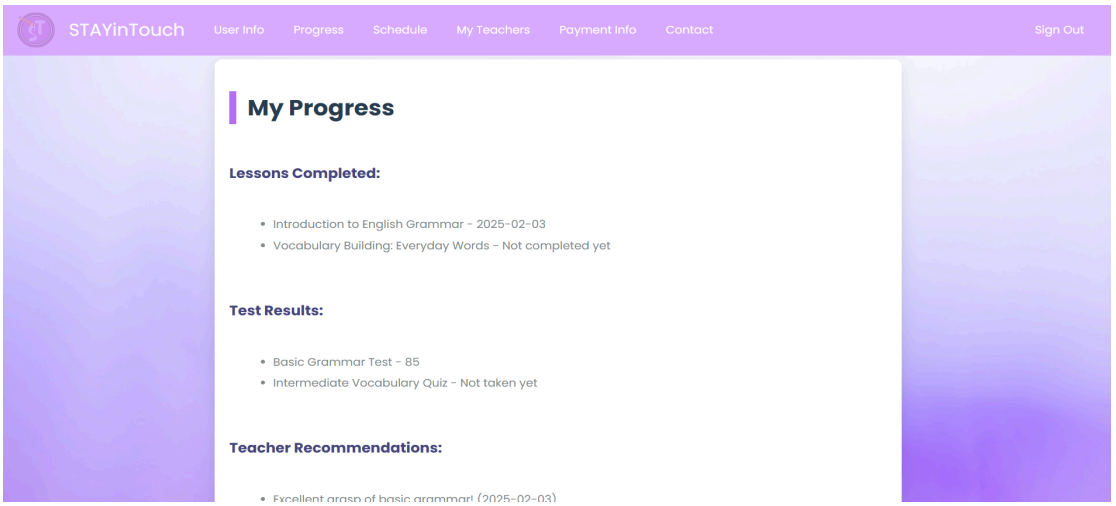

Рисунок 4.18 – Особистий кабінет користувача з інформацією про прогрес навчання

Джерело: розроблено автором.

На рисунку 4.19 представлено сторінку з розкладом занять. Користувач може переглядати дати та час запланованих уроків.

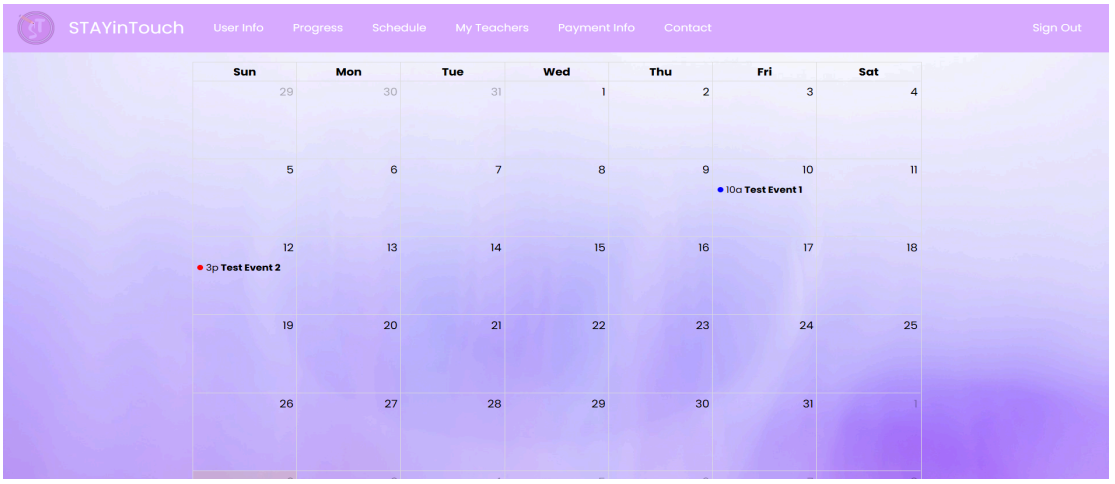

Рисунок 4.19 – Розклад занять у особистому кабінеті

Джерело: розроблено автором.

Якщо виникає необхідність зв’язатися з адміністратором чи викладачем, користувач може скористатися формою відправки повідомлень (рис. 4.20). Для цього потрібно обрати отримувача, ввести текст повідомлення та натиснути "Відправити".

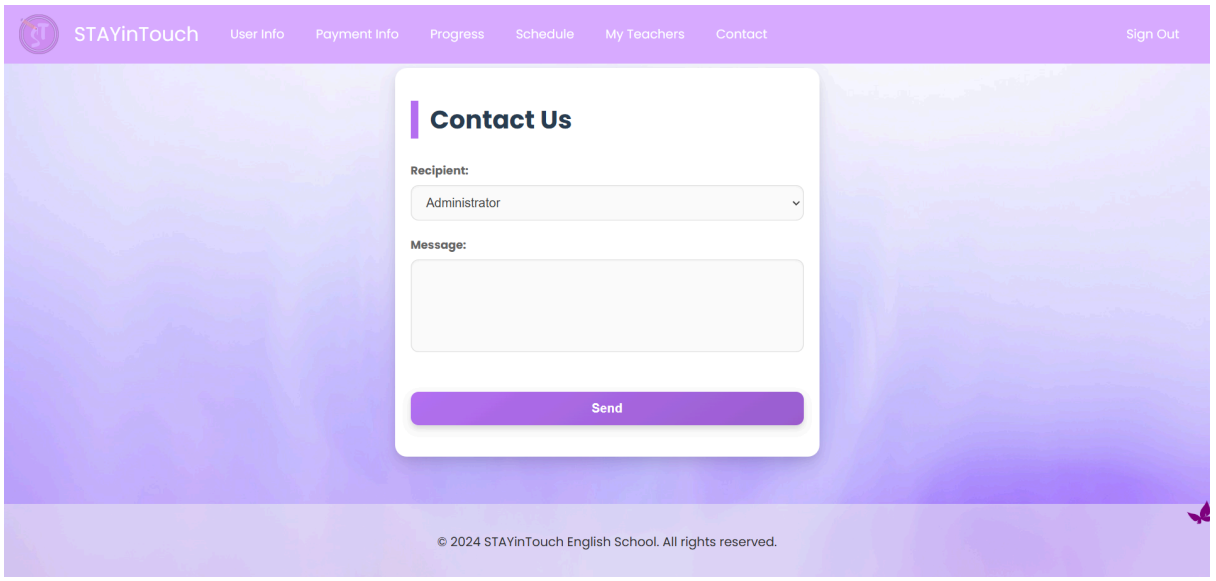

Рисунок 4.20 – Форма відправки повідомлень у особистому кабінеті

Джерело: розроблено автором.

На рисунку 4.21 зображено сторінку управління оплатами. Тут користувач може переглянути свій статус оплати, суму заборгованості та крайню дату платежу. У разі потреби користувач може натиснути на кнопку "Оплатити", щоб перейти до платіжної системи.

