НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри інформаційних систем і технологій

_____ проф., к.е.н., Швиденко М.З.

“30” травня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

Розробка типового веб-представництва студентської групи

Спеціальність 126 – «Інформаційні системи та технології»

Гарант освітньої програми
_____к.е.н., доцент_____ _____ _ Мокрієв М.В

Керівник бакалаврської кваліфікаційної роботи
_____к.е.н._____ _____ Саяпін С.П.

Виконав _____ Березовський Кирило Вадимович_____

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформаційних систем і технологій
_____ Швиденко М.З

«08» 01 2025р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Березовському Кирилу Вадимовичу

Спеціальність 126 – «Інформаційні системи та технології»

1. Тема бакалаврської кваліфікаційної роботи: **«Розробка типового веб-представництва студентської групи»**

затверджена наказом ректора НУБіП від 16.12.2024 р. № 2245-С

2. Термін подання завершеної роботи на кафедру 01.06.2025 р.

3. Вихідні дані до бакалаврської кваліфікаційної роботи: Веб-платформа студентської групи.

4. Перелік питань, що розглядаються:

1. Огляд та аналіз предметної області.
2. Інформаційне забезпечення
3. Прикладне програмне забезпечення
4. Тестування та вимоги системи

5. Висновки

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Визначення теми, постановка завдання, збір теоретичного матеріалу.	20.02-05.03	Визначення мети, завдань
2	Аналіз існуючої архітектури ІС	06.03-15.03	1 розділ
3	Моделювання вдосконаленої системи обліку, створення архітектури і логіки взаємодії	16.03-05.04	2 розділ
4	Розробка прототипу	06.04-30.04	3 розділ
5	Підготовка остаточного тексту дипломної роботи	01.05-15.05	Завершення, перевірка

Керівник дипломного проекту _____ ____Саяпін С.П.___

Завдання прийняв до виконання _____ ____Березовський К.В.___

Дата видачі завдання “08“ січня 2025 р.

РЕФЕРАТ

Бакалаврська кваліфікаційна робота
Березовського Кирила Вадимовича
на тему: «Розробка типового веб-представництва студентської групи»
Обсяг роботи: 58 сторінок, 26 рисунків, 1 додаток, 26 джерел
Графічна частина: 14 слайдів презентації
Керівник: Саяпін Сергій Петрович

У даній бакалаврській кваліфікаційній роботі розглянуто процес розробки типового веб-представництва для студентської академічної групи з метою централізованої організації навчальної, інформаційної та комунікаційної діяльності. Проведено аналіз сучасних інформаційних систем, таких як Moodle, Microsoft Teams, Google Classroom, та виявлено їх обмеження у контексті внутрішньогрупового інформаційного середовища.

Основну увагу приділено проектуванню архітектури багаторівневої веб-системи, що включає рівень представлення, рівень бізнес-логіки та рівень доступу до даних. Реалізовано низку функціональних модулів: управління розкладом, публікація оголошень, навчальні матеріали, облік складу групи, зворотний зв’язок, а також аналітичний і рекомендаційний модулі. Всі модулі забезпечують адаптивність, зручність користування та безпечний обмін інформацією.

Система реалізована із застосуванням технологій Node.js, Express.js, JavaScript, HTML/CSS, бази даних SQLite та ORM Sequelize. В роботі побудовано UML-діаграми (прецедентів, активності, послідовності), IDEF0-діаграму функціональної декомпозиції, ER-діаграму бази даних, BPMN-модель бізнес-процесів, а також описано інтерфейсну структуру взаємодії користувачів

Результатом роботи стала адаптивна веб-система, придатна до впровадження в будь-який навчальний заклад як універсальний шаблон для інформаційного супроводу діяльності студентської групи.

Ключові слова: веб-представництво, студентська група, інформаційна система, Node.js, розклад, навчальні матеріали, оголошення, зворотний зв’язок, ER-діаграма, авторизація.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

2 API — інтерфейс прикладного програмування (Application Programming Interface)

3 JSON — текстовий формат обміну структурованими даними (JavaScript Object Notation)

4 UI — інтерфейс користувача (User Interface)

5 UX — досвід користувача (User Experience)

6 JVM — віртуальна машина Java (Java Virtual Machine)

7 Swing — набір компонентів графічного інтерфейсу користувача в Java

8 HTML Report — звіт у форматі HTML, що формується системою

9 Deployment Diagram — UML-діаграма розгортання програмного забезпечення

10 ER-діаграма — діаграма зв’язків сутностей (Entity-Relationship Diagram)

11 IDEF0 — метод функціонального моделювання для опису процесів системи

12 IDEF5 — метод побудови онтологій у складних інформаційних системах

13 BPMN — нотація моделювання бізнес-процесів (Business Process Model and Notation)

ВСТУП

Особливої актуальності набуває створення ефективних веб-представництв для різних організаційних структур, зокрема й освітніх установ. Зростаюча кількість студентських груп, які прагнуть підтримувати власну присутність у цифровому середовищі, обумовлює необхідність розроблення спеціалізованих веб-платформ, що забезпечують інтеграцію навчальної, інформаційної, комунікаційної та адміністративної діяльності. Створення типового веб-представництва студентської групи дозволяє оптимізувати процес обміну інформацією між студентами, викладачами та адміністрацією, сприяє формуванню корпоративної культури навчального колективу, а також забезпечує прозорість і зручність доступу до навчальних матеріалів, розкладів, новин, оголошень та інших важливих ресурсів.

Сучасні веб-технології, включаючи HTML5, CSS3, JavaScript та серверні платформи на основі Node.js, створюють технічні передумови для формування гнучких, масштабованих та безпечних інформаційних систем для підтримки освітніх процесів. Розроблення типових рішень дозволяє не лише стандартизувати підхід до створення подібних веб-застосунків, а й забезпечити можливість швидкого розгортання систем у різних освітніх закладах із урахуванням локальних вимог і специфіки контингенту користувачів. Крім того, врахування актуальних стандартів безпеки, визначених зокрема у документах OWASP Top Ten та ISO/IEC 27001:2013, дозволяє забезпечити необхідний рівень захисту персональних даних та стабільності роботи системи [[11].

Метою даної роботи є розробка типового веб-представництва студентської групи, яке дозволяє централізовано зберігати, обробляти та надавати доступ до основної інформації навчального процесу із забезпеченням належної функціональності, зручності користування та інформаційної безпеки.

Для досягнення поставленої мети у роботі необхідно вирішити такі **завдання:**

1. Здійснити аналіз предметної області та наявних рішень у сфері веб-застосунків для підтримки освітнього процесу.

2. Сформувати технічне завдання та визначити функціональні та нефункціональні вимоги до системи.

3. Розробити архітектуру інформаційної системи та спроєктувати структуру бази даних.

4. Реалізувати основні функціональні модулі веб-представництва з використанням сучасних веб-технологій.

5. Провести тестування розробленої системи та оцінити її ефективність.

Об’єктом дослідження є процес організації інформаційного забезпечення діяльності студентських груп засобами веб-технологій. Предметом дослідження виступають методи, моделі та інструментальні засоби проектування, реалізації та впровадження веб-представництва студентської групи.

Практична цінність роботи полягає у створенні універсальної програмної системи, що може бути адаптована та впроваджена у різних освітніх закладах для забезпечення інтегрованої підтримки освітніх процесів, спрощення адміністративних процедур, оптимізації інформаційного обміну, покращення комунікації між студентами та викладачами, а також підвищення прозорості управлінських процесів.

Структурно робота складається з трьох розділів. У першому розділі проведено аналіз предметної області, визначено основні вимоги та сформовано технічне завдання. Другий розділ присвячено розробленню архітектури системи, моделюванню її функціональної структури та бази даних. У третьому розділі наведено процес реалізації функціональних компонентів системи, інтеграції з базою даних, проведення тестування та аналізу результатів роботи системи.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Організація інформаційної підтримки освітнього процесу у сучасних закладах вищої освіти потребує розробки інтегрованих рішень, здатних забезпечити централізоване зберігання, обробку, актуалізацію та своєчасну доставку даних всім учасникам навчального процесу. У межах такого підходу створення типового веб-представництва студентської групи виступає як окремий напрям розробки, що забезпечує акумулювання функціональних компонентів управління навчальними планами, розкладом занять, навчально-методичними матеріалами, комунікаційними потоками та результатами поточного контролю знань студентів. Реалізація такого рішення дозволяє формалізувати інформаційні потоки академічної групи, забезпечуючи їх достовірність, повноту та доступність у реальному часі [1].

Формування функціональної структури типового веб-представництва здійснюється з урахуванням специфіки організаційної моделі освітнього закладу, нормативних вимог до супроводу освітнього процесу, обсягу навчальної інформації та потреби в оперативному обміні даними між студентами, викладачами та адміністрацією. Веб-представництво трансформується із статичної інформаційної сторінки у динамічну інформаційну систему, що підтримує багатосторонню взаємодію, регулярне оновлення інформації та реалізацію низки сервісних функцій [7].

З урахуванням функціональних вимог до розроблюваної системи структура модулів типового веб-представництва студентської групи представлена у таблиці 1.1.

Таблиця 1.1 – Структурна модель функціональних модулів веб-представництва студентської групи

Функціональний модуль	Функціональне призначення
Розклад навчальних занять	Відображення актуального розкладу з деталізацією за дисциплінами, викладачами, аудиторіями та календарними періодами
Система адміністративних оголошень	Публікація офіційних повідомлень про зміни в навчальному процесі, важливі організаційні події та повідомлення деканату
Блок навчально-методичних матеріалів	Централізоване зберігання лекційних матеріалів, презентацій, інструктивних посібників, нормативних документів курсу
Облік відвідуваності та навчальної успішності	Формування та відображення даних щодо поточної успішності, підсумкових оцінок та контролю відвідування занять
Облік складу студентської групи	Відображення та адміністрування актуального спискового складу групи із зазначенням персональних даних студентів, статусів, контактів
Контактна система	Організація каналів комунікації зі старостою, кураторами груп, викладацьким складом, методичними службами
Зворотній зв’язок та запити студентів	Реалізація механізмів подання звернень, запитань, пропозицій та коментарів до адміністрації групи або факультету

Враховуючи характер використання системи, пріоритетність реалізації функціональних модулів визначається частотою їхнього використання. Найбільш актуальними для щоденної роботи користувачів є модулі розкладу занять, обліку успішності та доступу до навчальних матеріалів, оскільки вони забезпечують оперативний супровід освітнього процесу. Модулі зворотного зв’язку, адміністративних оголошень та контактної системи підтримують організаційні процеси групи й активуються за потреби. Облік складу студентської групи використовується при адмініструванні персональних даних, формуванні списків та звітів, що обумовлює його періодичну, але критично важливу функцію в системі.

Інтегрована реалізація всіх зазначених модулів у межах єдиного веб-представництва дозволяє усунути дублювання даних, підвищити ефективність

управління інформацією та забезпечити високу якість організації освітнього процесу [4].

1.2 Аналіз наявних рішень

У сфері організації освітнього процесу на сьогодні існує ціла низка широко використовуваних інформаційних систем, які забезпечують функціональність електронного супроводу навчання, адміністрування курсів, формування освітніх програм та інтеграцію комунікаційних засобів між усіма учасниками освітнього середовища. Серед таких рішень особливого поширення набули комплексні LMS-платформи: Google Classroom, Microsoft Teams for Education, Moodle, Canvas LMS та Blackboard Learn. Вони ефективно використовуються в межах академічних курсів, модулів, кафедральних та міжкафедральних навчальних блоків, однак значною мірою орієнтовані саме на організацію навчального процесу як предметно-курсової діяльності, а не на створення веб-представництв студентських груп у вузькому, внутрішньогруповому інформаційному контексті.

Одним із найбільш поширених рішень є система Google Classroom, що інтегрована до екосистеми Google Workspace. Вона орієнтована на централізоване управління навчальними курсами з акцентом на спрощення взаємодії студентів та викладачів, зручну організацію завдань, спільний доступ до файлів і календарів. Простота використання, мінімальна складність первинного налаштування та автоматична інтеграція з поштовим сервісом Gmail, Google Drive та Google Calendar дозволяє швидко розгортати освітній процес. Водночас Google Classroom фактично не орієнтована на представлення студентської групи як самостійного організаційного осередку: відсутня підтримка власних новинних стрічок групи, адміністративних блоків для старост, кураторів, локальної фото- та відеогалереї, публікації загальногрупових заходів тощо (рис. 1.1).

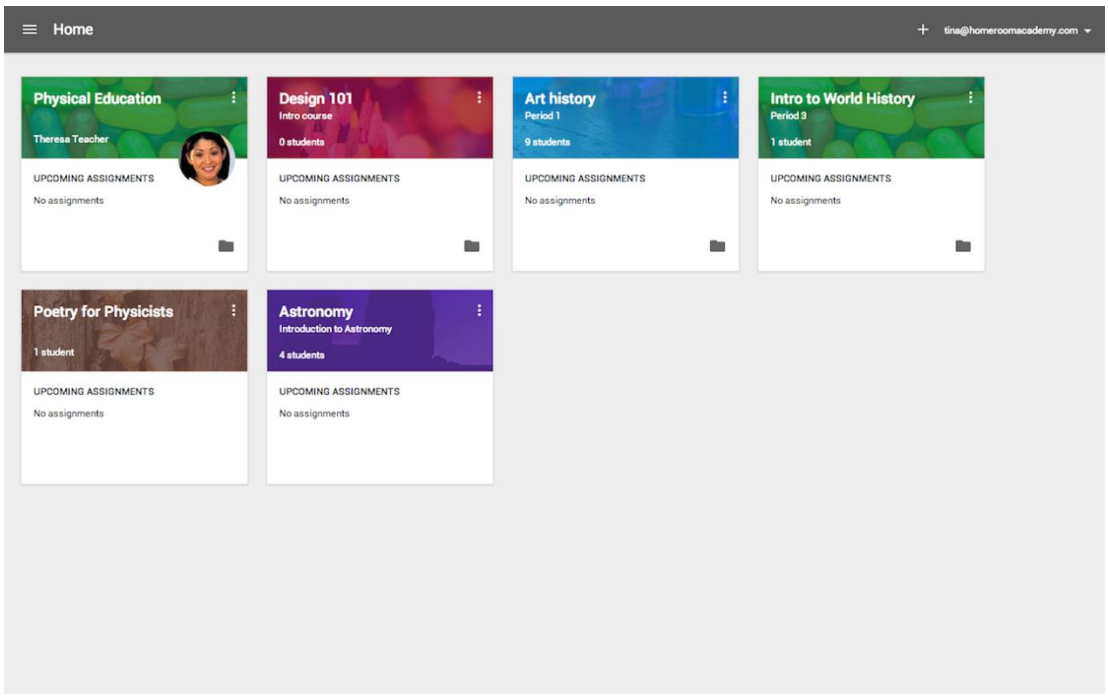


Рисунок 1.1 – Інтерфейс головної сторінки системи Google Classroom

Аналогічна ситуація спостерігається в платформі Microsoft Teams for Education, яка поєднує управління курсами, синхронну комунікацію (чат, відеозв'язок), колективну роботу з файлами та інтеграцію з Microsoft 365. Її ключова перевага полягає у гнучкому створенні команд за дисциплінами та групами, можливості адміністрування доступів, розподілу ролей викладачів, студентів і адміністративного персоналу. Однак при усіх цих перевагах функціональність Teams передбачає розгортання всередині корпоративної інфраструктури закладу освіти, що може бути надмірним для цілей окремого внутрішньогрупового веб-представництва (рис. 1.2).

