Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕКАНАТУ ДЛЯ ЕФЕКТИВНОГО УПРАВЛІННЯ ОСВІТНІМ ПРОЦЕСОМ

Виконав:

здобувач IV курсу

групи ІПЗ-41

спеціальності 121 «Інженерія програмного забезпечення»

Яловенко Любомир Володимирович

Науковий керівник:

к. т. н., доцент Крайчук С. О.

Рівне – 2025

ЗМІСТ

ВСТУП

Актуальність дослідження. В даний час вищі навчальні заклади стикаються з проблемою управління навчальним процесом, яка потребує ефективних інструментів для обробки інформації великих об’ємів, спрощення і автоматизації документообігу, та забезпечення продуктивного спілкування між студентами, викладачами і адміністрацією. Інформаційна система деканату надає такий інструмент об’єднуючи в собі функціонал для зручної маніпуляції над даними, своєчасному оприлюдненні та безперешкодному доступі до них. Це сприяє підвищенню рівня організації навчального процесу, зменшує кількість очевидних помилок під час обробки інформації, та збільшує швидкість прийняття управлінських рішень на основі актуальних даних.

Вище зазначена система була розроблена у 2024 році студентом Рівненського державного університету Яловенко Любомиром в якості виконання магістерської роботи [6]. Однак існуюча система також має низку недоліків, таких як примітивний інтерфейс, низька продуктивність, неповноцінний функціонал та недостатній рівень безпеки даних. Вирішення даних проблем обумовлює актуальність дослідження, адже в сучасних умовах ці проблеми є критичними для надійної та ефективної експлуатації системи.

Мета дослідження полягає в удосконаленні попередньо розробленої системи, виявивши та усунувши недоліки, покращивши функціонал та підвищивши її безпеку.

Завдання дослідження. Для досягнення поставленої мети визначено наступні завдання:

1. Аналіз існуючої інформаційної системи деканату та виявлення її слабких місць.
2. Проведення комплексного тестування системи (продуктивність, безпека, юзабіліті).
3. Розробка пропозицій щодо модернізації системи (архітектура, функціонал, інтерфейс).
4. Впровадження вдосконалень.

Об'єктом дослідження є процес функціонування інформаційної системи деканату в закладі вищої освіти, який визначається потребою в автоматизації управління навчальним процесом.

Дослідження спрямоване на вдосконалення цього процесу з метою підвищення ефективності управління освітньою діяльністю та оптимізації документообігу.

Предметом дослідження є архітектура, функціональні можливості, продуктивність та інформаційна безпека інформаційної системи деканату, які визначаються науковими цілями щодо її удосконалення.

Дослідження спрямоване на детальний аналіз цих аспектів для виявлення недоліків і розробки шляхів їх усунення.

Методи дослідження. Для розробки застосунку використано теоретичні методи аналізу та синтезу, а також емпіричні методи, зокрема спостереження та тестування розроблених алгоритмів на практиці.

Практичне значення дослідження. Результати роботи можуть бути використані для покращення ефективності управління освітнім процесом у закладах вищої освіти. Модернізована система охоплює більшу кількість рутинних задач, які вдалося автоматизувати, що дозволить працівникам адміністрації зосередитись на більш серйозних проблемах.

Апробація та впровадження результатів дослідження. Основні теоретичні результати дослідження були представлені на XVIII Всеукраїнській науково-практичної конференції здобувачів вищої освіти і молодих учених «Наука, освіта, суспільство очима молодих» (м. Рівне, 14 травня 2025р.) та подані у вигляді тез конференції [7].

Структура роботи. Робота складається з вступу, основної частини, яка включає чотири розділи, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕКАНАТУ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Інформаційна система деканату та її призначення

Сучасні заклади вищої освіти (ЗВО) потребують ефективних засобів управління навчальним процесом, що дозволяють швидко обробляти великі обсяги інформації, автоматизувати рутинні процеси та забезпечувати зручну комунікацію між усіма учасниками освітнього середовища. У цьому контексті інформаційна система деканату є важливим елементом цифрової інфраструктури ЗВО, що забезпечує підтримку адміністративних, навчальних і управлінських функцій.

Інформаційна система – це організований комплекс організаційно-технічних заходів (сукупність підприємств, підрозділів і фахівців), а також безпосередньо інформаційних технологій і інформаційних ресурсів, призначених для функціонування інформаційних процесів, зокрема створення, поширення, використання, систематизації, збереження і знищення інформації [3,с.83].

Опис та розробка інформаційної системи проводилось у попередній науковій роботі [6]. Тому, для кращого ознайомлення із сферою дослідження, варто коротко описати вже існуючу систему до етапу модернізації.

Безпосередньо сама ж інформаційна система деканату (ІСД) — це спеціалізоване програмне забезпечення, яке об’єднує дані про студентів, викладачів, навчальні плани, дисципліни, розклади занять, тощо. Основною метою такої системи є створення єдиного цифрового простору, де кожен користувач — студент, викладач, методист, адміністратор чи співробітник деканату — має доступ до необхідної інформації відповідно до своїх повноважень.

Для студентів ІСД надає персоналізований доступ до різноманітних елементів навчального процесу. Зокрема, студент може переглядати свій індивідуальний розклад, який формується з урахуванням факультету, курсу, навчальної групи та обраних дисциплін. Завдяки безперешкодному доступу до

розкладу, який оновлюється в реальному часі, студенти отримують змогу ефективно планувати свій час, для більш успішного навчання. Також система дозволяє відстежувати інформацію навчального процесу протягом року, таку як, розклад екзаменів та заліків, терміни проведення сесії, практики, канікул та державних екзаменів.

Викладачі використовують інформаційну систему для ознайомлення з розкладом занять, перегляду переліку закріплених дисциплін, а також контролю за виконанням свого навчального навантаження. Наявність централізованої платформи спрощує взаємодію з адміністрацією факультету, дозволяє своєчасно отримувати оновлення щодо розкладу та змін у навчальному процесі.

Методисти факультетів, які займаються плануванням та організацією навчального процесу, використовують ІСД для формування навчальних планів, внесення даних про дисципліни, кількість годин, кредити за ЄКТС, форми контролю. Система надає базові інструменти для введення та редагування цієї інформації, що дозволяє уникати дублювання даних і зменшити кількість помилок.

Менеджери деканату відповідають за організацію освітнього процесу в межах факультету. Вони використовують інформаційну систему для формування академічних груп, закріплення викладачів за дисциплінами, створення та коригування розкладу занять, управління обліковими записами співробітників. Однією з головних функцій ІСД є можливість оперативного оновлення розкладу в режимі реального часу. Це дає змогу швидко реагувати на ситуації коли викладач не має можливості провести пару, потрібна аудиторія зайнята або потрібно перенести заняття. Завдяки цьому ІСД виступає як інструмент координації між усіма учасниками навчального процесу, сприяючи підвищенню ефективності управління.

Адміністратори системи є відповідальними за технічну підтримку, безпеку та стабільність роботи інформаційної системи. Вони здійснюють управління правами доступу, створення та видалення облікових записів користувачів, контроль за цілісністю та захищеністю даних. Сучасна інформаційна система має підтримувати багаторівневу авторизацію, шифрування даних та систему логування для

забезпечення інформаційної безпеки. Важливим елементом є також регулярне резервне копіювання, що дозволяє відновити дані в разі технічного збою чи кібератаки.

Отже, інформаційна система деканату є багатофункціональним інструментом, що охоплює всі аспекти освітньої діяльності. Вона дозволяє не лише зберігати та обробляти великі обсяги даних, але й надає засоби для планування, моніторингу та аналізу освітнього процесу. Її впровадження сприяє підвищенню ефективності управління вищим навчальним закладом, покращенню якості освітніх послуг та створенню прозорого, зручного і динамічного освітнього середовища.

1.2. Технічна реалізація інформаційної системи

1.2.1. Архітектура проєкту

Інформаційна система деканату розроблена з дотриманням архітектури Model-View-Template.

Архітектура Модель-Представлення-Шаблон, також відома як MVT, — це шаблон проектування програмного забезпечення для розробки вебзастосунків [12]. Така архітектура забезпечує чітке розмежування функціональних складових застосунку, що сприяє полегшенню процесів розробки та супроводу, а також підвищенню масштабованості та підтримуваності коду.

Архітектура MVT включає три основні компоненти:

– Модель (Model) відповідає за структуру даних у застосунку Django. Вона реалізується за допомогою механізму об'єктно-реляційного відображення (ORM), що дозволяє працювати з даними бази даних як з об’єктами Python, уникаючи написання сирих SQL-запитів. Модель забезпечує засоби для додавання, оновлення, зчитування та видалення інформації з бази даних, тобто виконує повну обробку даних, необхідних для функціонування застосунку.

– Представлення (View) є функцією-обробником, яка приймає HTTP-запити, опрацьовує їх і повертає відповідь у вигляді HTTP-відповіді. Для формування відповіді представлення може отримувати дані з моделі та відображати їх за допомогою шаблонів. Воно також може динамічно створювати HTML-сторінки, заповнюючи їх актуальними даними, отриманими з моделі.

– Шаблон (Template) — це текстовий файл, що визначає структуру або макет інтерфейсу користувача. Найчастіше це HTML-файл, хоча можливе використання й інших форматів, таких як XML. Шаблони в Django можуть отримувати дані з представлення та відображати їх, використовуючи синтаксис Jinja, який дозволяє динамічно формувати вміст інтерфейсу.

Завдяки такій організації архітектури MVT забезпечується ефективне розмежування логіки роботи з даними, управлінням і відображенням, що істотно полегшує розробку складних та гнучких вебзастосунків.

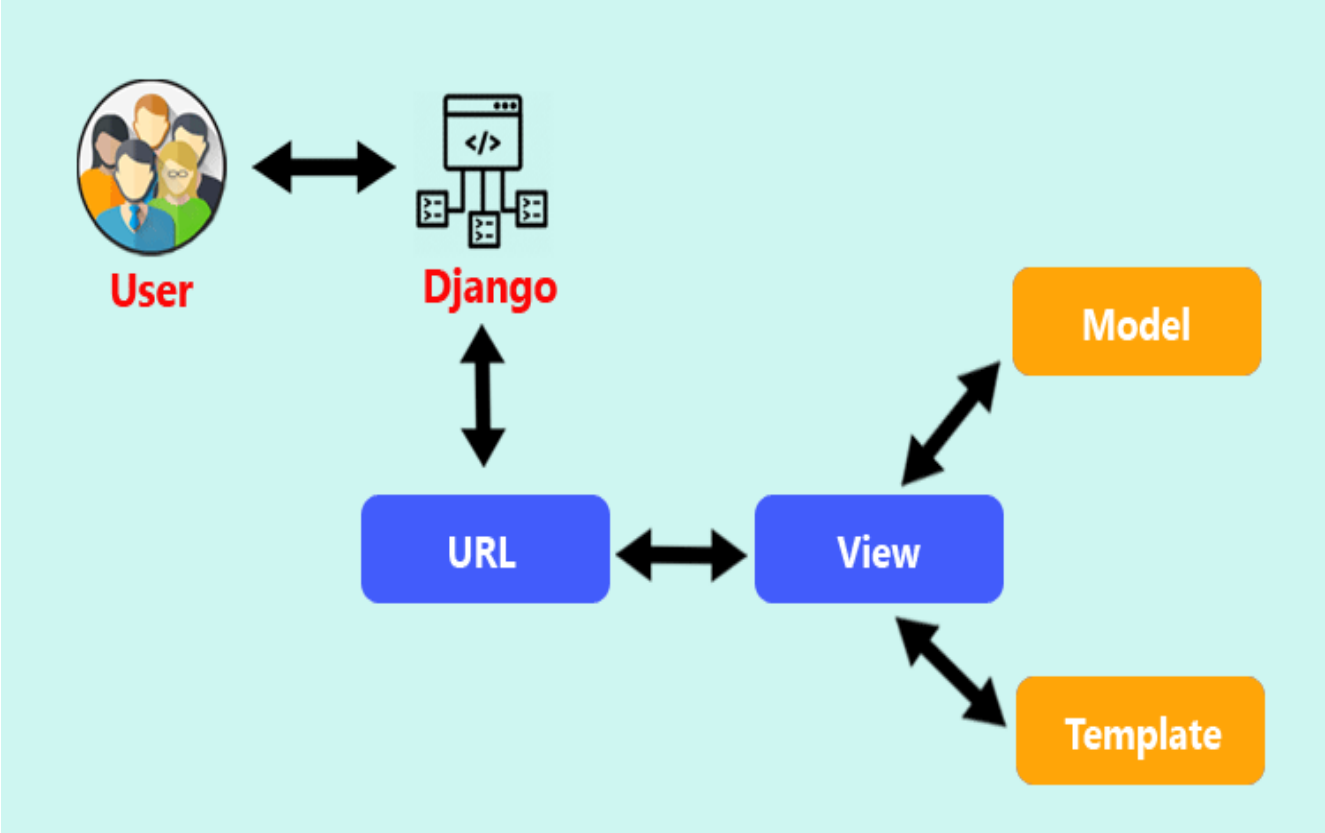

Рис. 1.1 - Робота архітектури MVT

У межах архітектури Model-View-Template (MVT) фреймворку Django керування потоком виконання здійснюється за наступним принципом:

1. Коли користувач звертається до вебзастосунку Django за допомогою URL-запиту, цей запит передається в систему маршрутизації, яка визначає відповідне представлення (View), виходячи з конфігурації URL-шляхів.

2. Після знаходження відповідного представлення відбувається його виклик.

3. Представлення звертається до моделі (Model) для отримання необхідних даних із бази даних, використовуючи механізм ORM (Object-Relational Mapping).

4. Отримані дані передаються до шаблону (Template), який формує динамічний HTML-вміст для відображення користувачеві.

Архітектура MVT у Django часто порівнюється з класичною архітектурною моделлю Model-View-Controller (MVC). Хоча обидві парадигми мають спільні риси, між ними існують принципові відмінності:

– Контролер проти представлення. У MVC за обробку взаємодії користувача та оновлення моделі й інтерфейсу відповідає контролер. У Django ж функціональність контролера розподілена між механізмами маршрутизації фреймворку (які визначають, яке представлення викликати) та функцією представлення, що виконує обробку запиту.

– Шаблони. Обидві архітектури використовують шаблони для формування інтерфейсу користувача, однак у Django шаблонна система інтегрована безпосередньо у фреймворк, що значно спрощує її використання.

– Простота. MVT не вимагає явного визначення контролера, завдяки чому зменшується кількість шаблонного коду, а процес розробки стає більш лаконічним і структурованим.

Таблиця 1.1

Порівняння ахітектур MVC та MVT

Аспект	**MVC (Model-View-Controller)**	**MVT (Model-View-Template)**
Роль контролера	Опрацьовує введення користувача та оновлює модель і представлення.	Реалізується механізмами маршрутизації Django та частково функцією представлення.
Представлення (View)	Відповідає за відображення інтерфейсу користувача, але отримує дані через контролер.	Поєднує логіку обробки даних і відображення інтерфейсу; взаємодіє безпосередньо з моделлю.
Шаблон (Template)	Використовується для опису структури інтерфейсу користувача.	Динамічно формує інтерфейс, отримуючи дані з представлення; інтегрований у фреймворк.
Розподіл відповідальності	Явний контролер розмежовує логіку, інтерфейс та маршрутизацію.	Відсутній явний контролер, що спрощує розробку.
Простота реалізації	Потребує більшої кількості шаблонного коду.	Має спрощену структуру з меншою кількістю компонентів.

