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

Факультет комп'ютерних інформаційних систем і технологій

Кафедра комп'ютерно-інформаційних технологій

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**Тема «Розробка клієнтської частини вебсистеми для генерації розкладу навчального процесу»**

Виконав студент групи ІК-9-22-Б1ІПЗ(4.0д) Овчаренко Р.Д.

Керівник: викладач _____

Допущено до захисту

Завідувач кафедри _____ д.е.н., професор Кавун С.В.

(підпис)

Київ, 2026

# РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 63 с., 30 рис., 26 джерел.

КЛІЄНТСЬКА ЧАСТИНА ВЕБСИСТЕМИ, ГЕНЕРАЦІЯ РОЗКЛАДУ, HTML, CSS, JAVASCRIPT, ВЕБІНТЕРФЕЙС, АДАПТИВНИЙ ДИЗАЙН, FLASK, JINJA2.

Об'єктом розробки є клієнтська частина вебсистеми для генерації розкладу навчального процесу.

Метою роботи є проєктування та розробка клієнтської частини вебсистеми, що забезпечує зручний інтерфейс для перегляду, редагування та керування розкладом навчального процесу, а також надає користувачам швидкий доступ до актуальної інформації незалежно від типу пристрою.

Методи розробки базуються на використанні технологій HTML5, CSS3 та JavaScript для реалізації клієнтської частини вебсистеми з підтримкою адаптивного дизайну та взаємодії з серверною частиною через API. Під час розробки також використовувалися принципи модульної організації інтерфейсу, семантична розмітка HTML та сучасні підходи до створення зручних вебінтерфейсів.

Результатом роботи є розроблена клієнтська частина вебсистеми для генерації розкладу навчального процесу, що включає п'ять функціональних сторінок з розмежуванням доступу для різних категорій користувачів. Реалізований інтерфейс забезпечує перегляд, створення та редагування розкладів, керування даними викладачів і студентських груп, а також формування готових розкладів у форматі PDF.

CLIENT-SIDE OF WEB SYSTEM, SCHEDULE GENERATION, HTML, CSS, JAVASCRIPT, WEB INTERFACE, ADAPTIVE DESIGN, FLASK, JINJA2.

The object of development is the client-side of a web system for generating an academic timetable.

The aim of the work is to design and develop the client-side of a web system that provides a convenient interface for viewing, editing, and managing the academic timetable, as well as ensuring quick access to up-to-date information from different devices.

The development methods are based on HTML5, CSS3, and JavaScript technologies for implementing the client-side of the web system with adaptive design support and interaction with the server-side via API. The development process also involved the use of semantic HTML markup, modular interface organization, and modern approaches to web interface design.

The result of the work is the developed client-side of a web system for academic timetable generation, including five functional pages with differentiated access for different categories of users. The implemented interface provides tools for viewing, creating, and editing schedules, managing information about teachers and student groups, and generating timetable documents in PDF format.

ЗМІСТ

# ВСТУП

Організація навчального процесу є складним завданням, що потребує обов’язкового узгодження декількох його складових:

1. наявність вільних аудиторій в момент проведення навчального процесу;
2. потреб та побажань викладача щодо устаткування аудиторії (комп’ютери, проектор, «розумна» дошка та інше);
3. навантаження викладачів;
4. кількість годин викладання предмету для групи [1].

Традиційний метод складання розкладу полягає у ручному складанні кураторами таблиць для всіх форм навчання однієї спеціальності - денної, денно-дистанційної та дистанційної [2].

Цей метод вимагає приділяти багато часу та уваги, що не виключає «людський» фактор, який є головною причиною помилок, через що у майбутньому в розклад треба вручну вносити зміни, повторно інформувати студентів, що створює незручності для всіх задіяних осіб: кураторів, викладачів та студентів. У зв'язку з цим автоматизація процесу складання розкладу навчального процесу є актуальною задачею для сучасних освітніх установ [3].

Одним із ключових елементів будь-якої вебсистеми є її клієнтська частина — інтерфейс, через який користувач взаємодіє з функціями системи. Якість, зручність та інтуїтивна зрозумілість клієнтської частини безпосередньо впливає на ефективність роботи із системою, тому її розробка вимагає ретельного підходу до проєктування та реалізації [4].

Об'єктом розробки є клієнтська частина вебсистеми для складання розкладу навчального процесу. Метою дипломної роботи є проєктування та розробка клієнтської частини вебсистеми, що забезпечує зручний інтерфейс для формування, редагування та перегляду розкладу навчального процесу [5].

Для виконання поставленої мети необхідно:

1. проаналізувати предметну область та існуючі системи управління розкладом;

2. виконати порівняльний аналіз технологій розробки клієнтської частини вебзастосунків;
3. спроектувати структуру та архітектуру інтерфейсу вебсистеми;
4. розробити HTML-структуру сторінок вебсистеми;
5. розробити стилі та адаптивний дизайн засобами CSS;
6. реалізувати логіку інтерфейсу та взаємодію з сервером засобами JavaScript [5].

Методи дослідження. У роботі застосовано методи порівняльного аналізу існуючих вебзастосунків, методи проєктування користувацького інтерфейсу, а також сучасні стандарти та практики розробки клієнтської частини вебсистем із використанням технологій HTML, CSS та JavaScript [5].

Практичне значення одержаних результатів. Розроблена клієнтська частина вебсистеми може бути впроваджена у закладах вищої освіти для автоматизації процесу формування та перегляду розкладу навчального процесу. Зокрема, систему розроблено в рамках навчального проєкту Міжрегіональної академії управління персоналом (МАУП) та може бути адаптована для практичного використання для інших закладів освіти [5].

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАСОБІВ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБСИСТЕМИ

## 1.1 Опис предметної області

Управління навчальним процесом у закладах вищої освіти охоплює широкий спектр організаційних завдань, серед яких одним із найбільш виснажливим є складання розкладу занять. Розклад є основним документом, що регулює навчальну діяльність усіх учасників освітнього процесу — студентів та викладачів. Від якості його складання повністю залежить ефективність навчального процесу, рівень навантаження на викладачів та студентів, зручність навчання [1].

Процес формування розкладу передбачає врахування великої кількості взаємопов'язаних обмежень і умов. Насамперед, необхідно забезпечити доступність аудиторій у визначений час. При цьому важливо, щоб технічне оснащення аудиторії відповідало вимогам конкретної дисципліни: наявність комп'ютерів для практичних занять з програмування, проектора для лекційних курсів або «розумної» дошки для інтерактивних занять та інше. Аудиторії з різним оснащенням не є взаємозамінними, що суттєво обмежує простір допустимих варіантів розкладу [1].

Крім аудиторій, необхідно враховувати індивідуальне навантаження кожного викладача. Викладач не може проводити одночасно два заняття, а його тижневе навантаження визначається нормами, встановленими навчальним закладом та трудовим законодавством. Додатково можуть існувати побажання викладачів щодо днів або часу проведення занять, які також бажано брати до уваги при формуванні розкладу [1].

Також не менш важливим є дотримання навчальних планів для кожної групи, який визначає перелік дисциплін та кількість годин, призначених на кожну з них протягом семестру. Розклад має рівномірно розподілити ці години впродовж навчального тижня, уникаючи як надмірного перевантаження, так і нераціонального використання навчального часу [2].

Сучасні заклади вищої освіти, зокрема МАУП, підтримують кілька форм навчання в межах однієї спеціальності: денну, денно-дистанційну та дистанційну.

Кожна з цих форм має свої особливості щодо частоти та тривалості занять, що додатково ускладнює процес формування. Для дистанційної форми необхідно враховувати доступність онлайн-платформ та особливості проведення занять у віддаленому форматі [1].

Міжрегіональна академія управління персоналом (МАУП) є одним із найбільших приватних закладів вищої освіти України. Академія має розгалужену структуру, що включає вісім навчально-наукових інститутів та факультетів: Навчально-науковий інститут управління, економіки та бізнесу (ННІУЕБ), Навчально-науковий інститут права імені князя Володимира Великого (ННІП), Навчально-науковий інститут психології та соціальних наук (ННІПСН), Інститут комп'ютерно-інформаційних технологій та дизайну (ІКІТД), Інститут медичних та фармацевтичних наук (ІМФН), Інститут безпеки (ІБ), Інститут освіти дорослих (ІОД) та Інститут міжнародної освіти (ІМО). Кожен інститут має власні навчальні групи на різних курсах та формах навчання, що суттєво ускладнює процес формування загального розкладу [2].

Специфіка МАУП полягає у підтримці трьох форм навчання одночасно в межах одного інституту — денної, денно-дистанційної та дистанційної. Для кожної форми навчання складається окремий розклад із різною частотою та тривалістю занять. Денна форма передбачає щоденні заняття протягом тижня, тоді як денно-дистанційна — чергування очних та дистанційних занять. Для дистанційної форми заняття проводяться переважно онлайн через відповідні платформи. Це означає, що куратор має одночасно складати і узгоджувати декілька варіантів розкладу для груп одного інституту, уникаючи при цьому конфліктів у використанні аудиторій та навантаженні викладачів [2].

Формування розкладу в МАУП наразі здійснюється кураторами деканату переважно вручну із використанням електронних таблиць. Процес починається зі збору від викладачів інформації про зручні пари та дні для проведення занять, після чого куратор вручну розподіляє заняття по аудиторіях з урахуванням їх доступності та оснащення. Готовий розклад узгоджується з адміністрацією та доводиться до відома студентів через оголошення або публікацію у загальних чатах. При

виникненні конфліктів або змін весь процес повторюється частково або повністю. Автоматизація цього процесу засобами вебсистеми дозволить суттєво скоротити час на формування розкладу та зменшити кількість помилок [2].

На теперішній час більшість українських закладів вищої освіти використовують напівавтоматизовані або повністю ручні методи складання розкладу. Куратор вручну формує таблиці занять, а після цього узгоджує їх із викладачами та адміністрацією. Такий підхід займає багато часу, але й систематично призводить до помилок, зумовлених «людським» фактором: співпадіння занять у розкладі викладача або групи, некоректний розподіл аудиторій, порушення встановленого навчального навантаження. Виправлення цих помилок потребує повторного перегляду всього розкладу, внесення змін та додаткового інформування студентів і викладачів, що створює незручності для всіх задіяних осіб [2].

З розвитком інформаційних технологій та широким впровадженням вебтехнологій у сферу освіти з'явилась можливість автоматизувати процес генерації розкладу. Автоматизовані вебсистеми здатні враховувати всі зазначені обмеження одночасно, генерувати оптимальний розклад за лічені секунди та надавати миттєвий доступ до нього через браузер з будь-якого пристрою. Це суттєво знижує адміністративне навантаження на працівників вищого закладу та підвищує загальну якість організації навчального процесу [2].

Клієнтська частина такої системи є ключовим елементом взаємодії користувача з її функціоналом. Саме через інтерфейс адміністратор вводить вихідні дані — перелік аудиторій, викладачів, груп та дисциплін, — запускає процес генерації розкладу та переглядає отриманий результат. Студенти і викладачі, в свою чергу, використовують інтерфейс для перегляду актуального розкладу занять, фільтрації за інститутом та курсом. Якість та зручність клієнтської частини безпосередньо визначає ефективність використання системи в цілому та рівень задоволеності кінцевих користувачів [2].

## 1.2 Аналіз існуючих систем генерації розкладу навчального процесу

На сьогодні існує декілька підходів до автоматизації складання розкладу у закладах вищої освіти. Їх можна умовно поділити на дві категорії: спеціалізовані десктопні програми та онлайн-системи з вебінтерфейсом. У рамках даного аналізу розглянуто два найбільш поширених та показових представники кожної з цих категорій — ASC Timetables та UniTime, — що дозволяє визначити переваги і недоліки існуючих підходів та обґрунтувати доцільність розробки власного рішення [15][16].

1.2.1 Аналіз системи ASC Timetables. ASC Timetables — програма для автоматичного складання розкладу, яка була розроблена словацькою компанією Applied Software Consultants у 1993 році. Вона дозволяє враховувати доступність аудиторій, навантаження викладачів та вимоги навчальних планів. Інтерфейс побудований на основі таблиць і форм, що робить роботу з нею зрозумілою для адміністраторів навчального закладу. Програма підтримує автоматичну генерацію розкладу з урахуванням великої кількості обмежень одночасно. Після створення розклад можна експортувати у форматах PDF та Excel, роздрукувати для кожної групи, викладача або аудиторії, а також опублікувати онлайн. Додатково система підтримує керування замінами викладачів та має мобільний застосунок для перегляду розкладу. Приклад інтерфейсу програми ASC Timetables зображено на рис. 1.1 [15].

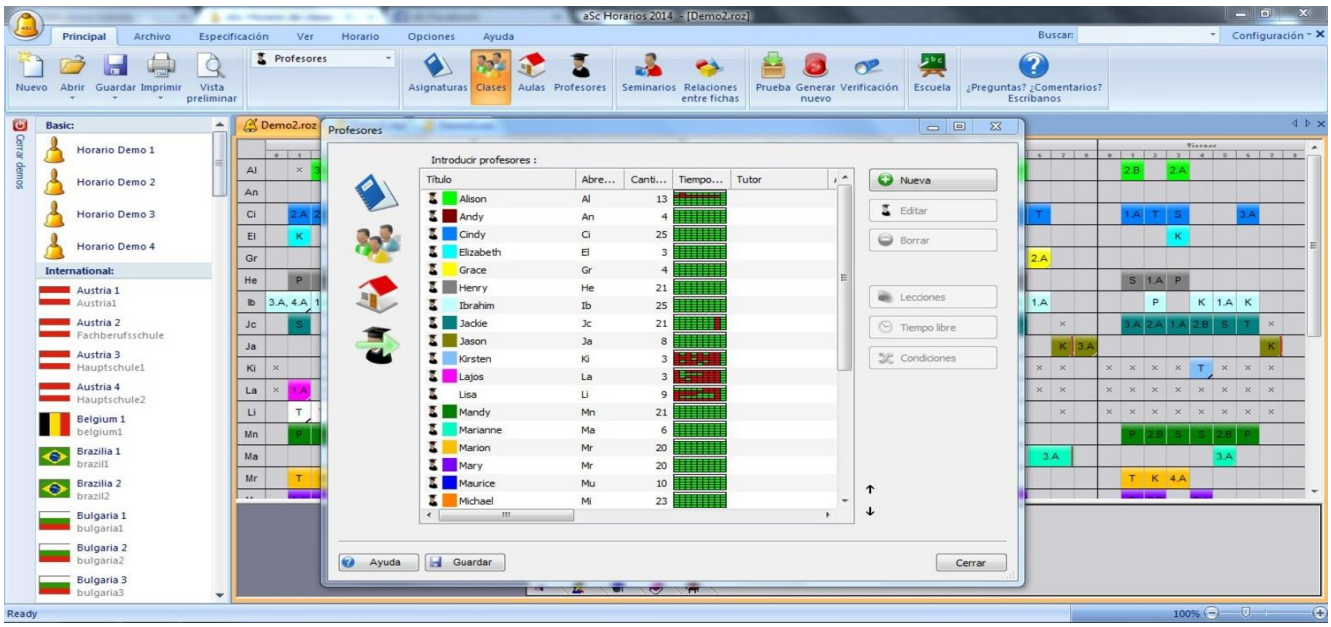


Рис. 1.1 – Інтерфейс програми ASC Timetables

Переваги системи: автоматична генерація розкладу з урахуванням великої кількості обмежень; підтримка експорту у PDF та Excel; наявність безкоштовної пробної версії з повним функціоналом [15].

Недоліки системи: платна ліцензія; інтерфейс морально застарів і не відповідає сучасним стандартам вебдизайну; відсутність повноцінного вбудованого вебінтерфейсу для кінцевих користувачів — студентів і викладачів [15].

У роботі було проаналізовано зовнішній вигляд та функціонал інтерфейсу системи ASC Timetables, що дозволило визначити підхід до відображення розкладу у вигляді таблиці та структуру форм введення даних, які були враховані при проєктуванні клієнтської частини розроблюваної системи [15].

1.2.2. Аналіз системи UniTime. UniTime — система з відкритим вихідним кодом, що підтримує складання розкладу курсів та іспитів, керування змінами в розкладі та розподіл студентів по групах. Система створена у 2008 році. Вона активно використовується у багатьох університетах світу і регулярно оновлюється спільнотою розробників. На відміну від десктопних рішень, UniTime має повноцінний вебінтерфейс, доступний через браузер без встановлення додаткового програмного забезпечення. Система підтримує розподілену роботу — кілька адміністраторів різних підрозділів можуть одночасно працювати з розкладом, узгоджуючи спільні ресурси. Додатково UniTime дозволяє керувати подіями,

резервувати аудиторії та надає студентам доступ до їхнього персонального розкладу [16].

Інтерфейс введення даних системи UniTime надає адміністратору зручні засоби для налаштування курсів, аудиторій та часових слотів. Кожен курс може мати декілька типів занять з індивідуальними налаштуваннями. Інтерфейс введення даних системи UniTime зображено на рис. 1.2 [16].

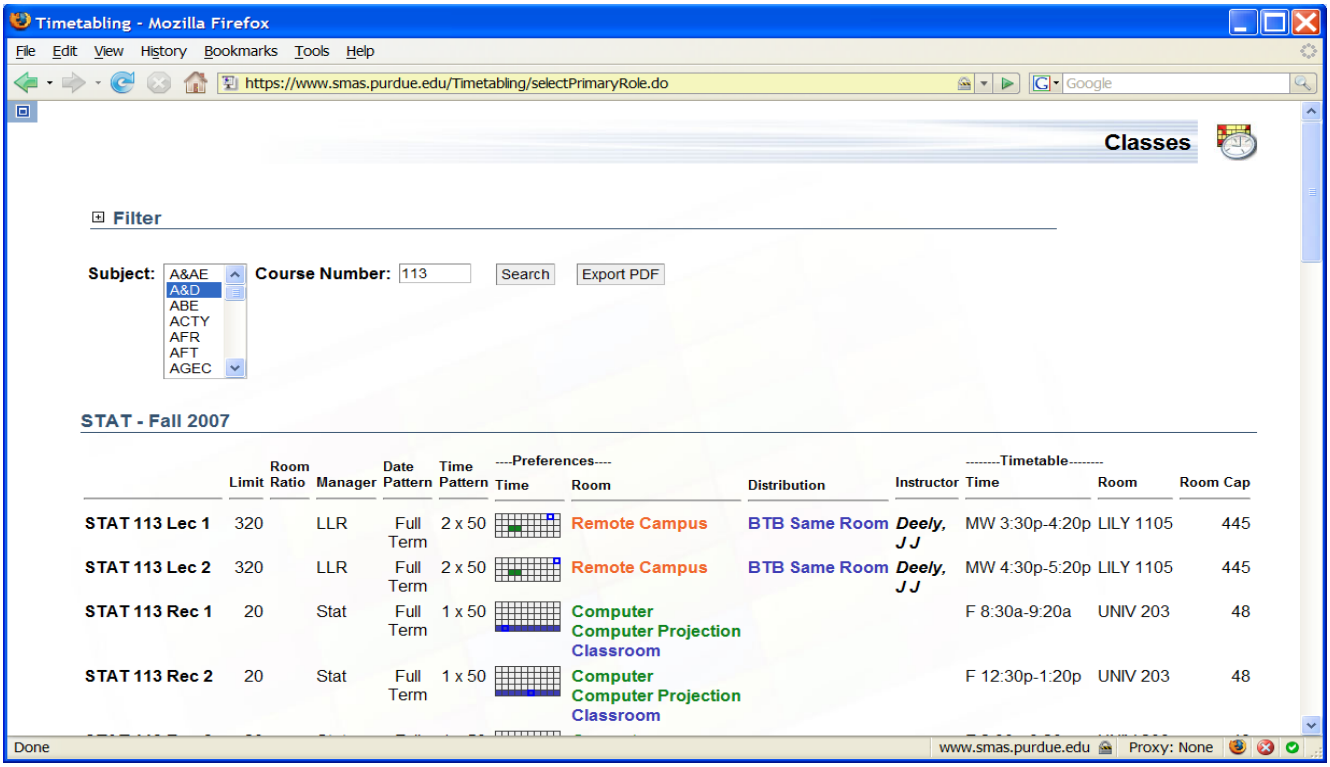


Рис. 1.2. Інтерфейс введення даних системи UniTime

Після завершення налаштування система автоматично генерує оптимальний розклад і відображає його у вигляді кольорової тижневої сітки, де різні типи занять виділяються різними кольорами. Це дозволяє адміністратору швидко оцінити розподіл навантаження та виявити можливі конфлікти. Відображення згенерованого розкладу в системі UniTime зображено на рис. 1.3 [16].

