Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до кваліфікаційної роботи бакалавра
за спеціальністю 123 «Комп'ютерна інженерія»

на тему: АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ РОЗКЛАДОМ

Проектував _____ Є. М. Гриб

Керівник роботи _____ С. В. Сьомочкина

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2024

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти бакалавр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____ А. І. Купін

"____" _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гриб Єлизаветі Максимівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Автоматизована система управління розкладом*
_____
_____

керівник роботи *Сьомочкина Світлана Володимірівна, доц. каф. КСМ, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "*27*" *листопада* 20*23* року №5

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи *Розклад керівника з сайту asu.knu.edu.ua, документація Google Calendar API*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ, Загальна характеристика підходів до управління розкладом, Аналіз існуючих тенденцій використання розкладу занять, Аналіз необхідних умов для проведення онлайн занять, Порівняння функцій та можливостей сучасних платформ для онлайн зустрічей, Аналіз існуючих систем управління розкладом, Постановка задач та вимог, Структурна схема Взаємозв'язки між модулями, Створення веб-сервера, Створення бази даних, Інтеграція з Google Calendar API, Розробка коду алгоритму системи, Візуалізація роботи системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація у Microsoft PowerPoint в електронному та друкованому вигляді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	*Сьомочкина С. В. , доц. каф. КСМ, канд. техн. наук*		
2	*Сьомочкина С. В. , доц. каф. КСМ, канд. техн. наук*		
Нормконтроль	*Кузнєцов Д. І., доц. каф. КСМ, канд. техн. наук*		

7. Дата видачі завдання *08.02.2024*

КАЛЕНДАРНИЙ ПЛАН

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.02.24 - 20.02.24*
2	*Загальна характеристика підходів до управління розкладом*	*20.02.24 - 23.03.24*
3	*Система автоматизованого управління розкладом*	*12.03.24 - 25.05.24*
4	*Висновки*	*10.05.24 - 15.05.24*
5	*Оформлення пояснювальної записки*	*25.05.24 - 01.06.24*
6	*Підготовка презентації*	*01.06.24 - 05.06.24*
7	*Підготовка доповіді до захисту*	*05.06.24 - 13.06.24*

Студент _____ *Гриб Є. М.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Сьомочкина С. В.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 42 сторінок, 16 рисунків, 2 таблиці, 5 додатків, 30 використаних джерел.

Об'єкт аналізу - процес управління розкладом занять у навчальних закладах, зокрема в умовах онлайн-навчання.

Проект складається з двох розділів.

У першому розділі проведено комплексний аналіз підходів до управління розкладом занять, особливо в умовах онлайн-навчання. Оцінено сучасні тенденції використання розкладу занять, необхідні умови для проведення онлайн-занять, функції та можливості платформ для онлайн-зустрічей. Аналізовано існуючі системи управління розкладом та їх інтеграцію з іншими системами.

У другому розділі описано алгоритм роботи автоматизованої системи управління розкладом, її архітектуру та процеси взаємодії. Детально розглянуто структурну схему системи, алгоритм зчитування даних з файлів Excel, створення веб-сервера за допомогою фреймворку Django, та інтеграцію з Google Calendar API.

Ключові слова: АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ РОЗКЛАДОМ, ОНЛАЙН-НАВЧАННЯ, ДИСТАНЦІЙНА ОСВІТА, GOOGLE CALENDAR API, DJANGO, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПЛАТФОРМИ ДЛЯ ВІДЕОКОНФЕРЕНЦІЙ.

ABSTRACT

Explanatory note: 42 pages, 16 figures, 2 tables, 5 appendix, 30 used sources.

Object of analysis - the process of schedule management in educational institutions, particularly in the context of online learning.

The project consists of two chapters.

In the first chapter, a comprehensive analysis of schedule management approaches, especially in the context of online learning, is conducted. Current trends in schedule utilization, necessary conditions for conducting online classes, and functions and capabilities of online meeting platforms are evaluated. Existing schedule management systems and their integration with other systems are analyzed.

In the second chapter, the algorithm of the automated schedule management system, its architecture, and interaction processes are described. The structural scheme of the system, the algorithm for reading data from Excel files, the development of a web server using the Django framework, and the integration with Google Calendar API are detailed.

Keywords: AUTOMATED SCHEDULE MANAGEMENT SYSTEM, ONLINE LEARNING, DISTANCE EDUCATION, GOOGLE CALENDAR API, DJANGO, CLIENT-SERVER ARCHITECTURE, SOFTWARE, ONLINE MEETING PLATFORMS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CRM - Customer Relationship Management (управління взаєминами з клієнтами);

ERP - Enterprise Resource Planning (планування ресурсів підприємства);

VPN - Virtual Private Network (віртуальна приватна мережа);

SSID - Service Set Identifier (ідентифікатор набору послуг);

VoIP - Voice over Internet Protocol (голосова передача даних через Інтернет);

FTP - File Transfer Protocol (протокол передачі файлів);

SMTP - Simple Mail Transfer Protocol (простий протокол передачі пошти).

ВСТУП

В умовах сучасного світу технології інформації та комунікації відіграють критично важливу роль у всіх сферах життя, включаючи освіту. З розвитком онлайн-навчання та дистанційної освіти виникає потреба у створенні нових підходів до організації навчального процесу, зокрема, управління розкладом занять. Автоматизовані системи управління розкладом стають все більш актуальними для забезпечення ефективного та зручного планування навчальних заходів.

Сьогоднішні реалії, зокрема пандемія COVID-19, значно прискорили впровадження онлайн-навчання, що вимагає нових підходів до організації та управління навчальним процесом. Традиційні методи управління розкладом часто не відповідають вимогам сучасного освітнього середовища, оскільки вони не враховують гнучкості та мобільності, які необхідні для дистанційного навчання. Автоматизовані системи управління розкладом можуть вирішити ці проблеми, забезпечуючи ефективне використання часу та ресурсів як для викладачів, так і для студентів.

Метою даної роботи є розробка автоматизованої системи управління розкладом занять, яка забезпечувала б зручне та ефективне планування, синхронізацію та контроль за навчальним процесом в умовах онлайн-навчання. Система повинна враховувати сучасні вимоги та технологічні можливості, забезпечуючи інтеграцію з існуючими платформами для відеоконференцій та іншими освітніми інструментами.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Провести аналіз існуючих тенденцій та проблем в управлінні розкладом занять в умовах онлайн-навчання.
2. Оцінити сучасні платформи для проведення онлайн-занять та визначити їхні основні функції та можливості.
3. Розробити концептуальну модель автоматизованої системи управління розкладом.
4. Визначити основні вимоги до функціональності та архітектури системи.
5. Реалізувати прототип системи, що забезпечуватиме створення та редагування розкладів, інтеграцію з платформами для відеоконференцій, автоматичні сповіщення та нагадування.

Об'єктом дослідження є процес управління розкладом занять у навчальних закладах, зокрема в умовах онлайн-навчання. Предметом дослідження є автоматизовані системи управління розкладом, їхні функціональні можливості, архітектура та технології, що використовуються для їх реалізації.

У роботі використано комплекс методів, включаючи аналітичний огляд літератури та існуючих рішень, порівняльний аналіз платформ для онлайн-навчання, проектування інформаційних систем та програмування. Використання таких методів дозволяє здійснити всебічний аналіз проблеми та запропонувати оптимальне рішення для автоматизації управління розкладом занять.

Наукова новизна роботи полягає в розробці та впровадженні нової автоматизованої системи управління розкладом занять, яка враховує сучасні вимоги онлайн-навчання та забезпечує інтеграцію з популярними платформами для відеоконференцій. Практичне значення роботи полягає у можливості використання розробленої системи в навчальних закладах для підвищення ефективності організації навчального процесу.

РОЗДІЛ 1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПІДХОДІВ ДО УПРАВЛІННЯ РОЗКЛАДОМ

1.1 Аналіз існуючих тенденцій використання розкладу занять

В останні роки спостерігається значне зростання популярності онлайн-навчання та дистанційної освіти. Ця тенденція виникла з ряду причин, зокрема, технологічних зрушень, культурних змін у способах навчання та, безперечно, впливу глобальних обставин, таких як пандемія.

Замикання шкіл, університетів та інших освітніх установ в реакції на кризу спричинило необхідність швидкої адаптації до нових умов. Онлайн-навчання стало ключовим інструментом для забезпечення продовження освітнього процесу.

Перехід до онлайн-формату навчання має різні наслідки. Від викладачів до учнів, всі учасники освітнього процесу відчувають вплив цих змін. Це також створює нові вимоги до управління навчальним процесом, зокрема, до створення нових систем для управління й автоматизованої підтримки розкладу занять, загалом процесу навчання тощо. У зв'язку з цим, аналіз існуючих тенденцій використання розкладу занять у онлайн-навчанні є ключовим етапом для розуміння потреб та вимог освітнього середовища. Цей аналіз дозволить виявити важливі аспекти, які необхідно враховувати при розробці та впровадженні автоматизованих систем управління розкладом занять.

Відмінність онлайн-навчання полягає в тому, що воно потребує інших методів та інструментів управління часом та ресурсами порівняно з традиційними заняттями в аудиторіях. На відміну від статичних розкладів занять, які можна легко розробити для очного навчання, враховуючи лише фактори знаходження усіх учасників в одному місці, де немає так багато різноманіття, розклад для онлайн-курсів потребує більшої гнучкості та автоматизації.

Однією з ключових причин, які спонукають до автоматизації управління розкладом занять, є необхідність забезпечення ефективного використання часу та ресурсів. Через те, що онлайн-навчання може відбуватися в будь-який час та будь-де, оптимальне розподілення часу має велике значення. Іншою причиною є потреба відповідати на індивідуальні потреби студентів. Завдяки автоматизованій системі управління розкладом занять можна легко адаптувати розклад до різних часових зон, індивідуальних графіків та потреб студентів й викладачів. Завжди простіше, за наявності інтернет-зв’язку натиснути на вже створене онлайн посилання та опинитися безпосередньо на самому занятті.

Аналіз тенденцій використання розкладу занять у контексті онлайн-навчання відкриває можливості для розробки і впровадження ефективних автоматизованих систем управління розкладом занять, які відповідали б сучасним потребам освітнього процесу.

На сьогоднішній день вже існують різні системи управління розкладом занять, які використовуються у навчальних закладах та на платформах онлайн-навчання. Деякі з них можуть бути призначені для фізичних занять, а інші - для дистанційного навчання. Проте, відомо, що багато з цих систем не завжди відповідають сучасним вимогам та потребам освітнього процесу, особливо у зв'язку з ростом онлайн-навчання та змінами в глобальній освітній парадигмі.