1.2.2. Backend (Django)

У якості серверної частини (бекенду) для розробки інформаційної системи деканату було використано фреймворк Django — сучасний інструмент для створення вебзастосунків мовою програмування Python.

Django — це вебфреймворк, що реалізує архітектурну модель MVT (Model-View-Template) і призначений для створення вебзастосунків. Він позиціонує себе як фреймворк із підходом «batteries included», тобто має вбудовану підтримку широкого спектра функціональностей, необхідних для повноцінної розробки. Django вирізняється своєю надійністю, простотою у використанні та спрямованістю на написання чистого, ефективного й продуктивного коду. Цей фреймворк є одним із найвідоміших і найпопулярніших у світі. Його активно використовують великі компанії та організації, зокрема Instagram, YouTube, Google і навіть NASA, що свідчить про його масштабованість, стабільність і високий рівень безпеки [18].

Рис 1.2 Логотип «Django»

Фреймворк Django надає широкий набір інструментів, що значно спрощують процес розробки вебзастосунків. Варто виділити такі:

– Django реалізує механізм ORM, що забезпечує відповідність між таблицями бази даних і класами Python. Завдяки цьому розробники можуть працювати з даними на високому рівні абстракції, не використовуючи безпосередньо SQL-запити.

– Фреймворк має вбудовану систему керування аутентифікацією користувачів та розподілом прав доступу, що дозволяє ефективно реалізовувати контроль доступу до ресурсів застосунку.

– Django автоматично генерує веб-інтерфейс адміністратора для керування моделями та даними в базі. Цей інтерфейс є висококонфігурованим і може бути розширений для підтримки спеціалізованих функцій.

– Фреймворк підтримує засоби інтернаціоналізації та локалізації, що дозволяє створювати багатомовні вебресурси з урахуванням географічних або культурних особливостей цільової аудиторії.

– Система маршрутизації в Django дає змогу визначати відповідність між URL-адресами та функціями подання, створюючи логічну, зрозумілу та масштабовану структуру шляхів у вебзастосунку.

– Вбудований шаблонізатор Django дає змогу створювати динамічні вебсторінки, поєднуючи HTML з Python-кодом. Це забезпечує гнучке формування інтерфейсу користувача.

Жодна технологія не є абсолютно досконалою, і це цілком справедливо стосується фреймворку Django. Попри те, що він належить до провідних інструментів для веброзробки мовою Python та має велику кількість переваг, під час роботи з ним можуть виникати певні труднощі. Розглянемо основні сильні та слабкі сторони цього фреймворку.

Переваги Django:

1. Філософія «batteries included». Django дотримується підходу «все включено», що означає наявність великої кількості вбудованих інструментів і модулів, необхідних для повноцінної розробки вебзастосунків. Замість того, щоб розробляти функціонал «з нуля», розробник може просто імпортувати необхідні пакети.

До вбудованих можливостей Django належать:

– ORM (об’єктно-реляційне відображення);

– система міграцій баз даних;

- підтримка багатомовності та мультисайтовості;
- модель MVC/MVT;
- система автентифікації (auth);
- адміністративна панель (admin);
- підтримка RSS та Atom-стрічок;
- генерація API;
- маршрутизація URL-адрес;
- інтеграція з AJAX тощо [8].

2. Стандартизована структура. Django як фреймворк задає структуру проекту. Вона допомагає розробникам розуміти, де і як додавати нову функціональність. Завдяки однакової для всіх проектів структурі набагато простіше знайти вже готові рішення або отримати допомогу від спільноти.

3. Великий набір бібліотек. У бібліотеках зберігаються готові функції, які дозволяють вирішувати цілий ряд задач. Це також спрощує роботу програміста та пришвидшує процес створення сайту. Крім того, ймовірність помилок знижується за рахунок використання готових та перевірених бібліотек.

4. Універсальність і масштабованість. Django здатний обробляти високі навантаження, що дозволяє використовувати його в масштабних проєктах. Його застосовують у сферах управління контентом, наукових обчислень, автоматизації та корпоративного адміністрування.

5. Активна спільнота. Фреймворк має велику й активну спільноту розробників, яка постійно вдосконалює інструмент. Документація до Django є вичерпною й може слугувати повноцінним навчальним ресурсом для початківців.

6. Безпека. Django розроблений з урахуванням сучасних вимог безпеки. Він має вбудовані механізми запобігання поширених атак на зразок SQL-ін’єкцій (XSS) і підробки міжсайтових запитів (CSRF).

Недоліки Django:

1. Відсутність конвенцій. У порівнянні з іншими фреймворками, як-от Ruby on Rails, у Django відсутні чітко визначені конвенції. Більшість елементів

потрібно налаштовувати вручну, що збільшує обсяг конфігураційного коду та може уповільнювати розробку для новачків.

2. Монолітна структура. Хоча єдина структура проєкту може здаватися логічною, вона обмежує гнучкість розробника. Django вимагає дотримання суворої файлової структури та встановлених правил, що може ускладнити адаптацію фреймворку до специфічних вимог проєкту.

3. Невідповідність для невеликих проєктів. Через складну структуру та об’ємність коду Django не завжди є оптимальним вибором для проєктів з обмеженим функціоналом або невеликим навантаженням, оскільки може створювати зайві витрати ресурсів сервера.

4. Django розвивається повільно. Django є великим і монолітним фреймворком. Це дозволяє спільноті розробляти сотні універсальних модулів і додатків, але знижує швидкість розробки самого Django. Крім того, фреймворк повинен підтримувати зворотну сумісність, тому він розвивається відносно повільно.

5. Високий поріг входження. Фреймворк має доволі складну структуру для початківців. Особливо це стосується розробників, які переходять з інших мов програмування або фреймворків з іншим підходом до організації проєктів. Незвичний синтаксис і велика кількість концепцій можуть ускладнити процес навчання.

Django є потужним інструментом для веброзробки, який поєднує в собі гнучкість, масштабованість і високий рівень безпеки. Його використання у проєкті дало змогу реалізувати повноцінну функціональність інформаційної системи деканату, забезпечивши при цьому зручність підтримки та розширення в майбутньому.

1.2.3 Frontend (React.js)

ReactJS — це потужна бібліотека JavaScript, що широко використовується розробниками для створення користувацьких інтерфейсів вебзастосунків. Вона була розроблена компанією Facebook у 2011 році з метою підвищення

продуктивності вебресурсів, і відтоді здобула значну популярність серед розробників усього світу. Однією з ключових переваг ReactJS є використання віртуального DOM, який забезпечує ефективне оновлення елементів інтерфейсу без потреби перезавантаження всієї вебсторінки. Крім того, бібліотека сприяє створенню багаторазових (реюзабельних) компонентів, що значно спрощує розробку складних застосунків, покращує структуру коду та прискорює процес розробки [24].

Рис 1.3 Логотип «React»

Під час модифікації клієнтської частини інформаційної системи деканату було прийнято рішення використати бібліотеку ReactJS з метою удосконалення та модернізації інтерфейсу користувача. Вибір саме цього інструменту зумовлений потребою підвищити функціональність, гнучкість і зручність взаємодії з системою. ReactJS надає низку можливостей, які дозволяють створювати сучасні, інтерактивні й масштабовані вебінтерфейси:

1. Компонентно-орієнтована архітектура. React підтримує модульний підхід до розробки, розділяючи інтерфейс на багаторазові (реюзабельні) компоненти. Це значно спрощує процес розробки та супроводу, оскільки кожен компонент можна створювати, тестувати та оновлювати незалежно від інших [14].

2. Віртуальний DOM. Завдяки віртуальному DOM React оптимізує продуктивність, оновлюючи лише ті частини інтерфейсу, які зазнали змін, без

потреби повного перерендерення сторінки. Це забезпечує швидшу взаємодію з користувачем і покращує загальний досвід роботи з вебзастосунком [15].

3. Декларативний підхід до побудови інтерфейсу. У React розробник описує бажаний стан інтерфейсу, а фреймворк самостійно обробляє зміни, що робить код більш передбачуваним, стабільним та простішим для налагодження [19].

4. Синтаксис JSX. JSX є розширенням JavaScript, яке дозволяє писати HTML-подібний код безпосередньо в JavaScript-файлах. Це підвищує читабельність коду, полегшує створення структури компонентів і спрощує підтримку проєкту [15].

5. Сильна підтримка спільноти. React має широку та активну спільноту розробників, яка створює бібліотеки, інструменти та ділиться найкращими практиками. Це сприяє швидкому вирішенню проблем та підтримці актуальності технології [14].

6. SEO-орієнтованість. React може виконувати серверний рендеринг, що покращує продуктивність застосунків і сприяє кращій індексації контенту пошуковими системами. Це особливо важливо для вебресурсів, що залежать від видимості у пошукових результатах [19].

7. Ефективні інструменти розробки. React має розвинене середовище розробки з інструментами на кшталт React Developer Tools та Redux DevTools, які забезпечують зручне налагодження й аналіз стану застосунку, сприяючи підтримці високої якості коду [14].

8. Гнучкість і можливість інтеграції. React легко інтегрується з іншими бібліотеками та фреймворками, зокрема Redux (для керування станом) або Next.js (для серверного рендерингу), що дозволяє адаптувати технологічний стек до конкретних вимог проєкту [14].

Завдяки цим перевагам ReactJS широко застосовується для створення високопродуктивних, масштабованих і легких у підтримці вебзастосунків, що робить його одним з основних інструментів у сучасній фронтенд-розробці.

1.2.4. База даних

На етапі розробки та тестування інформаційної системи деканату як основну систему керування базами даних було обрано SQLite. Це рішення зумовлено її простотою інтеграції, мінімальними вимогами до налаштування середовища та можливістю зберігати всі дані в одному локальному файлі, що значно полегшує процес налагодження, тестування та початкової розробки. Оскільки SQLite не потребує окремого серверного компонента, вона дозволяє скоротити час на конфігурацію та зосередитись безпосередньо на функціональній реалізації програмного забезпечення.

SQLite — це легка, файлова система керування реляційними базами даних (RDBMS), яка широко використовується завдяки своїй простоті, зручності у використанні та відсутності потреби в конфігурації. Вона є безсерверною, автономною та не потребує попереднього налаштування, що робить її відмінним вибором для вбудованих систем, мобільних застосунків, а також для проєктів малого та середнього масштабу [13].

Водночас, у перспективі впровадження та постійної експлуатації системи передбачається перехід на повнофункціональну СУБД PostgreSQL, яка має вищу продуктивність, підтримку паралельних запитів, розширені механізми безпеки та кращу масштабованість. PostgreSQL також забезпечує ширші можливості для роботи з великими обсягами даних і складними транзакціями, що є критично важливим для стабільної та безпечної роботи інформаційної системи в реальному середовищі експлуатації. Вибір СУБД здійснюється з урахуванням різних етапів життєвого циклу програмного продукту та їх специфічних вимог.

Для забезпечення функціонування інформаційної системи було спроєктовано та реалізовано базу даних, яка включає низку сутностей з відповідними полями, призначених для структурованого зберігання й обробки інформації, пов’язаної з організацією освітнього процесу. Структура бази даних охоплює всі важливі аспекти академічної діяльності, що дозволяє ефективно моделювати бізнес-процеси закладу вищої освіти.

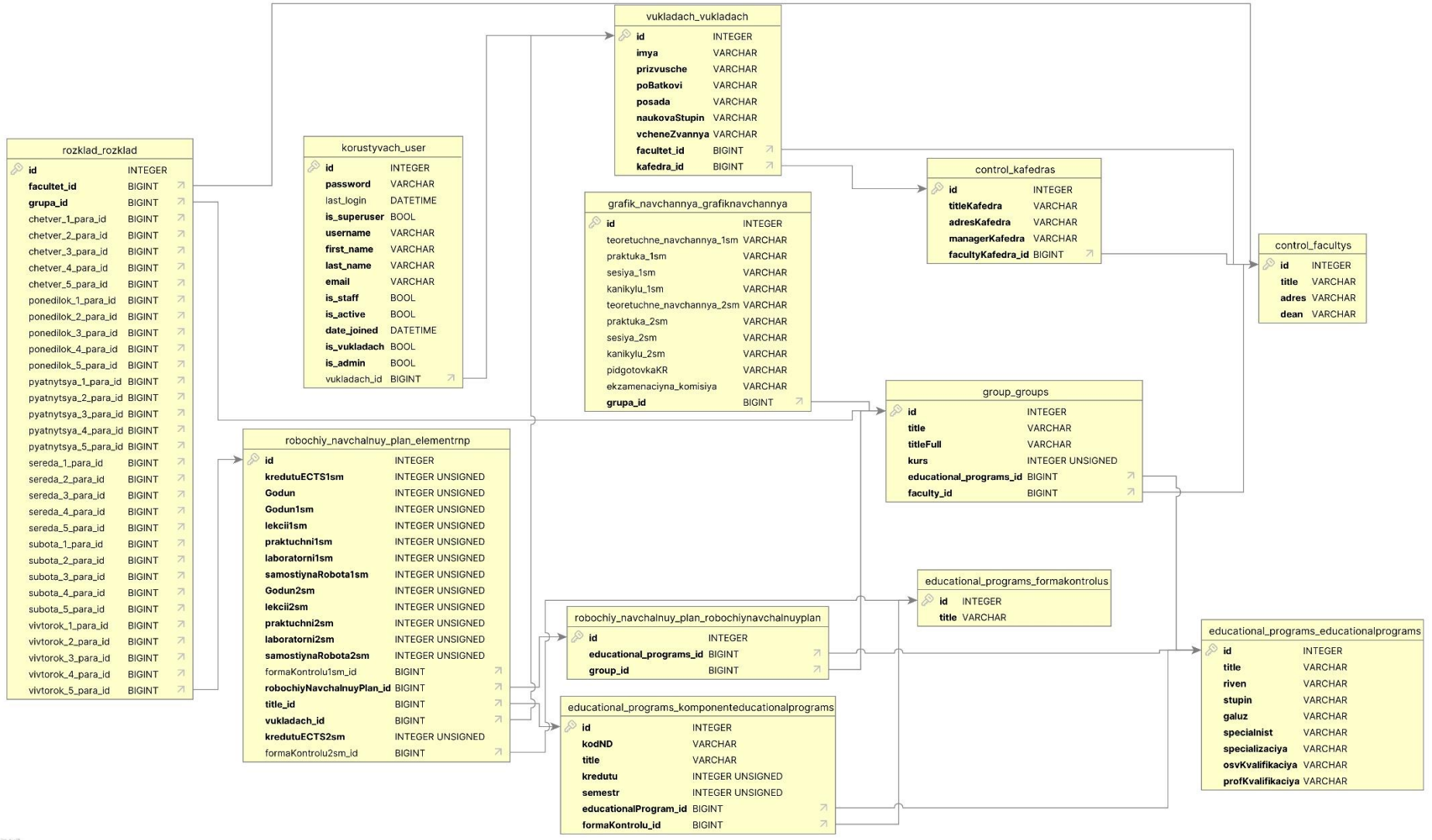

Рис. 1.4. Схема структури бази даних інформаційної системи

До складу бази даних входять наступні сутності:

– факультет;

– кафедра;

– група;

– викладач;

– освітня програма;

– компонент освітньої програми;

– робочий навчальний план;

– елемент робочого навчального плану;

– користувач;

– графік навчання;

– розклад;

– формат контролю [6, с. 45].