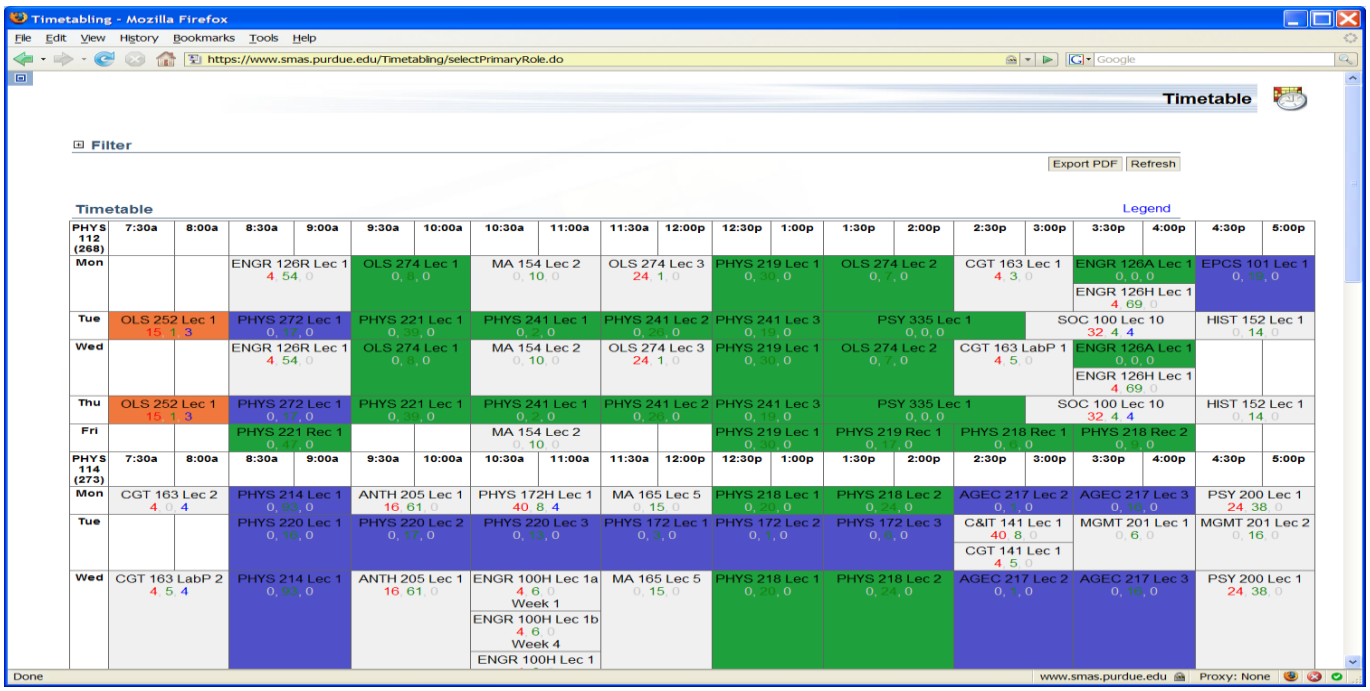


Рис. 1.3. Відображення згенерованого розкладу в системі UniTime

Окрім формування розкладу, система UniTime підтримує модуль розподілу студентів по групах. Студенти подають заявки на курси, після чого система автоматично розподіляє їх по групах з урахуванням побажань та обмежень. Інтерфейс модуля розподілу студентів зображено на рис. 1.4 [16].

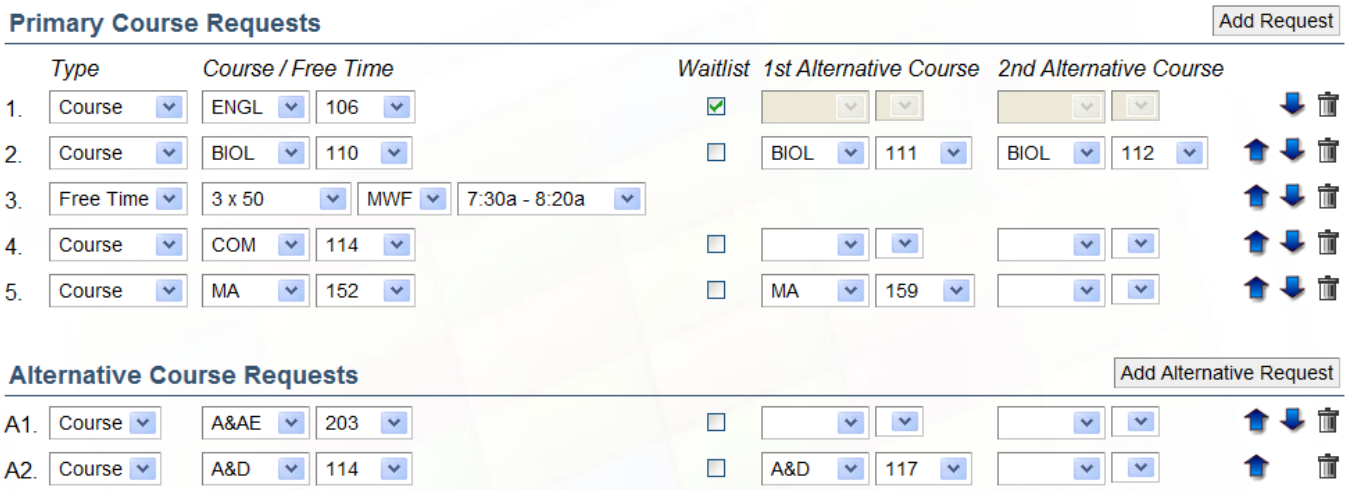


Рис. 1.4. Інтерфейс модуля розподілу студентів системи UniTime

Переваги системи: повністю безкоштовна з відкритим вихідним кодом; вебінтерфейс доступний через браузер без встановлення програм; підтримка великих університетів з багатьма підрозділами [16].

Недоліки системи: складна у налаштуванні та розгортанні; потребує окремого сервера та технічного адміністрування; інтерфейс є перевантаженим і не завжди зручним для звичайного користувача [16].

Аналіз інтерфейсу системи UniTime дозволив визначити доцільність розділення функцій між адміністративною частиною для керування даними та користувацькою частиною для перегляду розкладу, що було реалізовано у розробленій системі [16].

Більш наочне порівняння розглянутих систем за їх характеристиками зображено на рис. 1.5.

| Критерій | ASC Timetables | UniTime | MyTimetable | Розроблювана система |
|---|---|---|---|---|
| Тип доступу | Десктоп | Веб | Веб | Веб |
| Вартість | Платна | Безкоштовна | Умовно безкоштовна | Безкоштовна |
| Складність налаштування | Висока | Висока | Середня | Низька |
| Інтерфейс | Складний | Технічний | Сучасний | Простий та інтуїтивний |
| Підтримка ролей | Обмежена | Розширена | Розширена | Розширена |
| Адаптація під ЗВО | Часткова | Обмежена | Обмежена | Повна |
| Мобільність | Відсутня | Часткова | Повна | Повна |

Рис. 1.5. Порівняльна характеристика існуючих систем генерації розкладу навчального процесу та розроблюваної системи

Аналіз існуючих рішень показує, що більшість з них мають один або кілька суттєвих недоліків: відсутність зручного та інтуітивно зрозумілого вебінтерфейсу, висока вартість ліцензії, складність у налаштуванні або відсутність підтримки специфічних вимог українських закладів освіти, зокрема різних форм навчання в межах однієї спеціальності [2].

У зв'язку з цим розробка власної вебсистеми для генерації розкладу є обґрунтованим рішенням. Така система може бути адаптована під конкретні потреби навчального закладу та надавати зручний вебінтерфейс для всіх категорій користувачів — адміністраторів, викладачів і студентів. Клієнтська частина подібної системи має забезпечувати просту та зрозумілу взаємодію з її функціоналом, що і є метою даної роботи [11].

## 1.3 Формування мети та завдань роботи

Метою кваліфікаційної роботи є проєктування та розробка клієнтської частини вебсистеми для генерації розкладу навчального процесу, яка забезпечує зручний та інтуітивно зрозумілий інтерфейс для всіх категорій користувачів системи [12].

Розроблена клієнтська частина має відповідати сучасним вимогам до вебінтерфейсів: забезпечувати коректне відображення на різних пристроях та розмірах екрану, мати зрозумілу навігацію між сторінками та інтуїтивно зрозумілий порядок дій для кожного типу користувача. Реалізація інтерфейсу виконується з використанням технологій HTML, CSS та JavaScript без залучення зовнішніх фреймворків, що забезпечує легкість підтримки та інтеграції з серверною частиною системи [11].

Клієнтська частина має взаємодіяти з серверною частиною системи через програмний інтерфейс (API), отримуючи та надсилаючи дані у форматі JSON засобами JavaScript. Це забезпечує динамічне оновлення складових сторінки без їх повного перезавантаження та покращує загальний досвід користувача під час роботи з системою [11].

Для досягнення поставленої мети необхідно виконати такі завдання:

1. проаналізувати предметну область та особливості процесу складання розкладу навчального процесу;
2. виконати аналіз існуючих систем управління розкладом та визначити їх переваги і недоліки;
3. виконати порівняльний аналіз технологій розробки клієнтської частини вебзастосунків;
4. спроєктувати структуру сторінок та архітектуру інтерфейсу вебсистеми;
5. розробити HTML-структуру всіх сторінок вебсистеми;
6. розробити стилі оформлення та адаптивний дизайн засобами CSS;
7. реалізувати логіку інтерфейсу та взаємодію з серверною частиною засобами JavaScript;

8. проведено функціональне тестування, тестування адаптивності та валідації даних клієнтської частини;
9. перевірити коректність роботи розробленого інтерфейсу та проаналізувати результати розробки.

## 1.4 Висновки до першого розділу

Проаналізував предметну область, було встановлено, що складання розкладу навчального процесу є складним організаційним завданням, яке потребує одночасного врахування великої кількості обмежень — доступності аудиторій, навантаження викладачів, навчальних планів груп та особливостей різних форм навчання. Традиційний ручний підхід до складання розкладу є тривалим і схильним до помилок, що підтверджує актуальність розробки автоматизованої вебсистеми для вирішення цієї задачі [1].

Проаналізував існуючі програмні рішення для генерації розкладу — десктопну систему ASC Timetables та веборієнтовану систему UniTime — можна зробити висновок, що кожна з них має суттєві недоліки. ASC Timetables є зручною у роботі, проте не має повноцінного вебінтерфейсу для кінцевих користувачів і потребує придбання ліцензії. UniTime є безкоштовною і має вебінтерфейс, проте складна у налаштуванні та розгортанні, що робить її малопридатною для невеликих закладів без власного ІТ-відділу. Жодне з розглянутих рішень не є повністю придатним для використання в умовах українського закладу вищої освіти без значної адаптації [15][16].

Виходячи з цього, можна зробити висновок про доцільність розробки власної вебсистеми з якісною клієнтською частиною, яка забезпечить зручний доступ до функцій системи для адміністраторів, викладачів і студентів через браузер. Розробка клієнтської частини з використанням HTML, CSS та JavaScript дозволить створити легкий у підтримці та інтеграції інтерфейс, адаптований під конкретні потреби навчального закладу [11].

# РОЗДІЛ 2. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ

## 2.1 Аналіз схожих вебзастосунків для управління розкладом

Окрім спеціалізованих десктопних програм, розглянутих у першому розділі, на сьогодні існує декілька вебзастосунків, які надають користувачам можливість переглядати та керувати розкладом навчального процесу через браузер. Такі рішення є більш доступними для кінцевого користувача, оскільки не потребують встановлення додаткового програмного забезпечення і можуть використовуватися з будь-якого пристрою. Аналіз існуючих вебзастосунків дозволяє визначити ключові вимоги до інтерфейсу клієнтської частини розроблюваної системи — зокрема щодо структури сторінок, способу відображення розкладу, навігації між розділами та розподілу функцій між різними типами користувачів [11].

Розклад НТУУ «КПІ ім. Ігоря Сікорського» — один із найвідоміших і найбільш опрацьованих прикладів вебінтерфейсу для перегляду університетського розкладу в Україні. Система доступна за адресою rozklad.kpi.ua і дозволяє переглядати розклад для будь-якої групи, викладача або аудиторії університету. Головна сторінка системи містить поле пошуку, через яке користувач може знайти потрібну групу або викладача, після чого відкривається детальний розклад на поточний тиждень [17].

Інтерфейс побудований на основі тижневої сітки занять, де кожне заняття відображається як окремий блок із назвою дисципліни, типом заняття (лекція, практичне заняття або лабораторна робота), іменем викладача та номером аудиторії. Різні типи занять виділяються кольором, що дозволяє швидко орієнтуватися в розкладі. Система підтримує перемикання між тижнями, відображення розкладу для студентів та викладачів, а також має відображення для розкладу сессії, адаптивний дизайн для коректного відображення робить перегляд зручним і на мобільних пристроях [17].

З технічної точки зору система реалізована як односторінковий вебзастосунок, що динамічно завантажує дані з серверної частини без перезавантаження сторінки.

Це забезпечує швидку роботу інтерфейсу та комфортний досвід користувача під час перегляду розкладу [17].

Переваги системи: зрозуміла та лаконічна структура відображення розкладу у вигляді тижневої сітки; швидкий пошук за групою, викладачем або аудиторією; кольорове розрізнення типів занять; адаптивний дизайн для мобільних пристроїв; динамічне завантаження даних без перезавантаження сторінки [17].

Недоліки системи: відсутність можливості редагування розкладу через інтерфейс; немає адміністративної панелі для керування даними; система розроблена виключно під потреби КПІ і не може бути адаптована для інших закладів без суттєвих змін у коді. Приклад інтерфейсу системи зображено на рис. 2.1 [17].

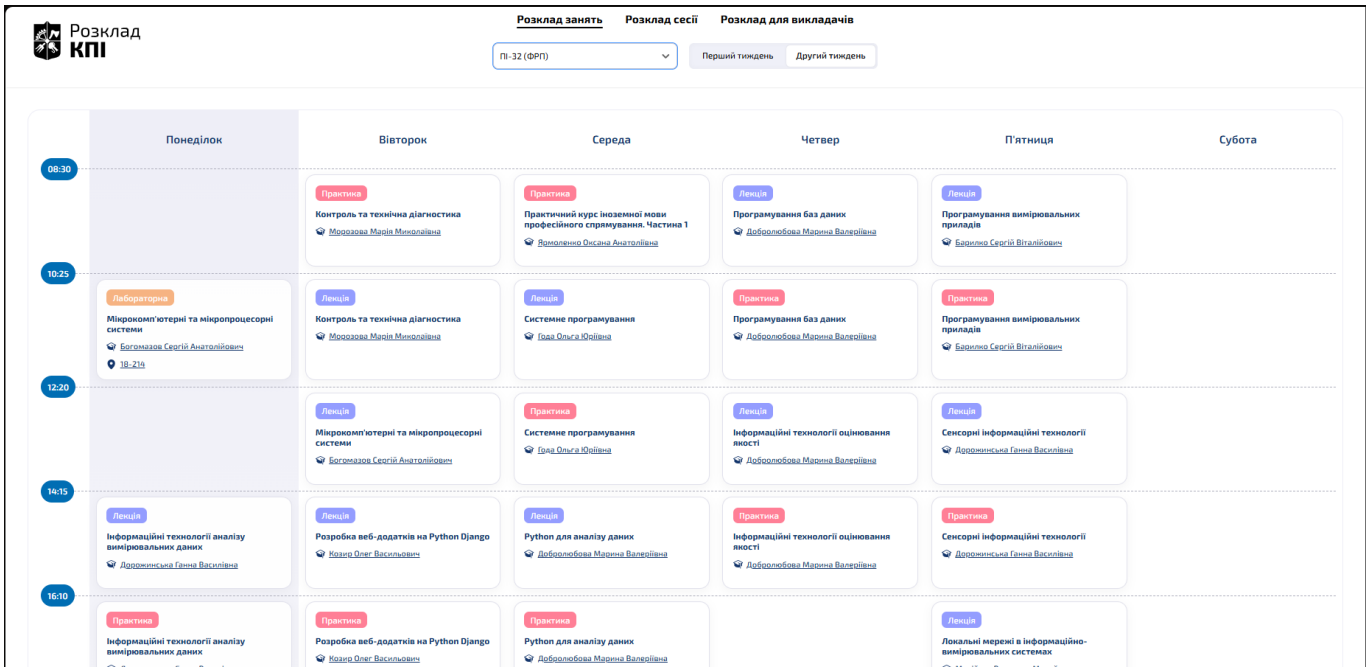


Рис. 2.1. Інтерфейс вебсистеми розкладу КПІ

MyTimetable — комерційний онлайн-сервіс для створення та перегляду розкладу, орієнтований на навчальні заклади різного рівня. Сервіс надає повноцінний вебінтерфейс як для адміністраторів, так і для кінцевих користувачів — студентів і викладачів. Адміністратор через окрему панель керування вводить дані про групи, викладачів та аудиторії, після чого система генерує розклад і надає студентам та викладачам посилання для його перегляду [18].

Для додавання розкладу до персонального перегляду користувач використовує спеціальну форму, через яку обирає потрібні дисципліни або групи зі списку доступних. Це дозволяє кожному студенту або викладачу налаштувати відображення лише тих занять, які стосуються безпосередньо його. Форму додавання розкладу зображено на рис. 2.2 [18].

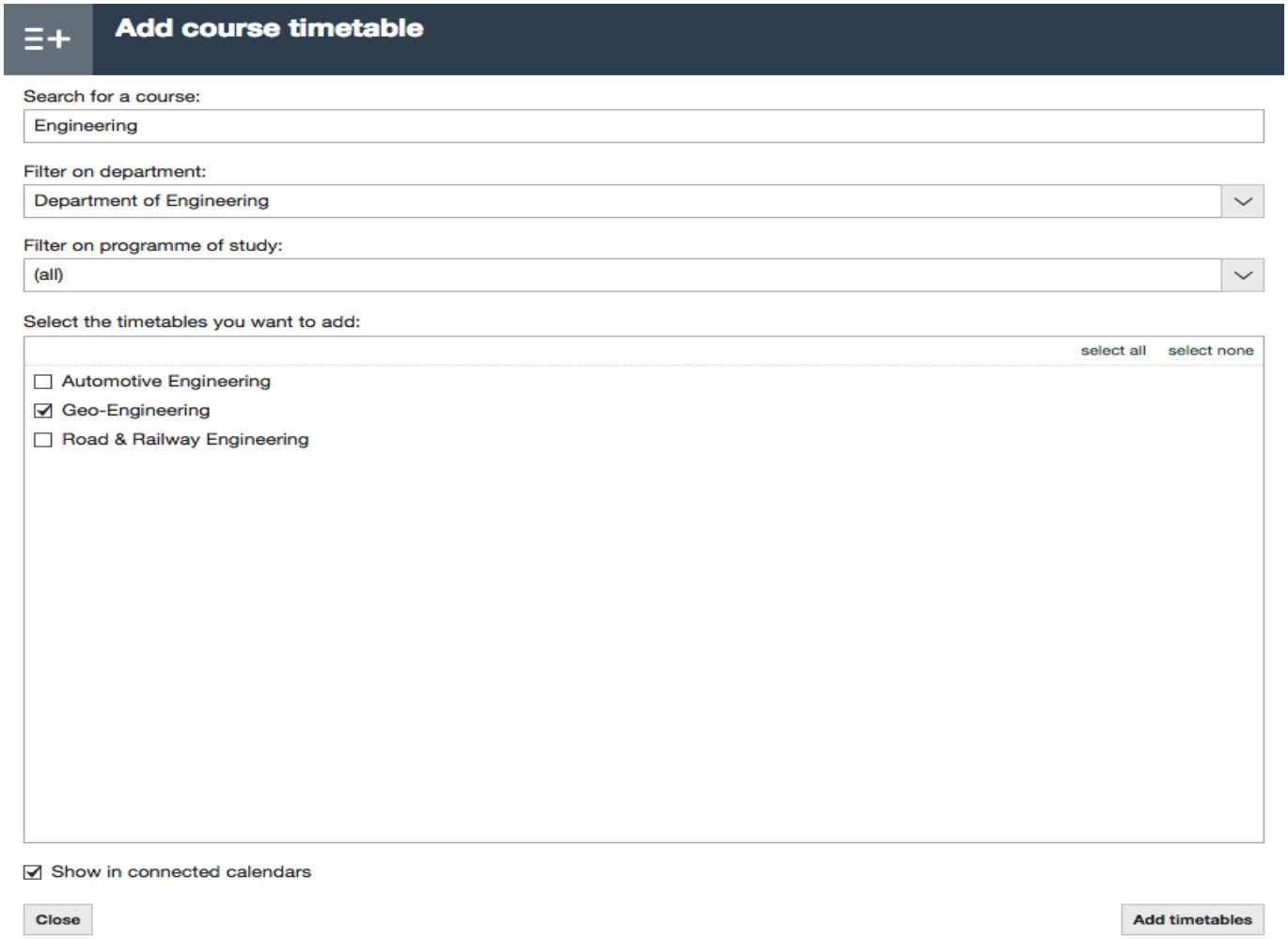


Рис. 2.2. Форма додавання розкладу в системі MyTimetable

Після вибору дисциплін система відображає повну таблицю розкладу у вигляді тижневої сітки, де по горизонталі розташовані дні тижня, а по вертикалі — години. Кожне заняття відображається як окремий блок із зазначенням назви дисципліни, типу заняття, аудиторії та викладача. Передбачена можливість перемикання між режимами відображення — за днем, тижнем або місяцем, а також мінікалендар для швидкого переходу до потрібної дати. Повну таблицю розкладу зображено на рис. 2.3 [18].

