ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»

Факультет комп'ютерно-інформаційних технологій

Кафедра комп'ютерних інформаційних систем і технологій

Абрамов Андрій Олександрович

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему**: «Розробка веб-застосунку для автоматизованого формування та візуалізації звітних документів у форматі PDF»**

Група: ІК-9-22-Б1ІПЗ(4.0д)

Спеціальність: Інженерія програмного забезпечення

Рівень вищої освіти: перший (бакалаврський)

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

Науковий керівник _____

Допущено до захисту ДЕК

Завідувач кафедри _____ Сергій КАВУН , д.е.н., професор

(підпис)

Київ, 2026

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 64 с., 17 рис., 2 додатки, 36 джерел.

ВІЗУАЛІЗАЦІЯ ДАНИХ, ФОРМАТ PDF, АВТОМАТИЗАЦІЯ ЗВІТНОСТІ, ПАРСИНГ ДАНИХ, ГЕНЕРАЦІЯ ДОКУМЕНТІВ, ДИНАМІЧНЕ МАКЕТУВАННЯ, СТРУКТУРА РОЗКЛАДУ, PYTHON, FLASK, REPORTLAB.

Об’єктом дослідження виступають методи автоматичного форматування даних для подальшого перетворення у візуальне представлення.

Метою роботи є розробка та впровадження програмного модуля у складі вебзастосунку для автоматизованого формування, та візуалізації навчального розкладу у форматі PDF

Методи розробки базуються на Python з використанням мікрофреймворку Flask та бібліотек ReportLab з використанням бази даних MySQL.

Результатом роботи є веб-застосунок з функцією автоматизації створення розкладу для університетів та подальшого експорту у PDF документ.

DATA VISUALIZATION, PDF FORMAT, REPORTING AUTOMATION, DATA PARSING, DOCUMENT GENERATION, DYNAMIC LAYOUT, SCHEDULE STRUCTURE, PYTHON, FLASK, REPORTLAB.

The object of the research is the methods of automatic data formatting for subsequent transformation into visual representations.

The goal of the work is to develop and implement a software module within a web application for the automated generation, and visualization of academic schedules in PDF format

The development methods are based on Python using the Flask microframework and the ReportLab library, integrated with a MySQL database.

The result of the work is a web application featuring an automated schedule creation function for universities with subsequent export to PDF documents.

ЗМІСТ

ВСТУП

В різних сферах, від суспільства до політики і економіки, потребують організації роботи та їх бізнес-процесів. І з кожним кроком сучасний світ, зі швидким темпом розвитку технологій, створює нові винаходи для полегшення життєвого циклу через автоматизацію процесів, завдяки чому з’явились каси самообслуговування, безпілотний транспорт, роботи-пилососи та штучний інтелект. Але потребою цього залишається організація плану роботи, яка в більшості випадків робиться власноруч.[1,2,3]

У сфері освіти така проблема доволі “гостра”, витрати часу на організацію роботи збільшуються разом з обсягом документообігу, і це призводить до затримки або помилок, а вимоги до звітності стають складнішими. Додавання подібних систем для роботи з документами в електронному вигляді з подальшим документообігом підвищує ефективність управління навчальними процесами. Робота з інформацією значно пришвидшується, зменшується робота з папером та стають достовірнішими через меншу ймовірність помилок за людським фактором. Все це створює зручні умови для кращого управління навчальним та робочими процесами.[1,2,3]

На даний період часу трудомістко створювати документи звітності через ручну обробку даних, це потребує значних часових витрат. Через те що доводиться опиратися виключно на людську працю, це призводить до низької оперативності та додає більшого шансу на помилки в документі, наприклад через втому, які необхідно мінімізувати, адже доведеться переробляти потім, тим самим призводячи до витрат ресурсів та годин. І в цей самий момент сучасні інформаційні системи накопичують багато даних, зокрема: дисципліни, їх початок та кінець, групи та викладачі. Звідси і виникає потреба в інструменті для представлення у зручному для читання вигляді звітності. Формат, який є одним із загальних стандартів, є формат PDF, в ньому інформація ніяк не зміниться навмисно, тим самим зручно переглядати на будь-якому програмно-апаратному середовищі.[1,2,3]

У кваліфікаційній роботі буде розглядатись проблема, яка пов’язана з ефективністю в засобах роботи з великою кількістю даних в базах інформаційних систем, та оформлення їх в офіційні документи, з дотриманням суворих державних стандартів. Але більшість наявних рішень добре функціонують з простими даними, але недостатньо гнучкі при роботі з більшою кількістю масивів, або їх складно інтегрувати у веб, бо більшість застосунків є застарілими і вони існують як десктопні. Опираючись на цю інформацію, ми отримуємо потребу у веб-інструменті, який буде здатний здійснювати динамічну генерацію складних звітів, на основі актуальних даних. [1,2,3]

Актуальність теми дослідження спрямована на вирішення проблеми ручного створення звітів, переходом на інструмент для автоматичного форматування у вигляді PDF-документів. Рішення повинно бути ефективним, масштабованим, здатним працювати на різних системах, і на комп’ютері, і на телефоні, та при цьому всьому враховувати специфіку української мови та складність табличної верстки. Для цього завдання треба обрати такий технологічний стек, який буде гнучким в плані архітектури мікрофреймворку, мати професійну верстку та разом бути синергією серверної мови, і вибір впав на мову Python з фреймворком Flask, та на бібліотеку ReportLab.[1]

Об’єктом дослідження виступають методи автоматичного форматування даних для подальшого перетворення у візуальне представлення у вебсистемах.

Предметом дослідження є методи, алгоритми і архітектурні патерни генерації документів у формат PDF за допомогою бібліотек ReportLab, Flask і ReportLab.

Метою кваліфікаційної роботи є розробка веб-застосунку для автоматичного створення розкладів, та їх підготовки у формат PDF. Система повинна забезпечити високу швидкість опрацювання даних, дотримуючись стандартів ДСТУ і водночас правильно відобразити складні структури даних у читабельному вигляді.[5]

Для досягнення мети поставлено наступні завдання:

1. Виконати аналіз стану автоматизації документообігу в Україні та визначити проблеми генерації звітності.

2. Обґрунтувати необхідність автоматизації процесів формування звітності.

3. Дослідити та проаналізувати вимоги державних стандартів до оформлення документів.

4. Зробити порівняльний аналіз сучасних бібліотек та підходів до створення PDF-файлів

5. Визначити вимоги і сценарії взаємодії користувача та розробити архітектурне проектування веб-застосунку .

6. Спроектувати та описати алгоритми динамічного формування структури документа, обробки таблиць та візуалізації даних.

7. Дослідити питання оптимізації продуктивності та розробити методику тестування коректності генерації бінарних файлів.

Практичне значення: Створення веб-застосунку для автоматизації формування розкладу для університетів з функцією експорту у PDF документ. Технічні рішення з використанням бібліотек Flask та Reportlab можуть бути використовані для впровадження в сучасні системи цифрового документообігу ЗВО

РОЗДІЛ 1.ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ АСПЕКТИ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ ЗВІТНОСТІ

1.1 Дослідження процесів автоматизації документообігу в освітніх закладах

Впровадження сучасних цифрових технологій - тема яка на даний період часу стає з кожним днем актуальнішою, як для державного управління так і для бізнесів та освіти, люди працюють з документообігом і для цього виникає потреба в покращенні цього процесу, інструмент, який автоматизує роботу з шаблонними документами. Це також пов’язано з євроінтеграційним курсом України, тим самим передбачає адаптацію національного законодавства до норм ЄС. Одним із напрямів державної політики є “Цифрова держава”, пов’язаний з європейським регламентом eIDAS, це передбачає перехід до режиму “paperless”, тобто повну відмову від паперу і перехід на технологічний напрям взаємодії.[6]

Кроки до оптимізації документообігу, для полегшення етапів роботи, від створення документа до його виконання і відправки в архів, спонукають до створення кращих умов організації праці, тим самим можна сказати, що все це робиться не тільки для того щоб позбутися паперових клопотів. Для закладу вищої освіти такі зміни автоматизації важливі, розклад - це основа для всіх викладачів з різними предметами, які вони викладають та студентів, які мають різні профілі та курси.[2]

Якщо говорити про впровадження цифрових технології у навчальні заклади, то це призводить до формування цілісного середовища, якщо раніше на планування навчального процесу витрачалось багато часу на ручну обробку даних та паперове оформлення, то завдяки цифровізації дані автоматично оформлюватимутся, та це пришвидшить процес набуття статусу офіційного документу, рутина праця зведена до мінімуму, з’являється більше часу на підвищення якості освітнього процесу.[2,4]

Впровадження механізмів автоматичної генерації спонукає покращенням роботи та робить крок до нового покоління “цифрового університету”. Значно

оптимізовуючи адміністративні витрати та мінімізувати людські помилки, а також внутрішні процеси управління повністю відповідатимуть європейським стандартам якості освіти. Тим самим прозорість, швидкість та велику точність у роботі закладу з великими масивами академічних даних дає перевагу закладу через технологічну досконалість інструментарію форматування документів.[4]

На основі спостереження за виробничими процесами можна виділити ряд проблем з автоматизацією звітності:

1. В Україні діють національні стандарти, які детально описують про розміщення реквізитів, які розміри полів треба задавати, який шрифт треба використовувати і т.п. Більшість зарубіжних платформ такі як Moodle або Google Classroom можуть генерувати розклади, але вони не відповідають вимогам оформлення документів, змушуючи персонал доопрацьовувати документи власноруч.[5,7]

2. Складність табличних структур. Розклади та навчальні плани мають складну структуру: об'єднані комірки, багаторівневі заголовки, змінна висота рядків. І алгоритмічним завданням становится автоматичне форматування елементів щоб все зберігалось на одній сторінці.

3. Проблеми локалізації. Коректне відображення Української мови, створює проблему для інструментів, розроблених без урахування підтримки Unicode або специфічних кодувань.

4. І для вирішення таких проблем треба переходити від використання готових шаблонів до генерації документів “на льоту” спеціальним програмним забезпеченням який здатний перетворювати дані та застосовувати до них правила верстки.

1.2 Аналіз нормативно-правової бази та державних стандартів оформлення звітності.

Університети, коледжі, школи, розробляють багато документів з розкладом занять, кількість документів множится на курси, спеціальності, класи, і для кожного докумета є стандарти які треба притримуватись. Без цих правил все це перетворюється на хаотичну макулатуру, де не буде зрозуміло

куди ідти і що викладати та для якого навчального закладу. То ж процеси пов’язані з стандартизацією це потрібні дії для стабільної праці.[2]

Згідно з українським законодавством, документ вважається легітимним тільки якщо в наявності є встановлені набори реквізитів, а саме: назва установи, підпис, дата, гриф затвердження, назва організації, назва документа тощо, вони повинні бути розташованими у чітко визначених місцях, як приклад - підписи та місця, для них повинні бути внизу документа, детальний розгляд розгляд вимог буде надалі розглядатись в цьому розділі. Стандартизація ДСТУ 4163:2020 дає гарантію що автоматично згенерований системою PDF-звіт, буде прийнятий як офіційний.[7]

Для освітнього закладу, стандартизація від якої іде відбір раціональних форм докуметнів означає, що розклад занять повинен мати однакову структуру для спрощення сприйняття інформації як студентам так і викладачам, що зменьшує кількість помилок при читанні. [8]

Згідно з національним стандартом, реквізит документа - це сукупність обов'язкових даних яка зафіксована в документі для ідентиффіцакції, без яких він не може бути підставою для обліку й не має юридичної сили. Стандарт визначає 32 наіменування як максимальний склад реквізитів, але для освітнього характеру, для розкладів всі не потрібні, а тільки головне. Цей огляд також потрібен для розробки модуля генерації, щоб враховувати розміщення потрібних елементів.[7]

1. Найменування установи (реквізит 07). Важливий елемент, він розміщується у верхній частині сторінки, та повинен мати повну, офіційну, зареєстровану в установчих документах назву закладу освіти, згідно зі стандартом. [7]

2. Назва документа(реквізит 09). Повинна бути написана верхнім регистром. Для навчальних закладів це зазвичай просто «РОЗКЛАД ЗАНЯТЬ» .[7]

3. Дата документа (реквізит 10). Є два способи написання дати затвердженим стандартом ДСТУ: цифровий (наприклад, 22.06.2005) та

словесно-цифровий (22 червня 2005 року). Але пріоритет для офіційних документів є саме другий варіант.[7]

4. Гриф затвердження документа (реквізит 16). Складається зі слова «ЗАТВЕРДЖУЮ», назва посади, підпису, власного ім’я та прізвища. І повинно розміщуватись у верхньому правому куту сторінки.[7]

5. Текст документа (реквізит 21). Для університетського розкладу текстом документа виступає сама таблиця. Кожна графа таблиці повинна мати чіткий заголовок, а кожна комірка мусить містити стандартний набір даних таких як: назва навчальної дисципліни, вид заняття, дані викладача та час. таблиця має бути цілісною. [7]

6. Підпис (реквізит 23). Його розміщення повинно бути нижче таблиці та складається з назви посади, сам підпис та місце для нього, власного імені і прізвища.[7]

Також окремо до оформлення документів є вимоги параметрів згідно ДСТУ 4163:2020, визначаються наступні фізичні параметри, а саме: розмір полів, шрифт та міжрядкові інтервали. Також обов’язкова регламентація розмірів берегів сторінки формату А4 (210×297 мм). Це необхідно для читабельності та можливості архівного підшивання документів. Вимоги є наступними:[7]

- Ліве поле - не менше 30 мм.
- Праве поле - не менше 10 мм.
- Верхнє та нижнє поля - не менше 20 мм.

Пункт 7.2 стандарту ДСТУ 4163:2020 регламентує вимоги до гарнітур, кеглів, кольору шрифтів, відступами та міжрядковим інтервалом.[7]

Для шрифтів рекомендовано використовувати класичні шрифти, це Times New Roman або Aria та використовувати чорний колір. Але перший варіант найчастіше використовують при формуванні документа. Розміри основного тексту зазвичай використовують кегль на 12-14 пунктів, заголовки та назви реквізитів можна виділити жирним або напів жирним накресленням, а у таблицях дозволяється використовувати менший розмір. Стандартом для

читабельності для міжрядкового інтервалу є 1,0-1,5 та абзацним відступом у 12,5 мм.[7]

1.3 Обґрунтування використання веб-застосунку замість десктоп.

Програма, функціональні можливості якої, реалізуються за допомогою серверу, та надаються користувачу через глобальну мережу Інтернет, та не встановлюється на сам комп'ютер, називається веб-застосунком.

Історично склалося що автоматизація створення документів розпочиналися з десктопних застосунків, як наприклад програми FET, Timetabling Turbo або Timetabling. Але з розвиттям у сфері веб-технологій поступово суспільство переходить з десктоп програм до веб-орієнтованих та гібридних рішень, вони забезпечують зручність через доступність та інтеграцію з іншими інформаційними системами.[9,10]

Кросплатформеність та відсутність вимог до інсталяції - головні переваги веб-застосунка проти десктоп. Очевидно що в одному університеті можуть працювати різні пристрої з різними операційними системами, Windows, Linux або навіть macOS, а коли іде мова про розклад то студенти частіш за все будуть його переглядати через мобільний пристрій, і десктоп в цьому плані буде програвати, оскільки він вимагав би окремої збірки, тестування, та налаштування середовища для кожного, і це значне навантаження на ІТ-відділ. Натомість краще використовувати браузер як універсальне середовище виконання. Від користувача не буде вимагатись зайвих труднощів як наприклад завантажування інсталяційних пакетів, налаштовування драйверів та інше. Для цього достатньо URL-адреси для початку роботи.[9,10]

Якщо порівнювати розгортання та обслуговування між веб-архітектурою та локальним ПЗ, то перший на порядок ефективніше ніж другий. Будь-яке оновлення для десктопних рішень, як наприклад - зміна шаблону розкладу коли вийшли нові стандарти ДСТУ, це вимагає повтороного розповсюдження для всіх користувачів додатка. Із-за цього виникає проблема коли частина працівників використовують стару версію додатка з застарілими алгоритмами, що надалі може бути причиною помилок. В той час для веб-застосунка якщо робиться

оновлення то воно робиться на сервері, і всі споживачі без необхідності встановлювати щось додаткове мають оновлену версію робочого інструмента. Для програмної генерації розкладу це означає що кожна робота буде відбуватись за єдиним затвердженим алгоритмом, тим самим мінімізуючи людський фактор.[9,10]

В десктопних варіантах програм, файли часто зберігаються на фізичному носії, чим є ризик втрати даних при поломці жорсткого диска або будь якої іншої причини. У веб застосунках така проблема вирішується набагато надійніше, вони працюють з єдиною централізованою базою даних саме на сервері.[9,10]

На сьогодні існує три основні підходи до розробки веб-додатків:

1) Односторінковий(SPA) (рис 1.1)
2) Багатосторінкові(MPA) (рис 1.2)
3) Прогресивні (PWA) (рис 1.3)

Перший варіант, SPA (Single Page Application) - веб-застосунок, він завантажує одну HTML-сторінку динамічно оновлюючи лише необхідний зміст без перезавантаження браузера.[11,12]