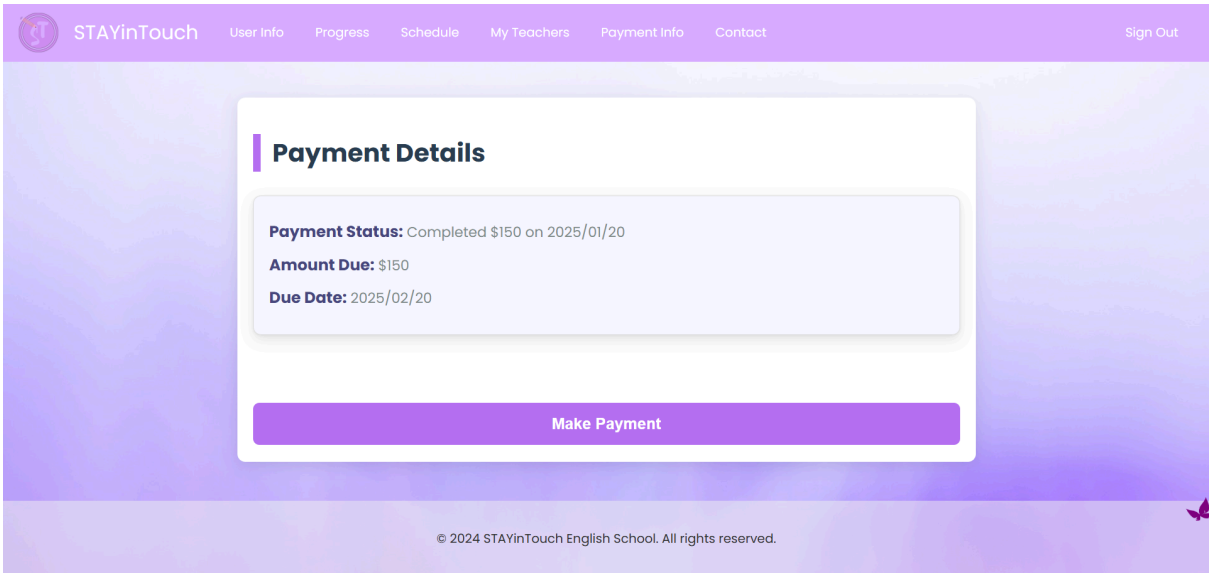

Рисунок 4.21 – Сторінка управління оплатами

Джерело: розроблено автором.

На рисунку 4.22 представлено вкладку "Мої викладачі", де відображається інформація про призначених викладачів, їх контактні дані, а також графік доступності для індивідуальних консультацій.

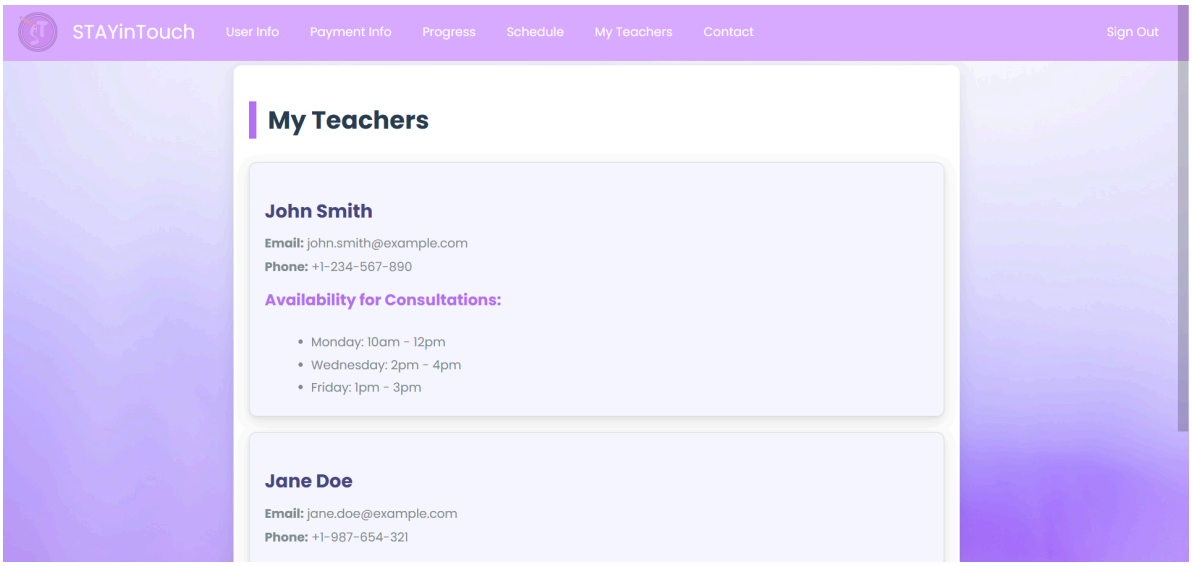

Рисунок 4.22 – Вкладка з інформацією про викладачів

Джерело: розроблено автором.

Таким чином, вебдодаток STAYinTouch забезпечує зручний інтерфейс для взаємодії між студентами, адміністрацією та викладачами, сприяючи ефективній організації навчального процесу.

4.3.2 Настанова з використання додатка в профілі викладача

У вебдодатку STAYinTouch передбачено окремий профіль для викладачів, який містить усі необхідні інструменти для організації навчального процесу та взаємодії зі студентами й адміністрацією школи. Інтерфейс профілю викладача є зрозумілим та інтуїтивно зручним, що сприяє ефективному виконанню викладацьких обов'язків.

Головна сторінка профілю викладача (рис. 4.23) – це панель керування, яка містить загальну інформацію про викладача, його заплановані уроки та список майбутніх занять. Викладач може переглядати розклад своїх уроків, що дозволяє йому ефективно планувати час та своєчасно готуватися до занять.

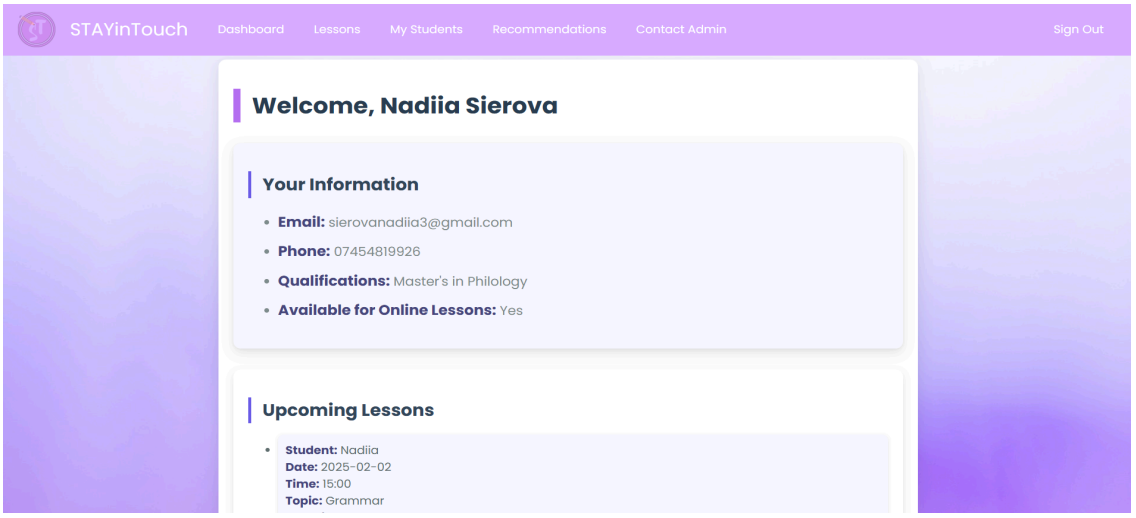

Рисунок 4.23 – Головна сторінка профілю викладача

Джерело: розроблено автором.

У вкладці "My Students" викладач має доступ до списку своїх студентів. Тут відображається детальна інформація про кожного учня, включаючи їхні імена, контактні дані, рівень знань, результати тестів та прогрес у навчанні. Завдяки цій функції викладач може відстежувати успіхи студентів та адаптувати програму навчання відповідно до їхніх потреб (рис. 4.24).