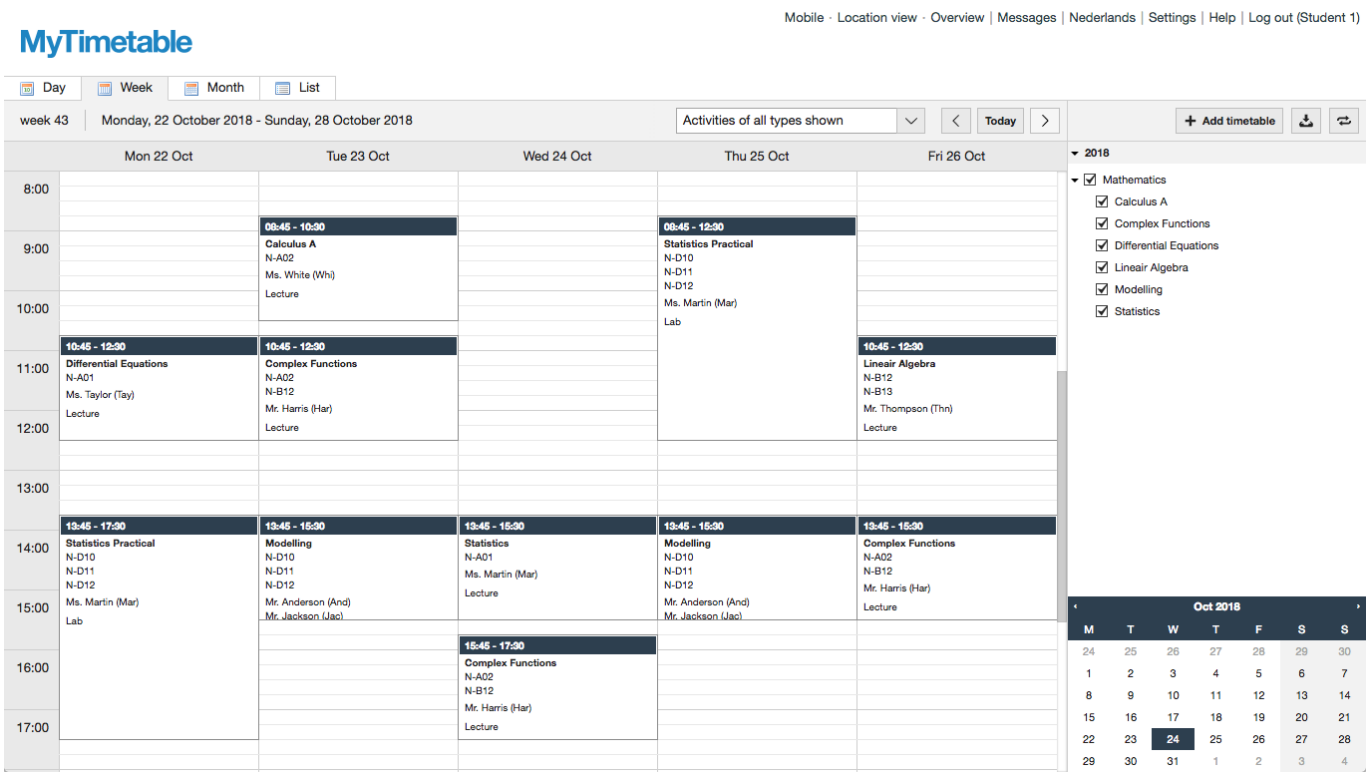


Рис. 2.3. Таблиця розкладу в системі MyTimetable

Сервіс має сучасний мінімалістичний дизайн, підтримує кольорове кодування різних дисциплін та інтеграцію з популярними календарними застосунками — Google Calendar та Apple Calendar. Це дозволяє користувачам автоматично синхронізувати розклад зі своїм смартфоном [18].

Переваги системи: сучасний та зрозумілий інтерфейс із мінімалістичним дизайном; чітке розділення ролей між адміністратором і кінцевим користувачем; персоналізований перегляд розкладу для кожного користувача; інтеграція з Google Calendar та Apple Calendar; доступність через браузер без встановлення програм [18].

Недоліки системи: платна ліцензія для повного функціоналу; інтерфейс англомовний і без знання англійської його використовування неможливе; відсутність підтримки специфічних вимог українських закладів освіти, зокрема різних форм навчання в межах однієї спеціальності [18].

Аналіз розглянутих вебзастосунків дозволяє виявити спільні підходи до побудови інтерфейсу систем управління розкладом. Усі розглянуті рішення використовують табличну або сіткову структуру для відображення розкладу — це є найбільш зрозумілим і звичним для користувача форматом, який дозволяє швидко

знайти потрібне заняття за днем тижня та часом. Кольорове розрізнення типів занять також є поширеною практикою, що суттєво покращує сприйняття інформації [11].

Важливою спільною рисою є розділення інтерфейсу на дві частини — адміністративну, через яку вводяться та редагуються дані, і користувацьку, через яку студенти та викладачі переглядають розклад. Такий підхід забезпечує зручність для кожної категорії користувачів окремо і не перевантажує інтерфейс зайвими функціями [11].

Виявлені особливості враховуються при проєктуванні клієнтської частини розроблюваної системи. Зокрема, передбачається реалізація табличного відображення розкладу, фільтрація за групою та курсом, а також окремий інтерфейс для адміністратора з можливістю керування даними системи. Реалізація цих вимог засобами HTML, CSS та JavaScript є основним завданням даної роботи [11].

Таким чином, аналіз зовнішнього вигляду та функціоналу розглянутих вебзастосунків дозволив сформулювати конкретні вимоги до інтерфейсу клієнтської частини розроблюваної системи, які були реалізовані в рамках даної роботи [11].

## 2.2 Порівняння технологій розробки клієнтської частини (HTML, CSS, JavaScript)

Для того щоб створити клієнтську частину будь-якого вебзастосунку, розробник використовує три основні технології: HTML, CSS та JavaScript. Ці технології завжди працюють разом і кожна з них відповідає за свою частину роботи. HTML відповідає за елементи сторінки, CSS — за вигляд сторінки, а JavaScript — за те, як це працює та реагує на дії користувача. Без жодної з цих складових повноцінний вебінтерфейс створити неможливо. У даному підрозділі розглядається кожна з цих технологій окремо, описуються їх можливості та пояснюється, чому саме цей набір було обрано для розробки клієнтської частини системи генерації розкладу [11].

HTML (HyperText Markup Language) — це мова розмітки, за допомогою якої описується структура вебсторінки. Простіше кажучи, HTML — це скелет сторінки. Саме в HTML-коді вказується, де буде заголовок, де таблиця, де кнопка, де поле для

введення тексту. Браузер читає HTML-код і на його основі будує те, що бачить користувач на екрані. Поточна версія мови — HTML5, яка була прийнята як стандарт у 2014 році [20].

HTML5 додав до мови так звані семантичні теги — <header>, <nav>, <main>, <section>, <footer> та інші. Раніше розробники використовували для всього один і той самий тег , що робило код важким для читання. Семантичні теги дозволяють одразу зрозуміти, яку роль виконує той чи інший блок на сторінці — це шапка сайту, це навігаційне меню, це основний вміст тощо. Завдяки цьому код стає значно зрозумілішим і простішим у підтримці [23].

Для розроблюваної системи HTML є основою всіх сторінок — сторінки входу, сторінки перегляду розкладу, адміністративної панелі та інших. HTML5 також має вбудовану підтримку різних типів полів у формах — текстові поля, випадаючі списки, перемикачі, кнопки — що є необхідним для реалізації форм введення даних в адміністративній частині системи. Браузер сам перевіряє правильність заповнення таких полів без необхідності писати додатковий код [23].

Важливою особливістю HTML5 є також підтримка атрибутів даних — data-*. Ці атрибути дозволяють зберігати довільні дані безпосередньо в HTML-елементах без використання прихованих полів або глобальних змінних. У розробленій системі цей механізм активно використовується: картки розкладів на головній сторінці мають атрибути data-institute та data-course, а кнопки редагування — data-id, data-fullname, data-subjects тощо. Це дозволяє JavaScript зчитувати необхідні дані безпосередньо з DOM без додаткових запитів до сервера [23].

Основні переваги HTML для даного проєкту: підтримується абсолютно всіма сучасними браузерами; семантична розмітка робить код зрозумілим і зручним для подальшої роботи; вбудована підтримка форм і таблиць, які активно використовуються в інтерфейсі системи розкладу; атрибути data-* дозволяють ефективно передавати дані між HTML та JavaScript [23].

CSS (Cascading Style Sheets) — це мова стилів, яка відповідає за зовнішній вигляд HTML-елементів. Якщо HTML — це скелет, то CSS — це одяг. За допомогою CSS задаються кольори, шрифти, розміри елементів, відступи між ними,

фони, рамки та загальне розташування блоків на сторінці. Головна ідея CSS полягає в тому, що стилі зберігаються окремо від HTML-коду — в окремому файлі. Завдяки цьому можна змінити зовнішній вигляд усіх сторінок сайту одразу, відредагувавши лише один файл стилів [20].

Поточна версія — CSS3, яка значно розширила можливості мови порівняно з попередніми версіями. Однією з найважливіших можливостей CSS3 є підтримка адаптивного дизайну через так звані медіазапити. Медіазапити дозволяють вказати, які стилі застосовувати залежно від розміру екрану пристрою. Наприклад, на великому моніторі розклад може відображатися у вигляді широкої таблиці на весь екран, а на смартфоні — у вигляді вертикального списку, зручного для читання на маленькому екрані. Це важливо, оскільки багато студентів переглядають розклад саме з телефону [24].

CSS3 також містить два сучасних інструменти для побудови розкладки сторінки — Flexbox та Grid. Flexbox зручно використовувати для вирівнювання елементів в один рядок або стовпець — наприклад, для розміщення кнопок навігації або елементів шапки сторінки. Grid дозволяє будувати складніші двовимірні сітки — наприклад, таблицю розкладу, де рядки це пари, а стовпці це дні тижня. Обидва інструменти суттєво спрощують верстку порівняно зі старими підходами і активно використовуються в даній роботі [25].

Додатковою можливістю CSS3 є підтримка CSS-змінних — так званих кастомних властивостей. Вони дозволяють визначити певне значення один раз і використовувати його по всьому коду стилів. Наприклад, основний корпоративний колір системи #0064b4 можна оголосити як змінну :root { --main-color: #0064b4; } і надалі використовувати скрізь через var(--main-color). Якщо виникне потреба змінити колір — достатньо змінити лише одне місце. Це суттєво спрощує підтримку та оновлення дизайну [25].

Основні переваги CSS для даного проєкту: повне відокремлення стилів від HTML-коду спрощує підтримку та внесення змін до дизайну; підтримка адаптивного дизайну через медіазапити забезпечує коректне відображення на різних пристроях; Flexbox та Grid дозволяють легко будувати складні інтерфейси [25].

JavaScript — це мова програмування, яка додає до вебсторінки інтерактивність та логіку. Якщо HTML і CSS — це статична картинка, то JavaScript робить її живою. За допомогою JavaScript можна реагувати на кліки користувача, перевіряти введені дані у формах, динамічно змінювати вміст сторінки без її перезавантаження, а також надсилати запити до сервера і отримувати відповіді. JavaScript є єдиною мовою програмування, яка вбудована в усі сучасні браузери і не потребує жодного додаткового встановлення [20].

Для розроблюваної системи JavaScript виконує кілька важливих завдань. По-перше, він відповідає за взаємодію з серверною частиною системи — збір даних з форм, їх серіалізацію у формат JSON та відправку на сервер. По-друге, JavaScript має повний доступ до всіх елементів HTML-сторінки через модель DOM, що дозволяє динамічно створювати, змінювати або видаляти елементи сторінки. По-третє, JavaScript відповідає за валідацію введених даних на стороні клієнта ще до їх відправки на сервер, що зменшує кількість зайвих запитів [26].

Важливим інструментом JavaScript у даному проєкті є метод JSON.stringify() та JSON.parse(). Оскільки HTML-форми за замовчуванням можуть передавати лише прості текстові значення, для передачі складних структур — наприклад, розкладу викладача у вигляді об'єкту з масивами — їх спочатку серіалізують у JSON-рядок і записують у приховане поле форми, а на сервері десеріалізують назад. Це простий і надійний спосіб передачі складних даних без використання AJAX [26].

Основні переваги JavaScript для даного проєкту: вбудована підтримка у всіх браузерах; можливість динамічної зміни інтерфейсу через DOM; серіалізація складних структур через JSON; валідація даних на стороні клієнта [26].

Порівняння з альтернативними підходами. Варто також розглянути альтернативні технологічні підходи до розробки клієнтської частини та пояснити чому їх не було обрано. Найпопулярнішими альтернативами є JavaScript-фреймворки — React, Vue.js та Angular [20].

React — бібліотека від компанії Meta для побудови інтерфейсів на основі компонентів. Вона підходить для великих застосунків зі складним станом, проте потребує встановлення Node.js, налаштування збірника модулів (Webpack або Vite)

та знання концепції JSX. Для масштабу розроблюваної системи це є надлишковим [20].

Vue.js — прогресивний фреймворк з більш простим порогом входу ніж React. Він також потребує системи збірки для повноцінного використання, хоча може підключатися і через CDN. Концептуально Vue.js міг би підійти, однак додає шар абстракції, який ускладнює розуміння базових механізмів роботи браузера [20].

Angular — повноцінний фреймворк від Google з вбудованим TypeScript, системою маршрутизації та ін'єкцією залежностей. Він призначений для великих корпоративних застосунків і є очевидно надмірним для системи з п'яти сторінок [20]. Порівняння обраного підходу з альтернативами наведено на рис. 2.4.

| Критерій | HTML/CSS/JS | React | Vue.js | Angular |
|---|---|---|---|---|
| Потреба у збірці | Не потрібна | Обов'язкова | Бажана | Обов'язкова |
| Поріг входу | Низький | Середній | Середній | Високий |
| Залежності | Відсутні | Багато | Помірно | Багато |
| Підходить для малих проєктів | Так | Ні | Частково | Ні |
| Підтримка браузерами | Нативна | Потребує компіляції | Потребує компіляції | Потребує компіляції |

Рис. 2.4. Порівняння технологічних підходів до розробки клієнтської частини

Таким чином, обраний набір технологій — HTML5, CSS3 та JavaScript — є оптимальним рішенням для розробки клієнтської частини вебсистеми генерації розкладу. Кожна з технологій виконує свою чітко визначену роль, разом вони забезпечують усі необхідні можливості для реалізації поставлених завдань, а їх поєднання є стандартною і перевіреною практикою у веброзробці [20].

## 2.3 Висновки до другого розділу

Проаналізувавши існуючі вебзастосунки для управління розкладом, зокрема систему розкладу КПІ та сервіс MyTimetable, було визначено основні підходи до побудови інтерфейсу подібних систем. Встановлено, що найбільш зрозумілим для користувача є відображення розкладу у вигляді тижневої сітки або таблиці з кольоровим розрізненням типів занять. Важливою особливістю є також чітке розділення інтерфейсу на адміністративну частину для керування даними та користувацьку частину для перегляду розкладу. Виявлені закономірності враховано при проєктуванні клієнтської частини розроблюваної системи [17][18].

Розглянувши технології розробки клієнтської частини, було обґрунтовано вибір стеку HTML5, CSS3 та JavaScript як основного інструментарію для даної роботи. HTML5 забезпечує структуру сторінок і підтримку семантичної розмітки та форм. CSS3 відповідає за зовнішній вигляд інтерфейсу, адаптивність дизайну для різних пристроїв та побудову складних розкладок за допомогою Flexbox і Grid. JavaScript забезпечує інтерактивність інтерфейсу, динамічну взаємодію з серверною частиною через Fetch API та керування елементами сторінки через DOM [20][22].

Використання чистого HTML, CSS та JavaScript без додаткових фреймворків є обґрунтованим рішенням для даного проєкту, оскільки забезпечує всі необхідні можливості без зайвої складності та залежностей від зовнішніх бібліотек [23][24].

# РОЗДІЛ 3. ОПИС РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБСИСТЕМИ

## 3.1 Проєктування архітектури та структури інтерфейсу

Перед початком розробки клієнтської частини вебсистеми було виконано проєктування її архітектури та структури. Цей етап є важливим, оскільки дозволяє чітко визначити з яких сторінок складається система, як вони пов'язані між собою та які функції доступні кожному типу користувача. Завчасне проєктування структури дозволяє уникнути помилок під час розробки та полегшує подальшу підтримку коду [12].

Розроблена вебсистема орієнтована на три типи користувачів: студента, викладача та адміністратора (куратора). Кожен з них має різний рівень доступу до функцій системи. Студент може лише переглядати затверджені розклади своєї групи. Викладач має доступ до перегляду власного розкладу. Адміністратор має повний доступ до всіх функцій системи — створення та редагування викладачів, груп і розкладів [12]. Структурну схему вебсистеми зображено на рис. 3.1.