Головною ознакою є те, що все необхідне знаходиться на одній сторінці, а коли виконується перехід від однієї вкладки до іншої - адреса не змінюється, таку поведінку можна наблюдати у Gmail або Telegram.web. Але для великих проектів є неоптимальним. [11,12]

Рисунок 1.1 - Принцип роботи SPA

Другий варіант, (MPA Multi-Page Application) - багатосторінковий сайт, принцип його роботи заключається в тому, що коли користувач робить будь яку дію це супроводжується оновленням сайту. Це зручний спосіб для роботи з великими проектами, але недоліком виступає швидкість, яка повільніша за SPA. Прикладами такої логіки можна наблюдати у інтернет магазинах, як Rozetka або Amazon. [11,12]

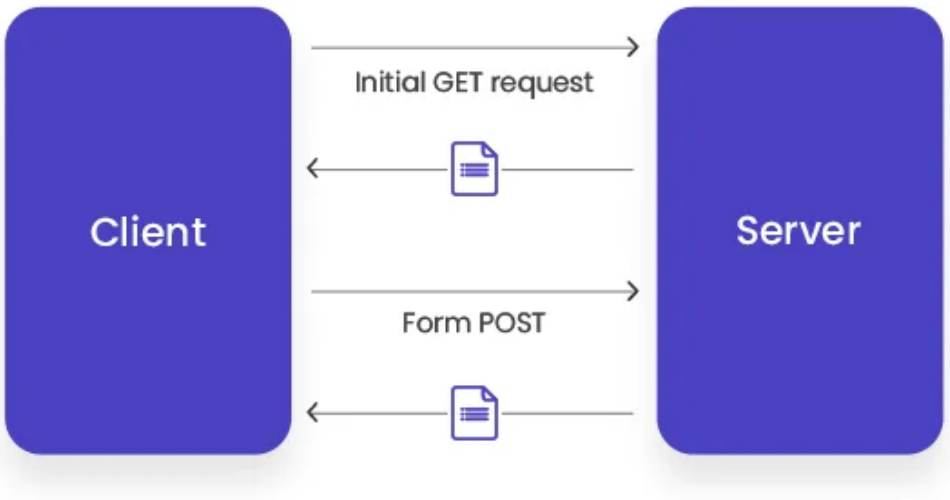

Рисунок 1.2 - Принцип роботи MPA

Третій варіант, PWA(Progressive Web Application) - тип сайтів які працюють у браузері але мають функціонал як у додатка, можуть працювати в офлайн режимі, надсилають push повідомлення, його можна встановити на пристрій та завдяки кешуванню працює швидко.[11,12]

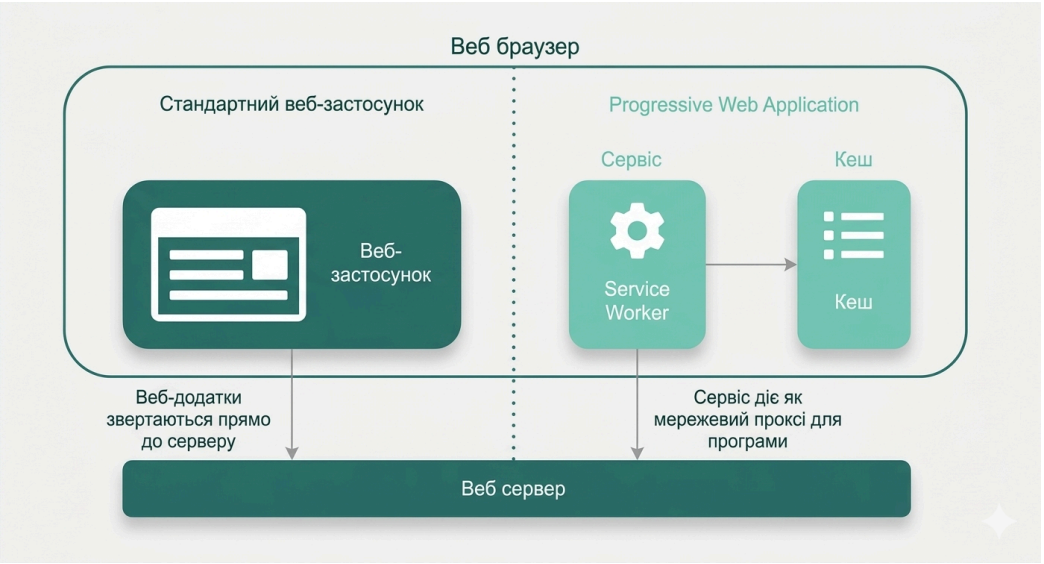

Рисунок 1.3 - Принцип роботи PWA

1.4 Огляд сучасних підходів до побудови архітектури «клієнт-сервер».

Архітектура «клієнт-сервер» - це спосіб взаємодії між споживачем, а саме клієнтом та постачальником ресурсів, тобто сервером. У веб-розробці на сьогодні виступає стандартом побудови розподілених обчислювальних систем та підходом до організації програмного забезпечення (рис1.4). [13]

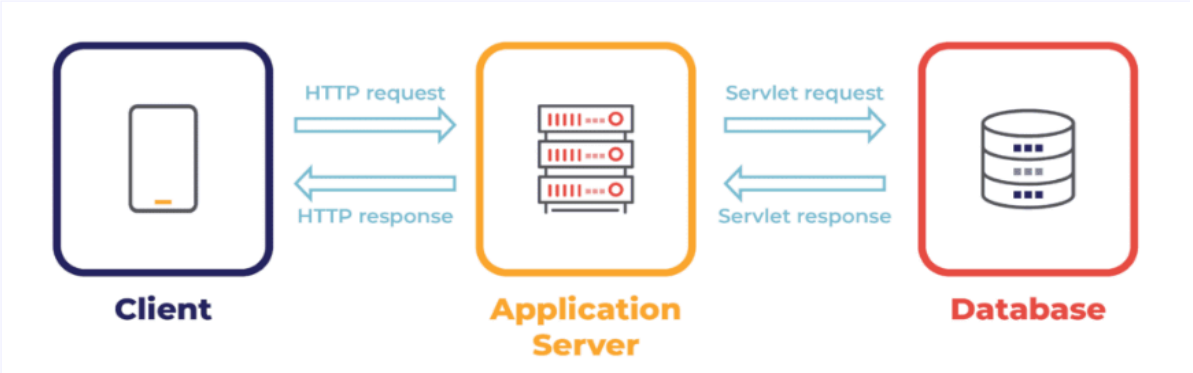

Рисунок 1.4 - Принцип роботи веб-застосунків

Теорія даної моделі базується на принципі ініціації запиту. Клієнт - формує запит на отримання даних, він діє як активна сторона, тоді як сервер - пасивна сторона, він чекає команди, і виконує обробку після надходження вхідних даних. Такий розподіл дозволяє реалізувати принцип розробки систем - Separation of Concerns (розподіл відповідальності). і Сервер не «знає» про деталі відображення інтерфейсу в браузері користувача, а клієнт не залучений до процесів звернення до бази даних.[13]

Поняття “рівня” або “ланки” позначає логічний чи фізичний розподіл функціональних компонентів системи. Розробники можуть ізолювати різні аспекти застосунку як бізнес-логіку, представлення даних т.і. використовуючи багаторівневі архітектури. Тим самим надаючи гнучкість системі, а тестування стає легше, та можна маштабувати окремі частини системи, незалежно одна від одної.[13]

Еволюційний розвиток архітектури пройшов через кілька етапів, кожен з яких наближав технології до сучасного стану веб-середовища:

Ера мейнфреймів (1960-ті - 1970-ті роки): В цей період використовувалась одноланкова архітектура це найпростіша форма, такі системи не використовують мережеву взаємодію для роботи, та на одному локальному комп’ютері в межах одного програмного пакету знаходяться дані, логіка обробки та інтерфейс. Відрізняються високою швидкістю доступу до даних через відсутність мережевих затримок, але унеможливлює колективну роботу, тобто дані залишаються ізольованими на одному пристрої, будь яка зміна вимагає оновлення ПЗ на кожному пристрої. То ж в цьому періоді, до великого та потужного комп’ютера підключалися термінали які не мали своїх обчислювальних потужностей та не було росподіленої обробки даних, вони могли тільки відобразити текстову інформацію. Модель була жорсткою, без графічних інтерфейсів.(рис.1.5) [13]

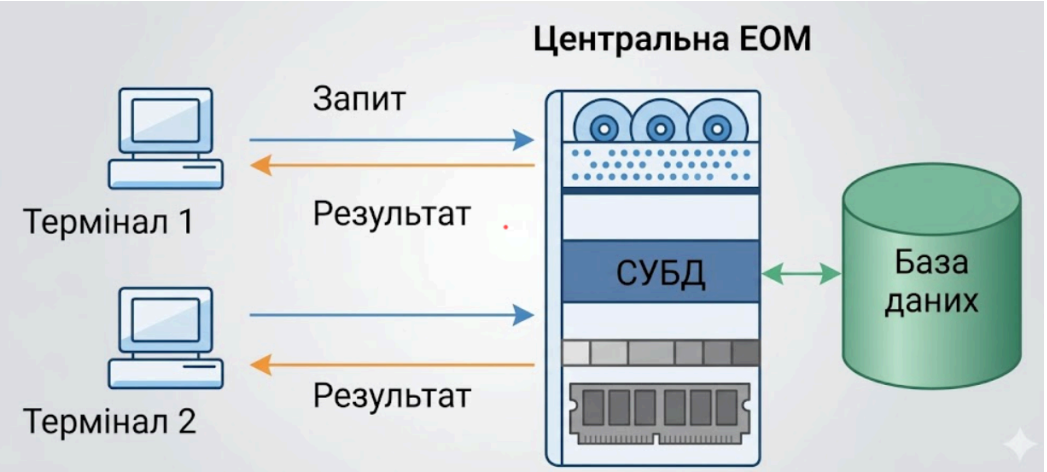

Рисунок 1.5 - Принцип роботи мейнфреймів

Модель файлового сервера (1980-ті роки): Коли почали з’являтись персональні комп’ютери, то виникла потреба у спільному доступі до файлів. Тут вже використовувалась архітектура Дволанкової архітектури. В цій схемі інтерфейс та бізнес логіка зосереджені на стороні кієнта, а на сервер лежить відповідальність за сховище бази даних. Але недоліком є високе навантаження на мережу та вразлива безпека через те що застосунок має прямий доступ до структури таблиць даних. А щоб зробити будь яку операцію треба було

завантажувати цілі файли на свій диск, а якщо двоє почнуть редагувати один і той самий файл одночасно один може затерти зміни іншого.(рис.1.6)[13]

Архітектура «товстого клієнта» (Fat Client, 1990-ті роки) Теж використовує Дволанкову архітектуру бізнес-логіка та інтерфейс користувача знаходилися на локальному ПК, але тепер сервер використовується як база даних. Проблемою подібного метода тепер виступає складність підтримки, це означає що будь-яка зміна в програмі потребувала оновлення програмного забезпечення на кожному комп'ютері[13]

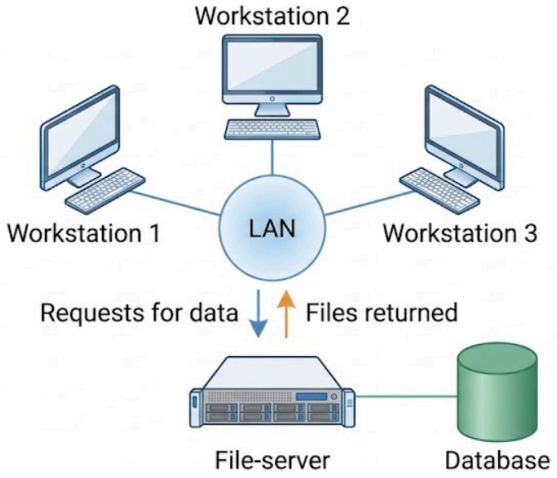

Рисунок 1.6 - Принцип роботи файлового сервера.

Архітектура «тонкого клієнта»(2000-ні - дотепер): Останній на даний період часу спосіб. Використовує Триланкову архітектуру який і є стандартом для сучасних веб застосунків. [13]

Перший рівень - представлення: верхній рівень, представлений веб-браузером користувача. Відповідає за візуал інтерфейса, прийом даних та відображення результату.[13]

Другий рівень - логіка застосунку, виступає посередником між клієнтом та базою даних, захищаючи від прямого впливу тут зосереджена вся частина системи як валідація даних, маршрутизація запитів т.і. (рис.1.7) [13]

Третій рівень - дані, нижній рівень де зберігаються вся інформація Це може бути реляційна база або структуровані файли. [13]

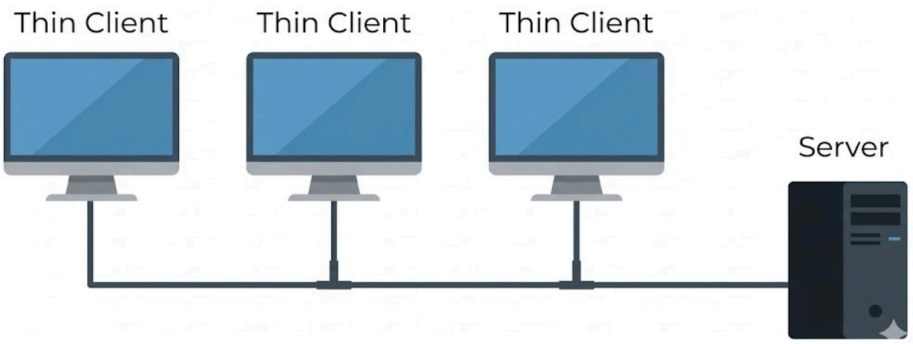

Рисунок 1.7 - Архітектура «тонкого клієнта»

Також сьогодні ще використовують багаторівневу або N-ланкову архітектуру (N-Tier) - подальше розвинення триланкової моделі, рівень логіки може бути розбитий на купу мікросервісів. Один може відповідати тільки за аутентифікацію, інший за збір даних. Забезпечує високий рівень відмовостійкості та маштабованості. [13]

Будь який застосунок працює на механізмі, відомий як цикл “Запит-Відповідь”, який забезпечує трансформацію вхідних даних від користувача до серверу, який обробляє їх і повертає результат. І цей маршрут між ними у розроблювальній системі можна розділити на кілька ключових фаз.[14]

1. Ініціація запиту та передача даних.

Процес розпочинається на стороні клієнта. Коли користувач натискає кнопку на сайті, або переходить за посиланням, браузер формує HTTP-запит з методом у тілі. [14]

2. Передача.

Перед тим як передати запит, клієнт повинен дізнатись IP адресу сервера, використовуючи доменне ім’я. Робится запит до системи DNS.[14]

3. Маршрутизація.

Коли запит отримується, сервер звертається до своєї таблиці маршрутів, він аналізує адресу та за потреби звертається до бази даних, також виконуються певні дії, наприклад якщо людина хоче зареєструватись виконується спеціальна обробка, перевіряють права доступу, логують час надходження.[14]

4. Формування та відправка відповіді.

Після обробки сервер надсилає клієнту статус-код, залежно від ситуації, перелік основних кодів буде далі в цьому параграфі. Разом з цим відправляються заголовки, які пояснюють “клієнту” що саме вони повертають, а також сам зміст.[14]

5. Завершення циклу на стороні клієнта.

Браузер отримує потік даних, закриває HTTP-з'єднання та інтерпретує дані, після цього цикл вважається завершеним.[14]

Для ефективного функціонування клієнт-серверної архітектури для веб застосунків використовуються стандартизовані протоколи передачі даних. Основним “каналом” зв’язку є протокол HTTP (HyperText Transfer Protocol) і його захищена версія HTTPS а JSON (JavaScript Object Notation) є найбільш поширеним форматом представлення даних.[15,16]

HTTP - прикладний протокол передачі даних, обмін повідомленнями йде за звичайною схемою “запит-відповідь”. Браузер виступає ініціатором сесії відправляючі певні запити на сервер. Особливість такого методу є його відсутність збереження стану, тобто сервер не зберігає інформацію про попередні запити клієнта.[15,16]

Запити в веб програмах використовують для роботи frontend частини з сервером, в кожного є своє призначення, ось перелік основних використовуваних запитів:[15,16,17]

- GET: Запитує вміст вказаного ресурсу. Згідно зі стандартом HTTP, запити типу GET вважаються ідемпотентний - багаторазове повторення одного і того ж запиту GET повинне приводити до однакових результатів (за умови, що сам ресурс не змінився за час між запитами). Це дозволяє кешувати відповіді на запити GET.[17]
- POST: Використовується для передачі даних на сервер. При цьому передані дані включаються в тіло запиту, а не в URL-адресі. На відміну від GET, цей не вважається ідемпотентним, багаторазове повернення одних і тих же запитів може повертати різні результати. [17]

- PUT: Використовується для оновлення вже існуючого ресурса. Зазвичай замінює ресурс цілком, але якщо відправляти з неповними даними, то вони можуть затертись.[17]
- DELETE: Видаляє вказаний ресурс. [17]

Код стану - тризначні числа, які сервер надсилає у відповідь на запит клієнта (браузера), вказуючи на результат обробки: успіх, перенаправлення або помилку, тобто він інформує про результат операції. І розуміння подібних запитів важливе для вирішення проблем з веб-застосунком далі буде наведені.[18]

Коди які починаються з 1xx - вказує на відповідь, на практиці вони зустрічаються рідко, та складається лише з статусного рядка, він інформує про отриману відповідь і продовження роботи сервера, перелік основних запитів приведено у таблиці 1.1.[18]

Таблиця 1.1

Статус коди 1xx

Код	Статус	Опис
100	Continue	Сервер отримав запит, клієнт може продовжити роботу.
101	Switching Protocols	Сервер пропонує перейти на більш відповідний для зазначеного ресурсу протокол.
102	Processing	Сервер прийняв запит, але на обробку треба більше часу
103	Early Hints	Проміжний статус відповіді, що дозволяє серверу відправити браузеру заголовки з посиланнями на критичні ресурси (CSS, JS, шрифти), перш ніж буде сформовано основний HTML-документ.

