ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для автоматизації розкладу занять у закладі освіти»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Марк БІЛОУС
(підпис)

Виконав: здобувач вищої освіти групи ПД-41
Марк БІЛОУС

Керівник: Світлана ШЕВЧЕНКО
к.п.н., доцент
Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМІНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Бакалавр
Спеціальність 121 Інженерії програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Білоусу Марку Олександровичу

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для автоматизації розкладу занять у закладі освіти»
керівник кваліфікаційної роботи к.п.н., доцент Світлана ШЕВЧЕНКО,
Затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про автоматизацію розкладу занять у закладі освіти; інструменти розробки програмного забезпечення.

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

1. Провести аналіз науково-технічних джерел щодо вимог до розробки web-застосунків.
2. Проаналізувати існуючі рішення для автоматизації розкладу занять.
3. Здійснити огляд та аналіз існуючих рішень, що використовують для автоматизації розкладу у навчальних закладах.
4. Сформувати вимоги до розробки клієнт-серверного застосунку для автоматизації розкладу занять у закладі освіти.
5. Спроєктувати архітектуру клієнт-серверного застосунку.

6. Розробити клієнт-серверний застосунок для автоматизації навчального розкладу у закладі освіти.
7. Провести тестування клієнт-серверного застосунку.

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів.
   2. Вимоги до програмного забезпечення.
   3. Програмні засоби реалізації.
   4. Діаграма варіантів використання.
   5. Структура бази даних.
   6. Діаграма класів основної логіки.
   7. Діаграма класів авторизації.
   8. Екранні форми.
   9. Апробація результатів дослідження.

6. Дата видачі завдання «25» лютого 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Огляд існуючих методів автоматизації, теоретичні основи автоматизації навчального розкладу	14.03 – 16.03.2025	
4	Проектування web-застосунку для автоматизації навчального розкладу	17.03 – 31.03.2025	
5	Програмна реалізація застосунку	01.04 – 22.04.2025	
6	Тестування застосунку	22.04 – 28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Марк БІЛОУС
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ШЕВЧЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 38 стор., 1 табл., 19 рис., 26 джерел.

Мета роботи – підвищення ефективності управління навчальним процесом за рахунок автоматизації розкладу занять у закладі освіти.

Об'єкт дослідження – процес автоматизованого складання розкладу занять у закладі освіти.

Предмет дослідження – web-застосунок для автоматизації розкладу занять у закладі освіти.

Короткий зміст роботи: В роботі проаналізовано предметну галузь та методи реалізації застосунку автоматизації навчального розкладу в закладах освіти. Проведено огляд вже існуючих засобів для автоматизації розкладу: МКР, UniTime. Розроблено вебсайт, телеграм чат-бот та програмно реалізовано основні функціональні можливості, такі як: авторизація, налаштування розкладу в навчальних закладах, система відображення розкладу як для студентів, так і викладачів. Проведено модульне тестування застосунку. В роботі використано фреймоворки Node.js Express для розробки бекенду та фронтенду відповідні, RestFullApi Node, MongoDB для зберігання даних, Python для чат-боту.

Сферою використання застосунку є навчальні заклади, які прагнуть реалізувати та впровадити автоматизацію навчального розкладу.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ РОЗКЛАДУ, BACK-END, FRONT-END, PYTHON, RESTful API, WEB-ЗАСТОСУНОК, MONGO DB, JAVASCRIPT, NODE JS.

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах цифрової трансформації освіти зростає потреба в ефективних інструментах для автоматизації організаційних процесів у навчальних закладах. Одним із ключових напрямів є автоматизація розкладу занять. Традиційні підходи, що базуються на паперовому документообігу або офлайн-редакторах, є неефективними, трудомісткими та не забезпечують необхідної гнучкості й оперативності в управлінні навчальним процесом. Web застосунок для автоматизації навчального розкладу дозволяє спрямувати зусилял на вирішення ключових питань, серед яких, мінімізувати вплив людського фактора при формування розкладу, забезпечити зручний доступ до інформації для викладачів, студентів та адміністрації, а також підвищити прозорість і контроль над освітнім процесом. Однак одним з найзручніших способів автоматизації навчального розкладу є web-застосунок та telegram чат-бот. Web-застосунок дозволяє централізовано керувати навчальними даними: створювати та редагувати розклад занять, здійснювати моніторинг навантаження викладачів. Доповнення системи чат-ботом забезпечує оперативну взаємодію з кінцевими користувачами – студентами та викладачами. Telegram є одним із найпопулярніших месенджерів серед молоді. За даними Statista, у лютому 2025 року Telegram був третім за популярністю месенджерів у світі [1]. Відкритий API в месенджері дозволяє інтегрувати навчальні функції у звичне для користувачів середовище. В 2019 році в звіті European Schoolnet зазначено, використання месенджерів для навчальної комунікації значно покращує залучення студентів і дозволяє адаптувати навчальні процеси до цифрових звичок сучасного покоління [2]. Крім того, чат-бот забезпечує швидке надсилання персоналізованих повідомлень, автоматичні нагадування про розклад на наступний день, повідомлення про зміни в розкладі, що зменшує ризики пропусків і підвищує дисципліну. Вище перераховане підтверджує актуальність даного дослідження.

Метою роботи є підвищення ефективності управління навчальним процесом за рахунок автоматизації розкладу занять у закладі освіти.

Для досягнення цієї мети в роботі необхідно вирішити такі *завдання:*

1. Провести аналіз науково-технічних джерел щодо вимог до розробки web-застосунків.
2. Проаналізувати існуючі рішення для автоматизації розкладу занять.
3. Здійснити огляд та аналіз існуючих рішень, що використовують для автоматизації розкладу у навчальних закладах.
4. Сформувати вимоги до розробки клієнт-серверного застосунку для автоматизації розкладу занять у закладі освіти.
5. Спроєктувати архітектуру клієнт-серверного застосунку.
6. Розробити клієнт-серверний застосунок для автоматизації навчального розкладу у закладі освіти.
7. Провести тестування клієнт-серверного застосунку.

Об'єктом дослідження є процес автоматизованого складання розкладу занять у закладі освіти.

Предметом дослідження є web-застосунок для автоматизації розкладу занять у закладі освіти.

Методи дослідження. Для вирішення вищезгаданих завдань у роботі використано наступні методи: системно-структурні методи, порівняльний аналіз. Для розробки програмного забезпечення використані такі технології: React/Angular/Vue для фронтенду, Node.js/Django для бекенду, MongoDB/PostgreSQL для зберігання даних.

Наукова новизна роботи визначається наступним функціональними складовими: чат-бот асистент з цілодобовим доступом, підтримка оповіщення про зміни в розкладі. Web-застосунок постає як всебічний пакет усіх найбільш затребуваних інструментів в одному місці.

Практична значущість результатів полягає в можливості використання реалізованого застосунку для ефективної автоматизації навчального розкладу та зменшення трудомісткості в управлінні навчальним процесом.

1 ТЕОРИТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО РОЗКЛАДУ

1.1 Поняття та принципи побудови навчального розкладу

Навчальний розклад – це дуже чітко структурований план проведення навчальних занять, який визначає послідовність, час та місце. Завдання полягає в забезпеченні належної організації освітньої діяльності та ефективному управлінні ресурсами закладу. Коли складають розклад занять, основна складність полягає в тому, що втомлюваність студентів залежить від багатьох соціальних і психологічних факторів. Їхній спільний вплив може як збільшити, так і зменшити рівень втоми або ж залишити його незмінним [3]. Основні вимоги до розкладу в закладах вищої освіти є:

1. Відсутність конфліктів – тож одна група не може бути присутня одночасно на кількох парах, те саме стосується викладачів та аудиторій.
2. Дотримання навчального плану – усі дисципліни мають бути розміщені у відповідній кількості годин.
3. Рівномірне навантаження – уникнення перевантаження студентів і викладачів протягом одного дня.
4. Персональні побажання викладачів або обмеження щодо їх присутності в певні дні або години.

Під час формування ручним способом розкладу виникає безліч проблем, які негативно впливають на розклад, а саме:

1. Велика кількість змінних – у процесі формування враховується безліч параметрів (групи, викладачі, аудиторії, дисципліни)
2. Людський фактор – помилки при розподілі годин, пропуск занять або конфлікти між подіями
3. Трудомісткість – складання розкладу вручну займає значну кількість часу, особливо для великих навчальних закладів.

4. Складність адаптації – у разі змін, перерахунок розкладу потребує багато зусиль.

Автоматизація цього процесу при використанні web-застосунку має можливість підвищувати ефективність багато разів, зменшувати кількість помилок та забезпечувати гнучкість у плануванні. Така система може вміщати різні обмеження та переваги та забезпечувати оптимальний розподіл ресурсів [4-8].

1.2 Сучасні підходи до автоматизації розкладу

Задача автоматизації розкладу занять має низку характерних особливостей, які значно впливають на процес проєктування та розробки відповідного програмного забезпечення.

1. Багатокритеріальність.

Процес формування розкладу повинен враховувати велику кількість взаємопов’язаних обмежень і побажань: наявність викладачів, доступність аудиторій, особливості навчального плану, пріоритети курсів, санітарні норми, навантаження тощо. Це робить задачу складною для ручного вирішення і потребує застосування спеціалізованих алгоритмів.

2. Динамічність.

Розклад занять у навчальному закладі не є статичним: можуть змінюватися час проведення занять, групи студентів, викладачі або аудиторії. Тому система повинна передбачати можливість швидкого редагування, автоматичного оновлення даних і збереження історії змін.

3. Масштабованість.

Рішення має бути придатним як для невеликих коледжів, так і для великих університетів з десятками факультетів і сотнями груп. Відповідно, програмний продукт повинен зберігати стабільність та ефективність при зростанні обсягу даних і кількості користувачів.

Розглянемо засоби для автоматизації навчального розкладу, а саме: програмні засоби — МКР та UniTime, які застосовуються для вирішення автоматизації

розкладу та традиційний метод реалізації навчального розкладу – ручне планування.

Автоматизована Система Управління «МКР» - це найпоширеніший застосунок, який використовують в Україні. Він допомагає в більшості автоматизувати задачі в закладі.

Це програмне забезпечення представляє собою комплекс взаємопов'язаних програм, призначених для управління закладом вищої освіти в межах єдиного інформаційного простору. Воно містить модулі, що функціонують у середовищі Windows.

Він має такі елементи:

1. Створення картки здобувача освіти в АСУ «МКР». Закріплення та зарахування студентів.
2. Відрахування, переведення, поновлення, академічна відпустка, завершення навчання.
3. Картка студента.
4. Видача довідок, а саме (академічна довідка, довідка про навчання студента, табель успішності, додаток до диплома, обхідні листи).

Також, вони мають web-портал, який допомагає у відображенні розкладу для студентів та викладачів, індивідуальні плани студентів. Таблиця розкладу представлена на рисунку 1.1.

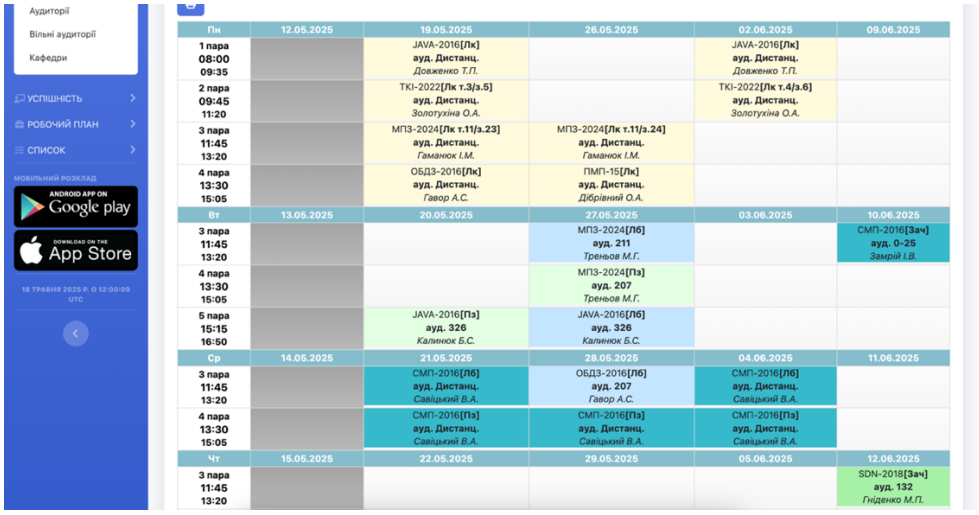


Рис. 1.1 Web-портал перегляду розкладу в МКР

Переваги використання АСУ «МКР»:

1. Система створена з урахуванням вимог та особливостей українських навчальних закладів, включаючи структуру навчального процесу, часові сітки, типи занять, назви дисциплін та інше.
2. МКР уже інтегровано з розкладом багатьох українських університетів і коледжів, що дозволяє студентам і викладачам одразу підключатися до свого розкладу без зайвих налаштувань.
3. У разі змін у розкладі (перенесення пар, зміна викладача чи аудиторії) інформація оновлюється в додатку автоматично, що дозволяє уникнути плутанини та втрати часу.

Цей застосунок добре працює для великих навчальних закладів, але має великі труднощі для налаштування роботи в менших закладах, а саме коледжі.

UniTime – система для автоматизованого складання розкладів у ЗВО. Вона є безкоштовною, має відкритий код і високу гнучкість у налаштуванні. UniTime добре підходить для великих університетів зі складною структурою. Її переваги — це широка функціональність і підтримка складних сценаріїв розкладу. Система дозволяє ефективно керувати розкладом та адаптувати його під потреби конкретного закладу. Інтерфейс представлено на рисунку 1.2.

Переваги використання UniTime:

1. Дозволяє враховувати численні обмеження та пріоритети: завантаження викладачів, доступність приміщень, конфлікти розкладу, побажання студентів або викладачів. Користувачі можуть створювати власні правила і сценарії планування.
2. Підтримує повний цикл роботи з розкладом: створення навчальних курсів, планування занять, розподіл студентів за групами, планування іспитів та управління аудиторним фондом. Це дозволяє закладу освіти централізовано організувати весь навчальний процес.
3. Можливість автоматичного та ручного коригування. Розклад можна генерувати автоматично з урахуванням усіх заданих

параметрів, а потім редагувати вручну для точного налаштування окремих деталей.

4. Відкритий код та безкоштовне використання. UniTime — проєкт з відкритим програмним кодом, що дозволяє безкоштовно використовувати його, а також змінювати і розширювати функціонал відповідно до потреб навчального закладу.
5. Підтримка англомовною спільнотою. У системи є активна спільнота користувачів та розробників, які обговорюють оновлення, діляться досвідом та надають технічну підтримку через форуми та офіційний сайт.