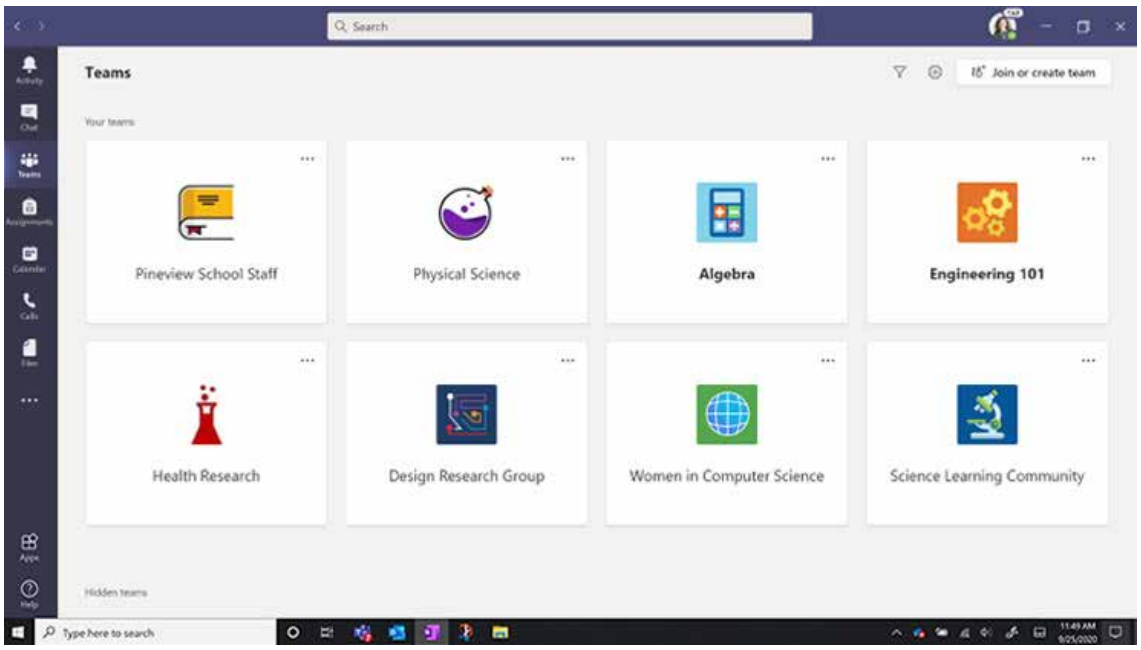


Рисунок 1.2 – Головна сторінка навчальних груп у системі Microsoft Teams for Education

Особливе місце займає відкрита система управління навчанням Moodle, що дозволяє будувати складні ієрархії навчальних курсів, реалізовувати багаторівневий контроль знань, формувати великі бази студентів, інтегрувати численні плагіни та забезпечувати самостійне розгортання на серверах навчальних закладів. Головна перевага Moodle — гнучкість налаштувань та багатофункціональність. Проте в контексті розробки компактної, простої у використанні платформи представлення студентської групи Moodle є надлишковою — система потребує високої кваліфікації адміністратора, складних технічних налаштувань та суттєвих обчислювальних ресурсів для підтримки (рис. 1.3).

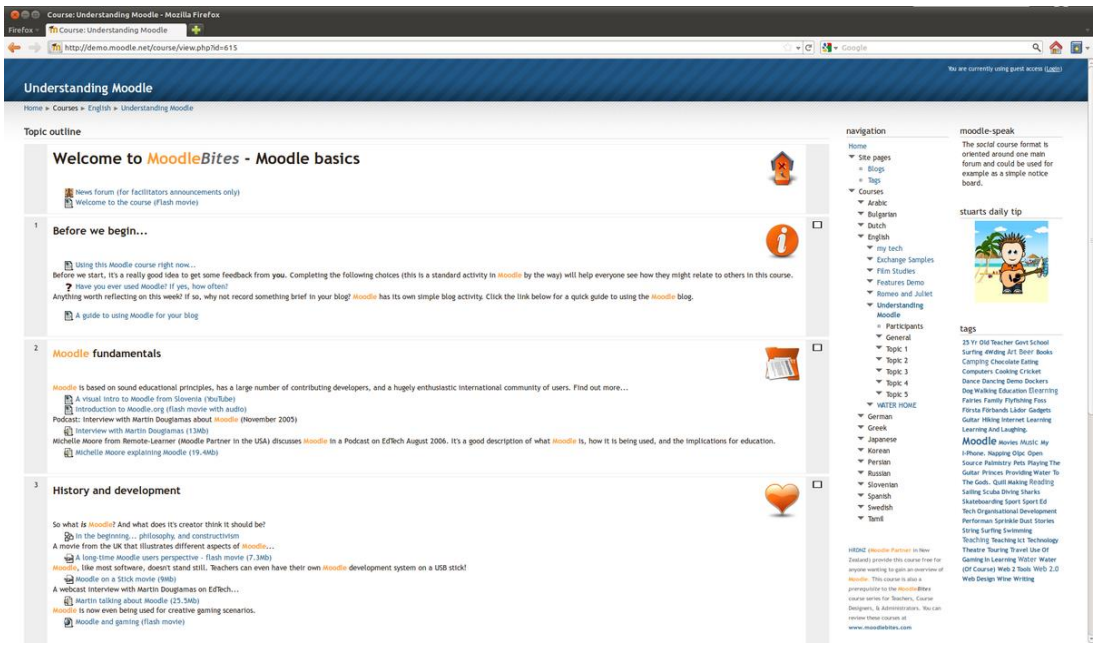


Рисунок 1.3 – Інтерфейс курсу в системі Moodle

Схожу концепцію реалізує Canvas LMS, що активно використовується в освітніх установах США, Канади та ЄС. Canvas пропонує адаптивний інтерфейс, зручне ведення курсів, журналів, завдань та оцінювання. Особливою перевагою є вбудовані аналітичні панелі моніторингу успішності студентів. Однак і тут, як і в попередніх випадках, відсутня підтримка повноцінного внутрішньогрупового інформаційного середовища (рис. 1.4).

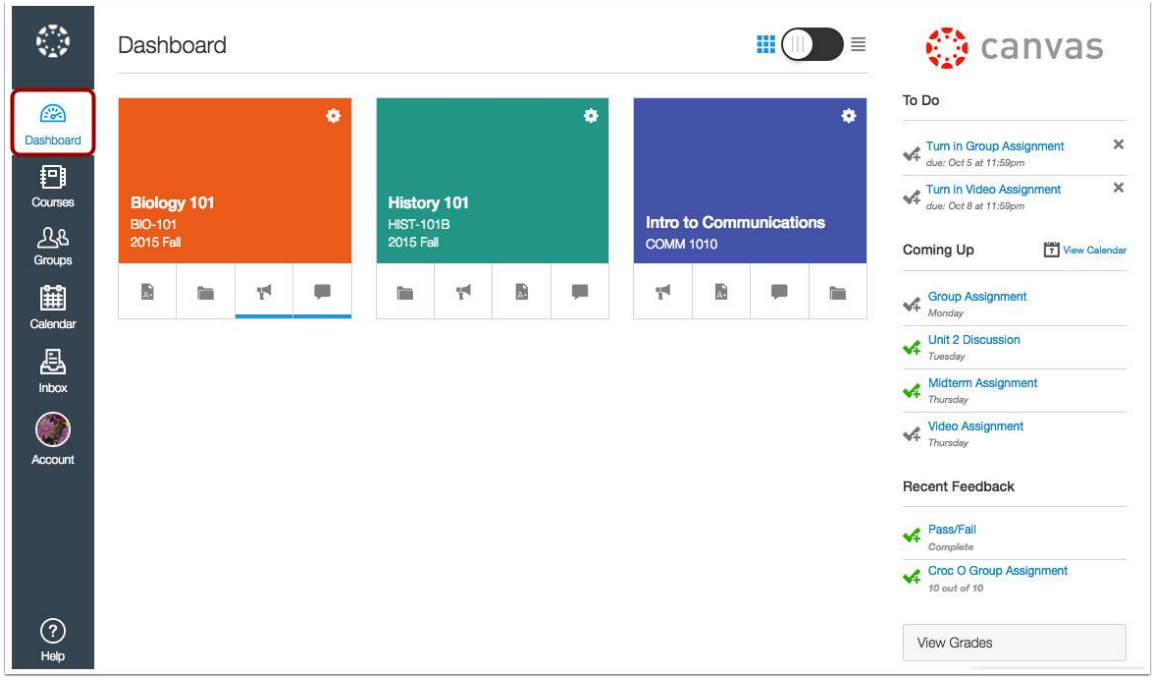


Рисунок 1.4 – Робочий простір навчальних курсів у Canvas LMS

Останнім у розгляді є рішення Blackboard Learn, яке вирізняється широкими можливостями адаптації освітнього середовища до вимог ВНЗ, підтримкою стандартів SCORM, аналітикою та масштабованістю. Система ефективно працює у великих корпоративних мережах, проте складність впровадження, потреба в спеціалізованому технічному обслуговуванні та орієнтація на міждисциплінарні курсові програми роблять її малопридатною для простого веб-представництва окремої академічної групи (рис. 1.5).

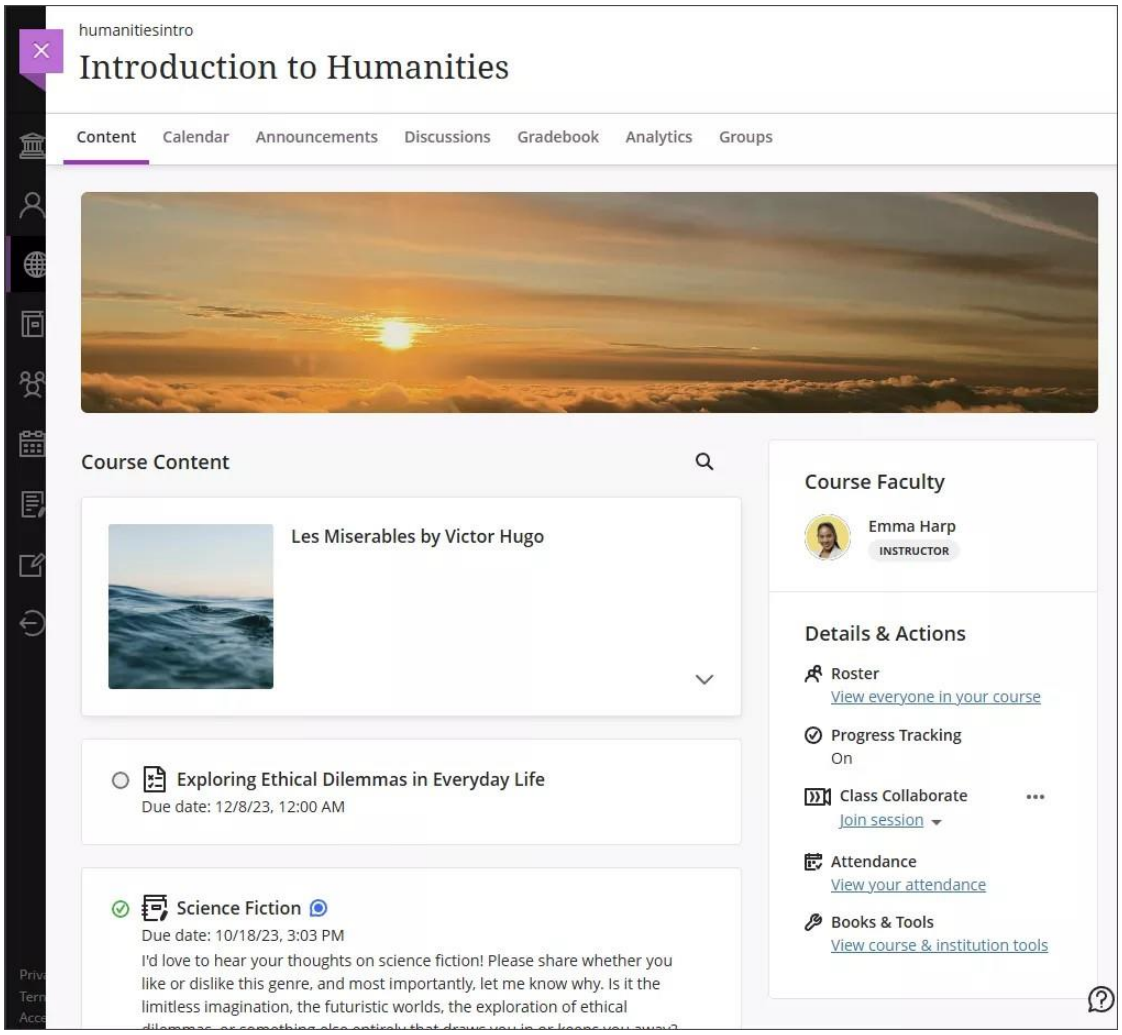


Рисунок 1.5 – Інтерфейс курсу в системі Blackboard Learn

Таким чином, всі розглянуті рішення функціонально орієнтовані на побудову курсів і розгортання навчальних модулів, але не охоплюють низку внутрішньогрупових сервісів, критично важливих у межах студентського самоврядування, організації внутрішніх новин, комунікації між студентами, старостами, кураторами, презентації групи, фотоархівів, локальних оголошень тощо.

Варто відзначити, що в окремих випадках у мережі вже існують незалежні реалізації веб-сайтів студентських груп, створених переважно ентузіастами самих студентів (WordPress-сайти, прості статичні сайти на HTML/CSS/JS, Google Sites тощо). Такі проєкти виконують базову функцію інформаційної вітрини групи — публікують склад групи, контакти, розклад занять, фотоальбоми, новини про участь у заходах. Проте більшість із них не мають глибокої інтеграції зі службовими освітніми системами університетів, не підтримують гнучкого адміністрування, не забезпечують синхронізовану комунікацію або можливості аналітики

1.3 Визначення вимог до інструментальних засобів та розробка технічного завдання

Після проведеного аналізу предметної області та існуючих рішень визначено повний перелік вимог до функціонування системи веб-представництва студентської групи. Вимоги класифіковано за п’ятьма основними категоріями: функціональні, нефункціональні, інтерфейсні, технічні та інформаційні.

Функціональні вимоги визначають перелік основних задач, які повинна виконувати система у процесі забезпечення освітньої діяльності групи. Основні положення функціональних вимог наведено у таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до системи

Вимога	Опис
Формування розкладу	Створення, редагування та перегляд розкладу занять для студентської групи
Публікація оголошень	Додавання адміністративних оголошень для студентів та викладачів
Завантаження навчальних матеріалів	Зберігання, організація та доступ до навчальних ресурсів групи
Обробка звернень	Реєстрація звернень, заявок, запитань студентів до куратора або старости

Нефункціональні вимоги визначають загальні характеристики системи, що забезпечують її ефективність, надійність та зручність використання, і наведені у таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до системи

Параметр	Характеристика
Продуктивність	Швидка обробка запитів користувачів без затримок
Надійність	Стабільна робота системи за стандартних та пікових навантажень
Безвідмовність	Мінімальний час простою та високий рівень готовності
Захищеність	Захист персональних даних студентів та адміністраторів

Інтерфейсні вимоги відображають особливості організації користувацької взаємодії з системою, які наведено у таблиці 1.5.

Таблиця 1.5 – Інтерфейсні вимоги до системи

Параметр	Характеристика
Структуризація	Розподіл інтерфейсу на логічні розділи за видами функціоналу
Інтуїтивність	Зрозумілий та передбачуваний процес роботи для кожного користувача
Доступність	Підтримка роботи як на десктопних, так і мобільних пристроях
Авторизація	Забезпечення ідентифікації та розмежування прав доступу
Локалізація	Підтримка роботи системи українською мовою

Технічні вимоги визначають умови розгортання, експлуатації та підтримки працездатності системи, що подано у таблиці 1.6.