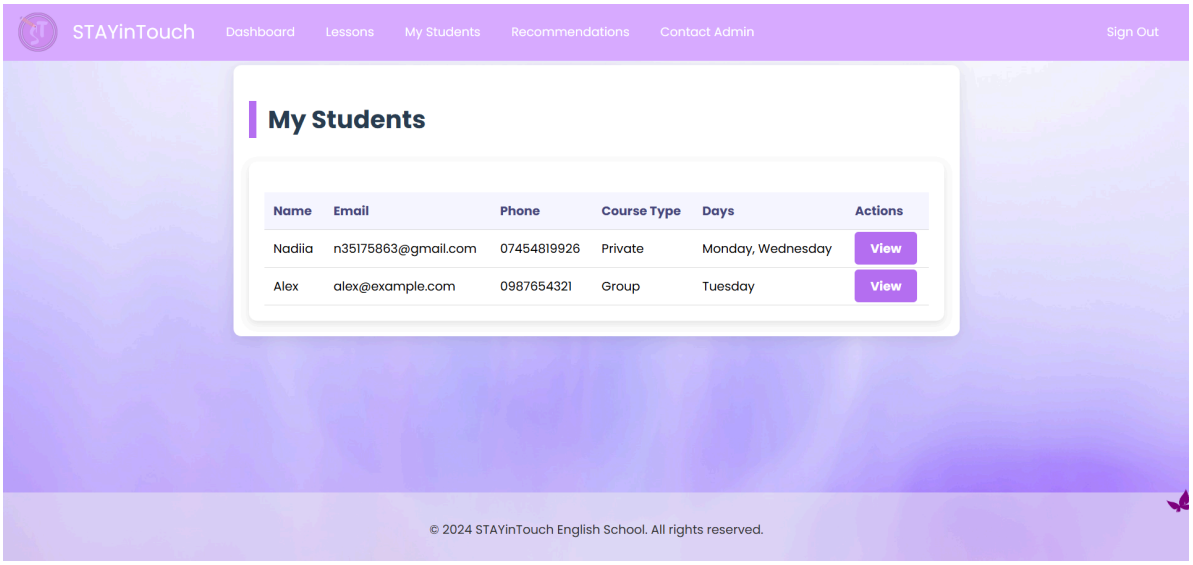

Рисунок 4.24 – Сторінка керування базою студентів

Джерело: розроблено автором.

Функція "Send Recommendation" дозволяє викладачу надавати студентам персоналізовані рекомендації щодо покращення їхніх навичок. Викладач може обрати конкретного студента, сформулювати рекомендацію та надіслати її безпосередньо через платформу. Це сприяє індивідуальному підходу до навчання та підтримує мотивацію студентів (рис. 4.25).

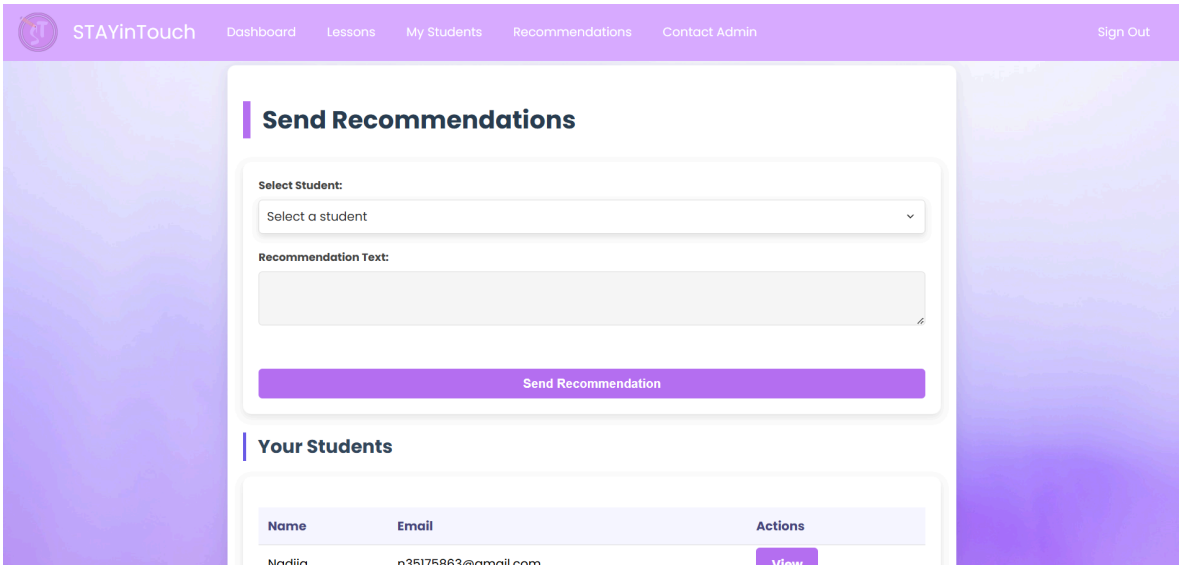

Рисунок 4.25 – Сторінка функції відправки рекомендацій

Джерело: розроблено автором.

Для зручного спілкування з адміністрацією школи викладачі можуть використовувати функцію "Contact Admin" (рис. 4.26). Вона дозволяє надсилати запити, повідомлення або уточнення щодо організаційних питань, розкладу чи інших аспектів роботи. Це забезпечує ефективну комунікацію між викладачами та адміністрацією, що сприяє злагодженій роботі школи.

Таким чином, профіль викладача у вебдодатку STAYinTouch забезпечує всі необхідні інструменти для зручної взаємодії зі студентами, адміністрування навчального процесу та оперативного зв’язку з адміністрацією школи.

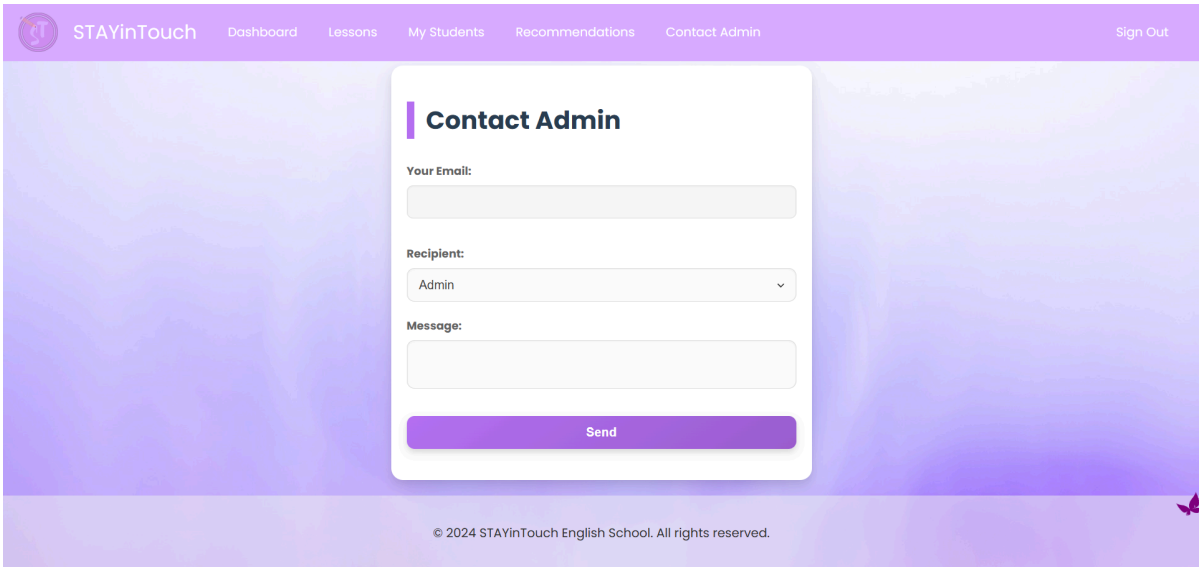

Рисунок 4.26 – Форма відправки повідомлень у кабінеті викладача

Джерело: розроблено автором.