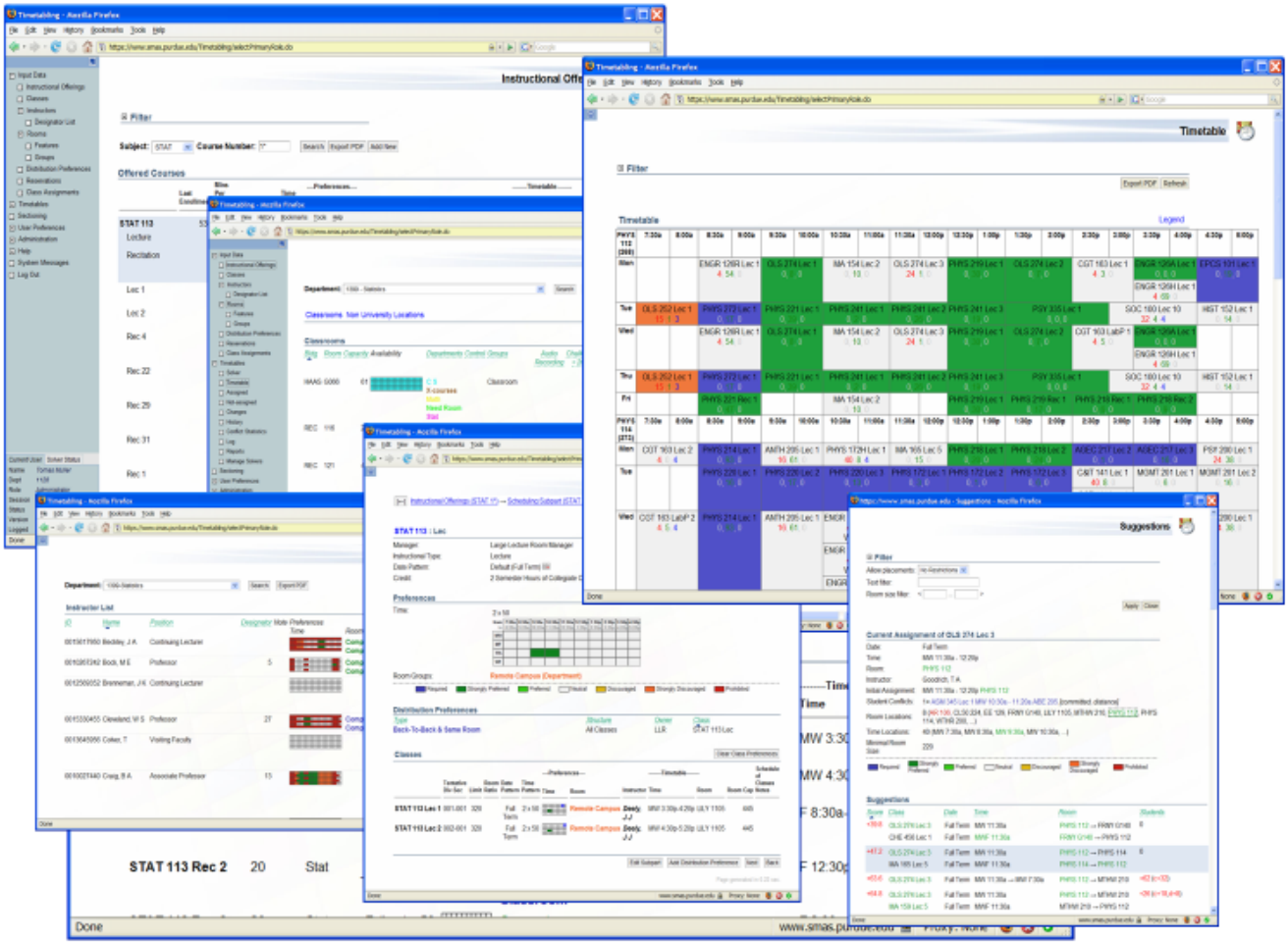


Рис. 1.2. Приклад використання UniTime для автоматизації навчального розкладу

Основним недоліком UniTime можна вважати складність у впровадженні та налаштуванні**.** Через велику кількість функцій і гнучкість система може вимагати значних технічних знань та часу для адаптації під конкретний навчальний заклад.

У минулому використання електронних таблиць для складання розкладу занять вважалося зручним і практичним способом. Це було зумовлено тим, що більшість навчальних закладів не мали доступу до спеціалізованих інформаційних систем. Табличний формат дозволяв швидко вносити дані, організовувати їх у вигляді простого сіткового розкладу та оперативно оновлювати інформацію вручну.

Однак з розвитком цифрових технологій та збільшенням обсягів даних у сучасних навчальних процесах такий підхід втрачає свою ефективність. Основна проблема полягає у складності відстеження завантаженості викладачів, особливо при наявності великої кількості груп, курсів та аудиторій. Часто доводиться робити додаткові позначки вручну або у примітках, що підвищує ризик помилок і ускладнює коректність формування розкладу. Приклад таблиці для планування розкладу наведено на рисунку 1.3.

Розклад занять **Осінній сем**

Час початку: 7:30:00 Часовий інтервал: 30 (у хвил

Час	Пн	Вт	Ср	
7:30:00	Сніданок	Сніданок	Сніданок	Сн
8:00:00	Бізнес: лекція Корпус B, аудиторія 256	Фізика: лабораторне заняття Корпус J, аудиторія 309	Бізнес: лекція Корпус B, аудиторія 256	Фізика: З Корпус
8:30:00				
9:00:00	Прикладна математика Корпус E, аудиторія 100		Прикладна математика Корпус E, аудиторія 100	
9:30:00				
10:00:00				
10:30:00				
11:00:00				
11:30:00				

Рис. 1.3. Приклад таблиці планування розкладу

Переваги використання електронних таблиць у процесі планування:

− зручний та інтуїтивно зрозумілий інтерфейс, що відповідає табличній структурі самого розкладу;

− можливість створення списків підстановки та автоматичного заповнення клітинок, що значно полегшує введення даних і знижує ризик помилок при повторному введенні однакових назв дисциплін.

Втім, попри ці переваги, електронні таблиці мають низку суттєвих обмежень, які ускладнюють їх застосування для формування розкладу. Серед основних недоліків варто відзначити:

1. Створення ефективних шаблонів потребує досвіду та знання функціоналу таблиць.
2. Зі збільшенням обсягу даних таблиці стають громіздкими і складними для навігації.
3. Необхідність ручного копіювання шаблонів і формул для кожного нового навчального періоду.
4. Введення даних здійснюється вручну, що призводить до ймовірних помилок та пропусків.
5. Відсутність автоматичної структурування інформації за категоріями (наприклад, викладач, аудиторія, дисципліна).
6. Неможливість централізованого контролю і перевірки конфліктів у реальному часі.

З огляду на ці обмеження, електронні таблиці можна вважати тимчасовим і застарілим засобом, який не відповідає сучасним вимогам до гнучкості, точності та автоматизації навчального процесу.

Зведені результати аналізу характеристик розглянутих додатків наведено у таблиці 1.1.

Таблиця 1.1

Зведені результати аналізу характеристик додатків для планування розкладу

Показник	**АСУ «МКР»**	**UniTime**	**Електронні таблиці**
Платформи	Windows, Web	Windows, Web	Windows
Наявність звітності загрузки викладачів	+	+	-
Автоматична структурування інформації	+	+	-
Додавання груп, спеціальностей та курсів	+	+	Лише вручну
Сповіщення про зміни в розкладі	-	-	-
Перегляд розкладу на інших пристроях	+	+	-

1.3 Цільова аудиторія для автоматизації навчального розкладу

Цільовими користувачами системи автоматизації розкладу занять є адміністрація навчального закладу, викладачі та студенти. Кожна з цих категорій має свої специфічні потреби та очікування щодо функціональності програмного продукту.

1. Адміністрація навчального закладу – основні користувачі системи. Потребують ефективних інструментів для планування та управління навчальним процесом. До таких інструментів належать: засоби для створення, редагування та публікації розкладу, можливість розподілу ресурсів (аудиторій, викладачів), автоматичне виявлення конфліктів та аналітика завантаження персоналу
2. Викладачі потребують доступу до особистого розкладу, можливості бачити зміни в реальному часі.

3. Студенти – це основні споживачі розкладу. Прагнуть мати постійний і своєчасний доступ до актуального розкладу занять, включаючи зміни, що відбуваються в реальному часі. Їм важливо отримувати розклад у зручному форматі, з можливістю фільтрації за днями, викладачами, а також користуватись мобільною версією застосунку або додатком.

Серед спільних вимог усіх категорій користувачів — інтуїтивно зрозумілий інтерфейс, швидкий доступ до інформації і високий рівень захисту персональних та навчальних даних.

Таким чином, незважаючи на велику кількість уже наявних систем, ринок все ще потребує універсального рішення, яке поєднувало б простоту, адаптивність, підтримку сучасних освітніх сценаріїв і доступність для закладів різного масштабу. Саме ці потреби й стали основою для розробки власного web-застосунку, що пропонує гнучкість, простоту та оптимізацію під реальні умови роботи типових українських навчальних закладів.

2 РОЗРОБКА WEB-ЗАСТОСУНКУ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО РОЗКЛАДУ

2.1 Постановка задачі та вимоги до системи

Метою даної роботи є розробка web-застосунку для автоматизованого формування навчального розкладу в закладах вищої освіти. Система повинна забезпечити ефективне створення, редагування та публікацію розкладів, уникати конфліктів між заняттями, рівномірно розподіляти навантаження та надавати зручний доступ для всіх учасників освітнього процесу.

Функціональні вимоги – це ключові вимоги. Описують, що система повинна робити. До функціональних вимог віднесемо головні пункти роботи застосунку:

1. Система повинна дозволяти представнику навчального представника створювати, видаляти та редагувати основну інформацію (час пар, групи, курс, спеціальності, викладачі).
2. Користувач повинен мати змогу переглядати розклад за групою, викладачем.
3. Система повинна надати можливість пошуку розкладу за ключовими параметрами.
4. Користувач повинен мати змогу додати заявку на підключення системи до навчального закладу.
5. Система повинна дозволяти адміністратору створювати нових користувачів.

А нефункціональні вимоги – описують, як саме застосунок повинен це робити, під час аналізу ми визначили такі пункти:

1. Інтерфейс повинен бути адаптивним і коректно відображатись на всіх пристроях.
2. Система повинна забезпечувати захист персональних даних користувачів.
3. ПЗ повинно бути сумісним з основними сучасними браузерами.

2.2 Середовище розробки

Інтегроване середовище розробки — це програмне забезпечення, яке об'єднує в собі всі основні інструменти, необхідні для створення програм: редактор коду, засоби збірки, налагодження (debug), інтеграцію з системами контролю версій тощо. Такий підхід забезпечує зручність і ефективність у роботі програміста, оскільки усі компоненти доступні в одному середовищі [24].

Visual Studio Code — це легкий, але надзвичайно потужний редактор коду від компанії Microsoft, який отримав широку популярність серед JavaScript-розробників завдяки своїй гнучкості, швидкодії та розширюваності. Попри те, що VS Code технічно не є повноцінним IDE, завдяки великій кількості доступних розширень він може забезпечити всі функції повноцінного середовища розробки.

Для JavaScript та суміжних технологій (TypeScript, Node.js, React тощо) VS Code є одним із найкращих варіантів [25]. Завдяки підтримці розширень, таких як:

1. ESLint для автоматичної перевірки синтаксису,
2. Prettier для форматування коду,
3. Live Server для перегляду вебсторінок у реальному часі,
4. GitLens для роботи з Git.

VS Code забезпечує розробникам усе необхідне для повного циклу розробки JavaScript-додатків. Крім того, інтеграція з Node.js та можливість запуску терміналу безпосередньо в редакторі роблять роботу з серверною частиною JavaScript (наприклад, Express.js) зручною та швидкою.

PyCharm — це повноцінне інтегроване середовище розробки (IDE), створене компанією JetBrains та орієнтоване насамперед на мову Python. Воно містить широкий набір інструментів для професійної роботи з Python, зокрема підтримку популярних фреймворків, таких як Django, Flask і FastAPI, що забезпечує високу ефективність при розробці вебзастосунків.

Крім того, PyCharm пропонує вбудовані засоби для роботи з базами даних, а також підтримку HTML, CSS і JavaScript, що робить його зручним інструментом для повноцінної веброзробки на основі Python [26].

2.3 Обґрунтування вибору технологій розробки

У межах даної роботи розробляється web-застосунок. Для backend-частини проєкту було обрано JavaScript — високорівневу мову програмування, яка підтримує імперативний, функціональний і подієво-орієнтований підходи. Вона має динамічну типізацію й зазвичай використовується для написання сценаріїв або скриптів. Такі послідовності, як правило, інтерпретуються, а не компілюються, що дозволяє обійтися без додаткових інструментів для трансляції коду. Мова JavaScript була створена майже 30 років тому, і на сьогодні залишається однією з найпопулярніших у світі. Згідно з рейтингом PYPL, у травні 2025 року JavaScript займала третє місце за популярністю серед мов програмування [9]. Цей рейтинг базується на аналізі кількості пошукових запитів до мовних посібників у Google [9]. Для реалізації серверної логіки було використано платформу Node.js — сучасне середовище виконання JavaScript, що дозволяє створювати ефективні та масштабовані backend-рішення. Node.js забезпечує асинхронну обробку запитів, що позитивно впливає на продуктивність, особливо в застосунках із великою кількістю одночасних з'єднань. Завдяки використанню JavaScript як на frontend, так і на backend, досягається уніфікованість стеку технологій, що спрощує розробку та підтримку застосунку. Для клієнтської частини також було обрано мову JavaScript, що забезпечує узгодженість між frontend і backend та дозволяє використовувати спільні підходи до обробки даних і взаємодії з сервером.

Для реалізації front-end частини було використано бібліотеку React [10]. Вона є однією з найпопулярніших JavaScript-бібліотек: за даними JetBrains, React очолював рейтинг у 2023 році [11], а у 2023 та 2024 роках посідав друге місце серед найпопулярніших web-фреймворків серед розробників у світі [12]. React широко застосовується для створення надійних і динамічних користувацьких інтерфейсів у web-застосунках. Його основна перевага полягає у декларативному підході до розробки UI, що робить код зрозумілішим, легшим у підтримці та масштабуванні. Ключовою технологічною особливістю React є використання віртуального DOM,

який дозволяє ефективно оновлювати лише ті компоненти інтерфейсу, що зазнали змін, без необхідності повного перерендерення сторінки. Такий підхід забезпечує високу продуктивність і спрощує процес створення інтерактивних елементів у сучасних web-інтерфейсах.

Для збереження розкладу, викладачів, академічні групи, спеціальностей та іншого було використано базу даних MongoDB. MongoDB – це документо-орієнтована система керування базами даних з відкритим кодом, яка не потребує опису схеми таблиць [13]. Вона поєднує в собі переваги високої продуктивності та масштабованості систем, що працюють з форматом типу «ключ-значення», із гнучкістю та функціональністю реляційних баз даних. Також підтримує зберігання документів в JSON-подібному форматі, що дозволяє легко інтегруватися з веб-застосунками, побудованими на JavaScript та Node.js. Система підтримує створення індексів за будь-якими полями документів, що значно пришвидшує доступ до даних. Крім того, MongoDB дозволяє зберігати великі бінарні об’єкти (наприклад, файли), забезпечуючи при цьому високу швидкодію і надійність обробки запитів.

Для телеграм бота, який допомагає в швидкому доступі переглянути розклад, отримувати розклад на наступний день автоматично. Та також сповіщення про зміни в розкладі було обрано мову програмування Python. Python – це сучасна, зручна та популярна мова, яка підходить як для початківців, так і для досвідчених розробників. Вона дозволяє швидко писати зрозумілий код, що особливо важливо при створенні прототипів і навчальних проєктів. Мову створили у 1991 році, а саме на 2025 рік їй виповнилось 24 роки. Вона була натхненна навчальною мовою ABC [16]. Вона інтерпретована мова, тобто її код виконується без попередньої компіляції. Це дозволяє розробнику швидше знаходити та виправляти помилки. Також вона автоматично керує пам’яттю, що зменшує ймовірність помилок при роботі з ресурсами. За даними PYPL, у травні 2025 року Python була першою за популярністю мовою у світі [9].

Наше дослідження присвячене розробці web-застосунку для автоматизації розкладу занять у закладі освіти. Враховуючи, що система оперує конфіденційною інформацією — зокрема, даними про викладачів, здобувачів освіти, навчальні

приміщення та програмне забезпечення — розробка такої системи потребує особливої уваги до питань безпеки, включаючи обов’язкову реєстрацію користувачів та шифрування даних. Сучасна система автоматизованого розкладу [14, 15] передбачає взаємодію користувача з вебінтерфейсом та серверною частиною, де зберігаються основні дані. API виступає ядром системи, яке містить усю інформацію щодо занять для різних курсів, груп та спеціальностей. Для захисту даних необхідне впровадження таких механізмів, як хешування паролів та авторизація за допомогою токенів.

Хешування паролів [17] є ключовим методом захисту даних у середовищі Node.js. Для цього часто використовується бібліотека bcryptjs, яка забезпечує надійне хешування з додаванням "солі" — випадкових даних, що ускладнюють злам методом перебору. У нашому проєкті для паролів застосовується 10 раундів хешування, що забезпечує баланс між рівнем безпеки та швидкістю обробки. Паролі зберігаються у зашифрованому вигляді, що дозволяє мінімізувати ризики у разі витоку даних із бази.