Таблиця 1.6 – Технічні вимоги до системи

Параметр	Характеристика
Архітектура	Веб-орієнтована архітектура з централізованим сервером
Платформи	Підтримка сучасних браузерів та операційних систем
Безпека	Захист бази даних, шифрування передавання даних, контроль доступу
Резервне копіювання	Наявність регулярного резервного збереження баз даних
Обслуговування	Простота оновлення та супроводу програмного забезпечення

Інформаційні вимоги відображають склад і структуру даних, які необхідно зберігати в межах функціонування системи. Узагальнені інформаційні вимоги наведено у таблиці 1.7.

Таблиця 1.7 – Інформаційні вимоги до системи

Категорія даних	Опис
Студентські дані	ПІБ, номер групи, контактна інформація
Дані викладачів	ПІБ, дисципліна, контакти
Розклад занять	Назва дисципліни, дата, час, аудиторія, викладач
Навчальні матеріали	Перелік назв матеріалів для завантаження, файл, дата публікації
Оголошення	Заголовок, зміст оголошення, дата публікації

Визначені вимоги є основою для формування технічного завдання на розробку інформаційної системи веб-представництва студентської групи. Вони чітко фіксують обсяг функціональності, логіку організації даних, характер взаємодії користувачів із системою та умови її технічної експлуатації, що забезпечує подальшу можливість формалізованого проектування архітектури та вибору програмних рішень для реалізації системи.

1.4 Постановка технічного завдання, сценарії використання системи

З метою реалізації ефективного веб-представництва студентської групи необхідно сформулювати технічне завдання, яке відображає повний спектр вимог до функціональності, структури, інтерфейсних особливостей, безпеки та організації взаємодії користувачів із системою. Визначення такого завдання ґрунтується на результатах проведеного аналізу предметної області, порівняння з наявними рішеннями та виявленням обмежень існуючих платформ.

Технічне завдання передбачає розроблення інформаційної системи, яка реалізовуватиме наступні ключові функціональні блоки:

- модуль формування, редагування та перегляду розкладу занять;
- модуль публікації та оновлення адміністративних оголошень;
- модуль завантаження та доступу до навчально-методичних матеріалів;
- модуль зворотного зв’язку з можливістю подання звернень до старости, куратора або адміністрації групи;

- модуль обліку складу студентської групи та контактної інформації;
- інтерфейсна система з підтримкою ролей (студент, староста, куратор, адміністратор) з відповідними правами доступу;
- базовий механізм автентифікації користувачів.

Система має бути адаптивною до різних типів пристроїв (ПК, планшет, смартфон), забезпечувати багатомовну підтримку (мінімум українську) та функціонувати у середовищі сучасних браузерів без встановлення додаткового програмного забезпечення.

З урахуванням різних типів користувачів у системі, було визначено типові сценарії використання, які відображають характерні дії та потоки взаємодії у межах запропонованої архітектури. Ці сценарії є основою для побудови моделей випадків використання (use case) у наступному розділі.

Основні сценарії використання системи:

1. Сценарій 1. Перегляд розкладу занять
   - Користувач (студент або викладач) авторизується в системі.
   - Обирає вкладку «Розклад».
   - Переглядає список занять за поточним тижнем із зазначенням дисциплін, аудиторій та викладачів.
   - За потреби — завантажує розклад у PDF.

2. Сценарій 2. Додавання оголошення
   - Авторизований користувач із роллю адміністратора або старости переходить у розділ «Оголошення».
   - Натискає кнопку «Додати».
   - Вводить заголовок, текст повідомлення, обирає термін дії.
   - Публікує оголошення, яке стає доступним усім учасникам групи.

3. Сценарій 3. Завантаження навчальних матеріалів
   - Користувач з правом доступу (наприклад, куратор або староста) переходить до вкладки «Матеріали».

- Обирає категорію або тему.
- Завантажує файл (PDF, DOCX, презентацію тощо) та додає короткий опис.
- Інші користувачі отримують доступ до перегляду або завантаження матеріалів.

4. Сценарій 4. Надсилання звернення
   - Студент після авторизації переходить у розділ «Звернення».
   - Обирає тип звернення (запитання, скарга, пропозиція).
   - Вказує тему, текст повідомлення та отримувача (староста, куратор, адміністратор).
   - Надсилає запит, який реєструється в системі.

5. Сценарій 5. Перегляд списку групи
   - Користувач переходить у вкладку «Склад групи».
   - Отримує таблицю з ПІБ, контактною інформацією, роллю в групі.
   - Адміністратор може додавати або змінювати дані при необхідності.

Запропоновані сценарії дозволяють моделювати поведінку користувачів у системі, визначити точки взаємодії з інтерфейсом і сформувати вимоги до логіки роботи окремих модулів. Вони також є основою для побудови діаграм прецедентів (use case diagram), які будуть представлені у розділі 2 під час розробки архітектури програмного забезпечення.

Таким чином, постановка технічного завдання і формалізація сценаріїв взаємодії користувачів із системою дозволяє створити чітке уявлення про функціональне наповнення, логіку роботи та критерії реалізації веб-представництва студентської групи.

1.5 Висновок до першого розділу

У результаті проведеного дослідження предметної області було обґрунтовано доцільність створення спеціалізованої інформаційної системи для представлення та інформаційного супроводу діяльності студентської групи в цифровому середовищі. Аналіз наявних рішень у сфері управління освітніми процесами (Google Classroom, Microsoft Teams for Education, Moodle, Canvas LMS, Blackboard Learn) показав, що вони в основному орієнтовані на організацію дистанційного навчання, адміністрування курсів і управління освітнім контентом. Проте ці платформи не передбачають гнучкого налаштування під потреби окремої академічної групи, не забезпечують локального контролю за структурою групи, не підтримують публічне веб-представництво та інтегровану комунікаційну взаємодію.

Також було досліджено приклади окремих ініціативних реалізацій веб-сайтів студентських груп, створених на базі WordPress, Google Sites або з використанням HTML/JS. Хоча подібні проєкти дозволяють виконувати базові інформаційні функції, вони не забезпечують необхідного рівня масштабованості, безпеки, адміністрування та підтримки ролей користувачів, що обмежує їх придатність до типового впровадження в масштабах різних навчальних підрозділів.

На основі результатів аналізу сформовано структурну модель системи, яка охоплює ключові функціональні модулі: розклад занять, публікація оголошень, навчальні матеріали, зворотний зв’язок, контактна система, а також облік складу групи. Визначено набір вимог до системи, включаючи функціональні, нефункціональні, інтерфейсні, технічні та інформаційні аспекти. Уточнено, що система не дублює функції електронного деканату або LMS, а натомість слугує допоміжним інструментом внутрішньої організації та інформаційного самоврядування студентської групи.

Крім того, сформульовано технічне завдання та типові сценарії використання системи, що враховують реальні потреби учасників навчального

процесу. Визначені сценарії стали основою для подальшого моделювання архітектури та логіки роботи інформаційної системи.

Таким чином, перший розділ створює методологічне й технічне підґрунтя для проєктування архітектури веб-представництва студентської групи, що реалізуватиметься в наступних етапах дослідження.

2 ПРОЄКТУВАННЯ ДОРАДНИЦЬКОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Моделювання предметної області

У процесі проектування інформаційної системи веб-представництва студентської групи важливим етапом є побудова моделі предметної області, яка дозволяє формалізувати основні процеси, учасників системи та характер їх взаємодії. Моделювання дозволяє описати функціональну структуру майбутньої системи, інформаційні потоки та залежності між компонентами.

На першому етапі моделювання було сформовано діаграму прецедентів, яка відображає основні дії користувачів системи. Основними суб’єктами системи визначено студентів, викладачів та адміністратора групи. Студент взаємодіє із системою з метою перегляду оголошень, розкладу занять, успішності, навчальних матеріалів, а також для надсилання зворотного зв’язку. Адміністратор групи виконує функції адміністрування користувачів, а також здійснює управління навчальними матеріалами та оголошеннями. Структуру основних функціональних взаємодій відображено на рисунку 2.1.

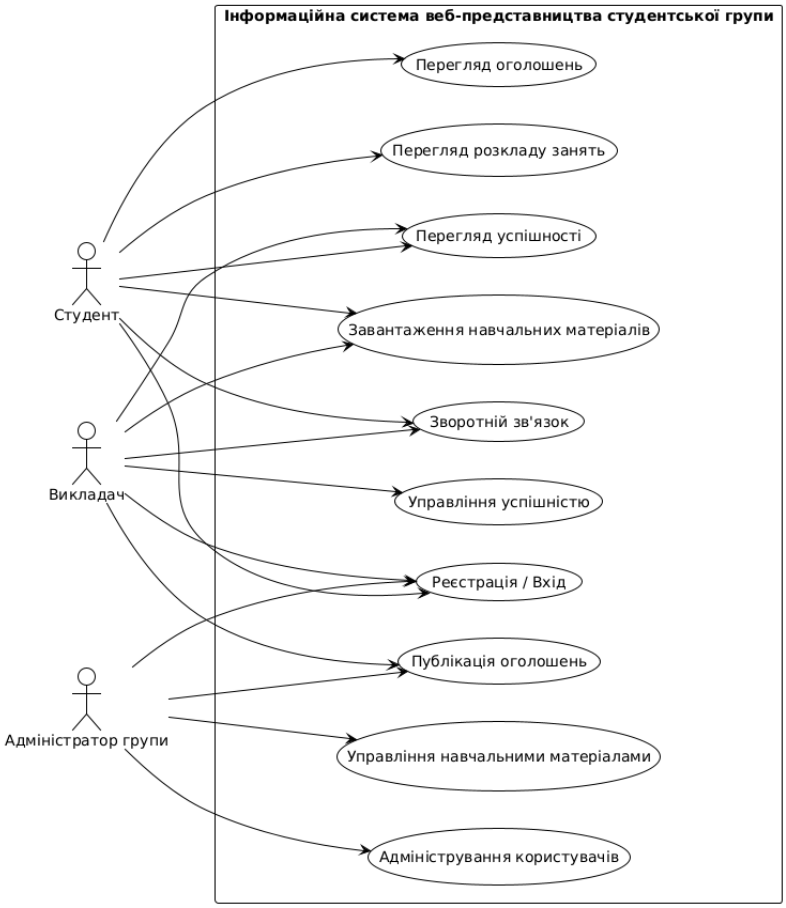


Рисунок 2.1 – Діаграма прецедентів для інформаційної системи веб-представництва студентської групи

Для деталізації логіки функціонування системи виконано побудову діаграми активності, яка демонструє послідовність дій користувача при роботі з системою. Діаграма структурована за двома основними відділами: користувач (студент або викладач) та система, яка додатково розподілена на рівень прикладної логіки та рівень бази даних. На початковому етапі користувач здійснює автентифікацію, після чого обирає функціональний модуль роботи: перегляд розкладу, доступ до навчальних матеріалів, зворотній зв'язок або отримання інформації про успішність. Кожен вибраний запит обробляється відповідною підсистемою, яка формує запит до бази даних, отримує результати та формує відповідь для користувача. Побудовану модель активності системи наведено на рисунку 2.2.

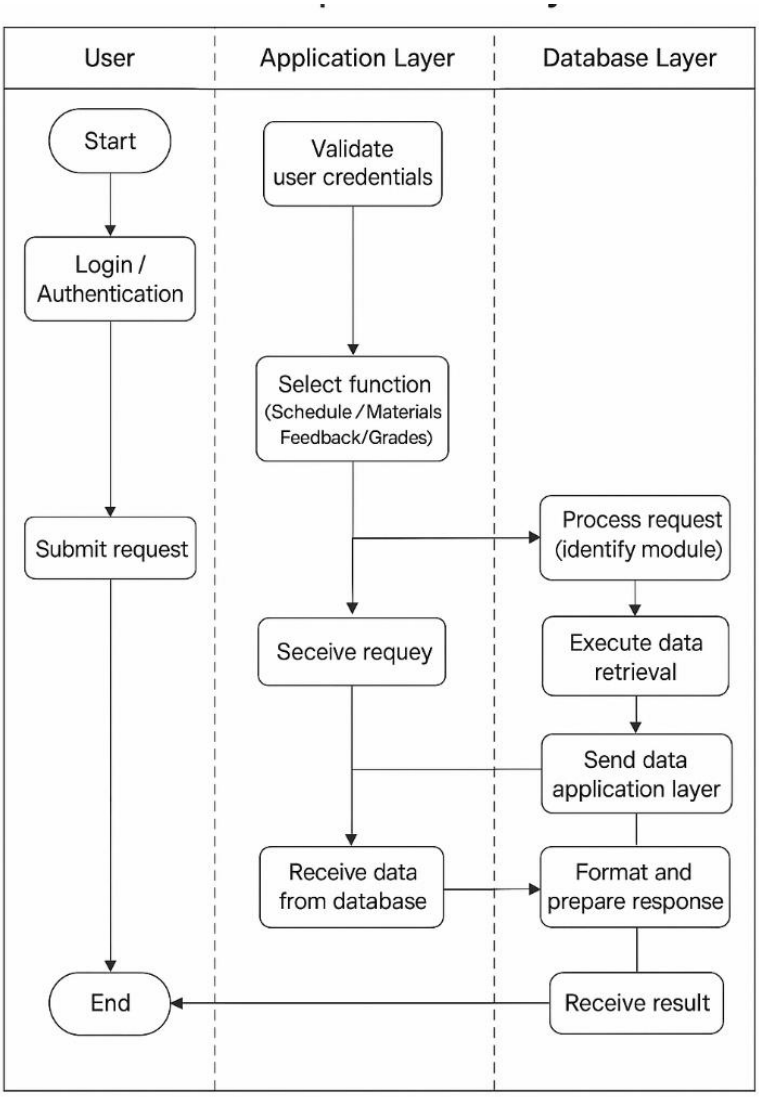


Рисунок 2.2 – Діаграма активності взаємодії користувача із системою

Для більш глибокої деталізації логіки обробки запитів в системі побудовано діаграму послідовності на прикладі операції перегляду розкладу занять. Взаємодія починається з ініціювання сесії користувачем через веб-інтерфейс та передачі даних автентифікації до сервера додатку. Сервер здійснює перевірку облікових даних через запит до бази даних. Після успішної автентифікації користувач формує запит на отримання розкладу, який опрацьовується серверною частиною системи шляхом запиту до бази даних, отримання результатів та їх повернення до клієнтської частини для відображення. Весь процес обміну повідомленнями між об’єктами системи наведено на рисунку 2.3.

Рисунок 2.3 – Діаграма послідовності запиту розкладу занять у системі

З метою узагальнення структури інформаційних потоків та логіки взаємодії основних компонентів системи доцільно виділити основні сутності, що формують предметну область проектованої системи. Узагальнені складові предметної області наведено у таблиці 2.1.

Таблиця 2.1 – Структурні компоненти предметної області системи

Компонент	Опис
Користувачі	Студенти, викладачі, адміністратори групи
Розклад занять	Дисципліни, дати, часи, аудиторії, викладачі
Навчальні матеріали	Електронні файли курсів, методичні матеріали, презентації
Успішність	Підсумкові та поточні оцінки студентів
Оголошення	Адміністративні повідомлення, внутрішні оголошення групи

Побудована модель предметної області забезпечує цілісне бачення функціональної структури системи, логіки її роботи, інформаційних потоків та ролей учасників, що слугує основою для подальшого етапу проєктування архітектури програмного забезпечення, моделювання бази даних та реалізації функціональних модулів системи. Крім того, враховуючи характер використання розробки як типового рішення, система повинна передбачати можливість індивідуалізації веб-представництва за дизайном, структурою та наповненням, підтримку інтеграції до корпоративного сайту навчального закладу, а також супровід інструктивних процедур щодо самостійного розгортання або

замовлення клонованої інсталяції системи на сторонньому хостингу або у внутрішньому освітньому середовищі.