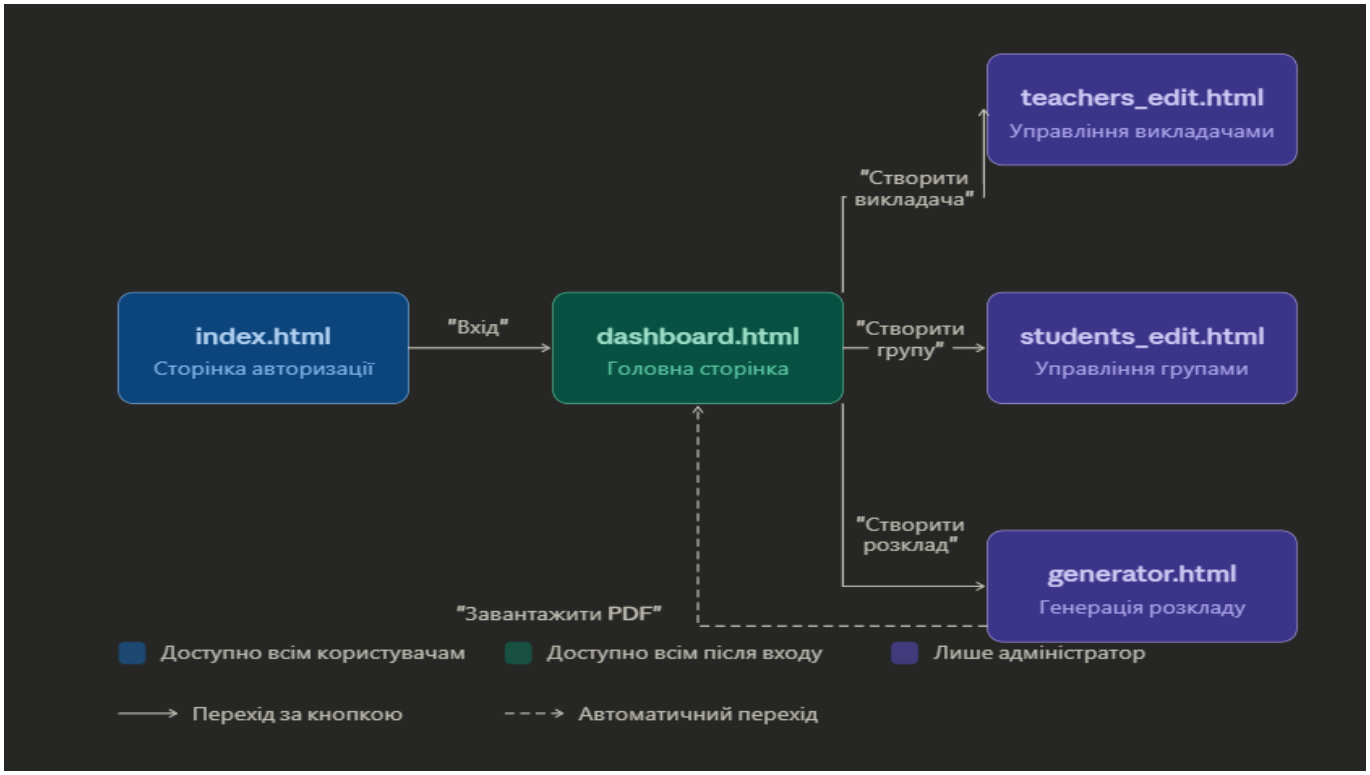


Рис. 3.1. Структурна схема вебсистеми

Система складається з п'яти основних сторінок, кожна з яких виконує свою окрему роль. Розглянемо кожну з них детальніше.

3.1.1. Сторінка авторизації (index.html). Це перша сторінка, яку бачить будь-який користувач при відкритті системи. Вона містить форму з двома полями — логін та пароль — і кнопку входу. Також на сторінці присутнє посилання "Увійти як студент", яке дозволяє студентам потрапити на головну сторінку без введення облікових даних. У шапці сторінки розміщені логотип та назва МАУП, а також посилання на головний сайт університету. Після успішної авторизації користувач перенаправляється на головну сторінку системи. Макет сторінки авторизації зображено на рис. 3.2

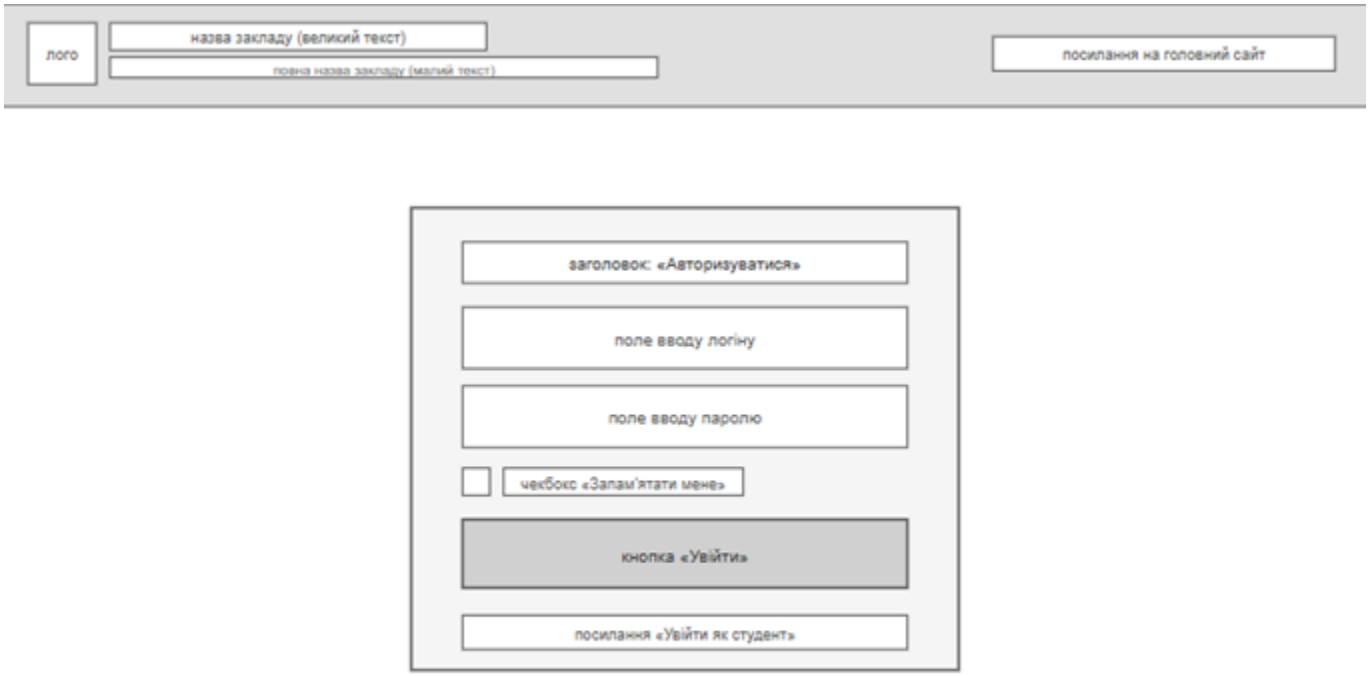


Рис. 3.2. Wireframe-макет сторінки авторизації (index.html)

3.1.2. Головна сторінка (dashboard.html). Це центральна сторінка системи, яка відображається як для студентів і викладачів, так і для адміністратора. Для студентів і викладачів сторінка показує лише затверджені розклади у вигляді мініатюр PDF-файлів з можливістю їх перегляду. При цьому студент бачить розклади лише своєї групи, відфільтровані за інститутом та курсом через відповідні випадаючі списки. Для адміністратора сторінка додатково містить три кнопки керування у правому верхньому куті — "Створити викладача", "Створити розклад" та "Створити групу" — які переводять на відповідні сторінки управління. Також адміністратору та

викладачам доступний розділ перегляду розкладів викладачів з фільтрацією за прізвищем. Макет головної сторінки зображено на рис. 3.3.

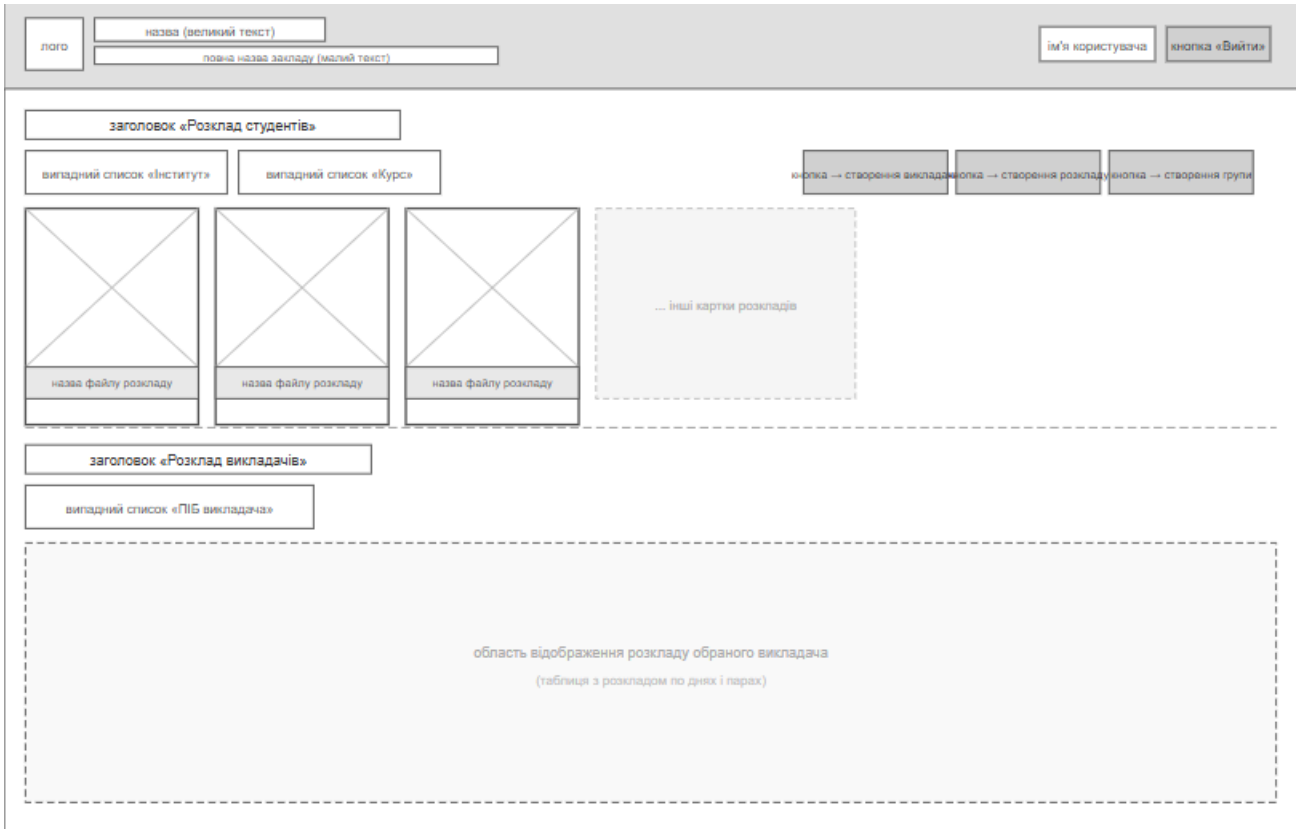


Рис. 3.3. Wireframe-макет головної сторінки (dashboard.html)

3.1.3. Сторінка управління викладачами (teachers_edit.html). Ця сторінка доступна лише адміністратору. Вона складається з двох частин. Верхня частина містить форму для створення нового викладача — поля для введення прізвища, імені та по батькові, предмету та кількості годин на тиждень, інтерактивну таблицю для вибору зручних пар за днями тижня, а також поле для вказання аудиторії. Нижня частина сторінки відображає таблицю з переліком усіх вже створених викладачів, де для кожного є кнопки редагування та видалення. Макет сторінки управління викладачами зображено на рис. 3.4.

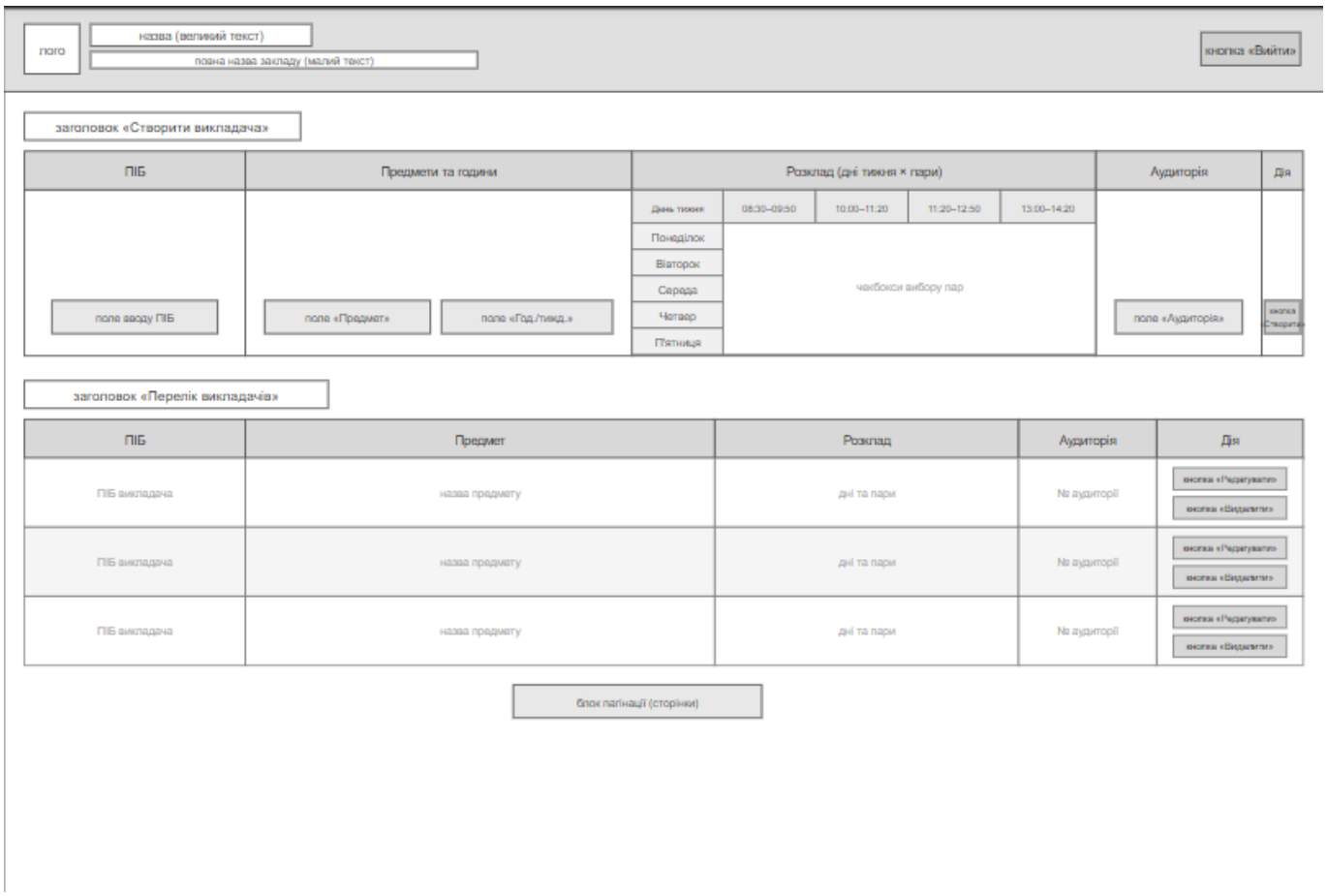


Рис. 3.4. Wireframe-макет сторінки управління викладачами (teachers_edit.html)

3.1.4. Сторінка управління групами (students_edit.html). Також доступна лише адміністратору. Аналогічно до попередньої сторінки, вона ділиться на дві частини. Верхня частина містить форму для створення нової групи студентів — випадаючий список для вибору інституту, поле для вибору номера курсу та поле для введення назви групи. Нижня частина відображає таблицю з переліком усіх створених груп із зазначенням інституту, курсу та назви групи, а також кнопки редагування та видалення для кожного запису. Макет сторінки управління групами зображено на рис. 3.5.

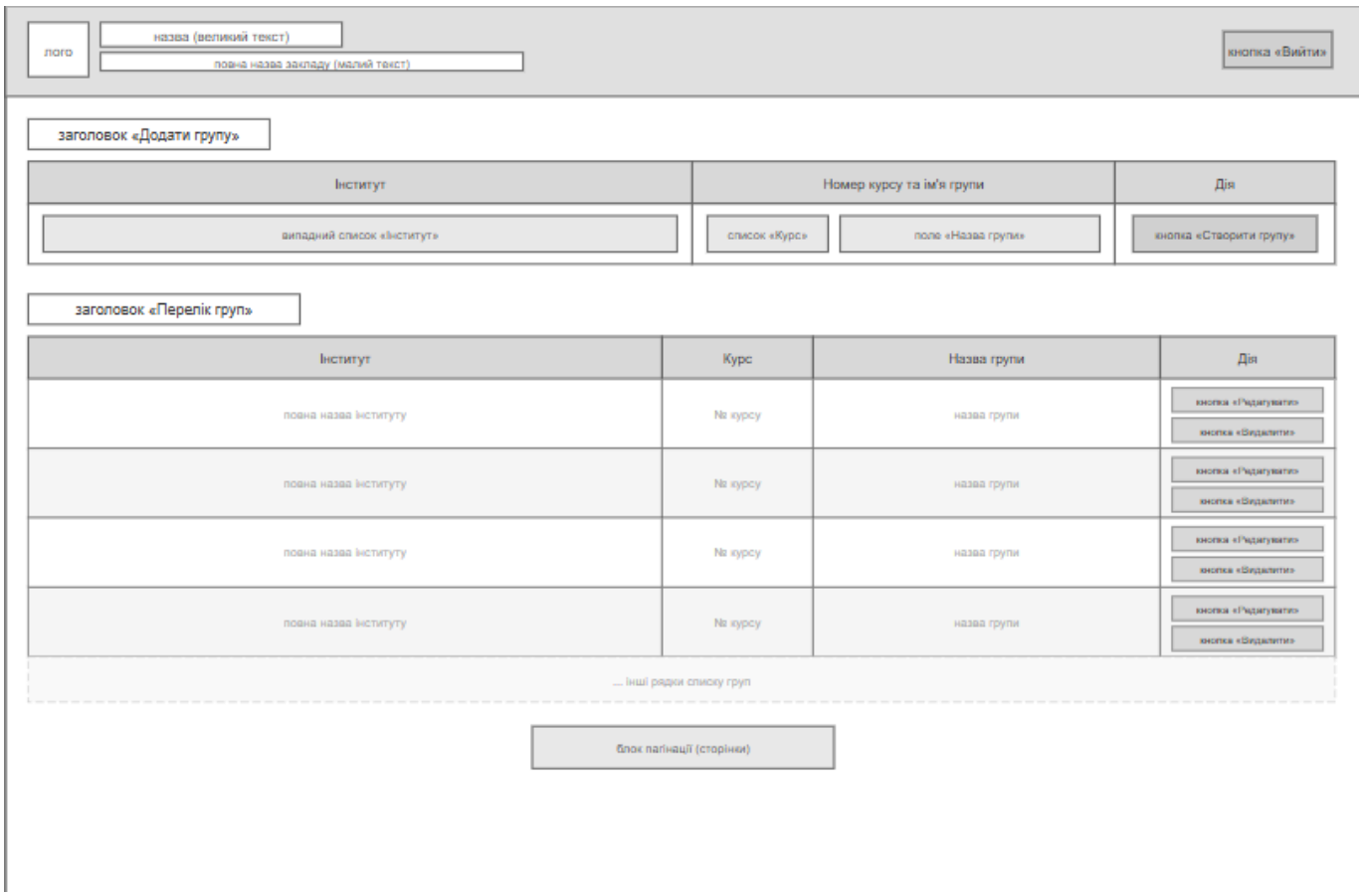


Рис. 3.5. Wireframe-макет сторінки управління групами (students_edit.html)

3.1.5. Сторінка генерації розкладу (generator.html). Це фінальна сторінка робочого процесу адміністратора. На ній куратор формує розклад для конкретної групи — обирає групу та предмети зі списків, вводить додаткові дані: прізвище декана, заступника начальника управління, дати початку та кінця семестру. Після заповнення всіх полів доступні дві кнопки — "Переглянути PDF" для попереднього перегляду сформованого розкладу та "Завантажити PDF" для його затвердження і збереження в системі. Після завантаження розклад з'являється на головній сторінці у вигляді мініатюри і стає доступним для перегляду студентам і викладачам. Макет сторінки генератора розкладу зображено на рис. 3.6.