4.3.3 Настанова з використання додатка в профілі адміністратора

У вебдодатку STAYinTouch передбачено окремий профіль для адміністратора, який містить всі необхідні інструменти для управління школою, викладачами та студентами. Інтерфейс адміністратора дозволяє ефективно відстежувати діяльність школи та забезпечувати організацію навчального процесу.

Головна сторінка профілю адміністратора (рис. 4.27) – це панель керування, яка містить основну інформацію про школу, включаючи кількість зареєстрованих студентів, кількість поданих заявок на роботу викладачів та останні події. Ця панель дає можливість адміністраторам швидко отримувати актуальні дані про стан школи та приймати необхідні рішення.

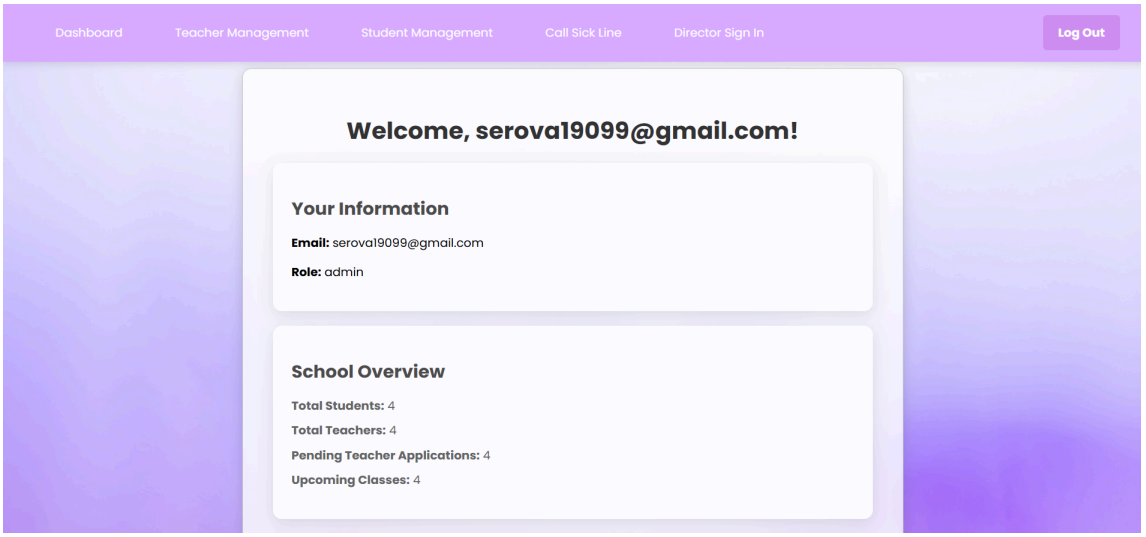

Рисунок 4.27 – Головна сторінка профілю адміністратора

Джерело: розроблено автором.

Управління викладачами відбувається через модуль "Teacher Management", який містить дві основні вкладки:

- Teacher Applications – сторінка, де відображаються заявки на роботу викладачів. Адміністратор може переглядати подані заявки, аналізувати кандидатів та ухвалювати рішення щодо їхнього прийому на роботу (рис. 4.28).

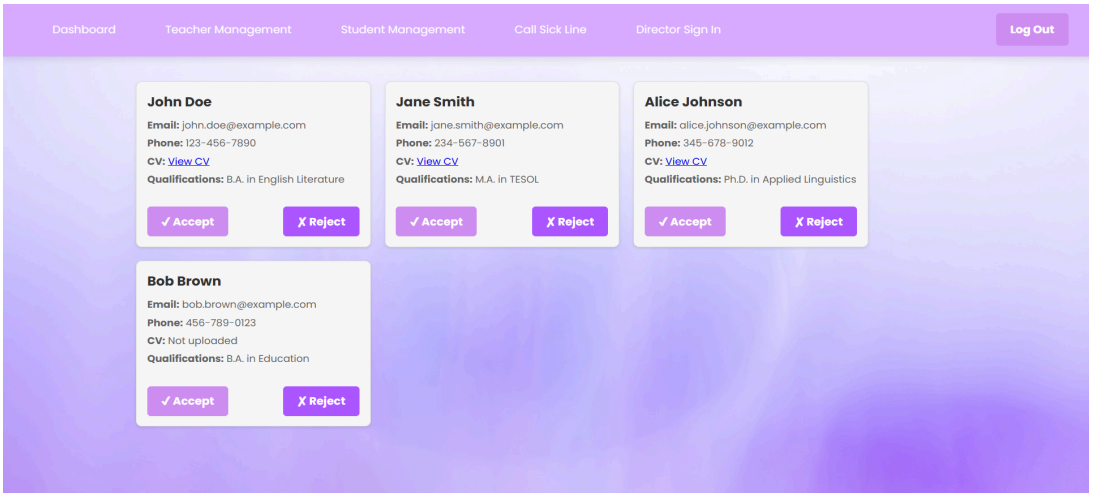

Рисунок 4.28 – Сторінка з таблицею заявок на роботу викладачів

Джерело: розроблено автором.

- View Teachers – список викладачів, які вже працюють у школі. Тут можна переглядати інформацію про викладачів, включаючи їхні контактні дані, дисципліни, які вони викладають, та статус їхньої активності в системі (рис. 4.29).

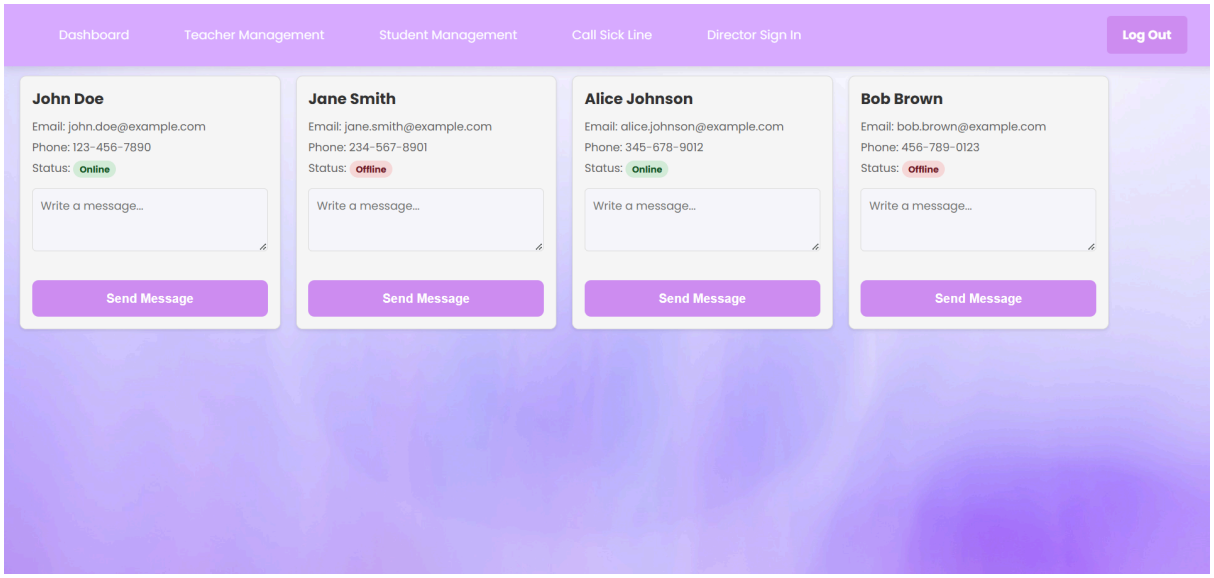

Рисунок 4.29 – Сторінка з інформацією про викладачів

Джерело: розроблено автором.