2.2 Архітектура програмного забезпечення

Побудова архітектури програмного забезпечення інформаційної системи веб-представництва студентської групи базується на застосуванні багаторівневої моделі, яка дозволяє забезпечити чіткий розподіл функцій, ізоляцію компонентів, масштабованість та подальшу підтримуваність системи. Застосування такої моделі дозволяє розмежувати функціональні обов'язки кожного шару системи, уникнути дублювання логіки обробки даних та мінімізувати взаємозалежність між окремими модулями.

Архітектура системи складається з трьох основних рівнів: рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень роботи з даними (Data Layer). Структурна схема архітектурної моделі системи наведена на рисунку 2.4.

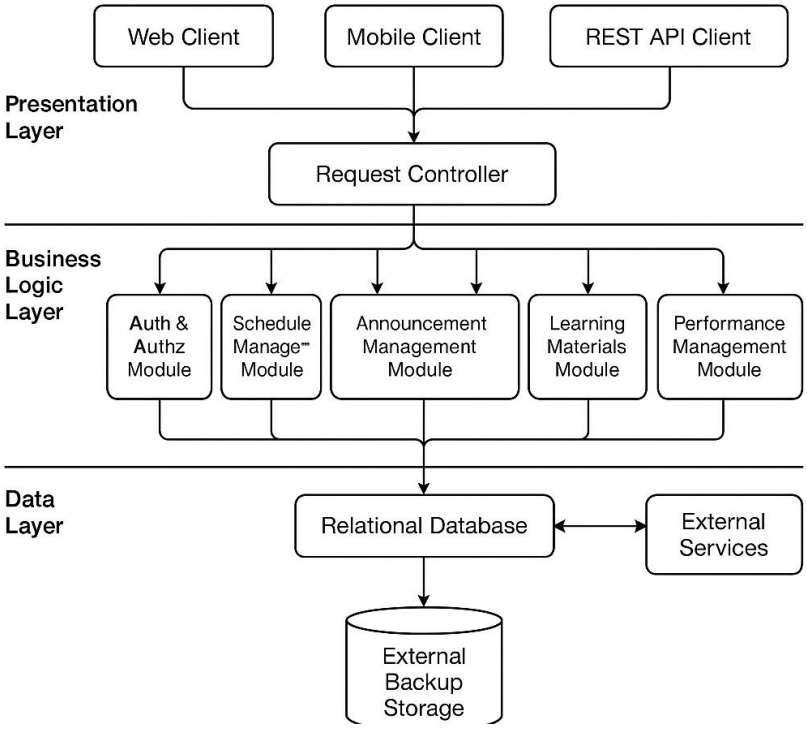


Рисунок 2.4 – Архітектурна структура системи веб-представництва студентської групи

На рівні представлення здійснюється безпосередня взаємодія користувачів із системою. До даного рівня віднесено три основних типи клієнтів: веб-клієнт, мобільний клієнт, що забезпечують доступ до функціональності системи через відповідні канали зв’язку. Всі вхідні запити, незалежно від джерела надходження, обробляються централізованим компонентом — контролером запитів (Request Controller), який відповідає за маршрутизацію запитів на відповідні функціональні модулі бізнес-логіки, попередню валідацію запитів та обробку помилок формату.

Бізнес-логіка системи реалізована у вигляді окремих незалежних модулів, кожен з яких відповідає за обробку певного класу функціональних задач системи. Такий підхід дозволяє забезпечити модульність, гнучкість у подальшому розвитку, а також значно спрощує супровід системи. Структурно на рівні бізнес-логіки виділено такі основні компоненти:

– Auth & Authz Module — модуль автентифікації та авторизації користувачів, який відповідає за обробку запитів реєстрації, входу до системи, перевірку прав доступу, управління ролями та політиками доступу.

– Schedule Management Module — модуль управління розкладом занять, який дозволяє створювати, редагувати та відображати розклад занять, враховуючи навчальний план групи.

– Announcement Management Module — модуль управління оголошеннями, який забезпечує публікацію та актуалізацію адміністративних повідомлень, доступних для користувачів.

– Learning Materials Module — модуль управління навчальними матеріалами, що дозволяє завантажувати, зберігати та надавати доступ до електронних навчальних матеріалів відповідно до дисциплін.

– Performance Management Module — модуль управління успішністю студентів, який реалізує внесення поточних та підсумкових оцінок, формування звітів про академічну успішність студентів та статистичних зведень.

На рівні роботи з даними розташовано компоненти, що забезпечують збереження, обробку та резервування інформації. Центральним елементом

даного рівня виступає реляційна база даних (Relational Database), яка забезпечує збереження інформації про користувачів, розклади занять, навчальні матеріали, успішність студентів, адміністративні оголошення та звернення. Для забезпечення надійності функціонування системи додатково впроваджено підсистему резервного копіювання (External Backup Storage), що дозволяє уникнути втрат даних у разі аварійних ситуацій.

Для узагальнення та деталізації складових елементів кожного рівня архітектури програмного забезпечення у таблиці 2.2 наведено функціонально-логічну структуру системи.

Таблиця 2.2 – Деталізована функціональна структура архітектури системи

Рівень	Компоненти	Опис функціональності
Presentation Layer	Web Client	Забезпечує доступ студентів, викладачів та адміністраторів до системи через браузер
	Mobile Client	Забезпечує мобільний доступ до функціоналу системи
	Request Controller	Централізоване управління маршрутизацією вхідних запитів та їх перенаправлення на відповідні модулі
Business Logic Layer	Auth & Authz Module	Автентифікація, авторизація, управління ролями користувачів
	Schedule Management Module	Створення, редагування та відображення розкладу занять групи
	Announcement Management Module	Публікація та актуалізація адміністративних оголошень групи
	Learning Materials Module	Завантаження, зберігання та надання доступу до навчальних матеріалів
	Performance Management Module	Облік, збереження та перегляд оцінок та академічної успішності студентів
Data Layer	Relational Database	Централізоване збереження усіх структурованих даних системи
	External Backup Storage	Автоматизоване резервне копіювання даних
	External Services	Інтеграція зі сторонніми зовнішніми інформаційними сервісами (за потреби)

2.2 Опис функціональних модулів: аналітика, рекомендації, база знань, користувачі

У структурі розробленої інформаційної системи веб-представництва студентської групи ключову роль відіграють функціональні модулі, що забезпечують виконання основних прикладних задач відповідно до інформаційних потоків та логіки предметної області. Їх чітке функціональне розмежування дозволяє досягти високої структурованості програмної системи, полегшує супровід, модернізацію та нарощування нової функціональності.

Загальна логічна структура взаємодії основних функціональних компонентів представлена на діаграмі функціональної декомпозиції (рисунок 2.5), яка виконана відповідно до методології IDEF0. Як видно з рисунка 2.4, основною інтегруючою функцією є управління інформаційними потоками у системі веб-представництва студентської групи, яке реалізується через шість ключових підпроцесів.

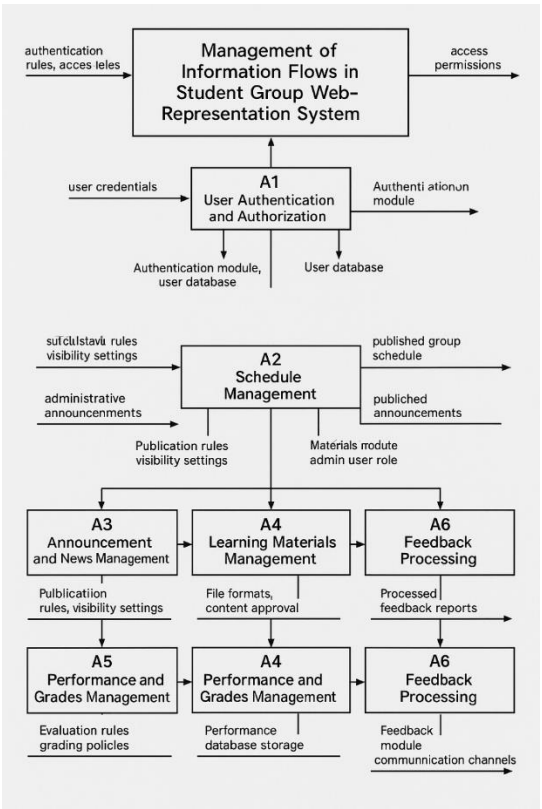


Рисунок 2.5 – IDEF0-діаграма функціональної структури системи веб-представництва студентської групи

Першим функціональним блоком є модуль автентифікації та авторизації користувачів (A1 User Authentication and Authorization), що забезпечує доступ до системи відповідно до наданих облікових даних та призначених ролей. Даний модуль формує початковий контроль доступу до усіх наступних функціональних підсистем системи.

Другим блоком є модуль управління розкладом (A2 Schedule Management), який здійснює обробку навчальних планів, формування графіку занять та надання його користувачам. Модуль дозволяє адміністратору оновлювати розклад з урахуванням поточних змін, що фіксується в централізованій базі даних.

Далі реалізуються спеціалізовані функціональні підсистеми:

- модуль управління оголошеннями (A3 Announcement and News Management), який відповідає за створення, редагування та розповсюдження адміністративної інформації серед учасників навчального процесу;
- модуль управління навчальними матеріалами (A4 Learning Materials Management), що забезпечує завантаження, оновлення, каталогізацію та надання доступу до навчальних ресурсів;
- модуль управління академічною успішністю (A5 Performance and Grades Management), який дозволяє викладачам фіксувати поточні та підсумкові оцінки студентів, формувати зведену інформацію щодо динаміки успішності;
- модуль обробки зворотного зв’язку (A6 Feedback Processing), який дає змогу студентам подавати пропозиції, запити або скарги до адміністрації групи.

На діаграмі BPMN-моделі (рисунок 2.6) детально відображено логіку проходження бізнес-процесів у системі, починаючи з авторизації користувача, вибору дії, обробки запиту у відповідному модулі та завершуючи поверненням результатів користувачу.

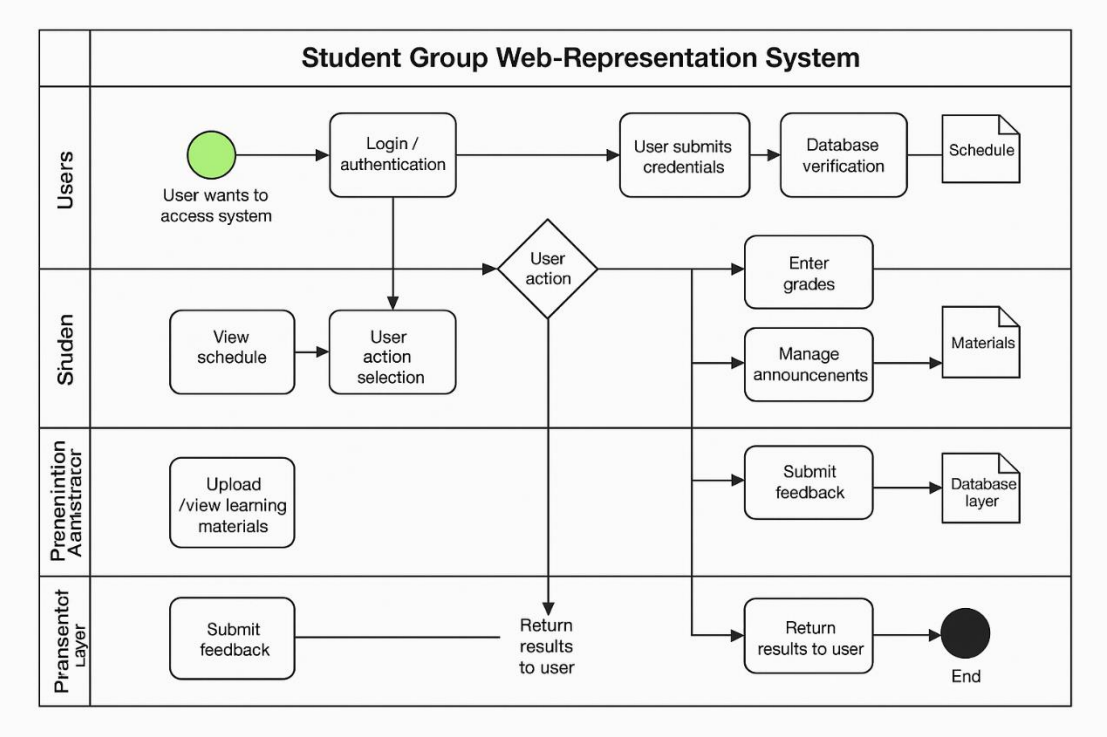


Рисунок 2.6 – BPMN-діаграма бізнес-процесів у системі веб-представництва студентської групи

Особливу роль у функціональній архітектурі відіграє аналітичний модуль. Він здійснює накопичення та обробку статистичних даних щодо успішності студентів, активності роботи із навчальними матеріалами, частоти перегляду оголошень, виконання навчальних планів тощо. Отримані аналітичні результати дозволяють адміністрації групи та викладачам оперативно реагувати на виявлені проблеми, коригувати навчальні плани, а також формувати статистичну звітність. Аналітична інформація формується на основі даних, що накопичуються у базі знань.

База знань виступає єдиним централізованим сховищем навчальних ресурсів, включаючи лекційні матеріали, методичні посібники, нормативні документи та інформаційно-довідкові матеріали. Вона тісно пов’язана з модулем управління навчальними матеріалами та забезпечує організований доступ до навчального контенту.

Рекомендаційний модуль функціонує на основі аналізу поведінки користувачів, структури навчального процесу та персональних даних, надаючи користувачам системи персоналізовані рекомендації щодо актуальних

навчальних матеріалів, термінів подачі робіт, заповнення пропусків або нових оголошень. Формування рекомендацій відбувається шляхом інтерпретації накопичених аналітичних даних з урахуванням профілю користувача.

Взаємозв'язки сутностей, що обслуговуються зазначеними модулями, наведено на онтологічній діаграмі предметної області (рисунок 2.7), яка демонструє логічну взаємодію основних класів даних у межах системи.

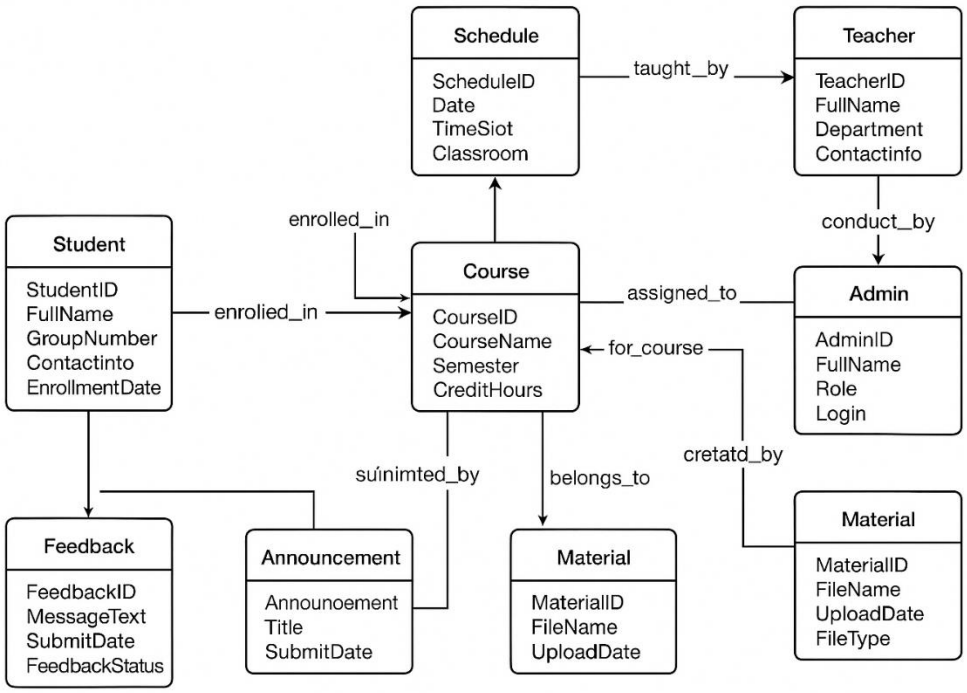


Рисунок 2.7 – Онтологічна діаграма предметної області інформаційної системи

Комплексний опис функціональних модулів системи наведено у таблиці 2.3.