Розклад занять у онлайн-навчанні вимагає більшої гнучкості та упорядкування, процес потребує бути полегшеним для того, щоб полегшувати наджодження інформації з обох сторін оскільки студенти можуть навчатися у різний час та з різних часових зон та викладачі можуть знаходитися у різних умовах та на різних відстанях від університету, фізичного розкладу тощо. Також, індивідуальні потреби студентів у контексті онлайн-навчання можуть відрізнятися, що вимагає адаптивності систем управління розкладом.

Отже, можна зробити висновок, що є необхідність у нових та вдосконалених рішеннях, які б краще відповідали потребам сучасного освітнього середовища. Помітно, що існує різка потреба автоматизувати створення онлайн-посилань на лекції, лабораторні роботи та зустрічі, так як це допоможе викладачеві більше фокусуватися на якості навчального процесу.

1.2 Аналіз необхідних умов для проведення онлайн занять

Проведення онлайн занять передбачає ретельний аналіз низки умов, що впливають на ефективність та успішність таких занять. Серед ключових умов можна виділити наступні:

- Інтернет-з'єднання та технічні засоби: Наявність стабільного інтернет-з'єднання та відповідних технічних засобів (комп'ютери, ноутбуки, планшети, смартфони) є ключовим фактором для успішного проведення онлайн занять.
- Платформи та програмне забезпечення: Вибір відповідних платформ для відеоконференцій та спеціалізованого програмного забезпечення для онлайн навчання є критичним для забезпечення ефективного спілкування та взаємодії між викладачами та студентами.
- Методична підтримка: Наявність чіткої методичної підтримки з боку викладачів та адміністративного персоналу, яка включає в себе інструкції з використання платформ та програмного забезпечення, а також підтримку в разі технічних проблем.
- Оцінювання та зворотний зв'язок: Наявність системи оцінювання та зворотного зв'язку, яка дозволяє викладачам ефективно оцінювати навчальний прогрес студентів та надавати їм конструктивний фідбек для покращення результатів.

Аналіз та врахування цих умов дозволяє ефективно реалізувати проведення онлайн занять з метою забезпечення якісної освіти та навчання.

Важливо використовувати якісні засоби, які дозволяють ефективно організовувати, проводити та участь в онлайн заняттях. Вони дозволяють здійснювати ефективну взаємодію між викладачами та студентами, забезпечуючи високу якість навчання та спілкування. Такі засоби сприяють активній участі студентів у навчальному процесі.

Існує ряд спільних характеристик та функціоналу, які є характерними для багатьох засобів онлайн комунікації та спільної роботи. Серед них можна виділити можливість відеоконференцій, спільної роботи над документами, обмін повідомленнями та інтеграцію з іншими інструментами для ефективної організації навчального процесу.

Онлайн календарі є важливим інструментом для організації та планування розкладу занять в онлайн форматі. Вони дозволяють викладачам створювати розклади, які автоматично синхронізуються з календарями студентів, надаючи можливість ефективно використовувати час та планувати заняття. Такі календарі також забезпечують можливість нагадувань та сповіщень, що допомагає уникнути пропуску важливих подій.

1.3 Порівняння функцій та можливостей сучасних платформ для онлайн зустрічей

В сучасному світі інформаційних технологій інструменти для проведення онлайн зустрічей стають все більш популярними та розповсюдженими. Розглянемо та порівняймо функції та можливості кількох провідних платформ для онлайн занять:

- Zoom:

Відеоконференції. Zoom надає можливість проводити відеоконференції з великою кількістю учасників одночасно.

Екранний спільний доступ. Учасники можуть спільно працювати над документами або веб-сайтами, використовуючи функцію екранного спільного доступу.

Чат та віртуальна дошка. Можливість обміну повідомленнями в чаті та використання віртуальної дошки для спільної роботи над ідеями та концепціями.

- Microsoft Teams:

Інтеграція з іншими сервісами Microsoft: Teams інтегрується з іншими інструментами Microsoft, такими як Word, Excel та PowerPoint, що спрощує спільну роботу над документами та презентаціями.

Чат та обмін файлами. Учасники можуть спілкуватися через чат, обмінюватися файлами та створювати групові чати для спільної роботи.

- Google Meet:

Інтеграція з Google Workspace. Google Meet легко інтегрується з іншими інструментами Google, такими як Gmail, Google Calendar та Google Drive.

Можливість підключення до зустрічі без обов'язкового облікового запису Google: Учасники можуть приєднатися до зустрічі навіть без облікового запису Google, що робить доступ до зустрічі більш зручним.

- Cisco Webex:

Додаткові функції безпеки. Webex пропонує додаткові функції безпеки, такі як шифрування даних, контроль доступу та можливість встановлення паролів для зустрічей.

Велика кількість учасників. Платформа дозволяє проводити зустрічі з великою кількістю учасників, що особливо корисно для великих команд або великих груп.

Кожна з цих платформ має свої переваги та функції, і вибір між ними може залежати від конкретних потреб та вимог користувача. Проведений аналіз допомагає визначити оптимальний вибір платформи для конкретних цілей онлайн зустрічей.

В таблиці 1.1 представлено порівняння сучасних платформ для проведення онлайн зустрічей.

Таблиця 1.1 - Порівняння платформ для проведення онлайн зустрічей

Платформа	Zoom	Microsoft Teams
Відеоконференції	Можливість проводити зустрічі з великою кількістю учасників	Підтримка великої кількості учасників, інтеграція з іншими сервісами Microsoft
Екранний спільний доступ	Так	Так
Чат	Так	Так
Віртуальна дошка	Так	Ні
Інтеграція	Ні	Інтеграція з іншими сервісами Microsoft
Безпека	Базові функції безпеки	Керування доступом, шифрування даних
Максимальна кількість учасників	до 100 осіб	до 300 осіб
Відеоконференції	Підтримка великої кількості учасників, інтеграція з Google Workspace	Здатність проводити зустрічі з великою кількістю учасників, функції безпеки
Екранний спільний доступ	Ні	Так
Чат	Так	Так
Віртуальна дошка	Ні	Ні

Платформа	Zoom	Microsoft Teams
Інтеграція	Інтеграція з Google Workspace	Ні
Безпека	Базові функції безпеки	Додаткові функції безпеки
Максимальна кількість учасників	до 100 осіб	до 50 осіб

1.4 Аналіз існуючих систем управління розкладом

Системи управління розкладом є інструментами, які допомагають організовувати розклад подій, завдань або ресурсів у відповідності з певними критеріями та обмеженнями. Ці системи можуть бути корисними для широкого спектра сфер, включаючи навчальні заклади.

Основні функції систем управління розкладом включають:

- Створення розкладів: дозволяє користувачам створювати розклади для подій, зустрічей, завдань або ресурсів на певний період часу.
- Оптимізація розкладів: деякі системи можуть автоматично оптимізувати розклади з урахуванням різних факторів, таких як доступні ресурси, пріоритети, обмеження та умови.
- Моніторинг та оновлення: дозволяє вести моніторинг виконання розкладу, вносити зміни у випадку необхідності та оновлювати інформацію.
- Синхронізація та спільний доступ: багато систем управління розкладом підтримують синхронізацію розкладів між різними пристроями та спільний доступ до них для групової роботи.
- Повідомлення та нагадування: деякі системи можуть надсилати користувачам повідомлення та нагадування про наближення подій або завдань.
- Аналіз та звітність: деякі системи можуть надавати можливості для аналізу використання часу, продуктивності та ефективності розкладу через звіти та аналітику.
- Інтеграція з іншими програмами: багато систем управління розкладом можуть інтегруватися з іншими програмами, такими як електронна пошта, календарі, завдання та інші.

Деякі приклади систем управління розкладом включають Google Календар, Microsoft Outlook, Asana, Trello, JIRA та багато інших. Вони можуть бути використані як для особистих потреб, так і для командної роботи в організаціях будь-якого масштабу.

Система управління розкладом добре поєднується з платформами проведення онлайн занять, що дозволяє створювати ще більш ефективне та зручне середовище для навчання та організації подій. Інтеграція цих двох рішень дозволяє автоматизувати процес планування, моніторингу та

виконання навчальних заходів, забезпечуючи зручність для вчителів, учнів та адміністраторів.

Наприклад, платформи для проведення онлайн занять, такі як Zoom, Google Meet або Microsoft Teams, можуть інтегруватися з системами управління розкладом, такими як Google Календар або Microsoft Outlook Calendar. Це дозволяє автоматично створювати розклади занять на основі інформації з обох систем, надсилати сповіщення про заплановані заняття та посилання для участі у них, а також вести моніторинг виконання розкладу та зберігати дані про участь.

Таке поєднання допомагає покращити організацію навчального процесу, знижуючи ризик збігу часів, покращуючи доступність та забезпечуючи ефективну взаємодію між учасниками навчання.

В таблиці 1.2 представлено порівняння сучасних систем управління розкладом.

Таблиця 1.2 - Порівняння сучасних систем управління розкладом

Характеристика	Google Календар	Microsoft Outlook
Інтерфейс	Сучасний, простий у використанні, зручний для користувачів.	Має класичний вигляд, з основною панеллю та панелями з боків.
Інтеграція з Gmail	Інтегрується з Gmail, що дозволяє автоматично додавати події з електронної пошти.	Інтегрований з Outlook Email, але не з Gmail.
Перегляд подій	Події можна переглядати у вигляді дня, тижня, місяця або розкладу.	Події можна переглядати у вигляді дня, тижня, місяця або розкладу.
Додаткові можливості	Має можливість створювати різнокольорові категорії подій, розподіляти доступ, обмінюватися подіями тощо.	Має можливість створювати різнокольорові категорії подій, додавати напоминання, обмінюватися календарями.
Мобільні додатки	Має мобільні додатки для Android та iOS з широким функціоналом.	Має мобільні додатки для Android та iOS з відмінною синхронізацією з ПК.

Характеристика	Google Календар	Microsoft Outlook
Командна робота	Можливість додавати спільні календарі та ділитися подіями з іншими користувачами Google.	Можливість додавати спільні календарі та ділитися подіями з іншими користувачами Outlook.

1.5 Постановка задач та вимог

Для розробки ефективної автоматизованої системи управління розкладом необхідно врахувати такі вимоги:

Функціональні вимоги:

1. Створення та редагування розкладів:

- Система повинна дозволяти створення розкладів для різних курсів, занять та подій.
- Внесення змін у розклад повинно бути інтуїтивно зрозумілим та швидким.

2. Інтеграція з іншими системами:

- Інтеграція з платформами для онлайн-зустрічей (Zoom, Microsoft Teams, Google Meet).
- Синхронізація з календарями (Google Calendar, Outlook).

3. Повідомлення та нагадування:

- Автоматичні сповіщення про зміни в розкладі.
- Нагадування про майбутні події для викладачів та студентів.