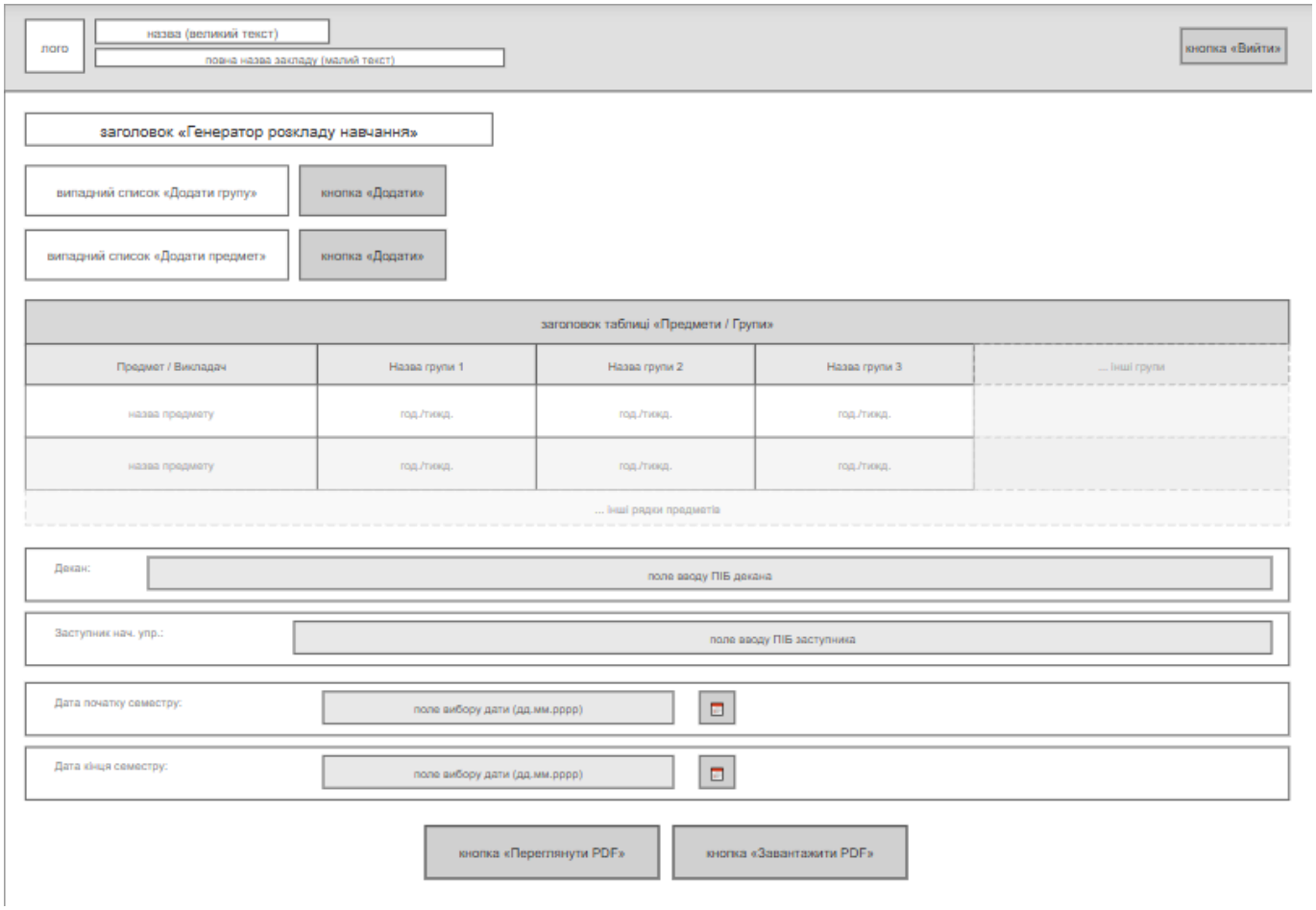


Рис. 3.6. Wireframe-макет сторінки генератора розкладу (generator.html)

Таким чином, спроєктована структура чітко розділяє функції між різними типами користувачів і забезпечує логічний порядок роботи з системою — від авторизації до перегляду готового розкладу. Кожна сторінка виконує одну чітко визначену задачу, що робить інтерфейс зрозумілим і зручним у використанні.

## 3.2 Розробка HTML-структури вебсторінок

Розробка HTML-структури є першим практичним етапом створення клієнтської частини вебсистеми. На цьому етапі для кожної сторінки визначається набір HTML-елементів, їх ієрархія та семантичне значення. Правильно побудована HTML-структура є основою для подальшого стилізування засобами CSS та додавання інтерактивності через JavaScript.

Усі сторінки системи, крім сторінки авторизації, побудовані на основі спільного базового шаблону — файлу base.html. Використання базового шаблону є стандартною практикою у веброзробці, яка дозволяє уникнути дублювання коду.

Шаблон містить спільні для всіх сторінок елементи: підключення CSS-файлів та іконок, структуру шапки сторінки та блок для підстановки унікального вмісту кожної сторінки. Механізм шаблонізації реалізовано засобами Jinja2 — шаблонного рушія Flask, який дозволяє використовувати директиви {% extends %} та {% block %} для успадкування та заповнення шаблону [7] [21].

Шапка сторінки, визначена в base.html, має єдину структуру для всього застосунку. Вона містить тег <header> з логотипом університету у вигляді зображення <img>, назвою закладу у двох рядках та блоком з іменем поточного користувача і посиланням для виходу з системи. Така структура забезпечує впізнаваність інтерфейсу та зручну навігацію на всіх сторінках.

3.2.1. HTML-структура сторінки авторизації (index.html). Сторінка авторизації є єдиною сторінкою системи, яка не успадковує базовий шаблон, оскільки має унікальний дизайн без стандартної шапки адміністративної панелі. Вона побудована як самостійний HTML-документ з власним тегом <head>, де підключаються необхідні метатеги, CSS-файл стилів, бібліотека іконок Boxicons та фавікон із логотипом університету [11].

Структура тіла сторінки (<body>) поділена на три логічні блоки. Перший — шапка (<header>), яка містить логотип та назву МАУП зліва і посилання на офіційний сайт університету справа. Другий — основний блок (<main>), що містить форму авторизації з атрибутами action="/login" та method="post". Форма включає заголовок <h1>, два поля введення (<input type="text"> для логіну та <input type="password"> для паролю), кожне з яких обгорнуте в  разом з іконкою від бібліотеки Boxicons. Також форма містить чекбокс "Запам'ятати мене" та кнопку відправки <button type="submit">. Третій блок розміщений поза формою і містить посилання "Увійти як студент", яке веде на окремий маршрут швидкого входу без введення облікових даних. HTML-структуру сторінки авторизації без застосування стилів зображено на рис. 3.7.

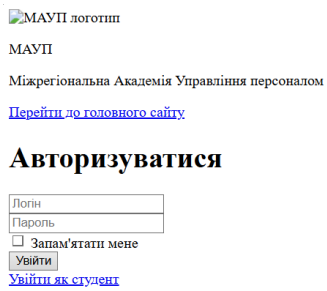


Рис. 3.7. HTML-структура сторінки авторизації без стилів (index.html)

3.2.2. HTML-структура головної сторінки (dashboard.html). Головна сторінка успадковує базовий шаблон і весь її унікальний вміст розміщений у відповідному блоці шаблону. Структура сторінки побудована з використанням умовної логіки — залежно від ролі користувача відображаються різні елементи інтерфейсу.

Для адміністратора у верхній частині сторінки відображається блок  з трьома посиланнями-кнопками для переходу до створення викладача, розкладу та групи. Далі розміщені два кастомні випадаючі списки для фільтрації за інститутом та курсом — вони реалізовані не через стандартний тег <select>, а через власну структуру з  та <ul>, що дозволяє гнучко стилізувати їх зовнішній вигляд. Список інститутів містить дев'ять пунктів із скороченими назвами та повними назвами у тегу <abbr> для відображення підказки при наведенні.

Нижче розміщений контейнер , який динамічно заповнюється картками PDF-файлів розкладів. Кожна картка є посиланням <a> з вбудованим фреймом <iframe> для попереднього перегляду PDF та підписом з назвою файлу. Аналогічна структура використовується для відображення розкладів викладачів у другому контейнері. Для наочної демонстрації структури сторінки до

неї було додано тестові дані — кілька прикладів розкладів студентів та викладачів. HTML-структуру головної сторінки без застосування стилів зображено на рис. 3.8.


Рис. 3.8. HTML-структура головної сторінки без стилів (dashboard.html)

3.2.3. HTML-структура сторінки управління викладачами (teachers_edit.html). Сторінка складається з двох основних блоків, обгорнутих у спільний контейнер . Перший блок — форма створення викладача, реалізована через тег <form> з атрибутами action="/add_teacher" та method="POST". Форма побудована у вигляді таблиці <table> з п'ятьма стовпцями: ПІБ, предмети та години, розклад, аудиторія та дія.

Найбільш складним елементом форми є вкладена таблиця розкладу. Вона містить рядки для кожного дня тижня (понеділок–п'ятниця) та чотири стовпці для кожної пари (08:30–09:50, 10:00–11:20, 11:20–12:50, 13:00–14:20). На перетині кожного дня та пари розміщений чекбокс <input type="checkbox"> зі значенням у форматі "день_номер" (наприклад, monday_0, tuesday_2), що дозволяє точно визначити зручні для викладача часові слоти.

Другий блок сторінки — таблиця переліку викладачів, яка динамічно заповнюється даними з бази. Для наочної демонстрації структури до таблиці було додано тестові записи з прикладами реальних викладачів та їх даних. Кожен рядок таблиці містить ПІБ, предмет, розклад та аудиторію викладача, а також кнопки

"Редагувати" і "Видалити". HTML-структуру сторінки управління викладачами без застосування стилів зображено на рис. 3.9.

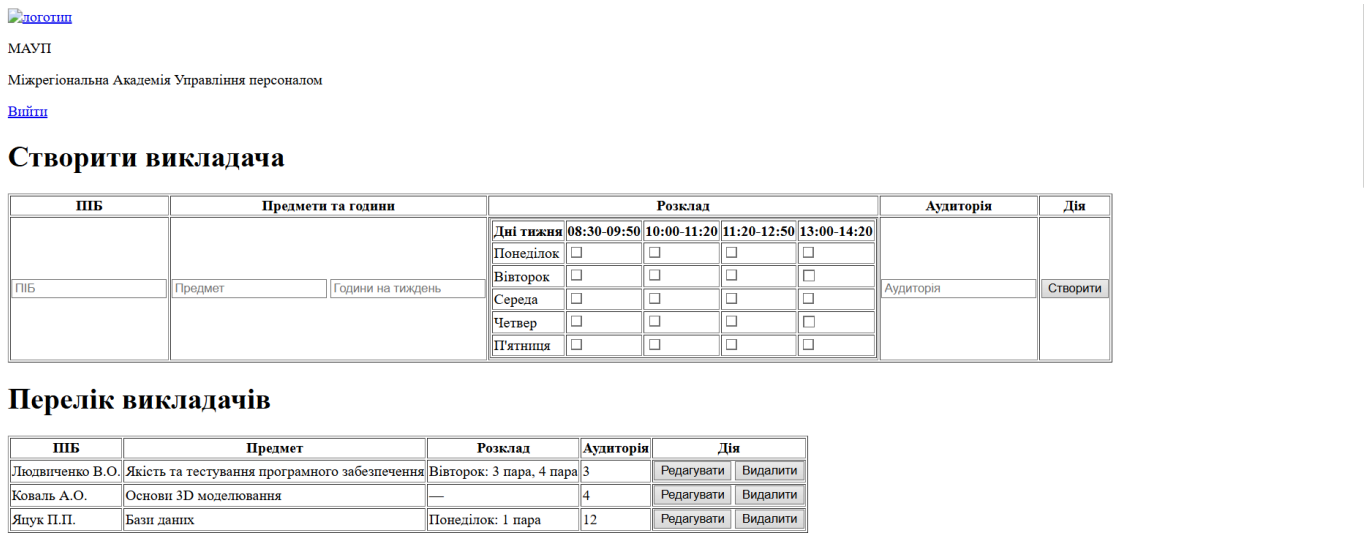


Рис. 3.9. HTML-структура сторінки управління викладачами без стилів (teachers_edit.html)

3.2.4. HTML-структура сторінки управління групами (students_edit.html). Структура сторінки аналогічна до попередньої і також поділена на дві частини. Форма додавання групи містить таблицю з трьома стовпцями. У першому стовпці розміщений стандартний тег <select> зі списком усіх восьми інститутів МАУП у вигляді варіантів <option>. Кожен варіант має атрибут value зі скороченою та повною назвою інституту, що дозволяє передавати на сервер повну інформацію для збереження в базі даних. У другому стовпці розміщений рядок з двох елементів — випадаючий список курсів від 1 до 5 та текстове поле для введення назви групи. Третій стовпець містить кнопку створення групи.

Таблиця переліку груп динамічно заповнюється даними з бази. Для наочності до неї було додано тестові записи з прикладами груп різних інститутів, що дозволяє оцінити як структура таблиці виглядатиме з реальними даними. Для кожного рядка виводяться назва інституту, номер курсу та назва групи, а також кнопки редагування та видалення. HTML-структуру сторінки управління групами без застосування стилів зображено на рис. 3.10.

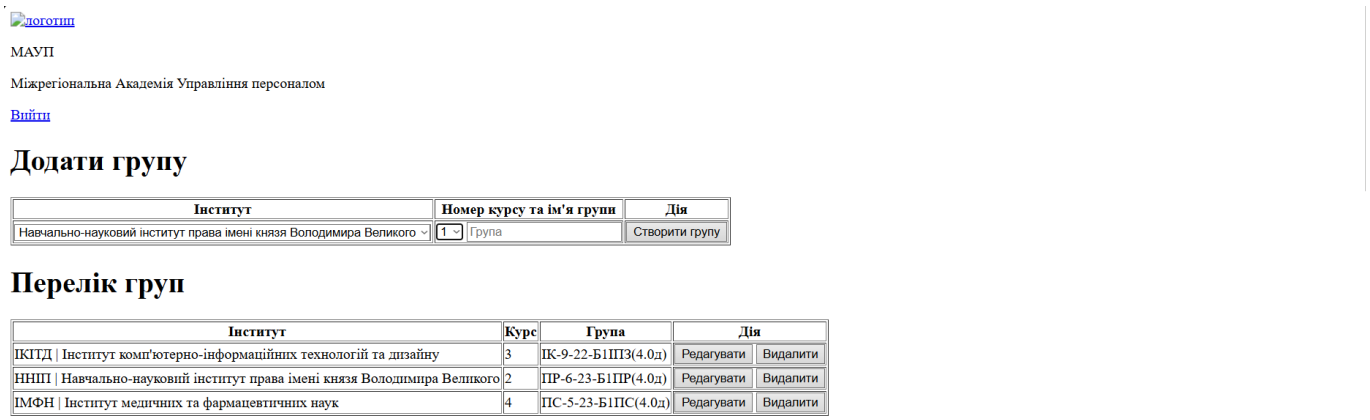


Рис. 3.10. HTML-структура сторінки управління групами без стилів (students_edit.html)

3.2.5. HTML-структура сторінки генератора розкладу (generator.html). Ця сторінка має найбільш складну HTML-структуру серед усіх сторінок системи. На початку підключається зовнішня бібліотека Choices.js через CDN — вона розширює стандартний тег <select> функцією пошуку по списку, що є важливим при великій кількості груп та предметів [22].

Структура сторінки складається з кількох послідовних блоків. Перший — контейнер вибору груп із тегом <select>, який динамічно заповнюється даними з бази, де кожна група передає атрибути data-name, data-institute та data-course для подальшої обробки через JavaScript. Аналогічний блок реалізований для вибору предметів та викладачів. Далі розміщена таблиця відповідності предметів та груп, заголовок та рядки якої заповнюються динамічно через JavaScript після вибору користувача. Для наочної демонстрації структури до таблиці було додано тестові дані — кілька прикладів груп та предметів з кількістю годин на тиждень.

Нижня частина сторінки містить форму з додатковими полями: текстові поля для введення ПІБ декана та заступника начальника управління, а також поля вибору дати <input type="date"> для початку та кінця семестру. Завершують сторінку дві

кнопки — "Переглянути PDF" та "Завантажити PDF". HTML-структуру сторінки генератора розкладу без застосування стилів зображено на рис. 3.11.

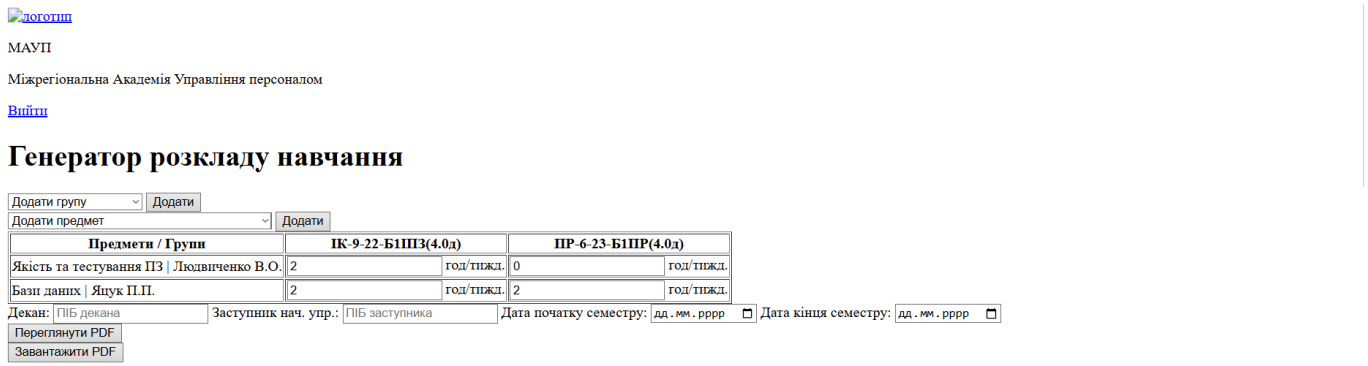


Рис. 3.11. HTML-структура сторінки генератора розкладу без стилів (generator.html)

Таким чином, HTML-структура всіх сторінок системи побудована з урахуванням їх функціонального призначення. Використання базового шаблону забезпечує єдність шапки та загального стилю, семантичні теги покращують читабельність коду, а механізм шаблонізації Jinja2 дозволяє динамічно формувати вміст сторінок на основі даних із серверної частини.

## 3.3 Розробка стилів та адаптивного дизайну засобами CSS

Після побудови HTML-структури сторінок наступним етапом розробки клієнтської частини є створення стилів за допомогою CSS. Стилі визначають зовнішній вигляд усіх елементів інтерфейсу — кольори, шрифти, розміри, відступи та розташування елементів на сторінці. У розробленій системі стилі організовані у вигляді окремих CSS-файлів для кожної сторінки, а також спільного базового файлу base.css, який містить стилі для елементів, що є спільними для всіх сторінок.

3.3.1. Базові стилі (base.css). Файл base.css підключається до кожної сторінки через базовий шаблон і задає глобальні стилі для всього застосунку. На початку файлу застосовується універсальний селектор *, який скидає стандартні відступи

браузера та встановлює єдиний шрифт для всіх елементів. Це є стандартною практикою у веброзробці, яка забезпечує однакове відображення сторінок у різних браузерах.

Основним кольором інтерфейсу системи обрано відтінок синього — #0064b4, який відповідає корпоративному стилю МАУП. Цей колір використовується як фон шапки, колір кнопок та заголовків таблиць на всіх сторінках системи. Шапка (<header>) стилізована з використанням Flexbox — властивість display: flex дозволяє розташувати логотип, назву закладу та блок користувача в один рядок з автоматичним вирівнюванням.

Для забезпечення коректного відображення на мобільних пристроях у файлі реалізований медіазапит @media (max-width: 768px). При ширині екрану менше 768 пікселів шапка перебудовується з горизонтального у вертикальний макет — елементи вирівнюються по центру, шрифт заголовків зменшується, а блок з іменем користувача переміщується під назву закладу. Це забезпечує зручне відображення системи на смартфонах та планшетах.

3.3.2. Стилі сторінки авторизації (index.css). Сторінка авторизації має унікальний дизайн, що відрізняється від інших сторінок системи. Фон сторінки реалізований за допомогою CSS-градієнту — властивість background: linear-gradient(180deg, #0064b4 15%, #4eafff 80%) створює плавний перехід від темно-синього кольору вгорі до світло-блакитного внизу, що надає сторінці сучасного та привабливого вигляду.