Для управління студентами адміністратор використовує модуль "Student Management", який включає вкладку View Students (рис. 4.30). На цій сторінці можна переглядати список студентів із детальною інформацією про кожного з них, включаючи прогрес у навчанні, контактні дані та інші важливі аспекти.

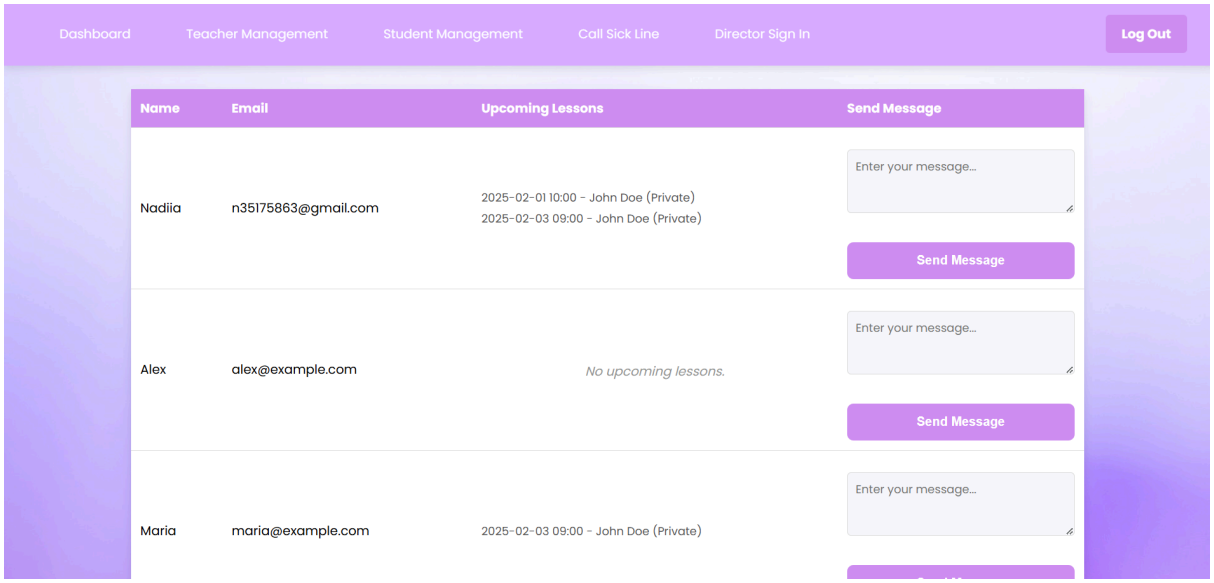

Рисунок 4.30 – Сторінка з інформацією про студентів

Джерело: розроблено автором.

У вкладці "Payment Status" адміністратор може переглядати інформацію про оплату за навчання. Відображаються дані щодо статусу платежів, заборгованостей та крайньої дати оплати (рис. 4.31).

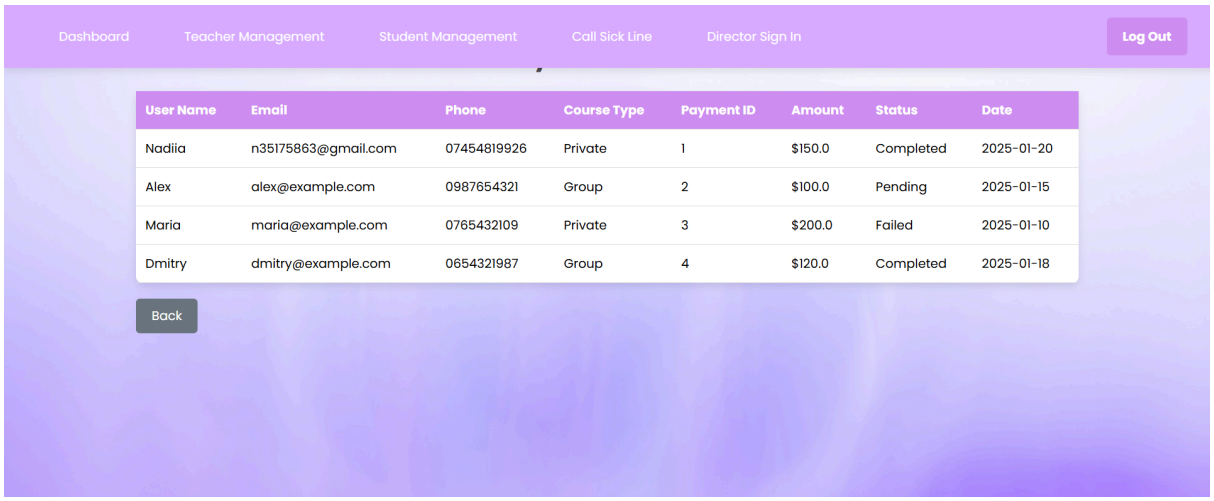

Рисунок 4.31 – Сторінка з інформацією про оплату

Джерело: розроблено автором.

Функція "Upcoming Lessons" дозволяє адміністраторам переглядати найближчі заплановані уроки. Це допомагає контролювати навчальний процес і забезпечувати відповідну організацію занять (рис. 4.32).

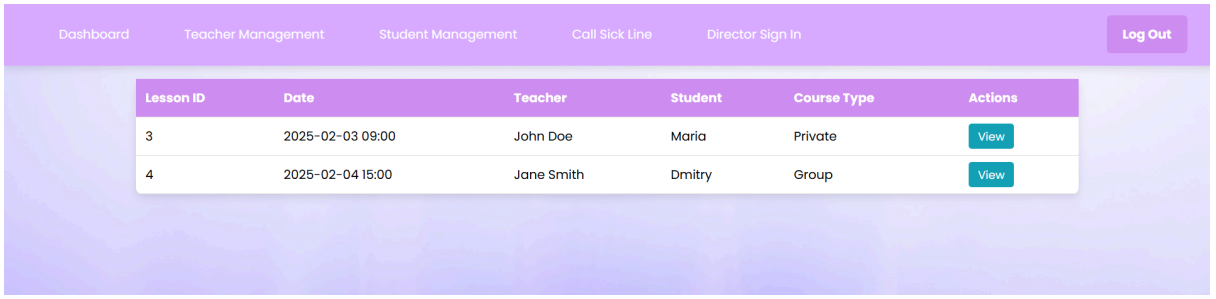

Рисунок 4.32 – Сторінка для перегляду найближчих запланованих уроків

Джерело: розроблено автором.

У вкладці "Call Sick Line" міститься інформація про викладачів та студентів, які повідомили про хворобу. Це дозволяє адміністраторам своєчасно реагувати на відсутність учасників навчального процесу та коригувати розклад занять (рис. 4.33).