Кожна з перелічених сутностей має набір атрибутів, що відповідають її функціональному призначенню в системі, а також зв’язки з іншими сутностями, що забезпечують цілісність даних та логічну взаємодію між різними компонентами інформаційної системи.

1.2.5. Хостинг

Для забезпечення віддаленого доступу до інформаційної системи деканату, а також з метою проведення тестування та демонстрації функціональності, програмний продукт було розгорнуто на хмарному хостингу PythonAnywhere за посиланням «https://yalovenko.pythonanywhere.com/». Обрання цієї платформи обумовлено її високою сумісністю з технологічним стеком, що використовувався під час розробки системи, зокрема — з фреймворком Django та мовою програмування Python.

«PythonAnywhere— це хмарна платформа, яка дозволяє писати, запускати та розміщувати код на Python безпосередньо в браузері. Завдяки наявності безкоштовного тарифного плану, вона є привабливим варіантом для користувачів, які бажають ознайомитися з веброзробкою або автоматизацією без необхідності

керування серверами чи складними налаштуваннями. Незалежно від рівня підготовки — чи ви початківець, чи досвідчений програміст — PythonAnywhere надає низку інструментів, що дозволяють швидко розгортати проєкти» [20].

PythonAnywhere є вебсервісом, що надає інфраструктуру для розміщення, запуску та обслуговування вебзастосунків, написаних на Python, без необхідності налаштування окремого фізичного чи віртуального сервера. Це дозволяє значно скоротити витрати часу та ресурсів на розгортання системи, спростити її адміністрування та забезпечити доступність з будь-якої точки з доступом до Інтернету.

Основні можливості, які надає платформа PythonAnywhere:

1. Веборієнтоване середовище розробки. Користувачі можуть писати й запускати Python-код безпосередньо в браузері, без потреби встановлення Python на локальному комп’ютері.

2. Хостинг вебзастосунків. PythonAnywhere дозволяє розміщувати вебзастосунки на Python із підтримкою популярних фреймворків, таких як Django, Flask і web2py.

3. Серверний Python. На відміну від багатьох інших платформ, PythonAnywhere орієнтований переважно на виконання Python-коду на стороні сервера, що робить його ідеальним вибором для розробки серверних вебзастосунків, скриптів та інших бекенд-проєктів.

4. Підтримка баз даних. PythonAnywhere підтримує роботу з популярними СУБД, зокрема MySQL і PostgreSQL, що дає змогу розробляти повнофункціональні вебсистеми з надійним збереженням даних.

5. Файлове сховище. Користувачам надається певний обсяг дискового простору для зберігання коду, даних та іншої інформації, необхідної для роботи застосунків [22].

У процесі розгортання інформаційної системи було створено окреме віртуальне середовище, в якому були встановлені всі залежності, необхідні для коректної роботи Django-проєкту. Налаштовано маршрутизацію вебзастосунку,

здійснено міграцію бази даних та перевірено працездатність всіх основних функцій системи в онлайновому режимі.

Крім того, розміщення на PythonAnywhere дозволяє демонструвати систему замовникам, викладачам або адміністрації навчального закладу без необхідності встановлення її на локальні машини або внутрішні сервери. Це є особливо важливим на етапі презентації або пілотного впровадження системи в реальних умовах освітнього процесу.

Хоча платформа PythonAnywhere є зручним і функціональним середовищем для розробки, тестування та початкового розгортання вебзастосунків, її використання для повноцінної експлуатації інформаційної системи деканату є обмеженим. Безкоштовна версія сервісу має низку обмежень, щодо продуктивності, обсягу ресурсів, гнучкості налаштувань і можливості підключення власного доменного імені. Тому на етапі впровадження та стабільної експлуатації системи доцільним є перехід на платний хостинг з розширеними можливостями, що забезпечить вищу продуктивність, надійність, більший контроль над конфігурацією середовища виконання, а також дозволить використовувати власне доменне ім’я для публічного доступу до системи. Це посприяє підвищенню професійного рівня розгортання програмного забезпечення та відповідатиме вимогам до надійності й безперервності роботи інформаційної системи в реальних умовах експлуатації.

1.3 Аналіз існуючих рішень

Для глибшого розуміння функціональних можливостей автоматизованих систем, призначених для управління вищими навчальними закладами, було проведено порівняльний аналіз трьох поширених програмних продуктів: АСУ «Університет» від ТОВ «UNITEX+», пакет програмних систем від ПП «Політек-софт» та АСУ «ВНЗ», розроблена Науково-дослідним інститутом прикладних інформаційних технологій. Кожна з цих систем має унікальну структуру організації модулів, що визначає їхню ефективність та зручність використання [6, с.21].

АСУ «Університет» вирізняється комплексним підходом до автоматизації пешочергових аспектів навчального процесу. Основою системи є модуль «Структура ВНЗ», який моделює організаційну ієрархію закладу, включаючи інформацію про персонал та керівництво. Цей модуль задає стандарти термінології та оформлення звітності. Модуль «Навчальна частина» відповідає за облік навчальних дисциплін, їх розподіл між кафедрами та формування навчальних планів для різних спеціальностей і форм навчання. Він автоматично розраховує навантаження викладачів і студентів, а також генерує звітність у різних форматах. Модуль «Кафедра» автоматизує управління діяльністю кафедр, включаючи облік співробітників, розподіл навчального навантаження та формування відповідних звітів. Система підтримує створення розкладів, індивідуальних планів роботи та тематичних планів дисциплін, забезпечуючи зручний документообіг завдяки можливості конвертації звітів у популярні формати. Модуль «Навчальний розклад» автоматизує створення та облік розкладів, враховуючи аудиторний фонд та графік освітнього процесу. Він забезпечує автоматичне формування розкладу занять та його експорт у різні формати, а також візуалізацію на вебсайті закладу з можливістю фільтрації. Модуль «Абітурієнт» автоматизує роботу приймальної комісії, включаючи реєстрацію вступників, обробку їхніх даних, формування необхідної документації та проведення конкурсного відбору. Система забезпечує зберігання всієї інформації про абітурієнтів, автоматичне генерування документів, формування списків для конкурсного відбору та публікацію результатів [2].

Пакет програм «Деканат» від ПП «Політек-софт» пропонує низку модулів, орієнтованих на різні аспекти управління навчальним процесом. Модуль «Навчальний план» автоматизує створення, редагування та розрахунок навчальних планів, дозволяючи формувати звітність та вакансії для педагогічного навантаження. Система враховує години для різних видів навчальної діяльності для різних форм навчання. Модуль «Навчальний процес (Університет)» призначений для управління навчальною діяльністю, відображаючи структуру освітнього процесу, розраховуючи та розподіляючи навантаження між підрозділами та викладачами. Він формує різноманітну звітність, що полегшує управління та

оптимізацію навчального процесу. Модуль «П-Кафедра-Web» є веб-орієнтованим інструментом для кафедр, що дозволяє фіксувати та аналізувати виконання навчального навантаження без необхідності встановлення програмного забезпечення. Модуль «Розклад» призначений для формування розкладів занять на основі навчального навантаження та даних про аудиторний фонд, з можливістю друку розкладів та функцією автоматичного складання в тестовому режимі. Модуль «ПС-Студент-Web» є веб-інтерфейсом для деканатів, що забезпечує облік студентів та контроль їхньої успішності за різними системами оцінювання, а також формування різноманітних звітів про успішність. Модулі «ПС-Додаток до диплому-Web» та «ПС-Академ. довідка-Web» призначені для автоматичного друку відповідних документів на основі даних з модуля «ПС-Студент-Web» через веб-браузер [4].

АСУ «ВНЗ» від Науково-дослідного інституту прикладних інформаційних технологій зосереджена на комплексному управлінні інформацією про студентів, навчальні плани та персонал. Модуль «Студенти» забезпечує централізоване зберігання та управління всією інформацією про студентів у межах розширеної анкети, включаючи персональні дані, навчальний статус, інформацію про зарахування та успішність. Модуль «Контракти студентів» забезпечує управління фінансовими угодами зі студентами, включаючи відстеження оплат та термінів. Модуль «Сесії» автоматизує процеси підготовки екзаменаційних відомостей, ведення журналу, занесення оцінок та переведення студентів. Модуль «Навчальні плани» дозволяє створювати, редагувати та відстежувати навчальні плани, забезпечуючи доступ до розкладів та навчальних матеріалів. Модуль «Відділ кадрів» забезпечує управління персоналом закладу, включаючи облік працівників, їхню кваліфікацію та заробітну плату. Модуль «Освітньо-професійна програма підготовки (ОПП)» визначає стандарти навчання для різних освітньо-кваліфікаційних рівнів. Модуль «Дисципліни вільного вибору студентів (ДВВС)» призначений для управління дисциплінами, які студенти можуть обирати. Модуль «ІНП студента (Індивідуальний навчальний план)» забезпечує формування індивідуальних навчальних планів. Модулі «Академічні довідки студента» та

«Індивідуальні відомості» забезпечують облік та формування відповідних документів. Модуль «Створення замовлення додатків до дипломів» дозволяє замовляти друк додатків до дипломів. Модуль «Створення резервної копії» забезпечує збереження важливих даних системи [1].

Узагальнюючи, можна констатувати, що всі розглянуті системи мають свої сильні сторони, проте кожна з них акцентує увагу на різних аспектах управління та автоматизації університетських процесів. АСУ «Університет» відзначається широким набором функцій для організації навчального процесу та документообігу. Пакет програм від ПП «Політек-софт» пропонує зручну інтеграцію через веб-інтерфейси та ефективні інструменти для управління навчальним навантаженням та успішністю. АСУ «ВНЗ», у свою чергу, забезпечує глибоке управління інформацією про студентів та викладачів, а також кадровими процесами. Вибір оптимальної системи залежить від індивідуальних потреб та пріоритетів конкретного закладу вищої освіти. Якщо ключовою метою є всебічна автоматизація навчального процесу та управління навчальними планами, АСУ «Університет» може бути кращим варіантом. Для закладів, які цінують зручність веб-інтеграції та спрощення контролю за навчальним навантаженням, система від ПП «Політек-софт» надасть значні переваги. У випадку потреби в детальному аналізі студентських даних та ефективному управлінні кадровими процесами, АСУ «ВНЗ» завдяки своїй розвиненій функціональності в обліку студентів та викладачів є найбільш підходящим рішенням [6, с.30-31].

РОЗДІЛ 2
ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕКАНАТУ

2.1 Функціональне тестування системи

Функціональне тестування є різновидом тестування програмного забезпечення, що передбачає перевірку відповідності роботи системи визначеним вимогам і технічним специфікаціям. Основною метою цього процесу є забезпечення того, щоб програмний продукт коректно реалізовував усі передбачені функціональні вимоги. Тестування орієнтоване переважно на результат обробки, моделюючи реальні сценарії використання системи, без урахування її внутрішньої структури чи коду. В основу функціонального тестування покладено принцип перевірки кожної окремої функції програмного забезпечення на відповідність вимогам і специфікаціям. Тестування здійснюється шляхом подання відповідних вхідних даних, визначення очікуваних результатів та подальшого порівняння фактичного вихідного результату з очікуваним [5].

Тест 1. Перегляд розкладу, графіку начання та робочого навчального плану.

Вхідні дані: користувач заходить на сайт, вибирає потрібний факультет та відповідну групу. Натискає кнопки «Показати графік навчання», «Показати розклад», «Показати робочий навчальний план».

Очікуваний результат: відображення розкладу, графіку навчання та робочого навчального плану на сторінці сайту згідно тих параметрів які були обрані у списках вибору «Факультет» та «Група».

Отримані результати: відображення розкладу, графіку навчання та робочого навчального плану на сторінці сайту згідно тих параметрів які були обрані у списках вибору «Факультет» та «Група». Демонтрацію результатів можна побачити на рисунку 2.1 та 2.2.

Тест пройдено успішно.