Крім того, реалізована система авторизації на основі токенів. Наприклад, якщо користувач має нестандартну роль (наприклад, Institution, а не User), то при запиті даних із сервера спочатку відбувається перевірка токена авторизації та ролі користувача. Лише після цього система дозволяє доступ до відповідної інформації.

Розробка web-застосунку для автоматизованого розкладу є важливою складовою ефективного функціонування навчального процесу, а захист персональних даних — критично важливим завданням, яке потребує постійного вдосконалення та оновлення безпекових механізмів.

2.4 Проєктування інтерфейсу та структури системи

Проєктування інтерфейсу та структури web-застосунку є важливим етапом у процесі розробки, оскільки саме від нього залежить зручність користування, швидкість взаємодії користувача із системою та стабільність збереження й обробки

даних. З функціональних вимог було визначено кількість користувачів, які повинні бути обов'язково в системі. Серед них:

1. Адміністратор сервісу. Він має змогу перегляду, створення та видалення користувачів з сервісу та повне адміністрування.
2. Представник навчального закладу має такі функції, як вводить дані про дисципліни, викладачів, групи, періоди та аудиторії. Формування розкладу, внесення змін в нього та перегляд звітування про навантаження викладачів в цьому місяці.
3. Користувач (незареєстрований) мають можливість переглядати розклади за допомогою фільтрації за учнем або викладачем. Також вони мають можливість приєднатися до чат-бота Telegram, щоб отримувати сповіщення про розклад на наступний день або зміни в розкладі.

Кожна роль має обмежений доступ до функціоналу системи відповідно до потреб.

Для створення прототипу сервісу використали програмний засіб Figma. Він використовується для роботи з векторною графікою, дозволяючи створювати різноманітні макети, елементи інтерфейсу, зображення, ілюстрації, дизайни та багато інших творчих ідей [18]. Нижче описано основні кроки, використані для прототипування за допомогою Figma.

1. Створення нового проєкту:

   - реєстрація та авторизація в figma: відвідування web-застосунку; створення нового облікового запису або авторизація у існуючий;

   - створення нового файлу: на головній сторінці натиснути кнопку "Create" і обрати дизайн.
2. Додавання фрейму. Слугує контейнером для векторних елементів та зображення.
4. В нижній частині екрану в панелі інструментів обрати "Frame" та створити по розмірам веб-сайту.
3. Додавання елементів UI. Використовуємо інструменти "Rectangle", "Line", "Elipse" та інші для створення основних форм та елементів інтерфейсу.

Тож було розроблено базові прототипи, які демонструють вигляд основних сторінок:

1) головна сторінка web-застосунку (рис.2.1). Це перша сторінка, яку користувач бачить про сервіс. Відображає інформацію про застосунок, про функції та безпеку в додатку.

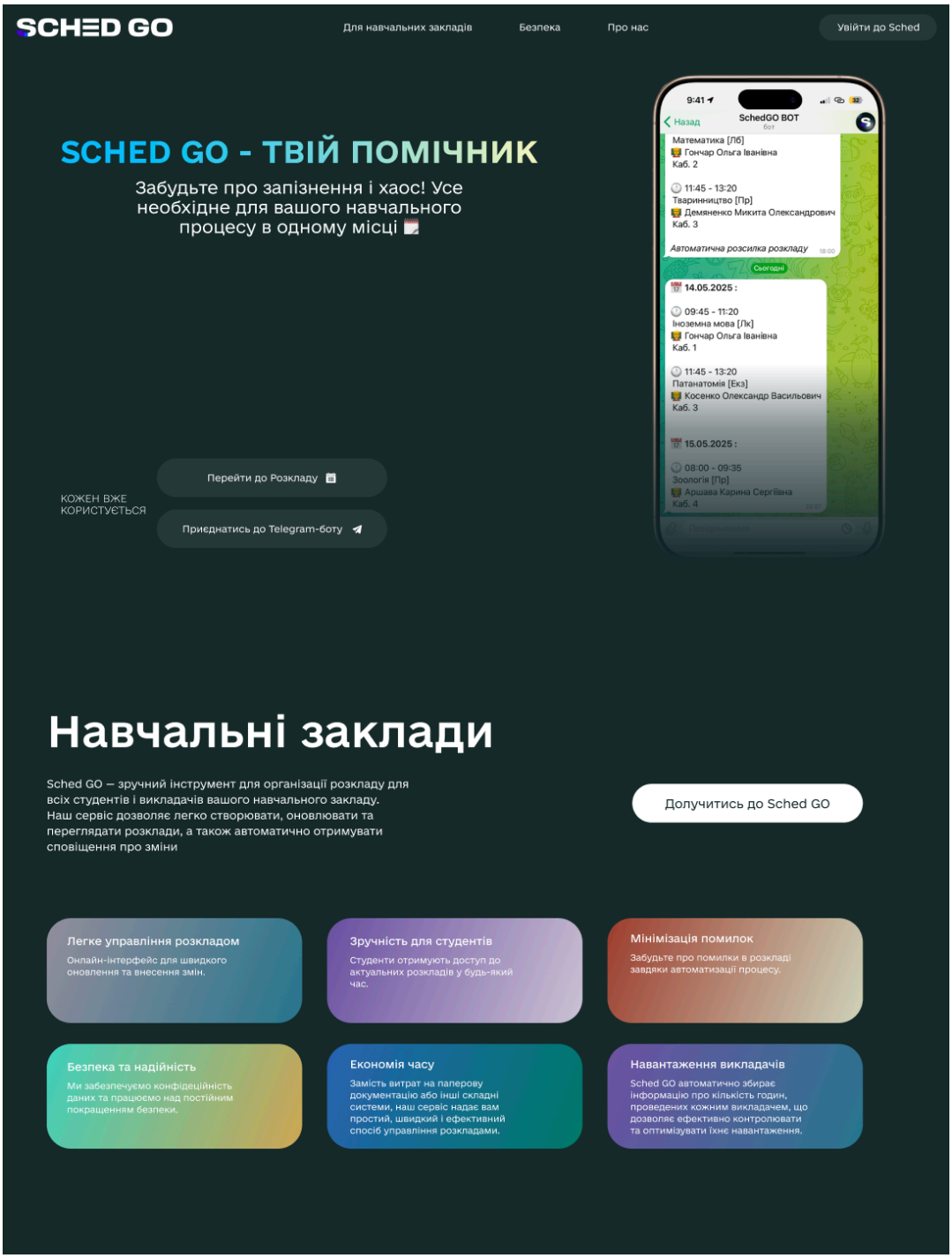


Рис. 2.1. Головна сторінка застосунку

2) сторінка авторизації є екраном взаємодії користувача з системою та виконує функцію контролю доступу до персоналізованого функціоналу (рис. 2.2). Вона забезпечує ідентифікацію користувача та призначення відповідної ролі.

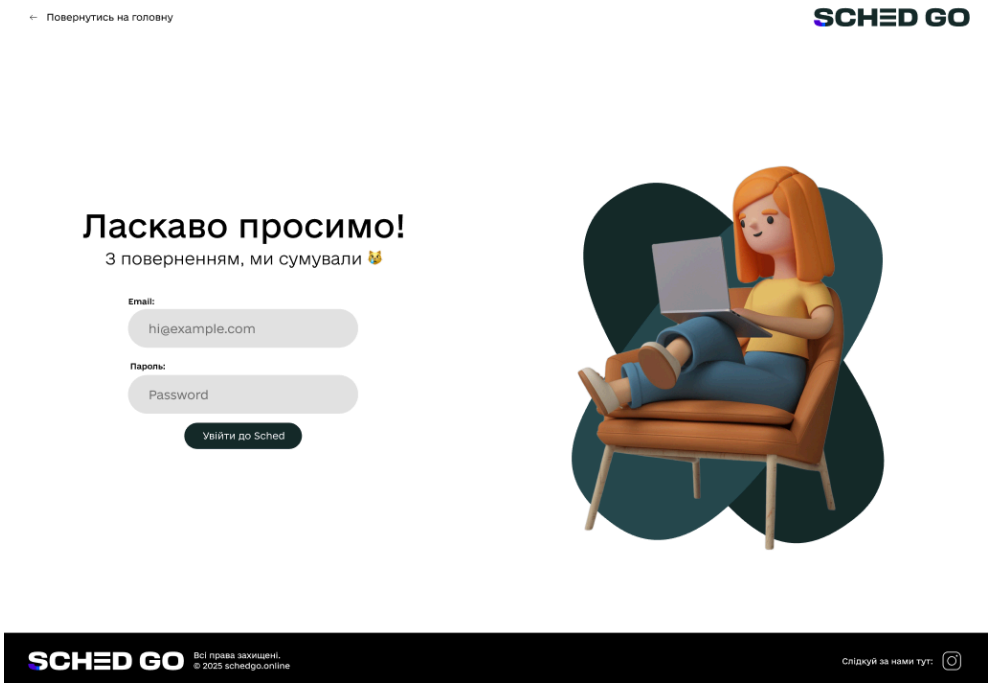


Рис. 2.2. Авторизація в застосунок

3) панель адміністратора є основним інструментом керування сервісом (див. рисунок 2.3). Саме адміністратор має повний доступ до всіх функцій адміністрування. Створення нових користувачів перегляд звітності.

4) панель керування представника навчального закладу – це інтерфейс користувача, який надає доступ до редагування, створення та редагування навчального розкладу (рис.2.4). Також внесення даних про дисципліни, групи, спеціальності, викладачів, періоди пар.

Рис. 2.3. Панель адміністратора

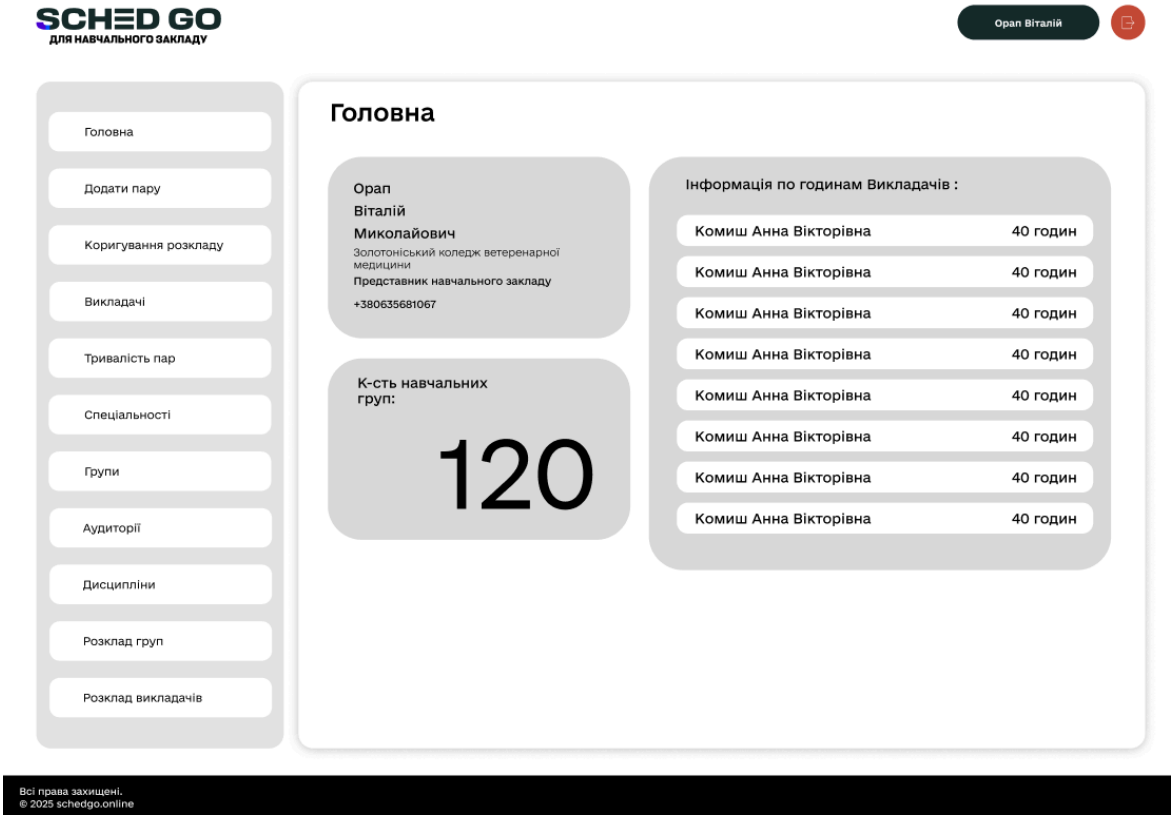


Рис. 2.4. Панель керування представника навчального закладу

База даних є критичним компонентом системи, оскільки зберігає всіх вхідні та вихідні дані, необхідно для формування розкладу. Було розроблено діаграму сутність-зв’язок (рис.2.5), що включає основні сутності:

1) Users (id, firstName, lastName, position, email, password, role);

2) Teachers (id, firstName, lastName, middleName);

3) Specialties (id, name);

4) Schedules (id, subject, teacher, period, lessonType, group, room, dayOfWeek, date, specialty, course);

5) Rooms (id, name);

6) Requests (id, requestType, schoolName, studentCount, contactNumber, email, createdAt);

7) Periods (id, name, startTime, endTime);

8) Groups (id, name, course, specialty);

9) Courses (id, name, specialty).

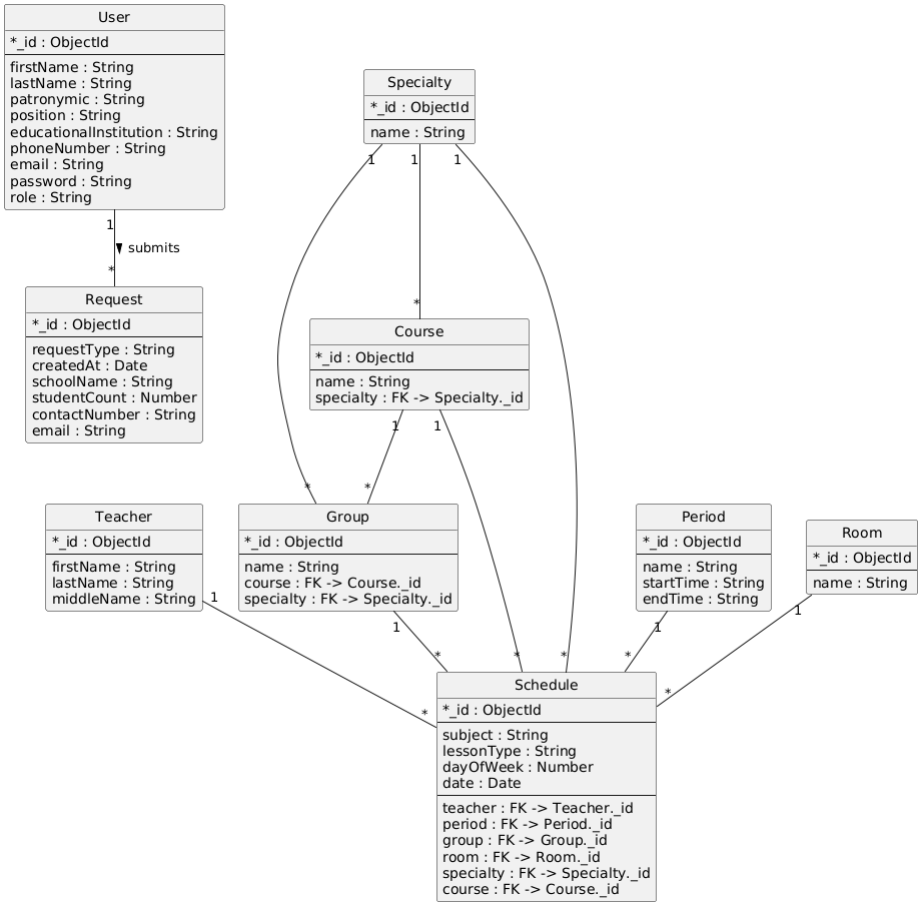


Рис. 2.5. Діаграма сутність-зв’язок

3 РЕАЛІЗАЦІЯ ВЕБ-СИСТЕМИ ТА ЧАТ-БОТА

3.1 Програмна реалізація клієнтської частини

Клієнтська частина web-застосунку було реалізовано за допомогою бібліотеки React – одна з найпопулярніших JavaScript-бібліотек для створення інтерфейсів користувача. Вона дозволяє розробляти масштабовані, динамічні та швидкодіючі застосунки.