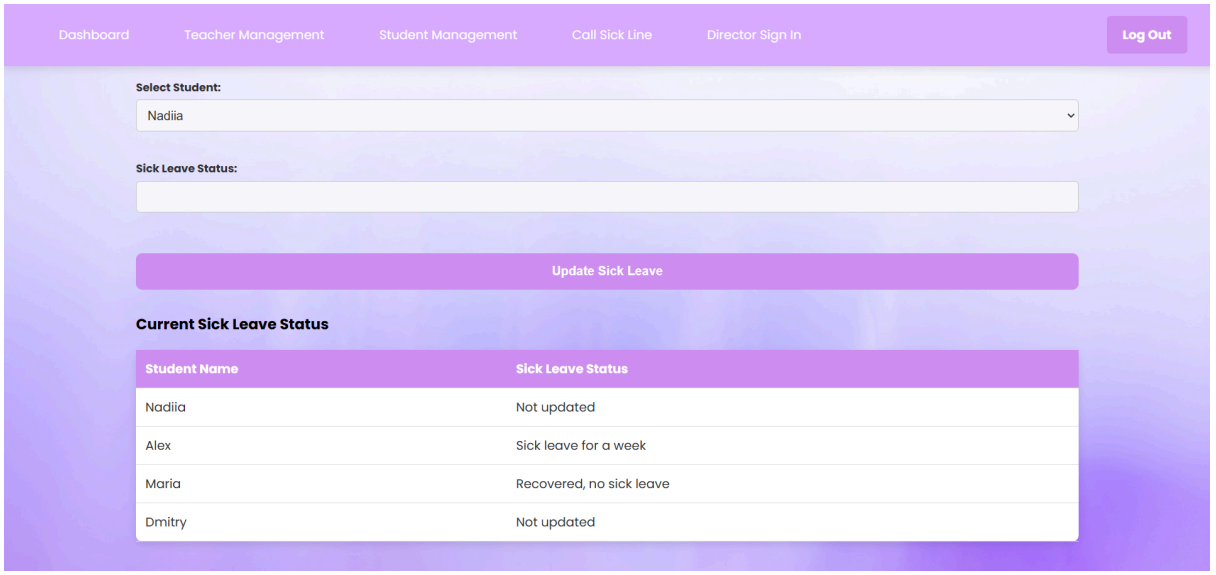

Рисунок 4.33 – Сторінка з інформацією про захворювання

Джерело: розроблено автором.

Завдяки цим інструментам адміністратор має змогу ефективно керувати навчальним процесом, взаємодіяти з викладачами та студентами, а також слідкувати за розвитком школи STAYinTouch.

4.4 Тестування розробки

Тестування вебдодатку для онлайн школи STAYinTouch є важливим етапом для перевірки його працездатності, коректності функціонування та безпеки. Для цього було створено тестові сценарії, які охоплюють основні функціональні можливості системи, зокрема: авторизацію, перегляд розкладу занять, відправку повідомлень, керування оплатами та перегляд прогресу навчання.

Тестування виконувалося за допомогою бібліотеки pytest для Python, яка дозволяє автоматизувати процес перевірки роботи вебдодатку.

На рисунку 4.34 представлено тест на перевірку успішної авторизації користувача з коректними даними.

На рисунку 4.35 наведено тест на перевірку доступності сторінки розкладу занять після авторизації.

Рисунок 4.36 демонструє тест, який перевіряє функціональність відправки повідомлень від студента до адміністратора. У тесті імітується введення тексту повідомлення та його успішна доставка.

```python
@pytest.fixture
def client():
    # Create the app with test-specific configurations
    app = create_app(testing=True)
    app.config['TESTING'] = True
    app.config['WTF_CSRF_ENABLED'] = False

    with app.test_client() as client:
        with app.app_context():
            db.create_all()

            # Add a test user to the database
            user = User(
                name='Test User',
                email='testuser@example.com'
            )
            user.set_password('securepassword')
            db.session.add(user)
            db.session.commit()

        yield client
```

Рисунок 4.34 – Тест на успішну авторизацію користувача

Джерело: розроблено автором.

```python
def test_schedule_page_access_after_login(client):
    response = client.post('/login')
    assert response.status_code == 200


    response = client.get('/schedule')
    assert response.status_code == 200
    assert b"Schedule Page" in response.data
```

Рисунок 4.35 – Тест на перегляд розкладу занять

Джерело: розроблено автором.

```python
def test_send_message(client):
    test_message = {"message": "Привіт, адміністраторе!"}
    response = client.post('/send_message', json=test_message)

    assert response.status_code == 200

    response_data = response.get_json()
    assert response_data["status"] == "Message sent"
    assert response_data["message"] == test_message["message"]
```

Рисунок 4.36 – Тест на відправку повідомлень

Джерело: розроблено автором.

На рисунку 4.37 показано тест, який перевіряє роботу функції управління оплатами, зокрема перевірку статусу платежу та коректність відображення суми заборгованості.

```python
def test_payment_status_and_outstanding_amount(client):
    # Test payment status for a valid user
    response = client.get('/payment_status/1')
    assert response.status_code == 200
    data = response.get_json()
    assert data["status"] == "paid"
    assert data["outstanding"] == 0

    # Test payment status for a user with an outstanding amount
    response = client.get('/payment_status/2')
    assert response.status_code == 200
    data = response.get_json()
    assert data["status"] == "pending"
    assert data["outstanding"] == 200

    # Update payment status and verify changes
    update_data = {"user_id": 2, "status": "paid", "outstanding": 0}
    response = client.post('/update_payment', json=update_data)
    assert response.status_code == 200
    data = response.get_json()
    assert data["status"] == "Payment updated"
    assert data["data"]["status"] == "paid"
    assert data["data"]["outstanding"] == 0
```

Рисунок 4.37 – Тест на управління оплатами

Джерело: розроблено автором.

На рисунку 4.38 зображено результат успішного проходження всіх тестів. Це підтверджує коректну роботу основних функцій системи.

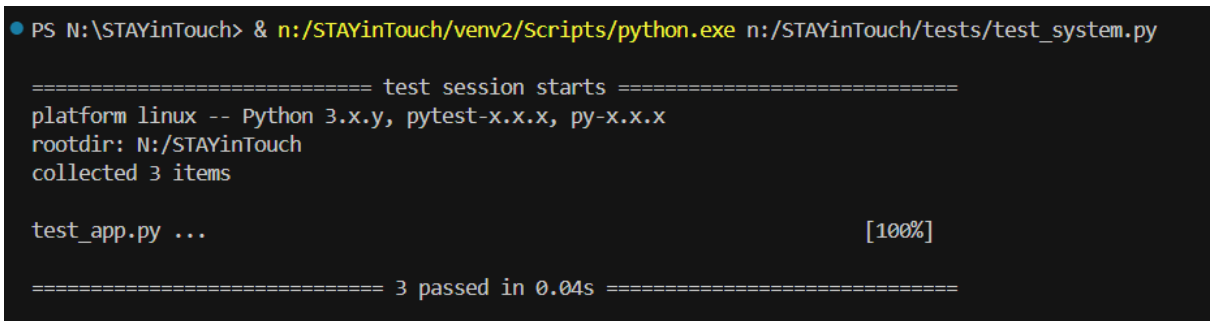

Рисунок 4.38 – Результати успішного тестування

Джерело: розроблено автором.

Тестування підтвердило надійність вебдодатку STAYinTouch у виконанні ключових задач для організації навчального процесу.

ВИСНОВКИ

Веборієнтована інформаційна система STAYinTouch має на меті суттєво вдосконалити управління діяльністю онлайн-школи іноземних мов, зосереджуючи увагу на специфіці дистанційного навчання. Основною перевагою системи є оптимізація управління інформацією через централізовану базу даних, яка забезпечує надійний доступ до актуальних даних про студентів, викладачів і курси. Це дозволяє адміністраторам оперативно отримувати необхідну інформацію та контролювати успішність навчання.

Аналіз стану досліджень у сфері дистанційної освіти та інформаційних систем показав, що дистанційне навчання вимагає ефективних і зручних систем для управління навчальними процесами. Існуючі рішення мають обмеження, які можуть бути покращені через інтеграцію різних функціональних блоків в єдину платформу, що дозволяє краще відповідати на потреби сучасних онлайн-шкіл.

Функціональні вимоги визначили необхідні можливості для задоволення потреб користувачів, а описані нефункціональні вимоги гарантують, що платформа буде зручною, безпечною, продуктивною і масштабованою. Це дозволило забезпечити стабільну роботу системи навіть при зростанні кількості користувачів.

Архітектура системи STAYinTouch була розроблена з урахуванням всіх етапів навчального процесу, що забезпечує цілісність і ефективність роботи онлайн-школи. Визначення основних компонентів, таких як модулі для управління студентами, викладачами, курсами та фінансами, дозволяє системі бути зручною для використання всіма учасниками навчального процесу.