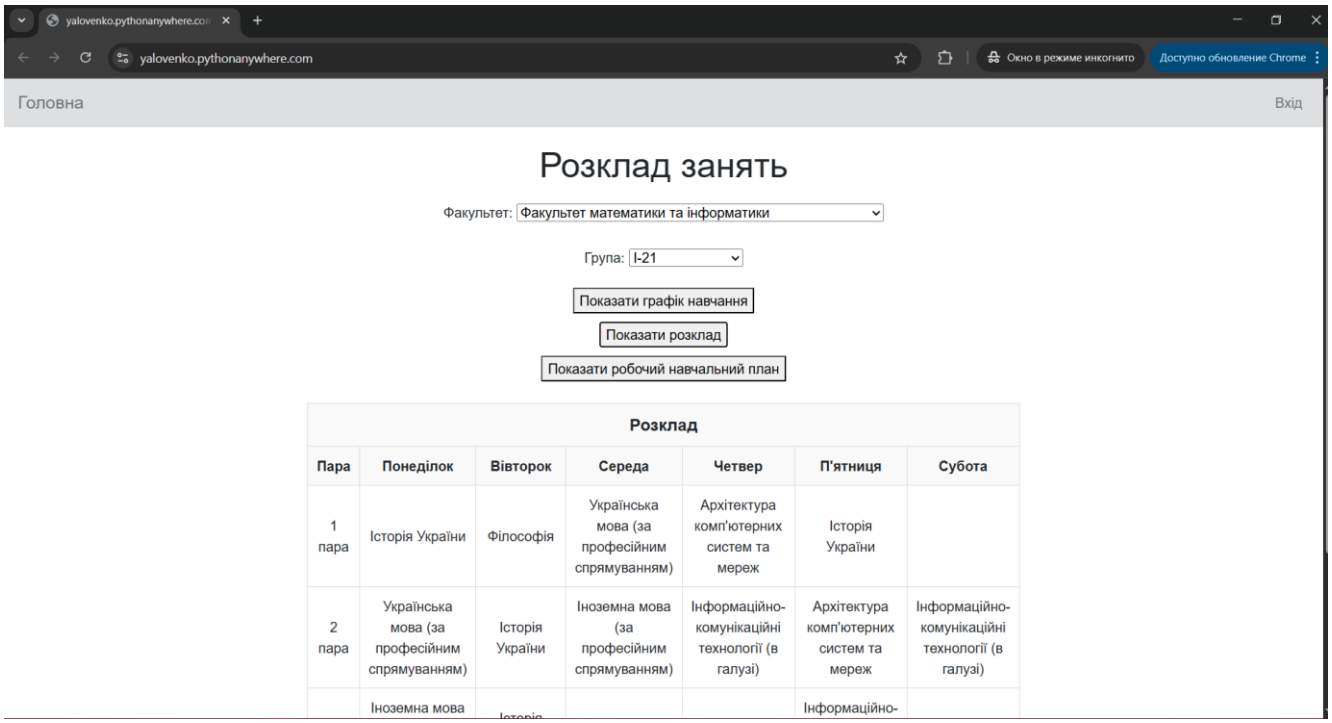

Рис. 2.1. Відображення розкладу на сторінці сайту

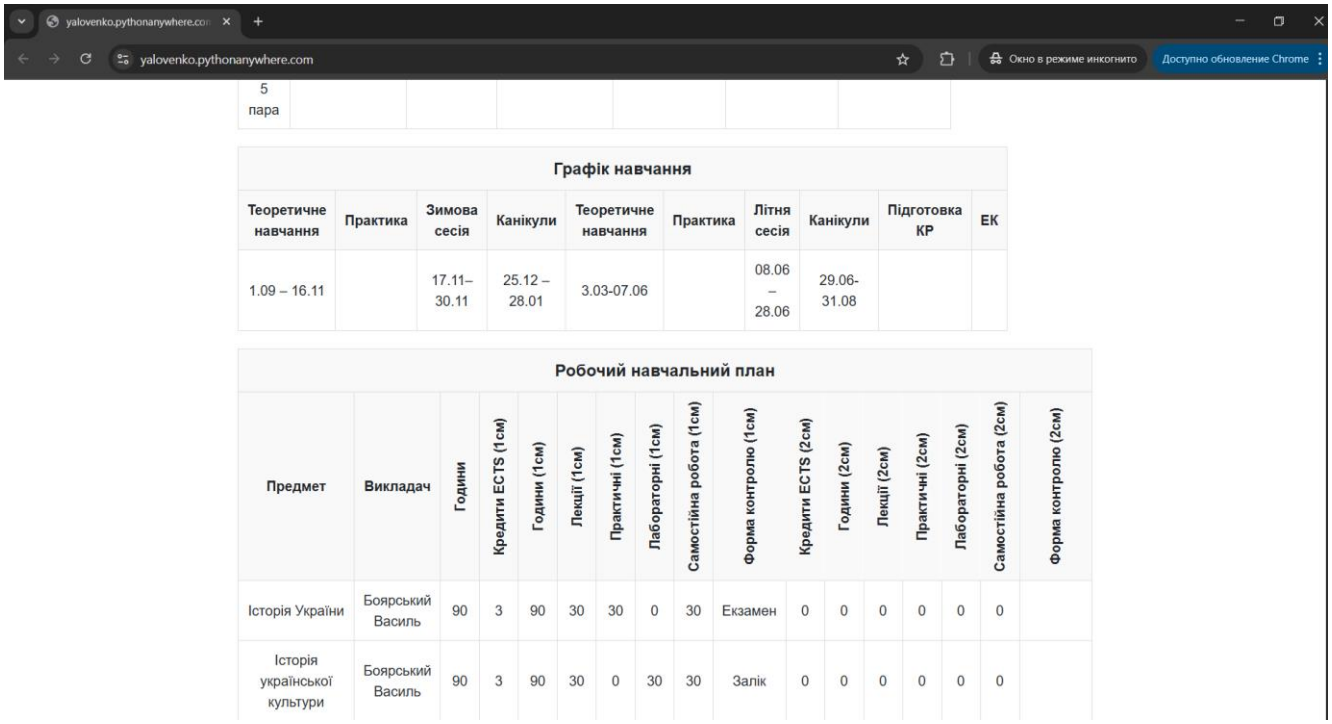

Рис. 2.2. Відображення графіку навчання та робочого навчального плану на сторінці сайту

Тест 2. Вхід в особистий кабінет та перегляд профілю викладача.

Вхідні дані: користувач вводить на сторінці авторизації дані авторизації «Логін та пароль» (Рис. 2.3).

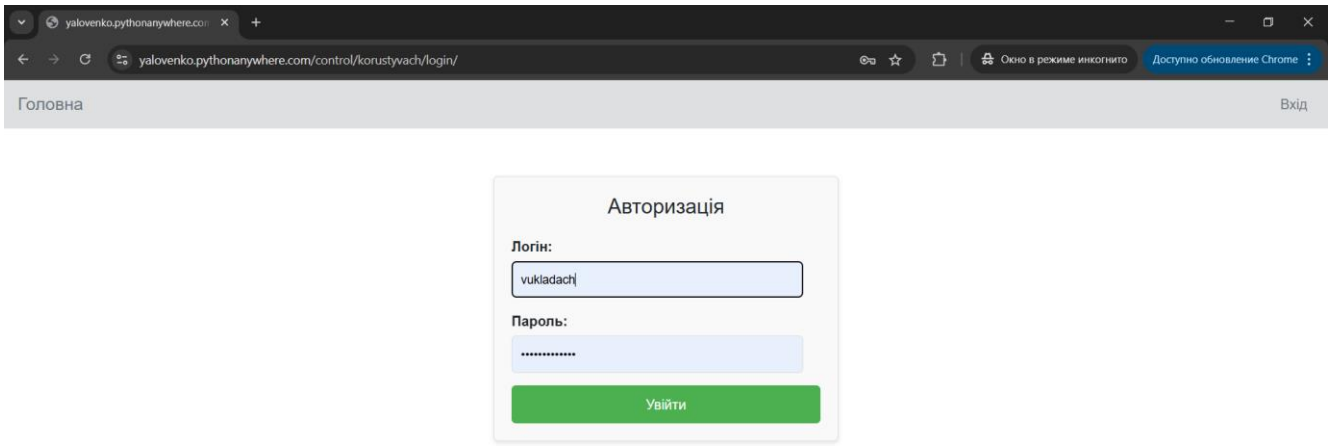

Рис. 2.3. Введення в форму авторизації логіну та паролю

Очікуваний результат: успішна авторизація та відображення сторінки особистого профілю викладача (Рис. 2.4).

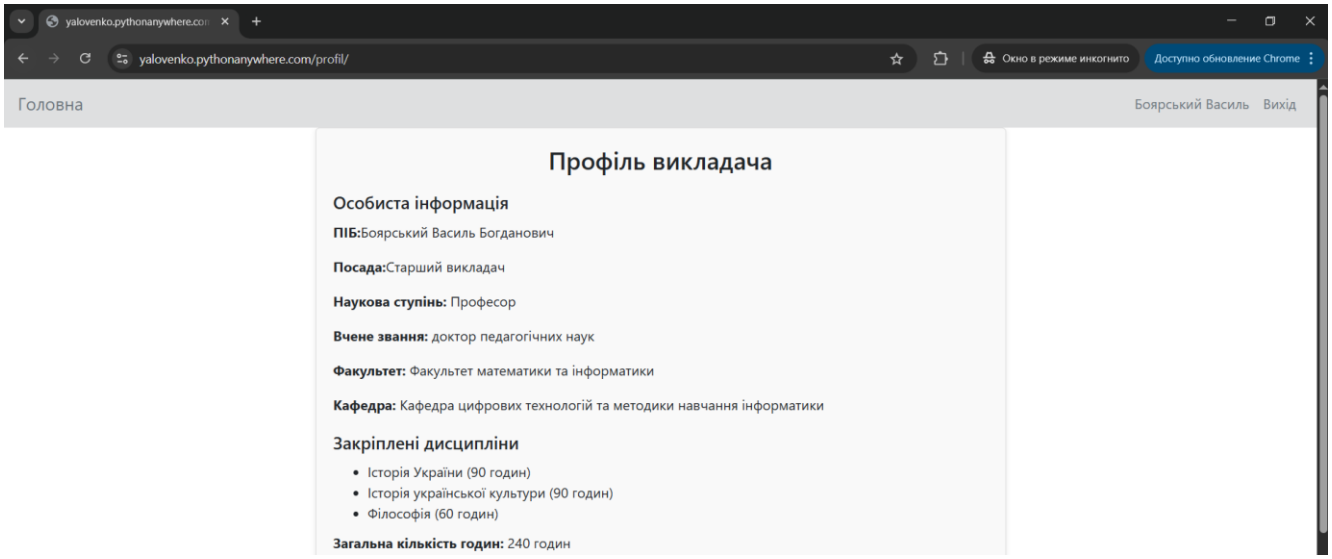

Рис. 2.4. Особистий профіль викладача

Отримані результати: успішна авторизація та відображення головної сторінки сайту.

Тест пройдено частково. Потрібно виправити переадресацію після успішної авторизації.

Тест 3. Створення графіку навчання групи.

Вхідні дані: авторизований користувач під профілем адміністратора натискає кнопку «Створити» на сторінці «Графіки навчання груп» та вводить потрібні дані на сторінці «Додати графік навчання» (Рис. 2.5) та натискає кнопку «Зберегти».

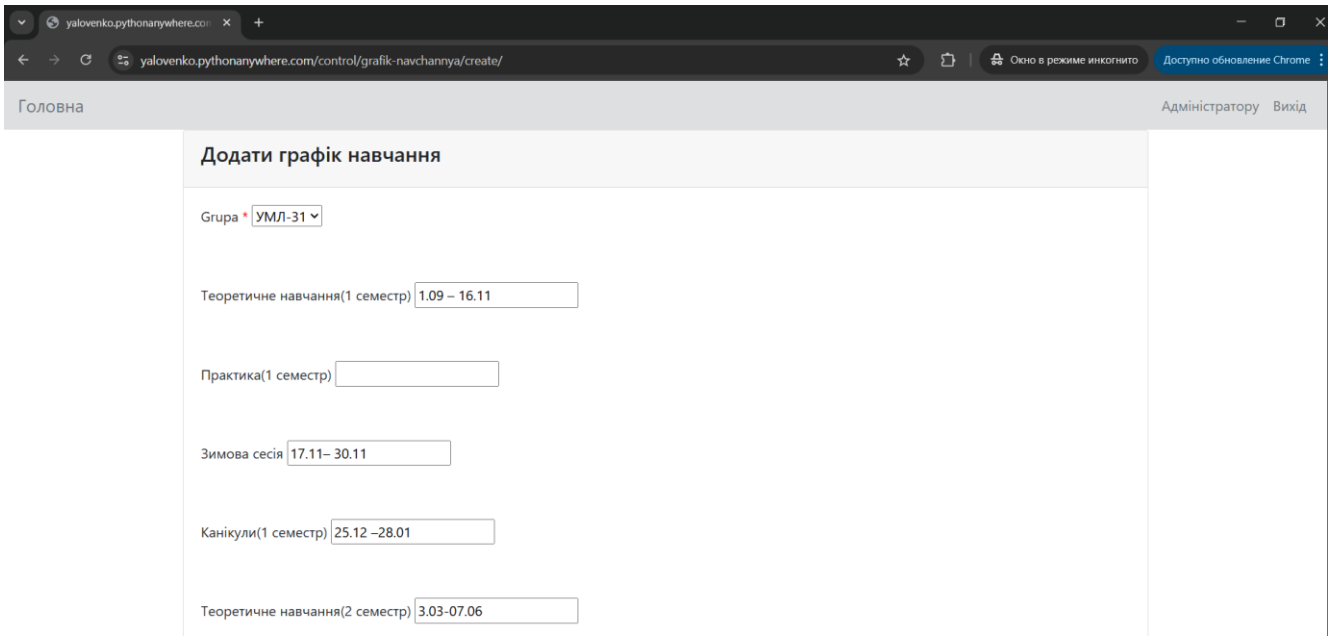

Рис. 2.5. Сторінка «Додати графік навчання»

Очікуваний результат: успішне створення графіку навчання та переадресація на сторінку «Графіки навчання груп».

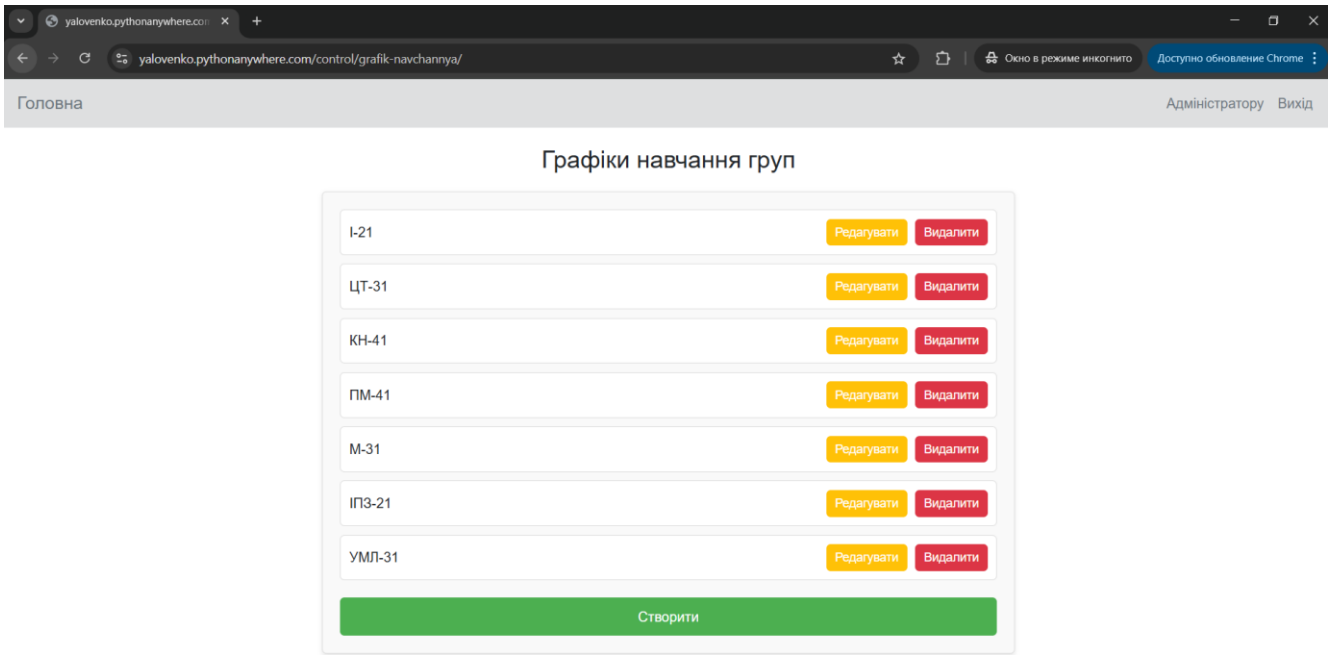

Рис. 2.6. Сторінка «Графіки навчання груп»

Отримані результати: відбулась переадресація на сторінку «Графіки навчання груп» (Рис. 2.6), створений графік відображається у списку.

Тест пройдено успішно.

Тест 4. Редагування графіку навчання групи.

Вхідні дані: авторизований користувач під профілем адміністратора натискає кнопку «Редагувати» на сторінці «Графіки навчання груп» (Рис. 2.6) та редагує потрібні дані на сторінці «Додати графік навчання» (Рис. 2.5) та натискає кнопку «Зберегти».

Очікуваний результат: успішне редагування графіку навчання та переадресація на сторінку «Графіки навчання груп».

Отримані результати: відбулась переадресація на сторінку «Графіки навчання груп», зміни у відредагованому графіку збереглись.

Тест пройдено успішно.

Тест 5. Видалення графіку навчання групи.

Вхідні дані: авторизований користувач під профілем адміністратора натискає кнопку «Видалити» на сторінці «Графіки навчання груп» (Рис. 2.6) та підтверджує видалення на сторінці «Видалення графіку навчання» (Рис. 2.7) натиснувши кнопку «Видалити графік навчання».