4. Користувацький інтерфейс:

- Зручний та інтуїтивно зрозумілий інтерфейс для всіх категорій користувачів (викладачі, студенти, адміністратори).
- Підтримка різних мов та адаптивний дизайн для мобільних пристроїв.

2. Нефункціональні вимоги:

1. Безпека:

- Захист персональних даних користувачів.
- Надійна аутентифікація та авторизація.

2. Продуктивність:

- Висока швидкість обробки запитів та оновлень розкладу.
- Масштабованість для обробки великої кількості користувачів та подій.

3. Надійність:

- Висока доступність системи (мінімальні простої).
- Захист від втрати даних через регулярне резервне копіювання.

4. Сумісність:

- Підтримка різних операційних систем та браузерів.
- Легка інтеграція з існуючими освітніми платформами та системами управління навчанням.

Технічні вимоги:

1. Технологічний стек:

- Використання сучасних фреймворків для веб-розробки (Django).
- Бази даних з високою продуктивністю та масштабованістю (PostgreSQL).

2. Архітектура:

- Клієнт-серверна архітектура для ефективного розподілу навантаження.
- Використання API для інтеграції з зовнішніми сервісами.

3. Документація та підтримка:

- Детальна технічна документація для розробників та користувачів.
- Регулярні оновлення та технічна підтримка.

Висновок до розділу 1

У розділі 1 проведено комплексний аналіз підходів до управління розкладом занять, особливо в контексті онлайн-навчання. Розглянуто основні тенденції та виклики, з якими стикаються освітні установи при переході на дистанційне навчання, а також необхідність автоматизації для ефективного управління розкладом. Було встановлено, що традиційні методи не відповідають сучасним вимогам, що підкреслює важливість розробки нових систем.

Для досягнення цього було проаналізовано функціональні можливості та обмеження існуючих платформ для онлайн-зустрічей (Zoom, Microsoft Teams, Google Meet, Cisco Webex). Порівняння показало, що кожна платформа має унікальні переваги, такі як інтеграція з іншими сервісами та додаткові функції безпеки. Цей аналіз допоміг визначити ключові характеристики, які мають бути враховані при виборі платформи для інтеграції з системою управління розкладом.

Також був проведений огляд існуючих систем управління розкладом, включаючи їхні основні функції та можливості. Системи, такі як Google Календар, Microsoft Outlook, Asana, Trello та JIRA, були порівняні за критеріями функціональності, інтеграції, синхронізації та оптимізації. Результати показали, що хоча ці системи мають багато корисних функцій, вони не завжди повністю задовольняють сучасні потреби освітніх установ, особливо в умовах зростаючої популярності онлайн-навчання.

На основі проведеного аналізу було сформульовано вимоги до нової автоматизованої системи управління розкладом. Вони включають створення та редагування розкладів, автоматизацію оптимізації, інтеграцію з платформами для онлайн-зустрічей, автоматичні сповіщення та нагадування, зручний інтерфейс користувача, високу безпеку, продуктивність, надійність та сумісність з різними операційними системами та браузерами. Ці вимоги мають забезпечити ефективне управління розкладом, задовольняючи потреби як викладачів, так і студентів.

РОЗДІЛ 2 СИСТЕМА АВТОМАТИЗОВАНОГО УПРАВЛІННЯ РОЗКЛАДОМ

2.1 Алгоритм роботи системи

Автоматизована система управління розкладом має можливість зчитувати файл розкладу, використовуючи файл із освітнього порталу http://asu.knu.edu.ua/. У кожного викладача є власний розклад, що може бути експортований у форматі .xls. Цей файл є вхідною точкою до алгоритму системи.

Алгоритм зчитування інформації полягає в знаходженні комірок, які відповідають заняттям. Щоб визначити дату та час проведення заняття необхідно тимчасово зберігати значення з попереднього рядка, що відповідає датам, та першої комірки, де визначено час заняття. Комірка заняття включає в себе дані про назву групи, назву предмету та аудиторію. Таким чином формується масив подій із інформацією:

- дата проведення
- час проведення
- назва події (заняття)
- назва групи

На рисунку 2.1 зображено алгоритм системи управління розкладом (отримання вхідного файлу).

Рисунок 2.1 - Алгоритм системи управління розкладом (отримання вхідного файлу)

На рисунку 2.2 зображено алгоритм системи управління розкладом (створення подій).

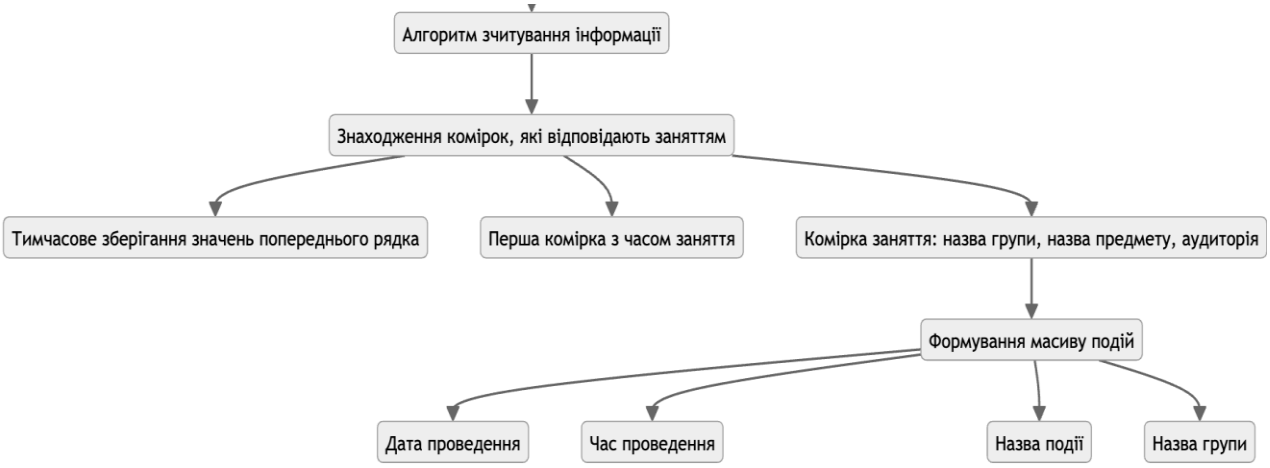

Рисунок 2.2 - Алгоритм системи управління розкладом (створення подій)

2.2 Структурна схема

Архітектура клієнт-сервер забезпечує ефективне управління і обробку даних, дозволяючи користувачам зручно взаємодіяти з системою через веб-додаток. Це розподіляє навантаження між клієнтською та серверною частинами, забезпечуючи високу продуктивність та гнучкість системи управління розкладом.

При кожному запиті до системи користувач отримує відповідь. Така схема відповідає роботі веб-додатка (рисунок 2.3).

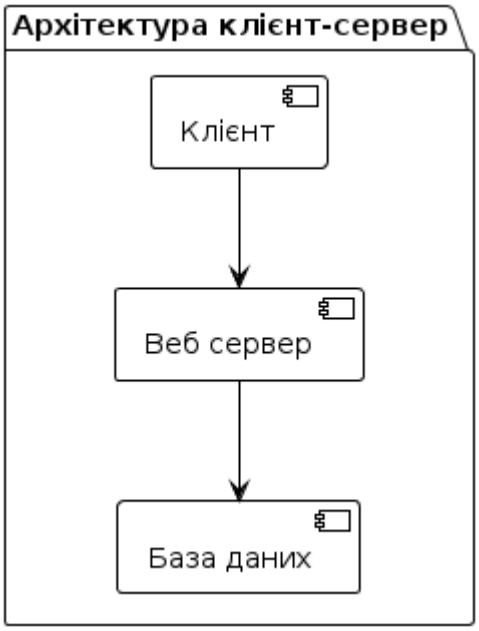

Рисунок 2.3 - Клієнт-серверна архітектура

Взаємодія студентів та викладача при створенні онлайн занять включає кілька ключових етапів. Викладач починає з перевірки розкладу занять, який зазвичай зберігається в електронному форматі або у спеціальному програмному забезпеченні для управління навчальним процесом. Після перегляду розкладу та визначення групи, для якої потрібно створити онлайн заняття, викладач переходить до організації цього заняття.

Перший крок полягає в зборі контактної інформації всіх студентів, які належать до визначеної групи. Викладач використовує список студентів, що містить їхні імейл адреси, зберігаючи цю інформацію у захищеному місці для забезпечення конфіденційності. Викладач може отримати ці дані з університетської бази даних або через навчальну платформу.

Наступним кроком є створення самого онлайн заняття. Викладач вибирає платформу для проведення занять, наприклад, Zoom, Microsoft Teams або Google Meet. Використовуючи інтерфейс обраної платформи, викладач створює нове заняття, задаючи його дату, час та тривалість. До опису заняття можуть бути додані додаткові матеріали або інструкції для студентів.

Після налаштування онлайн заняття викладач запрошує студентів приєднатися. Для цього він використовує зібрані імейл адреси студентів. Викладач надсилає кожному студенту запрошення на заняття, яке містить деталі про час, платформу та посилання для підключення. Такі запрошення можуть бути надіслані безпосередньо через платформу для проведення занять або через університетську систему електронної пошти. Діаграма послідовності дій викладача для створення онлайн зустрічі зображена на рисунку 2.4.

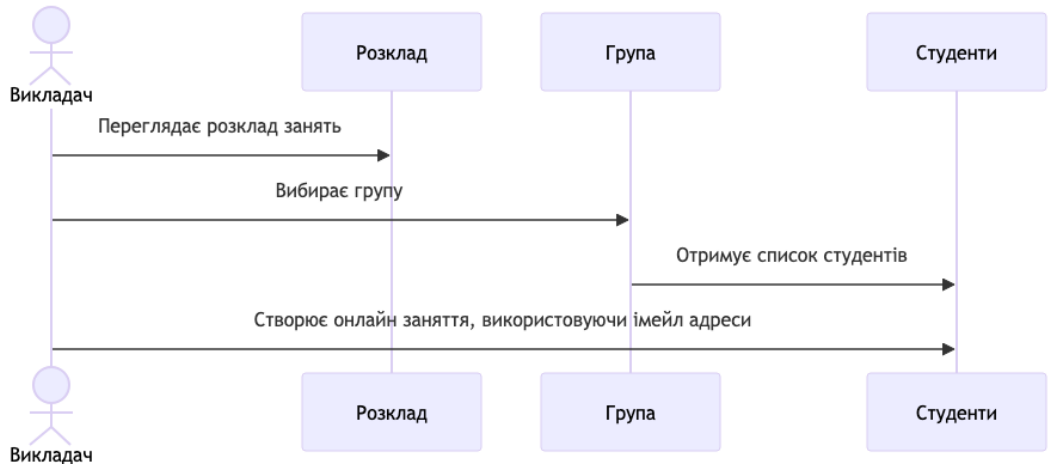

Рисунок 2.4 - Діаграма послідовності дій викладача для створення онлайн зустрічі

Таким чином, процес створення онлайн заняття включає ретельне планування та комунікацію з усіма учасниками, щоб забезпечити ефективне та продуктивне навчання.