1. Створення проєкту на React (рис.3.1).

Для ініціалізації нового проєкту використовується утиліта Create React App, яка автоматично налаштовує структуру, залежності та базові конфігурації [19]. Для створення проєкту виконується команда:

Npx create-react-app schedule-app

де schedule-app – назва папки з майбутнім застосунком.

Після виконання команди автоматично створюється структура проєкту зі стандартними директоріями. Наведено в списку опис:

1) Каталог public/ - статичні ресурси;
2) Каталог src/ – основний код клієнтської частини;
3) Каталог node_modules/ - зберігає бібліотеки та пакети, які були завантаженні через npm. Цей каталог є важливим для забезпечення функціонування;
4) App.js - головний компонент застосунку;
5) Index.js - точка входу.

Рис. 3.1. Приклад структури React проєкту після ініціалізації

Подальшим кроком було розділення інтерфейсу на незалежні компоненти, кожен з яких відповідає за окрему частину функціоналу. Серед основних реалізованих компонентів:

1) Login.js авторизація користувачів, головним це було підключення до API та визначення const Email, Password, які передаються до серверу;

2) Dashboard.js – панель керування. В нашому додатку ми відійшли від створення двох різних панель керування. Реалізовано через JSON Web Token – це компактний, URL-безпечний формат представлення тверджень, який широко використовується для реалізації безпечної автентифікації у веб-застосунках. Він дозволяє передавати дані між сторонами у вигляді об’єкта JSON, зашифрованого для забезпечення довіри. Використання безставних автентифікаційних механізмів, таких як JWT, покращує масштабованість веб-сервісів, особливо в розподілених системах і мікросервісній архітектурі [20]. Щодо обмеження доступу до даних через токен авторизації, якщо роль користувача відрізняється від стандартної (наприклад, Institution замість User), то, як показано на рисунку 3.2, під час звернення до сервера спочатку відбувається перевірка токена авторизації та ролі

користувача. Лише у разі успішної перевірки система надає доступ до відповідної інформації [21].


```
_id: ObjectId('67db4e3e4ad1ba4b5d1ca12a')
firstName : "Марк"
lastName : "Білоус"
patronymic : "Олександрович"
position : "Власник Сервісу"
educationalInstitution : "Державний університет інформаційно-комунікаційних технологій "
phoneNumber : "+380657658967"
email : "bilous@gmail.com"
password : "$2b$10$ggQkGXd.kyka61RD8XszfenABIczZBJ8VsBCk/jfKZ0AuOdX.bl92"
role : "institution"
__v : 0
```


Рис. 3.2 Інформація про користувача з MongoDB

3) StudentSchedule.js – відповідає за відображення розкладу занять для студентів у зручному візуальному форматі. Його основне призначення – надання швидкого доступу до актуальної інформації про час, предмет, викладача та місце проведення занять для конкретної студентської групи. Функціональні можливості – отримання даних розкладу з API на основі ID групи. Відображення розкладу у табличному вигляді (рис.3.3).

Рисунок 3.3. Екранна форма реалізації сторінки сторінки розкладу занять студента

4) TeacherSchedule.js – виконує аналогічну функцію, але орієнтований на викладачів. Він дозволяє переглядати персональний розклад занять, включаючи інформацію про групи, предмет, час та аудиторію (рис.3.4).

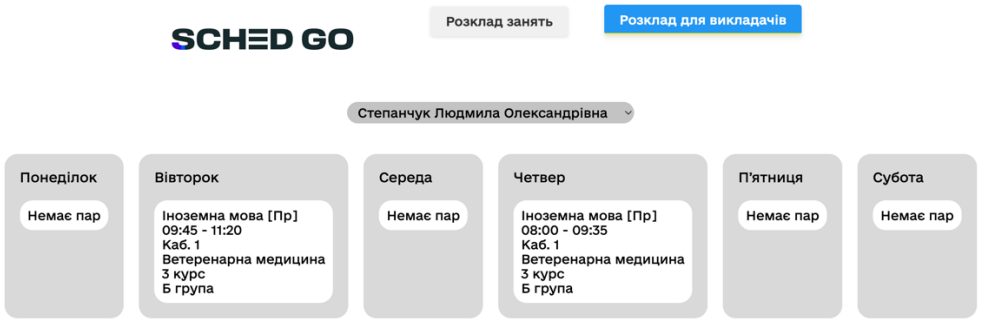


Рис. 3.4. Екранна форма реалізації сторінки розкладу занять викладача

Після наповнення та реалізації сторінок, наступним кроком є налаштування навігації між сторінками реалізована з використанням бібліотеки react-router-dom [22]. Основні маршрути застосунку router.js (див. рисунок 3.5).

```
<Router basename="/">
            <BodyColorWrapper>
                <Routes>
                    <Route path="/" element={<Home />} />
                    <Route path="/auth" element={<Login />} />

                    <Route path="/dashboard" element={
                        <PrivateRoute>
                            <Dashboard />
                        </PrivateRoute>
                    } />

                    <Route path="/schedule" element={
                            <Schedule />
                    } />

                    <Route path="*" element={<Home />} />
                </Routes>
            </BodyColorWrapper>
        </Router>
```

Рис. 3.5. Реалізація навігації

Так же було реалізовано захищеного маршруту в React, тобто компоненту, який обмежує доступ до певних сторінок застосунку лише для автентифікованих користувачів (див. рисунок 3.6).

```
import React, { useEffect, useState } from "react";
import { Navigate } from "react-router-dom";

const PrivateRoute = ({ children }) => {
    const [isAuthenticated, setIsAuthenticated] = useState(null);

    useEffect(() => {
        const token = localStorage.getItem("token");
        setIsAuthenticated(!!token);
    }, []);

    if (isAuthenticated === null) {
        return <div>Загрузка...</div>;
    }

    return isAuthenticated ? children : <Navigate to="/auth" />;
};

export default PrivateRoute;
```

Рис. 3.6. Реалізація приватного маршруту

Він приймає дочірні компоненти – тобто ту сторінку або компонент, до якого повинен мати доступ лише авторизований користувач. Також превіряє наявність токена в localStorage. Якщо токен існує – користувач вважається авторизованим, і йому показується захищена сторінка. А якщо токену немає – відбувається перенаправлення на сторінку авторизації.

Для зручності користувача в інтерфейсі веб-додатку реалізована функція автоматичного визначення та виведення розкладу лише на поточний тиждень — від понеділка до суботи. Це дозволяє уникнути перевантаження інформацією і концентрує увагу на актуальних заняттях.Для визначення початку (startOfWeek) та кінця (endOfWeek) поточного навчального тижня використовується хук useMemo, який гарантує, що обчислення будуть виконані лише один раз при завантаженні компонента. Приклад коду наведено на рисунку 3.7:

```
const { startOfWeek, endOfWeek } = useMemo(() => {
        const today = new Date();
        const currentDay = today.getDay() === 0 ? 7 : today.getDay();
        const start = new Date(today);
        start.setDate(today.getDate() - currentDay + 1);
        start.setHours(0, 0, 0, 0);

        const end = new Date(start);
        end.setDate(start.getDate() + 5);
        end.setHours(23, 59, 59, 999);

        return { startOfWeek: start, endOfWeek: end };
    }, []);
```

Рис. 3.7. Реалізація функції визначення та виведення розкладу лише на поточний тиждень

На цьому рисунку визначається:

- currentDay визначає номер дня тижня (1 — понеділок, 7 — неділя);
- startOfWeek встановлюється на понеділок поточного тижня;
- endOfWeek встановлюється на суботу, тобто розклад охоплює шість навчальних днів.

Коли користувач обирає групу, запускається асинхронна функція fetchSchedule, яка завантажує розклад із сервера за відповідним ідентифікатором групи (див. рисунок 3.8). Після отримання повного розкладу для групи, він фільтрується згідно з датами startOfWeek та endOfWeek.

```
useEffect(() => {
        const fetchSchedule = async () => {
            if (!selectedGroup) return;
            setLoading(true);
            try {
                const response = await axios.get(`${process.env.REACT_APP_API_URL}/schedule/${selectedGroup}`);
                const filteredSchedule = response.data.filter(lesson => {
                    const lessonDate = new Date(lesson.date);
                    return lessonDate >= startOfWeek && lessonDate <= endOfWeek;
                });
                setSchedule(filteredSchedule);
            } catch {
                setError("Помилка загрузки розкладу");
            } finally {
                setLoading(false);
            }
        };

        fetchSchedule();
    }, [selectedGroup, startOfWeek, endOfWeek]);
```

Рис. 3.8. Реалізація функції фільтрації розкладу на поточний тиждень

Цей фрагмент гарантує, що у компонент буде передано лише ті заняття, які мають дату, що потрапляє у визначений інтервал — поточний тиждень.

3.2 Реалізація серверної частини

Серверна частину web-застосунку була реалізована за допомогою платформи Node.js та фреймворку Express.js. Такий стек технологій було обрано через його високу продуктивність, простоту налаштування та широку підтримку з боку спільноти. Ініціалізація проєкту - це перший крок до створення нового проєкту за допомогою менеджера пакетів npm командою npm init –y.

Ця команда створює package.json, у якому зберігається інформація про залежності, скрипти запуску та інші параметри проєкту. Далі командою npm install express встановлюємо фреймворк express.

Наступним кроком створення основного файлу сервера, який є точкою входу до серверного застосунку. У ньому відбувається конфігурація сервера.
Організація структури виглядає наступним чином (див. рисунок 3.9).

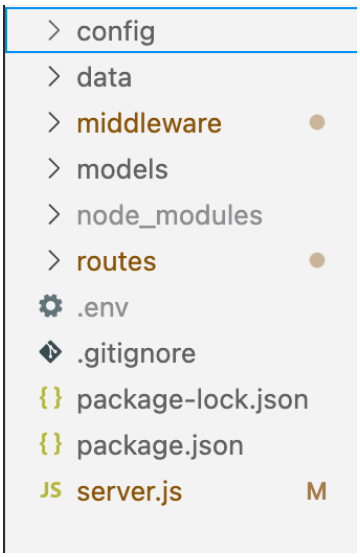


Рис. 3.9. Структура сервера

/Config- підключення бази даних до серверу, для зберігання інформації використовується база даних MongoDB;
/data – статичні дані про дні неділі;

/middleware – проміжне ПЗ, авторизація через JWT;

/models – головні моделі для АПІ, а саме Course, Group, Period, Request, Room, Schedule, Specialty, Teacher, User;

/routes – маршрути застосунку.

Створення маршрутів головне для АПІ, так як клієнтська частина повинна мати шлях до отримання інформації. На рисунку 3.10 наведено маршрути всього серверу.

```
app.use("/api/rooms", authMiddleware, require("./routes/roomsRoutes"));
app.use("/users",authMiddleware, usersRoutes)
app.use('/api/schedule', scheduleRoutes);
app.use('/api/groups', groupRoutes);
app.use('/specialties', specialtyRoutes);
app.use('/api/periods', authMiddleware, periodRoutes);
app.use('/auth', authRoutes);
app.use("/courses", courseRoutes);
app.use('/api', weekdayRoute);
app.use("/api/requests", authMiddleware, requestRoutes);
app.use('/teachers', teacherRoutes);
```

Рис. 3.10. Маршрути серверної частини застосунку

У даному фрагменті серверного коду реалізовано підключення маршрутів для обробки HTTP-запитів у web-застосунку на Node.js з використанням фреймворку Express.js. Кожен маршрут відповідає за окремий функціональний блок системи.

Шлях /api/rooms використовує проміжне ПЗ authMiddleware для авторизації та обробляє запити, пов’язані з навчальними приміщеннями (аудиторіями). Для маршруту /users також встановлено авторизаційний middleware, що гарантує доступ лише для зареєстрованих користувачів при взаємодії з обліковими записами. Ресурс /api/schedule відповідає за операції з розкладом занять, включаючи перегляд, фільтрацію та оновлення.

Маршрути /api/groups та /specialties забезпечують доступ до інформації про академічні групи та спеціальності відповідно. Авторизація на цих маршрутах тимчасово вимкнена або передбачена на рівні окремих дій. Шлях /api/periods, який

захищено middleware, обслуговує дані про навчальні періоди — семестри або модулі.

Маршрут /auth надає точки входу для автентифікації, зокрема вхід, реєстрацію та отримання токенів доступу. Підключення до /courses дозволяє оперувати навчальними дисциплінами. Шлях /api, що веде до weekdayRoute, надає функціонал, пов’язаний з календарними днями тижня.

Маршрут /api/requests обробляє запити, наприклад, на зміну розкладу, і для нього також застосовується перевірка авторизації. Нарешті, шлях /teachers відповідає за роботу з інформацією про викладачів — перегляд, редагування, призначення курсів тощо.

Загальна структура маршрутів підтримує логіку REST API та забезпечує чіткий розподіл відповідальності між модулями системи. Це сприяє масштабованості, повторному використанню компонентів і підвищенню безпеки web-застосунку.

3.3 Реалізація чат-бота

У даному розділі розглянуто реалізацію програмного забезпечення, призначеного для автоматизованого отримання та обробки розкладу занять у вигляді Telegram-бота. Обрано мову програмування Python та бібліотеку aiogram як основу розробки завдяки їхній гнучкості, асинхронній природі та активній спільноті розробників.

1. Створення проєкту та налаштування середовища.

   Розробка розпочиналась зі створення окремого ізольованого середовища для встановлення залежностей. Для цього використовувалась стандартна бібліотека venv, що дозволяє уникати конфліктів між пакетами:

   python -m venv .venv

   source .venv/bin/activate

   pip install aiogram aiohttp

2. Архітекура проєкту.

Розробка бота виконувалась із дотриманням принципів модульності та чіткого розділення відповідальностей. Структура проєкту виглядає наступним чином:

/database/ – моделі даних (Користувачів чат-бота)

/handlers/ – обробники команд користувача

/services/

├── api_client.py – взаємодія з бекенд-сервером через HTTP-запити

├── config.py – зчитування налаштувань із .env

└── main.py – ініціалізація та запуск бота

requirements.txt – файл із залежностями

3. Реалізація автоматичних сповіщень у чат-боті.

Однією з найзручніших функцій, реалізованих у чат-боті, є автоматична розсилка розкладу занять на завтра**.** Такий функціонал значно покращує взаємодію користувача із системою, підвищує залученість та виконує роль персонального помічника для студентів та викладачів. Щоденно о 15:00 бот надсилає кожному користувачу, який завершив реєстрацію (вибрав роль, спеціальність, курс і групу), детальний розклад занять на наступний день. Це дозволяє уникати пропущених пар, планувати день заздалегідь та зменшити потребу вручну запитувати розклад.

Для організації щоденних задач було використано бібліотеку apscheduler, а саме клас AsyncIOScheduler, який дозволяє інтегрувати планувальник із асинхронним циклом подій asyncio.

from apscheduler.schedulers.asyncio import AsyncIOScheduler

Функція broadcast_tomorrow_schedule викликається щоденно та виконує наступні кроки:

1. Отримує список усіх зареєстрованих користувачів.
2. Перевіряє, чи кожен з них має обрану групу.
3. Завантажує розклад групи на наступний день.
4. Якщо заняття є, форматує повідомлення.

5. Надсилає повідомлення кожному користувачеві.