Якщо коди починаються з 2xx - це добрий знак, сервер повідомляє про успіх, клієнтський запит був отриманий, прийнятий та оброблений, перелік основних запитів приведено у таблиці 1.2.[18]

Таблиця 1.2

Статус коди 2xx

Код	Статус	Опис
200	OK	Успішна оброблена операція, відрізняється “успіхом” в залежності від запиту
201	Created	Запит успішно виконаний і в результаті був створений новий ресурс.
202	Accepted	Запрос прийнято, але поки не виконано.
203	Non-Authoritative Information	Використовується для індикації модифікації метаданих проміжним сервером
204	No Content	Успішна обробка запиту, але були відправлені тільки заголовки, без змісту
205	Reset Content	Повідомляє користувача агента про необхідність скинути документ, що відправив цей запит.
206	Partial Content	Сервер вдало виконав частковий GET-запит, повернувши тільки частину контенту.

Якщо статус коди мають на початку 3xx - це перенаправлення, сервер вказує що клієнт має виконати певні дії для завершення виконання запиту, як наприклад, перейти на іншу адресу. Клієнтська програма не повинна перенаправляти запит більш ніж у п'ять разів, так як такі переадресації зазвичай призводять до нескінченного циклу. Основний перелік подібних статус кодів наведено у таблиці 1.3[18]

Таблиця 1.3

Статус коди 3xx

Код	Статус	Опис
300	Multiple Choice	Надсилається у випадку якщо запит має одну з можливих відповідей, або користувач, або агент, повинен обрати один з варіантів

301	Moved Permanently	URL адрес було змінено назавжди, нову адресу вказано в заголовку.
302	Found	Запитаний ресурс тимчасово доступний за іншим URI
303	See Other	Ця відповідь відправляється для того, щоб направити клієнта для отримання запитуваного ресурсу по іншому URI
304	Not Modified	Використовується для кешування, повідомляє що відповідь не була змінена.
307	Temporary Redirect	Вказує клієнту що вказаний ресурс треба отримати за іншим URI тим же методом який використовувався в попередньому запиті
308	Permanent Redirect	Ресурс знаходится постійно за іншим URI.

Статус код 306 (Switch Proxy) та 305(Use Proxy) не наведений в цьому переліку оскільки на сьогоднішній день їх вже не використовують, але вони зарезервовані.[18]

Код 306 повинен був використовуватись для динамічного налаштування проксі-серверів, але був виключений із сучасних стандартів через порушення принципу інкапсуляції мережевих налаштувань та відсутність чіткої специфікації часу життя (TTL) примусового проксування.[18]

Код 305 використовувався як повідомлення, що відповідь повинна бути доступною через проксі-сервер, це створювало вразливість до атак MitM, Зловмисник міг відправити цю відповідь та змусити браузер проганяти весь трафік через свій підконтрольний сервер, перехоплюючі логіни та паролі.[18]

Статус коди які починаються з 4xx, повідомляють про помилки на стороні клієнта, а також у випадках якщо клієнт робить невірні дії, повинні вказувати пояснення щодо помилкової ситуації. Їх багато, але у наведеному переліку буде вказані основні, та розділені на три категорії.[18]

Перший це перелік помилок ідентифікації доступа, наприклад якщо у людини немає доступу, або він заблокований перелік наведено у таблиці 1.4[18]

Таблиця 1.4

Статус коди ідентифікації доступа

Код	Статус	Опис
401	Unauthorized	Означає що клієнт не авторизований. Для отримання запитуваної відповіді потрібна аутентифікація.
403	Forbidden	Сервер зрозумів запит, але відмовляється виконувати через обмежені права користувача.
407	Proxy Authentication Required	Те ж саме що і 401, але для проксі-сервера.

Наступний перелік пов’язаний з помилками запиту та ресурсу, наприклад коли користувач заповнив не ті дані, або коли ресурс взагалі не знайдено, або сервер не зміг зрозуміти команди (див.табл.1.5)[18]

Таблиця 1.5

Статус коди помилок запиту та ресурсу

Код	Статус	Опис
400	Bad Request	Ця відповідь означає, що сервер не зміг зрозуміти клієнтський запит з причини невірного синтаксису.
404	Not Found	Сервер не може знайти запитуваний ресурс.
405	Method Not Allowed	Метод запиту відомий серверу, але був відключений і не може бути використаний.
406	Not Acceptable	Сервер не може віддати дані у форматі, який запросив клієнт.

Третій перелік, Помилки стану та обмежень - Це ситуації, коли запит клієнта є технічно правильним, але він не може бути виконаний через поточний стан даних на сервері або встановлені технічні ліміти. (див. табл. 1.6)[18]

Таблиця 1.6

Статус коди помилок стану та обмежень

Код	Статус	Опис
409	Conflict	Запит конфліктує із поточним станом сервера
410	Gone	Відправляється у випадку якщо запитаний ресурс було видалено.
413	Payload Too Large	Занадто велике тіло запросу. Сервер може закрити з’єднання або повернути поле заголовку “Retry-After” із зазначенням часу, після закінчення якого можна буде повторити аналогічний запит.
415	Unsupported Media Type	Формат запитуваних типів даних не підтримується сервером.
429	Too Many Requests	Використовується для захисту від спам атак, у випадках якщо від одного користувача було занадто багато запитів.

Якщо код починається з 5xx - це означає проблему на стороні сервера, тобто виникла помилка, або не в змозі виконати запит з тої чи іншої причини, а для розробника це сигнал про необхідність перевірки логів сервера та виправлення багів, основний перелік запитів наведено у таблиці 1.7.[18]

Таблиця 1.7

Статус коди 5xx

Код	Статус	Опис
500	Internal Server Error	Сервер не знає що робити, тобто виник непередбачуваний збій
501	Not Implemented	Метод запиту не підтримується сервером і не може бути опрацьований.

502	Bad Gateway	Виникає, коли один сервер працюючи як проксі або шлюз, отримує неприпустиму відповідь від іншого сервера до якого він звернувся за виконанням запиту.
503	Service Unavailable	З технічних причин сервер не має змогу обробляти запити
504	Gateway Timeout	Сервер-посередник не дочекався вчасної відповіді від основного сервера.
505	HTTP Version Not Supported	Версія протоколу HTTP, яка використовується в запиті, не підтримується сервером
508	Loop Detected	Сервер виявив нескінченний цикл під час обробки запиту.

1.5 Висновки до першого розділу.

На базі проведеного аналізу процесів автоматизації документообігу в вищих навчальних закладах було обґрунтовано важливість концепції “paperless” замість часозатратної ручної праці. Окремо виявлено проблеми існуючих підходів та пояснення потреби розробити спеціальні інструменти для автоматичної генерації документів. Було проведено дослідження нормативно-правової бази, де пояснювались конкретні вимоги до документа для набутя статусу офіційного документа. Окремо було обґрунтовано доцільність розробки саме веб застосунку замість використання десктоп, та проведений теоретичний аналіз підходів до розробки, та теорія архітектури “клієнт-сервер”

РОЗДІЛ 2.АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ АНАЛОГІВ ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ.

2.1.Аналіз аналогів.

Для сучасного стану автоматизації планування процесів в навчальних закладах є доволі велика кількість комерційних програмних продуктів. І не буде зайвим провести аналіз існуючих рішень, це дає краще розуміння на визначення ключових векторів розвитку технологій у цій сфері.

На ринку представлено низку програмних продуктів, які можна розділити на три категорії:

1. Глобальні ERP-системи: Такі інструменти загалом охоплюють усі головні аспекти університету. І не зважаючи на це їх складність доволі висока, окрім цього, вартість ліцензій та важкість у налаштуванні під специфічні потреби закладів роблять їх слабкими по ефективності для автоматизації окремих модулів, по типу візуальної подачі розкладів.[19,20]
2. Спеціалізовані ПЗ для складання розкладів: Такі програми мають при собі потужні математичні двигуни для генерації документів. Але вони часто є десктопними, і це створює незручності з питаннями кросплатформи та водночас ускладнює інтеграцію в сучасні веб середовища та автоматизацію процесу видачі результатів у веб браузер.Таким прикладом є застосунок ascTimetables. [21,22]
3. Хмарні сервіси та Open Source рішення: Подібні веб платформи відрізняються тим що вони доступніщі, але проблемаю таких інструментів завзичай характеризуються обмеженням функціоналу, придатні для офіційного документообігу зокрема, а також не всі платформи у своєму безкоштовному функціоналі забезпечують генерацію структурованих PDF-файлів з дотриманням стандартів оформлення.[23,24]

Після аналізу основних методів планування розкладу, можна зазначити основні проблеми існуючих рішень, надмірна складність та висока вартість

робить процес малоефективним для специфічних задач, десктопні застосунки обмежує кросплатформеність а доступні відкриті рішення можуть бути обмеженими і не закривати потреби при розробці офіційного документа.[25]

2.2 Обґрунтування вибору мови програмування Python.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Розроблена на початку 1990-х років Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє повторному використанню коду. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, аспектно-орієнтована та функціональна.[26,27]

Обираючи дану мову як основу реалізації системи, наведемо деякі аргументи, вибір зумовлений зрілістю екосистеми для роботи з автоматизацією та обробкою структурованих даних. У Python є широкий набір бібліотек для роботи з документами, як для низькорівневого керування версткою так і для інструментів компонування. [26,27]

Python, на відміну від інших мов програмування, як JavaScript, або PHP, має легший синтаксис, завдяки динамічній типізації та багатим вибором бібліотек цикл розробки в рази коротший. При реалізації алгоритмів генерації розкладу не треба буде витрачати багато ресурсів на низькорівневі деталі керування пам'яттю або суворої типізації. [26,27]

Через легший синтаксис, мова має свою перевагу з точки зору підтримки системи, а саме його читабельність, то ж після тривалої затримки при написанні, код залишається зрозумілим та при поверненні на роботу над ним, також зменшує витрати на супровід при передачі проекту іншим розробникам або залученням нових, через це, ймовірність зробити помилку при внесенні змін до логіки генерації документів зменшена. [26,27]

Окремо важливим аргументом в сторону Python це те, що є можливість реалізувати всі етапи до купи одного технічного стеку: отримання даних веб-запитом, їх подальша обробка з розрахунком координат елементів та фінальна генерація PDF документа, це спрощує архітектуру системи і усуває необхідність передачі даних між різними мовами або середовищами.[26,27]

В контексті сучасних підходів до розгортання - а саме контейнеризація засобами Docker, який відомий завдяки відсутності залежностей, головним фактором повинна бути кросплатформеність, і Python це надає, надаючи стабільну роботу модуля генерації документів в різних системах без зайвого налаштування середовищ. Також важлива підтримка роботи з різними шрифтами та кодуваннями по типу UTF-8, ця мова разом із бібліотекою ReportLab має перевірені інструменти для роботи з шрифтами з правильним відтворенням кирилічного тексту. А ітераційні можливості дозволяють без проміжних конвертерів поєднати модуль генерації та веб-частину застосунку. Це спрощує відлагодження та зменшує кількість відмови в системі, отримання даних через API та їх обробка з подальшою генерацією в PDF реалізуються в межах одного серверного процесу. [26,27]

2.3 Порівняльний аналіз бібліотек мови Python для генерації PDF.

Інструмент для генерації документів треба обрати такий, який має при собі баланс між швидкістю розробки, та точним візуалом. На “ринку” бібліотек можна обрати один із трьох основних підходів, залежно від вимог, кожен пропонує свою власну логіку побудови документа.[28]

Перший такий підхід це конвертація розмітки, передбачає створення HTML шаблону який бібліотека перетворює у PDF документ, плюсом є те що такий метод має низький поріг, бібліотеки як WeasyPrint підходять для проектів де пріоритет - швидкість ітерацій та використання навичок фронтенду, однак точність верстки залежить від особливостей конкретного рушія, тобто не завжди передбачувана. [28]

Перевагою такого методу є можливість використання сучасних засобів CSS такі як Flexbox, Grid, застосування складних шрифтів та стилізованих

таблиць. Однак така простота має свій підводний камінь, продуктивність обмежена, а конкретно на рендеринг важких HTML-документів витрачається значна кількість пам’яті. Хоча WeasyPrint підтримує стандарт CSS Paged Media, стабільність відображення елементів між версіями рушіїв не дає гарантії.[28]

Другий підхід - Програмне малювання, базується на принципі віртуального полотна, розробник має повний контроль над кожним пікселем документа, він в коді задає координати кожному елементу, тексту, ліній, зображень. Бібліотеки на кшталт ReportLab Canvas або FPDF/PyFPDF надають повний контроль над розміщенням. Даний метод доцільний у випадках нестандартних або складних структур документа, у якого немає “шаблонного” стиля верстки.[29]

Також така методологія має перевагу в точності відтворення, і буде мати однаковий вигляд на будь якому пристрої, а швидкість генерації досягається через відсутності проміжуточного етапу парсингу HTML. Але є неприємний суттєвий недолік: Бібліотека не виконує автоматичного керування потоком тексту, не знає де закінчується сторінка або як перенести довгий текст на другу сторінку, розробнику треба самостійно реалізовувати алгоритми пагінації та розрахунку переносів, і виходить що заради тотального контролю треба жертвувати обсягом коду, а також збільшується ймовірність помилок у складних макетах.[29]

Третій метод - Об'єктно-орієнтована верстка, замість прямого керування координатами, оперування абстрактними об’єктами такі як параграфи, таблиці, фрейми та шаблонами сторінок, використовуються високорівневі двигуни компонування. Представники цього класу є також ReportLab Platypus та бібліотека Borb. Підхід також непогано працює зі створенням багатосторінкових документів зі складною структурою документів.[30]

Бібліотеки цього класу використовують складні алгоритми для розміщення елементів на сторінці. Тобто якщо таблиця перевищує розмір аркуша система перенесе частину на наступну сторінку, зберігаючи заголовки. Такі інструменти підтримують генерацію структурованих PDF з металаними,

закладками та інтерактивними формами. Але основна складність виникає у вищому порозі входження через специфічну термінологію як наприклад розуміння як працюють Flowables. Водночас витрати на навчання забезпечує стабільність і масштабованість при генерації документів великого обсягу. [30,31]

Після порівняльного аналізу бібліотек та методів для генерації PDF документів в Python встановлено, що жоден з цих методів не є універсальним для складних систем. Метод конвертації HTML - зручний, але має обмежену придатність через велике споживання ресурсів та нестабільність роботи з великими обсягами даних . Програмне малювання хоч і забезпечує точність, однак із-за цього на реалізацію витрачається багато часу щоб отримати базову логіку пагінації та керування потоком. На основі цих даних для розробки системи обрано бібліотеку ReportLab, він поєднує контроль над версткою з високорівневими інструментами. І цей вибір обґрунтовується трьома ключовими характеристиками[28,29,30]:

Перша з них, бібліотека поєднує низькорівневий об’єкт Canvas з фреймворком Platypus, перший використовується для точного розміщення статичних елементів, логотипів і т.п. а друге для автоматизованої верстки динамічних таблиць.[28,29,30]

Друге - автоматизація, використання об’єктив Flowables забезпечує автоматичне керування потоком даних, пагінацією та переносами, це потрібно для коректної генерації документів з різним обсягом інформації.[28,29,30]

І третій - стабільність, оскільки в нього відсутість залежностей від зовнішніх системних файлів як у PDFKit, це дає легку контейнеризацію та гарантію на надійну роботу в хмарних середовищах. Роблячи висновки обрана бібліотека виступає повноцінним layout-двигуном - формує документи заданим стандартам та має мінімальні витрати по ресурсам.[28,29,30]

2.4 Обґрунтування вибору мікрофреймворку Flask.

Flask - мікрофреймворк для веб додатків, створений з використанням Python. Станом на лютий 2022 року стабільна версія Flask має номер 2.02.