Веб-додаток є центральним компонентом системи управління розкладом, через який взаємодіють викладачі та інші користувачі. Він надає інтерфейс для доступу до різних модулів системи, таких як модуль розкладу та модуль студентів. Веб-додаток забезпечує зручний та інтуїтивно зрозумілий спосіб

для користувачів виконувати свої завдання, такі як перегляд розкладу, управління списками студентів та створення нових занять.

Модуль розкладу відповідає за управління розкладом занять. Він зберігає та обробляє інформацію про заняття, такі як дати, часи, викладачі та групи студентів. Цей модуль взаємодіє з базою даних розкладу, де зберігаються всі дані про розклад. Викладачі використовують веб-додаток для доступу до цього модуля, щоб переглядати та оновлювати розклад своїх занять.

База даних розкладу є сховищем для всієї інформації, що стосується розкладу занять. Вона містить дані про заняття, включаючи дати, часи, місця проведення, викладачів та студентські групи. Модуль розкладу використовує цю базу даних для збереження та отримання необхідної інформації для складання та управління розкладом.

Модуль студентів займається управлінням інформацією про студентів. Він зберігає дані про студентів, такі як ім'я, імейл адреси, групи та інші релевантні відомості. Цей модуль взаємодіє з базою даних студентів, де зберігається вся інформація про студентів. Веб-додаток використовує цей модуль для надання викладачам доступу до списків студентів для визначених груп.

База даних студентів є сховищем для всієї інформації, що стосується студентів. Вона містить дані про студентів, включаючи їх імена, імейл адреси, групи та інші важливі відомості. Модуль студентів використовує цю базу даних для збереження та отримання необхідної інформації про студентів, що дозволяє викладачам ефективно управляти списками студентів.

Модуль створення занять відповідає за процес створення нових занять. Він дозволяє викладачам, через веб-додаток, створювати нові заняття, використовуючи списки студентів та їх імейл адреси для організації онлайн занять. Цей модуль взаємодіє з базою даних студентів для отримання необхідної інформації про студентів і взаємодіє з зовнішнім сервісом для автоматичного створення заняття.

На рисунку 2.5 структурна схема системи управління розкладом.

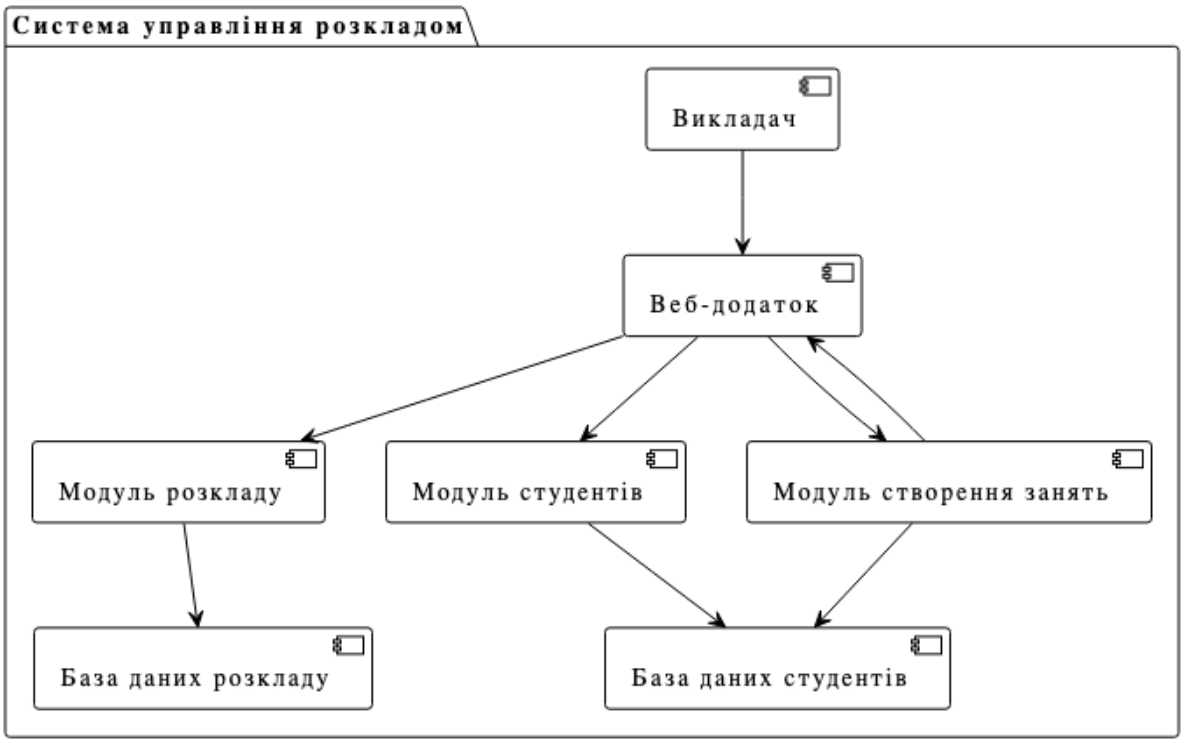

Рисунок 2.5 - Структурна схема системи управління розкладом

Взаємозв'язки між модулями;

- Викладач - Веб-додаток: Викладач використовує веб-додаток для взаємодії з іншими модулями системи.
- Веб-додаток - Модуль розкладу: Веб-додаток взаємодіє з модулем розкладу для надання викладачам можливості переглядати та оновлювати розклад занять.
- Веб-додаток - Модуль студентів: Веб-додаток використовує модуль студентів для доступу до інформації про студентів.
- Модуль розкладу - База даних розкладу: Модуль розкладу зберігає та отримує дані про розклад з бази даних розкладу.
- Модуль студентів - База даних студентів: Модуль студентів зберігає та отримує дані про студентів з бази даних студентів.
- Модуль створення занять - Веб-додаток: Модуль створення занять взаємодіє з веб-додатком для створення нових занять.
- Модуль створення занять - База даних студентів: Модуль створення занять отримує дані про студентів з бази даних студентів для організації нових занять.
- Веб-додаток - Модуль створення занять: Веб-додаток надає інтерфейс для використання модуля створення занять.

На рисунку 2.6 зображено схема взаємодії модулю створення онлайн занять.

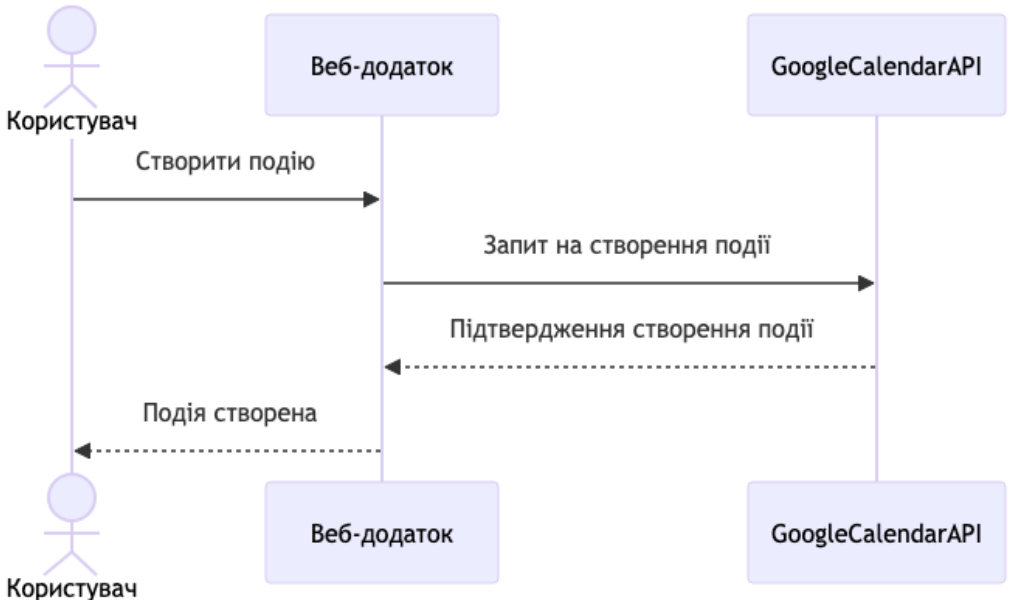

Рисунок 2.6 - Схема взаємодії модулю створення онлайн занять

2.3 Створення веб-сервера

Використання веб-фреймворка значно спрощує та прискорює розробку веб-додатків, забезпечуючи при цьому високий рівень безпеки та стабільності.

Django є високорівневим веб-фреймворком для розробки веб-додатків на мові програмування Python. Він надає зручний і ефективний спосіб створення веб-додатків, де програміст може зосередитися на розробці функціональності, а не на деталях низькорівневого веб-програмування. Основні принципи роботи Django включають:

- Модель-представлення-шаблон (MVT)
- ORM (Об'єктно-реляційне відображення)
- URL-адресація та маршрутизація
- Шаблони
- Адміністративний інтерфейс
- Безпека

Django використовує парадигму MVT, де модель відповідає за дані, представлення - за відображення даних та контролер - за обробку запитів. Модель MVT зображена на рисунку 2.7.

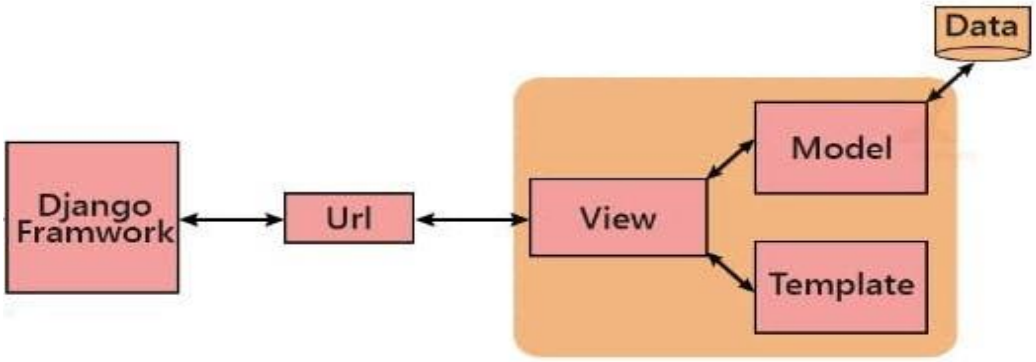

Рисунок 2.7 - Модель MVT в Django

Django надає ORM (Object–relational mapping), який дозволяє вам взаємодіяти з базою даних через високорівневі об'єкти Python, замість прямої роботи з SQL (рисунок 2.8).