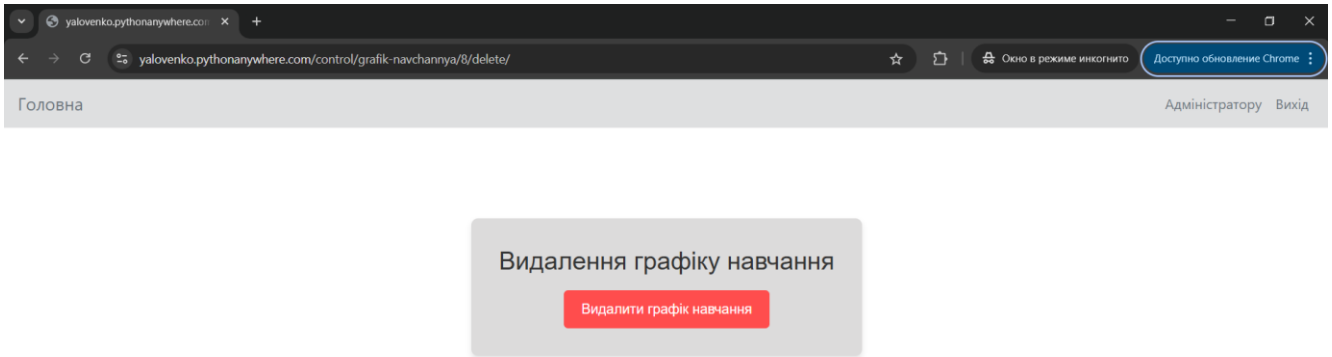

Рис. 2.7. Сторінка «Видалення графіку навчання»

Очікуваний результат: успішне видалення графіку навчання та переадресація на сторінку «Графіки навчання груп».

Отримані результати: відбулась переадресація на сторінку «Графіки навчання груп», видалений графік навчання у списку відсутній.

Тест пройдено успішно. Виявлено недолік на сторінці «Видалення графіку навчання» (Рис. 2.7), відсутня кнопка «Скасувати». Потрібно додати кнопку «Скасувати» та реалізувати її функціонал.

2.2 Unit-тестуваня

Модульне тестування — це метод тестування програмного забезпечення, який передбачає перевірку окремих модулів або компонентів програмного застосунку в ізоляції від решти системи. Метою модульного тестування є переконатися, що кожен модуль або компонент працює відповідно до очікувань, а також виявити та усунути помилки на ранніх етапах розробки [10].

Модульне тестування відіграє важливу роль у забезпеченні стабільності та надійності програмного забезпечення, оскільки дозволяє розробникам перевірити логіку окремих функцій, класів або методів без впливу зовнішніх залежностей. Такий підхід дає змогу швидко локалізувати помилки, спрощує процес рефакторингу коду та підвищує впевненість у правильності реалованої функціональності.

Також модульне тестування мотивує програмістів писати код максимально оптимізованим, проводити рефакторинг (спрощення коду програми, не зачіпаючи її функціональність), так як за допомогою Юніт-тестування можна легко перевірити працездатність окремого компонента.

Важливо відзначити, що модульне тестування зазвичай відбувається на початку процесу розробки як проактивний захід або перед впровадженням нового коду в існуючу систему. Включення модульного тестування програмного забезпечення у ваш існуючий проект може принести, як позитивні так і негативні аспекти.

Позитивні:

- Економія часу та грошей;
- Покращує якість;

– Надає документацію;

– Підвищує загальну ефективність.

Негативні:

– Потрібно більше коду;

– Не є універсальним інструментом;

– Ускладнює зміни.

У середовищі Django модульне тестування зазвичай реалізується за допомогою вбудованого модуля unittest або сторонніх бібліотек, таких як pytest. Django надає зручні інструменти для створення тестових випадків, включно з можливістю імітації HTTP-запитів, роботи з тестовою базою даних та перевірки поведінки окремих компонентів, зокрема моделей, представлень (views), форм та серіалізаторів.

Приклад unit-тестування наведено на рисунку 2.8.

```python
from django.test import TestCase
from .models import Facultys, Kafedras

# Create your tests here.

class TestControl(TestCase):

    def test_control_home(self):
        response=self.client.get('/control/')
        self.assertEqual(response.status_code, 200)

    def setUp(self):

        self.faculty = Facultys.objects.create(
            title='Факультет інформаційних технологій',
            adres='м. Київ, вул. Академіка 1',
            dean='Іван Петренко'
        )

    def test_facultys_str(self):

        self.assertEqual(str(self.faculty), 'Факультет інформаційних технологій')

    def test_create_kafedra(self):

        kafedra = Kafedras.objects.create(
            titleKafedra='Кафедра програмування',
            adresKafedra='м. Київ, вул. Академіка 2',
            managerKafedra='Олена Іванова',
            facultyKafedra=self.faculty
        )
        self.assertEqual(Kafedras.objects.count(), 1)
        self.assertEqual(kafedra.facultyKafedra.title, 'Факультет інформаційних технологій')
```

Рис. 2.8. Приклад Unit-тестування

2.3 Integration-тестування

Інтеграційне тестування в розробці програмного забезпечення перевіряє, чи правильно взаємодіють різні частини програми. Воно гарантує, що один фрагмент коду «розуміє» інший і правильно обмінюється з ним даними. Завдяки цьому вдається виявити й усунути помилки на ранньому етапі, що робить програму кращою та надійнішою. Інтеграційне тестування можна порівняти з перевіркою того, як шестерні в механізмі з'єднуються між собою, щоб уся система працювала без збоїв. Це важливий етап, який допомагає забезпечити коректну роботу програм, якими ми користуємось щодня [11].

Інтеграційне тестування може допомогти командам розробників завчасно виявляти та виправляти проблеми та максимізувати продуктивність додатків. Виконання інтеграційного тестування відразу після модульного тестування модулів програмного забезпечення має низку переваг:

- Визначення проблеми інтеграції між модулями
- Більш комплексні, ніж модульні тести
- Завчасне вирішення помилок
- Вдосконалення тестового покриття та надійності

Інтеграційне тестування є важливим кроком для більшості команд розробників, але це не означає, що воно на 100% ідеальне. Це складний процес, який може зайняти багато часу, а це означає, що важливо ретельно планувати та координувати інтеграційне тестування. Тестування інтеграції може поставити перед командами програмного забезпечення багато проблем:

- Інтеграційне тестування є ресурсомістким
- Складне у виконанні
- Інтеграційне тестування вимагає часу
- Усунення проблем, які виникають під час тестування

Інтеграційне тестування особливо корисне тоді, коли в застосунку є багато пов'язаних між собою компонентів, як-от моделі, представлення (views), форми та

URL-маршрути. Це дає змогу переконатися, що вони не лише працюють окремо, а й коректно взаємодіють між собою в межах загальної логіки застосунку. Наприклад, коли користувач надсилає форму для створення нового запису, тест повинен перевірити, що дані успішно обробляються у view, зберігаються в базу через модель і після цього користувач перенаправляється на потрібну сторінку.

Для прикладу, у рамках проєкту інформаційної системи , яка включає моделі Facultys та Kafedras, інтеграційне тестування дозволяє перевірити повний цикл: створення факультету через форму, відображення списку кафедр, оновлення запису або видалення. На рисунку 2.9 наведено приклади таких тестів, що імітують реальні дії користувача та перевіряють, як система реагує на них.

```python
class ControlIntegrationTests(TestCase):

    def setUp(self):
        self.client = Client()
        self.faculty = Facultys.objects.create(
            title='Факультет філології',
            adres='вул. Літературна, 10',
            dean='Наталя Левченко'
        )

    def test_facultys_list_view(self):
        response = self.client.get(reverse('facultys_list'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'control/faculty_list.html')
        self.assertContains(response, 'Факультет філології')

    def test_facultys_create_view(self):
        response = self.client.post(reverse('facultys_create'), {
            'title': 'Факультет фізики',
            'adres': 'вул. Наукова, 5',
            'dean': 'Андрій Коваленко'
        })
        self.assertEqual(response.status_code, 302)
        self.assertTrue(Facultys.objects.filter(title='Факультет фізики').exists())

    def test_facultys_update_view(self):
        response = self.client.post(reverse('facultys_update', args=[self.faculty.id]), {
            'title': 'Факультет української філології',
            'adres': 'вул. Літературна, 10',
            'dean': 'Наталя Левченко'
        })
        self.assertEqual(response.status_code, 302)
        self.faculty.refresh_from_db()
        self.assertEqual(self.faculty.title, 'Факультет української філології')

    def test_kafedras_create_view(self):
        response = self.client.post(reverse('kafedras_create'), {
```

Рис. 2.9. Приклад Integration –тестування

РОЗДІЛ 3
БЕЗПЕКА ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕКАНАТУ

3.1 Двофакторна автентифікація (2FA)

У цифрову епоху, коли кількість витоків даних і кіберзагроз невпинно зростає, захист конфіденційної інформації набуває особливого значення. Двофакторна автентифікація (2FA) виступає одним із важливих засобів у системі заходів кібербезпеки. Цей механізм безпеки передбачає використання двох різних факторів підтвердження особи перед отриманням доступу до системи або облікового запису. Це забезпечує додатковий рівень захисту, що перевищує звичну комбінацію імені користувача та пароля.

Двофакторна автентифікація (2FA) є процедурою забезпечення безпеки, яка вимагає від користувача надання двох різних типів ідентифікаційних даних перед отриманням доступу до облікового запису або інформаційної системи. Вона поєднує щось, що знає користувач (наприклад, пароль), із тим, що він має (наприклад, смартфон або апаратний токен), або з тим, чим він є (наприклад, відбиток пальця чи розпізнавання обличчя). Такий дворівневий підхід суттєво підвищує рівень безпеки, оскільки навіть у разі компрометації одного з факторів імовірність несанкціонованого доступу значно знижується [17].

В рамках підвищення рівня інформаційної безпеки та модернізації інформаційної системи деканату було прийнято рішення реалізувати механізм двофакторної автентифікації, що ґрунтується на застосуванні зовнішнього сервісу Twilio та бібліотеки pyotp мови програмування Python. Таке поєднання програмних засобів забезпечить можливість генерування одноразових кодів та їх передавання користувачам через SMS на мобільний телефон, що, у свою чергу, підвищує захищеність процесу автентифікації від потенційних загроз.

Сервіс Twilio надає зручний хмарний API та мову розмітки, що дозволяють підприємствам швидко створювати масштабовані, надійні та функціонально розвинені додатки для голосового зв’язку та обміну SMS-повідомленнями. API для

роботи з SMS дає змогу надсилати й отримувати текстові повідомлення безпосередньо з програмних застосунків. Крім того, він підтримує використання технології коротких кодів, що забезпечує вищу швидкість та більший обсяг надсилання повідомлень. Серед типових сценаріїв використання API можна виокремити автоматичне надсилання SMS клієнтам, реалізацію багатофакторної автентифікації та проведення інтерактивного опитування. API реалізовано за принципами REST-архітектури, а відповіді формуються у форматі XML [9].

PyOTP — це бібліотека Python для генерації та перевірки одноразових паролів. Її можна використовувати для реалізації методів двофакторної (2FA) або багатофакторної (MFA) автентифікації у вебзастосунках та інших системах, які вимагають входу користувачів у систему. Відкриті стандарти MFA визначені в RFC 4226 (HOTP: алгоритм одноразового пароля на основі HMAC) та в RFC 6238 (TOTP: алгоритм одноразового пароля на основі часу). PyOTP реалізує підтримку на стороні сервера для обох цих стандартів. Підтримку на стороні клієнта можна ввімкнути, надсилаючи коди автентифікації користувачам через SMS або електронну пошту (HOTP), або, для TOTP, доручивши користувачам використовувати Google Authenticator , Authy або іншу сумісну програму [21].

Для реалізації механізму двофакторної автентифікації в інформаційній системі деканату використано поєднання бібліотеки pyotp, що забезпечує генерацію одноразових паролів, та сервісу Twilio, який використовується для передавання згенерованих кодів користувачеві за допомогою SMS.

Функція korustyvach_login() (Рис. 3.1) відповідає за початкову автентифікацію користувача за логіном і паролем. У разі успішної перевірки облікових даних вона ініціює процес двофакторної автентифікації шляхом виклику функції send_otp(), а також зберігає ім’я користувача в сесії для подальшої перевірки одноразового пароля.

```python
def korustyvach_login(request):
    error_message=None
    if request.POST:
        username = request.POST['username']
        password = request.POST['password']
        user=authenticate(request, username=username, password=password)
        if user is not None:
            send_otp(request)
            #login(request, user=user)
            request.session['username'] = username
            logger.info(f"User {username} login")
            return redirect('otp_korustyvach')
        else:
            error_message='Невірний логін або пароль'

    return render(request, 'korustyvach/login.html', {'error_message': error_message})
```

Рис. 3.1. Функція korustyvach_login()

Функція korustyvach_otp() (Рис. 3.2) обробляє введення одноразового пароля, отриманого користувачем. Вона перевіряє дійсність пароля з урахуванням часу його придатності та здійснює завершальну частину автентифікації — вхід до системи із застосуванням механізму авторизації користувача у разі успішної верифікації коду.

```python
def korustyvach_otp(request):
    error_message=None
    if request.POST:
        otp=request.POST['otp']
        username=request.session['username']
        otp_secret_key = request.session['otp_secret_key']
        otp_valid_date = request.session['otp_valid_date']
        if otp_secret_key and otp_valid_date is not None:

            valid_date = datetime.fromisoformat(otp_valid_date)
            if valid_date > datetime.now():
                totp = pyotp.TOTP(otp_secret_key, interval=60)
                if totp.verify(otp):
                    user=get_object_or_404(User, username=username)
                    login(request, user)
                    logger.info(f"User {username} otp")
                    del request.session['otp_secret_key']
                    del request.session['otp_valid_date']
                    return redirect('home')
                else:
                    error_message = 'Невірний пароль'
                    print(error_message)
            else:
                error_message = 'Час дії паролю закінчився'
                print(error_message)
        else:
            error_message = 'Упс... Щось пішло не так'
            print(error_message)
    return render(request, 'korustyvach/otp.html', {'error_message': error_message})
```

Рис. 3.2. Функція korustyvach_otp()

Функція send_otp() (Рис. 3.3) реалізує процес генерації одноразового пароля з використанням алгоритму TOTP (Time-based One-Time Password). Генерація виконується на основі випадкового секретного ключа, створеного методом pyotp.random_base32(). Одноразовий пароль діє впродовж 60 секунд, що відповідає параметру interval. Згенерований секретний ключ та строк дії пароля зберігаються в сесії користувача для подальшої перевірки.

```python
import pyotp
from datetime import datetime, timedelta
from twilio.rest import Client

def send_otp(request):
    totp= pyotp.TOTP(pyotp.random_base32(), interval=60)
    otp=totp.now()
    request.session['otp_secret_key'] = totp.secret
    valid_date=datetime.now() + timedelta(minutes=1)
    request.session['otp_valid_date'] = str(valid_date)

    send_sms(otp)
    print(f"Пароль otp: {otp}")
```

Рис. 3.3. Функція send_otp()

Функція send_sms() (Рис. 3.4) відповідає за безпосередню передачу одноразового пароля користувачу через сервіс Twilio. Для цього створюється об’єкт клієнта Client з використанням облікових даних — ідентифікатора облікового запису (account_sid) та токена авторизації (auth_token). Після цього за допомогою методу messages.create() відбувається відправлення SMS (Рис. 3.5) з вмістом повідомлення (паролем) на вказаний номер телефону користувача. Відправка здійснюється з номера, наданого сервісом Twilio.

```python
def send_sms(parol):
    account_sid = 'AC61dc631b56e208334d3aca80012ae552'
    auth_token = '890feb93ceec725d1d79589dd824d849'
    client = Client(account_sid, auth_token)

    mesege= parol

    msg = client.messages.create(from_="+16187268706", body=mesege, to="+380980367814")

    print(msg.body)
```

Рис. 3.4. Функція send_otp()

Ця функція (Рис. 3.4) демонструє реалізацію використання REST API сервісу Twilio для інтеграції SMS-обміну з програмними компонентами автентифікації.