Форма авторизації розміщена по центру сторінки завдяки Flexbox на елементі <main> з властивостями justify-content: center та align-items: center. Сама форма має напівпрозорий фон, реалізований через background: rgba(255, 255, 255, 0.1) та ефект розмиття фону backdrop-filter: blur(20px), що створює характерний ефект скла. Поля введення стилізовані з прозорим фоном та білою рамкою, що гармонійно вписується в загальний синій фон сторінки.

Кнопка "Увійти" має білий фон з синім текстом і при наведенні змінює кольори на протилежні завдяки CSS-переходу transition: 0.3s ease. Також реалізований ефект легкого збільшення кнопки при наведенні через transform:

scale(1.03), що покращує інтерактивність інтерфейсу. Медіазапит для мобільних пристроїв перебудовує шапку у вертикальний вигляд та зменшує розміри шрифтів і відступів форми. Фінальний вигляд сторінки авторизації зображено на рис. 3.12.

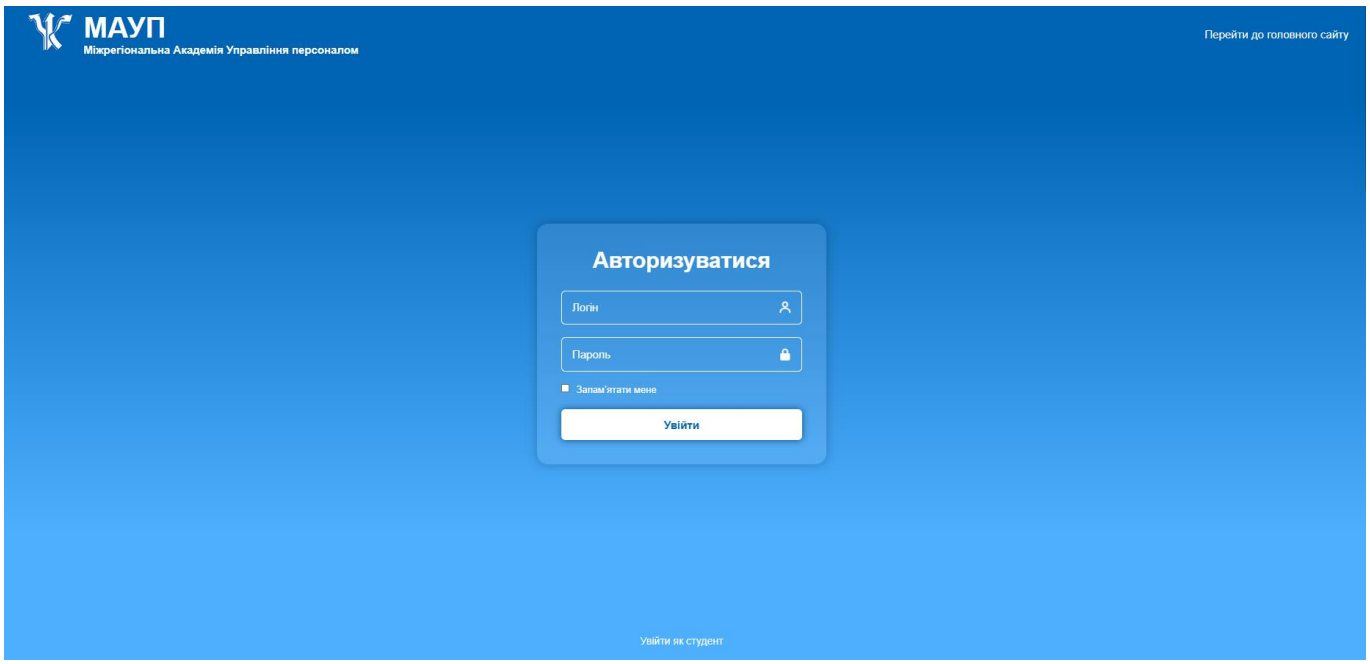


Рис. 3.12. Зовнішній вигляд сторінки авторизації (index.html)

3.3.3. Стилі головної сторінки (dashboard.css). Головна сторінка є найбільш насиченою елементами, тому її CSS-файл є одним із найбільших у проєкті. Заголовки розділів "Розклад студентів" та "Розклад викладачів" стилізовані як великий білий текст розміром 30 пікселів з жирним накресленням, що чітко виділяє їх на синьому фоні сторінки.

Кнопки адміністратора ("Створити викладача", "Створити розклад", "Створити групу") розміщені у фіксованому блоці в правому куті екрану через властивість position: fixed з відступами right: 20px та top: 200px. Це означає що кнопки залишаються видимими під час прокрутки сторінки. Самі кнопки мають заокруглену форму (border-radius: 20px), синій фон та плавну анімацію при наведенні.

Кастомні випадаючі списки для фільтрації реалізовані повністю через CSS без використання стандартного вигляду браузера. Контейнер списку має клас select зі стилями синього фону та заокругленими кутами. При кліку через JavaScript додається клас menu-open, який змінює display: none на display: block і робить

список видимим. Стрілка-індикатор реалізована через CSS-трикутник з нульовою шириною та висотою і властивостями border, а при відкритті списку повертається на 180 градусів через transform: rotate(180deg).

Контейнери з PDF-картками використовують display: flex з flex-wrap: nowrap та overflow-x: auto, що дозволяє картки розміщувати в один горизонтальний рядок з можливістю горизонтальної прокрутки при великій їх кількості. Для мобільних пристроїв усі фіксовані елементи переводяться в звичайний потік документа, кнопки розтягуються на всю ширину екрану, а випадаючі списки перебудовуються у вертикальний стек. Фінальний вигляд головної сторінки зображено на рис. 3.13.


Рис. 3.13. Зовнішній вигляд головної сторінки (dashboard.html)

3.3.4. Стилі сторінок управління (teachers_edit.css, students_edit.css). Обидва файли стилів мають схожу структуру, оскільки сторінки управління викладачами та групами побудовані за однаковим принципом. Основним стилістичним елементом є таблиці з заокругленими кутами, реалізовані через комбінацію border-radius: 16px на контейнері та окремі правила для першого і останнього рядків таблиці — th:first-child, th:last-child, tr:last-child td:first-child та tr:last-child td:last-child. Оскільки стандартний CSS не дозволяє напряму заокруглювати кути таблиці при border-collapse: collapse, використано border-collapse: separate з border-spacing: 0, що дає змогу застосувати border-radius до окремих комірок.

Заголовки таблиць (<th>) мають синій фон #0064b4, а рядки даних — прозорий фон, що успадковує синій колір фону сторінки. Кнопки "Редагувати" та

"Видалити" стилізовані по-різному — кнопка редагування має синій фон, а кнопка видалення — червоний #d9534f, що є стандартною практикою для позначення небезпечних дій в інтерфейсах. Обидві кнопки мають заокруглену форму та анімацію збільшення при наведенні.

Особливістю файлу teachers_edit.css є стилі для вкладеної таблиці розкладу. Вона обгорнута в контейнер .schedule-container з властивістю overflow: hidden та синім фоном #6FBBF6, що забезпечує коректне відображення заокруглених кутів. Сама таблиця розкладу використовує border-collapse: collapse для суцільних меж між комірками. Фінальний вигляд сторінки управління викладачами зображено на рис. 3.14.

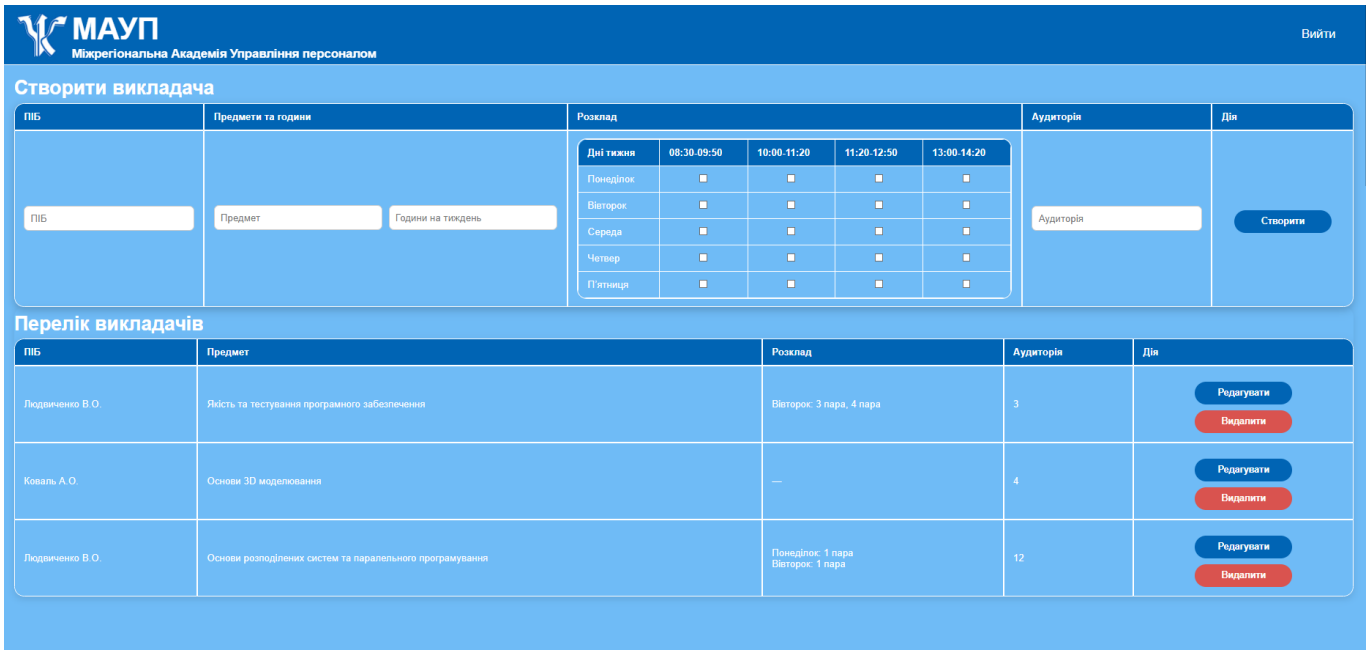


Рис. 3.14. Зовнішній вигляд сторінки управління викладачами (teachers_edit.html)

Фінальний вигляд сторінки управління групами зображено на рис. 3.15.

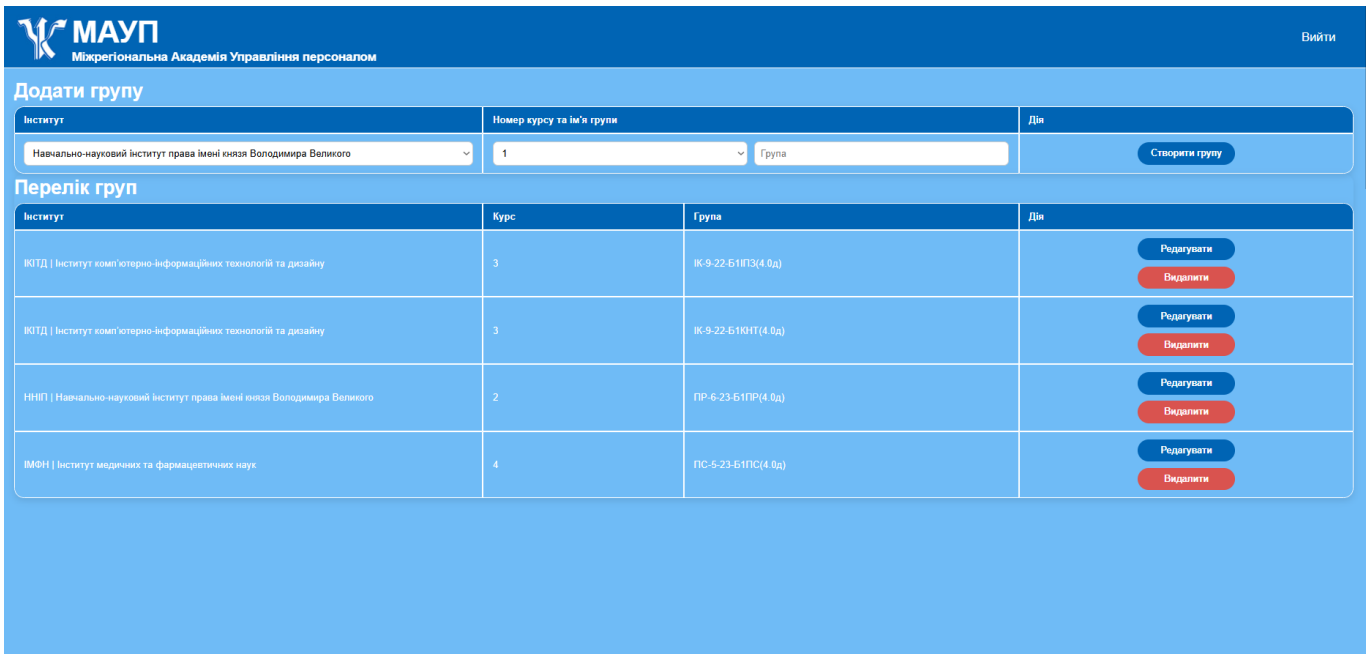


Рис. 3.15. Зовнішній вигляд сторінки управління групами (students_edit.html)

3.3.5. Стилі сторінки генератора (generator.css). CSS-файл генератора розкладу містить стилі для найбільш інтерактивної сторінки системи. Особливу увагу приділено стилізації бібліотеки Choices.js — оскільки ця бібліотека генерує власну HTML-розмітку, для її стилізації використовуються специфічні селектори класів бібліотеки: .choices__inner, .choices__list--single, .choices__list--dropdown та інші. Це дозволяє повністю змінити стандартний вигляд бібліотеки відповідно до загального стилю системи — білий фон, заокруглені кути та синя рамка при фокусі.

Таблиця відповідності предметів та груп має горизонтальну прокрутку через overflow-x: auto на контейнері, оскільки кількість стовпців залежить від кількості обраних груп і може бути довільною. Поля введення дат та тексту стилізовані з плавним переходом кольору рамки при фокусі та ефектом синього свічення box-shadow: 0 0 5px rgba(0, 123, 255, 0.5). Фінальний вигляд сторінки генератора зображено на рис. 3.16.

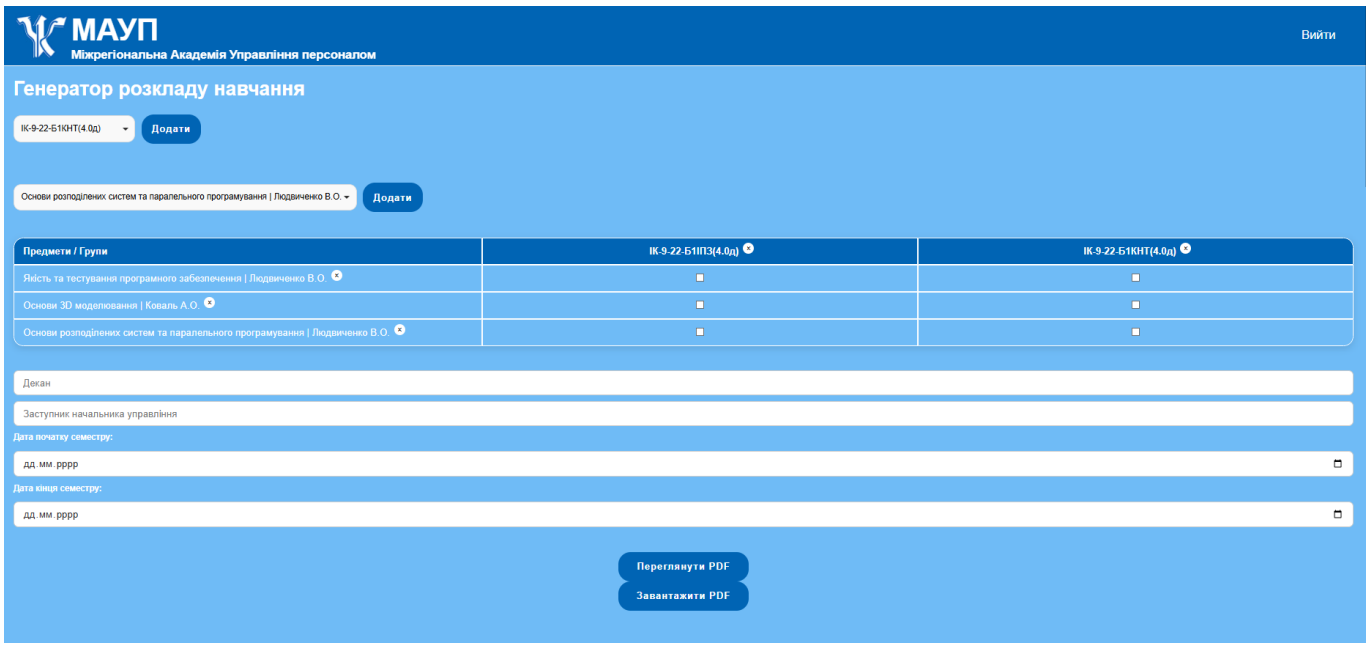


Рис. 3.16. Зовнішній вигляд сторінки генератора розкладу (generator.html)

Таким чином, стилі клієнтської частини системи забезпечують єдиний корпоративний дизайн у синіх тонах на всіх сторінках, зручний і сучасний інтерфейс з анімаціями та переходами, а також адаптивність для коректного відображення на мобільних пристроях.

## 3.4 Розробка логіки інтерфейсу та взаємодії з сервером засобами JavaScript

JavaScript-файли клієнтської частини відповідають за всю інтерактивну логіку системи — реакцію на дії користувача, динамічну зміну вмісту сторінок та передачу даних на серверну частину. Кожна сторінка має власний JS-файл, який підключається наприкінці HTML-документа. Розглянемо логіку роботи кожного скрипту детально — з поясненням ключових фрагментів коду.

3.4.1. Логіка головної сторінки (dashboard.js). Скрипт головної сторінки реалізує функціонал кастомних випадаючих списків та фільтрацію карток розкладів. Після завантаження сторінки через подію DOMContentLoaded скрипт знаходить усі елементи з класом .dropdown і для кожного з них призначає обробник кліку. При кліку на список через метод classList.toggle додається або прибирається клас menu-open, що через CSS робить список видимим або прихованим. Стрілка-індикатор одночасно повертається на 180 градусів через клас caret-rotate.

Ініціалізація кастомних списків реалізована таким чином:

```
document.addEventListener("DOMContentLoaded", () => {
  document.querySelectorAll(".dropdown").forEach(dropdown => {
    const select = dropdown.querySelector(".select");
    const menu = dropdown.querySelector(".menu");
    const caret = dropdown.querySelector(".caret");

    select.addEventListener("click", () => {
      menu.classList.toggle("menu-open");
      caret.classList.toggle("caret-rotate");
    });
  });
});
```

При виборі пункту зі списку викликаються дві функції фільтрації — filterStudents() та filterTeachers(). Функція filterStudents() зчитує поточні вибрані значення інституту та курсу і проходить по всіх картках розкладів у контейнері #docs-container. Кожна картка має атрибути data-institute та data-course, які порівнюються з обраними значеннями. Картки що не відповідають фільтру приховуються через style.display = "none", а відповідні — відображаються:

```
function filterStudents() {
  const instituteValue =
    document.querySelector(".dropdown-container .dropdown:nth-child(1)
      .menu li.active")?.dataset.value || "";
  const courseValue =
    document.querySelector(".dropdown-container .dropdown:nth-child(2)
      .menu li.active")?.dataset.value || "";

  const docs = document.querySelectorAll("#docs-container .doc");
  let visible = 0;

  docs.forEach(doc => {
    let docInstitute = doc.dataset.institute || "";
    let docCourse = doc.dataset.course || "";
    docInstitute = docInstitute.split("|")[0].trim();

    if (
      (instituteValue === "" || docInstitute.includes(instituteValue)) &&
      (courseValue === "" || docCourse === courseValue)
    ) {
      doc.style.display = "block";
      visible++;
```

```
        } else {
            doc.style.display = "none";
        }
    });
    showMessage("#docs-container", visible);
}
```

Якщо після фільтрації не залишилось жодної картки, функція showMessage() динамічно створює і додає у контейнер повідомлення "Документи за вказаними критеріями не знайдено." через document.createElement("p"). Аналогічно працює функція filterTeachers() для розкладів викладачів. Схему роботи скрипту головної сторінки зображено на рис. 3.17.