Використовується для розробки таких проектів як Pinterest, LinkedIn.[32]

Він називається саме мікрофреймворком через те, що не вимагає спеціальних засобів чи бібліотек, в ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Але натомість він має підтримку розширень, а самі розширення оновлюються частіше ніж сам Flask. Тим самим хоч інші фреймворки такі як Jango і мають повний набір вбудованих компонентів, у першого є змога підключити тільки потрібне, та не використовувати зайві розширення які не будуть затронуті при написанні коду.[32]

Коли йде мова про вибір інструмента для бекенд-розробки, часто розглядається Django - високорівневий відкритий Python-фреймворк для розробки вебсистем. В нього вже є великий набір інструментів, постачається з готовою ORM-системою, вбудованою панеллю адміністратора, механізмами авторизації та багатьма іншими стандартними компонентами. Хоча на створення якогось великого багатокористувацького порталу або складної системи управління контентом такий підхід витрачає набагато меньше часу, проте для модуля генерації розкладів більшість вбудованих функцій залишатимуться невикористаними. Натомість, Flask дозволяє уникнути такої надмірності, архітектура програми буде залишатись легкою, та є повний контроль над компонентами і їх взаємодія друг з другом.[32]

Іншим популярним альтернативним вибором є FastAPI - веб-фреймворк для створення HTTP-сервісних API на Python 3.8+. Він використовує Pydantic та підказки типів для перевірки, серіалізації та десеріалізації даних. FastAPI також автоматично генерує документацію OpenAPI для API, побудованих з його допомогою. Також його сильна сторона полягає у підтримці асинхронного програмування. Він добре справляється з великою кількістю одночасних легких мережевих запитів. Однак генерація розкладів через асинхронну архітектуру фреймворку не забезпечує ті переваги. Створення розкладу - синхронна задача, навіть якщо обернути цей процес в асинхронний код, операція залишається

блокуючою. Через це переваги одночасно обробляти числені запити втрачаються. [32]

Розробка програмного модуля для автоматизації академічного розкладу вимагає налаштування взаємодії між клієнтською частиною, базою даних та безпосередньо генератором документів. Flask надає простий механізм створення кінцевих точок (ендпоінтів), які приймають запити від користувача та повертають готовий PDF-потік у пам'яті сервера через об'єкт BytesIO. Відсутність жорстко нав'язаної структури проекту дає змогу організувати логіку так, як це найбільш доцільно для вирішення задачі візуалізації. Інтеграція з базою даних MySQL реалізується через мінімалістичні модулі, зберігаючи прозорий доступ до виконання SQL-запитів без прив'язки до важких абстракцій. Баланс між мінімалізмом ядра та гнучкістю підключення розширень робить цей інструмент раціональним вибором для побудови вузькоспеціалізованих сервісів. [32]

2.5 Обгрунтування вибору MySQL.

Для того щоб модуль міг працювати над генерацією розкладу, треба мати інформацію викладачів, по яким предметам він працює, його робочий час та окремо, інформацію про групи студентів, і її треба зберігати в базу даних. Через те що академічні дані мають сувору структуру, використання нереляційних баз даних, таких як MongoDB є недоцільним, так як не мають жорстких схем та підтримки складних трансакційних запитів. Серед реляційних систем керування альтернативами виступають SQLite, PostgreSQL та MySQL.[33]

SQLite - компактна, вбудована реляційна система управління базами даних, яка не потребує сервера, це зручний інструмент для локального розроблення, проте зберігає всі дані у звичайному файлі, а також має обмеження при роботі декількох людей одночасно, для багатокористувацього середовища де один заповнює дані про викладачів а інший створює розклад, ця база даних не дієва. [33,34]

Наступна альтернатива - PostgreSQL, потужна об'єктно-реляційна система управління базами даних, має глибоку підтримку складних аналітичних

функцій. Однак для генерації розкладу де немає потреби у складній обробці геоданих чи специфічних типів масивів, її функціонал є надлишковим.[33,35]

MySQL - одна з найпопулярніших у світі відкритих систем управління реляційними базами даних, вона пропонує значно легше налаштування, меньше споживання ресурсів сервера, та високу продуктивність в читанні даних. Також за допомогою розширення Flask-MySQLdb реалізується природно. Завдяки чому дозволяється виконувати прозорі SQL-запити без необхідності використання важких абстракцій ORM-систем. У підсумку оптимальним рішенням буде використовувати цю базу даних. [33,36]

2.6 Висновок по другому розділу

У другому розділі було виконано аналіз існуючих програмних аналогів та огляд світових патентних рішень та визначено проблеми візуалізації даних, обґрунтовано вибір інструментів для розробки модуля генерації розкладу. Проведено порівняльний аналіз бібліотек для автоматичного створення документів у форматі PDF, за підсумками якого обрано ReportLab через наявність механізмів автоматичного керування потоком даних та пагінацією. Окремо пояснювався вибір мікрофреймворку Flask для серверної частини застосунку. На основі отриманих результатів стає можливим проектування архітектури та безпосередня розробка веб застосунку для автоматизованого формування звітних документів у подальшій частині роботи.

РОЗДІЛ 3.ПРОЕКТУВАННЯ ВЕБ ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОГО ФОРМУВАННЯ PDF ДОКУМЕНТІВ.

3.1 Постановка завдання на розробку модуля візуалізації розкладу.

Після проведеного аналізу теоретичної частини веб-застосунків, дослідження існуючих засобів генерації розкладів та вимог до документів, обгрунтовується необхідність у розробці програмного модуля автоматичного створення документів. Труднощами в розробці виступає невідповідність вихідних даних від алгоритму який формує JSON-об’єкт з неструктурованими даними, та фінальний вигляд документа з його вимогами стандартам оформлення, повинен мати читабельну табличну структуру та придатним до друку без редагування.

Вимогами до розроблювального модуля є обробка запитів моментально, повна підтримка кирилиці та дотримання державних стандартів оформлення документації, а результатом розробки буде інструмент, який буде автоматично форматувати документ у PDF, враховуючи дані розкладу, позбавившись від ручного оформлення.

Для розробки модуля генерації розкладу треба поставити завдання для досягнення мети.

Першочерговою ціллю буде розробити алгоритм форматування вхідних даних для подальшої обробки. Для цього треба зробити інтерфейс передачі даних, це означає проектування внутрішніх протоколів, між модулем обробки та модулем генерації документа. А далі забезпечити нормалізацію контенту, встановлення даних у відповідні комірки з подальшим форматуванням за потребою, та очищення зайвих не використаних рядків.

Наступним завданням виступає проектування логічної та візуальної структури документа, що означає створення макета, який має всю необхідну інформацію та який буде легко прочитати. Для цього треба задати архітектурну сітку розкладу, тобто проектування композиції сторінки, враховуючи ієрархію заголовків, часові інтервали, назви дисциплін, та дані про кабінет.

Далі треба забезпечити адаптивність макета, враховуючи кількість навчальних пар протягом дня та довжину тексту, система верстки буде автоматично підлаштовуватись під параметри, усуваючи візуальне накладання елементів та порушення цілісності документа. Разом з цим треба враховувати вимоги ДСТУ, шрифти, відступи, вирівнювання, та розміщення реквізитів, для придатності офіційного використання документа.

Наступним кроком буде технічна реалізація візуалізації, а саме безпосередня програмна побудова, для цього треба розробити метод, який буде враховувати обсяг контексту а далі автоматично розраховувати висоту рядків та ширину колонок, для компактності та акуратності таблиць. Далі інтеграція та конфігурація кириличних шрифтів за вимогами створення документа.

І фінальний етап - вбудовування модуля у веб середовище, створення спеціальних маршрутів для ініціалізації генерації документа “на льоту” у відповідь на запит. Реалізація перегляду розкладу безпосередньо в клієнтську частину, у браузер, використовуючи бінарні потоки тим самим реалізація миттєвого попереднього перегляду сформованого розкладу у вікні переглядача з опцією подальшого завантаження або відправки на друк.

3.2 Функціональні і нефункціональні вимоги до системи.

Функціональні вимоги повинні давати відповідь на питання “що саме повинна робити система?” і повинно описати її поведінку, функції, операцію та обробку даних, все щоб задовольнити потреби користувача. Одним з таких можливостей - динамічне формування таблиць, це означає що треба автоматично підлаштовувати структуру документа враховуючи отримані дані, оскільки кількість груп для одного розкладу не однакова через різні спеціальності, напрями, та різна кількість предметів, є ймовірність що в конкретну годину може не проводитись навчальні заняття, що потребує редагування зайвих клітинок, через це робити статичний шаблон не кращий варіант.

Для забезпечення читабельності документа треба розробити механізм адаптації шрифтів. В залежності від довжини комірки шрифт всередині буде

зменшувати кегль символів, це необхідно для того щоб уникнути ризик виходу тексту за рамки та накладання тексту, зберігаючи цілісність верстки.

Інтелектуальна обробка дат це окреме завдання, замість того щоб прописувати для кожного дня коли буде перший та останній день навчання, достатньо обрати лише дві дати, початок та кінець семестру, система самостійно зробить розрахунки спираючись на введені користувачем дані та окремо розподіляє дати від понеділка до п’ятниці.

Програма повинна автоматично виставляти статичні боки реквізитів такі як “ЗАТВЕРДЖУЮ”, ініціали директора та заступника начальника управління, назву інституту т.і. згідно з державними стандартами, щоб готовий файл був юридично оформленим. Також важлива робота з PDF потоком, завдяки об’єкту BytesIO це дозволяє створювати документ в оперативній пам’яті сервера, надаючи безпеку даних, та має гарантію того що документ точно зробиться та не загубиться.

Нефункціональні вимоги описують саме як повинна працювати программа, описується продуктивність, надійність, безпека, масштабованість зручність т.і. Насамперед це точність верстки та відповідність стандартам. Документ повинен використовувати формат сторінки A4 та гарнітуру Times New Roman, завдяки TTFont в ReportLab. Закриваючи питання про вимогу щодо локалізації. Завдяки примусовій реєстрації зовнішніх файлів шрифтів є повну підтримка кирилиці, уникаючи некоректне відображення літер та “квадратиків”.

Підсумовуючи функціональні і нефункціональні вимоги до системи автоматизованої генерації документів, вона має забезпечувати динамічне формування таблиць розкладу без жорстких шаблонів, адаптивний розмір тексту для збереження верстки та розумний розрахунок дат семестру, автоматично додаючи необхідні державні реквізити. Процес створення PDF повинен безпечно відбуватися в оперативній пам'яті сервера. Водночас система повинна працювати швидко та без затримок завдяки інструментам бібліотеки ReportLab, забезпечувати стовідсоткову підтримку кирилиці та гарантувати

сувору відповідність стандартам ДСТУ.

3.3 Опис сценаріїв взаємодії користувача з системою (Use Case)

Цей сценарій є описом того як користувач, зокрема диспетчерська або заступник декану, повинен користуватись модулем генерації розкладу для того щоб отримати кінцевий результат у вигляді готового документа.

Передбачається що користувач до початку роботи заповнив дані викладачів, з їх побажаннями працювати в конкретні години та які предмети викладати, окремо прописав інформацію про самі групи з спеціальним кодом та з інститутом. Якщо ці попередні дії виконані можна приступати до генерації розкладу.

Процес розпочинається з того моменту коли користувач взаємодіє з інтерактивною формою. Спочатку обираються групи які заздалегідь були створені, наступним кроком обираються викладачі які будуть викладати в цьому потоці, та виставляються мітки хто з ким буде працювати. Наступними діями є написання прізвища та ініціалів декану та заступника начальника управління і наостанок залишається обрати дати початку навчального семестра та кінцеву. У випадку якщо користувач не заповнив те чи інше, при спробі переглянути чи створити, система попередить про це, і дасть підказку над тим чи іншим блоком що треба додати або дописати (рис.3.1)

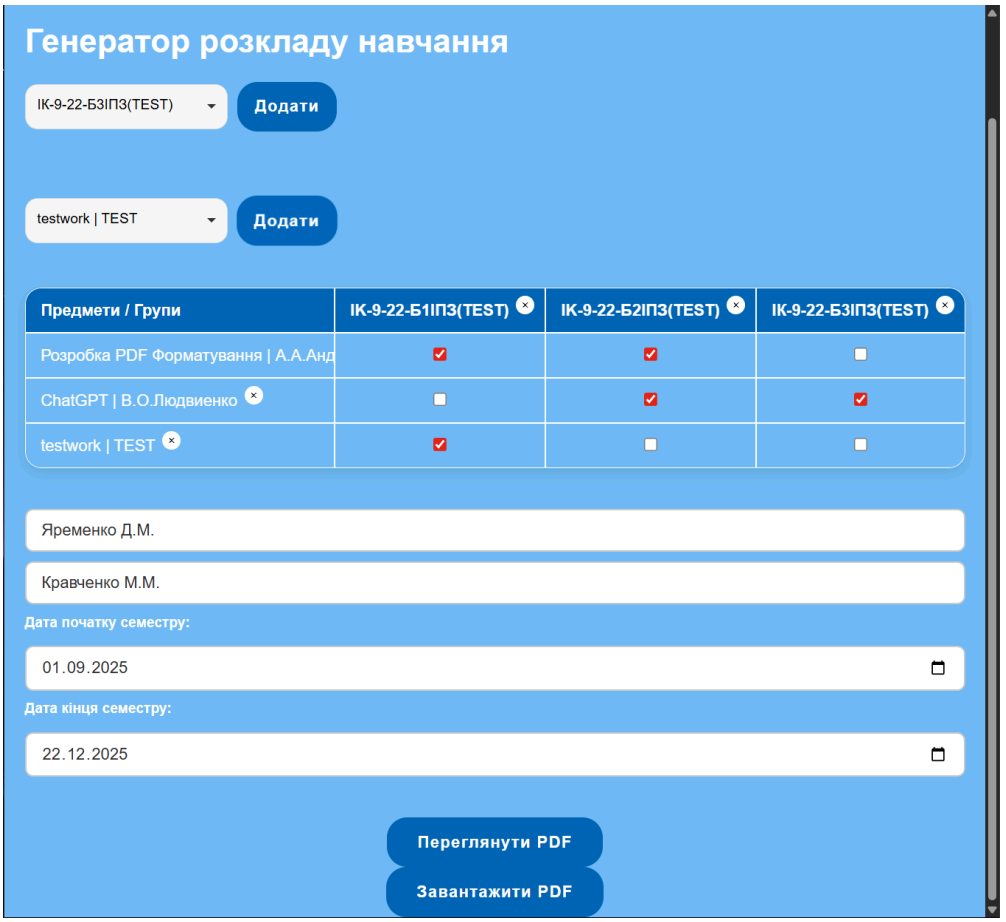

Рисунок 3.1 Інтерфейс генерації розкладу

Після заповнення всіх форм користувач може обрати одну з двох опцій на вибір, спочатку можна переглянути коректність роботи програми, а потім завантажити документ на сервер, якщо користувач впевнений у роботі він може одразу перейти до другого єтапу.

В тому чи іншому випадку, відправляється POST запит та передається зі змінною preview = True або False, який запускає ланцюжок операцій, а саме:

1) Серверна частина отримує дані з веб форми який надходить до коду у вигляді масиву. Система розбирає цей об’єкт витягуючи ідентифікатори груп та встановлює їх відповідність до обраних ідентифікаторів викладачів. Формується структура даних для подальшої обробки.

2) На основі отриманих ID система формує та виконує запити до бази даних. Відбувається вибірка інформації з таблиць студентів та викладачів.

3) Далі управління передається внутрішнім модулям логіки, які

відповідають за сортування викладачів, запобіганням часових конфліктів та отриманням даних з бази, після якого викликається головна функція, яка ініціалізує об’єкти розкладу та запускає алгоритм розподілу по аудиторіях і часових слотах.

4) Перед генерацією документа система готує метадані, де розраховуються точні дати дня тижня, номер поточного семестру та курс студента який конвертується у римські цифри.

5) Підготовлений масив передається до функції верстки документа, створюється байтовий буфер, підключає кириличні шрифти, будується багатовимірна таблиця де для кожної комірки розділяється вхідний рядок на назву дисципліни, прізвище та номер аудиторії. Далі динамічно вираховує ширину тексту і у разі перевищення лімітів комірки зменшує розмір шрифту. Наостанок створюються статичні об’єкти (назви інститутів, місця для підписів, тощо.) і компілюється.

6) Фінальний етап - маршрутизація та відповідь користувачу. В залежності від вибору користувача, виконується одна з двох дій.

- Якщо користувач обрав попередній перегляд, сервер додає заголовок “Content-Disposition: inline” і файл повертається у браузер і відкривається для перегляду
- Інакше при виборі завантаження система фізично зберігає згенерований документ у директорію на сервері, та перенаправляє користувача на головну сторінку.

3.4 Проектування архітектури та взаємодії між частинами.

Архітектурна побудова працює на принципі розподілу відповідальності для ізоляції логіки обробки даних від процесів візуального представлення зі збереженням. Центральним елементом рівня керування виступає контролер на базі фреймворку Flask, для розподілу завдань.

Головна функція render_data виконує роль координатора обробки запитів, перехоплює вхідні HTTP POST-запити, здійснює десеріалізацію параметрів та ініціює взаємодію з базою даних.

Цей етап важливий через об’єднання різноманітної інформації у єдиний словник pdf_context. Також виконуються розрахунки календарної сітки перетворюючи вибрану користувачем дату у масив робочих днів тижня.

Паралельно з цим, функціонує рівень генерації документів через ReportLab. В цомьу місці системи трансформуются дані у файл формату PDF. Завдяки системі компонування Platypus є можливість автоматизувати найбільш трудні процеси верстки. SimpleDocTemplate відповідає за дотримання полів.

Текстові дані обертаються у клас Paragraph, якщо текст виходить за рамки він самостійно перенесе текст на наступний рядок. А сама сітка формується через об’єкт Table, який приділяється увага стилізації через TableStyle, задаючи товщину сітки, колір, довжину т.і.