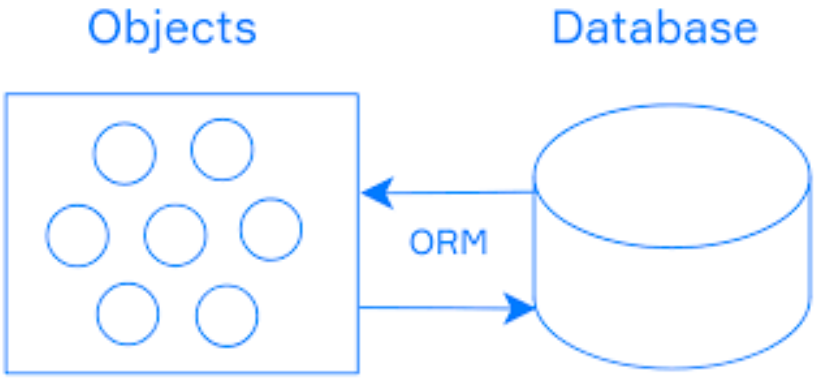

Рисунок 2.8 - Модель ORM в Django

Використання ORM необхідне для опису таблиць студентів та груп у базі даних. Приклад коду зображений на рисунку 2.9.

```python
from django.db import models

# Модель для представлення групи
class Group(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()

    def __str__(self):
        return self.name

# Модель для представлення студента
class Student(models.Model):
    name = models.CharField(max_length=100)
    group = models.ForeignKey(Group, on_delete=models.CASCADE)

    def __str__(self):
        return self.name
```

Рисунок 2.9 - Представлення моделей в середовищі Django

Подальші взаємодії, такі як додання, видалення, оновлення та перегляд, доступні в додатку при використанні класів Student та Group.

Django має систему URL-адресації, яка дозволяє визначати, які частини вашого коду будуть викликані для різних URL-адрес. Щоб користувач міг

звернутися до сервера, буде використано спеціальний шлях, у відповідності до якого виконуватиметься функція управління календарем.

Django використовує шаблони для відображення веб-сторінок, що дозволяє розділити логіку відображення від логіки бізнес-логіки. Головна сторінка веб-сайту в Django описується за допомогою мови HTML, яка може бути розширена за допомогою Django Template Language (DTL). DTL дозволяє вбудовувати динамічний контент, такий як змінні, вирази та логічні конструкції, прямо в HTML-код шаблону (рисунок 2.10). Це необхідно для відображення графічного інтерфейсу, буде будуть елементи взаємодії, такі як кнопка для відправки файлу з розкладом на сервер для подальшої обробки.

```
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    <div class="form-group">
        <label for="id_lecturer_email">Введіть ваш email</label>
        <input type="email" name="lecturer_email" id="id_lecturer_email" required>
    </div>
    <div class="form-group">
        {{ form.file.label_tag }}<br>
        {{ form.file }}
    </div>
    <div class="text-center">
        <button type="submit" class="btn btn-upload">Завантажити</button>
    </div>
</form>
```

Рисунок 2.10 - Інтерфейс користувача

Django постачається з вбудованим адміністративним інтерфейсом, який автоматично створюється на основі ваших моделей даних і дозволяє легко керувати вмістом вашого сайту. Це дає можливість створювати записи студентів, де зазначати Email'и, що необхідні для коректної роботи системи (рисунок 2.11).

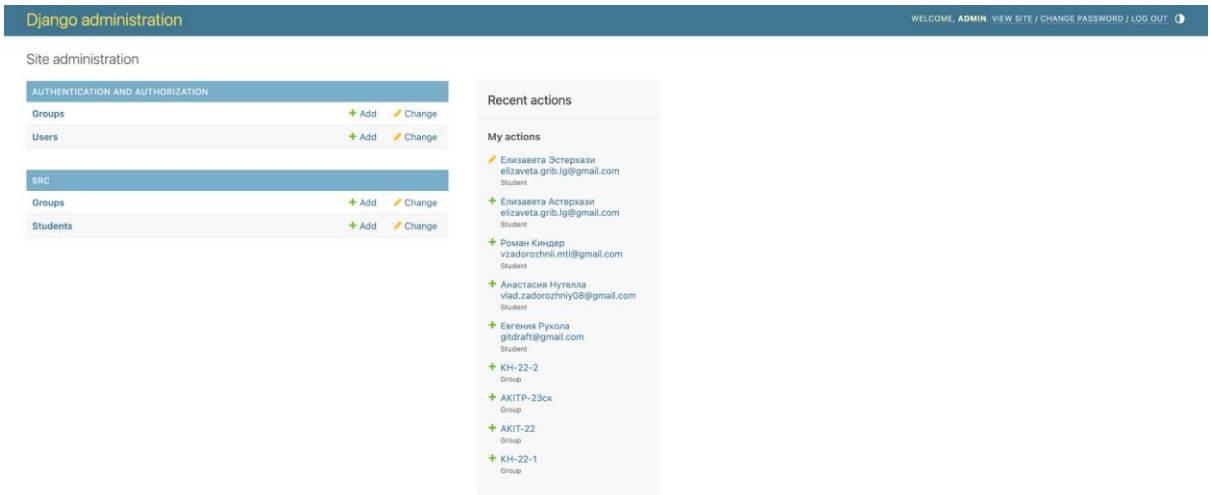

Рисунок 2.11 - Адміністративна панель

2.4 Створення бази даних

Зберігання даних про студентів та групи у базі даних є ключовим для системи управління розкладом. Структура бази даних наступна:

1. Таблиця "Group" (Група):

- В даній таблиці кожен запис представляє одну групу студентів.
- Поля можуть включати:
- ID: Унікальний ідентифікатор групи.
- Назва: Назва групи (наприклад, "Група АКІТ-15").
- Дата створення: Дата створення запису про групу.

2. Таблиця "Student" (Студент):

- Кожен запис цієї таблиці відображає одного студента.
- Поля можуть включати:
- ID: Унікальний ідентифікатор студента.
- Ім'я: Ім'я студента (наприклад, "Василь").
- Прізвище: Прізвище студента (наприклад, "Петренко").
- Електронна пошта: Унікальна електронна адреса студента.
- Група: Зовнішній ключ, що посилається на таблицю "Group", вказує на те, до якої групи належить студент.

Ці дві таблиці взаємопов'язані, оскільки кожен студент належить до певної групи. Використовуючи цю структуру даних, система може ефективно керувати студентами та їх групами, створювати розклади занять для кожної групи та забезпечувати доступ до інформації про студентів та групи для адміністраторів та викладачів.

Зберігання даних про студентів та групи є достатнім для функціонування системи. Використання додаткових таблиць для викладачів, предметів та аудиторій не є необхідним, оскільки метою даної системи є використання розкладу з платформи asu.knu.edu.ua та створення подій в календарі для студентів та викладачів. Це означає, що для подій потрібна лише інформація про аудиторії та назви предметів. Таким чином, вказаними таблицями можна обмежитися, що є достатнім для виконання основної функції системи.

У контексті баз даних можна виділити легковісні та продвинуті бази даних:

1. Легковісні бази даних: Це бази даних, які зазвичай призначені для швидкого розгортання та прототипування. SQLite є прикладом легковісної бази даних, оскільки вона працює на рівні файлової системи і не потребує окремого сервера. Вона чудово підходить для простих або одноразових проектів.

2. Продвинуті бази даних: Ці бази даних мають більші можливості щодо масштабованості, безпеки, функціональності та надійності. PostgreSQL є прикладом продвинутої бази даних, яка надає розширені можливості для управління даними, високої доступності та відмовостійкості.

Використання бази даних PostgreSQL, як вже згадувалося, забезпечить надійність, масштабованість та продуктивність для системи управління розкладом, особливо якщо вона збільшується за обсягом.

Використання PostgreSQL зазвичай має кілька переваг порівняно з легковісними системами баз даних як SQLite3 особливо для веб-додатків, як Django:

1. Масштабованість: PostgreSQL надає кращу підтримку для великих обсягів даних та високої навантаженості. Він може працювати з більшими об'ємами даних та більшою кількістю одночасних підключень.
2. Функціональні можливості: PostgreSQL має розширений набір функцій, таких як декларативні валідації, тригери, відкладені транзакції, вьюшки, регулярні вирази та багато іншого. Це робить його більш потужним у порівнянні з SQLite3.
3. Безпека: PostgreSQL має більше можливостей для управління правами доступу, автентифікацією, аудитом та шифруванням даних, що робить його більш безпечним для використання в різних сценаріях.
4. Підтримка мережевої роботи: PostgreSQL має більш розвинуті засоби роботи з мережею, включаючи реплікацію та кластеризацію, що дозволяє легше масштабувати вашу базу даних.
5. Підтримка типів даних та індексів: PostgreSQL має більш багатий набір типів даних та можливостей індексації, що дозволяє більш ефективно працювати з різними типами даних та запитами.

2.5 Інтеграція з Google Calendar API

Інтеграція з Google Calendar API дозволяє автоматично створювати події у календарі користувача. У вашому коді це реалізовано за допомогою використання бібліотеки googleapiclient, яка забезпечує зв'язок з API Google Calendar, та використання потоків OAuth для автентифікації.

Google Console - це веб-інтерфейс, розроблений Google, який надає доступ до різних сервісів та інструментів Google Cloud Platform (GCP), включаючи Google Calendar API. Він дозволяє реєструвати додатки, отримувати API ключі, керувати доступом та налаштовувати інші параметри для використання сервісів Google.

Для взаємодії з Google Calendar API необхідно отримати файл з даними доступу до сервіс аккаунта Google.

Отримання credentials.json з Google Console та пояснення вмісту цього файлу:

1. Отримання credentials.json з Google Console:

- Треба увійти до Google Cloud Console.
- Створити новий проект або вибрати вже існуючий.
- У меню бокової панелі перейти до "APIs & Services" > "Credentials".
- Натиснути кнопку "Create credentials" і обрати "Service account key".
- Вибрати або створити сервісний обліковий запис, обрати роль "Project > Editor" та формат ключа JSON.
- Натиснути "Create" і зберегти згенерований credentials.json.

2. Пояснення вмісту credentials.json:Файл credentials.json містить інформацію, необхідну для автентифікації вашого додатка в Google сервісах. Ось основні поля, які зазвичай знаходяться в цьому файлі:

- type: Тип облікового запису. Зазвичай це "service_account".
- project_id: ID вашого проекту в Google Cloud.
- private_key_id: Унікальний ідентифікатор приватного ключа.
- private_key: Приватний ключ, який дозволяє вашому додатку автентифікуватися у Google API.
- client_email: Email облікового запису сервісу.
- client_id: Унікальний ідентифікатор клієнта.
- auth_uri, token_uri: URI для отримання авторизації та токену доступу.
- auth_provider_x509_cert_url: URL-адреса, що містить сертифікати авторизації.
- client_x509_cert_url: URL-адреса публічного ключа клієнта.