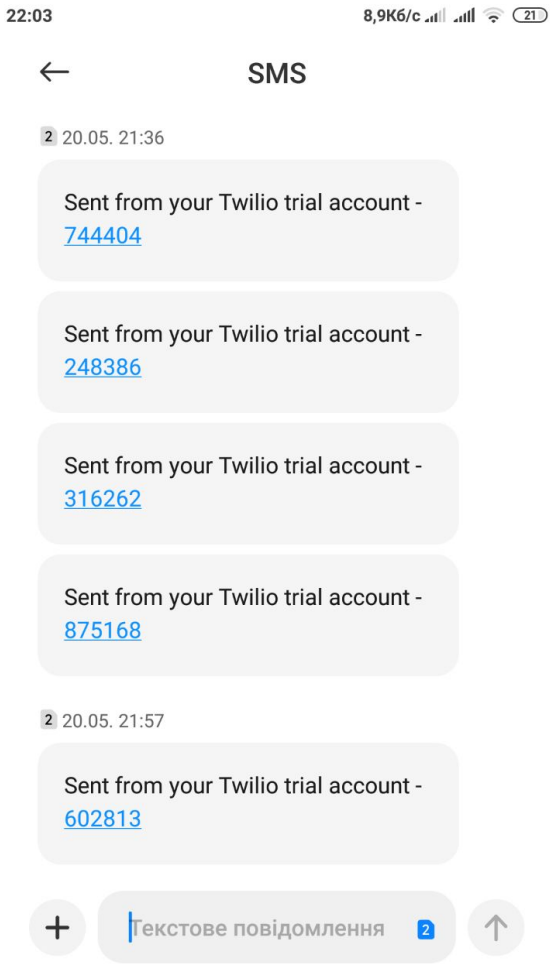

Рис. 3.5. SMS з паролем для двофакторної автентифікації

Розглянутий код реалізує повноцінний механізм двофакторної автентифікації шляхом поєднання генерації одноразових кодів з їх надсиланням через зовнішній сервіс, що значно підвищує рівень захисту інформаційної системи деканату від несанкціонованого доступу.

3.2 Реалізація логування як засіб забезпечення безпеки системи

Ще одним етапом підвищення безпеки даних та удосконалення інформаційної системи деканату стало реалізація логування. Журнал подій

створюється під час виникнення будь-якої дискретної події в програмній системі. Прикладами таких подій є вхід користувача в систему, виконання певних дій створення, редагування та видалення інформації або ж просто перегляду її. Події мають бути відстежуваними, щоб у разі збою, порушення логіки виконання або несанкціонованих дій можна було вжити відповідних заходів. Одним із засобів забезпечення спостережуваності програмних систем є ведення журналу подій (логування). Завдяки журналам можна відтворити хід подій у середовищі розробки, здійснити діагностику помилок та ввести відповідні заходи.

Журнали (логи) — це записи подій, що відбуваються під час виконання програмного забезпечення. Вони містять відомості про застосунок, продуктивність системи або дії користувачів. Об’єкти, з якими взаємодіє розробник для виведення інформації у журнал, називаються логерами. Логери дозволяють визначити, яку саме інформацію слід фіксувати та яким чином це має здійснюватися [23].

Бібліотека logging у мові Python додає до журналів кілька типів метаданих, що надають корисний контекст до повідомлень журналу. Це сприяє ефективній діагностиці проблем або аналізу внутрішніх процесів у коді під час виконання застосунку. Наприклад, рівні журналювання визначають ступінь важливості подій, що фіксуються, і дозволяють класифікувати записи, щоб у певний момент отримати найактуальнішу інформацію.

Логер є точкою входу до системи журналювання. Кожен логер представляє собою іменований контейнер, до якого можуть записуватися повідомлення для подальшої обробки. Логер налаштовується на певний рівень журналювання, що визначає серйозність повідомлень, які він буде обробляти. У Python передбачено такі рівні журналювання:

– DEBUG — низькорівнева системна інформація, що використовується для налагодження;
– INFO — загальна інформація про стан системи;
– WARNING — повідомлення про незначну проблему, яка виникла;

– ERROR — повідомлення про серйозну помилку, що сталася;

– CRITICAL — повідомлення про критичну помилку, яка впливає на роботу системи [16].

Для забезпечення ефективного моніторингу роботи інформаційної системи деканату реалізовано систему журналювання на основі вбудованих засобів фреймворку Django. Основою конфігурації є словник LOGGING, який відповідає вимогам стандарту журналювання версії 1 Python.

Журналювання реалізується з використанням двох основних обробників: console та file. Обробник console виводить повідомлення рівня DEBUG у стандартний потік виводу, що зручно під час розробки. Натомість обробник file фіксує повідомлення рівня INFO і вище до зовнішнього журналу django.log, розташованого у кореневій директорії проєкту. Таким чином, забезпечується збереження найважливіших повідомлень у файловій системі для подальшого аналізу.

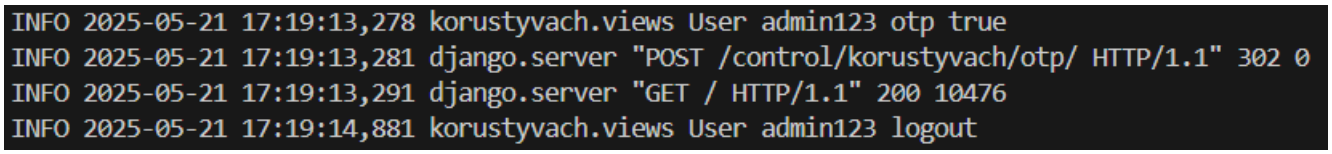

Рис. 3.6. Приклад логів формату simple

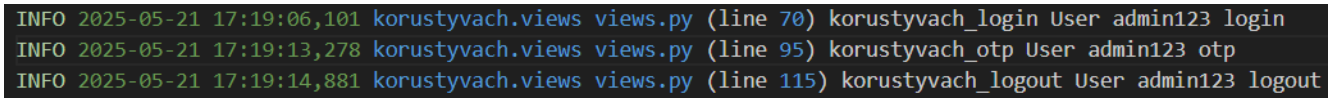

Рис. 3.7. Приклад логів формату verbose

Для форматування повідомлень використовуються два шаблони: simple та verbose. Формат simple забезпечує короткий запис рівня важливості, часу події, імені логера та самого повідомлення. Формат verbose, у свою чергу, надає розширений контекст, включаючи модуль, рядок коду та назву функції, що полегшує трасування помилок.

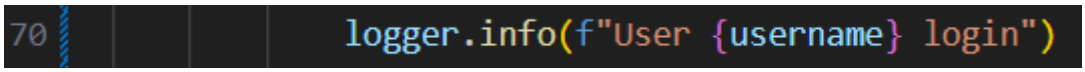

Рис. 3.8. Функцяя запису логу входу користувача в систему

Рис. 3.9. Функцяя запису логу виходу користувача з системи

Для підвищення рівня безпеки інформаційної системи деканату впроваджується механізм журналювання (логування) дій користувачів. Це дозволяє здійснювати контроль над усіма основними діями, що виконуються в системі, зокрема фіксуються факти входу (Рис. 3.8) та виходу (Рис. 3.9) користувачів, а також дії, пов’язані зі створенням, редагуванням і видаленням даних. Ведення журналу подій забезпечує можливість оперативного реагування на інциденти, пов’язані з несанкціонованим доступом або порушенням правил використання системи.

РОЗДІЛ 4
ПРОПОЗИЦІЇ ЩОДО МОДЕРНІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЕКАНАТУ ТА ВПРОВАДЖЕННЯ ПОКРАЩЕНЬ

4.1 Пропозиції модернізації та покращення інформаційної системи деканату

У процесі взаємодії користувача з інформаційною системою деканату першим елементом інтерфейсу, з яким він стикається, є головна сторінка (Рис. 4.1). Відповідно, її зовнішній вигляд значною мірою впливає на загальне сприйняття системи, зручність навігації та комфорт роботи. У наявній версії системи спостерігається відсутність належної стилізації інтерфейсних елементів, що ускладнює орієнтування в доступному функціоналі та може створювати враження застарілості чи недопрацьованості.

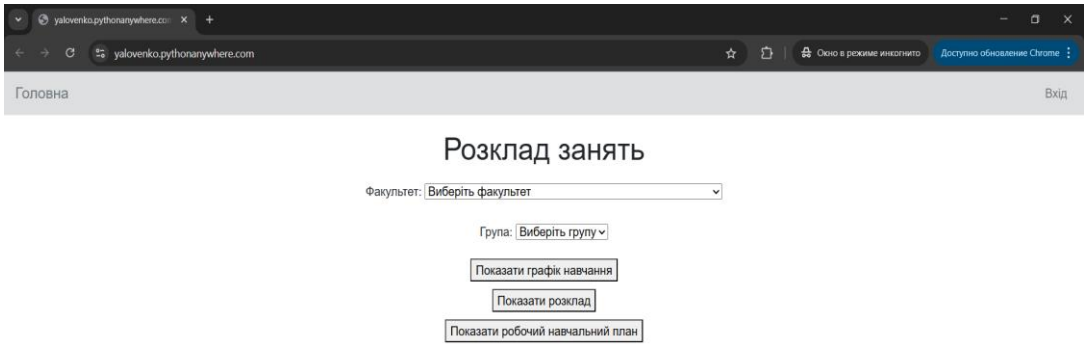

Рис. 4.1. Головна сторінка інформаційної системи деканату

З метою підвищення ергономічності та естетичної привабливості інтерфейсу пропонується здійснити стилізацію головної сторінки, застосувавши сучасні підходи до вебдизайну. Зокрема, доцільно реалізувати єдиний візуальний стиль для основних елементів інтерфейсу: кнопок, таблиць, списків вибору, гіперпосилань тощо. Для цього рекомендується використовувати CSS-фреймворки або бібліотеки

компонентів (наприклад, Bootstrap, Tailwind CSS), які дозволяють забезпечити візуальну цілісність інтерфейсу, адаптивність до різних типів пристроїв та зменшити час розробки.

Ще одним важливим напрямом удосконалення інформаційної системи деканату є розширення обсягу представленої інформації на головній сторінці з метою підвищення її інформативності для студентів. На сьогодні головна сторінка системи містить обмежений набір функціональних блоків, що ускладнює доступ до актуальних навчальних матеріалів та організаційної інформації, особливо в період сесії чи підготовки до неї.

З метою покращення інформативного наповнення пропонується створити на головній сторінці окрему функціональну зону під умовною назвою «Додаткова інформація». У межах цієї зони доцільно згрупувати інформаційні блоки, які містять:

– Розклад екзаменів — таблиця, яка містить дати, дисципліни та викладачів;

– Розклад заліків — подібна таблиця до розкладу екзаменів;

– Розклад дзвінків — таблиця, яка містить дані про час початку і закінчення кожної пари та перерви в кожній із змін навчання;

– Інформація про роботу екзаменаційної комісії — таблиця, яка включає дату засідання та склад комісії;

– Робочі навчальні плани — таблиця, яка містить дані про предмети, які вивчає певна група;

– Графік навчання — таблиця з інформацією про графік навчання на цілий рік.

Зазначене структурування дозволить розвантажити головну сторінку від надмірної кількості елементів першого рівня, водночас забезпечивши студентам швидкий доступ до найбільш затребуваних ресурсів у зручному форматі. Крім того, зонування дозволяє логічно згрупувати схожі за змістом та призначенням

елементи, що відповідає сучасним підходам до організації інтерфейсів інформаційних систем.

Інформація у блоці «Додаткова інформація» має бути представлена у форматі компактних таблиць. Це забезпечить не лише зручність використання, але й підвищить ефективність сприйняття великого обсягу даних. Динамічне оновлення таких блоків відповідальними особами (наприклад,) дозволить підтримувати актуальність інформації без залучення користувача до додаткових дій.

Упровадження описаних змін сприятиме підвищенню прозорості навчального процесу, зменшенню кількості звернень до адміністрації з уточнюючими питаннями, а також забезпеченню цілісного інформування студентів про важливі етапи освітнього процесу.

Одним із найбільш об’ємних та критично важливих напрямів удосконалення інформаційної системи деканату є модернізація модуля управління розкладом. На поточному етапі реалізації функціоналу система дозволяє створювати та редагувати розклад (Рис. 4.2) лише для окремої академічної групи, що істотно обмежує ефективність планування навчального процесу на рівні всього факультету. Такий підхід ускладнює координацію між різними групами та підвищує ризики накладок — як у використанні аудиторного фонду, так і у плануванні дисциплін, що викладаються одним і тим самим викладачем.

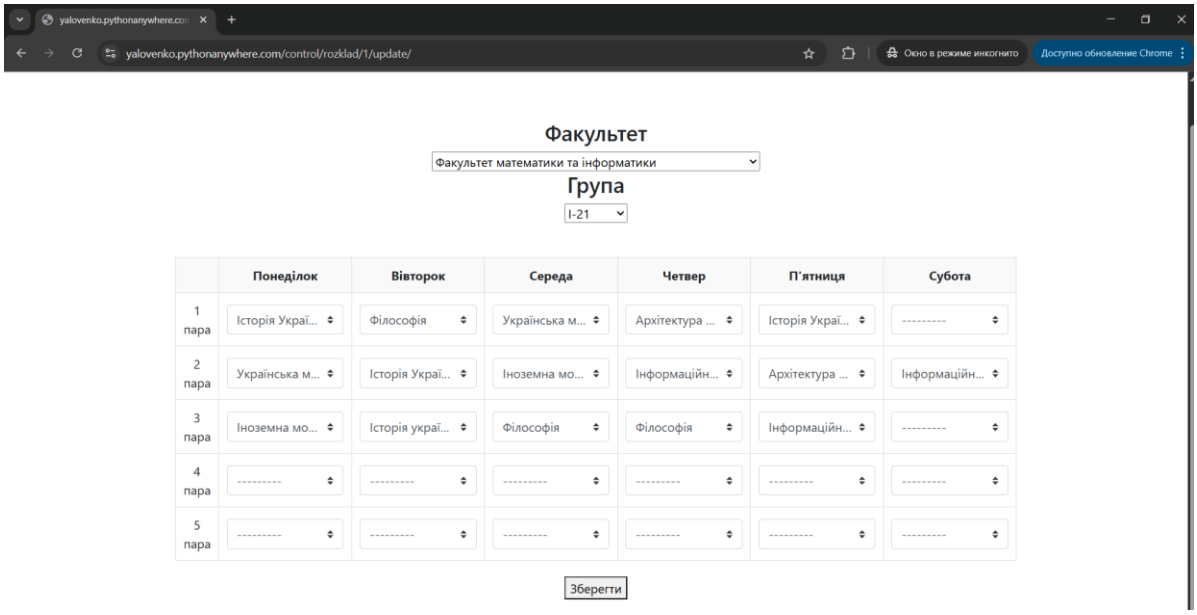

Рис. 4.2. Сторінка редагування розкладу для групи

Для вирішення вказаних проблем пропонується реалізувати розширену інтерактивну таблицю, яка дозволить здійснювати одночасне створення та редагування розкладу для всіх академічних груп факультету. Такий формат дозволить адміністрації мати цілісне уявлення про завантаженість аудиторій, викладачів та студентів у певний день чи тиждень, а також спростить процес виявлення конфліктів та дублювань.