Нижній рівень архітектури займається система керування базами даних, вона відповідає за збереження нормалізованих даних про навчальний процес.

Процес створення документа починається з того моменту, коли користувач після заповнення форми підтверджує її відправку. В цей момент Flask активує процедуру збору даних, він звертається до бази для вилучення потрібних записів. Як тільки словник pdf_context наповнюється всією необхідною інформацією управління передається функції generate_pdf_file. Базове формування документа відбувається швидко в оперативній пам’яті, в залежності від режиму роботи яку вибрав користувач, попередній перегляд чи завантаження, система або віддає байтовий потік в HTTP-відповідь, або зберігає файл у систему сервера для подальшого використання.(рис 3.2.)

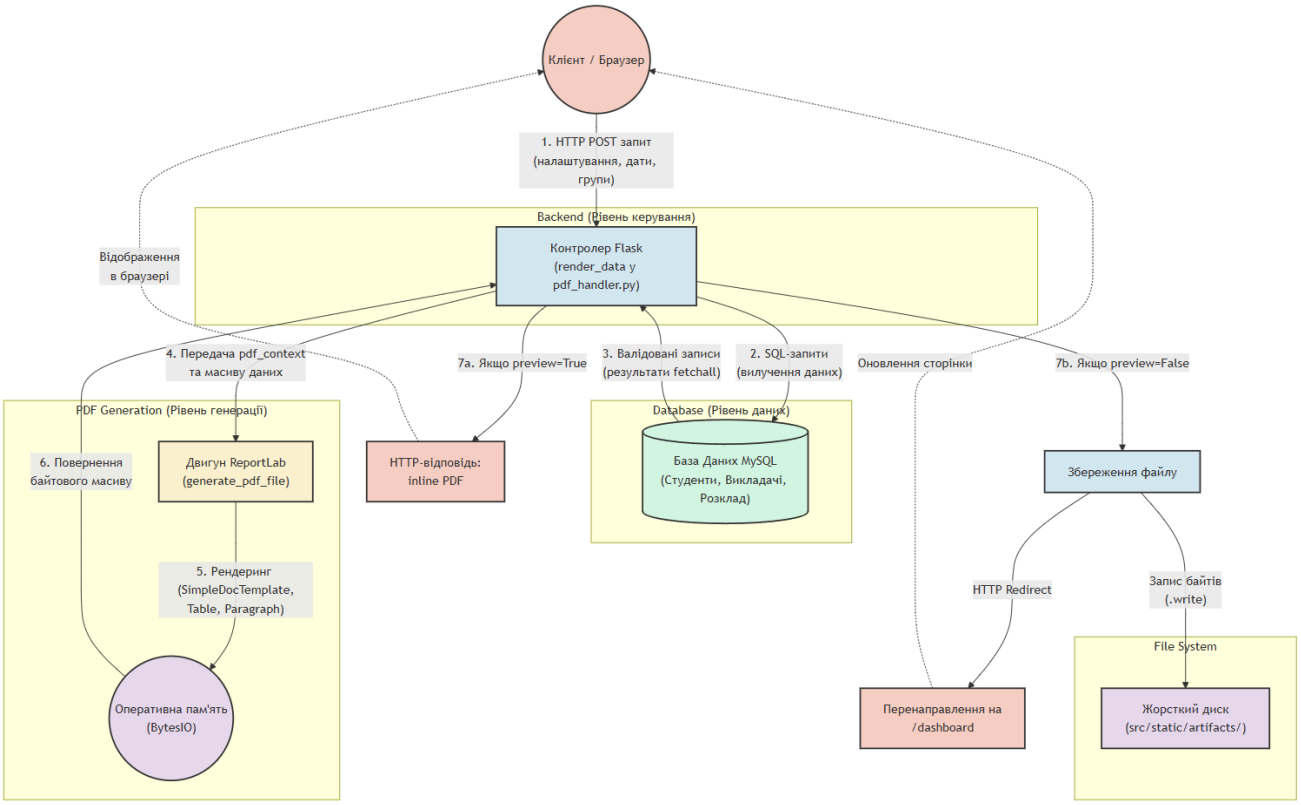

Рисунок 3.2. Діаграма архітектури модуля

3.5 Специфікація формату вхідних даних та розробка структури макета.

Модуль візуалізації отримує дані з контролера у вигляді об’єктів, словник діє як контейнер, який формується після обробки запиту користувача

Дані які передаються є:

1. Календарні: Алгоритм обчислює дати на основі початкового дня та кінцевого, і результат розрахунку зберігає для подальшої обробки. Разом з цим розраховуються навчальні роки.

2. Академічні: Інформація про навчальний процес, а саме номер курсу, окремо римськими цифрами та номер семестру який розраховується залежно від обраного курсу та дати початку.

3. Організаційні реквізити: Назви інституту, його повна версія з верхнім та нижнім регістром для документа, окремо скорочена назва для наіменування файлу, прізвище посадових осіб, а саме декан та заступник начальника управління.

4. Багатовимірний масив який згенерований алгоритмом розподілу і який містить розклад предметів.

Кожен запис дисципліни у масиві передається у вигляді єдиного текстового рядка певного формату, [Назва дисципліни] - [ПІБ викладача] - (Аудиторія:[Номер]). Для того щоб парсеру безпечніше розділяти інформацію.

Візуальна структура документа формується на двигуні макетування Platypus та враховує вимоги до оформлення документації. параметри виставляються фізично, формат аркуша А4, поля: ліве - 2см, праве - 1см, верхнє та нижнє - 2 см. Використовується шрифт Times New Roman. (рис.3.3)

Топологія сторінки розбита на три ключові блоки:

1. Шапка документа:

Цей блок реалізує обов'язкові державні реквізити:

1. Гриф затвердження (Реквізит 16)*:* Розташований у верхньому правому куті. Включає слово "ЗАТВЕРДЖУЮ", посаду ("Директор"), назву інституту, місце для підпису та дату затвердження.

2. Назва документа (Реквізит 09)*:* Центрована назва документа "Розклад занять" з додатковою інформацією про семестр, навчальний рік, форму навчання та курс.

3. Окремо у верхньому правому куту документа прописан курс.

2. Основна таблиця:

Таблиця формується динамічно і складається з трьох зон. В першій колонці розташована інформація про дні тижня та дати семестру. Друга колонка відповідає за часові проміжки пар і подальші колонки, кількість якої залежить від обраних користувачем груп, мають інформацію про самі завняття, хто викладає, який предмет та номер аудиторії.

3. Блок підписів:

Для реалізації реквізиту 26, а саме підписи деканату та заступника начальника управління було створені спеціальні поля, які розташовуються рівно під таблицею.Оскільки кількість груп та довжина назв дисциплін можуть кардинально відрізнятися, макет спроектовано за принципами адаптивності:

1. Загальна ширина однієї початкової колонки це 16 см, в залежності від кількості груп які обрав користувач, ширина стовпців рівномірно ділится на етапі генерації.

2. Розроблено функцію адаптації розміру шрифта, яка після отримання потрібного тексту, функція в циклі зменшує розмір шрифту поки текст не поміститься у відведену йому комірку. Для того щоб виключити накладання на межі таблиці у випадку довгих назв.

Приклад того як саме виглядає документ можна ознайомитись у додатку(Б)

3.6 Висновок до третього розділу

У третьому розділі було виконано системне проектування програмного модуля, визначені функціональні та нефункціональні вимоги до системи до побудови табличних структур та механізмів адаптивної верстки. Було описано детальні сценарії взаємодії користувача з інтерфейсом, від заповнення форм до отримання документа. Описана трирівнева архітектура взаємодії між Flask, базою даних MySQL та бібліотекою ReportLab. Окремо визначено специфікацію вхідних даних та структуру макета з урахуванням обов’язкових реквізитів згідно ДСТУ 4163:2020, завдяки чому сформовані технічні рішення для подальшої реалізації застосунку.

РОЗДІЛ 4.РЕАЛІЗАЦІЯ ВЕБ ЗАСТОСУНКУ.

4.1 Організація серверної частини та взаємодія з базою даних

Для початку програмної реалізації треба ініціалізувати об’єкт застосунка та налаштувати зв’язок з базою даних MySQL. У конфігурації прописуються стандартні параметри доступу, а саме адреса хоста, ім’я користувача, пароль до бази та сама назва.:

- MYSQL_HOST: В межах розробки встановлюється як “localhost”, для того щоб розробка велась локально, але якщо робота проводится у контейнеризованому середовищи Docker Compose натомість використовується “db”
- MYSQL_USER та MYSQL_PASSWORD: Облікові дані для отримання доступу до самої бази даних.
- MYSQL_DB: Назва бази даних, в якому зберігається інформація про студентів, викладачів та користувачів.

Для того щоб при першому запуску забезпечити цілісність структури даних, викликається функція initialize_database(app,mysql) - вона відповідає за створення потрібних таблиць та перевірку початкових налаштувань.

Щоб у Flask була можливість взаємодіяти з даними, використовується розширення “flask_mysqldb”, яка використовується для прямих SQL-запитів до бази без використання громіздких ORM-систем, використання даного методу зумовлено продуктивністю модуля. Основним механізмом отримання інформації це робота з курсорами “mysql.connection.cursor” завдяки якому здійснюється виконання команд та отримання результатів у вигляді кортежів даних.

Ключовою функцією є функція “getDataTeachers” - вона забезпечує вибірку інформації про викладачів за ідентифікаторами. Процес включає динамічне формування рядка для SQL-запиту, використовується для безпеки передаваємого списку ID будь-якої довжини, а також для захисту від ін’єкцій:

```
def getDataTeachers(selected_teachers_ids):
```

```
        cur = mysql.connection.cursor()
                    format_strings    =    ','.join(['%s']    *
len(selected_teachers_ids))
            cur.execute(f"SELECT  *  FROM  teachers  WHERE  ID  IN
({format_strings})", tuple(selected_teachers_ids))
        var = cur.fetchall()
        cur.close()
        return var
```

В ролі контролера та центральним компонентом управління виступає функція “render_data”. Головна функція якої це збір даних, для цього вона виконує наступні критичні завдання:

1. Прийом та десеріалізація даних: Після того як дана функція отримує HTTP POST запити від фронтенд-частини, вона вилучає дати початку та кінця семестру, разом з цим перелік груп та викладачів які обрав користувач та переданий у форматі JSON.

2. Обробка часових параметрів: Використовуючи бібліотеку datetime, система розраховує календарну сітку тижня. Окремо визначаються дати для кожного дня (від понеділка до неділі), враховуючи зміщення відносно вказаних дат.

3. Формування контексту для генерації: Після обробки та вибірки даних про студентів та викладачів система створює словник “pdf_context” яка зберігає в собі всю потрібну інформацію для роботи візуалізації, а саме номери курсів, семестрів, назви інститутів, ініціали посадових осіб та розраховані дати.

4. Маршрутизація результату: В залежності від того, що обрав користувач, попередній перегляд або завантаження, змінюється прапорець preview, відштовхуючись від нього, контролер поверне або сформований бінарний потік безпосередньо у браузер для попереднього перегляду PDF файла, або збереже файл на сервері, та перенаправить користувача на панель керування.

```
response = make_response(pdf_content)
    response.headers['Content-Type'] = 'application/pdf'
```

```
        if preview:
            response.headers['Content-Disposition'] = f'inline;
filename*=UTF-8\'\'{render_name}'
            return response
        else:
            output_dir = os.path.join(os.getcwd(), "src",
"static", "artifacts")
            os.makedirs(output_dir, exist_ok=True)
            filepath = os.path.join(output_dir, safe_filename)
            with open(filepath, 'wb') as f:
                f.write(pdf_content)
            return redirect('/dashboard')
```

4.2 Алгоритми підготовки та форматування вхідних даних

Після успішної ініціалізації серверної частини, критично важливим етапом буде робота з обробкою сирих даних, саме через цей процес забезпечується трансформація вхідних параметрів веб форми та результатів SQL-запитів у структурований вигляд, який буде придатним для роботи двигуна верстки

Особливою функцією даного модуля є автоматичний розрахунок навчальних дат. Завдяки чому користувачу не потрібно зайвий раз заглядати в календар та прописувати кожен ден початкового тижня та кінцевого тижня, а достатньо ввести дві початкові точки, початок семестру та кінець семестру

Процес підготовки календарних даних у функції “render_data” реалізований за наступною логікою:

1. Парсинг вхідних рядків: Дати перетворюються з формату рядка у об'єкти datetime для подальшої роботи.

```
start_date = request.form.get("start_date")
start_day = datetime.strptime(start_date, "%Y-%m-%d")
end_date = request.form.get("end_date")
end_day = datetime.strptime(end_date, "%Y-%m-%d")
start_weekday = start_day.weekday()
end_weekday = end_day.weekday()
```

2. Розрахунок зміщення : Для кожного дня тижня обчислюється різниця у днях відносно start_weekday та end_weekday.

3. Генерація словників тижнів: створюються два об’єкти, а саме “week” яка є датою початку, та week_end - кінцевим, ключем є назва дня а значення- відформатований рядок дати.

```
    week = {}
    for i, day in enumerate(week_start_name):
     offset = i - start_weekday if i >= start_weekday else i - start_weekday + 7
week[day]=(start_day+ timedelta(days=offset)).strftime("%d.%m.%Y")
    week_end = {}
    for i, day in enumerate(week_end_name):
    offset = i - end_weekday if i <= end_weekday else i - end_weekday - 7
week_end[day]=(end_day+timedelta(days=offset)).strftime("%d.%m.%Y")
```

Наступним кроком виступає десеріалізація та структурування академічного вибору. Вхідні дані про обраних користувачем груп та викладачів, надходять на сервер у форматі JSON-рядка у selected_schedule. цей алгоритм виконує наступні дії:

- Розбір структури: Рядок проходить процес перетворення даних у об’єкт, після чого кожна пара "група:викладач" розділяється за допомогою методу split(":", 1).
- Групування у словник: Створюється об'єкт, де ключами виступають ідентифікатори студентських груп, а значеннями - списки обраних викладачів.
- Перевірка типів: Система контролює, щоб значення для кожної групи завжди були представлені у вигляді списку, навіть якщо обрано лише одного викладача.

```
    if request.method == 'POST':
            selected_students_ids = []
            selected_schedule = json.loads(request.form.get('selected_schedule'))
```

```
            arr = {}
            for pair in selected_schedule:
                key, value = pair.split(":", 1)
                key, value = key.strip(), value.strip()
                selected_students_ids.append(key)
                if key in arr:
                    if isinstance(arr[key], list):
                        arr[key].append(value)
                    else:
                        arr[key] = [arr[key], value]
                else:
                    arr[key] = [value]
            print(arr)
            cur = mysql.connection.cursor()
            if selected_students_ids:
                format_strings = ','.join(['%s'] *
len(selected_students_ids))
                cur.execute(f"SELECT * FROM students WHERE ID IN
({format_strings})", tuple(selected_students_ids))
                selected_students = cur.fetchall()
            else:
                selected_students = []
```

Останнім кроками для підготовки до відправки на генерацію розкладу є отримання та розрахунок похідних даних, ім’я та прізвища деканату та заступника начальника управління, назва інституту, її коротка та повна назва, коди груп, розрахунок семетру та отримання курсу з конвертацією у римське число.

```
dek = request.form.get('dekan')
boss = request.form.get('boss')
course = (json.loads(selected_students[0][2]))["course"]
curs = course
rim_course_help = rim_convert(int(course))
rim_curs = rim_course_help
```

```
group = (json.loads(selected_students[0][2]))["group"]
sem_count = course_convert(int(course))
sem_check_day = start_day.month
```

Отримавши імена, починається робота з числом курса, спочатку відправляється у функцію “rim_convert” де йде просте перетворення у римське число, далі відправляється у “course_convert”, там залежно від курсу іде розрахунок семестру для подальшої її обробки.

```
def rim_convert(num):
    rim_numerals = {1:'I',2:'II',3:'III',4:'IV',5:'V'}
    return rim_numerals.get(num, str(num))

def course_convert(num):
    course_numerals = {1:1,2:3,3:5,4:7,5:9}
    return course_numerals.get(num, num)
```

Якщо місяць меньше червня, система розуміє що триває другий семестр навчального року і коригує дані.

Наостанок, розбиття назви інституту на коротку назву, яка використовується для найменування документа, та повну, що відправляється у експлуатацію генератора документа.

```
    institute = selected_students[0][1]
    inst = institute
    if inst:
        params = inst.split('|')
        inst_short = params[0] if len(params) > 1 else None
        inst_long = params[1] if len(params) > 1 else None
```

Результатом роботи вище описаних алгоритмів є словник pdf_context, який містить понад 20 підготовлених параметрів готових для відправки у модуль генерації розкладу в “чистому” виді.

4.3 Програмна реалізація генератора PDF-документів.

Головним компонентом системи візуалізації є pdf_generator.py, саме він відповідає за трансформацію структурованих раніше даних у формат PDF. Цей

модуль є зв'язуючою ланкою усієї роботи програми та основою функцією веб-сайту.