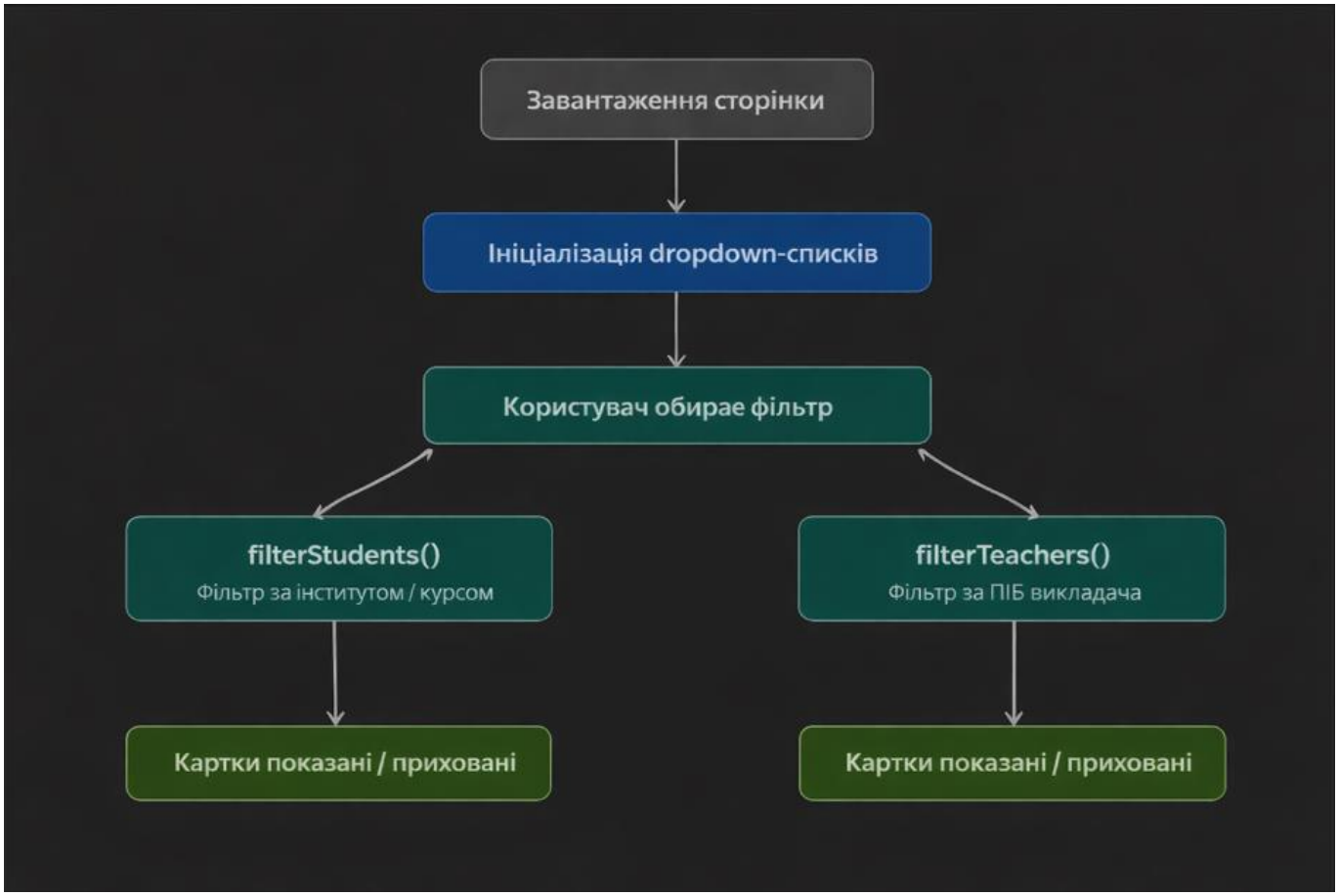


Рис. 3.17. Схема роботи dashboard.js

3.4.2. Логіка сторінки управління викладачами (teachers_edit.js). Скрипт цієї сторінки реалізує два режими роботи форми — створення нового викладача та редагування існуючого. Для керування режимами використовуються дві змінні — editMode (булеве значення) та editingTeacherId (ідентифікатор викладача що редагується).

При відправці форми через подію submit скрипт перехоплює стандартну поведінку браузера через event.preventDefault() і самостійно збирає дані. Дані про розклад збираються з усіх відмічених чекбоксів — значення кожного чекбоксу має формат "день_номер", який розбивається методом split('_') на ключ та індекс. На основі цих даних формується об'єкт розкладу:

```
const schedule = {
   monday: [], tuesday: [], wednesday: [], thursday: [], friday: []
};

document.querySelectorAll('input[type="checkbox"]:checked').forEach(cb => {
   const [day, period] = cb.value.split('_');
   schedule[day].push(parseInt(period));
});
```

Аналогічно збираються предмети та години — скрипт проходить по всіх парах полів subject[] та hoursperweek[] і формує словник subjectsDict де ключем є назва предмету, а значенням — кількість годин на тиждень. Усі зібрані дані об'єднуються в один об'єкт teacherData та серіалізуються у JSON через JSON.stringify():

```
const subjectsDict = {};
subjects.forEach((input, index) => {
   const subject = input.value.trim();
   const hours = hoursPerWeek[index].value.trim();
   if (subject && hours) {
      subjectsDict[subject] = parseInt(hours);
   }
});

const teacherData = { fullname, subjects: subjectsDict,
   roomassignment, schedule };
hiddenInput.value = JSON.stringify(teacherData);
```

Залежно від режиму форма надсилається на різні маршрути — /add_teacher для створення або /update_teacher/{id} для оновлення. При натисканні кнопки "Редагувати" скрипт зчитує дані викладача з data-атрибутів кнопки та заповнює форму. Чекбокси скидаються і встановлюються заново відповідно до збереженого розкладу через перебір ключів об'єкту schedule:

```
Object.keys(schedule).forEach(day => {
    schedule[day].forEach(period => {
        const checkbox = document.querySelector(
            `input[value="${day}_${period}"]`
        );
        if (checkbox) checkbox.checked = true;
    });
});
```

Окремою функцією реалізовано форматування відображення розкладу у таблиці викладачів. Скрипт зчитує збережений JSON з комірки таблиці, конвертує числові ключі у назви днів та формує зрозумілий текстовий рядок типу "Понеділок: 1 пара, Вівторок: 2 пара". Схему роботи скрипту зображено на рис. 3.18.

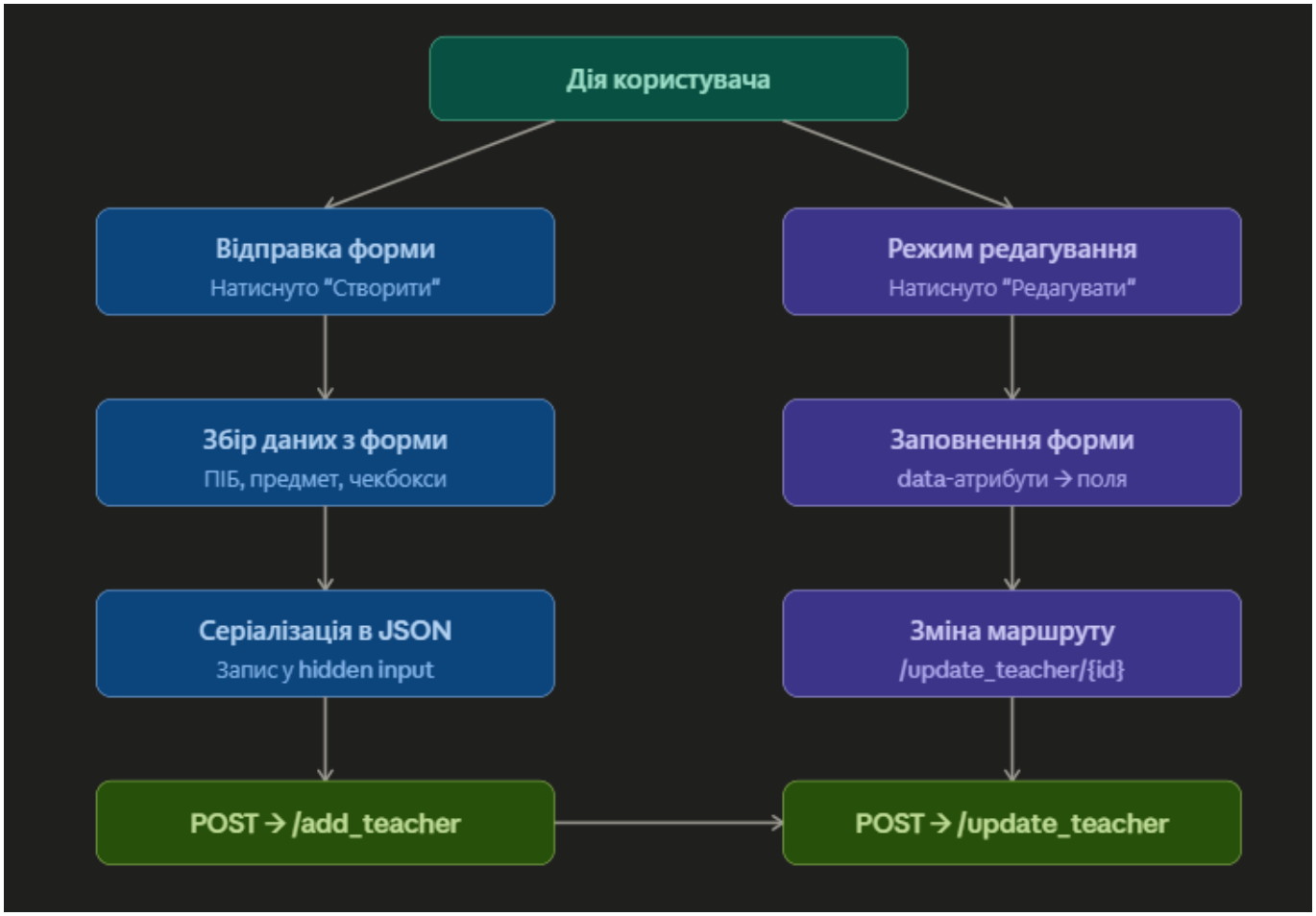


Рис. 3.18. Схема роботи teachers_edit.js

3.4.3. Логіка сторінки управління групами (students_edit.js). Скрипт сторінки груп працює за схожим принципом до попереднього. При відправці форми перехоплюється стандартна поведінка, збираються значення інституту, курсу та назви групи. Дані серіалізуються у JSON і записуються у приховане поле:

```
document.getElementById('student-form').addEventListener('submit',
    function (event) {
    event.preventDefault();

    const studentData = { course: course, group: group };
    document.getElementById('student-data').value =
        JSON.stringify(studentData);

    this.submit();
});
```

При натисканні кнопки "Редагувати" скрипт зчитує data-атрибути кнопки — data-id, data-institute, data-course, data-group — та заповнює відповідні поля форми. Якщо приховане поле з ідентифікатором студента ще не існує, скрипт динамічно створює його через document.createElement('input') і додає до форми. Кнопка відправки при цьому змінює текст з "Створити групу" на "Оновити групу".

3.4.4. Логіка сторінки генератора розкладу (generator.js). Скрипт генератора є найскладнішим у системі. Він реалізує динамічне формування таблиці розкладу та підготовку даних для надсилання на сервер.

При натисканні кнопки "Додати" для груп скрипт зчитує обране значення зі списку Choices.js через groupChoices.getValue(), перевіряє унікальність через масив selectedGroups і якщо групу ще не додано — динамічно додає новий стовпець до таблиці. Заголовок стовпця формується з назви групи, а у кожному рядку таблиці для цього стовпця створюється комірка з числовим полем для введення кількості годин:

```
document.getElementById('add-group-btn').addEventListener('click', () => {
    const selected = window.groupChoices.getValue();
    if (!selected || !selected.value) return;

    const groupId = selected.value;
    if (selectedGroups.includes(groupId)) return;

    selectedGroups.push(groupId);

    const th = document.createElement('th');
    th.dataset.group = groupId;
    th.innerHTML = `${selected.customProperties.name}
        <button class="remove-group">×</button>`;
    headerRow.appendChild(th);
});
```

Перед відправкою форми скрипт збирає всі дані — обрані групи, предмети, кількість годин, а також значення текстових полів та дат. Всі ці дані записуються у відповідні приховані поля обох форм одночасно — і для перегляду, і для завантаження PDF. Це необхідно, щоб обидві кнопки мали актуальні дані в момент натискання.

Додатково реалізована валідація перед надсиланням — скрипт перевіряє три умови: чи додана хоча б одна група, чоча б один предмет та чи заповнені обов'язкові поля. При порушенні будь-якої умови у відповідному блоці  відображається повідомлення про помилку, а форма не надсилається. Схему роботи скрипту генератора зображено на рис. 3.19.

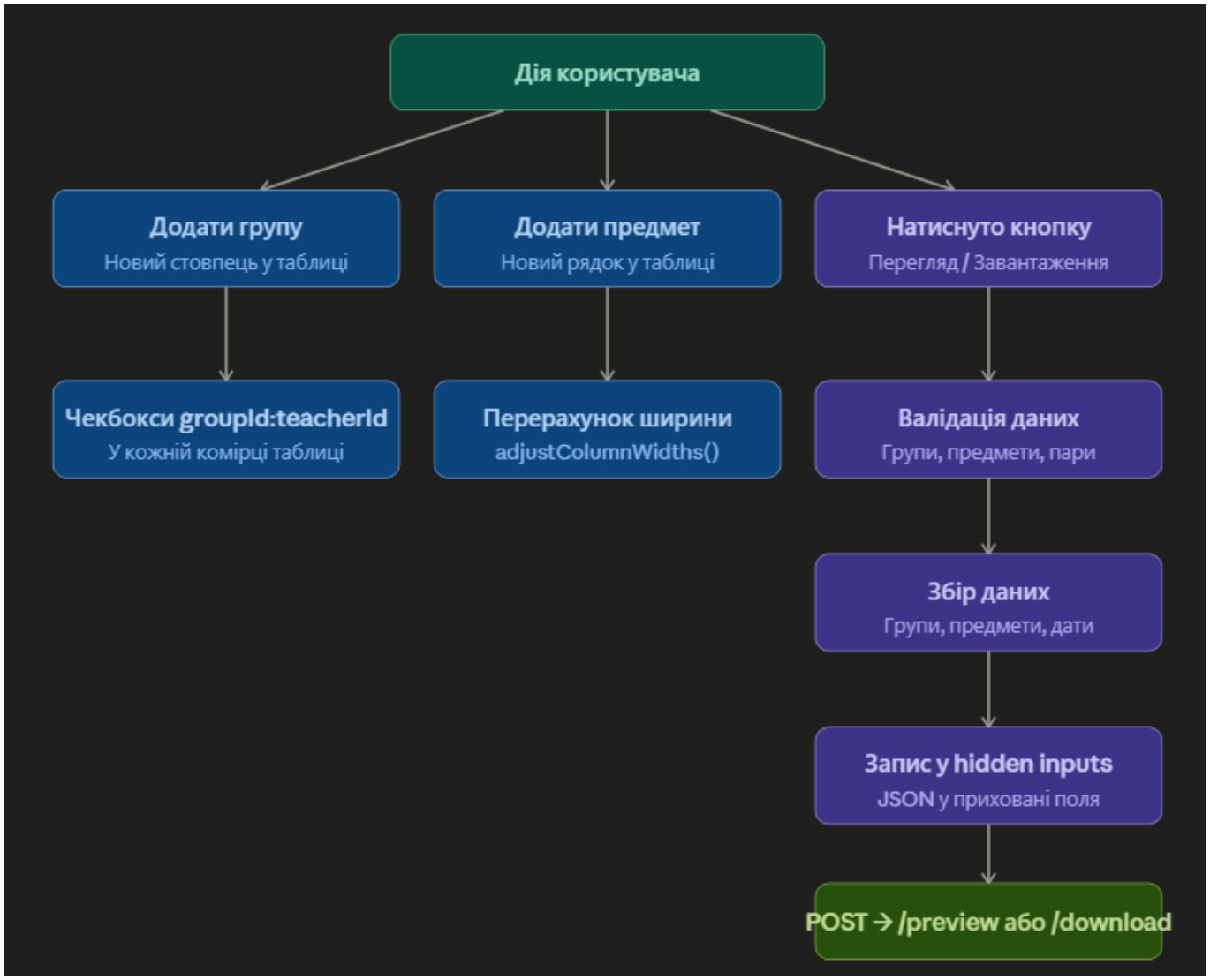


Рис. 3.19. Схема роботи generator.js

Таким чином, JavaScript-логіка клієнтської частини системи забезпечує повноцінну інтерактивність усіх сторінок. Кожен скрипт реалізує чітко визначений функціонал своєї сторінки — фільтрацію карток розкладу, динамічне формування таблиць або підготовку і серіалізацію даних перед надсиланням на сервер. Використання event.preventDefault() та ручної серіалізації даних у JSON дозволяє передавати складні структури даних через стандартний механізм HTML-форм, що добре інтегрується з серверною частиною на Flask.

## 3.5 Тестування клієнтської частини

Важливим етапом розробки будь-якої вебсистеми є перевірка коректності роботи її інтерфейсу. Тестування клієнтської частини дозволяє виявити помилки у верстці, логіці взаємодії з користувачем, валідації даних та адаптивності дизайну до того, як система буде передана кінцевим користувачам. У рамках даної роботи було проведено функціональне тестування всіх п’яти сторінок системи, тестування адаптивності на різних пристроях, а також перевірку валідації введених даних на сторінках генерації розкладу та керування даними.

### 3.5.1 Функціональне тестування.

Функціональне тестування проводилось за методом «чорної скриньки» — перевірялась відповідність поведінки інтерфейсу очікуваним сценаріям без аналізу внутрішнього коду. Для кожної сторінки було складено перелік тест-кейсів, що охоплюють основний функціонал.

Тестування сторінки авторизації (index.html). На цій сторінці перевірялися:

1. відображення форми входу з полями «Логін», «Пароль», чекбоксом «Запам’ятати мене» та кнопкою «Увійти»;
2. наявність посилання «Увійти як студент»;
3. реакція на спробу входу з порожніми полями (система не надсилає запит, якщо поля не заповнені, завдяки вбудованій валідації HTML5);
4. перехід на головну сторінку після успішної авторизації.
5. Тестування головної сторінки (dashboard.html) включало перевірку:
6. відображення розділу «Розклад студентів» з випадаючими списками «Інститут» та «Курс»;
7. фільтрації карток PDF-розкладів при виборі значень у списках

8. відсутності кнопок адміністратора («Створити викладача», «Створити розклад», «Створити групу») для ролі студента та викладача;
9. появи цих кнопок після входу під обліковим записом адміністратора;
10. коректного відкриття PDF-файлу при кліку на картку.
11. Тестування сторінки управління викладачами (teachers_edit.html) охоплювало:
12. заповнення форми створення викладача: ПІБ, предмет і години, вибір часових слотів у таблиці розкладу, введення аудиторії;
13. перевірку відображення списку викладачів із тестовими даними (Teacher01, Teacher02), де розклад відформатовано у вигляді рядка (наприклад, «Понеділок: 1 пара, 3 пара, 4 пара»);
14. перехід у режим редагування при натисканні кнопки «Редагувати» — дані викладача автоматично заповнюють форму;
15. видалення викладача зі списку після підтвердження.
16. Тестування сторінки управління групами (students_edit.html) перевіряло:
17. вибір інституту, курсу та введення назви групи у формі додавання;
18. відображення переліку груп із тестовими даними;
19. функцію редагування: при кліку на «Редагувати» заповнюються відповідні поля, кнопка змінюється на «Оновити групу»;
20. видалення групи.
21. Тестування сторінки генератора розкладу (generator.html) було найбільш об’ємним. Перевірялися:
22. вибір групи та предмета з випадаючих списків, що містять пошук (бібліотека Choices.js);
23. додавання групи до таблиці відповідності предметів і груп;
24. введення додаткових полів: прізвищ декана та заступника начальника управління, дат початку і кінця семестру;
25. робота кнопок «Переглянути PDF» та «Завантажити PDF» за наявності коректних даних.