Крім того, доцільним є впровадження автоматизованої системи перевірки конфліктів під час редагування розкладу. Цей функціонал повинен у режимі реального часу відстежувати спроби призначення:

– однієї і тієї ж дисципліни різним групам з тим самим викладачем у той самий час;

– тієї ж навчальної аудиторії для декількох груп одночасно;

– дублювання одного й того самого викладача в декількох місцях розкладу на один і той самий часовий слот.

У разі виявлення подібних конфліктів система має автоматично генерувати повідомлення з відповідним попередженням, що сприятиме оперативному усуненню проблеми до моменту збереження змін. Така реалізація забезпечить підвищення точності формування розкладу, а також значно зменшить кількість помилок, які потребують подальшого ручного коригування.

Візуалізація розкладу для всіх груп у загальному вигляді також сприятиме оптимізації навчального процесу, дозволяючи адміністративному персоналу оцінити загальне навантаження факультету в певні дні, виявляти періоди пікової завантаженості чи простої, а також коригувати розклад відповідно до доступних ресурсів — як людських, так і матеріально-технічних.

Таким чином, реалізація вдосконаленого модуля управління розкладом дозволить значно підвищити ефективність організаційної діяльності деканату, забезпечити злагодженість навчального процесу та уникнути накладок, які можуть негативно впливати на якість освітнього середовища.

4.2 Реалізація запропонованих покращень інформаційної системи деканату

Відповідно до пропозиції модернізації та покращення інформаційної системи деканату було здійснено ряд змін, спрямованих на покращення вигляду головної сторінки та підвищення зручності користування інтерфейсом. Зокрема, в першу чергу було поведено адаптацію стилю головної сторінки системи (Рис. 4.3) до сучасних вимог дизайну та юзабіліті.

Для кнопок навігації «Головна» та «Вхід» додано іконки, змінено стиль шрифту, а також переглянуто палітру кольорів. Блок, у якому користувач здійснює вибір групи, було виділено на передній план шляхом застосування сірого фону.

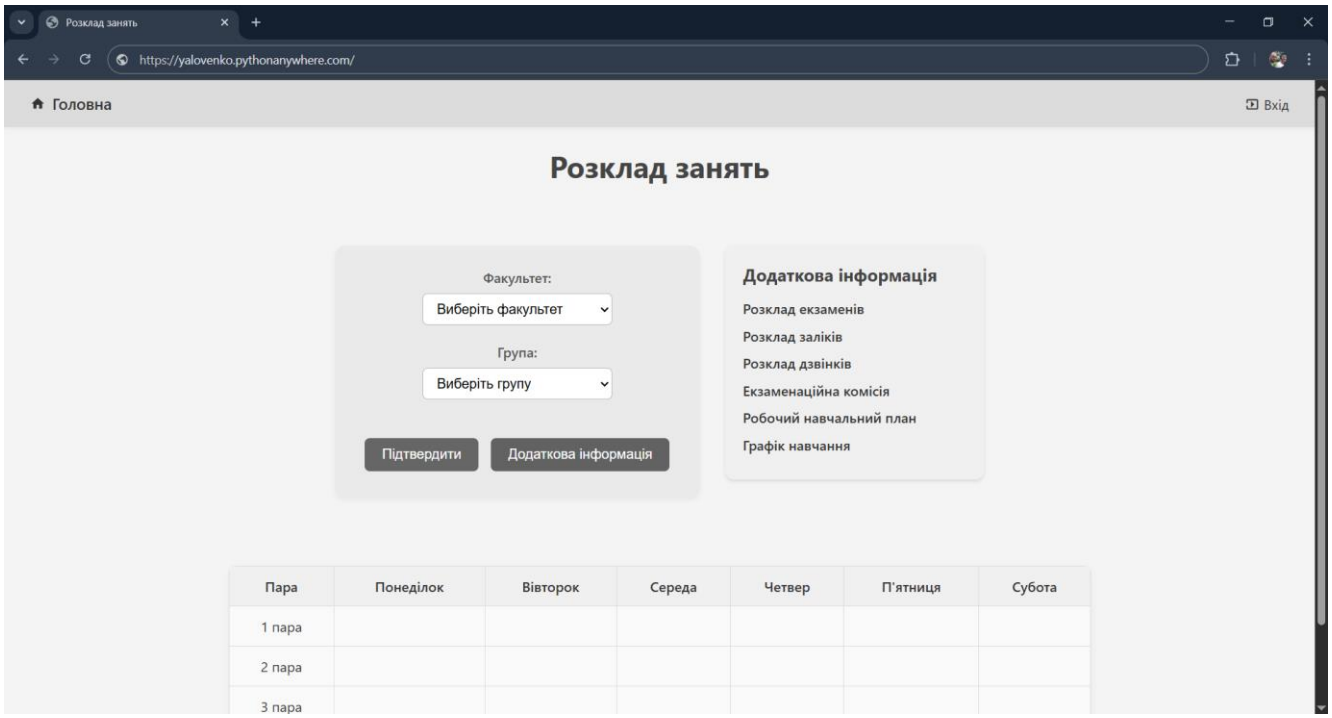

Рис. 4.3. Покращена головна сторінка інформаційної системи деканату

Інтерактивні компоненти, такі як списки вибору та кнопки зазнали стилістичних змін . Застосовано скруглення кутів та переглянуто їх кольорову гаму, що дозволило зробити інтерфейс більш сучасним та зручним для взаємодії. Після вибору факультету та групи, а також натискання кнопки «Підтвердити», на сторінці з’являється новий елемент керування — кнопка «Додаткова інформація». Після

кліку по ній користувачеві відображається відповідний інформаційний блок, що розташовується праворуч від основного блоку вибору групи. Така послідовність дій зумовлена тим, що контент блоку додаткової інформації формується для кожної окремої групи, що призводить до необхідністі його динамічного завантаження лише після визначення конкретних параметрів.

У новій версії головної сторінки було переміщено кнопки, які раніше відповідали за відображення графіка навчання та робочого навчального плану. Тепер вони знаходяться у вищезгаданому блоці додаткової інформації, де представлені під назвами «Графік навчання» та «Робочий навчальний план» відповідно.

Згідно запропонованій модифікації інформаційної системи деканату додано модулі «Графік екзаменів», «Графік заліків», «Екзаменаційна комісія» в проект. Для кожного з модулів в базу даних додано їх сутності.

Сутності «Графік екзаменів» і «Графік заліків» передбачають збереження номера ідентифікатора, дати, групи, предмету та викладача який екзамен/залік.

Таблиця 4.1

Поля сутностностей «Графік екзаменів» і «Графік заліків»

Назва поля	Опис	Тип даних
id	Номер ідентифікатора (первинний ключ)	integer
group	Група (зовнішній ключ)	integer
data	Дата	varchar(32)
predmet	Компонент освітньої програми(зовнішній ключ)	integer
vukladach	Викладач (зовнішній ключ)	integer

Сутність «Екзаменаційна комісія» передбачає збереження номера ідентифікатора, дати, голови комісії, та двох членів комісії.

Таблиця 4.2

Поля сутностності «Екзаменаційна комісія»

Назва поля	Опис	Тип даних
id	Номер ідентифікатора (первинний ключ)	integer
data	Дата	varchar(32)
golova	Викладач (зовнішній ключ)	integer
thlen_1	Викладач (зовнішній ключ)	integer
thlen_2	Викладач (зовнішній ключ)	integer

В кожному з модулів створено модель відповідно до сутності в базі даних, яка дозволяє проводити CRUD операції над даними. Для обробки запитів користувачів реалізовано представлення, які зв’зують модель із шаблонами які призначені для візуалізації інформації, що зберігається в базі даних, та забезпечують користувача зрозумілим і зручним інтерфейсом.

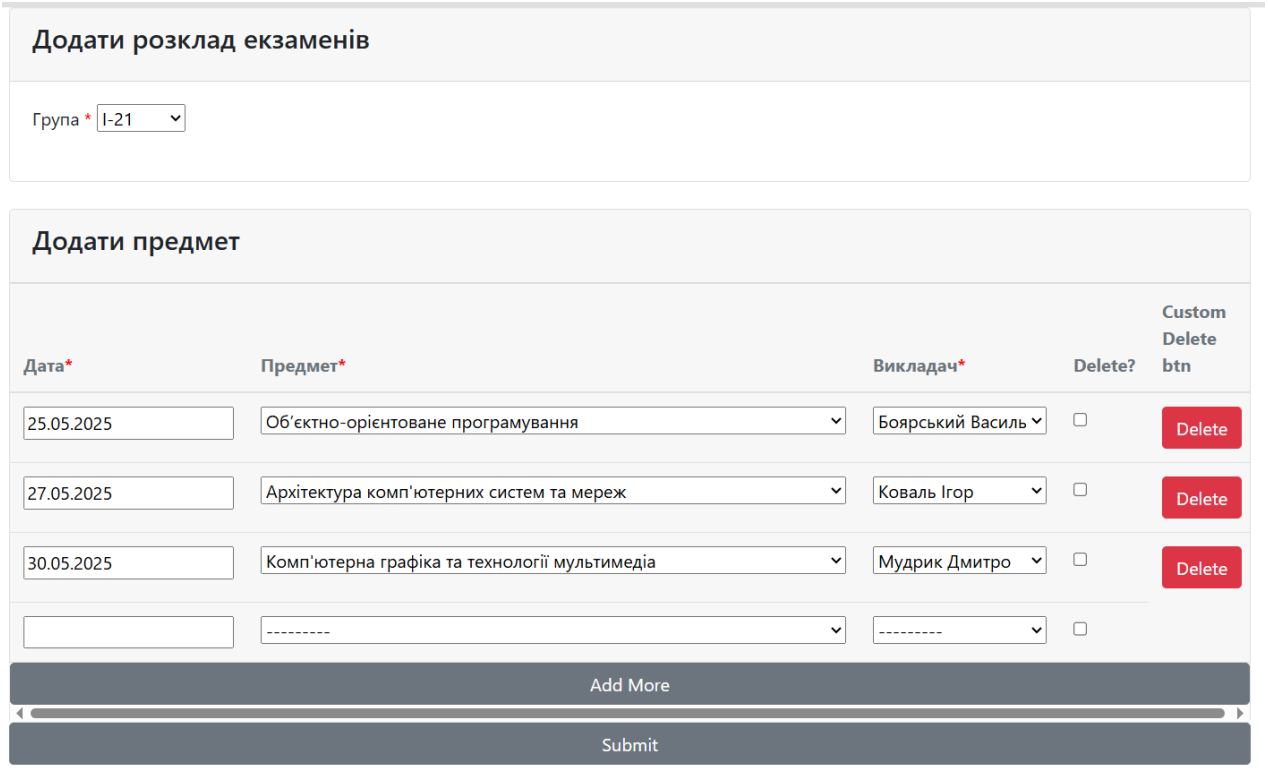

Рис. 4.4. Інтерфейс модуля «Графік екзаменів»

Інтерфейс нових модулів (Рис. 4.4) передбачає сторінки для створення і редагування інформації для кожної групи. Створені об’єкти модулів (Рис. 4.5) виводяться у список з можливістю їх видалення або редагування.

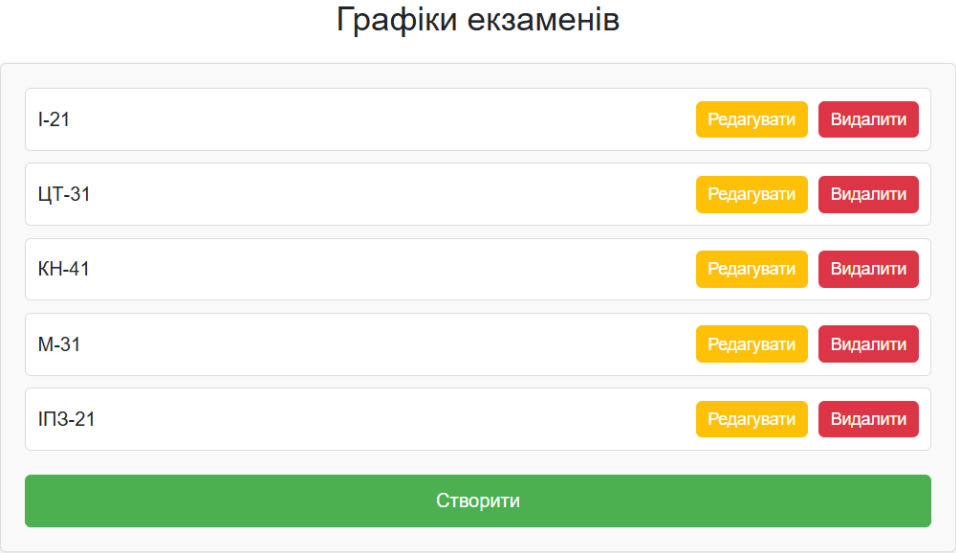

Рис. 4.5. Об’єкти модуля «Графік екзаменів»

На головній сторінці системи можна переглянути інформацію з певного модуля попередньо вибравши потрібну групу.

Найважливішою модифікацією інформаційної системи деканату є модифікація модуля «Розклад». В першу чергу для реалізації запропонованих рішень потрібно в базу даних потрібно додати таблицю «Аудиторія», що дозволить в розкладі також вказувати аудиторію де буде проводитись лекція.

Сутність «Аудиторія» передбачає збереження номера ідентифікатора, назви та адреси.

Таблиця 2.3

Поля сутностності «Аудиторія»

Назва поля	Опис	Тип даних
id	Номер ідентифікатора (первинний ключ)	integer
title	Назва	varchar(64)
adres	Адреса	varchar(256)

Слідуючим етапом роботи є модифікація файлу views.py адже він відповідає за передачу даних з моделі до шаблону і навпаки. Потрібно додати функцію вибору факультету для якого буде створюватись розклад. Звернувшись до моделі запитуємо усі факультети з бази даних, та відображаємо їх у шаблоні для користувача. Після вибору факультету проводиться редірект на наступну функцію, яка на основі вибраного об’єкта запитує з бази даних відповідні групи для передачі у шаблон. Також запитуються та передаються предмети з робочих навчальних планів груп і аудиторії. В шаблоні на основі усіх цих даних будується таблиця (Рис. 4.5), яка відображає в першій колонці дні, у другій пари, а в усіх наступних колонках предмети та аудиторії для груп які можна обрати у списках вибору.