Першочергово треба зайнятись текстом, оскільки стандартні шрифти бібліотеки ReportLab не містять необхідних гліфів для кирилиці, важливо було подбати про підтримку української мови. Для вирішення даного питання було зареєстровано механізм реєстрації Times New Roman, це позбавляє проблеми з “квадратами” та прибирає невідповідність ДСТУ.

Процес реєстрації включає завантаження чотирьох основних накреслень:

- TimesNewRoman: основний текст документа.
- TimesNewRomanBold: для заголовків та виділення номерів аудиторій.
- TimesNewRomanItalic: для ініціалів викладачів та посадових осіб.
- TimesNewRomanBoldItalic: для комбінованого виділення.

Приклад реєстрації першого шрифта:

```
font_path =
os.path.join(os.path.dirname(os.path.abspath(__file__)),
'..','..','static', 'fonts', 'times.ttf')
pdfmetrics.registerFont(TTFont('TimesNewRoman', font_path)
```

Після реєстрації шрифтів починається заповнення документа, але спочатку жорстко задаються поля згідно з національним стандартом, ліве - 3см, праве - 1см, верхнє та нижнє - по 2 см. Потім іде наповнення реквізитами такими як “ЗАТВЕРДЖУЮ”, поля для заповнення директором, початкова дата семестру, та назва інституту верхнім регистром. Потім інформація над розкладом це назва документа “розклад занять”, дані про семестр та рік навчання, та назва інституту маленьким текстом, для кожного елемента задається “Paragraph” для гнучкого контролю позиції.

Особливістю реалізації є дворівнева структура таблиць. Хоча візуально цього не видно, кожна комірка основного розкладу є вкладеною таблицею, завдяки чому стає можливим встановити потрібні дані у свої місця, верхній рядок відповідає за назву дисципліни, а нижній, ПІБ викладача з ліва, та номер

аудиторії з права. Приклад створення комірки:

Стилізація таблиць реалізується через об’єкт TableStyle який відповідає за стилізацію таблиці, через нього визначається товщина меж “GRID”, відступи в середині клітинок “LEFTPADDING”, та об’єднання комірок “SPAN”.

```
inner_table = Table([
    [subject_p, ""],
    [teacher_p, room_p]
], colWidths=['50%', '50%'])
inner_table.setStyle(TableStyle([
    ('SPAN', (0, 0), (1, 0)),
    ('VALIGN', (0, 0), (1, 0), 'MIDDLE'),
    ('VALIGN', (0, 1), (1, 1), 'BOTTOM'),
    ('LEFTPADDING', (0, 0), (-1, -1), 2),
    ('RIGHTPADDING', (0, 0), (-1, -1), 2),
    ('TOPPADDING', (0, 0), (-1, -1), 1),
    ('BOTTOMPADDING', (0, 0), (-1, -1), 1),
]))
return inner_table
```

Генератор не створює проміжних тимчасових файлів під час формування звіту, використовуючи об’єкт BytesIO - віртуальний буфер в оперативній пам’яті. Це потрібно для забезпечення високої продуктивності.

Алгоритм завершення генерації включає наступні кроки:

1. Компіляція об'єктів: Виклик “doc.build(elements)”, що запускає процес рендерингу всіх доданих параграфів та таблиць.

2. Скидання покажчика: Перехід до початку буфера через buffer.seek(0) для зчитування готового бінарного вмісту.

3. Формування HTTP-відповіді: Передача байтового масиву контролеру Flask з відповідними заголовками Content-Type: application/pdf.

```
doc.build(elements,onFirstPage=lambdacanvas_obj,doc:
add_static_text(canvas_obj, ctx))
buffer.seek(0)
response = make_response(buffer.read())
```

```
response.headers['Content-Type'] = 'application/pdf'
response.headers['Content-Disposition']='inline;
filename=FILE_NAME.pdf'
return buffer.getvalue()
```

Завдяки такому підходу система може генерувати розклад майже на льоту, забезпечуючи миттєвий виклик для користувача. Наприкінці роботи генератор повертає об’єкт “getvalue()” - бінарне представлення документа для відправки клієнту для перегляду або збереження.

4.4 Реалізація адаптивного макетування таблиць розкладу

Оскільки кількість обраних груп та викладачів в одному розкладі може бути різною, а також довжина імен та назв дисциплін також відрізняється, треба враховувати цілісність верстки, інакше використання жорстко заданих розмірів шрифтів, комірок та стовпців призвело би до проблем з накладанням тексту, вихід за рамки або за межі друкованого аркуша, це стає складним з технічної точки зору задачею, оскільки в самій бібліотеці немає автоматичного керування розмірами, і для вирішення цього питання було спроектовано систему адаптивного макетування.

Ширина робочої області сторінки є фіксованою, на таблицю з розкладом виділяється загалом сімнадцять сантиметрів, з яких два закріплено для роботи з колонкою днів тижня, і один - для колонки з часом. Це обмеження є необхідним для подальшої роботи з адаптацією сторінки.

Останні чотирнадцять сантиметрів розраховує ширину динамічно на основі кількості обраних користувачем груп. Для цього викликається наступна формула:

```
group_cell_width = (14 * cm) / len(groups)
```

Результат використовується як для адаптації розміру тексту, так і для розрахунку ширини стовпців для груп.

```
col_widths = [2*cm, 1*cm] +[group_cell_width]*len(groups)
```

Головною функцією для адаптації шрифтів для дисциплін є алгоритм автоматичного підбору, реалізований у функції get_adaptive_font_size. Задача

якої - вмістити великий текст у комірку заданої ширини без переносів слів на нові рядки.

Логіка роботи алгоритму є ітеративною:

1. Функція приймає рядок тексту, назву шрифту, розмір та максимально допустиму ширину в пунктах (max_width).
2. Використовуючи вбудований метод stringWidth з модуля reportlab.pdfbase.pdfmetrics, система обчислює ширину, яку займе текст при поточному розмірі шрифту.
3. Запускається цикл, який перевіряє умову. Якщо текст виходить за межі комірки, розмір шрифту зменшується на 0.5 пункту, і перевірка повторюється.
4. Мінімальний поріг встановлено на рівні 8 пунктів для збереження базової читабельності.

```
def get_adaptive_font_size(text, font_name, default_size,
max_width):
    if not text:
        return default_size
    size = default_size
    while size >= 8:
        text_width = stringWidth(text, font_name, size)
        if text_width <= max_width:
            break
        size -= 0.5
     return size
```

Цей метод працює разом з класом Paragraph, який автоматично зменшує міжрядковий інтервал пропорційно до розміру шрифта.

Для кожної комірки з предметами була реалізована ще одна вкладена таблиця, оскільки звичайна текстова клітинка в таблиці розкладу не може забезпечити складне форматування, як наприклад, вставити ім’я викладача у лівому нижньому кутку, та окремо у правому кутку номер аудиторії.

Процес побудови подібної комірки відбувається наступним чином:

1) Сирий рядок передається у функцію, де розбивається на три змінні, для викладачів, для номера аудиторії та для назви предмета

```
def parse_subject(text):
    parts = text.split(" - ")
        if len(parts) < 3:
        return "", "", ""
    teacher = parts[1].strip()
    subject_part = parts[2]
    room = ""
    subject = subject_part.strip()
    if "(Аудиторія:" in subject_part:
        subject, room = subject_part.split("(Аудиторія:")
        subject = subject.strip()
        room = room.replace(")", "ауд.").strip()
    return subject, teacher, room
```

2) Вираховується корисна ширина для тексту з урахуванням внутрішніх відступів таблиці

3) Викликається get_adaptive_font_size для кожного елемента окремо. А також для тексту використовується окремий шрифт, для дисципліни використовується звичайний шрифт, для викладача - курсив, для аудиторії - напівжирний.

4) Створюється внутрішня таблиця розмірністю 2x2. У першому рядку розміщується назва предмета з об'єднанням стовпців . У другому рядку комірки для викладача та аудиторії. Далі додаються правила для стилізації цієї таблиці.

```
def create_cell(paras_text, cell_width):
    subject, teacher, room = parse_subject(paras_text)
    subject_max_width = cell_width - 8
    half_width = (cell_width / 2) - 4
            subj_size = get_adaptive_font_size(subject,
'TimesNewRoman', 10, subject_max_width)
...
...
```

```
        subject_p = Paragraph(subject, ParagraphStyle(
            name='subject',
            fontName='TimesNewRoman',
            fontSize=subj_size,
            alignment=1,
            leading=subj_size + 1
        )) #такий самий код і для інших об’єктів
    inner_table = Table([
            [subject_p, ""],
            [teacher_p, room_p]
        ], colWidths=['50%', '50%'])
        inner_table.setStyle(TableStyle([
            ('SPAN', (0, 0), (1, 0)),
            ('VALIGN', (0, 0), (1, 0), 'MIDDLE'),
            ('VALIGN', (0, 1), (1, 1), 'BOTTOM'),
            ('LEFTPADDING', (0, 0), (-1, -1), 2),
            ('RIGHTPADDING', (0, 0), (-1, -1), 2),
            ('TOPPADDING', (0, 0), (-1, -1), 0),
            ('BOTTOMPADDING', (0, 0), (-1, -1), 0),
        ]))
```

Після того як всі окремі комірки згенеровані, вони збираються в єдину макроструктуру. Алгоритм перебирає масив, що містить дані для кожного дня тижня та часового слота.

Щоб візуально об'єднати комірки днів тижня, використовується прапорець day_displayed та словник day_start_idx. Система запам'ятовує індекс першого рядка для кожного дня. З цими умовами далі виконується цикл для наповнення таблиці елементами з годинами та групами.

```
        for day, subjects in sched:
            day_displayed = False
            for time, *paras in subjects:
             if any(paras[:len(groups)]):
                    time_style = Paragraph(time,
ParagraphStyle(name='time', fontName='TimesNewRoman', fontSize=7,
alignment=1, leading=6))
```

```
                    group_cells = [create_cell(paras[i],
group_cell_width) if i < len(paras) and paras[i] else "" for i in
range(len(groups))]
                    row = [day if not day_displayed else "",
time_style] + group_cells
                    data.append(row)
                    if not day_displayed:
                        day_start_idx[day] = len(data) - 1
                    day_displayed = True
```

Після повного наповнення масиву даних, запускається цикл постобробки стилів, який додає правило від стартового індексу дня до його кінця:

```
        for day, start_idx in day_start_idx.items():
            end_idx = len(data) - 1
            for row in range(start_idx + 1, len(data)):
                if data[row][0] != "":
                    end_idx = row - 1
                    break
            style.add('SPAN', (0, start_idx), (0, end_idx))
            style.add('VALIGN', (0, start_idx), (0, end_idx),
'TOP')
```

І на фінальний етап створення таблиці, застосовується стиль до всієї сітки, за вказаними правилами промальовує межі, робить вирівнювання, встановлює шрифти для інших об’єктів та фоновий білий колір. А потім відправляється у загальний список з усіма реквізитами які були раніше описані, та передаються для безпосередньої компіляції PDF-документа у байтовый буфер.

4.5 Висновок до четвертого розділу

В розділі було розроблена програмна реалізація серверної частини веб застосунку та модуля візуалізації розкладу. Було описано процес налаштування взаємодії Flask з базою даних MySQL. Детально розглянуто алгоритми підготовки вхідних даних, логіку автоматичного розрахунку календарної сітки та формування словника для передачі у генератор. Також оглянуто процес створення PDF-документа бібліотекою ReportLab в пам’яті сервера.

РОЗДІЛ 5. ТЕСТУВАННЯ МОДУЛЮ

5.1 Тестування валідації вхідних даних.

Для фінального етапу було проведено ряд тестів, один з них це перевірка підготовки даних, який є важливим, оскільки коректність сформованого масиву впливає на працездатність модуля генерації. В планах тестування було перевірено роботу контролера render_data та допоміжних функцій.

Було проведено тестування обробки рядка який надходить з фронтенду перевірялось на коректність метод json.loads та подальший поділ рядків, формують словник, де ключами виступають ідентифікатори груп, а значеннями - списки викладачів. Результат вважається успішним у випадку уникнення помилок при передачі порожнього списку, або масив з одним викладачем на групу.(рис 5.1)

4 курс

"ЗАТВЕРДЖУЮ"
Директор
ІНСТИТУТ
КОМП'ЮТЕРНО-ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

_____ _____

1 вересня 2025

Розклад занять
На 7 семестр 2025/2026 навчального року денної форми навчання IV курс
Інститут комп'ютерно-інформаційних технологій та дизайну

(назва інституту)

День тижня	Пара	ІК-9-22-БІІПЗ(TEST)
Понеділок 01.09.2025-29.12.2025	8:30-9:50	TEST *TEST* 1 ауд.

Декан _____ TEST

Заступник начальника управління _____ TEST

Рисунок 5.1. Тестування з одним викладачем

Окремо було проведено тестування розрахунку календарної сітки, перевірялась логіка перетворення введених користувачем дат для подальшого алгоритму розрахунку тижня, і як очікуваний результат - коректне наповнення словників з правильними датами, незалежно від того, з якого дня тижня починається семестр. (рис.5.2)

День тижня	Пара
Понеділок 01.09.2025-29.12.2025	8:30-9:50

Декан

Рисунок 5.2. Тестування календарної сітки

Було додаткове тестування функцій роботи з обраним курсом, а конкретно конвертації арабських цифр курсу у відповідні римські, а також правильність розрахунку поточного семестру, тобто якщо обирається 4 курс, це може бути вже сьомий або восьмий семестр, залежно від початкової дати. (рис 5.3)

1 вересня 2025

Розклад занять

На 7 семестр 2025/2026 навчального року денної форми навчання IV курс

Інститут комп'ютерно-інформаційних технологій та дизайну

(назва інституту)

Рисунок 5.3. Тестування обробки курсу.

5.2 Тестування генерації PDF-документа та адаптивної верстки

У цьому тестуванні перевірялась робота модуля генерації документа, який відповідає за рендеринг документа в оперативній пам'яті.

Було перевірено алгоритм адаптивного макетування, тестування проводилося шляхом передачі в функцію спеціально довгих текстів для назв дисциплін або прізвищ викладачів. Було підтверджено, що цикл коректно зменшує розмір шрифту з кроком 0.5 пункту, якщо розрахована ширина перевищує максимальну допустиму ширину.(Рис 5.4)

Пара	ІК-9-22-Б1ПЗ(TEST)
8:30-9:50	TEST *TEST* **1 ауд.**
10:00-11:20	TEST *TESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTE STTESTEST* **2 ауд.**
11:30-12:50	TESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTES T *TEST* **3 ауд.**

Рисунок 5.4. Тестування форматування тексту

Також було проведено тестування розрахунку ширини колонок залежно від кількості обраних груп. А саме формула яка повинна коректно розподілити вільний простір таблиці, та зберігаючи загальну ширину таблиці. (Рис.5.5)

(назва інституту)

День тижня	Пара	ІК-9-22-Б1ІПЗ(TEST)	ІК-9-22-Б2ІПЗ(TEST)	ІК-9-22-Б3ІПЗ(TEST)
Понеділок 01.09.2025-29.12.2025	8:30-9:50	TEST *TEST* **1 ауд.**	TEST *TEST* **1 ауд.**	TEST *TEST* **1 ауд.**

Рисунок 5.5. Тестування розподілу комірок

Окремо візуально було перевірено розташування статичних реквізитів такі як “Затверджую” та місця підписів, попереднє тестування довели що вони стоять у відповідних місцях за стандартом ДСТУ.

5.3 Тестування кінцевих сценаріїв взаємодії.

Останній етап тестування був зосереджений на інтеграційній перевірці повного циклу "Запит-Відповідь" у контролері Flask на основі прапорця preview.