Основні кроки інтеграції з Google Calendar API у коді додатку:

1. Автентифікація: Здійснюється за допомогою файлу credentials.json. Після автентифікації отримується доступ до служби Google Calendar через об'єкт service.
2. Створення подій у календарі: Кожна подія, яку потрібно додати, представляється словником з необхідними даними, такими як назва, опис, початковий та кінцевий час, учасники тощо. Ці дані передаються до методу events().insert().
3. Часові зони: Для відображення правильних часів подій необхідно встановлювати правильну часову зону. Використовується часова зона 'Europe/Kyiv'.

Цей код демонструє базовий приклад інтеграції з Google Calendar API, але можна розширити його функціонал для відповідності конкретним потребам вашого проекту, наприклад, додавання додаткових полів подій, обробка помилок тощо.

2.6 Розробка коду алгоритму системи

Основна робота система розділена на частини:

1. Витяг даних розкладу для викладача
2. Формування подій відповідно до даних з розкладу
3. Використання Google Calendar API для створення подій для студентів.

Витяг даних з файлів Excel у форматі xlsx використовується бібліотека pandas, яка є однією з найпоширеніших бібліотек для обробки та аналізу даних у Python. За допомогою функції pd.read_excel(), що надається бібліотекою pandas, дані з Excel файлу зчитуються безпосередньо у DataFrame, який є основною структурою даних у pandas.

DataFrame дозволяє легко працювати з даними у вигляді таблиць, а методи та функції pandas роблять роботу з даними більш зручною та

ефективною. В коді використовуються методи DataFrame, такі як iloc[], щоб отримати доступ до конкретних рядків та стовпців, і функція shape, щоб отримати розміри DataFrame.

Визначення днів тижня та номерів пар виконується за допомогою попередньо створених списків, які містять дні тижня (у цьому випадку, українські назви днів тижня) та номери пар. Вони використовуються для ідентифікації відповідних рядків та стовпців у файлі Excel.

Функція is_class_number перевіряє, чи містить комірка номер пари.

Функція extract_schedule_data:

- Ця функція отримує шлях до файлу Excel і використовує бібліотеку pandas, щоб прочитати дані з цього файлу.

- Потім вона ітерується по кожному рядку у DataFrame, шукаючи рядки, що містять дні тижня.

- Після знаходження рядків, що містять дні тижня, функція визначає дати та інформацію про заняття, що відповідають кожному дню тижня.

- За допомогою функції is_class_number, вона визначає, чи містить комірка номер пари.

- Якщо умови виконуються, дані про заняття додаються до списку extracted_data у вигляді словників з ключами "date", "class" та "content".

- На завершення функція повертає витягнуті дані у вигляді списку словників.

Цей код ефективно витягує дані з Excel файлу з розкладом у форматі xlsx, створюючи зручну структуру даних для подальшого використання.

Опис функціоналу базується на аналізі файлів views.py, forms.py, models.py, calendar_events.py та parse_schedule.py, які складають основну частину проєкту.

1. src/models.py

Модель даних:

- Student: Зберігає інформацію про студентів, включаючи їхні контактні дані та групи.

- Event: Містить дані про події, такі як лекції, семінари, лабораторні роботи, з датою, часом та описом.

- Group: Відображає студентські групи, до яких належать студенти.

2. src/forms.py

Форми:

- EventForm: Форма для створення та редагування подій. Включає поля для назви події, дати, часу та опису.

- ScheduleUploadForm: Форма для завантаження розкладу у форматі Excel, що буде парситися та додаватися до системи.

3. src/views.py

Представлення:

- upload_file_view: Відповідає за обробку завантаження файлу з розкладом та електронної пошти викладача. Після завантаження та перевірки файлу, дані передаються у функцію create_calendar_events.

4. src/utils/calendar_events.py

Інтеграція з календарем:

- add_event_to_calendar: Функція для додавання події до календаря Google. Використовує Google Calendar API для створення події в календарі користувача.

- remove_event_from_calendar: Видаляє подію з календаря Google.

- update_event_in_calendar: Оновлює дані про подію в календарі Google, якщо було внесено зміни.

5. src/utils/parse_schedule.py

Парсинг розкладу:

- parse_schedule: Функція для парсингу файлу розкладу у форматі Excel. Зчитує дані з файлу, формує події та додає їх до системи.

- validate_schedule_format: Перевіряє правильність формату завантаженого файлу розкладу.

- extract_event_data: Витягує дані про події з файлу розкладу та перетворює їх у формат, придатний для додавання до бази даних.

Користувачі мають можливість завантажувати файл з розкладом у форматі Excel через веб-інтерфейс. Форма ScheduleUploadForm дозволяє вибрати файл та завантажити його на сервер. Функція parse_schedule обробляє файл, витягує дані про події та зберігає їх у базі даних.

Проєкт підтримує інтеграцію з календарем Google. Функції з calendar_events.py дозволяють автоматично додавати, оновлювати та видаляти події в календарі Google користувача. Це забезпечує синхронізацію розкладу з особистими календарями студентів та викладачів, полегшуючи управління часом.

Увесь код міститься в Github репозиторії - https://github.com/monteri/student_events. Також навений функціонал представлений у додатках А, Б, В, Г, Д.

2.7 Візуалізація роботи системи

Вхід на сторінку /upload. Перехід на сторінку завантаження:

1. Користувач відкриває сторінку /upload, де відображається форма для завантаження файлу розкладу у форматі XLSX та введення електронної пошти викладача. Інтерфейс сторінки включає поле для завантаження файлу та поле для введення електронної пошти (рисунок 2.13)

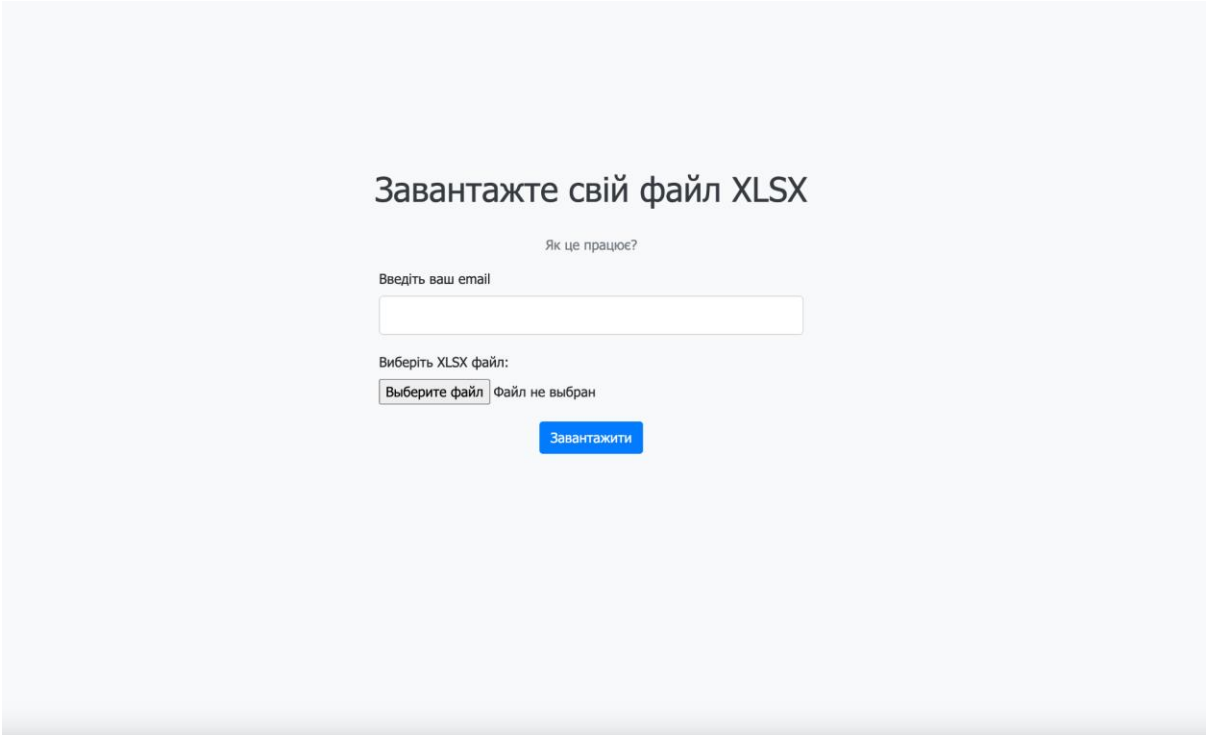

Рисунок 2.13 - Головна сторінка.

При натисканні на текст “Як це працює”, висвічується повідомлення, зображене на рисунку 2.14. Підказка надає інформацію про функціонал системи.

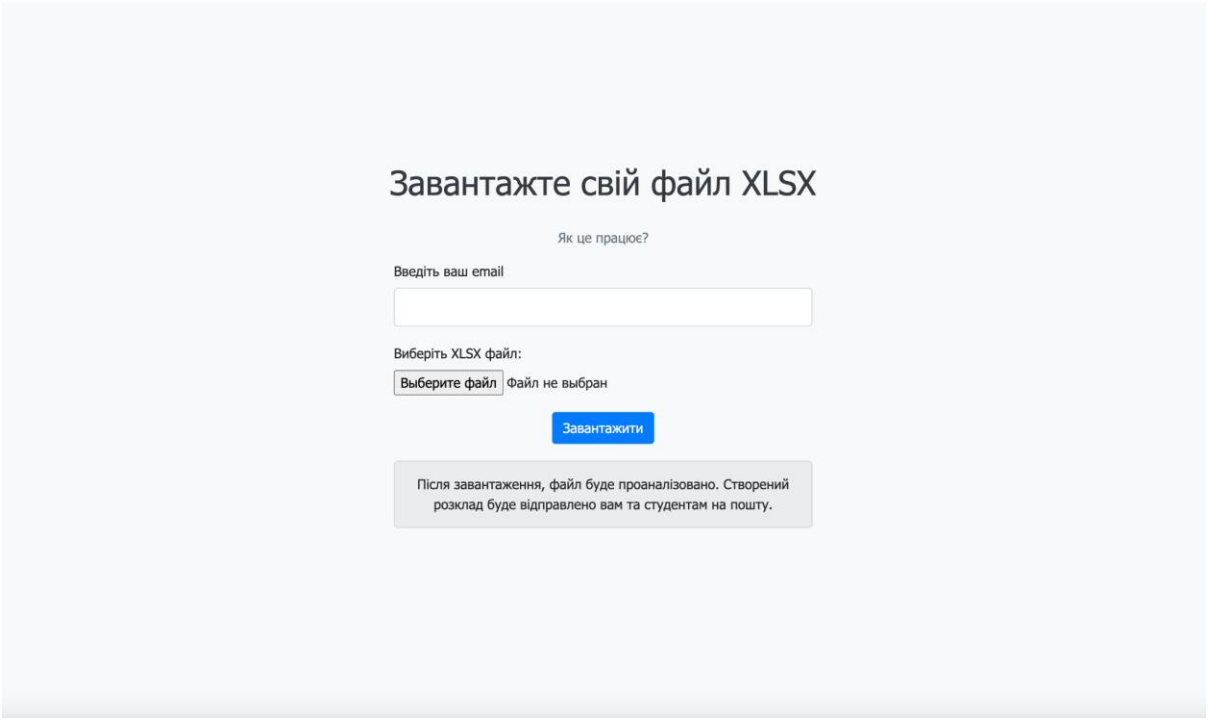

Рисунок 2.14 - Підказка про функціонал сторінки.