Моделювання системи включало створення діаграм варіантів використання, які чітко розмежовують ролі користувачів та їх взаємодію з основними функціями платформи. Це дозволяє краще розуміти, як система буде працювати на практиці, і які операції користувачі можуть здійснювати.

Структура бази даних була розроблена для ефективного зберігання та обробки інформації про студентів, викладачів, курси та фінансові транзакції. Зв'язки між

таблицями дозволяють зручно відслідковувати успішність студентів, оплату курсів і розклад занять.

Система забезпечує централізований облік студентів, викладачів і курсів. Завдяки інтеграції даних всі учасники навчального процесу матимуть доступ до актуальної інформації в режимі реального часу. Це дозволило автоматизувати адміністрування, уникнути дублювання даних і значно скоротити час на виконання рутинних завдань. Функціонал обліку оплат і автоматичних нагадувань дозволяє уникнути затримок платежів та забезпечує фінансову прозорість. Адміністратори мають змогу відстежувати статуси оплат у реальному часі, а також генерувати звіти для аналізу фінансової діяльності. Для студентів передбачена функція отримання автоматичних нагадувань про наближення термінів оплати, що зменшить кількість прострочених платежів. Це сприяє стабільному фінансовому стану школи.

Система включає функції відстеження академічних показників студентів, що дозволило оцінювати прогрес навчання та ідентифікувати слабкі місця в їх підготовці. Викладачі отримали інструменти для аналізу результатів кожного студента, а також можливість адаптувати навчальний процес до індивідуальних потреб учнів. Для студентів і їхніх батьків передбачено доступ до персоналізованих звітів про досягнення.

Врахування потенційних ризиків реалізації проєкту є ще однією вагомою перевагою STAYinTouch. Система включає механізми для моніторингу ризиків і їхнього управління, що дозволяє вчасно реагувати на можливі перешкоди та забезпечувати стабільність і надійність роботи платформи.

Таким чином, система STAYinTouch, з урахуванням усіх вимог і архітектурних рішень, стане потужним інструментом для управління діяльністю онлайн-школи, здатним інтегрувати різні функціональні елементи в єдину платформу, підвищуючи ефективність навчання та взаємодії між учасниками освітнього процесу. В майбутньому система може бути розширена для забезпечення ще більшої зручності та персоналізації процесу навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adams, L. Comparative Analysis of Moodle and Blackboard in Modern E-Learning // Educational Software Review. 2020.

2. Andrews, T. Design Thinking in E-Learning Development // Journal of Digital Learning. 2022. № 8. С. 34–40.

3. Bailey, J. Artificial Intelligence in Personalized Education. London : TechLearn Press, 2021. 248 с.

4. Booch, G., Rumbaugh, J., Jacobson, I. The Unified Modeling Language User Guide. Boston : Addison-Wesley, 2005. 512 с.

5. Brown, D. Розробка веб-додатків на Python : посібник для початківців. Львів : Видавництво ЛНУ, 2020. 234 с.

6. Brown, T. Modern Web Development Frameworks and E-Learning. Cambridge : EduTech Press, 2020. 185 с.

7. Гудзь, В. В. Сучасні тенденції у дистанційній освіті // Вісник Національного університету «Львівська політехніка». 2021. № 3. С. 45–51.

8. Davis, M. Integrating Gamification into Online Learning Platforms // Journal of Educational Innovations. 2022. № 5. С. 57–66.

9. Dennis, A., Wixom, B. H., Roth, R. M. Systems Analysis and Design. John Wiley & Sons, 2012. 595 с.

10. Green, P. Cloud-Based Solutions for Distance Education. San Francisco : CloudTech, 2020. 198 с.

11. Harrison, R. Mobile Applications for E-Learning // Mobile Learning Review. 2021. № 9. С. 22–29.

12. Hoffer, J. A., Ramesh, V., Topi, H. Modern Database Management. Pearson, 2013. 816 с.

13. Johnson, P. E-Learning Systems and User Experience Design // Journal of Educational Technology. 2021. № 2. С. 89–97.

14. Котляр, В. П. Інформаційні системи в управлінні : теорія та практика. Харків : ХНУ, 2017. 365 с.

15. Lee, K. Application of Django and Flask in Developing E-Learning Platforms // International Journal of Web Applications. 2022. № 5. С. 23–30.

16. Martinez, R. Personalized Learning Environments in Language Education // Language and Learning Journal. 2022. № 7. С. 134–140.

17. Novak, V. Стратегії впровадження LMS у навчальні заклади // Освітні інновації. 2020. № 11. С. 45–52.

18. Parker, J. Edmodo and the Evolution of Social Learning Platforms // Journal of Online Education. 2021. № 6. С. 48–56.

19. Patel, M. Automation in E-Learning Systems: Financial and Administrative Tasks // Online Education Review. 2021. № 4. С. 77–82.

20. Петрова, І. М. Ефективність використання інформаційних технологій у навчанні іноземних мов // Педагогіка та психологія. 2020. № 9. С. 27–33.

21. Peterson, G. Data Security in Online Education Platforms. Cambridge : DataEdu, 2021. 312 с.

22. Романенко, С. М. Цифрова трансформація вищої освіти : підходи та перспективи. Київ : Наукова думка, 2021. 256 с.

23. Simmons, K. User-Centered Design in Digital Learning Environments. New York : UXEdu Press, 2020. 178 с.

24. Smith, J. E-Learning Platforms and Student Management Systems: Essentials for Modern Education. Boston : Education Press, 2020. 320 с.

25. Thompson, S. E-Learning and the Shift to Online Education. Academic Publishing, 2020. 278 с.

26. Wang, T. Improving Communication Features in Educational Systems // Interactive Learning Environments. 2022. № 4. С. 98–104.

27. Чалий, В. М. Електронне навчання в умовах дистанційної освіти. Харків : Видавничий Дім, 2022. 294 с.

28. Використання інформаційних систем в освіті // EduTech.

29. Платформи для дистанційного навчання : огляд та аналіз // Освіта України.

30. Сидоренко, О. М. Модернізація навчального процесу в умовах дистанційної освіти : дис. канд. пед. наук. Дніпро, 2019. 215 с.

Додаток А
Планування робіт

А.1. Ідентифікація мети ІТ-проекту.

Для чіткого визначення мети ІТ-проекту використовується метод SMART, що дозволяє сформулювати її відповідно до конкретних, вимірюваних, досяжних, релевантних та обмежених у часі критеріїв. Деталізацію мети представлено в Таблиці А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробка веборієнтованої інформаційної системи передбачає створення функціоналу для управління даними про студентів, викладачів, курси та платежі, що забезпечить підтримку організації навчального процесу онлайн-школи іноземної мови.
Measurable (вимірювана)	Система повинна бути протестована на 100% функціональності, включаючи управління даними про не менше 100 студентів і 10 викладачів.
Achievable (досяжна)	Реалізація проекту можливе за умови використання існуючих технологій (Flask, SQLAlchemy, тощо) та доступу до необхідних ресурсів (сервер, база даних).
Relevant (реалістична)	Проект відповідає потребам сучасного ринку онлайн-освіти, оскільки забезпечує доступ до якісного навчання та управління навчальним процесом.
Time-framed (обмежена у часі	Завершення проекту заплановано протягом 6 місяців, включаючи етапи планування, розробки, тестування та впровадження.

Джерело: розроблено автором.

А.2. Планування змісту структури робіт IT-проекту

Для організації структури робіт проєкту використовується ієрархічний підхід WBS (Work Breakdown Structure), що дозволяє деталізувати завдання на декілька рівнів. Структурований перелік робіт наведено в Таблиці А.2.