У першому сценарію був проведений тест попереднього перегляду, імітувався POST-запит, коли клієнт хоче лише переглянути документ та переконатися що його задовольняє результат. Виконуючі ці дії файл не повинен завантажуватись на сервер або на диск, а також аналізувались заголовки відповіді, система повинна повертати статус-код 200, Content-Type зі значенням application/pdf та заголовок Content-Description зі значенням inline. Це тестування підтвердило, що браузер користувача коректно відкриє байтовий потік у вбудованому PDF-переглядачі. (Рис 5.6)

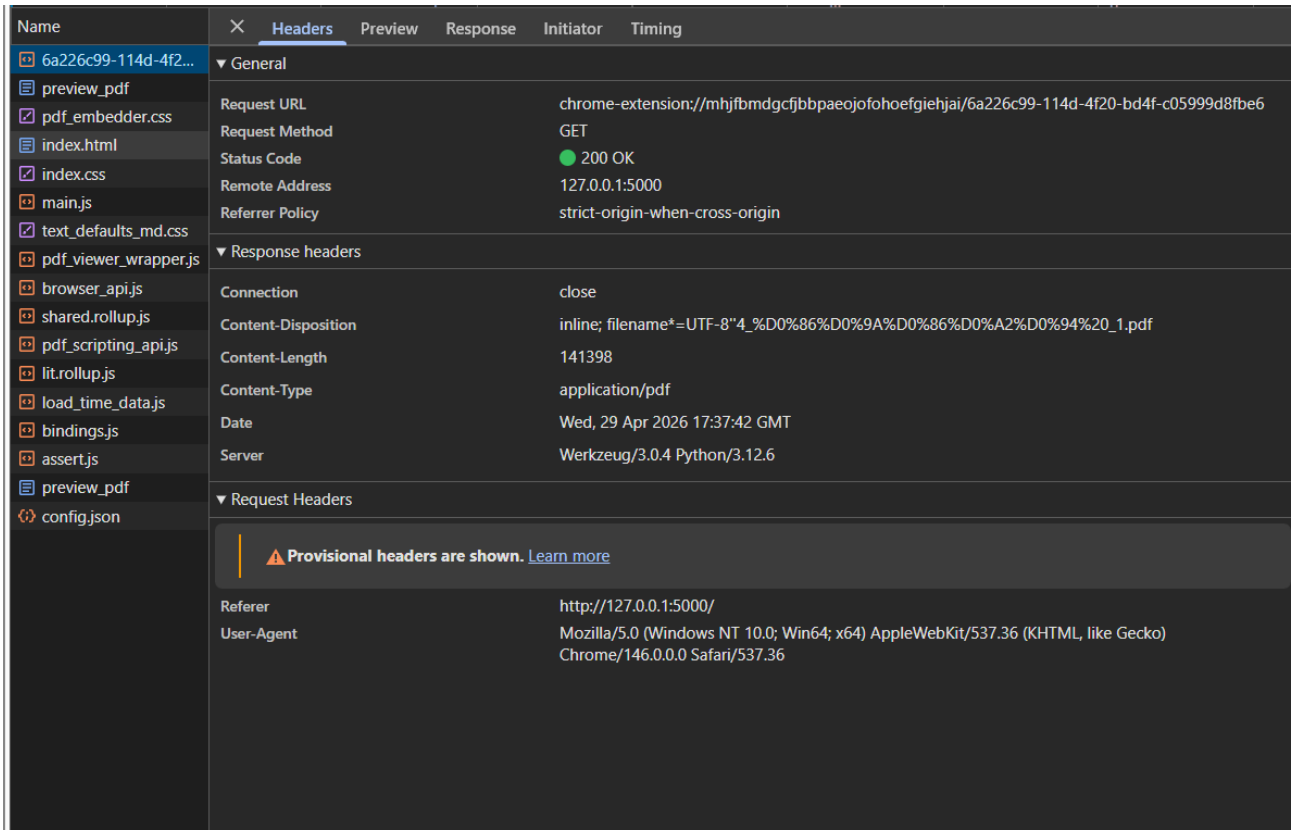

Рисунок 5.6. перевірка попереднього перегляду.

У другому сценарії вже перевірялось саме завантаження на сервер, коли користувач вже впевнений що його задовольняє результат, він натискає на “зберегти PDF”, імітувався POST-запит з вимогою зберегти розклад. Тестувалась саме логіка роботи з файловою системою, конкретно збереження документа у певну директорію, якщо цього маршруту на сервері не має, програма повинна створити його та зберегти файл у відповідному місці, окремо перевірялась назва згенерованого файлу на відповідність формату курс, коротка назва інституту,рік початку навчання(рис 5.7.), а також під час збереження перевірялась, що контролер повертає статус-код 302 та коректно перенаправляє користувача на головну сторінку.(Рис.5.8)

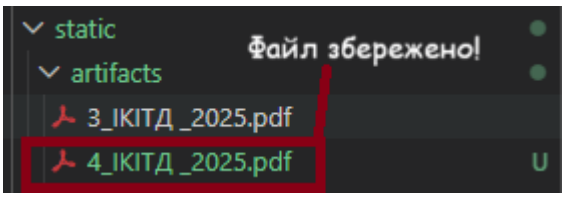

Рисунок 5.7. Перевірка збереженого файлу

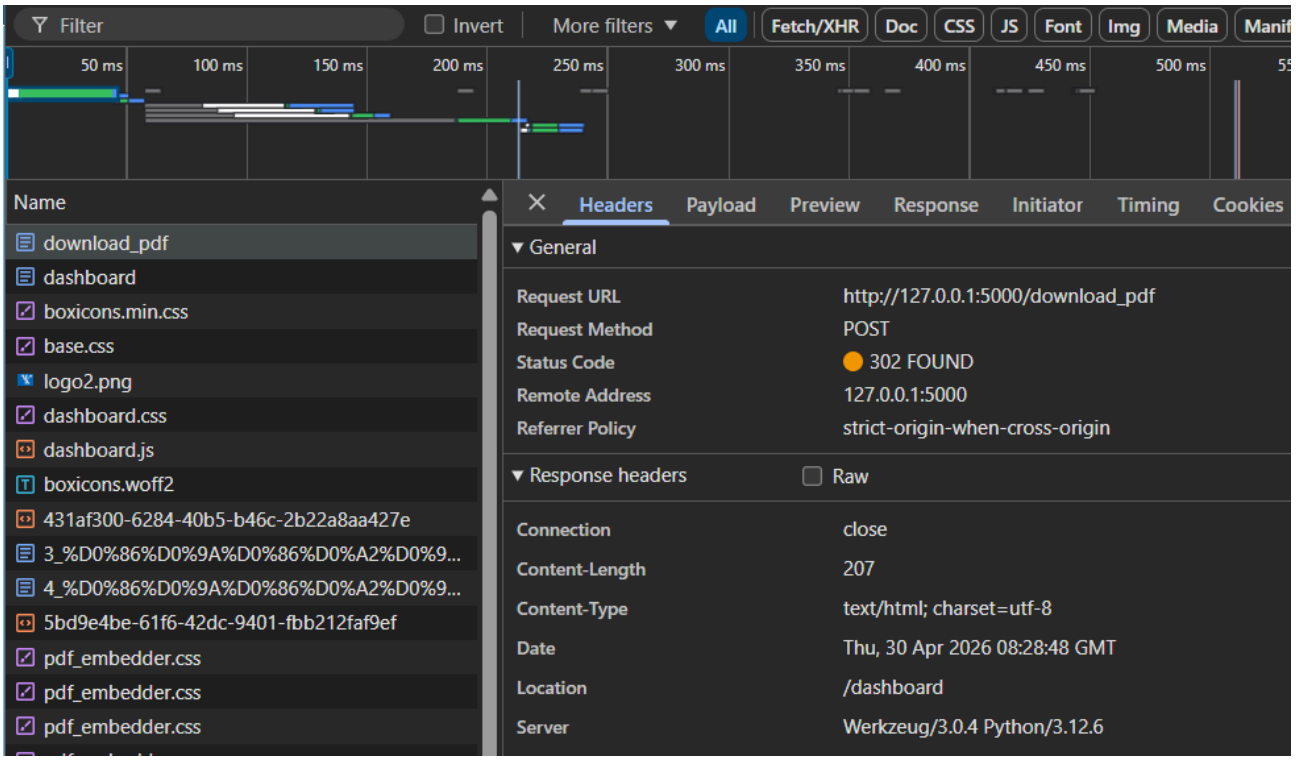

Рисунок 5.8. Перевірка перенаправлення та збереження

5.4Висновок до п’ятого розділу

У п’ятому розділі було проведено тестування розроблених програмних модулів, перевірено валідацію вхидних даних, яка підтвердила коректну та стабільну роботу, алгоритми парсингу масивів даних з фронтенду, розрахунку календарної сітки семестру та конвертації номерів курсу працюють безпомилково.

Також було перевірено адаптивне макетування, тестування підтвердило що програма здатна динамічно зменьшити шрифт у випадках довгих назв дисциплін та прізвищ викладачів, і окремо успішним результатом показло себе розподілення ширини таблиці на колонки, в залежності від кількості груп.

Аналіз кінцевих сценаріїв підтвердив правильну маршрутизацію HTTP-запитів. Система успішно забезпечує обидва випадки, і попередній перегляд без завантаження, і саме завантаження документа на сервер.

Підсумовуючи, всі етапи тестування завершилися успішно. Результати перевірок підтверджують, що розроблений веб застосунок функціонує стабільно, не допускає візуального руйнування верстки та повністю готовий до виконання завдань з автоматизації створення звітної документації.

ВИСНОВОК

В процесі виконання роботи було досягнуто мету розробити та реалізувати веб застосунок для автоматичного формування та візуалізації навчальних розкладів у форматі PDF.

Для досягнення мети було виконано наступні кроки:

- Проаналізовано сучасні проблеми ручного документообігу в освітніх закладах.
- Обґрунтовано необхідність автоматизації
- Проаналізовані вимоги ДСТУ 4163:2020
- Обґрунтовано вибір програмних засобів розробки
- Визначені вимоги та сценарії взаємодії користувача.
- Розроблено модулі форматування даних та генерація документа
- Тестування різних аспектів модулів для генерації документа

Розроблений модуль для веб застосунка полегшить роботу зі створенням розкладів та позбавить рутинну працю. Зводить до мінімуму людський фактор, швидко опрацьовує інформацію та генерує офіційні звіти.

ДЖЕРЕЛА

1. Проблеми електронного документообігу та шляхи їх вирішення - URL:https://magazine.faaf.org.ua/problemi-elektronnogo-dokumentoobigu-ta-shlyahi-ih-virishennya.html (дата звернення 05.01.2026).

2. Переваги електронного документообігу та навіщо він потрібний - URL: https://lawyer.net.ua/ua/statti/elektronnyy-dokumentoobig (дата звернення 05.01.2026).

3. Про електронні документи та електронний документообіг - URL: https://zakon.rada.gov.ua/laws/show/851-15#Text (дата звернення 05.01.2026).

4. Про схвалення Концепції розвитку цифрової економіки та суспільства України на 2018-2020 роки та затвердження плану заходів щодо її реалізації - URL: https://zakon.rada.gov.ua/laws/show/67-2018-%D1%80#Text (дата звернення 05.01.2026).

5. Вимоги ДСТУ 4163:2020 до оформлення документів - URL: https://dzplatforma.com.ua/article/944-pravila-oformlennya-dokumentv-za-dstu-41632020?tokenCoock=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjIxNjA5OTYwMywic2Vzc2lvbklkIjoiMGU4YzA4Y2UtM2MxNC00Y2U0LWI3MDktMzI5YzBkZTUzOWJiIiwidHlwZSI6ImlkVG9rZW4iLCJpYXQiOjE3ODA0ODI4MDIsImV4cCI6MTc5NjAzNDgwMn0.BDfBEGMu1IEZZhLwcvdZJ8gp5G1wpKAFBK-xRgp9VOQ (дата звернення 05.01.2026).

6. EU Digital Identity Wallet запустять до кінця 2026 року — команда Дії протестувала документи - URL: https://thedigital.gov.ua/news/technologies/eu-digital-identity-wallet-zapustyat-do-kintsya-2026-roku-komanda-dii-protestuvala-dokumenti (дата звернення 05.01.2026).

7. ДСТУ 4163:2020. Уніфікована система організаційно-розпорядчої документації. Вимоги до оформлення документів. - Київ : ДП «УкрНДНЦ», 2020. - 26 с.

8. Електронний документообіг в закладі освіти: впровадження, можливості, успішний досвід - URL: https://inbase.com.ua/elektronnyj-dokumentoobig-v-zakladi-osvity-vprovadzhennya-mozhlyvosti-uspishnyj-dosvid/ (дата звернення 07.01.2026).

9. Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA - URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/(дата звернення: 21.01.2026).

10. Що таке десктопні додатки: визначення, переваги та тренди - URL: https://wezom.com.ua/ua/blog/desktop-dodatok (дата звернення: 21.01.2026).

11. PWA vs. MPA vs. SPA – What's the Best Choice for Your App? Neoteric.–2022.–URL: https://neoteric.eu/blog/pwa-vs-mpa-vs-spa-whats-the-best-choice-for-your-app (дата звернення: 21.01.2026).

12. PWA, SPA, AMP – Яку технологію обрати для вашого сайту? - URL: https://whileweb.com/uk/blog/pwa-spa-amp-yaku-tehnologiyu-obrati-dlya-vashogo-sajtu/ (дата звернення: 21.01.2026).

13. Пасічник В. В., Пасічник В. А. Еволюція побудови архітектур інформаційних систем // Вісник Національного університету «Львівська політехніка». Інформаційні системи та мережі. 2017. № 879. С. 268-282.

14. Що відбувається, коли ви вводите URL-адресу в браузері та натискаєте Enter - URL: https://dou.ua/forums/topic/45468/ (дата звернення: 21.01.2026).

15. Http vs https: у чому різниця і як зберегти дані? - URL: https://merge.academy/blog/http-vs-https-u-chomu-riznitsia-i-iak-zbierieghti-dani (дата звернення: 21.01.2026).

16. Overview of HTTP - URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview (дата звернення: 26.01.2026).

17. HTTP request methods - URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods (дата звернення: 26.01.2026).

18. HTTP response status codes - URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status (дата звернення: 27.01.2026).

19. Огляд ERP систем - URL: https://bjetpro.com/blog/oglyad-erp-system (Дата звернення: 03.02.2026).

20. UniTime - URL: https://www.unitime.org (Дата звернення: 03.02.2026).

21. aScTimetables - URL: https://www.asctimetables.com (Дата звернення 03.02.2026)

22. aScTimetables - URL: https://www.timetablemaster.com/asctimetables-alternatives (Дата звернення 03.02.2026)

23. Free Document Management Solutions vs. Paid Solutions - URL: https://medium.com/@revver/free-document-management-solutions-vs-paid-solutions-bcd4b589146f (Дата звернення 03.02.2026)

24. Gimozdo - URL: https://gizmodo.com/download/fet (Дата звернення 03.02.2026)

25. timetablemaster - URL: https://www.timetablemaster.com/fet-alternatives (Дата звернення 03.02.2026)

26. Переваги і недоліки Python - URL: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python (Дата звернення: 06.02.2026)

27. Python Docs - URL: https://docs.python.org/3/ (Дата звернення 13.02.2026)

28. Creating PDF Reports with Pandas, Jinja and WeasyPrint - URL: https://pbpython.com/pdf-reports.html (Дата звернення 13.02.2026)

29. Python PDF Libraries Comparison (Free & Paid Tools) - URL: https://ironpdf.com/python/blog/compare-to-other-components/python-pdf-library-comparison/ (Дата звернення 14.02.2026)

30. PLATYPUS - Page Layout and Typography Using Scripts - URL: https://docs.reportlab.com/reportlab/userguide/ch5_platypus/ (Дата звернення 15.02.2026)

31. PDF creation that actually saves time - URL: https://borbpdf.com (Дата звернення 15.02.2026)

32. Which Is the Best Python Web Framework: Django, Flask, or FastAPI? - URL: https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/ (Дата звернення 17.02.2026)

33. SQLite vs MySQL vs PostgreSQL: A Comprehensive Comparison - URL:https://medium.com/chat2db/sqlite-vs-mysql-vs-postgresql-a-comprehensive-comparison-6e5fc33ecc03 (Дата звернення 17.02.2026)