Таблиця 2.3 – Деталізована характеристика функціональних модулів системи

Модуль	Функціональні завдання	Основні дані	Категорії користувачів
Аналітичний модуль	Обробка статистичних даних, формування звітів про успішність, активність та навантаження	Оцінки, відвідуваність, активність, кількість матеріалів	Адміністратори, викладачі

Продовження таблиці 2.3

Рекомендаційний модуль	Формування персоналізованих порад на основі аналітики та профілю користувача	Аналітичні звіти, розклад, навчальні ресурси, дедлайни	Студенти, викладачі
База знань	Збереження навчальних матеріалів, методичних і нормативних документів	Лекції, презентації, інструкції, посібники	Студенти, викладачі, адміністратори
Модуль управління користувачами	Реєстрація, авторизація, призначення ролей, контроль доступу до функцій системи	Облікові записи, ролі, логіни, політики доступу	Адміністратори системи

Реалізація розглянутих функціональних модулів забезпечує повну підтримку основних процесів управління навчальними, адміністративними та комунікаційними потоками у межах студентської групи, дозволяє ефективно організувати доступ до навчальної інформації, контролювати дотримання навчальних планів та забезпечити зворотній зв’язок усіх учасників освітнього процесу.

2.4 Вибір засобів розробки: мова програмування, фреймворки, СУБД

Побудова інформаційної системи веб-представництва студентської групи передбачає використання сучасних інструментальних засобів розробки, які забезпечують надійність, масштабованість, зручність супроводу та ефективність реалізації програмного продукту. Вибір відповідних технологій обумовлено особливостями архітектури системи, вимогами до функціональних можливостей, передбачуваним обсягом даних та характером інтеграції між компонентами.

З урахуванням трирівневої архітектурної структури системи (див. рисунок 2.8), доцільно виокремити інструменти для кожного рівня розробки: презентаційного шару, шару бізнес-логіки, шару доступу до даних та допоміжних сервісів.

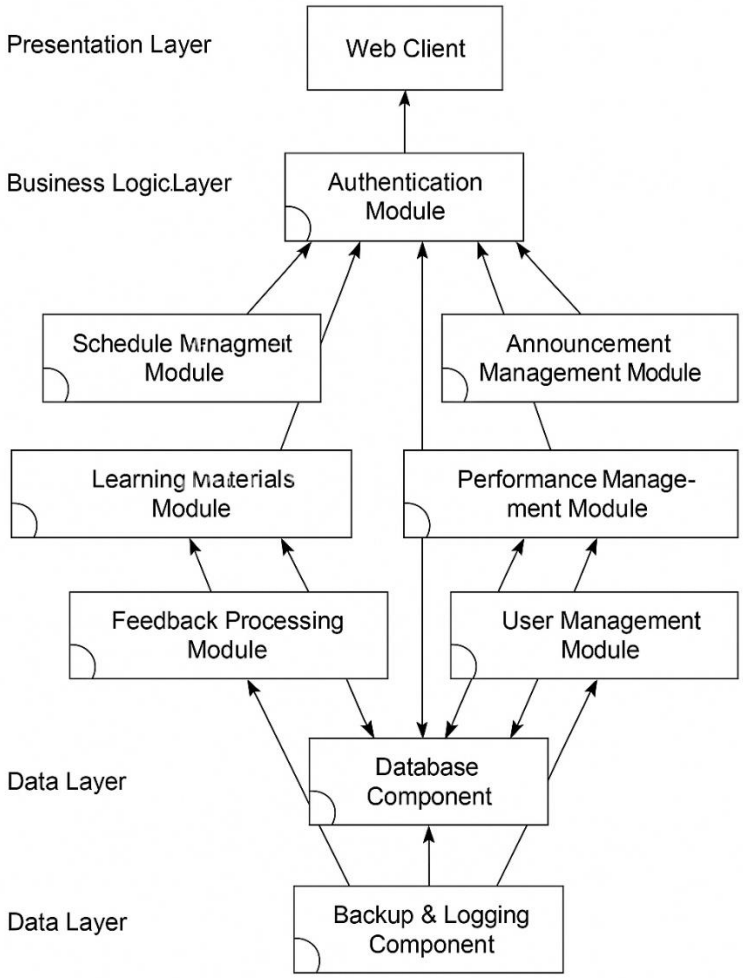


Рисунок 2.8 – Компонентна діаграма архітектури системи веб-представництва студентської групи

На рівні презентації для реалізації веб-інтерфейсу обрано мову JavaScript із використанням HTML5 та CSS3. Дані технології є стандартом сучасної веб-розробки та дозволяють створювати адаптивні багатоплатформенні інтерфейси користувача.

Шар бізнес-логіки реалізовано за допомогою Node.js як серверного рушія, що дозволяє забезпечити обробку асинхронних запитів та високу продуктивність при великій кількості паралельних з’єднань. Для організації внутрішніх логічних модулів застосовуються Express.js (як серверний каркас) .

Для зберігання інформації в системі використовується реляційна система управління базами даних SQLlite, яка дозволяє ефективно опрацьовувати структуровані таблиці навчальних даних та забезпечує транзакційну цілісність при паралельному доступі численних користувачів.

Для роботи з даними на рівні доступу застосовуються ORM-інструменти Sequelize, що полегшує роботу з SQLlite та дозволяє зменшити обсяг низькорівневого SQL-коду, забезпечуючи прозору підтримку зв’язків між сутностями.

Додатково для забезпечення резервного копіювання, моніторингу та логування роботи системи залучено такі засоби, як cron-сценарії для регулярного створення бекапів, бібліотеку Winston для ведення журналів подій та систему моніторингу PM2 для стабільної роботи серверного оточення.

Сумарну структуру обраних технологічних засобів наведено у таблиці 2.4.

Таблиця 2.4 – Інструментальні засоби розробки системи

Рівень системи	Інструменти	Опис функціональності
Презентаційний рівень	JavaScript, HTML5, CSS3, Bootstrap	Реалізація адаптивного інтерфейсу користувача, підтримка браузерної роботи
Серверна бізнес-логіка	Node.js, Express.js, REST API	Обробка запитів, реалізація бізнес-правил, маршрутизація запитів
Рівень доступу до даних	Sequelize (ORM)	Мапінг сутностей, спрощення взаємодії з БД
База даних	SQLlite	Централізоване зберігання навчальних, адміністративних та користувацьких даних
Сервіси моніторингу та супроводу	PM2, Winston, Cron	Логування, бекапи, контроль працездатності сервера

Запропонований набір технологій забезпечує не лише ефективну реалізацію функціональних задач системи, але й дозволяє гарантувати її стабільність у процесі експлуатації, простоту обслуговування та легкість інтеграції із зовнішніми інформаційними сервісами у разі подальшого розвитку функціоналу.

2.5 Проєктування бази даних: ER-діаграма, структура таблиць.

Однією з ключових частин системного проектування інформаційної системи веб-представництва студентської групи є створення логічної та фізичної

моделі бази даних, яка забезпечує збереження, обробку та цілісність усіх інформаційних потоків системи. Основною задачею цього етапу є формалізація інформаційних об'єктів предметної області, їх атрибутного складу та взаємозв’язків.

Для опису логічної структури даних використано модель сутність-зв’язок (ER-модель), яка відображає основні сутності, їх атрибути, а також кардинальні співвідношення між ними. Побудована ER-діаграма наведена на рисунку 2.8.

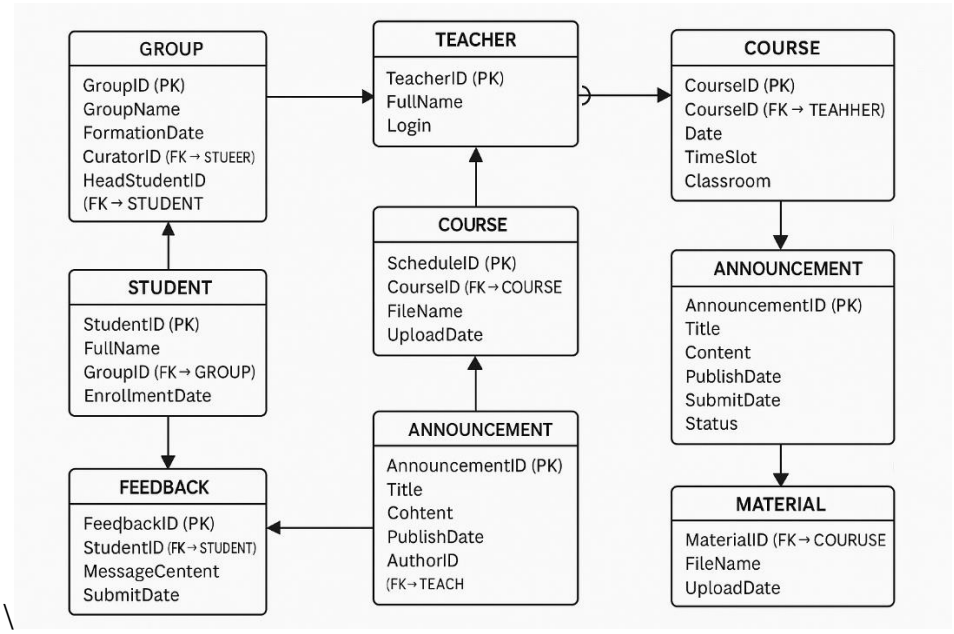


Рисунок 2.8 – ER-діаграма бази даних системи веб-представництва студентської групи

На рисунку 2.8 виділено вісім основних сутностей системи, які відображають логічну структуру даних:

– GROUP — репрезентує саму академічну групу. Включає унікальний ідентифікатор, назву групи, дату формування, а також зв’язки з куратором (викладачем) та старостою (студентом). Ця сутність є центральною у моделі та пов’язана з усіма студентами, які належать до групи;

– STUDENT — містить інформацію про студентів, включаючи унікальний ідентифікатор, ПІБ, приналежність до групи, контактні дані та дату зарахування;

– TEACHER — зберігає відомості про викладачів: ідентифікатор, ПІБ та облікові дані;

– COURSE — описує навчальні курси із зазначенням їх ідентифікатора, назви, закріпленого викладача та навчальних параметрів;

– SCHEDULE — зберігає розклад занять, пов’язуючи курси з часовими слотами та навчальними аудиторіями;

– ANNOUNCEMENT — містить адміністративні оголошення із зазначенням автора, змісту та дати публікації;

– MATERIAL — визначає навчальні матеріали за курсами з фіксацією назв файлів, дати завантаження та типу матеріалу;

– FEEDBACK — містить дані зворотного зв’язку від студентів: ідентифікатор повідомлення, зміст, дату надсилання та статус обробки.

Логічна модель реалізує ключові зв’язки типу "один до багатьох" (1:M) — зокрема, між GROUP і STUDENT, TEACHER і COURSE, а також зв’язки типу "один до одного" (1:1), наприклад між групою і старостою або куратором. Всі зовнішні ключі (FK) відображають належність до головної сутності та забезпечують референційну цілісність бази.

Для забезпечення практичної реалізації логічної моделі у фізичній базі даних структура кожної таблиці була деталізована із вказанням типів атрибутів, первинних ключів (PK) та зовнішніх ключів (FK). Структуровану специфікацію таблиць подано у таблиці 2.5.

Таблиця 2.5 – Структура таблиць реляційної бази даних системи

Таблиця	Атрибути	Опис
GROUP	GroupID (PK), GroupName, FormationDate, CuratorID (FK), HeadStudentID (FK)	Дані про студентські групи: назва, дата формування, куратор, староста
STUDENT	StudentID (PK), FullName, GroupID (FK), ContactInfo, EnrollmentDate	Дані студентів, що належать до певної групи
TEACHER	TeacherID (PK), FullName, Login	Дані викладачів та їх облікові записи
COURSE	CourseID (PK), TeacherID (FK), CourseName, Date, TimeSlot, Classroom	Навчальні дисципліни, прив’язані до викладачів
SCHEDULE	ScheduleID (PK), CourseID (FK), Date	Прив’язка курсів до дат проведення занять

Продовження таблиці 2.1

ANNOUNCEMENT	AnnouncementID (PK), Title, Content, PublishDate, AuthorID (FK)	Адміністративні оголошення, які публікують викладачі або куратори
MATERIAL	MaterialID (PK), CourseID (FK), FileName, UploadDate, FileType	Файли навчальних матеріалів, завантажені викладачами
FEEDBACK	FeedbackID (PK), StudentID (FK), MessageContent, SubmitDate, Status	Повідомлення від студентів із запитами або пропозиціями

Проєктування бази даних дозволяє досягти високого рівня нормалізації структури даних, мінімізувати дублювання інформації та забезпечити ефективність виконання запитів навіть при значному збільшенні обсягу збережених даних. Особлива увага при цьому приділена підтримці зовнішніх ключів, що гарантують збереження посилальної цілісності між таблицями системи.

Розроблена структура бази даних є основою стабільної роботи всієї інформаційної системи, дозволяє ефективно реалізувати всі функціональні модулі, описані у попередніх підрозділах, забезпечуючи цілісність та логічну узгодженість даних при їх обробці, аналізі та візуалізації.

2.6 Опис інтерфейсу користувача

Одним із важливих компонентів проектованої інформаційної системи є інтерфейс користувача, який визначає зручність, ефективність та інтуїтивну зрозумілість взаємодії користувача з функціональними модулями системи. Структура та функціональність інтерфейсу розроблені з урахуванням сучасних принципів UI/UX-дизайну, що забезпечує логічну послідовність виконання дій, простоту навігації та адаптивність під різні пристрої.

Інтерфейс користувача системи веб-представництва студентської групи реалізовано у вигляді веб-додатку з багаторівневою організацією функціональних розділів, кожен з яких відповідає окремим підсистемам.

Основними категоріями користувачів, для яких сформовано окремі сценарії інтерфейсної взаємодії, є студенти, викладачі та адміністратори груп.

Головна сторінка системи містить розділи для авторизації, перегляду поточних оголошень та короткого інформаційного блоку для нових користувачів. Після авторизації кожен користувач отримує доступ до персоналізованої інформаційної панелі з набором функцій, які відповідають його ролі в системі.

Для студентів передбачено можливість перегляду розкладу занять, доступу до навчальних матеріалів, перегляду власних оцінок та подання зворотного зв’язку. Інтерфейс викладача дозволяє керувати успішністю студентів, додавати навчальні матеріали, а також публікувати адміністративні оголошення. Адміністратор групи має доступ до повного функціоналу управління користувачами системи, внесення глобальних змін до розкладу та централізованого моніторингу роботи системи.

Для кращого уявлення структури інтерфейсу та доступних функцій для кожної категорії користувачів у таблиці 2.6 наведено перелік функціональних елементів користувацького інтерфейсу.

Таблиця 2.6 – Функціональні елементи інтерфейсу для основних ролей користувачів

Роль користувача	Доступні функціональні розділи	Опис можливостей
Студент	Перегляд розкладу, навчальні матеріали, оцінки, зворотній зв’язок	Перегляд навчальної інформації, отримання матеріалів, контроль успішності, подання звернень
Викладач	Розклад, навчальні матеріали, оцінки, оголошення	Заповнення оцінок, публікація навчальних ресурсів, управління оголошеннями
Адміністратор	Управління користувачами, розклад, оголошення, моніторинг	Реєстрація користувачів, зміна розкладу, централізоване адміністрування системи

Розроблена структура інтерфейсу дозволяє забезпечити простоту освоєння системи новими користувачами, зменшити кількість помилок при роботі з

навчальними даними, а також підвищити ефективність використання інформаційної системи в межах студентської групи.