Реалізуємо функцію розсилки для чат бота:

```
async def broadcast_tomorrow_schedule(app):
users = await get_all_user_ids()
tomorrow = datetime.now(ZoneInfo("Europe/Kyiv")).date() + timedelta(days=1)

for uid in users:
    user = await get_user(uid)
    if not user or not user.get("group_id"):
        continue

    lessons = await get_schedule(user["group_id"])
    tomorrow_lessons = [
        lesson for lesson in lessons
        if parse_lesson_date(lesson["date"]).date() == tomorrow
    ]

    if not tomorrow_lessons:
        continue

    message = format_schedule(tomorrow_lessons)
    message += "\n\n<em>Автоматична розсилка розкладу</em>"

    try:
        await app.bot.send_message(chat_id=uid, text=message, parse_mode="HTML")
        logging.info(f"Розклад надіслано користувачу {uid}")
    except Exception as e:
        logging.error(f"Не вдалося надіслати {uid}: {e}")
```

Перевагами цієї функції є:

1) Асинхронна архітектура**:** не блокує основний цикл роботи бота.
2) Часова зона**:** завдяки ZoneInfo("Europe/Kyiv") час виконання точно відповідає локальному.
3) Масштабованість**:** В подальшому можна додати інші типи розсилок (нагадування про іспити, збори тощо).
4) Журнали (логи)**:** уся активність зберігається у logging, що дозволяє відслідковувати помилки розсилки.

Автоматична розсилка — це приклад того, як бот стає не лише інтерфейсом запиту інформації, а й активним помічником, який сам ініціює контакт із користувачем. Це підвищує ефективність освітнього процесу та зручність користування ботом.

3.4 Хостинг застосунку

Для розгортання системи було обрано хмарний хостинг DigitalOcean, оскільки він забезпечує стабільну інфраструктуру, просту інтеграцію з GitHub та підтримку різних технологій (Node.js, React, Python) [23].

Усі три частини системи (frontend на React, backend на Node.js та Telegram-бот на Python) були реалізовані як окремі сервіси в середовищі DigitalOcean App Platform — це платформа автоматизованого деплою, яка дозволяє легко керувати додатками через вебінтерфейс. Для кожного з трьох проєктів було налаштовано підключення до відповідних гілок GitHub-репозиторію. App Platform автоматично відстежує зміни у репозиторії — тобто, коли відбувається push (завантаження нових комітів), система автоматично пересобирає і деплоїть додаток без додаткових дій з боку розробника. Це забезпечує зручний CI/CD-процес (Continuous Integration / Continuous Deployment) значно пришвидшує розгортання нових функцій.

Для сервісу було зареєстровано доменне ім’я**:** schedgo.online.

Цей домен використовується для основного вебінтерфейсу (React-додатку), що доступний для користувачів. Окрім цього, був створений піддомен для серверної частини (API), що працює на Node.js: 8x2y6al2clfy9c8g.schedgo.online.

Використання піддомену дозволяє розділити веб-інтерфейс та серверну логіку, спростити налаштування CORS і забезпечити масштабованість проєкту в майбутньому.

Таким чином, завдяки використанню DigitalOcean App Platform у поєднанні з GitHub, вдалося реалізувати ефективну, масштабовану та легко підтримувану інфраструктуру для застосунку автоматизованого розкладу.

4 ТЕСТУВАННЯ

4.1 Тестування серверної частини застосунку

Для перевірки роботи серверної частини застосунку було виконано модульне тестування основних маршрутів та моделей, що реалізовані на базі Node.js, Express і MongoDB. Тестування проводилося із використанням бібліотеки Jest разом з Supertest для емуляції HTTP-запитів, а також mongodb-memory-server для створення ізольованого середовища бази даних без потреби у підключенні до реального сервера.

Метою було впевнитися, що основні операції створення, отримання та видалення даних у моделях Teacher, User, Specialty працюють очікувано, а також що маршрут отримання дня тижня (/api/weekday) коректно обробляє як валідні, так і невалідні запити. Перевагами модульного тестування є:

1) ізольованість – кожен тест виконується у незалежному середовищі, що дозволяє легко ідентифікувати джерело помилки;
2) автоматизація – тести можна запускати в автоматичному режимі після кожної зміни коду;
3) підвищення якості коду – регулярне написання тестів сприяє створенню надійного, масштабованого та зрозумілого коду;
4) запобігання регресії – дозволяє впевнитися, що нові зміни не призводять до порушення вже реалізованого функціоналу.

1. Було створено окремі блоки тестування для кожного з основних API:

Teachers API**:**

1) POST /teachers – перевіряє створення викладача.
2) GET /teachers/with-hours – перевіряє отримання викладачів із підрахованими годинами.
3) DELETE /teachers/:id – перевіряє видалення викладача та обробку випадку з неіснуючим ID.

Users API**:**

4) GET /users – перевіряє отримання списку користувачів.

Specialty API**:**

5) GET /specialties – перевірка отримання всіх спеціальностей.
6) POST /specialties – перевірка створення нової спеціальності.

Weekday API**:**

7) GET /api/weekday?date=2025-05-27 – перевірка визначення дня тижня для переданої дати.
8) GET /api/weekday – перевірка обробки запиту без параметра date.

Перед кожним тестом запускається інстанс у пам'яті MongoMemoryServer, який забезпечує повну ізольованість та чистоту тестового середовища. Після кожного тесту колекції очищуються, а після завершення всіх тестів – підключення до бази закривається. Для запуску тестів використовується наступна команда:

Npm test

```
mark@Noutbuk-Mark schedule-api % npm test

> schedule-api@1.0.0 test
> jest

 PASS  tests/api.test.js
  Teachers API
    ✓ POST /teachers – створити викладача (147 ms)
    ✓ GET /teachers/with-hours – отримати викладачів з годинами (8 ms)
    ✓ DELETE /teachers/:id – видалити викладача (12 ms)
    ✓ DELETE /teachers/:id – видалити неіснуючого викладача (4 ms)
  Users API
    ✓ GET /users – отримати список користувачів (81 ms)
  Specialty API
    ✓ GET /specialties – отримати всі спеціальності (6 ms)
    ✓ POST /specialties – додати спеціальність (5 ms)
  Weekday API
    ✓ GET /api/weekday?date=2025-05-27 – отримати день тижня (5 ms)
    ✓ GET /api/weekday без дати – повертає помилку (4 ms)

Test Suites: 1 passed, 1 total
Tests:       9 passed, 9 total
Snapshots:   0 total
```

Рис. 4.1. Успішне проходження усіх тестів

Всі тести було успішно пройдено зі статусом PASS, що свідчить про правильну реалізацію базового функціоналу API. Це підтверджує, що система стабільно виконує основні операції з даними та коректно реагує на типові та виняткові ситуації.

ВИСНОВОК

У результаті виконання роботи було створено web-застосунок для автоматизації процесу формування та управління розкладом навчальних занять у закладах освіти. Розробка здійснювалася з використанням JavaScript, Node.js, React і MongoDB, а також Telegram-бота на Python, Figma для прототипування інтерфейсів користувача. Система дозволяє зменшити витрати часу на створення розкладу, мінімізувати людський фактор та забезпечити зручний доступ до актуальної інформації для студентів і викладачів.

У ході роботи було виконано наступні задачі:

1. Проведено аналіз предметної області: визначено основні потреби навчального закладу у сфері планування занять; описано специфіку взаємодії студентів, викладачів і адміністрації з розкладом; проаналізовано існуючі рішення, виявлено їх переваги й недоліки; на основі цього сформульовано вимоги до функціоналу системи, зокрема:

1) система повинна дозволяти представнику навчального закладу створювати, видаляти та редагувати основну інформацію (час пар, групи, курс, спеціальності, викладачі);
2) користувач повинен мати змогу переглядати розклад за групою, викладачем;
3) формування та фільтрація розкладу за параметрами (група, аудиторія, викладач);
4) система повинна надати можливість пошуку розкладу за ключовими параметрами;
5) користувач повинен мати змогу додати заявку на підключення системи до навчального закладу;
6) система повинна дозволяти адміністратору створювати нових користувачів;
7) створення Telegram-бота з підтримкою реєстрації студентів, запитів до розкладу та автоматичною розсилкою на щоденній основі.

2. Обрано технології розробки:

Середовища розробки: VS Code (для фронтенду та бекенду), PyCharm (для Telegram-бота), мови програмування: JavaScript, Python, фреймворки та бібліотеки: React (інтерфейс користувача), Node.js + Express (бекенд), Mongoose (ORM для MongoDB), базу даних: MongoDB, систему контролю версій: Git (GitHub), інструмент для візуалізації: Figma (прототипи інтерфейсу).

3. Розроблено прототип системи: створено архітектурну структуру застосунку, побудовано моделі сутностей (група, курс, спеціальність, заняття, викладач тощо), розроблено REST API для взаємодії з фронтендом та ботом.
4. Реалізовано функціональний веб-застосунок: створено інтерфейс для адміністратора, налаштовано маршрути API, реалізовано Telegram-бот з інтерактивною клавіатурою, протестовано основні API-маршрути із використанням Jest, Supertest та in-memory MongoDB.

Система успішно пройшла модульне тестування, що підтвердило стабільність її роботи. Telegram-бот забезпечує зручну комунікацію з користувачами, а веб-інтерфейс дозволяє легко створювати та редагувати розклад.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Білоус М.О., Шевченко С.М. Розробка web-застосунку для автоматизації розкладу занять у закладі освіти // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна, С. 64-66.
2. Білоус М.О. Захист інформації у web-застосунку для автоматизації розкладу занять у закладі освіти // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених «Інформаційні технології – 2025», 15.05.2025, КСУ імені Бориса Грінченка, Київ, Україна, С.263-265.

ПЕРЕЛІК ПОСИЛАНЬ

1. Statista – Most popular global mobile messenger apps of February 2025, based on number of monthly active users – statista.com – [Електронний ресурс] – URL: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/ (date of access 10.05.2025)
2. European Commission/EACEA/Eurydice. (2019). *Digital education at school in Europe*. Luxembourg: Publications Office of the European Union. URL: https://doi.org/10.2797/763 (date of access 11.05.2025)
3. Поглибляцький НВК – Основні вимоги до складання розкладу - pohybliak.edukit.ck.ua – [Електронний ресурс] – URL: http://pohybliak.edukit.ck.ua/navchaljnij_proces/vimogi_do_skladannya_rozkladu/ (date of access 11.05.2025)
4. Хайтан О.М. (2020). Автоматизація розкладу навчальних курсів університету. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки , 1 (2), 58–66. https://doi.org/10.32838/2663-5941/2020.2-1/09 (date of access 11.05.2025)
5. Снитюк В.Є., Сіпко О.М. Технологія еволюційного формування розкладів у закладах вищої освіти. Київ: Видавець ФОП Піча Ю.В., 2022. – 136 с.
6. Almantsri, Ahmed & Alhamrouni, Mohamed & E'atiwa, Aisha. (2022). An automated study schedule via genetic algorithm. 10.13140/RG.2.2.17770.39363.
7. Aida-zade, Kamil & Ismibayli, Reshad & Rzayeva, Sona. (2024). Automated Schedule System for Universities under the Bologna Education Process. Cybernetics and Computer Technologies. 75-90. 10.34229/2707-451X.24.1.6.
8. Mustapa, Muhammad & Salahuddin, Lizawati & Hashim, Ummi. (2022). Automated Study Plan Generator using Rule-based and Knapsack Problem. International Journal of Advanced Computer Science and Applications. 13. 10.14569/IJACSA.2022.0130847.

9. PYPL PopularitY of Programming Language - pypl.github.io – [Електронний ресурс] – URL: https://pypl.github.io/PYPL.html (date of access 14.05.2025)
10. Посібник: знайомство з React. – uk.legacy.reactjs.ord – [Електронний ресурс] – Режим доступу: https://uk.legacy.reactjs.org/tutorial/tutorial.html (date of access 14.05.2025)
11. JavaScript - jetbrains.com – [Електронний ресурс] – URL: https://www.jetbrains.com/lp/devecosystem-2023/javascript/ (date of access 14.05.2025)
12. Most used web frameworks among developers worldwide, as of 2024 - www.statista.com – [Електронний ресурс] – URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (date of access 13.05.2025)
13. Jeffrey Erickson. What Is MongoDB? An Expert Guide. – [Електронний ресурс] – oracle.com– URL: https://www.oracle.com/ua/database/mongodb/ (date of access 14.05.2025)
14. Бондаренко О., Ушкаленко І. (2017). Безпека web-додатків: актуальні проблеми та їх аналіз. Формування ринкової економіки в Україні. Вип. 38., 28-36.
15. Боскін О.О., Корніловська Н.В., Поліщук В.М., Сарафаннікова Н.В.(2023). Безпека веб-додатків та хакерські атаки. Вісник Херсонського національного технічного університету, № 3(86), 83-92. https://doi.org/10.35546/kntu2078-4481.2023.3.11 (date of access 14.05.2025)
16. Що таке Python? – acode.com.ua – [Електронний ресурс] – URL: https://acode.com.ua/intro-python/ (date of access 14.05.2025)
17. Хешування паролів: використання солі та bcrypt - drukarnia.com.ua - [Електронний ресурс]. – URL: https://drukarnia.com.ua/articles/kheshuvannya-paroliv-vikoristannya-soli-ta-bcrypt-fsme- (date of access 11.05.2025)
18. Що таке Figma та для кого вона потрібна – cases.media – [Електронний ресурс] – URL: https://cases.media/en/article/sho-take-figma-ta-dlya-kogo-

vona-potribna?srsltid=AfmBOoowgN5TBC73d3Loejhx9GpnHqnTwLRL30af_CKA0jGArstRfPrT (date of access 16.05.2025)

19. Створюємо новий React-додаток – uk.legacy.reactjs.org – [Електронний ресурс] – URL: https://uk.legacy.reactjs.org/docs/create-a-new-react-app.html (date of access 17.05.2025)
20. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston : Addison-Wesley, 2020. – 560 p.
21. Білоус М.О., Шевченко С.М. Розробка web-застосунку для автоматизації розкладу занять у закладі освіти // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.25, ДУІКТ, Київ, Україна, 652 с.
22. Banks A., Porcello E. Learning React / Alex Banks, Eve Porcello. – 3rd ed. – Sebastopol : O'Reilly Media, 2023. – 450 с. – ISBN 978-1-0981-3120-2.
23. DigitalOcean – надійна хмарна інфраструктура для розробників – brights.io - [Електронний ресурс]. – URL: https://brights.io/ua/technology/digital-ocean (date of access 20.05.2025)
24. Sommer P. Modern Software Development. – Berlin: Springer, 2020. – 312 с.
25. Bejsovec O. Learning JavaScript Development with Visual Studio Code / O. Bejsovec. – Birmingham: Packt Publishing, 2020. – 280 с.
26. Quigley D. Mastering PyCharm / D. Quigley. – Birmingham: Packt Publishing, 2021. – 350 с.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка WEB-застосунку для автоматизації розкладу занять у закладі освіти

Виконав студент 4 курсу
групи ПД-41
Білоус Марк Олександрович
Керівник роботи
к.п.н., доц., доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – підвищення ефективності управління навчальним процесом за рахунок автоматизації розкладу занять у закладі освіти.
- **Об'єкт дослідження** – процес автоматизованого складання розкладу занять у закладі освіти.
- **Предмет дослідження** – web-застосунок для автоматизації розкладу занять у закладі освіти.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз науково-технічних джерел щодо вимог до розробки web-застосунків.
2. Проаналізувати існуючі рішення для автоматизації розкладу занять.
3. Здійснити огляд та аналіз існуючих рішень, що використовують для автоматизації розкладу у навчальних закладах.
4. Сформувати вимоги до розробки клієнт-серверного застосунку для автоматизації розкладу занять у закладі освіти.
5. Спроєктувати архітектуру клієнт-серверного застосунку.
6. Розробити клієнт-серверний застосунок для автоматизації навчального розкладу у закладі освіти.
7. Провести тестування клієнт-серверного застосунку.