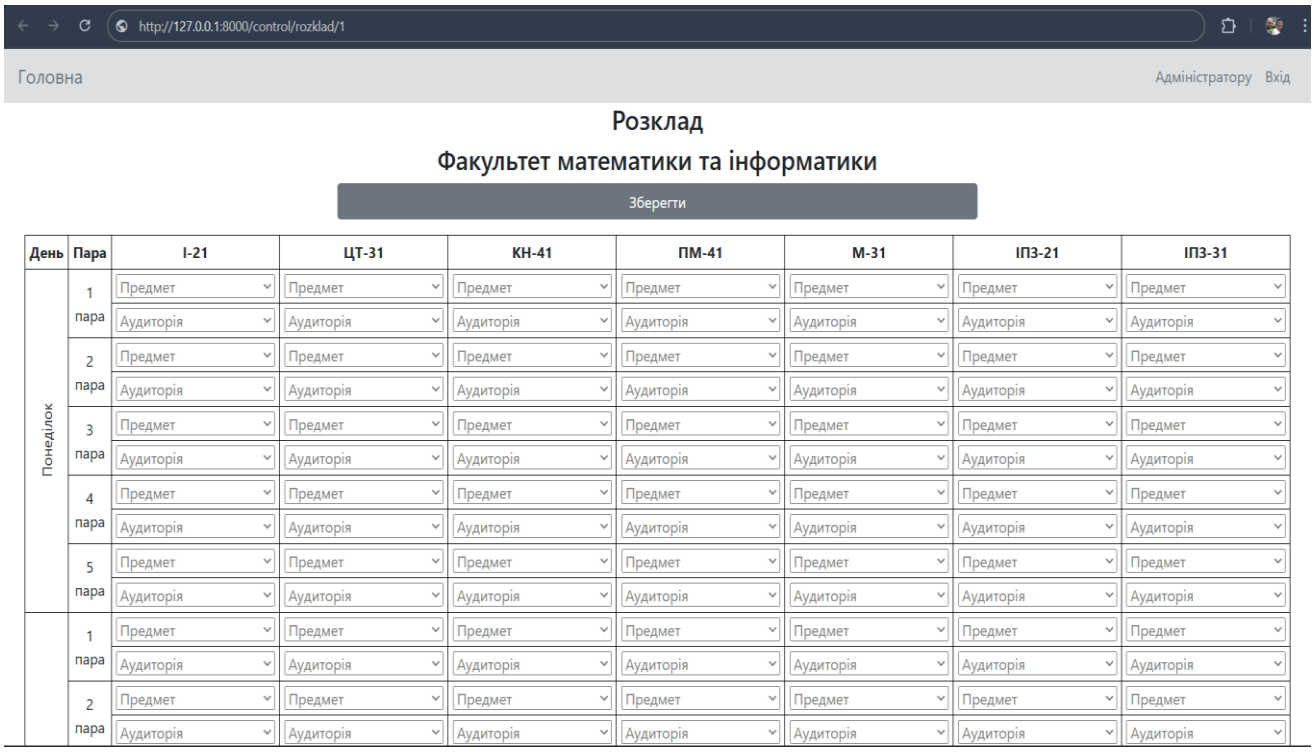

Рис. 4.5. Інтерфейс модуля «Розклад»

Передавши дані на клієнтську сторону, при заповнені таблиці відбувається перевірка на збіги по використанню однієї і тої ж аудиторії різними групами в один час, а також на збіг по викладачу який закріплений за даним предметом у відповідної групи. Це дозволить вибирати одні і ті ж предмети в однаковий час коли їх читають різні викладачі.

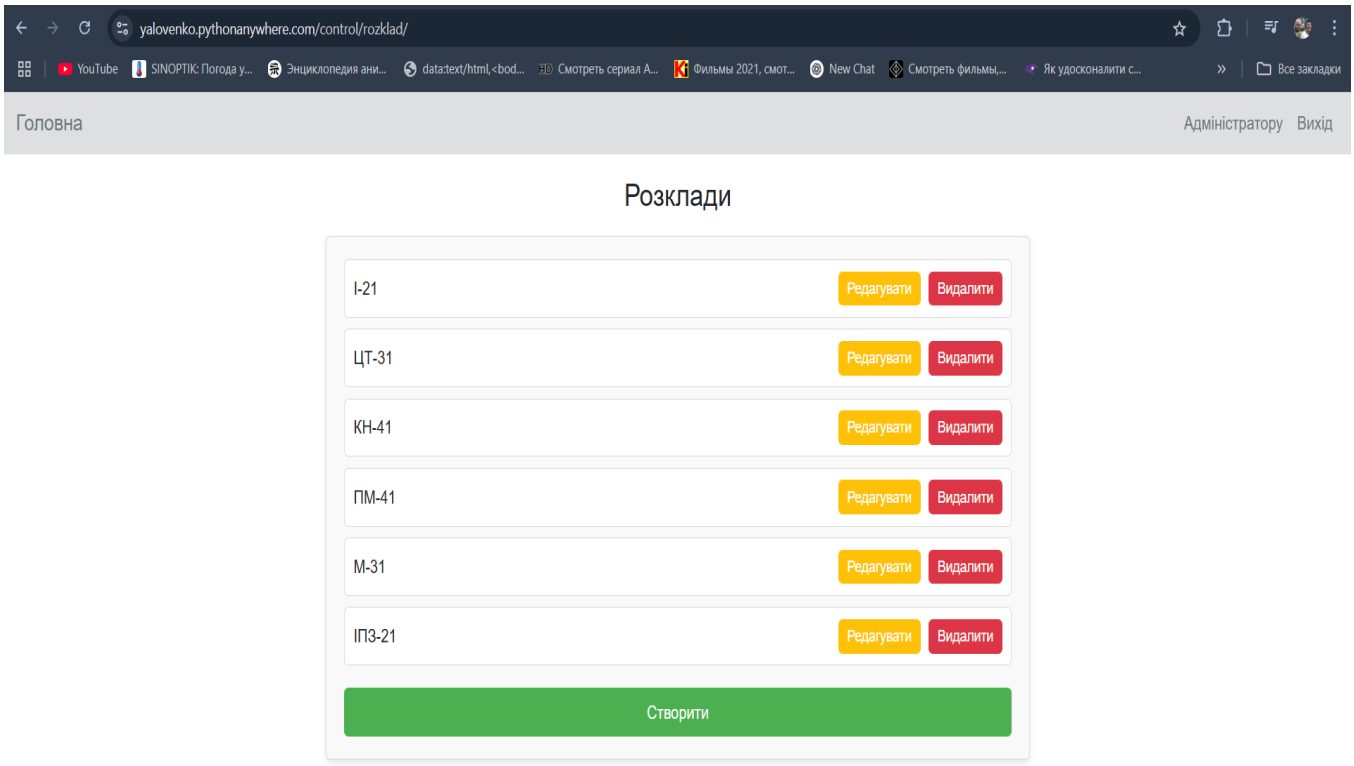

Рис. 4.6. Список розкладів модуля «Розклад»

Заповнивши таблицю даними та натиснувши кнопку «Зберегти» вони передаються на сервер де проходять валідацію і заповняються у базу даних. Новостворений розклад відображається у списку розкладів модуля «Розклад» (Рис. 4.6). На цій же сторінці можна видалити розклад натиснувши кнопку «Видалити» або ж редагувати натиснувши кнопку «Редагувати». При редагуванні відкриється схоже вікно, як і при створені, в якому після натиску кнопки «Зберегти» усі внесенні зміни будуть збережені.

ВИСНОВКИ

В результаті виконання дипломної роботи на тему «Розробка інформаційної системи деканату для ефективного управління освітнім процесом» подано теоретичні аспекти функціонування попередньо розробленої інформаційної системи деканату, яка була взята за основу для її модернізації та покращення. Безпосередньо розглянуто, що собою являє інформаційна система деканату та її функціонал. Описано технічну складову проекту, таку як архітектура, серверна і клієнтська частини, база даних та хостинг на якому розміщено систему.

Згідно поставлених завдань дослідження було проведено аналіз та комплексне тестування інформаційної системи деканату для виявлення її слабких місць, недоліків, та потенційно можливих модернізацій. Функціональне тестування дозволило виявити ряд багів у інтерфейсі системи, що характеризувалися невідповідністю до функціональних вимог або ж узагілі відсутністю будь-якого функціоналу. Юніт-тести дозволили переконатися що кожна функція серверної частини працює коректно. А інтеграційні тести підтвердили правильну взаємодію між окремими модулями та компонентами системи.

Для покращення безпеки даних було вирішено в систему додати двофакторну автентифікацію, яка забезпечує додатковий спосіб підтвердження особи користувача перед наданням доступу до системи. Функцію було реалізовано за допомогою бібліотеки pyotp яка дозволяє генерувати тимчасові паролі. Передачу паролю користувачеві вдалось досягти за допомогою сервісу Twilio, який дозволяє через API передавати згенеровані паролі користувачам через SMS.

Ще одним етапом для покращення безпеки даних було введення логування дій авторизованих користувачів. Це дозволяє відстежувати кожен вхід і вихід користувача із системи, а також відстежувати операції додавання, редагування та видалення інформації в системі.

На основі результатів тестування та аналізу також було модифіковано систему. Так як головна сторінка це перше з чим стикається користувач у системі, було вирішено змінити її та додати стилізації. Адже примітивний стиль HTML-сторінки не приваблює користувачів, та в деяких випадках навіть відлякує. Також

детальний аналіз показав, що в системі не вистачає модулів «Графік екзаменів», «Графік заліків» і «Екзаменаційна комісія» які б відповідали за представлення користувачам інформації, яка відповідає назві модулів. Було запропоновано ще одну модифікацію, мабуть найголовнішу, модифікацію модуля «Розклад». Адже в попередній версії системи він був примітивним і дозволяв створювати та редагувати розклад лише для однієї групи. Нова ж версія розкладу передбачає одночасне редагування розкладу для всього факультету та реалізовує функції запобігання збігів предметів та аудиторій на один і той же час.

Таким чином, у результаті виконання дипломної роботи було досягнуто поставленої мети - удосконалити попередньо розроблену систему, виявити та усунути недоліки, покращити функціонал та підвищити її безпеку. В подальшому, для розвитку системи, варто ввести її в експлуатацію, щоб отримати відгуки реального досвіду роботи із системою, на основі яких будуть побудовані подальші етапи модернізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. АС «деканат» - АСУ «ВНЗ». АСУ «ВНЗ». URL: https://vuz.osvita.net/as-dekanat/ (дата звернення: 16.01.2025).

2. Автоматизована система управління вищим навчальним закладом III - IV рівня акредитації. Юнітех +. URL: http://www.unitex.com.ua/products/commercial-software/automated-system-for-higher-education-institution/ (дата звернення: 15.01.2025).

3. Лазор Я. О. Поняття та види інформаційних систем. Academic Journals and Conferences. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/jun/4950/vnulpurn201683714.pdf (дата звернення: 09.01.2025).

4. Пакет програм «деканат» для вищих навчальних закладів III-IV рівнів акредитації. Політек-СОФТ. URL: http://www.politek-soft.kiev.ua/index.php?do=products&product=deanery34 (дата звернення: 16.01.2025).

5. Різниця між функціональним і нефункціональним тестуванням. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/ (дата звернення: 17.02.2025).

6. Яловенко Л. Розробка інформаційної системи деканату для ефективного управління освітнім процесом : Магістерська робота. Рівне, 2024. 101 с.

7. Яловенко Л., Крайчук С. Інформаційна система деканату для ефективного управління освітнім процесом. // Наука, освіта, суспільство очима молодих: збірник тез та доповідей XVIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених (м. Рівне, 14 трав. 2025 р.). Рівне, 2025.

8. Ahmed N. 🚀 Why django is the ultimate □ “batteries-included” framework for scalable 🌐 web development. Medium. URL: https://medium.com/@naeem.ahmed.bdn/why-django-is-the-ultimate-batteries-

included-framework-for-scalable-web-development-746ffcebaa7d (date of access: 14.01.2025).

9. Api W. Weekly API: twilio SMS API and alternatives. Medium. URL: https://medium.com/@weeklyapi/weekly-api-twillio-sms-api-and-alternatives-e94f5cc481ea (date of access: 17.02.2025).

10. Azie M. G. Getting started with unit testing in python using pytest: a beginner's guide. Medium. URL: https://medium.com/@xstepnort/getting-started-with-unit-testing-in-python-using-pytest-a-beginners-guide-840d801c1d12 (date of access: 17.02.2025).

11. Edirisinghe D. M. Integration testing in .NET 8. Medium. URL: https://blog.devgenius.io/integration-testing-in-net-8-b8ab83e7531d (date of access: 17.02.2025).

12. Gentile A. Basics of django: model-view-template (MVT) architecture. Medium. URL: https://angelogentileiii.medium.com/basics-of-django-model-view-template-mvt-architecture-8585aecffbf6 (date of access: 12.01.2025).

13. Hassan A. H. Getting started with sqlite in python on windows. Medium. URL: https://medium.com/@hannanmentor/getting-started-with-sqlite-in-python-on-windows-149e95d6c21f (date of access: 11.01.2025).

14. Kamilo M. What is reactjs? A beginner's guide to understanding its purpose and role in web development. Medium. URL: https://kamelodee.medium.com/what-is-reactjs-a-beginners-guide-to-understanding-its-purpose-and-role-in-web-development-f28f714486f2 (date of access: 15.01.2025).

15. Khire A. Introduction to reactjs: a quick start guide for web developers. Medium. URL: https://medium.com/@abhijeetkhire/introduction-to-reactjs-a-quick-start-guide-for-web-developers-d5f781aa952e (date of access: 15.01.2025).

16. Logging | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.2/topics/logging/ (date of access: 21.02.2025).

17. Manoharan I. (. Understanding Two-Factor Authentication (2FA). Medium. URL: https://medium.com/@ilakk2023/309-understanding-two-factor-authentication-2fa-e27e8c378075 (date of access: 21.02.2025).

18. Nader Y. What is django? Advantages and disadvantages of using django. Hackr.io. URL: https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django (date of access: 14.01.2025).

19. Priya S. What is ReactJS?. Medium. URL: https://medium.com/@sudhapriyaofficial/what-is-reactjs-1ac3282a60cb (date of access: 13.02.2025).

20. Programming P.-C. P. PythonAnywhere’s free version – yourname.pythonanywhere.com domain and building a discord bot. Medium. URL: https://medium.com/@ccpythonprogramming/pythonanywheres-free-version-yourname-pythonanywhere-com-domain-and-building-a-discord-bot-e24fc3dd1775 (date of access: 17.01.2025).

21. Pyotp. PyPI. URL: https://pypi.org/project/pyotp/ (date of access: 02.02.2025).

22. Techtutorsti. Python anywhere. Medium. URL: https://medium.com/@techtutorsti/python-anywhere-0279a0905142 (date of access: 16.01.2025).

23. Verma P. Django logging | tutorial & best practices. Medium. URL: https://medium.com/django-unleashed/django-logging-tutorial-best-practices-ed5052b97f1d (date of access: 21.02.2025).

24. You M. F. What is react js? What does it do? The 2023 guide. Medium. URL: https://medium.com/@mtcpe.fy/what-is-react-js-what-does-it-do-the-2023-guide-c5bf604e1664 (date of access: 15.01.2025).