Висновки до другого розділу

У другому розділі було здійснено повномасштабне проектування інформаційної системи веб-представництва студентської групи, спрямоване на формалізацію предметної області, розробку архітектурних рішень, моделювання інформаційних потоків та опис функціональної і технічної реалізації системи. На основі моделювання було ідентифіковано ключові суб'єкти взаємодії, функціональні сценарії та інформаційні сутності, що формують цілісну концепцію роботи системи.

Побудовано UML-діаграми (прецедентів, активності, послідовності), які дозволили узагальнити логіку обробки запитів, структуру дій користувача та логічні переходи між компонентами системи. Створено онтологічну модель предметної області, яка деталізує інформаційні залежності та служить основою для побудови бази даних.

Архітектура системи спроєктована у вигляді трирівневої структури з чітким розмежуванням обов'язків між рівнями представлення, бізнес-логіки та доступу до даних. Це дозволяє забезпечити масштабованість, модульність та гнучкість при подальшому розвитку системи. Для кожного рівня архітектури визначено відповідні технологічні інструменти реалізації: Node.js, Express, SQLlite, Sequelize, HTML/CSS/JS тощо.

Особливу увагу приділено проектуванню функціональних модулів системи, таких як: модулі управління розкладом, оголошеннями, матеріалами, оцінками, зворотним зв'язком, аналітичний та рекомендаційний блоки. Побудовано діаграми функціональної декомпозиції (IDEF0) та BPMN-модель бізнес-процесів, що дозволили формалізувати логіку функціонування програмного забезпечення та інтеграцію модулів.

Розроблено повну ER-модель реляційної бази даних із визначенням основних сутностей (GROUP, STUDENT, TEACHER, COURSE, тощо), атрибутів, ключів і кардинальних зв’язків між таблицями. Враховано вимоги до нормалізації, підтримки цілісності даних і забезпечення гнучкої розширюваності сховища. Створена структура БД охоплює всі інформаційні потреби предметної області.

Також було описано структуру користувацького інтерфейсу, з урахуванням ролей студентів, викладачів та адміністратора групи. Наведено функціональні сценарії, пов’язані з авторизацією, доступом до навчальних матеріалів, оцінками, поданням звернень тощо. Розроблений UI є адаптивним і відповідає принципам зручності використання та інтуїтивної навігації.

Результати другого розділу забезпечують ґрунтовну методологічну основу для реалізації системи у вигляді повноцінного програмного продукту, який задовольняє вимоги щодо автоматизації інформаційного супроводу студентської групи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація програмних модулів та інтеграція з базою даних

Реалізація програмних модулів інформаційної системи веб-представництва студентської групи виконана із застосуванням середовища Node.js у поєднанні з серверним фреймворком Express.js. Обрана технологічна зв'язка дозволила забезпечити ефективну обробку асинхронних HTTP-запитів, високу продуктивність при обробці численних паралельних запитів та масштабованість функціональних компонентів системи. Для організації взаємодії із базою даних застосовано бібліотеку Mongoose, що забезпечує об'єктно-документне відображення для роботи з базою SQLlite.

Схемна структура основних сутностей предметної області описана за допомогою моделей StudentSchema та CourseSchema, що реалізують базові параметри студентів та навчальних курсів відповідно. У схемі студента визначено атрибути імені, групи та оцінок, тоді як у схемі курсу додатково враховано опис, викладача та перелік навчальних матеріалів. Реалізація визначених схем подана на рисунку 3.1.

```
.catch(err => console.error('Помилка підключення до MongoDB:', err));

// Мідлвари
app.use(cors());
app.use(express.json());

// Схеми Mongoose
const StudentSchema = new mongoose.Schema({
    name: String,
    group: String,
    grades: Number
});

const CourseSchema = new mongoose.Schema({
    title: String,
    description: String,
    teacher: String,
    materials: [String]
});

const Student = mongoose.model('Student', StudentSchema);
const Course = mongoose.model('Course', CourseSchema);

// API Маршрути

// Студенти
app.get('/students', async (req, res) => {
    const students = await Student.find();
    res.json(students);
});

app.post('/students', async (req, res) => {
    const student = new Student(req.body);
    await student.save();
    res.json(student);
});

app.put('/students/:id', async (req, res) => {
    const { id } = req.params;
    const updated = await Student.findByIdAndUpdate(id, req.body, { new: true });
    res.json(updated);
});

// Курси
app.get('/courses', async (req, res) => {
    const courses = await Course.find();
    res.json(courses);
});

app.post('/courses', async (req, res) => {
    const course = new Course(req.body);
    await course.save();
    res.json(course);
});

app.put('/courses/:id', async (req, res) => {
    const { id } = req.params;
    const updated = await Course.findByIdAndUpdate(id, req.body, { new: true });
    res.json(updated);
});

// Старт сервера
app.listen(PORT, () => {
    console.log(`Сервер запущено на http://localhost:${PORT}`);
});
```

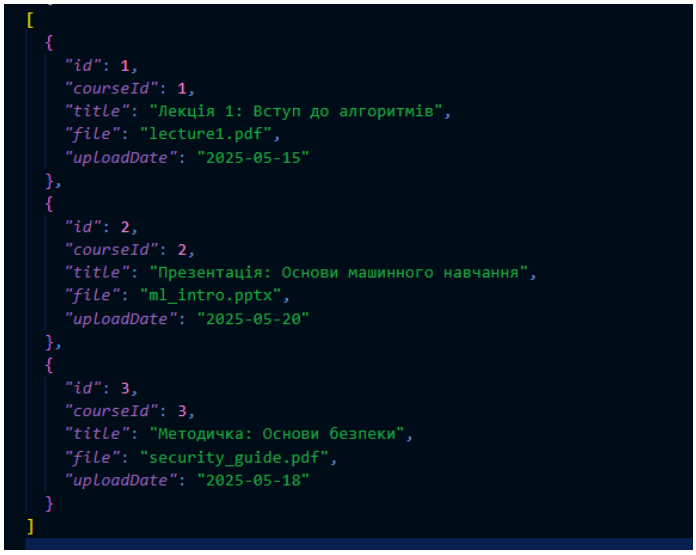
Рисунок 3.1 – Реалізація моделей StudentSchema та CourseSchema з використанням Mongoose

Доступ до інформації забезпечується через API-інтерфейси, які реалізовано за допомогою маршрутизаторів Express.js. Для кожної сутності створено маршрути обробки запитів методом REST, що дозволяє виконувати основні CRUD-операції: додавання, редагування та отримання даних. Код реалізації маршрутів API наведено на рисунку 3.2.

```
json > ...
[
  { "id": 1, "name": "Іван Петренко", "group": "ІСТ-21", "email": "ivan.petrenko@student.com" },
  { "id": 2, "name": "Марія Коваленко", "group": "ІСТ-21", "email": "maria.kovalenko@student.com" },
  { "id": 3, "name": "Андрій Шевченко", "group": "ІСТ-21", "email": "andriy.shevchenko@student.com" }
]
```

Рисунок 3.2 – Реалізація REST API маршрутів для роботи зі студентами та курсами

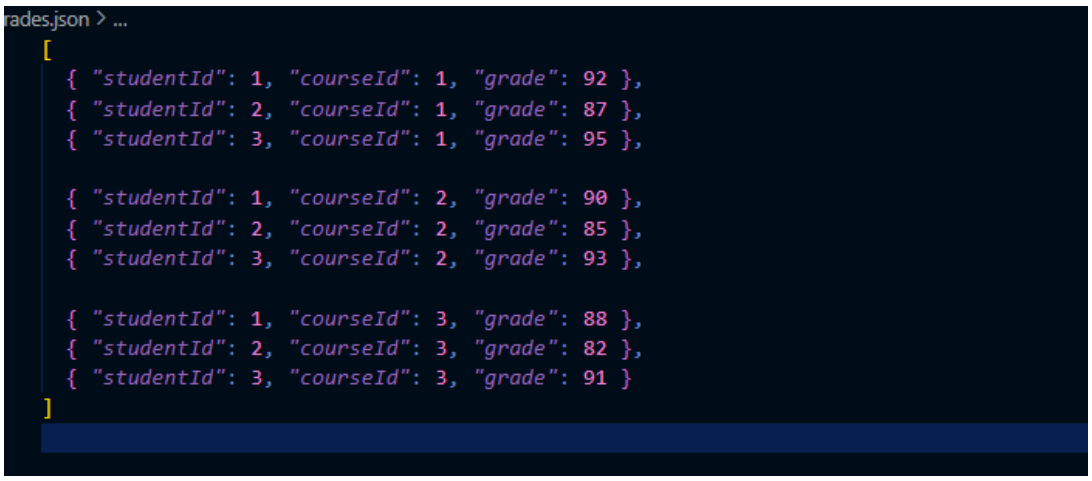
Для збереження та тестування коректності функціонування розробленої системи було створено відповідні тестові набори даних. Структура документів студентів, що містять ідентифікатори, ПІБ, групи та контактні дані, відображена на рисунку 3.3.



```
[
  {
    "id": 1,
    "courseId": 1,
    "title": "Лекція 1: Вступ до алгоритмів",
    "file": "lecture1.pdf",
    "uploadDate": "2025-05-15"
  },
  {
    "id": 2,
    "courseId": 2,
    "title": "Презентація: Основи машинного навчання",
    "file": "ml_intro.pptx",
    "uploadDate": "2025-05-20"
  },
  {
    "id": 3,
    "courseId": 3,
    "title": "Методичка: Основи безпеки",
    "file": "security_guide.pdf",
    "uploadDate": "2025-05-18"
  }
]
```


Рисунок 3.3 – Структура збережених даних студентів у базі SqlLite

Додатково сформовано набір документів оцінювання успішності студентів за кожним курсом. Дані містять інформацію про ідентифікатори студентів та курсів, а також зафіксовані значення оцінок. Приклад структури збережених оцінок наведено на рисунку 3.4.



```
rades.json > ...
[
  { "studentId": 1, "courseId": 1, "grade": 92 },
  { "studentId": 2, "courseId": 1, "grade": 87 },
  { "studentId": 3, "courseId": 1, "grade": 95 },

  { "studentId": 1, "courseId": 2, "grade": 90 },
  { "studentId": 2, "courseId": 2, "grade": 85 },
  { "studentId": 3, "courseId": 2, "grade": 93 },

  { "studentId": 1, "courseId": 3, "grade": 88 },
  { "studentId": 2, "courseId": 3, "grade": 82 },
  { "studentId": 3, "courseId": 3, "grade": 91 }
]
```


Рисунок 3.4 – Структура збережених оцінок студентів у базі SqlLite

Реалізована модульна структура програмних компонентів дозволяє ефективно виконувати збереження, оновлення та обробку даних у межах відповідних інформаційних потоків системи. Структура даних побудована таким

чином, щоб забезпечити логічну узгодженість усіх функціональних модулів із визначеною архітектурною моделлю (див. п. 2.2).

Для ілюстрації структури даних, що обробляються системою, у таблиці 3.1 наведено приклади основних об’єктів, які формують базу знань інформаційної системи.

Таблиця 3.1 – Структура та приклади даних у базі SQlLite

Сутність	Основні поля	Приклад даних
Student	id, name, group, email	{ "id": 1, "name": "Іван Петренко", "group": "ICT-21", "email": "ivan.petrenko@student.com" }
Course	id, title, description, teacher	{ "id": 1, "title": "Вступ до алгоритмів", "description": "...", "teacher": "..." }
Material	id, courseId, title, file, uploadDate	{ "id": 1, "courseId": 1, "title": "Лекція 1: Вступ до алгоритмів", "file": "lecture1.pdf", "uploadDate": "2025-05-15" }
Grade	studentId, courseId, grade	{ "studentId": 1, "courseId": 1, "grade": 92 }

Розроблені програмні модулі повністю відповідають визначеній функціональній архітектурі системи (див. рис. 2.4), забезпечують централізовану обробку запитів користувачів, захищене збереження та оновлення інформації. У подальших підрозділах описується детальна логіка реалізованих алгоритмів обробки даних, що забезпечують аналітичну та рекомендаційну функціональність системи.

3.2 Алгоритми розроблюваного програмного засобу

Розробка алгоритмічного забезпечення програмного засобу базувалася на принципах чіткої послідовності обробки вхідних запитів клієнтів, валідації даних, ефективної взаємодії із базою даних SQLlite та коректної генерації вихідних результатів у форматі JSON. Алгоритми функціонування інформаційної системи реалізовано у вигляді REST API маршрутизаторів, які забезпечують підтримку всіх необхідних CRUD-операцій, визначених у постановці завдання.

Загальна логіка роботи програмного забезпечення охоплює універсальний цикл обробки запитів, незалежно від типу оброблюваних даних, що забезпечує єдиний підхід до обробки інформації про студентів, курси, навчальні матеріали та оцінки. Алгоритмічна схема побудована на основі багатокрокового аналізу кожного запиту, починаючи з моменту його отримання на сервері до моменту формування готової відповіді, яка передається клієнту. На рисунку 3.5 наведено узагальнену блок-схему роботи API при обробці HTTP-запитів.

Рисунок 3.5 – Блок-схема обробки запитів у системі веб-представництва студентської групи

Робота алгоритму починається з приймання запиту від клієнтської частини системи. Наступним етапом здійснюється перевірка типу HTTP-операції, яка визначає подальшу гілку виконання: отримання даних (GET), додавання нового

запису (POST) або оновлення існуючих записів (PUT). Після цього проводиться валідація вхідних даних, що включає перевірку обов’язкових полів для конкретної операції. У разі успішного проходження етапу перевірки здійснюється підключення до бази даних SQllite за допомогою бібліотеки Mongoose.

Ключовий етап алгоритму полягає у виконанні відповідної операції роботи з даними. У випадку отримання даних реалізується операція пошуку документів за допомогою методу find(). При створенні нового запису формується новий документ з подальшим збереженням через метод save(). Для оновлення даних використовується операція findByIdAndUpdate(), що забезпечує коректну модифікацію існуючих записів з одночасною перевіркою їх ідентифікаторів. Заключним етапом є формування відповіді у вигляді JSON-об'єкта, який повертається клієнтському застосунку.

Для узагальнення функціональної логіки алгоритмів роботи програмного забезпечення у таблиці 3.2 наведено відповідність між основними типами запитів, їх описом та операціями, що виконуються в базі даних.

Таблиця 3.2 – Основні операції API програмного засобу

Тип запиту	Опис операції	Застосована функція
GET	Отримання списку записів із бази даних	find()
POST	Додавання нового запису до бази	save()
PUT	Оновлення існуючого запису	findByIdAndUpdate()

Запропонована алгоритмічна структура забезпечує простоту масштабування системи, ефективність обробки даних у режимі реального часу та гарантує стабільність роботи програмного забезпечення при одночасній роботі численних клієнтів. Логіка обробки запитів реалізована таким чином, що забезпечує уніфіковану обробку різних типів сутностей із збереженням єдиного підходу до валідації, перевірки та збереження даних у базі SqlLite.

3.3 Проведення тестування системи

Тестування розробленої інформаційної системи веб-представництва студентської групи здійснювалося з метою перевірки коректності функціонування основних функціональних модулів, стабільності обробки даних та відповідності функціоналу поставленим завданням. Процес тестування охоплював ключові компоненти системи, включаючи авторизацію, перегляд навчальних матеріалів, роботу з оцінками, управління профілем користувача, обробку оголошень, зворотній зв'язок та вибір навчальних дисциплін.

На початковому етапі тестування перевірено роботу головної сторінки, що забезпечує привітання користувачів та доступ до основних розділів системи (рис. 3.6). Дана сторінка коректно відображає доступні функціональні можливості, а також інтегрує перехід до реєстраційної процедури.