3

АНАЛІЗ АНАЛОГІВ

Показник	МКР	UniTime	SchedGO
Наявність веб-платформи:	+	-	+
Підтримка складних сценаріїв	+	+	+
Легкість налаштування	+	-	+
Сповіщення про зміни в розкладі	-	-	+
Друк розкладу	+	+	+
Звітність по загрузці викладачів	+	+	+
Основна цільова аудиторія	ЗВО з великою кількістю студента	ЗВО з великою кількістю студента	Всі навчальні заклади

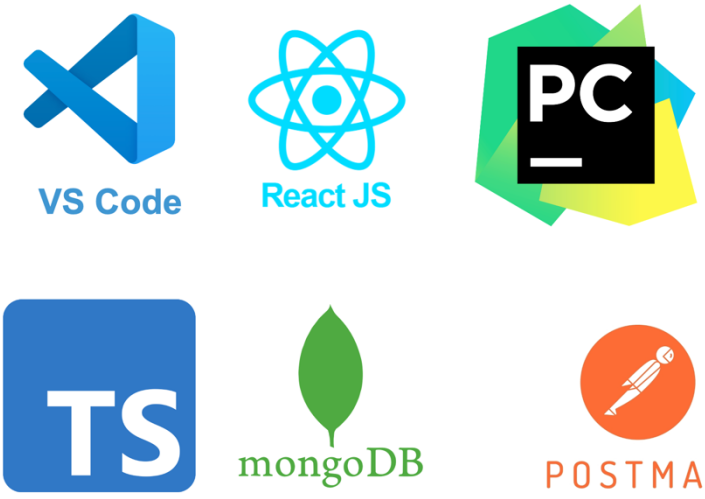
4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тип вимоги	Опис вимоги
Функціональна	Система повинна дозволяти представнику навчального представника створювати, видаляти та редагувати основну інформацію (час пар, групи, курс, спеціальності, викладачі)
Функціональна	Користувач повинен мати змогу переглядати розклад за групою, викладачем
Функціональна	Система повинна надати можливість пошуку розкладу за ключовими параметрами
Функціональна	Користувач повинен мати змогу додати заявку на підключення системи до навчального закладу
Функціональна	Система повинна дозволяти адміністратору створювати нових користувачів
Нефункціональна	Інтерфейс повинен бути адаптивним і коректно відображатись на всіх пристроях
Нефункціональна	Система повинна забезпечувати захист персональних даних користувачів
Нефункціональна	ПЗ повинно бути сумісним з основними сучасними браузерами

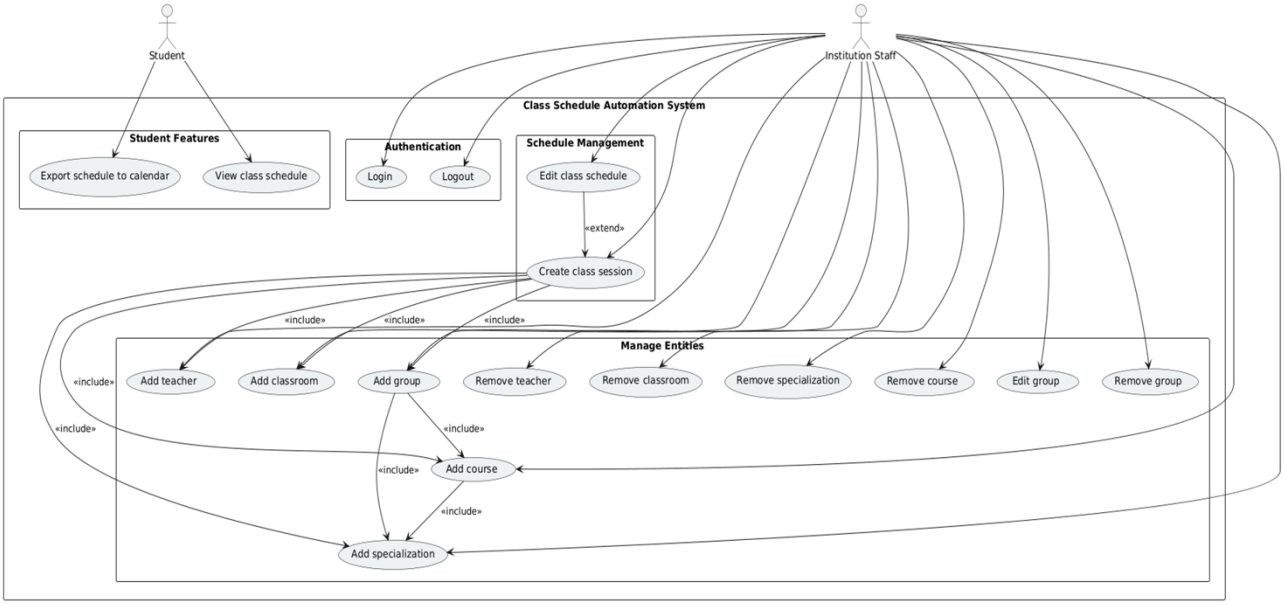
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

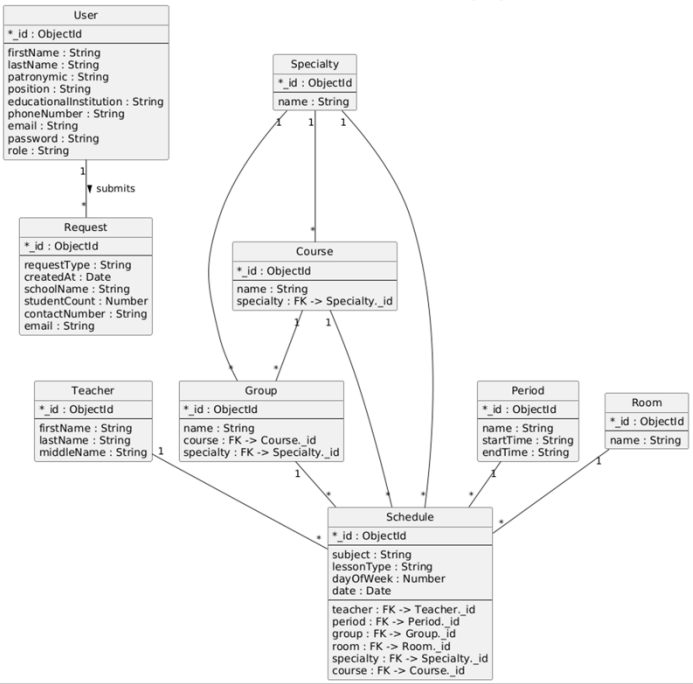


6

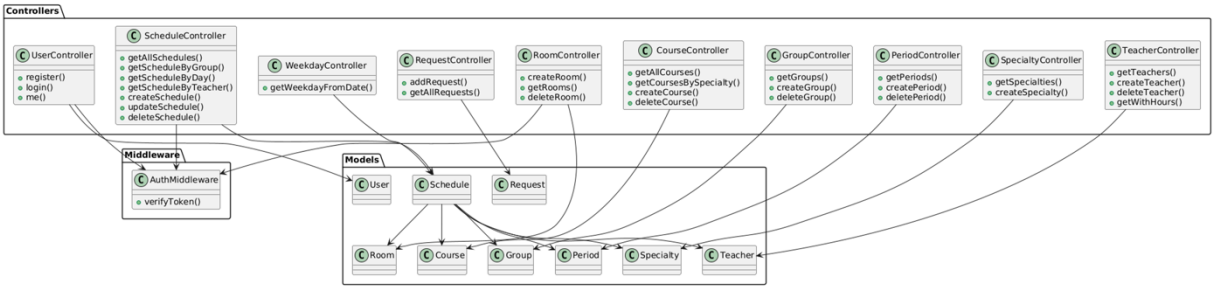
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



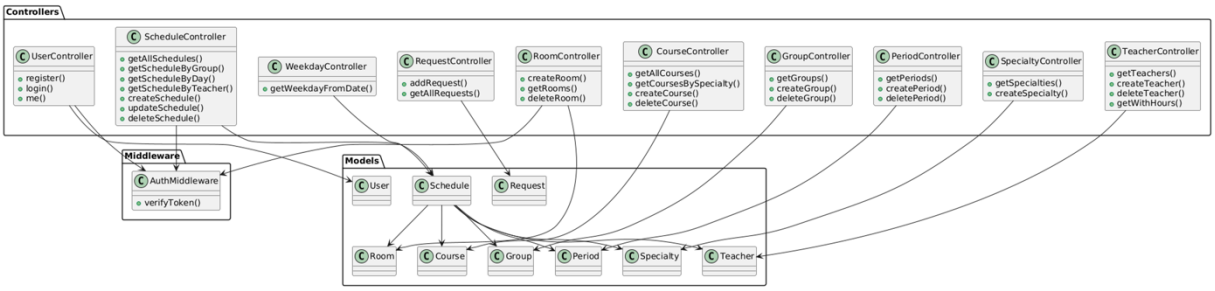
СТРУКТУРА БАЗИ ДАНИХ



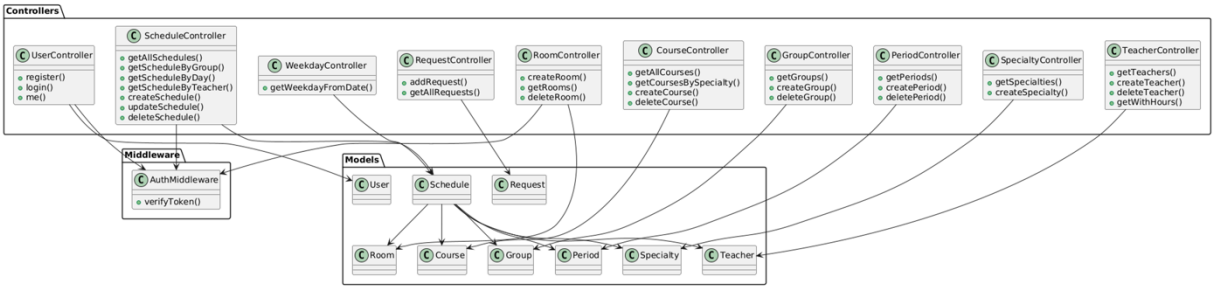
ДІАГРАМА КЛАСІВ ОСНОВНОЇ ЛОГІКИ



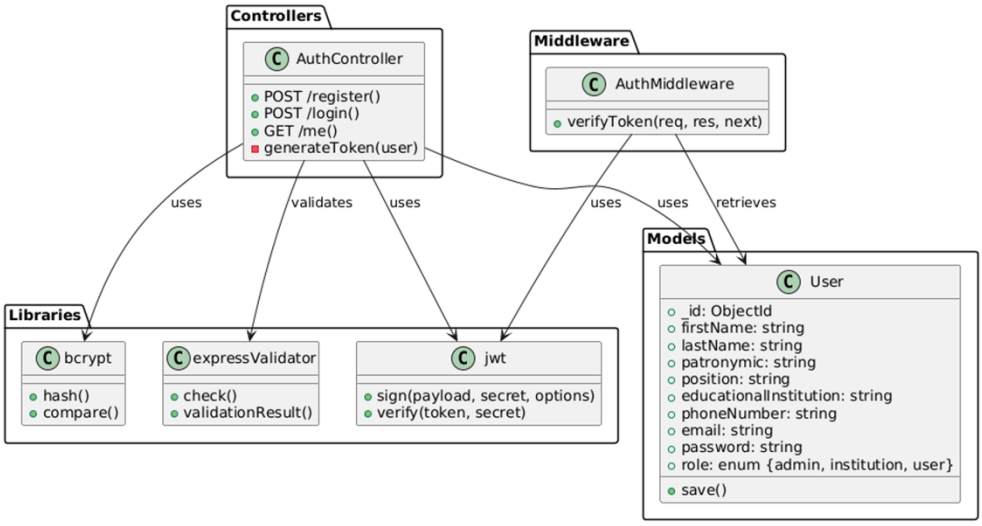
ДІАГРАМА КЛАСІВ ОСНОВНОЇ ЛОГІКИ



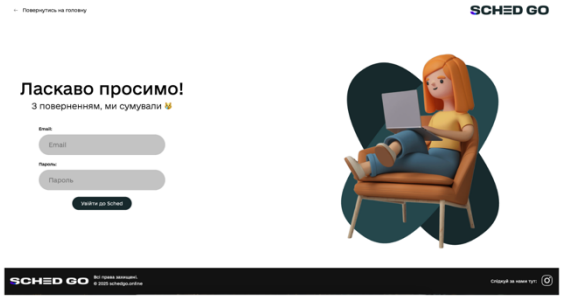
ДІАГРАМА КЛАСІВ ОСНОВНОЇ ЛОГІКИ



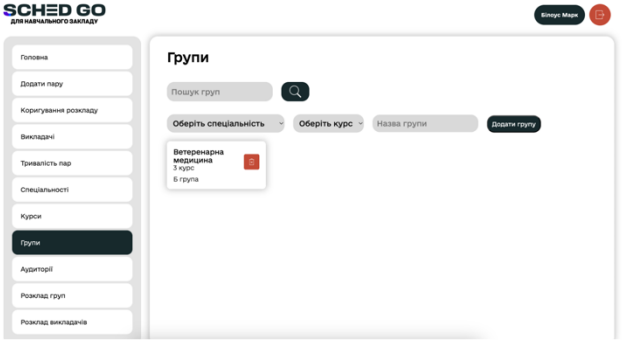
ДІАГРАМА КЛАСІВ АВТОРИЗАЦІЇ



ЕКРАННІ ФОРМИ



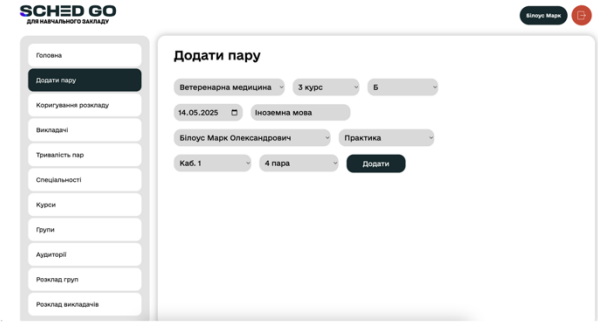
Авторизація



Додавання груп до бази даних

11

ЕКРАННІ ФОРМИ



Додавання пар

Коригування розкладу

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Білоус М.О., Шевченко С.М. Розробка web-застосунку для автоматизації розкладу занять у закладі освіти // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.25, ДУІКТ, Київ, Україна, с/ 652

2. Білоус М.О. Захист інформації у web-застосунку для автоматизації розкладу занять у закладі освіти // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених «Інформаційні технології – 2025», 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна, 362 с. ISSN: 2664-2638.