34. PostgreeSQL: Documentation - URL: https://www.postgresql.org/docs/ (Дата звернення 17.02.2026)

35. SQLite Documentation - URL: https://www.sqlite.org/docs.html (Дата звернення 17.02.2026)

36. MySQL Documentation - URL: https://dev.mysql.com/doc/ (Дата звернення 17.02.2026)

ДОДАТОК А

Лістинг програмного коду

pdf_handler.py

```
from flask import Flask, redirect, request,  make_response,json
from flask_mysqldb import MySQL
import os
import urllib.parse
from datetime import datetime, timedelta
from unicodedata import normalize
from modules.schedClass import Schedule
from modules.groupedTeachers import groupofteacher
from .rimConverter import rim_convert,course_convert
from modules.dbConfig import initialize_database
from modules.schedGen import assign_teachers_with_own_rooms, delete
from .pdf_generator import generate_pdf_file



app = Flask(__name__)
mysql = MySQL(app)
# app.config['MYSQL_HOST'] = 'db'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '12345'
app.config['MYSQL_DB'] = 'maupdb'

initialize_database(app, mysql)

def getDataTeachers(selected_teachers_ids):
    cur = mysql.connection.cursor()
    format_strings = ','.join(['%s'] * len(selected_teachers_ids))
        cur.execute(f"SELECT  *  FROM  teachers  WHERE  ID  IN  ({format_strings})",
tuple(selected_teachers_ids))
    var = cur.fetchall()

    cur.close()
    return var

def forPreviewFunc(arr, groupofteacher, assign_teachers, use_mysql=None):
    schedules = []
    for i in arr:
        var = getDataTeachers(arr[i])
```

```
        forSched = groupofteacher(var)
        schedule = Schedule()
        assign_teachers(forSched, schedule, use_mysql)
        schedules.append(schedule)
    return schedules

def render_data(preview=True):
    group = []
    just_for_pdf = []
    data_group = []
    start_date = request.form.get("start_date")
    start_day = datetime.strptime(start_date, "%Y-%m-%d")
    end_date = request.form.get("end_date")
    end_day = datetime.strptime(end_date, "%Y-%m-%d")

    start_weekday = start_day.weekday()
    end_weekday = end_day.weekday()
     week1 = ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday",
"sunday"]
        week2 = ["monday1", "tuesday2", "wednesday3", "thursday4", "friday5",
"saturday6", "sunday7"]
    week = {}
    for i, day in enumerate(week1):
         offset = i - start_weekday if i >= start_weekday else i - start_weekday +
7
        week[day] = (start_day + timedelta(days=offset)).strftime("%d.%m.%Y")

    week_end = {}
    for i, day in enumerate(week2):
        offset = i - end_weekday if i <= end_weekday else i - end_weekday - 7
        week_end[day] = (end_day + timedelta(days=offset)).strftime("%d.%m.%Y")

     months = {1: "січня", 2: "лютого", 3: "березня", 4: "квітня", 5: "травня", 6:
"червня", 7: "липня", 8: "серпня", 9: "вересня", 10: "жовтня", 11: "листопада",
12: "грудня"}

    monday = week['monday']
    tuesday = week['tuesday']
    wednesday = week['wednesday']
    thursday = week['thursday']
    friday = week['friday']
    saturday = week['saturday']
    sunday = week['sunday']
```

```
    monday1 = week_end['monday1']
    tuesday2 = week_end['tuesday2']
    wednesday3 = week_end['wednesday3']
    thursday4 = week_end['thursday4']
    friday5 = week_end['friday5']
    saturday6 = week_end['saturday6']
    sunday7 = week_end['sunday7']

    day = start_day.day
    month = months[start_day.month]
    year = start_day.year
    year1 = start_day.year

    if request.method == 'POST':
        selected_students_ids = []
        selected_schedule = json.loads(request.form.get('selected_schedule'))
        arr = {}
        for pair in selected_schedule:
            key, value = pair.split(":", 1)
            key, value = key.strip(), value.strip()
            selected_students_ids.append(key)
            if key in arr:
                if isinstance(arr[key], list):
                    arr[key].append(value)
                else:
                    arr[key] = [arr[key], value]
            else:
                arr[key] = [value]
        print(arr)
        cur = mysql.connection.cursor()
        if selected_students_ids:
            format_strings = ','.join(['%s'] * len(selected_students_ids))
             cur.execute(f"SELECT * FROM students WHERE ID IN ({format_strings})",
tuple(selected_students_ids))
            selected_students = cur.fetchall()
        else:
            selected_students = []

        group = [json.loads(student[2])["group"] for student in selected_students]
        data_group = group.copy()
        cur.close()
```

```
    if preview:
                            just_for_pdf    =    forPreviewFunc(arr,    groupofteacher,
assign_teachers_with_own_rooms)

    else:
                            just_for_pdf    =    forPreviewFunc(arr,    groupofteacher,
assign_teachers_with_own_rooms, mysql)
        delete(mysql)



    dek = request.form.get('dekan')
    boss = request.form.get('boss')
    course = (json.loads(selected_students[0][2]))["course"]
    curs = course
    rim_course_help = rim_convert(int(course))
    rim_curs = rim_course_help
    group = (json.loads(selected_students[0][2]))["group"]
    sem_count = course_convert(int(course))
    sem_check_day = start_day.month
    start_date_text = f"{day} {month} {year}"

    if sem_check_day < 6:
        sem_count+=1
        year-=1
    else:
        year1+=1

    institute = selected_students[0][1]
    inst = institute
    if inst:
        params = inst.split('|')
        inst_short = params[0] if len(params) > 1 else None
        inst_long = params[1] if len(params) > 1 else None

    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM teachers")
    var = cur.fetchall()
    cur.close()

    pdf_context = {
        "curs": curs,
        "rim_curs": rim_curs,
```

```
        "inst_short": inst_short,
        "inst_long": inst_long,
        "dek": dek,
        "boss": boss,
        "group": group,
        "data_group": data_group,
        "year": year,
        "start_date_text": start_date_text,
        "monday": monday,
        "tuesday": tuesday,
        "wednesday": wednesday,
        "thursday": thursday,
        "friday": friday,
        "saturday": saturday,
        "sunday": sunday,
        "monday1": monday1,
        "tuesday2": tuesday2,
        "wednesday3": wednesday3,
        "thursday4": thursday4,
        "friday5": friday5,
        "saturday6": saturday6,
        "sunday7": sunday7,
        "sem_count": sem_count,
        "year1": year1,
    }
    pdf_content = generate_pdf_file(just_for_pdf,data_group,pdf_context)
    filename = f"{curs}_{inst_short}_{year}.pdf"
    safe_filename = normalize('NFKC', filename)
    render_name = urllib.parse.quote(filename)
    response = make_response(pdf_content)
    response.headers['Content-Type'] = 'application/pdf'

    if preview:
                            response.headers['Content-Disposition']   =   f'inline;
filename*=UTF-8\'\'{render_name}'
        return response
    else:
        output_dir = os.path.join(os.getcwd(), "src", "static", "artifacts")
        os.makedirs(output_dir, exist_ok=True)
        filepath = os.path.join(output_dir, safe_filename)
        with open(filepath, 'wb') as f:
            f.write(pdf_content)
        return redirect('/dashboard')
```

pdf_generator.py

```
from flask import make_response
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.pdfbase import pdfmetrics
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
from reportlab.lib import colors
from reportlab.platypus import Paragraph
from reportlab.lib.units import cm
from reportlab.pdfbase.pdfmetrics import stringWidth
from reportlab.lib.styles import ParagraphStyle
from reportlab.lib.pagesizes import A4
from io import BytesIO
import os

font_path        =        os.path.join(os.path.dirname(os.path.abspath(__file__)),
'..','..','static', 'fonts', 'times.ttf')
font_path1       =        os.path.join(os.path.dirname(os.path.abspath(__file__)),
'..','..','static', 'fonts', 'timesbd.ttf')
font_path2       =        os.path.join(os.path.dirname(os.path.abspath(__file__)),
'..','..','static', 'fonts', 'timesbi.ttf')
font_path3       =        os.path.join(os.path.dirname(os.path.abspath(__file__)),
'..','..','static', 'fonts', 'timesi.ttf')

pdfmetrics.registerFont(TTFont('TimesNewRoman', font_path))
pdfmetrics.registerFont(TTFont('TimesNewRomanBold', font_path1))
pdfmetrics.registerFont(TTFont('TimesNewRomanBoldItalic', font_path2))
pdfmetrics.registerFont(TTFont('TimesNewRomanItalic', font_path3))

def add_static_text(canvas_obj, ctx):
    canvas_obj.setFont("TimesNewRoman", 14)
    canvas_obj.drawString(3.3 * cm, A4[1] - 80, f"{ctx['curs']} курс")

def get_adaptive_font_size(text, font_name, default_size, max_width):
    size = default_size
    while size >= 6:
        text_width = stringWidth(text, font_name, size)
        if text_width <= max_width:
            break
        size -= 0.5
```

```
    return size


def parse_subject(text):
    parts = text.split(" - ")

    if len(parts) < 3:
        return "", "", ""

    teacher = parts[1].strip()
    subject_part = parts[2]

    room = ""
    subject = subject_part.strip()

    if "(Аудиторія:" in subject_part:
        subject, room = subject_part.split("(Аудиторія:")
        subject = subject.strip()
        room = room.replace(")", " ауд.").strip()

    return subject, teacher, room




def create_cell(paras_text, cell_width):
    subject, teacher, room = parse_subject(paras_text)

    subject_max_width = cell_width - 8
    half_width = (cell_width / 2) - 4

            subj_size   =   get_adaptive_font_size(subject,   'TimesNewRoman',   10,
subject_max_width)
        teach_size  =  get_adaptive_font_size(teacher,  'TimesNewRomanItalic',  8,
half_width)
    room_size = get_adaptive_font_size(room, 'TimesNewRomanBold', 8, half_width)

    subject_p = Paragraph(subject, ParagraphStyle(
        name='subject',
        fontName='TimesNewRoman',
        fontSize=subj_size,
        alignment=1,
```

```
        leading=subj_size + 1
    ))

    teacher_p = Paragraph(teacher, ParagraphStyle(
        name='teacher',
        fontName='TimesNewRomanItalic',
        fontSize=teach_size,
        alignment=0,
        leading=teach_size + 1
    ))

    room_p = Paragraph(room, ParagraphStyle(
        name='room',
        fontName='TimesNewRomanBold',
        fontSize=room_size,
        alignment=2,
        leading=room_size + 1
    ))

    inner_table = Table([
        [subject_p, ""],
        [teacher_p, room_p]
    ], colWidths=['50%', '50%'])

    inner_table.setStyle(TableStyle([
        ('SPAN', (0, 0), (1, 0)),
        ('VALIGN', (0, 0), (1, 0), 'MIDDLE'),
        ('VALIGN', (0, 1), (1, 1), 'BOTTOM'),
        ('LEFTPADDING', (0, 0), (-1, -1), 2),
        ('RIGHTPADDING', (0, 0), (-1, -1), 2),
        ('TOPPADDING', (0, 0), (-1, -1), 0),
        ('BOTTOMPADDING', (0, 0), (-1, -1), 0),
    ]))

    return inner_table


def forPdfFunc(pdfOld,studentsPDF):
    arr = studentsPDF.get_schedule()
    for i in range(4):
        pdfOld[0][1][i].append(arr.get("Понеділок")[i])
        pdfOld[1][1][i].append(arr.get("Вівторок")[i])
        pdfOld[2][1][i].append(arr.get("Середа")[i])
```

```
        pdfOld[3][1][i].append(arr.get("Четвер")[i])
        pdfOld[4][1][i].append(arr.get("П'ятниця")[i])

def generate_pdf_file(theDay, groups,ctx):
    sched = [
              [f"Понеділок\n\n{ctx['monday']}-\n\n{ctx['monday1']} \n", [[f"8:30-
9:50"], ["10:00- 11:20"], ["11:30- 12:50"], ["13:00- 14:20"], ["14:30- 15:50"]]],
              [f"Вівторок\n\n{ctx['tuesday']}-\n\n{ctx['tuesday2']} \n", [["8:30-
9:50"], ["10:00- 11:20"], ["11:30- 12:50"], ["13:00- 14:20"], ["14:30- 15:50"]]],
            [f"Середа\n\n{ctx['wednesday']}-\n\n{ctx['wednesday3']} \n", [["8:30-
9:50"], ["10:00- 11:20"], ["11:30- 12:50"], ["13:00- 14:20"], ["14:30- 15:50"]]],
              [f"Четвер\n\n{ctx['thursday']}-\n\n{ctx['thursday4']} \n", [["8:30-
9:50"], ["10:00- 11:20"], ["11:30- 12:50"], ["13:00- 14:20"], ["14:30- 15:50"]]],
         [f"П'ятниця\n\n{ctx['friday']}-\n\n{ctx['friday5']} \n", [["8:30- 9:50"],
["10:00- 11:20"], ["11:30- 12:50"], ["13:00- 14:20"], ["14:30- 15:50"]]]
    ]

    for i in theDay:
        forPdfFunc(sched,i)



    buffer = BytesIO()

            doc = SimpleDocTemplate(buffer, pagesize=A4, leftMargin=3*cm,
rightMargin=1*cm, topMargin=2*cm, bottomMargin=2*cm)
        text_1_text = Paragraph("\"ЗАТВЕРДЖУЮ\"", ParagraphStyle(name='Normal',
fontName='TimesNewRoman', fontSize=12, alignment=1,leftIndent=140))
           text_2_text = Paragraph("Директор", ParagraphStyle(name='Normal',
fontName='TimesNewRoman', fontSize=12, alignment=1,leftIndent=100,spaceBefore=3,
spaceAfter=3))
                      text_3_text     =     Paragraph(ctx["inst_long"].upper(),
ParagraphStyle(name='Normal',       fontName='TimesNewRoman',       fontSize=12,
alignment=0,leftIndent=260,spaceAfter=13,leading=16))
                                                        text_4_text            =
Paragraph("_____     _____",
ParagraphStyle(name='Normal',        fontName='TimesNewRoman',        fontSize=8,
alignment=0,leftIndent=260,spaceAfter=4))
     text_5_text = Paragraph(ctx["start_date_text"], ParagraphStyle(name='Normal',
fontName='TimesNewRoman', fontSize=12, alignment=1,leftIndent=140))

        maup_text = Paragraph("Розклад занять", ParagraphStyle(name='Normal',
fontName='TimesNewRoman', fontSize=12, alignment=1,spaceBefore=5, spaceAfter=3))
```

```
                term_text    =    Paragraph(f"На    {ctx['sem_count']}    семестр
{ctx['year']}/{ctx['year1']}    навчального    року    денної    форми    навчання
{ctx['rim_curs']}  курс",  ParagraphStyle(name='Normal',  fontName='TimesNewRoman',
fontSize=12, alignment=1, spaceAfter=3))
           ipz_text   =   Paragraph(ctx["inst_long"],ParagraphStyle(name='Normal',
fontName='TimesNewRoman',       fontSize=12,       alignment=1,       spaceAfter=0,
spaceBefore=0,leading=0))
                                                              strip                =
Paragraph("_____
_____",ParagraphStyle(name='Normal',
fontName='TimesNewRoman',       fontSize=9,       alignment=1,       spaceAfter=0,
spaceBefore=10,leading=11))
        dash_text  =  Paragraph("(назва  інституту)",ParagraphStyle(name='Normal',
fontName='TimesNewRoman',       fontSize=6,       alignment=1,       spaceAfter=0,
spaceBefore=0,leading=8))
             dekan_name    =    Paragraph(ctx["dek"],ParagraphStyle(name='Normal',
fontName='TimesNewRoman',       fontSize=14,       alignment=0,       spaceAfter=0,
spaceBefore=5,leading=-77,leftIndent=327))
             boss_name    =    Paragraph(ctx["boss"],ParagraphStyle(name='Normal',
fontName='TimesNewRoman',       fontSize=14,       alignment=0,       spaceAfter=0,
spaceBefore=100,leading=-36,leftIndent=327))


    data_head = ["День тижня", "Пара"] + groups
    data = [data_head]
    day_start_idx = {}
    group_cell_width = (14 * cm) / len(groups)
    for day, subjects in sched:
        day_displayed = False
        for time, *paras in subjects:
         if any(paras[:len(groups)]):
                        time_style  =  Paragraph(time,  ParagraphStyle(name='time',
fontName='TimesNewRoman', fontSize=9, alignment=1, leading=9))
                    group_cells = [create_cell(paras[i], group_cell_width) if i <
len(paras) and paras[i] else "" for i in range(len(groups))]
                row = [day if not day_displayed else "", time_style] + group_cells
                data.append(row)
                if not day_displayed:
                    day_start_idx[day] = len(data) - 1
                day_displayed = True

    col_widths = [2*cm, 1*cm] +[group_cell_width]*len(groups)
    row_heights = [None] * len(data)
    table = Table(data, colWidths=col_widths, rowHeights=row_heights)
```

```
    style = TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.white),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.black),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('VALIGN', (0, 0), (-1, -1), 'TOP'),
        ('VALIGN', (0, 0), (-1, 0), 'MIDDLE'),
        ('FONTNAME', (0, 0), (-1, -1), 'TimesNewRoman'),
        ('FONTSIZE', (0, 0), (-1, -1), 8),
        ('FONTSIZE', (0, 1), (0, -1), 8),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 10),
        ('TOPPADDING', (1, 1), (-1, -1), 2),
        ('TOPPADDING', (0, 0), (-1, -1), 0.5),
        ('LEFTPADDING', (0, 0), (-1, -1), 0),
        ('RIGHTPADDING', (0, 0), (-1, -1), 0),
        ('LEADING', (0, 0), (-1, -1), 6),
        ('BACKGROUND', (0, 1), (-1, -1), colors.white),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),

    ])

    for day, start_idx in day_start_idx.items():
        end_idx = len(data) - 1
        for row in range(start_idx + 1, len(data)):
            if data[row][0] != "":
                end_idx = row - 1
                break
        style.add('SPAN', (0, start_idx), (0, end_idx))
        style.add('VALIGN', (0, start_idx), (0, end_idx), 'TOP')

    table.setStyle(style)
    under_data = [
        ["Декан", "_____", "_____"],
                        ["Заступник начальника управління", "_____",
"_____"]
    ]
    col_widths_1 = [7.5*cm, 4*cm, 6*cm]
    row_heights_1 = [1*cm,0.8*cm]
                                    under_table = Table(under_data,
colWidths=col_widths_1,rowHeights=row_heights_1)
    under_style = TableStyle([
        ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
        ('FONTNAME', (0, 0), (-1, -1), 'TimesNewRoman'),
```

```
        ('FONTSIZE', (0, 0), (-1, -1), 12),
        ('BOTTOMPADDING', (0, 0), (-1, -1), 2),
        ('TOPPADDING', (0, 0), (-1, -1), 2),
    ])
    under_table.setStyle(under_style)
     elements = [text_1_text, text_2_text, text_3_text,text_4_text, text_5_text,
maup_text,                  term_text,                  ipz_text,strip,dash_text,
table,dekan_name,boss_name,under_table,]
                doc.build(elements,    onFirstPage=lambda    canvas_obj,    doc:
add_static_text(canvas_obj, ctx))

    buffer.seek(0)
    response = make_response(buffer.read())
    response.headers['Content-Type'] = 'application/pdf'
    response.headers['Content-Disposition'] = 'inline; filename=FILE_NAME.pdf'
    return buffer.getvalue()
```

ДОДАТОК Б

Демонстраційний вигляд документа

4 курс

"ЗАТВЕРДЖУЮ"
Директор
ІНСТИТУТ
КОМП'ЮТЕРНО-ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

_____ _____

1 вересня 2025