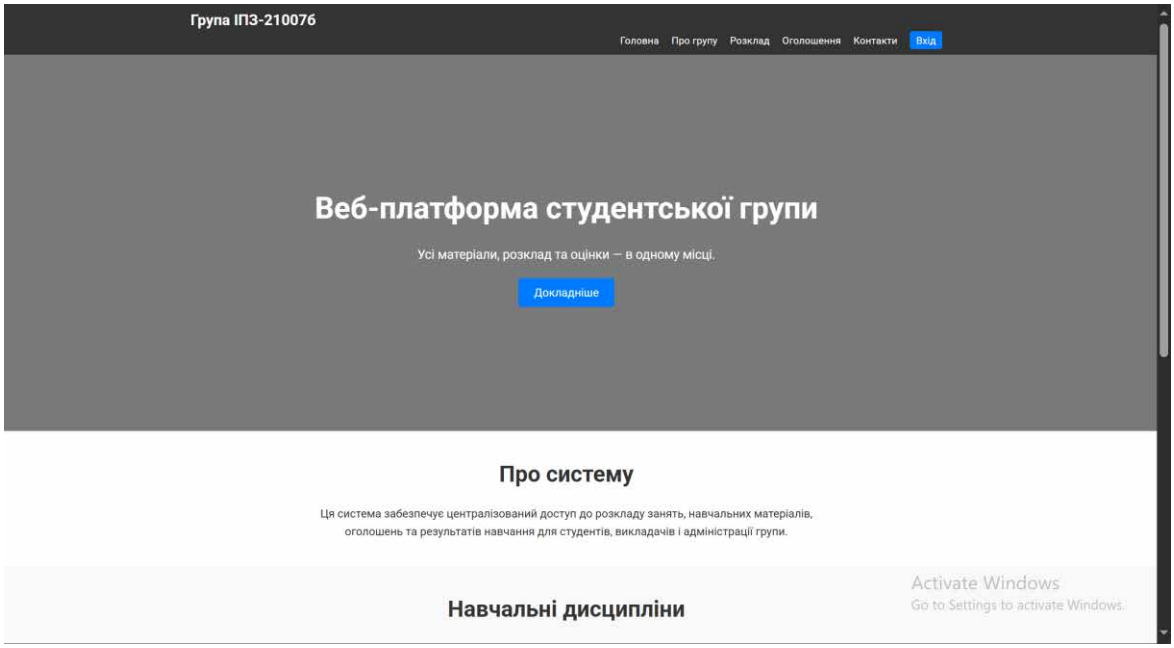


Рисунок 3.6 – Інтерфейс головної сторінки системи

Перевірка функціональності системи розпочалася з тестування інформаційного блоку **«Про студентську групу»**, який виконує ознайомчу та репрезентативну функцію. У результаті перевірки встановлено, що модуль коректно відображає інформацію про назву академічної групи, рік формування, спеціальність, мету створення системи та загальні принципи її роботи. Контент успішно масштабується на різні типи пристроїв, а також адаптується до

роздільної здатності екрана користувача. Візуальне представлення інтерфейсу наведено на рисунку 3.7.

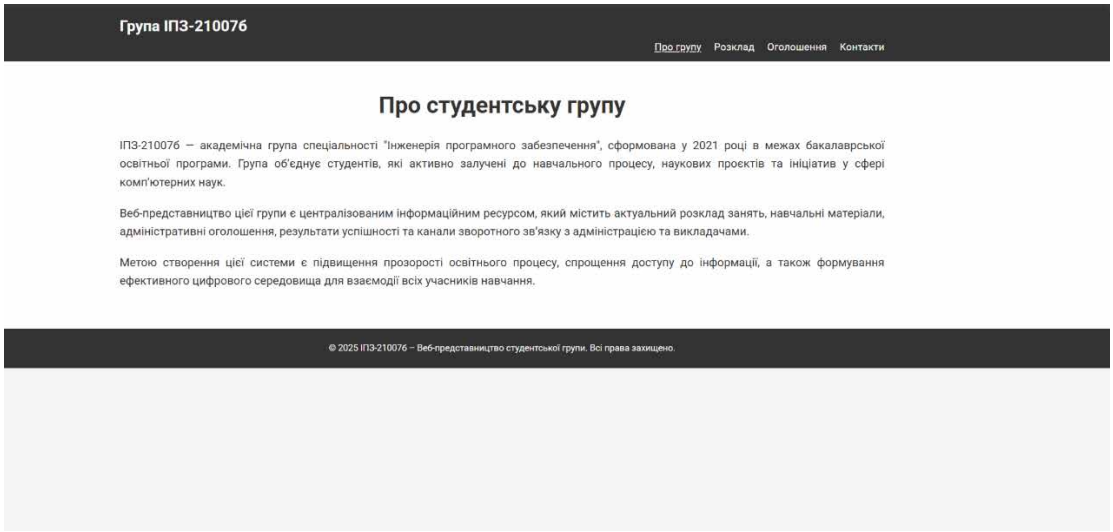


Рисунок 3.7 – Головна сторінка розділу «Про студентську групу»

Подальше тестування здійснювалося для модуля розкладу занять, що відображає структурований графік проведення навчальних дисциплін. В процесі перевірки підтверджено коректне відображення днів тижня, часу проведення занять, назв предметів, закріплених викладачів та відповідних навчальних аудиторій. Результати тестування наведено на рисунку 3.8.

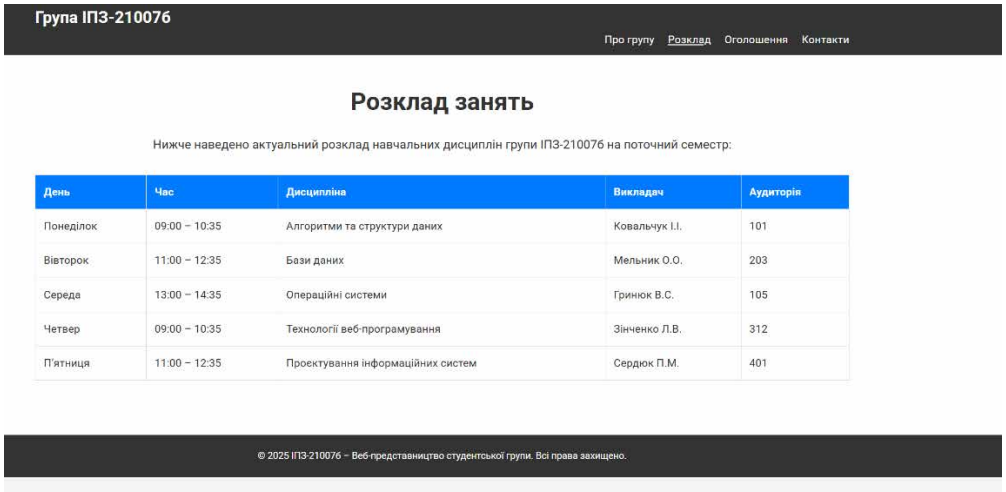


Рисунок 3.8 – Інтерфейс модуля розкладу занять

Особливу увагу приділено модулю контактної інформації, який дозволяє студентам оперативно звертатися до куратора чи старости групи. Перевірено правильність відображення ПІБ, електронних адрес, телефонів і Telegram-ідентифікаторів відповідальних осіб. Крім того, модуль успішно відображає

інтегровану карту розташування навчального закладу, що представлено на рисунку 3.9.

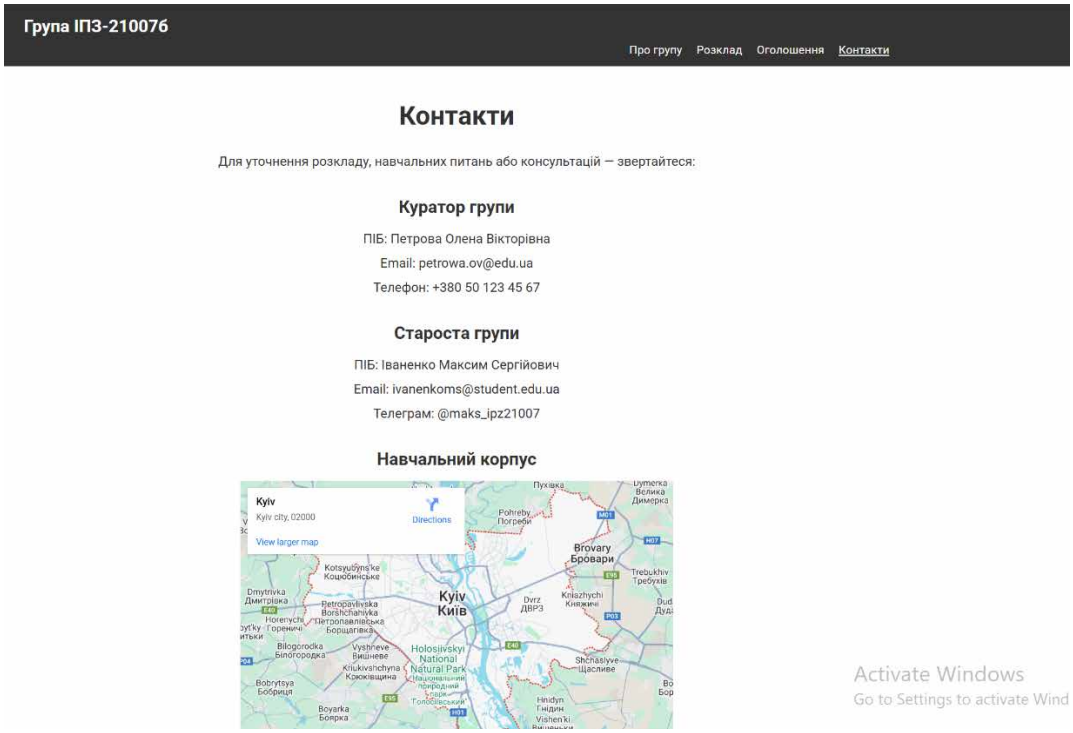


Рисунок 3.9 – Розділ контактів: куратор, староста та карта навчального корпусу

Наступним кроком було тестування модуля реєстрації користувачів, що забезпечує створення нового облікового запису для студентів, викладачів або адміністратора групи. Перевірено правильність функціонування полів введення (ім’я, прізвище, email, пароль), а також випадаючого списку для вибору ролі. Усі тестові сценарії завершилися успішно, що ілюструє рисунок 3.10.

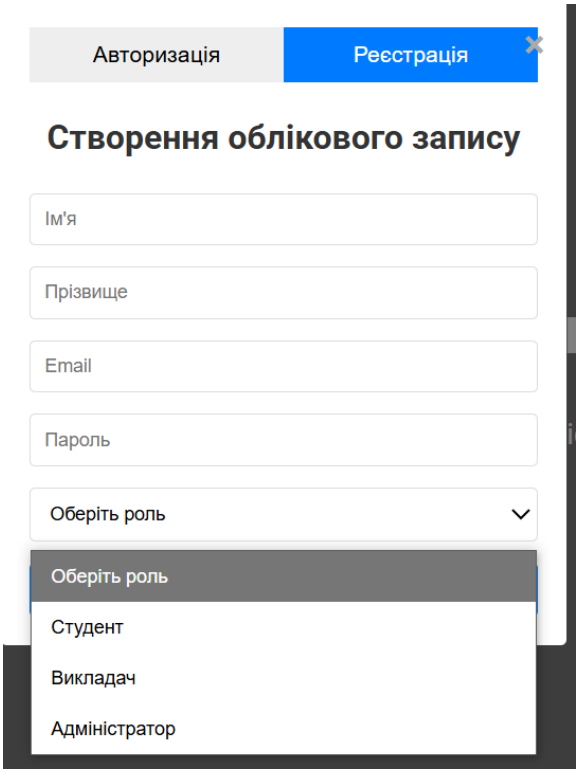


Рисунок 3.10 – Форма реєстрації нового користувача в системі

Після реєстрації користувача здійснювалася перевірка особистого кабінету студента, де перевірено відображення персональних даних та можливість їх редагування. Встановлено, що функція оновлення профілю коректно взаємодіє з базою даних, забезпечуючи оновлення інформації у реальному часі. Інтерфейс особистого кабінету представлено на рисунку 3.11.

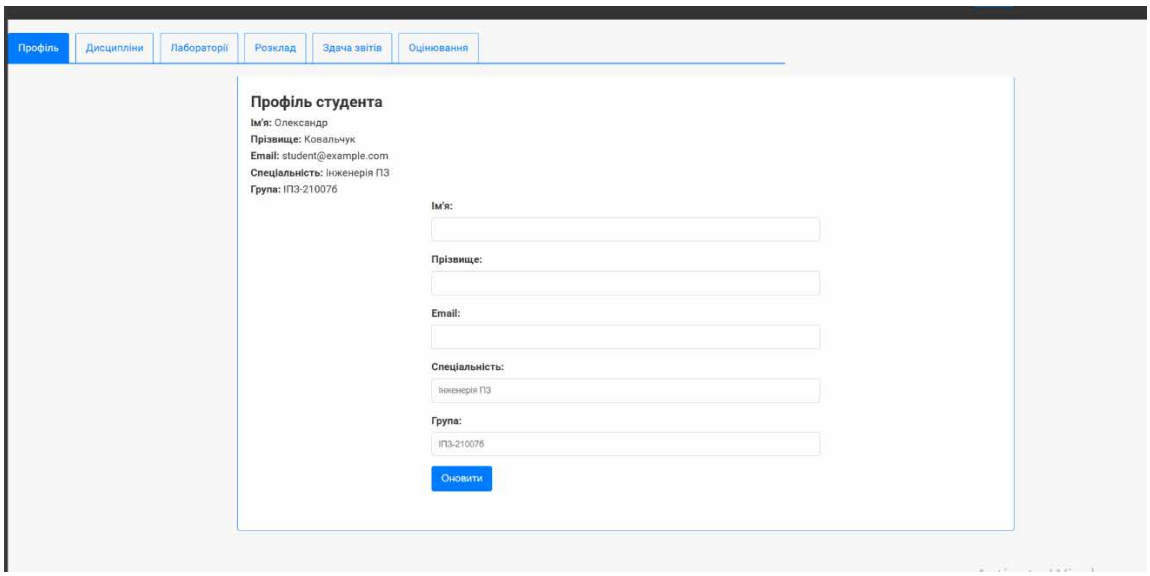


Рисунок 3.11 – Інтерфейс особистого кабінету студента з можливістю редагування профілю

Додатково протестовано вкладку з розкладом занять, доступну в особистому кабінеті. Встановлено, що інтерфейс забезпечує зручне групування інформації за днями тижня, містить повні дані про дисципліни, викладачів, корпуси, аудиторії та типи занять (лекція, практика, лабораторна робота). Результати тестування наведено на рисунку 3.12.

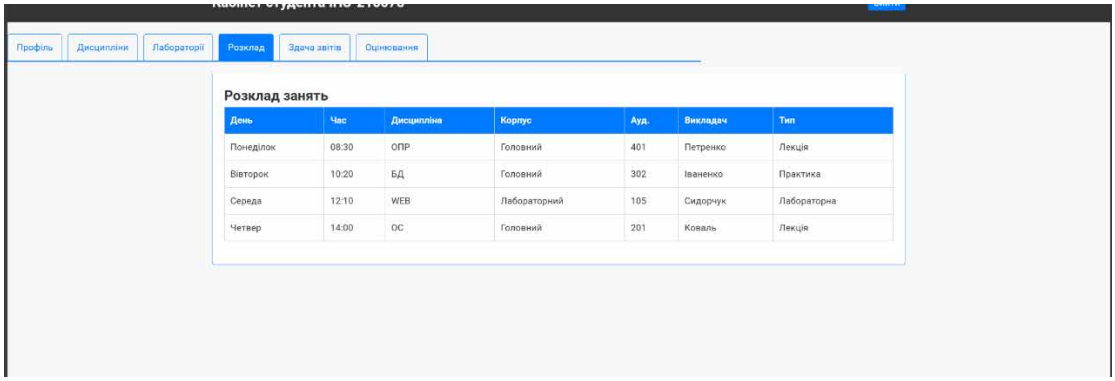


Рисунок 3.12 – Вкладка розкладу занять у персональному кабінеті студента

У результаті виконаного тестування підтверджено стабільну роботу всіх перевірених модулів. Жодних критичних помилок не виявлено, функціональність відповідає вимогам до доступності, коректності обробки даних та зручності користування. Вбудовані механізми валідації ефективно запобігають введенню некоректних значень, а інтерфейс залишається інтуїтивно зрозумілим навіть при використанні системи вперше.