ВИСНОВКИ

1. Проведено аналіз науково-техічної літератури щодо сучасних вимог до web-застосунків, а саме: розглянуто вимоги до безпеки, продуктивності, масштабованості, а також принципи UI/UX.
2. Проаналізовано існуючі системи автоматизації розкладу занять: МКР та UniTime, визначено їхні переваги, серед недоліків виділено відсутність веб-платформи та відсутність сповіщення про зміни в розкладі.
3. Сформовано функціональні та нефункціональні вимоги до клієнт-серверного застосунку автоматизації розкладу занять у закладі освіти. Серед основних вимог: підтримка авторизації, зручний інтерфейс для створення та редагування розкладу, багатокористувацький доступ, можливість перегляду розкладу в реальному часі тощо
4. Досліджено інструменти та технології для розробки клієнт-серверного застосунку автоматизації розкладу занять у закладі освіти, а саме: React/Angular/Vue для фронтенду, Node.js/Django для бекенду, MongoDB/PostgreSQL для зберігання даних.
5.Спроєктовано архітектуру системи з урахуванням ролей користувачів та розмежування доступу.
6. Розроблено веб-застосунок на основі обраних технологій і з урахуванням зазначених вимог для автоматизації рокладу занять у закладі освіти.
7. Проведено функціональне та нефункціональне тестування системи, яке показало, що застосунок відповідає зазначеним вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

МОДУЛІВ Вихідний код файлу React

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/ManageCourses.css";
import "../styles/CreateRoom.css";
import Delete from "../assets/svg/Remove.svg";
import Search from "../assets/svg/Search.svg";

const ManageCourses = () => {
   const [name, setName] = useState("");
   const [selectedSpecialty, setSelectedSpecialty] = useState("");
   const [specialties, setSpecialties] = useState([]);
   const [courses, setCourses] = useState([]);
   const [search, setSearch] = useState("");
   const [searchQuery, setSearchQuery] = useState("");
   const [error, setError] = useState("");
   const [loading, setLoading] = useState(false);

   useEffect(() => {
      const fetchData = async () => {
         try {
            const token = localStorage.getItem("token");
            const apiUrl = process.env.REACT_APP_API_URL;

            const [specialtiesResponse, coursesResponse] = await
Promise.all([
               axios.get(`${apiUrl}/specialties`,),
               axios.get(`${apiUrl}/courses`, {
                  headers: { Authorization: `Bearer ${token}` },
               }),
            ]);

            setSpecialties(specialtiesResponse.data);
            setCourses(coursesResponse.data);
         } catch (error) {
            console.error("Помилка загрузки курсів.", error);
            setError("Помилка загрузки курсів.");
         } finally {
            setLoading(false);
         }
      };

      fetchData();
   }, []);

   const handleCreateCourse = async (e) => {
      e.preventDefault();
      if (!name.trim() || !selectedSpecialty) return;
      setLoading(true);
      setError("");

      try {
         const token = localStorage.getItem("token");
         const apiUrl = process.env.REACT_APP_API_URL;
         const response = await axios.post(
            `${apiUrl}/courses`,
            { name, specialty: selectedSpecialty },
            { headers: { Authorization: `Bearer ${token}` } }
         );

         setCourses([...courses, response.data]);
         setName("");
      } catch (error) {
         console.error("Помилка створення курсу.", error);
         setError("Помилка створення курсу.");
```

```
import React, { useState, useEffect, useRef } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import moment from "moment";
import "../styles/CreateLesson.css"
import "../styles/CreateRoom.css";

const CreateLesson = () => {
  const navigate = useNavigate();
  const [specialties, setSpecialties] = useState([]);
  const [courses, setCourses] = useState([]);
  const [groups, setGroups] = useState([]);
  const [teachers, setTeachers] = useState([]);
  const [classrooms, setClassrooms] = useState([]);
  const [date, setDate] = useState("");
  const [selectedPeriod, setSelectedPeriod] = useState("");
  const [selectedWeekDayId, setSelectedWeekDayId] =
useState("");
  const [lessonType, setLessonType] = useState("");
  const [selectedSpecialty, setSelectedSpecialty] = useState("");
  const [selectedCourse, setSelectedCourse] = useState("");
  const [selectedGroup, setSelectedGroup] = useState("");
  const [selectedTeacher, setSelectedTeacher] = useState("");
  const [subject, setSubject] = useState("");
  const [selectedClassroom, setSelectedClassroom] =
useState("");
  const [error, setError] = useState("");
  const [periods, setPeriods] = useState([]);
  const [loading, setLoading] = useState(false);
  const [notification, setNotification] = useState("");
  const formRef = useRef();

  const API_URL = process.env.REACT_APP_API_URL;

  useEffect(() => {
   if (notification) {
    setTimeout(() => setNotification(""), 5000);
   }
  }, [notification]);


  useEffect(() => {
   const fetchData = async () => {
    setLoading(true);
    try {
     const token = localStorage.getItem("token");
     if (!token) {

      setLoading(false);
      return;
     }

     const [specialtiesResponse, teachersResponse,
classroomsResponse, periodsResponse] = await Promise.all([
      axios.get(`${API_URL}/specialties`, { headers: {
Authorization: `Bearer ${token}` } }),
      axios.get(`${API_URL}/teachers`, { headers: {
Authorization: `Bearer ${token}` } }),
      axios.get(`${API_URL}/api/rooms`, { headers: {
Authorization: `Bearer ${token}` } }),
      axios.get(`${API_URL}/api/periods`),
     ]);

     setSpecialties(specialtiesResponse.data);
```

```
      } finally {
        setLoading(false);
      }
    };

    const handleDeleteCourse = async (courseId) => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        await axios.delete(`${apiUrl}/courses/${courseId}`, {
          headers: { Authorization: `Bearer ${token}` },
        });

        setCourses(courses.filter((course) => course._id !==
courseId));
      } catch (error) {
        console.error("Помилка видаленню курсу.", error);
        setError("Помилка видаленню курсу.");
      } finally {
        setLoading(false);
      }
    };

    const handleSearch = () => {
      setSearchQuery(search);
    };

    const filteredCourses = courses.filter((course) =>

course.name.toLowerCase().includes(searchQuery.toLowerCase())
    );

    const getSpecialtyName = (id) => {
      const specialty = specialties.find((spec) => spec._id === id);
      return specialty ? specialty.name : "Невідома спеціальність";
    };

    return (
      <div className="MainCabinets">
        <div className="CabinetsTextDashboard">
          <p className="text">Курси</p>
        </div>

        <div className="ContainerForAddAndSearchCourse">
          <div className="SearchContainerCourse">
            <input
              type="text"
              placeholder="Пошук курсів"
              value={search}
              onChange={(e) => setSearch(e.target.value)}
              className="InputSerchCabinet"
            />
            <div className="handleSearch">
              <img src={Search} alt="Search"
onClick={handleSearch} />
            </div>
          </div>

          <form onSubmit={handleCreateCourse}
className="ContainerForAdd">
            <select
              value={selectedSpecialty}
              onChange={(e) =>
setSelectedSpecialty(e.target.value)}
              className="InputForAddCabinets"
              required
            >
              <option value="">Оберіть спеціальність</option>
              {specialties.map((spec) => (
```

```
        setTeachers(teachersResponse.data);
        setClassrooms(classroomsResponse.data);
        setPeriods(periodsResponse.data);
      } catch (error) {

      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, [API_URL]);

  useEffect(() => {
    if (!selectedSpecialty) return;

    const fetchCourses = async () => {
      try {
        const token = localStorage.getItem("token");
        const response = await
axios.get(`${API_URL}/courses/${selectedSpecialty}`, {
          headers: { Authorization: `Bearer ${token}` },
        });
        setCourses(response.data);
      } catch (error) {
      }
    };

    fetchCourses();
  }, [selectedSpecialty, API_URL]);
  useEffect(() => {
    if (!selectedCourse) return;

    const fetchGroups = async () => {
      try {
        const token = localStorage.getItem("token");
        const response = await
axios.get(`${API_URL}/groups/${selectedCourse}`, {
          headers: { Authorization: `Bearer ${token}` },
        });
        setGroups(response.data);
      } catch (error) {

      }
    };

    fetchGroups();
  }, [selectedCourse, API_URL]);

  useEffect(() => {
    if (!date) return;

    const isValidDate = moment(date, "YYYY-MM-DD",
true).isValid();
    if (!isValidDate) {

      return;
    }

    const fetchWeekDay = async () => {
      try {
        const token = localStorage.getItem("token");
        const response = await
axios.get(`${API_URL}/api/weekday`, {
          params: { date },
          headers: { Authorization: `Bearer ${token}` },
        });
        setSelectedWeekDayId(response.data.weekDayId);
      } catch (error) {
```

```
                    <option key={spec.  id} value={spec.  id}>
                      {spec.name}
                    </option>
                  ))}
                </select>

                <input
                  type="text"
                  placeholder="Назва курсу"
                  value={name}
                  className="InputForAddCabinets"
                  onChange={(e) => setName(e.target.value)}
                  required
                />
                <button type="submit" className="SubmitCabinets">
                  Додати курс
                </button>
              </form>
            </div>

            {error && <p className="error-message">{error}</p>}

            {loading ? (
              <p>Завантаження...</p>
            ) : filteredCourses.length > 0 ? (
              <div className="courses-list">
                {filteredCourses.map((course) => (
                  <div key={course._id} className="Cabinet">
                    <div className="text_For_Course">
                      <span
className="Course_Name">{getSpecialtyName(course.specialty)
}</span>
                      <span
className="Course  id">{course.name}</span>
                    </div>
                    <div className="ButtonForCourse">
                      <div className="DeleteRooms" onClick={()
=> handleDeleteCourse(course._id)}>
                        <img src={Delete} alt="Remove" />
                      </div>
                    </div>
                  </div>
                ))}
              </div>
            ) : (
              <p>В базі немає курсів</p>
            )}
          </div>
        );
};

export default ManageCourses;
```

```
    }
  };

  fetchWeekDay();
 }, [date, API_URL]);


 const handleSubmit = async (event) => {
  event.preventDefault();

  const token = localStorage.getItem("token");
  if (!token) {

    return;
  }


   if (!selectedCourse || !selectedSpecialty || !selectedGroup ||
!selectedTeacher || !subject || !lessonType || !selectedPeriod ||
!selectedClassroom || !date) {

    return;
  }
```

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/CreatePeriod.css";
import Delete from "../assets/svg/Remove.svg";

const ManagePeriods = () => {
   const [name, setName] = useState("");
   const [startTime, setStartTime] = useState("");
   const [endTime, setEndTime] = useState("");
   const [periods, setPeriods] = useState([]);
   const [searchQuery] = useState("");
   const [error, setError] = useState("");
   const [loading, setLoading] = useState(false);

   useEffect(() => {
      const fetchPeriods = async () => {
```

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/CreateRoom.css";
import Delete from "../assets/svg/Remove.svg";
import Search from "../assets/svg/Search.svg";

const CreateRoom = () => {
   const [name, setName] = useState("");
   const [rooms, setRooms] = useState([]);
   const [search, setSearch] = useState("");
   const [searchQuery, setSearchQuery] = useState("");
   const [error, setError] = useState("");
   const [loading, setLoading] = useState(false);

   useEffect(() => {
      const fetchRooms = async () => {
```

```
        setLoading(true);
        try {
          const token = localStorage.getItem("token");
          const apiUrl = process.env.REACT_APP_API_URL;
          const response = await axios.get(`${apiUrl}/api/periods`,
{
            headers: { Authorization: `Bearer ${token}` }
          });
          setPeriods(response.data);
        } catch (error) {
        } finally {
          setLoading(false);
        }
      };
      fetchPeriods();
    }, []);

    const handleCreatePeriod = async (e) => {
      e.preventDefault();
      if (!name.trim() || !startTime || !endTime) return;
      setLoading(true);
      setError("");

      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        const response = await axios.post(
          `${apiUrl}/api/periods`,
          { name, startTime, endTime },
          { headers: { Authorization: `Bearer ${token}` }}
        );

        setPeriods([...periods, response.data]);
        setName("");
        setStartTime("");
        setEndTime("");
      } catch (error) {
      } finally {
        setLoading(false);
      }
    };

    const handleDeletePeriod = async (periodId) => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        await axios.delete(`${apiUrl}/api/periods/${periodId}`, {
          headers: { Authorization: `Bearer ${token}` }
        });

        setPeriods(periods.filter(period => period._id !==
periodId));
      } catch (error) {
        console.error("Ошибка удаления временного интервала:",
error);
        setError("Ошибка при удалении временного интервала");
      } finally {
        setLoading(false);
      }
    };

    const filteredPeriods = periods.filter(period =>
      period.name &&
period.name.toLowerCase().includes(searchQuery.toLowerCase())
    );

    return (
      <div className="MainCabinets">
        <div className="CabinetsTextDashboard">
```

```
        setLoading(true);
        try {
          const token = localStorage.getItem("token");
          const apiUrl = process.env.REACT_APP_API_URL;
          const response = await
axios.get(`${apiUrl}/api/rooms`, {
            headers: { Authorization: `Bearer ${token}` }
          });
          setRooms(response.data);
        } catch (error) {
          console.error("Не можу з'єднатися з базою
аудиторій", error);
          setError("Не можу з'єднатися з базою аудиторій.
Спробуйте пізніше.");
        } finally {
          setLoading(false);
        }
      };

      fetchRooms();
    }, []);

    const handleCreateRoom = async () => {
      if (!name.trim()) return;
      setLoading(true);
      setError("");

      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        const response = await axios.post(
          `${apiUrl}/api/rooms`,
          { name },
          { headers: { Authorization: `Bearer ${token}` } }
        );

        setRooms([...rooms, response.data]);
        setName("");
      } catch (error) {
        console.error("Помилка при створенні аудиторії:",
error);
        setError("Помилка при створенні аудиторії.
Спробуйте пізніше.");
      } finally {
        setLoading(false);
      }
    };


    const handleDeleteRoom = async (roomId) => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        await axios.delete(`${apiUrl}/api/rooms/${roomId}`, {
          headers: { Authorization: `Bearer ${token}` }
        });

        setRooms(rooms.filter(room => room.  id !==
roomId));
      } catch (error) {
        console.error("Помилка при видаленні аудиторії:",
error);
        setError("Помилка при видаленні аудиторії.
Спробуйте пізніше.");
      } finally {
        setLoading(false);
      }
    };
```

```
            <p className="text">Тривалість пар</p>
          </div>

          <div className="ContainerForAddAndSearch">
            <form onSubmit={handleCreatePeriod}
className="ContainerForAdd">
              <input
                type="text"
                placeholder="Назва (Напр. 1 пара)"
                value={name}
                className="InputForAddCabinets"
                onChange={(e) => setName(e.target.value)}
                required
              />
              <input
                type="time"
                value={startTime}
                className="InputForAddCabinets"
                onChange={(e) => setStartTime(e.target.value)}
                required
              />
              <input
                type="time"
                value={endTime}
                className="InputForAddCabinets"
                onChange={(e) => setEndTime(e.target.value)}
                required
              />
              <button type="submit" className="SubmitCabinets">
                Додати
              </button>
            </form>
          </div>

          {error && <p className="error-message">{error}</p>}

          {loading ? (
            <p>Загрузка...</p>
          ) : filteredPeriods.length > 0 ? (
            <div className="specialties-list">
              {filteredPeriods.map(period => (
                <div key={period._id} className="Cabinet">
                  <div className="Text_for_Periods">
                  <span
className="Period_Name">{period.name}</span>
                  <span
className="Period_Time">{period.startTime} -
{period.endTime}</span>
                  </div>
                  <div className="ButtonForSpeciality">
                    <div className="DeleteRooms" onClick={()
=> handleDeletePeriod(period._id)}>
                      <img src={Delete} alt="Remove" />
                    </div>
                  </div>
                </div>
              ))}
            </div>
          ) : (
            <p>Не існує тривалості пар.</p>
          )}
        </div>
      );
};

export default ManagePeriods;
```

```
  const handleSearch = () => {
    setSearchQuery(search);
  };

  const filteredRooms = rooms.filter(room =>

room.name.toLowerCase().includes(searchQuery.toLowerCase
())
  );

  return (
    <div className="MainCabinets">
      <div className="CabinetsTextDashboard">
        <p className="text">Аудиторії</p>
      </div>

      <div className="ContainerForAddAndSearch">

        <div className="SearchContainer">
          <input
            type="text"
            placeholder="Пошук аудиторії"
            value={search}
            onChange={(e) => setSearch(e.target.value)}
            className="InputSerchCabinet"
          />
          <div className="handleSearch">
            <img src={Search} alt="Search"
onClick={handleSearch} />
          </div>
        </div>

        <form onSubmit={handleCreateRoom}
className="ContainerForAdd">
          <input
            type="text"
            placeholder="Номер аудиторії"
            value={name}
            className="InputForAddCabinets"
            onChange={(e) => setName(e.target.value)}
            required
          />
          <div className="SubmitCabinets"
onClick={handleCreateRoom}>Додати аудиторію</div>
        </form>
      </div>