Таблиця А.2 − Таблиця WBS

Код	**Назва елемента WBS**	**Опис**
1.0	STAYinTouch Project	Загальний проект веборієнтованої інформаційної системи.
1.1	Планування	Етап, що включає аналіз вимог, розробку плану проекту та складання технічного завдання.
1.2	Розробка	Створення бази даних, фронтенду та бекенду системи.
1.3	Тестування	Проведення функціонального та навантажувального тестування системи.

Продовження табл. А.2

1.4	Впровадження	Впровадження системи, навчання користувачів та налаштування серверів.
1.5	Підтримка	Забезпечення технічної підтримки та оновлень системи після впровадження.
1.2.1	Розробка бази даних	Створення та налаштування бази даних.
1.2.2	Розробка інтерфейсу	Створення користувацького інтерфейсу.
1.3.1	Функціональне тестування	Перевірка основних функцій системи.
1.3.2	Навантажувальне тестування	Оцінка продуктивності системи.
1.4.1	Документація	Створення документації для користувачів.

Джерело: розроблено автором.

Візуалізація ієрархії основних компонентів проєкту виконана у вигляді структурної діаграми. Її представлено на Рисунку А.1.

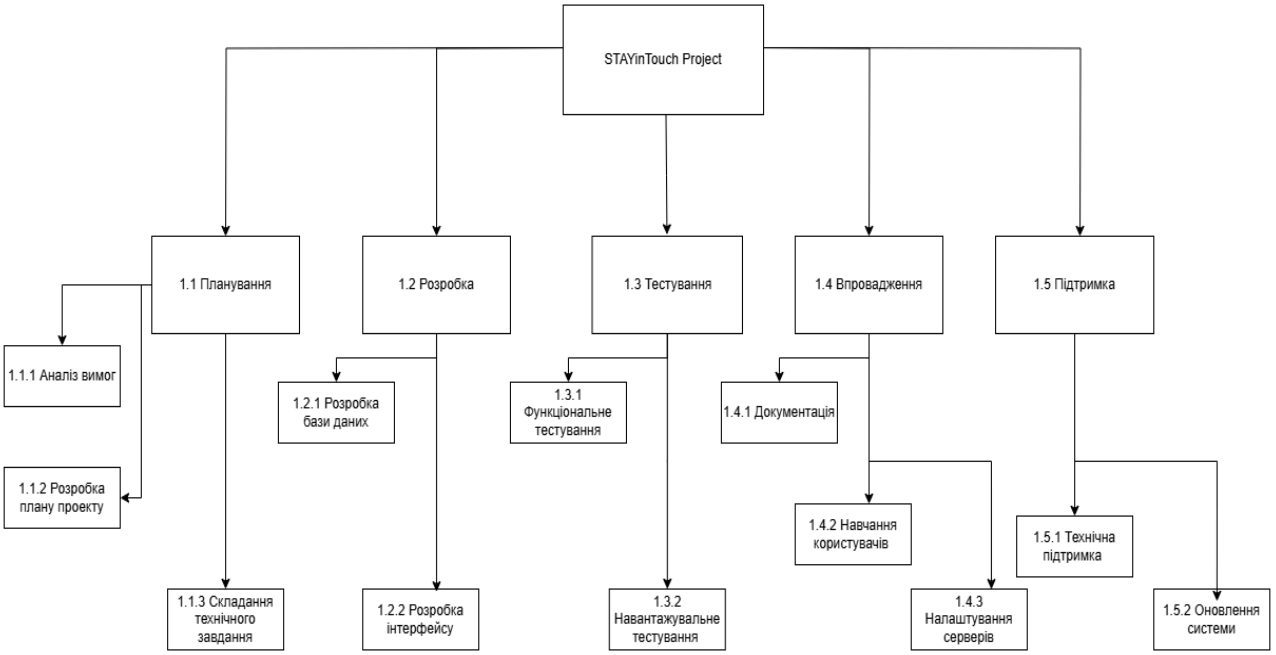

Рисунок А.1 - Структурна Діаграма проєкту STAYinTouch.

Джерело: розроблено автором.

Також для аналізу організаційної структури проєкту використовується OBS (Organization Breakdown Structure). Вона містить інформацію про розподіл ролей і відповідальності членів команди. Структуру подано в Таблиці А.3.

Таблиця А.3 − Таблиця OBS (Organization Breakdown Structure)

Рівень	**Код**	**Назва елемента**	**Опис**	**Відповідальні**
1	1.0	STAYinTouch	Веборієнтована інформаційна система	Сєрова Н.Г.
2	1.1	Планування	Загальне планування та складання завдань	Сєрова Н.Г.

Продовження табл. А.3

2	1.2	Розробка	Розробка веб-системи	Сєрова Н.Г.
3	1.2.1	Розробка бази даних	Створення та налаштування бази даних	Сєрова Н.Г.
3	1.2.2	Розробка інтерфейсу	Створення користувацького інтерфейсу	Сєрова Н.Г.
2	1.3	Тестування	Перевірка функціональності та якості	Сєрова Н.Г.
3	1.3.1	Функціональне тестування	Перевірка основних функцій системи	Сєрова Н.Г.
3	1.3.2	Навантажувальне тестування	Оцінка продуктивності системи	Сєрова Н.Г.
2	1.4	Впровадження	Впровадження системи в експлуатацію	Сєрова Н.Г.
3	1.4.1	Документація	Створення документації для користувачів	Сєрова Н.Г.
2	1.5	Підтримка	Технічна підтримка користувачів	Сєрова Н.Г.

Джерело: розроблено автором.

А.3. Побудова календарного графіку виконання ІТ - проекту

Для управління строками виконання робіт створено календарний графік, який дозволяє відстежувати послідовність завдань та терміни їх реалізації. Графік наведено у Рисунку А.2.

Рисунок А.2 – Діаграма Ганта, яка ілюструє етапи виконання IT-проекту, включаючи планування та терміни виконання основних завдань.

Джерело: розроблено автором.

А.4. Планування ризиків проекту

Для оцінки ризиків використовується матриця «Ймовірність – Втрати», що дозволяє визначити потенційні загрози проєкту та їхній вплив. Візуальне представлення ризиків наведено на Рисунку А.3.

Ризики:

Зміна цілей у ході реалізації проекту (Ймовірність: 4, Втрати: 4)

Збільшення навантаження під час реалізації проекту (Ймовірність: 3, Втрати: 3)

Зростання вимог до проекту (Ймовірність: 3, Втрати: 3)

Відсутність досвіду (Ймовірність: 3, Втрати: 2)

Людський фактор (Ймовірність: 2, Втрати: 3)

Відмова обладнання (Ймовірність: 2, Втрати: 4)

Відсутність кваліфікованого програміста (Ймовірність: 2, Втрати: 5)

Зміна строків виконання роботи (Ймовірність: 2, Втрати: 4)

Складнощі при впровадженні на місці (Ймовірність: 2, Втрати: 3)

Розрахунок загальних втрат

Загальні втрати можна обчислити шляхом множення ймовірності на величину втрат для кожного ризику і підсумовування:

Для ризику 1: 4×4=16

Для ризику 2: 3×3=9

Для ризику 3: 3×3=9

Для ризику 4: 3×2=6

Для ризику 5: 2×3=6

Для ризику 6: 2×4=8

Для ризику 7: 2×5=10

Для ризику 8: 2×4=8

Для ризику 9: 2×3=6

5					
4			12	16	20
3			6	9	12
2	2	4	6	8	10
1	1	2	3	4	5
	1	**2**	**3**	**4**	**5**

Рисунок А.3 – Матриця «Ймовірність – Втрати»

Джерело: розроблено автором.