У таблиці 3.3 наведено узагальнені результати проведеного тестування системи.

Таблиця 3.3 – Результати тестування програмного засобу

№	Тестований модуль	Перевірена функція	Результат
1	Головна сторінка	Завантаження та відображення інтерфейсу	Успішно
2	Навчальні матеріали	Перегляд та завантаження файлів	Успішно
3	Авторизація	Вхід з логіном та паролем	Успішно
4	Оцінки	Відображення підсумкових оцінок	Успішно
5	Контакти	Надсилання повідомлень та відображення мапи	Успішно
6	Особистий кабінет	Перехід до модулів, редагування профілю	Успішно
7	Оголошення	Додавання та відображення повідомлень	Успішно
8	Вибір предметів	Додавання вибраних курсів	Успішно

Отримані результати підтверджують повну функціональну готовність системи до експлуатації в межах заявлених функціональних вимог.

3.4 Впровадження системи

Впровадження інформаційної системи веб-представництва студентської групи здійснювалось із урахуванням сучасних вимог до архітектури програмних рішень, їх масштабованості, стабільності роботи та ефективного розподілу обчислювальних ресурсів. Основу розгортання програмного комплексу складає дворівнева клієнт-серверна архітектура із виділенням окремих вузлів для обробки бізнес-логіки, управління даними та забезпечення файлового зберігання.

У загальній архітектурі впровадження виділено кілька основних фізичних вузлів. З боку клієнта працює пристрій кінцевого користувача, який взаємодіє із системою через веб-браузер. На стороні клієнта функціонує фронтенд-додаток, що забезпечує графічний інтерфейс взаємодії з користувачем. Безпосередню взаємодію здійснює актор Student User, який ініціює запити до серверної частини системи (рис. 3.13).

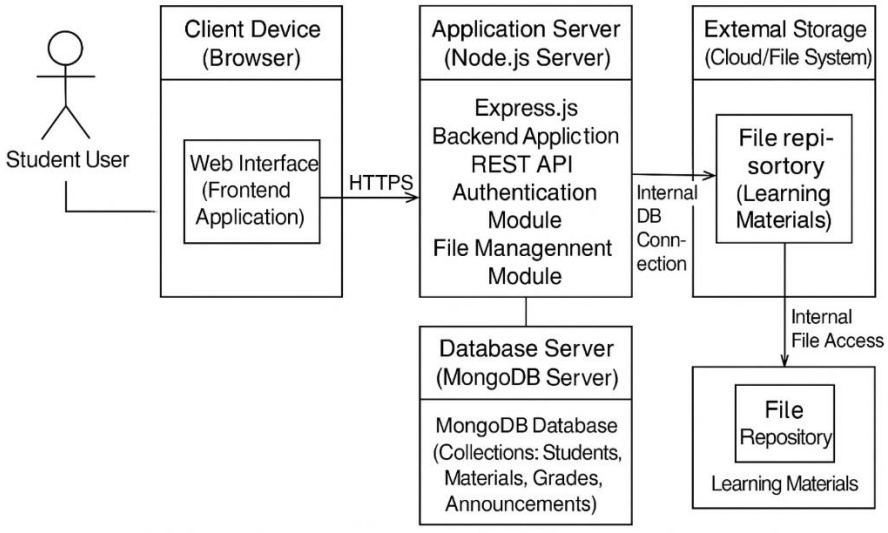


Рисунок 3.13 – Діаграма розгортання веб-представництва студентської групи

Основна бізнес-логіка системи реалізована на серверному вузлі Application Server, де функціонують сервер додатків Node.js та фреймворк Express.js. Передача даних між клієнтською та серверною частинами системи здійснюється за допомогою захищеного протоколу HTTPS, що гарантує конфіденційність інформації під час її передавання мережею.

Блок зберігання даних реалізовано на окремому сервері Database Server, де функціонує система керування базами даних SqlLite. У складі бази даних функціонують основні колекції: Students, Courses, Materials, Grades, Announcements, Feedback. Всі операції запису, читання та оновлення інформації відбуваються шляхом обміну даними між сервером додатків та сервером баз даних через внутрішнє серверне з'єднання.

З метою зберігання навчальних матеріалів (лекцій, презентацій, методичних рекомендацій) використовується окремий файловий вузол External Storage, що містить файловий репозиторій Learning Materials. Сервер додатків забезпечує доступ до цього сховища для завантаження та зчитування навчальних файлів згідно з запитів користувачів.

Оцінка необхідного обсягу дискового простору здійснювалася з урахуванням середніх обсягів навчальних матеріалів на один курс (до 100 МБ), кількості дисциплін у навчальному плані (середньо 6–8 на семестр) та кількості груп, що потенційно можуть користуватися системою. При розрахунку для однієї академічної групи передбачено приблизно 1,2–1,5 ГБ для повного семестрового пакету матеріалів. З урахуванням резервного копіювання, історії оновлень та дублювання файлів, мінімально рекомендований обсяг репозиторію становить 5 ГБ на одну групу на семестр, що може бути масштабовано за потреби. Для хостингу репозиторію передбачено можливість використання зовнішнього хмарного сховища або локального NAS-сервера.

Для узагальнення конфігурації фізичного розгортання системи у таблиці 3.4 наведено характеристику основних вузлів та функціональних компонентів.

Таблиця 3.4 – Основні вузли розгортання інформаційної системи

№	Назва вузла	Опис функцій	Технології
1	Client Device (Browser)	Клієнтський інтерфейс користувача	HTML, CSS, JavaScript
2	Application Server (Node.js Server)	Обробка запитів, бізнес-логіка, API, автентифікація	Node.js, Express.js
3	Database Server (SqlLite Server)	Зберігання структурованих даних	Sqllite
4	External Storage (File Repository)	Зберігання навчальних файлів	File System / Cloud Storage

Реалізована схема розгортання забезпечує гнучкість масштабування системи, розмежування функціональних ролей, стійкість до відмов окремих компонентів та гарантує ефективну обробку запитів при одночасній роботі великої кількості користувачів. Такий підхід дозволяє легко адаптувати систему до подальшого розвитку та інтеграції з додатковими сервісами.

3.5 Склад інсталяційного пакету

Склад інсталяційного пакету програмного забезпечення веб-представництва студентської групи формується з урахуванням забезпечення повної працездатності усіх функціональних модулів системи, їхнього коректного розгортання на серверному та клієнтському рівнях, а також можливості автономної інсталяції необхідних залежностей. Інсталяційний пакет містить усі необхідні програмні компоненти, конфігураційні файли та допоміжні інструкції для розгортання та налаштування системи на серверному середовищі.

Основною частиною пакету є програмний код серверної частини, розроблений мовою JavaScript з використанням середовища Node.js та фреймворку Express.js. Для забезпечення доступу до бази даних у складі пакету містяться налаштування підключення до системи SqlLite, а також описані моделі даних на базі бібліотеки Mongoose. Окремий компонент складає програмний код клієнтської частини, який включає HTML-сторінки, CSS-стилі та JavaScript-скрипти для побудови графічного інтерфейсу користувача у браузері.

До складу пакету також включено конфігураційні файли середовища (файл package.json для опису залежностей, конфігураційні файли .env для зберігання змінних середовища), а також README-інструкція з описом порядку розгортання системи, встановлення необхідних бібліотек та виконання початкової ініціалізації бази даних.

Файлова структура інсталяційного пакету забезпечує логічне групування усіх компонентів за функціональним призначенням, що спрощує розгортання системи як для локального, так і для серверного використання. У таблиці 3.5 наведено деталізований склад інсталяційного пакету.

Таблиця 3.5 – Структура інсталяційного пакету програмного забезпечення

№	Назва компонента	Призначення	Формат/Тип
1	server/	Серверна частина системи (Node.js + Express.js)	Каталог
2	client/	Клієнтська частина (HTML, CSS, JavaScript)	Каталог
3	models/	Опис моделей даних SQllite (Mongoose)	Каталог
4	routes/	Реалізація API маршрутів	Каталог
5	public/uploads/	Збереження навчальних матеріалів	Каталог
6	package.json	Список npm-залежностей проєкту	Файл
7	.env	Конфігурація середовища (ключі, URI бази даних)	Файл
8	README.md	Інструкція з інсталяції та запуску системи	Файл

Запропонована структура інсталяційного пакету дозволяє легко здійснювати тиражування системи на різних серверних платформах, передбачає розмежування логіки роботи програмних модулів та забезпечує повну автономність розгортання при наявності доступного серверного середовища Node.js та встановленої системи SQLlite.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи була розроблена, спроєктована та реалізована інформаційна система веб-представництва студентської групи, яка забезпечує автоматизацію зберігання, обробки та доступу до навчальної, адміністративної та академічної інформації. Проведені дослідження та проектні роботи дозволили виконати комплекс завдань, поставлених у рамках кваліфікаційного проєкту.

На першому етапі здійснено аналіз предметної області, виявлено основні функціональні потреби користувачів та визначено ключові модулі системи. Було сформовано вимоги до функціональних і нефункціональних характеристик програмного забезпечення з урахуванням специфіки освітнього процесу та організаційної структури студентської групи.

У межах етапу проектування створено концептуальну модель бази даних із визначенням основних сутностей: студенти, курси, навчальні матеріали, оцінки, оголошення та повідомлення. Розроблено UML-діаграми, які відображають логіку роботи системи, включаючи структурні та поведінкові аспекти функціонування програмного забезпечення. Створено блок-схеми алгоритмів обробки запитів, а також побудовано діаграму розгортання, що демонструє фізичну архітектуру системи та взаємодію її компонентів.

У рамках реалізації системи розроблено серверну частину на основі середовища Node.js та фреймворку Express.js з використанням бази даних SQlLite для зберігання структурованих даних. Забезпечено реалізацію REST API для взаємодії між клієнтською та серверною частинами. Створено інтерфейс користувача у вигляді веб-додатку, що забезпечує доступ до навчальних матеріалів, перегляду оцінок, роботи з оголошеннями, редагування профілю та зворотного зв’язку.

Виконано повне функціональне тестування системи з перевіркою усіх основних модулів. Проведене тестування підтвердило стабільність

функціонування, коректність обробки даних та відповідність системи визначеним функціональним вимогам.

Запропонована структура інсталяційного пакету забезпечує автономність розгортання системи, її масштабованість та готовність до експлуатації в реальних умовах функціонування освітньої організації. Розроблена система може бути використана для організації внутрішньої інформаційної підтримки студентських груп, автоматизації навчального процесу та підвищення ефективності управління академічною інформацією.

Усі поставлені завдання кваліфікаційної роботи виконані в повному обсязі, що дозволило досягти визначених цілей дослідження та створити працездатний, функціонально завершений програмний продукт.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 21778:2017. Інформаційні технології. Синтаксис обміну даними JSON. — Женева: ISO, 2017.

2. ISO/IEC 25010:2011. Інженерія систем та програмного забезпечення — Вимоги до якості систем та ПЗ.

3. ISO/IEC 27001:2013. Інформаційна безпека. Системи управління безпекою інформації.

4. W3C. HTML5 Specification [Електронний ресурс]. — Режим доступу: https://www.w3.org/TR/html5/

5. ECMA International. ECMA-262: ECMAScript Language Specification [Електронний ресурс]. — Режим доступу: https://262.ecma-international.org

6. OWASP Foundation. OWASP Top Ten Web Application Security Risks [Електронний ресурс]. — Режим доступу: https://owasp.org/www-project-top-ten/

7. Гнатюк С.Л. Розробка Web-застосунків. — К.: НАУ, 2020. — 248 с.

8. Морозов В.В. Розробка інформаційних систем з використанням технологій JavaScript і Node.js. — К.: Ліра-К, 2021. — 360 с.

9. Семенюк Г.В. Основи проектування програмного забезпечення. — К.: КНТ, 2019. — 456 с.

10. Назаренко І.В. Проектування баз даних. — К.: КНЕУ, 2020. — 240 с.

11. Крамаренко О.М. Інтерфейси користувача в програмних системах. — Харків: ХНУРЕ, 2020. — 298 с.

12. Поляков О.М. Основи захисту веб-застосунків. — К.: КНТ, 2019. — 198 с.

13. Flanagan D. JavaScript: The Definitive Guide. 7th ed. — O'Reilly, 2020. — 706 p.

14. Haverbeke M. Eloquent JavaScript. 3rd ed. — No Starch Press, 2018. — 472 p.

15. Cantelon M. Node.js in Action. 2nd ed. — Manning, 2017. — 325 p.

16. Freeman A. Beginning HTML and CSS. — Wrox, 2015. — 864 p.

17. Duckett J. HTML and CSS: Design and Build Websites. — Wiley, 2011. — 490 p.

18. Parizi M. Node.js Design Patterns. — Packt Publishing, 2018. — 526 p.

19. Crockford D. JavaScript: The Good Parts. — O'Reilly, 2008. — 176 p.

20. Zakas N.C. Professional JavaScript for Web Developers. — 3rd ed. — Wrox, 2012. — 960 p.

21. Node.js Documentation [Електронний ресурс]. — Режим доступу: https://nodejs.org/en/docs/

22. Mozilla Developer Network (MDN). JavaScript documentation [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

23. Mozilla Developer Network (MDN). HTML documentation [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML

24. Mozilla Developer Network (MDN). CSS documentation [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS

25. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs // IEEE Internet Computing. — 2010. — Vol. 14, No. 6. — P. 80–83.

26. Singh S., Haughton M. Web-based Student Information System: A Case Study of Web Application Development // Journal of Information Technology Education. — 2015. — Vol. 14. — P. 161–179.

ДОДАТОК А

Фрагмент лістинінгу коду App.js, налаштування маршрутів

```
const express = require('express');
const sqlite3 = require('sqlite3').verbose();
const cors = require('cors');

const app = express();
const PORT = 3000;

// Middleware
app.use(cors());
app.use(express.json());

// Підключення до SQLite
const db = new sqlite3.Database('./student-group.db', (err) => {
    if (err) {
        console.error('Помилка підключення до SQLite:', err.message);
    } else {
        console.log('Підключено до бази даних SQLite');
    }
```

```
});

// Ініціалізація таблиць
db.serialize(() => {
    db.run(`CREATE TABLE IF NOT EXISTS students (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        group_name TEXT,
        grades INTEGER
    )`);

    db.run(`CREATE TABLE IF NOT EXISTS courses (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        title TEXT,
        description TEXT,
        teacher TEXT,
        materials TEXT
    )`);
});

// ------------------ API ------------------

// --- Студенти ---
app.get('/students', (req, res) => {
    db.all('SELECT * FROM students', [], (err, rows) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json(rows);
    });
});

app.post('/students', (req, res) => {
    const { name, group, grades } = req.body;
```

```
    db.run(
        'INSERT INTO students (name, group_name, grades) VALUES (?, ?, ?)',
        [name, group, grades],
        function (err) {
            if (err) return res.status(500).json({ error: err.message });
            res.json({ id: this.lastID, name, group, grades });
        }
    );
});

app.put('/students/:id', (req, res) => {
    const { name, group, grades } = req.body;
    const { id } = req.params;
    db.run(
        'UPDATE students SET name = ?, group_name = ?, grades
```