2. Заповнення форми. Користувач вводить свою електронну пошту у відповідне поле та вибирає файл розкладу на своєму комп'ютері за

допомогою кнопки завантаження файлу (рисуноку 2.15). Після заповнення форми користувач натискає кнопку "Завантажити".

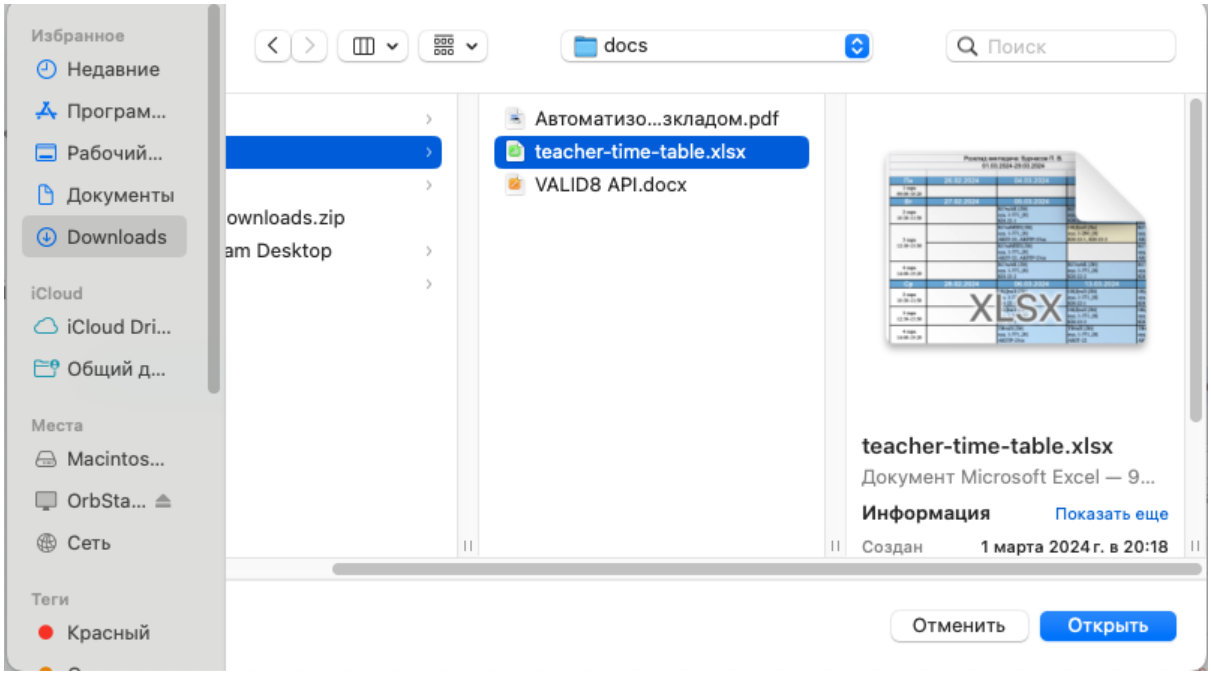

Рисунок 2.15 - Діалогове вікно вибору файлу

3. Валідація форми. Сервер перевіряє коректність введених даних. Якщо форма заповнена правильно, файл та електронна пошта передаються у функцію create_calendar_events для подальшої обробки.

4. Завантаження та обробка файлу. Функція extract_schedule_data обробляє завантажений файл розкладу. За допомогою бібліотеки pandas дані зчитуються з Excel файлу у DataFrame. Функція ітерується по кожному рядку DataFrame, шукаючи дні тижня та номери пар. Визначаються дати та інформація про заняття, які відповідають кожному дню тижня.

5. Створення подій у календарі. Після витягу даних з файлу, функція create_calendar_events створює події у Google Calendar для кожного заняття. Вона обчислює час початку та завершення занять, визначає групи студентів за допомогою функції retrieve_group_names та отримує їхні електронні адреси з бази даних через функцію get_students_emails_by_groups. Потім події створюються у Google Calendar з додаванням електронної пошти викладача як учасника.

6. Інтеграція з календарем Google та додавання подій до календаря. Використовуючи Google Calendar API, функція create_calendar_events створює події у календарі викладача та студентів. Події включають деталі заняття, такі як дата, час, назва та учасники (викладач та студенти).

7. Підтвердження завантаження. Після успішного створення подій у календарі, користувач перенаправляється на сторінку підтвердження завантаження, де відображається повідомлення про успішне завантаження файлу та створення подій (рисунок 2.16).

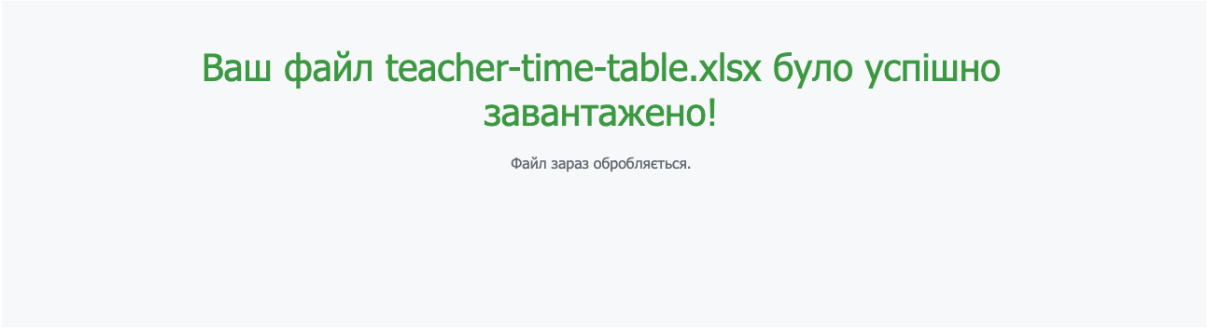

Рисунок 2.16 - Повідомлення про успішне завантаження файлу

Висновок до розділу 2

Розроблено алгоритм, який зчитує дані з файлів Excel, аналізує їх і формує масив подій з інформацією про дату, час, назву заняття та групу. Це забезпечує точне та ефективне витягування і обробку даних розкладу, що дозволяє створювати актуальні розклади занять.

Розроблено структурну схему клієнт-серверної архітектури. Використано клієнт-серверну архітектуру для розподілу навантаження між клієнтською та серверною частинами, забезпечуючи ефективну обробку даних. Архітектура забезпечує високу продуктивність та гнучкість системи, дозволяючи користувачам зручно взаємодіяти з системою через веб-додаток.

Вибрано Django як основний фреймворк для розробки системи. Використано Django для створення веб-сервера, що дозволяє швидко і ефективно розробляти функціональність системи. Django забезпечує стабільність, безпеку і зручність розробки веб-додатків, що підвищує ефективність системи.

Реалізовано інтеграцію з Google Calendar API. Використано бібліотеки Python та OAuth для автентифікації і взаємодії з Google Calendar API, що дозволяє автоматично створювати події в календарях користувачів. Інтеграція забезпечує синхронізацію розкладу з календарями користувачів, роблячи управління розкладом більш зручним і ефективним.

Написано код для витягу даних з файлів Excel. Використано бібліотеку pandas для зчитування даних з Excel файлів, а також створено функції для обробки і формування подій з даних розкладу. Код забезпечує ефективне витягування і обробку даних розкладу, що дозволяє автоматизувати процес створення і оновлення розкладів занять.

Розроблено інтерфейс для взаємодії користувачів з системою. Використано Django для створення веб-інтерфейсу, який дозволяє викладачам і студентам переглядати та редагувати розклади занять. Інтерфейс забезпечує зручний доступ до функціональності системи, підвищуючи її зручність та ефективність використання.

Описано роботу системи, включаючи всі її компоненти та модулі, їх взаємодію та загальну архітектуру.

ВИСНОВОК

У результаті проведеного дослідження було виконано розробку автоматизованої системи управління розкладом занять для навчальних закладів, що відповідає сучасним вимогам онлайн-навчання. Ця система дозволяє автоматизувати процес створення, редагування та моніторингу розкладів занять, що значно підвищує ефективність управління навчальним процесом.

Проведено комплексний аналіз існуючих підходів до управління розкладом занять, особливо в умовах онлайн-навчання. Оцінено сучасні платформи для проведення онлайн-занять, зокрема Zoom, Microsoft Teams, Google Meet та Cisco Webex, і визначено їх основні функції та можливості. На основі цього аналізу було визначено ключові вимоги до нової автоматизованої системи управління розкладом.

Проектування системи включає визначення функціональних та нефункціональних вимог, розробку архітектури системи та вибір технологій для її реалізації. Було використано клієнт-серверну архітектуру для забезпечення високої продуктивності та гнучкості системи. Для розробки веб-додатку було обрано фреймворк Django, який забезпечує стабільність, безпеку та зручність розробки. Інтеграцію з Google Calendar API було реалізовано для автоматичного створення подій у календарях користувачів. Для витягу даних з файлів Excel використано бібліотеку pandas, що забезпечує ефективне зчитування та обробку даних розкладу.

Розроблена система дозволяє створювати та редагувати розклади занять, автоматично інтегруватися з платформами для онлайн-зустрічей, надсилати сповіщення та нагадування користувачам, а також синхронізувати розклад з календарями користувачів. Це значно спрощує процес управління навчальним процесом, забезпечуючи його гнучкість та адаптивність до індивідуальних потреб викладачів та студентів. Система також забезпечує високу безпеку даних, продуктивність та надійність, що робить її ефективним інструментом для навчальних закладів у сучасному освітньому середовищі.