3.5.2 Тестування адаптивності та кросбраузерності. Однією з ключових вимог до клієнтської частини було забезпечення коректного відображення на пристроях із

різним розміром екрану. Тестування адаптивності проводилось шляхом зміни ширини вікна браузера, а також перевірки на реальних мобільних пристроях (iPhone 12, iPad) та в інструментах розробника (Chrome DevTools, режим симуляції пристроїв).

У ході тестування було підтверджено, що:

1. на екранах шириною менше 768px шапка сторінки перебудовується у вертикальний макет, логотип і назва вирівнюються по центру, а блок користувача переноситься під них;
2. навігаційні кнопки на головній сторінці (для адміністратора) втрачають фіксоване позиціонування і вирівнюються по ширині екрану, що запобігає перекриттю контенту;
3. кастомні випадаючі списки фільтрації розкладів студентів розташовуються один під одним, а не в рядок;
4. таблиці (перелік викладачів, груп) отримують горизонтальну прокрутку завдяки властивості overflow-x: auto та не виходять за межі екрану;
5. форми введення даних зберігають читабельність, поля та кнопки залишаються зручними для натискання пальцем (мінімальний розмір інтерактивних елементів — 44px).

Кросбраузерне тестування виконувалось в останніх версіях браузерів Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. У всіх браузерах інтерфейс відображався ідентично, розбіжностей у стилях або поведінці JavaScript не виявлено.

3.5.3 Тестування валідації даних. Оскільки частина даних передається на сервер через HTML-форми з попередньою обробкою на стороні клієнта, окрему увагу було приділено тестуванню валідації введених даних. Перевірялися такі сценарії:

Порожні обов’язкові поля. На сторінці авторизації спроба надіслати форму без логіну або пароля блокується стандартною валідацією HTML5 (атрибут required). На сторінці генератора розкладу при натисканні «Завантажити PDF» без додавання хоча б однієї групи або предмета з’являється повідомлення про помилку у блоці

, і форма не надсилається. При створенні викладача або групи обов’язкові поля (ПІБ, предмет, аудиторія, інститут) також позначені атрибутом required і не дозволяють надіслати порожню форму.

Некоректні числові значення. Поле «Години на тиждень» у формі викладача є числовим (type="number"), і введення від’ємних або нульових значень блокується браузером. У таблиці розкладу неможливо вибрати більше ніж 4 пари на день, оскільки кількість чекбоксів жорстко обмежена структурою таблиці.

Некоректний формат дати. Поля «Дата початку семестру» та «Дата кінця семестру» мають тип date, що забезпечує введення лише коректних дат через вбудований календар браузера. Спроба встановити дату кінця семестру раніше за дату початку логічно контролюється на серверній стороні, але на клієнті додатково перевіряється через JavaScript — стан кнопок «Переглянути PDF» та «Завантажити PDF» змінюється, якщо дати не заповнені.

Дублювання даних. На сторінці генератора розкладу при повторному додаванні тієї самої групи скрипт ігнорує дублікат, порівнюючи значення з масивом selectedGroups, і не створює зайвих стовпців у таблиці. При редагуванні викладача чи групи система не створює нового запису, а оновлює існуючий.

Результати тестування валідації підтвердили, що клієнтська частина коректно запобігає надсиланню некоректних або неповних даних, зменшуючи навантаження на серверну частину та покращуючи загальну стабільність роботи системи. Узагальнені результати функціонального тестування та перевірки валідації наведено на рис. 3.20.

| Сторінка | Сценарій | Очікуваний результат | Фактичний результат |
|---|---|---|---|
| index.html | Вхід з порожніми полями | Форма не надсилається, поля підсвічуються | Виконується |
| | Вхід із коректними даними | Перенаправлення на dashboard.html | Виконується |
| | Перехід «Увійти як студент» | Перехід на dashboard.html без авторизації | Виконується |
| dashboard.html | Фільтрація розкладів за інститутом та курсом | Відображаються лише відповідні картки PDF | Виконується |
| | Відсутність кнопок адміністратора для студента | Кнопки не відображаються | Виконується |
| | Клік на картку PDF | Відкривається повна версія документа в новій вкладці | Виконується |
| teachers_edit.html | Створення викладача з усіма полями | Запис з'являється в таблиці переліку викладачів | Виконується |
| | Редагування існуючого викладача | Поля форми заповнюються даними, після оновлення зміни зберігаються | Виконується |
| | Видалення викладача | Запис зникає з таблиці | Виконується |
| students_edit.html | Додавання нової групи | Група з'являється в переліку | Виконується |
| | Редагування групи | Зміни зберігаються, кнопка змінюється на «Оновити групу» | Виконується |
| generator.html | Генерація PDF без вибору груп | Повідомлення про помилку, форма не надсилається | Виконується |
| | Повторне додавання однієї групи | Дублікат ігнорується | Виконується |
| | Перегляд PDF з коректними даними | Відкривається згенерований PDF у новій вкладці | Виконується |

Рис. 3.20. Результати тестування основних сценаріїв

Таким чином, проведене тестування охопило всі ключові функціональні можливості клієнтської частини. Виявлених критичних дефектів, які б перешкоджали використанню системи, зафіксовано не було, що підтверджує коректність прийнятих технічних рішень та якість реалізації.

## 3.6 Аналіз результатів розробки клієнтської частини

У процесі розробки клієнтської частини вебсистеми було прийнято ряд технічних рішень, які потребують окремого обґрунтування. Деякі з них відрізняються від найбільш поширених підходів у веброзробці, однак були обрані свідомо з урахуванням конкретних вимог та умов проєкту.

Кастомні випадаючі списки замість стандартного <select>. На головній сторінці для фільтрації за інститутом та курсом використовуються власні випадаючі списки, побудовані на основі  та <ul> замість стандартного HTML-елемента <select>. Це рішення пояснюється суттєвими обмеженнями браузерного стилізування стандартного елемента <select> — зокрема неможливістю змінити форму, колір фону, шрифт та поведінку при розкритті списку у більшості браузерів.

Оскільки дизайн системи передбачає заокруглені списки із синім фоном та плавною анімацією, реалізувати це через стандартний елемент без втрати кросбраузерності неможливо. Кастомний список повністю контролюється через CSS та JavaScript, що дозволяє досягти однакового вигляду в усіх браузерах.

Використання бібліотеки Choices.js на сторінці генератора. На сторінці генератора розкладу для вибору груп та предметів підключена зовнішня бібліотека Choices.js. На відміну від головної сторінки, де списки невеликі та фіксовані, на сторінці генератора кількість груп та предметів може бути значною і залежить від даних у базі. Пошук по великому списку вручну є незручним, тому Choices.js обрано як легку бібліотеку що розширює стандартний <select> функцією живого пошуку без необхідності писати цю логіку з нуля. Бібліотека підключається через CDN і не потребує встановлення, що відповідає загальному підходу до розробки без зайвих залежностей.

Фіксоване розташування кнопок адміністратора. Кнопки "Створити викладача", "Створити розклад" та "Створити групу" на головній сторінці мають властивість position: fixed і залишаються видимими при прокрутці сторінки. Це рішення обумовлене тим, що картки розкладів можуть займати значний вертикальний простір, і адміністратору може знадобитись перейти до створення нового запису після перегляду існуючих. Фіксоване розміщення усуває необхідність прокручувати сторінку вгору для доступу до керуючих кнопок, що покращує зручність роботи адміністратора.

Шаблонізація через Jinja2 замість окремого API. Дані з бази передаються на сторінки через шаблонізатор Jinja2 безпосередньо при завантаженні сторінки, а не через окремі API-запити після завантаження. Такий підхід є доцільним для даного проєкту, оскільки більшість сторінок відображають статичні списки (викладачі, групи, PDF-файли), які не змінюються в режимі реального часу і не потребують динамічного оновлення. Серверна генерація HTML є простішою в реалізації та налагодженні, зменшує кількість мережевих запитів і забезпечує коректне відображення даних навіть при вимкненому JavaScript у браузері.

Попередній перегляд PDF через <iframe>. Для відображення мініатюр PDF-файлів розкладів на головній сторінці використовуються вбудовані фрейми <iframe>. Такий підхід дозволяє відображати реальний вміст PDF безпосередньо на сторінці без додаткових бібліотек для рендерингу. Браузер самостійно обробляє відображення PDF у фреймі, що є надійним і простим рішенням для даного випадку використання. При кліку на картку відкривається повна версія PDF у новій вкладці браузера.

Передача даних через приховані поля форми. На сторінці генератора розкладу зібрані JavaScript-ом дані передаються на сервер через приховані <input type="hidden"> поля у вигляді серіалізованого JSON. Альтернативним підходом могло бути використання Fetch API для надсилання асинхронного запиту, однак серверна частина повертає у відповідь PDF-файл для завантаження або перегляду. Обробка таких відповідей через Fetch API є значно складнішою, ніж через стандартне надсилання форми, тому використання прихованих полів є простішим і надійнішим рішенням для даного випадку.

Відсутність JavaScript-фреймворків. Вся клієнтська логіка системи реалізована на чистому JavaScript без використання фреймворків типу React, Vue.js або Angular. Це рішення обумовлене масштабом та характером проєкту — система має відносно невелику кількість сторінок з чітко визначеним функціоналом. Використання фреймворку додало б зайву складність у налаштуванні середовища розробки та збірки, збільшило б розмір файлів що завантажуються браузером, і ускладнило б інтеграцію з серверною шаблонізацією Jinja2. Чистий JavaScript є достатнім для реалізації всього необхідного функціоналу і забезпечує просту підтримку коду.

Стилізація таблиць із заокругленими кутами. Для досягнення заокруглених кутів у таблицях використано нестандартний підхід — властивість border-collapse: separate замість звичайного border-collapse: collapse. Стандартний підхід із collapse об'єднує межі комірок, але унеможливлює застосування border-radius до окремих комірок. Використання separate з нульовим border-spacing: 0 зберігає візуальний ефект суцільних меж і водночас дозволяє застосовувати заокруглення до кутових комірок таблиці через псевдоселектори :first-child та :last-child.

Для узагальненої оцінки результатів розробки наведемо порівняння між вимогами, сформульованими на етапі проєктування, та реалізованим функціоналом. Відповідність вимог та результатів розробки наведено у рис. 3.21.

| Вимога | Результат | Статус |
|---|---|---|
| Сторінка авторизації для всіх користувачів | Реалізовано index.html з формою входу та посиланням для студента | Виконано |
| Перегляд розкладів студентів з фільтрацією | Реалізовано на dashboard.html з фільтрами за інститутом та курсом | Виконано |
| Перегляд розкладів викладачів | Реалізовано на dashboard.html з фільтром за ПІБ | Виконано |
| Адміністративна панель керування викладачами | Реалізовано teachers_edit.html з формою та таблицею | Виконано |
| Адміністративна панель керування групами | Реалізовано students_edit.html з формою та таблицею | Виконано |
| Генерація розкладу у форматі PDF | Реалізовано generator.html з кнопками перегляду та завантаження | Виконано |
| Адаптивний дизайн для мобільних пристроїв | Реалізовано медіазапити для всіх сторінок | Виконано |
| Розмежування доступу за роллю | Реалізовано через умовну логіку Jinja2 | Виконано |
| Єдиний корпоративний стиль МАУП | Реалізовано через base.css та спільний базовий шаблон | Виконано |

Рис. 3.21. Відповідність вимог та результатів розробки

Як видно з таблиці, всі вимоги до клієнтської частини системи виконано у повному обсязі.

Таким чином, усі прийняті технічні рішення мають конкретне обґрунтування і відповідають вимогам проєкту. Пріоритетом при розробці була простота реалізації та підтримки, коректне відображення у різних браузерах і зручність використання для кінцевого користувача.

## 3.7 Висновки до третього розділу

У третьому розділі було виконано повний цикл розробки клієнтської частини вебсистеми для генерації розкладу навчального процесу — від проєктування структури до тестування готового інтерфейсу.

На етапі проєктування визначено структуру системи з п'яти сторінок та описано функції кожної з них для трьох типів користувачів — студента, викладача

та адміністратора. Розроблено wireframe-макети всіх сторінок, які слугували основою для подальшої реалізації.

У розділі розробки HTML-структури реалізовано спільний базовий шаблон base.html, який забезпечує єдину шапку та підключення ресурсів для всіх сторінок. Для кожної зі сторінок побудовано семантичну HTML-розмітку відповідно до її функціонального призначення — форми введення, таблиці даних, кастомні елементи інтерфейсу та динамічні блоки.

Розроблені CSS-стилі забезпечують єдиний корпоративний дизайн у синіх тонах на всіх сторінках системи. Реалізовано адаптивний дизайн за допомогою медіазапитів, що забезпечує коректне відображення інтерфейсу на мобільних пристроях. Застосовано сучасні інструменти розкладки Flexbox та CSS Grid для побудови складних елементів інтерфейсу.

JavaScript-логіка кожної сторінки реалізує відповідний функціонал — фільтрацію карток розкладу на головній сторінці, динамічне формування таблиці на сторінці генератора, два режими роботи форм на сторінках управління даними. Передача даних на серверну частину реалізована через стандартні HTML-форми з серіалізацією складних структур у формат JSON.

Проведено функціональне тестування всіх п'яти сторінок системи за методом «чорної скриньки», тестування адаптивності у браузерах Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, а також перевірку валідації введених даних. Критичних дефектів виявлено не було, що підтверджує коректність реалізації клієнтської частини.

У розділі аналізу результатів обґрунтовано основні технічні рішення, прийняті в ході розробки. Встановлено, що вибір стеку HTML5, CSS3 та JavaScript без фреймворків є обґрунтованим для даного масштабу проєкту і забезпечує простоту підтримки та розгортання системи.

# ВИСНОВКИ

Результатом роботи є розробка клієнтської частини вебсистеми для генерації розкладу навчального процесу. Розроблений інтерфейс забезпечує зручну взаємодію з системою для трьох категорій користувачів — студентів, викладачів та адміністраторів, і може бути впроваджений у закладах вищої освіти для автоматизації процесу формування та перегляду розкладу занять.

Клієнтська частина реалізована з використанням технологій HTML5, CSS3 та JavaScript без застосування зовнішніх фреймворків, що забезпечує простоту розгортання та підтримки системи. Для шаблонізації сторінок використано механізм Jinja2, який дозволяє динамічно формувати вміст сторінок на основі даних із серверної частини. Зовнішній вигляд інтерфейсу витримано в єдиному корпоративному стилі МАУП з використанням адаптивного дизайну для коректного відображення на мобільних пристроях.

Під час написання кваліфікаційної роботи були виконані такі завдання:

1. проаналізовано предметну область та особливості процесу складання розкладу навчального процесу;
2. виконано аналіз існуючих систем генерації розкладу — ASC Timetables та UniTime — та визначено їх переваги і недоліки;
3. виконано аналіз схожих вебзастосунків для управління розкладом — системи розкладу КПІ та сервісу MyTimetable;
4. виконано порівняльний аналіз технологій розробки клієнтської частини вебзастосунків — HTML5, CSS3 та JavaScript;
5. спроєктовано структуру системи з п'яти сторінок та розроблено wireframe-макети кожної з них;
6. розроблено HTML-структуру всіх сторінок вебсистеми на основі спільного базового шаблону;
7. розроблено стилі та адаптивний дизайн засобами CSS для всіх сторінок системи;
8. реалізовано логіку інтерфейсу та взаємодію з серверною частиною засобами JavaScript;

9. проведено функціональне тестування, тестування адаптивності та кросбраузерності, а також валідації даних клієнтської частини вебсистеми;
10. проаналізовано прийняті технічні рішення та обґрунтовано їх доцільність.

Подальшим розвитком системи може бути вдосконалення клієнтської частини шляхом додавання сповіщень для студентів та викладачів про зміни в розкладі, реалізації можливості експорту розкладу до календарних застосунків, а також розробки окремого мобільного застосунку для зручнішого перегляду розкладу на смартфонах.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сіденко В. Р. Організація навчального процесу у закладах вищої освіти. -- К. : Педагогіка, 2019. -- 318 с.

2. Іваненко П. С. Системи управління навчальним процесом у ЗВО. -- К. : Університет, 2017. -- 289 с.

3. CSS Cascading and Inheritance Level 3 / W3C [Електронний ресурс]. -- Режим доступу: https://www.w3.org/TR/css-cascade-3/ -- Дата доступу: 10.03.2026.

4. Гаврилюк М. О. Проєктування інтерфейсів користувача: навч. посіб. -- К. : КПІ, 2019. -- 178 с.

5. Романів О. Я. Людино-машинна взаємодія та проєктування інтерфейсів. -- Львів : Львівська політехніка, 2020. -- 244 с.

6. ECMAScript 2024 Language Specification / Ecma International [Електронний ресурс]. -- Режим доступу: https://ecma-international.org/publications-and-standards/standards/ecma-262/ -- Дата доступу: 11.03.2026.

7. Flask Documentation / Pallets Projects [Електронний ресурс]. -- Режим доступу: https://flask.palletsprojects.com -- Дата доступу: 15.03.2026.

8. Михайленко О. В. CSS3: сучасні підходи до стилізації вебсторінок. -- К. : Університет, 2021. -- 210 с.

9. Fetch Standard / WHATWG [Електронний ресурс]. -- Режим доступу: https://fetch.spec.whatwg.org -- Дата доступу: 13.03.2026.

10. Дудко С. В. Адаптивний дизайн вебзастосунків: принципи та практика. -- К. : Академія, 2020. -- 196 с.

11. Мельник В. А. Розробка клієнт-серверних вебдодатків. -- К. : Академія, 2022. -- 232 с.

12. Nielsen J. Designing Web Usability: The Practice of Simplicity / J. Nielsen. -- Indianapolis : New Riders, 2000. -- 432 p.

13. Simpson K. You Don’t Know JS: Up & Going / K. Simpson. -- Sebastopol : O’Reilly Media, 2015. -- 88 p.

14. Osmani A. Learning JavaScript Design Patterns / A. Osmani. -- Sebastopol : O'Reilly Media, 2023. -- 334 p.

15. ASC Timetables [Електронний ресурс]. -- Режим доступу: https://www.asctimetables.com -- Дата доступу: 05.03.2026.

16. UniTime University Timetabling System Highlights [Електронний ресурс]. -- Режим доступу: https://www.unitime.org/unitime_intro.php -- Дата доступу: 08.03.2026.

17. Розклад занять НТУУ «КПІ ім. Ігоря Сікорського» [Електронний ресурс]. -- Режим доступу: https://rozklad.kpi.ua -- Дата доступу: 07.03.2026.

18. MyTimetable [Електронний ресурс]. -- Режим доступу: https://www.mytimetable.net -- Дата доступу: 06.03.2026.

19. HTML Living Standard / WHATWG [Електронний ресурс]. -- Режим доступу: https://html.spec.whatwg.org -- Дата доступу: 10.03.2026.

20. MDN Web Docs — JavaScript, HTML, CSS / Mozilla [Електронний ресурс]. -- Режим доступу: https://developer.mozilla.org -- Дата доступу: 12.03.2026.

21. Jinja2 Documentation / Pallets Projects [Електронний ресурс]. -- Режим доступу: https://jinja.palletsprojects.com -- Дата доступу: 18.03.2026.

22. Choices.js [Електронний ресурс]. -- Режим доступу: https://choices-js.github.io/Choices -- Дата доступу: 20.03.2026.

23. Keith J. HTML5 for Web Designers / J. Keith, R. Andrew. -- New York : A Book Apart, 2016. -- 133 p.

24. Duckett J. HTML & CSS: Design and Build Websites / J. Duckett. -- Indianapolis : Wiley, 2011. -- 490 p.

25. Frain B. Responsive Web Design with HTML5 and CSS / B. Frain. -- Birmingham : Packt Publishing, 2020. -- 426 p.

26. Crockford D. JavaScript: The Good Parts / D. Crockford. -- Sebastopol : O'Reilly Media, 2008. -- 172 p.