Таким чином, розроблена автоматизована система управління розкладом занять відповідає сучасним вимогам та потребам освітнього процесу, забезпечуючи ефективне та зручне управління розкладом як для викладачів, так і для студентів. Впровадження цієї системи сприятиме покращенню організації навчального процесу, знижуючи ризик помилок та забезпечуючи більш ефективне використання часу та ресурсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джонсон Т. Effective Project Management. Лондон: McGraw-Hill, 2021. 320 с.
2. Іваненко О. А. Проектування інформаційних систем: підручник. Запоріжжя: ЗНТУ, 2020. 376 с.
3. Google Developers. Google Calendar API Documentation [Електронний ресурс]. Google Developers, 2021. Режим доступу: https://developers.google.com/calendar
4. Даниленко С. Сучасні методи аналізу даних: навчальний посібник. Тернопіль: ТНЕУ, 2020. 304 с.
5. OpenAI. An Overview of Artificial Intelligence [Електронний ресурс]. OpenAI Blog, 2021. Режим доступу: https://openai.com/blog/overview-artificial-intelligence/
6. Козакова І. А. Інформаційні технології в освіті: монографія. Луцьк: Вежа-Друк, 2019. 310 с.
7. Python Software Foundation. Python Official Documentation [Електронний ресурс]. Python.org, 2021. Режим доступу: https://docs.python.org/3/
8. Вільямс Р. Web Development with Django. Сан-Франциско: No Starch Press, 2020. 290 с.
9. Stack Overflow. Best Practices for Web Development [Електронний ресурс]. Stack Overflow, 2021. Режим доступу: https://stackoverflow.blog/2021/01/14/best-practices-for-web-development/
10. Іваненко О. А. Проектування інформаційних систем: підручник. Запоріжжя: ЗНТУ, 2020. 376 с.
11. GitHub. Open Source Projects for Education [Електронний ресурс]. GitHub, 2020. Режим доступу: https://github.com/topics/education
12. Максименко А. В. Машинне навчання та аналіз даних: підручник. Львів: ЛНУ, 2019. 320 с.
13. Coursera. Data Science Specialization [Електронний ресурс]. Coursera, 2021. Режим доступу: https://www.coursera.org/specializations/jhu-data-science
14. Ніколаєнко І. В. Інформаційні системи: проектування та використання: навчальний посібник. Чернівці: ЧНУ, 2018. 384 с.

15. W3Schools. HTML and CSS Tutorials [Електронний ресурс]. W3Schools, 2021. Режим доступу: https://www.w3schools.com/

16. Романенко В. П. Автоматизація управління проектами: монографія. Суми: СумДУ, 2019. 400 с.

17. Khan Academy. Computer Programming Course [Електронний ресурс]. Khan Academy, 2020. Режим доступу: https://www.khanacademy.org/computing/computer-programming

18. Купер Дж. Mastering Machine Learning with Python. Чикаго: O'Reilly Media, 2020. 370 с.

19. Карпенко Д. В. Програмування веб-додатків: підручник. Одеса: ОНПУ, 2021. 340 с.

20. Казакова І. А. Інформаційні технології в освіті: монографія. Луцьк: Вежа-Друк, 2019. 310 с.

21. OpenAI. An Overview of Artificial Intelligence [Електронний ресурс]. OpenAI Blog, 2021. Режим доступу: https://openai.com/blog/overview-artificial-intelligence/

22. Google Developers. Google Calendar API Documentation [Електронний ресурс]. Google Developers, 2021. Режим доступу: https://developers.google.com/calendar

23. Khan Academy. Computer Programming Course [Електронний ресурс]. Khan Academy, 2020. Режим доступу: https://www.khanacademy.org/computing/computer-programming

24. GitHub. Open Source Projects for Education [Електронний ресурс]. GitHub, 2020. Режим доступу: https://github.com/topics/education

25. Python Software Foundation. Python Official Documentation [Електронний ресурс]. Python.org, 2021. Режим доступу: https://docs.python.org/3/

26. Stack Overflow. Best Practices for Web Development [Електронний ресурс]. Stack Overflow, 2021. Режим доступу: https://stackoverflow.blog/2021/01/14/best-practices-for-web-development/

27. GitHub. Open Source Projects for Education [Електронний ресурс]. GitHub, 2020. Режим доступу: https://github.com/topics/education

28. W3Schools. HTML and CSS Tutorials [Електронний ресурс]. W3Schools, 2021. Режим доступу: https://www.w3schools.com/

29. Іваненко О. А. Проектування інформаційних систем: підручник. Запоріжжя: ЗНТУ, 2020. 376 с.

30. Романенко В. П. Автоматизація управління проектами: монографія. Суми: СумДУ, 2019. 400 с.

ДОДАТОК А

views.py

```
from django.shortcuts import render
from src.utils.calendar_events import create_calendar_events
from .forms import XlsxUploadForm

def upload_file_view(request):
    if request.method == 'POST':
        form = XlsxUploadForm(request.POST, request.FILES)
        if form.is_valid():
            file = request.FILES['file']
            lecturer_email = form.cleaned_data['lecturer_email']
            create_calendar_events(file, lecturer_email)
            return render(request, 'success.html', {'filename': file.name})
    else:
        form = XlsxUploadForm()

    return render(request, 'upload.html', {'form': form})
```

ДОДАТОК Б

forms.py

```
from django import forms


class XlsxUploadForm(forms.Form):
   lecturer_email = forms.EmailField(label="Введіть ваш email")
   file = forms.FileField(label="Виберіть XLSX файл")
```

models.py

```
from django.db import models

class Group(models.Model):
    name = models.CharField(max_length=255, unique=True)

    def __str__(self):
        return self.name

class Student(models.Model):
    name = models.CharField(max_length=255)
    email = models.EmailField(unique=True)
    group = models.ForeignKey(Group, on_delete=models.CASCADE)

    def __str__(self):
        return f'{self.name} {self.email}'

def get_students_emails_by_groups(group_names):
    try:
        groups = Group.objects.filter(name__in=group_names)
        students = Student.objects.filter(group__in=groups)
        emails = [student.email for student in students]
        return emails
    except Group.DoesNotExist:
        return []
```

parse_schedule.py

```
import pandas as pd

days_of_week = ["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Нд"]
class_numbers = ["1 пара", "2 пара", "3 пара", "4 пара", "5 пара"]


# Function to check if a cell contains a class number
def is_class_number(cell):
    return any(cn in cell for cn in class_numbers)


def extract_schedule_data(file):
    """Extracts schedule data from an Excel file.

    Args:
        file_path (str): The path to the Excel file.

    Returns:
        list: A list of dictionaries containing the extracted schedule data.
    """

    df = pd.read_excel(file)
    extracted_data = []
    # Iterate through the dataframe
    for i in range(df.shape[0]):
        row = df.iloc[i]

        # Check if the first cell of the current row is a day of the week
        if pd.notnull(row[0]) and row[0] in days_of_week:

            j = i + 1  # Start looking from the next row

            # Loop until the next valid day of the week is found
            while j < df.shape[0] and not (pd.notnull(df.iloc[j][0]) and
df.iloc[j][0] in days_of_week):
                j += 1

            if j == df.shape[0]:  # Reached end of DataFrame
                break

            # Iterate through the columns starting from the second column
            for col in range(1, df.shape[1]):
                date = row[col]

                # Check for valid dates and class information across
potentially multiple rows
                for k in range(i + 1, j):
                    class_info = df.iloc[k][0]
                    if pd.notnull(date) and pd.notnull(class_info) and
is_class_number(class_info):
                        content = df.iloc[k][col]
                        if pd.notnull(content):
                            extracted_data.append({
                                "date": date,
                                "class": class_info.split('\n')[0],
                                "content": content
                            })
    return extracted_data
```

calendar_events.py

```
import uuid
import datetime
from zoneinfo import ZoneInfo
from googleapiclient.discovery import build
from httplib2 import Http
from oauth2client import file, client, tools

from src.models import get_students_emails_by_groups
from src.utils.parse_schedule import extract_schedule_data

SCOPES = 'https://www.googleapis.com/auth/calendar'
store = file.Storage('token.json')
creds = store.get()
if not creds or creds.invalid:
    flow = client.flow_from_clientsecrets('credentials.json', SCOPES)
    creds = tools.run_flow(flow, store)
service = build('calendar', 'v3', http=creds.authorize(Http()))
timezone = 'Europe/Berlin'


def retrieve_group_names(input_string):
    last_line = input_string.rsplit('\n', 1)[-1]
    group_names = last_line.split(',')
    truncated_group_names = [name.strip() for name in group_names]

    return truncated_group_names


def create_calendar_events(file, lecturer_email):
    # Load the spreadsheet
    extracted_data = extract_schedule_data(file)
    print(extracted_data)

    for event_data in extracted_data[:2]:
        # 1. Parse the data
        date_str = event_data['date']
        class_time = event_data['class']
        content = event_data['content']

        # 2. Extract event start/end times (modify based on 'class' if
needed)
        start_datetime, end_datetime = calculate_event_times(date_str,
class_time)
        groups = retrieve_group_names(content)
        emails = get_students_emails_by_groups(groups)

        # 3. Add lecturer email to the list of attendees
        emails.append(lecturer_email)

        # 4. Build the event
        event_info = {
            'summary': content,
            'description': 'A class scheduled automatically',
            'start': {
                'dateTime': start_datetime.isoformat(),
                'timeZone': timezone,
            },
            'end': {
```

```
                'dateTime': end_datetime.isoformat(),
                'timeZone': timezone,
            },
            'attendees': [{'email': email} for email in emails],
            'conferenceData': {
                'createRequest': {
                    'requestId': str(uuid.uuid4()),
                },
            },
        }
        print('event', event_info)
        # 5. Create the event
        event = service.events().insert(calendarId='primary',
body=event_info, conferenceDataVersion=1).execute()
        print(f'Event created: {event.get("htmlLink")}')


def calculate_event_times(date_str, class_time):
    """Calculates event start and end times based on the date and class
information.

    Args:
        date_str: The date as 'DD.MM.YYYY'
        class_time: The class time information (e.g., "3 пара")

    Returns:
        A tuple of (start_datetime, end_datetime) as datetime objects.
    """

    # Convert the date string into a datetime object
    date = datetime.datetime.strptime(date_str, '%d.%m.%Y').date()

    # Add the start time based on your class schedule logic
    # Example mapping (adjust as needed):
    class_start_times = {
        '1 пара': datetime.time(9, 0),
        '2 пара': datetime.time(10, 30),
        '3 пара': datetime.time(12, 30),
        '4 пара': datetime.time(14, 0),
        '5 пара': datetime.time(15, 30)
    }
    start_time = class_start_times.get(class_time)

    # Combine date and start time into a datetime object
    start_datetime = datetime.datetime.combine(date, start_time,
tzinfo=ZoneInfo(timezone))

    # Now you can add the timedelta to get the end time
    end_datetime = start_datetime + datetime.timedelta(minutes=90)

    return start_datetime, end_datetime
```