      {loading ? (
        <p>Загрузка...</p>
      ) : filteredRooms.length > 0 ? (
        <div style={{ display: "flex", flexWrap: "wrap", gap:
"20px" }}>
          {filteredRooms.map((room) => (
            <div
              key={room._id}
              className="Cabinet"
            >
              <span>{room.name}</span>
              <div className="ButtonForCabinets">
                <div className="DeleteRooms"
onClick={() => handleDeleteRoom(room._id)} >
                  <img src={Delete} alt="Remove" />
                </div>
              </div>
            </div>
          ))}
        </div>
      ) : (
        <p>В базі не існує кабінетів.</p>
      )}
```

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/ManageSpecialties.css";
import "../styles/CreateRoom.css";
import Delete from "../assets/svg/Remove.svg";
import Search from "../assets/svg/Search.svg";

const ManageSpecialties = () => {
  const [name, setName] = useState("");
  const [specialties, setSpecialties] = useState([]);
  const [search, setSearch] = useState("");
  const [searchQuery, setSearchQuery] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    const fetchSpecialties = async () => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        const response = await axios.get(`${apiUrl}/specialties`,
{
          headers: { Authorization: `Bearer ${token}` }
        });
        setSpecialties(response.data);
      } catch (error) {

      } finally {
        setLoading(false);
      }
    };

    fetchSpecialties();
  }, []);

  const handleCreateSpecialty = async (e) => {
    e.preventDefault();
    if (!name.trim()) return;
    setLoading(true);
    setError("");

    try {
      const token = localStorage.getItem("token");
      const apiUrl = process.env.REACT_APP_API_URL;
      const response = await axios.post(
        `${apiUrl}/specialties`,
        { name },
        { headers: { Authorization: `Bearer ${token}` }}
      );

      setSpecialties([...specialties, response.data]);
      setName("");
    } catch (error) {
    } finally {
      setLoading(false);
    }
  };

  const handleDeleteSpecialty = async (specialtyId) => {
    setLoading(true);
    try {
      const token = localStorage.getItem("token");
      const apiUrl = process.env.REACT_APP_API_URL;
```

```
      </div>
    );
};

export default CreateRoom;

import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/CreateTeacher.css";
import "../styles/CreateRoom.css";
import Delete from "../assets/svg/Remove.svg";
import Search from "../assets/svg/Search.svg";

const CreateTeachers = () => {
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [middleName, setMiddleName] = useState("");
  const [teachers, setTeachers] = useState([]);
  const [search, setSearch] = useState("");
  const [searchQuery, setSearchQuery] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    const fetchTeachers = async () => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        const response = await
axios.get(`${apiUrl}/teachers`, {
          headers: { Authorization: `Bearer ${token}` }
        });
        setTeachers(response.data);
      } catch (error) {
        console.error("Помилка загрузки викладачів",
error);
        setError("Неможу знайти викладачів, спробуйте
пізніше :(");
      } finally {
        setLoading(false);
      }
    };

    fetchTeachers();
  }, []);

  const handleCreateTeacher = async (e) => {
    e.preventDefault();
    if (!firstName.trim() || !lastName.trim() ||
!middleName.trim()) return;
    setLoading(true);
    setError("");

    try {
      const token = localStorage.getItem("token");
      const apiUrl = process.env.REACT_APP_API_URL;
      const response = await axios.post(
        `${apiUrl}/teachers`,
        { firstName, lastName, middleName },
        { headers: { Authorization: `Bearer ${token}` } }
      );

      setTeachers([...teachers, response.data]);
      setFirstName("");
      setLastName("");
      setMiddleName("");
    } catch (error) {
      console.error("Помилка створення викладача", error);
      setError("Помилка створення викладача");
```

```
        await axios.delete(`${apiUrl}/specialties/${specialtyId}`, {
          headers: { Authorization: `Bearer ${token}` }
        });

        setSpecialties(specialties.filter(spec => spec._id !==
specialtyId));
      } catch (error) {
      } finally {
        setLoading(false);
      }
    };

    const handleSearch = () => {
      setSearchQuery(search);
    };

    const filteredSpecialties = specialties.filter(spec =>

spec.name.toLowerCase().includes(searchQuery.toLowerCase())
    );

    return (
      <div className="MainCabinets">
        <div className="CabinetsTextDashboard">
          <p className="text">Спеціальності</p>
        </div>

        <div className="ContainerForAddAndSearch">
          <div className="SearchContainer">
            <input
              type="text"
              placeholder="Пошук спеціальностей"
              value={search}
              onChange={(e) => setSearch(e.target.value)}
              className="InputSerchCabinet"
            />
            <div className="handleSearch">
              <img src={Search} alt="Search"
onClick={handleSearch} />
            </div>
          </div>

          <form onSubmit={handleCreateSpecialty}
className="ContainerForAdd">
            <input
              type="text"
              placeholder="Назва спеціальності"
              value={name}
              className="InputForAddCabinets"
              onChange={(e) => setName(e.target.value)}
              required
            />
            <button type="submit" className="SubmitCabinets">
              Додати спеціальність
            </button>
          </form>
        </div>

        {error && <p className="error-message">{error}</p>}

        {loading ? (
          <p>Завантаження...</p>
        ) : filteredSpecialties.length > 0 ? (
          <div className="specialties-list">
            {filteredSpecialties.map(spec => (
              <div
                key={spec._id}
                className="Cabinet"
              >
                <span>{spec.name}</span>
      } finally {
        setLoading(false);
      }
    };

    const handleDeleteTeacher = async (teacherId) => {
      setLoading(true);
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;
        await axios.delete(`${apiUrl}/teachers/${teacherId}`, {
          headers: { Authorization: `Bearer ${token}` }
        });

        setTeachers(teachers.filter(teacher => teacher._id !==
teacherId));
      } catch (error) {
        console.error("Помилка видалення викладача",
error);
        setError("Помилка видалення викладача");
      } finally {
        setLoading(false);
      }
    };

    const handleSearch = () => {
      setSearchQuery(search);
    };

    const filteredTeachers = teachers.filter(teacher =>

teacher.fullName.toLowerCase().includes(searchQuery.toLowe
rCase())
    );

    return (
      <div className="MainCabinets">
        <div className="CabinetsTextDashboard">
          <p className="text">Викладачі</p>
        </div>

        <div
className="ContainerForAddAndSerchTeachers">
          <div className="SearchContainerTeachers">
            <input
              type="text"
              placeholder="Пошук викладачів"
              value={search}
              onChange={(e) => setSearch(e.target.value)}
              className="InputSerchCabinet"
            />
            <div className="handleSearch">
              <img src={Search} alt="Search"
onClick={handleSearch} />
            </div>
          </div>

          <form onSubmit={handleCreateTeacher}
className="ContainerForAddTeachers">
            <input
              type="text"
              placeholder="Прізвище"
              value={lastName}
              className="InputForAddCabinets"
              onChange={(e) =>
setLastName(e.target.value)}
              required
            />
            <input
              type="text"
```

```
                    <div className="ButtonForSpeciality">
                      <div className="DeleteRooms" onClick={()
=> handleDeleteSpecialty(spec._id)} >
                        <img src={Delete} alt="Remove" />
                      </div>
                    </div>
                  </div>
                ))}
              </div>
            ) : (
              <p>В базі немає спеціальностей</p>
            )}
          </div>
    );
};

export default ManageSpecialties;
```

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import axios from "axios";
import HeaderForAuth from
"../Widget/HeaderForAuth/HeaderForAuth";
import "../styles/Auth.css";
import ImgForAuth from "../assets/svg/GirlWithLaptop.svg";
import FooterForAuth from
"../Widget/FooterForAuth/FooterForAuth";
```

```
                placeholder="Ім'я"
                value={firstName}
                className="InputForAddCabinets"
                onChange={(e) =>
setFirstName(e.target.value)}
                required
              />
              <input
                type="text"
                placeholder="По батькові"
                value={middleName}
                className="InputForAddCabinets"
                onChange={(e) =>
setMiddleName(e.target.value)}
                required
              />
              <button type="submit"
className="SubmitCabinets">
                Додати викладача
              </button>
            </form>
          </div>

          {error && <p className="error-
message">{error}</p>}

          {loading ? (
            <p>Завантаження...</p>
          ) : filteredTeachers.length > 0 ? (
            <div className="teacher-list">
              {filteredTeachers.map((teacher) => (
                <div
                  key={teacher._id}
                  className="Cabinet"
                >
                  <div className="Teachers-FullName">
                  <span className="Teachers-
Name">{teacher.lastName}</span>
                  <span className="Teachers-
Name">{teacher.firstName}</span>
                  <span className="Teachers-
Name">{teacher.middleName}</span>
                  </div>
                  <div className="ButtonForCabinets">
                    <div className="DeleteRooms"
onClick={() => handleDeleteTeacher(teacher._id)} >
                      <img src={Delete} alt="Remove" />
                    </div>
                  </div>
                </div>
              ))}
            </div>
          ) : (
            <p>В базі немає викладачів</p>
          )}
        </div>
    );
};

export default CreateTeachers;
```

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/ManageGroups.css";
import "../styles/CreateRoom.css";
import Delete from "../assets/svg/Remove.svg";
import Search from "../assets/svg/Search.svg";

const ManageGroups = () => {
  const [name, setName] = useState("");
```

```
const API_URL = process.env.REACT_APP_API_URL;

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  useEffect(() => {
    document.title = "SchedGo - Вхід";
  }, []);

  const handleLogin = async (e) => {
    e.preventDefault();
    setError(null);

    try {
      const response = await
axios.post(`${API_URL}/auth/login`, { email, password });
      localStorage.setItem("token", response.data.token);
      navigate("/dashboard");
    } catch (err) {
      setError("Помилка");
    }
  };

  return (
    <div className="auth-container">
      <HeaderForAuth />
      <div className="Form_For_Auth">
        <div className="Auth_Form">
          <div className="Text_Form">
            <h2 className="WelcomTextAuth">Ласкаво
просимо!</h2>
            <h3 className="SadTextAuth">З поверненням,
ми сумували 😿</h3>
          </div>
          {error && <p style={{ color: "red" }}>{error}</p>}
          <form onSubmit={handleLogin}
className="Form_SignIn">
            <div className="InputCss">
              <label htmlFor="username"
className="LabelCssForm">Email:</label>
              <input
                type="email"
                id="username"
                placeholder="Email"
                className="Form_input"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
              />
            </div>
            <div className="InputCss">
              <label htmlFor="password"
className="LabelCssForm">Пароль:</label>
              <input
                type="password"
                id="password"
                placeholder="Пароль"
                className="Form_input"
                value={password}
                onChange={(e) =>
setPassword(e.target.value)}
                required
              />
            </div>
            <button type="submit"
className="Form_button">Увійти до Sched</button>
```

```
  const [selectedSpecialty, setSelectedSpecialty] =
useState("");
  const [selectedCourse, setSelectedCourse] = useState("");
  const [specialties, setSpecialties] = useState([]);
  const [courses, setCourses] = useState([]);
  const [filteredCourses, setFilteredCourses] = useState([]); //
Новый стейт для фильтрованных курсов
  const [groups, setGroups] = useState([]);
  const [search, setSearch] = useState("");
  const [searchQuery, setSearchQuery] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const token = localStorage.getItem("token");
        const apiUrl = process.env.REACT_APP_API_URL;

        const [specialtiesResponse, coursesResponse,
groupsResponse] = await Promise.all([
          axios.get(`${apiUrl}/specialties`),
          axios.get(`${apiUrl}/courses`, { headers: {
Authorization: `Bearer ${token}` } }),
          axios.get(`${apiUrl}/api/groups`, { headers: {
Authorization: `Bearer ${token}` } }),
        ]);

        setSpecialties(specialtiesResponse.data);
        setCourses(coursesResponse.data);
        setGroups(groupsResponse.data);
      } catch (error) {
        console.error("Помилка загрузки даних.", error);
        setError("Помилка загрузки даних.");
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  useEffect(() => {
    if (selectedSpecialty) {

      const filtered = courses.filter((course) =>
course.specialty === selectedSpecialty);
      setFilteredCourses(filtered);
    } else {
      setFilteredCourses(courses);
    }
  }, [selectedSpecialty, courses]);

  const handleCreateGroup = async (e) => {
    e.preventDefault();
    if (!name.trim() || !selectedSpecialty || !selectedCourse)
return;
    setLoading(true);
    setError("");

    try {
      const token = localStorage.getItem("token");
      const apiUrl = process.env.REACT_APP_API_URL;
      const response = await axios.post(
        `${apiUrl}/api/groups`,
        { name, specialty: selectedSpecialty, course:
selectedCourse },
        { headers: { Authorization: `Bearer ${token}` } }
      );
```

```
            </form>
          </div>
          <div className="ImgForAuth">
            <img src={ImgForAuth} alt="GirlWithLaptop"
className="ImgForAuth" />
          </div>
        </div>
        <FooterForAuth />
      </div>
  );
};

export default Login;
```

```
      setGroups([...groups, response.data.group]);
      setName("");
    } catch (error) {
      console.error("Помилка створення групи.", error);
      setError("Помилка створення групи.");
    } finally {
      setLoading(false);
    }
  };
```

Вихідний код Express.js

```
const mongoose = require("mongoose");

const courseSchema = new mongoose.Schema({
  name: { type: String, required: true },
  specialty: { type: mongoose.Schema.Types.ObjectId, ref:
"Specialty", required: true }
});

module.exports = mongoose.model("Course", courseSchema);
```

```
const mongoose = require('mongoose');

const groupSchema = new mongoose.Schema({
  name: { type: String, required: true },
  course: { type: mongoose.Schema.Types.ObjectId, ref:
"Course", required: true },
  specialty: { type: mongoose.Schema.Types.ObjectId, ref:
'Specialty', required: true }
});

module.exports = mongoose.model('Group', groupSchema);
```

```
const mongoose = require('mongoose');

const periodSchema = new mongoose.Schema({
  name: { type: String, required: true },
  startTime: { type: String, required: true },
  endTime: { type: String, required: true }
});

module.exports = mongoose.model('Period', periodSchema);
```

```
const mongoose = require("mongoose");

const RequestSchema = new mongoose.Schema({
 requestType: { type: String, required: true },
 createdAt: { type: Date, default: Date.now },
 schoolName: { type: String },
 studentCount: { type: Number },
 contactNumber: { type: String },
 email: { type: String, required: true },
});

module.exports = mongoose.model("Request", RequestSchema);
```

```
const mongoose = require("mongoose");

const RoomSchema = new mongoose.Schema({
 name: { type: String, required: true },
});

RoomSchema.pre("save", function (next) {
 if (!this.name.startsWith("Каб.")) {
  this.name = `Каб. ${this.name}`;
 }
 next();
});
```

```
const mongoose = require('mongoose');

const scheduleSchema = new mongoose.Schema({
  subject: { type: String, required: true },
  teacher: { type: mongoose.Schema.Types.ObjectId, ref:
'Teacher', required: true },
  period: { type: mongoose.Schema.Types.ObjectId, ref: 'Period',
required: true },
  lessonType: {
    type: String,
    enum: ['Практика', 'Лекція', 'Лабораторна', 'Іспит',
'Навчальна практика', 'Виїзна практика'],
```

```
module.exports = mongoose.model("Room", RoomSchema);
```

```
const mongoose = require('mongoose');

const specialtySchema = new mongoose.Schema({
   name: String
});

module.exports = mongoose.model('Specialty',
specialtySchema);
```

```
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
   firstName: {
      type: String,
      required: [true, "Ім'я обов'язкове для заповнення"]
   },
   lastName: {
      type: String,
      required: [true, "Прізвище обов'язкове для заповнення"]
   },
   patronymic: {
      type: String,
      required: false
   },
   position: {
      type: String,
      required: [true, "Посада обов'язкова для заповнення"]
      required: true
   },
   group: { type: mongoose.Schema.Types.ObjectId, ref: 'Group',
required: true },
   room: { type: mongoose.Schema.Types.ObjectId, ref: 'Room',
required: true },
   dayOfWeek: { type: Number, required: true },
   date: { type: Date, required: true },
   specialty: { type: mongoose.Schema.Types.ObjectId, ref:
'Specialty', required: true },
   course: { type: mongoose.Schema.Types.ObjectId, ref:
'Course', required: true }
});

module.exports = mongoose.model('Schedule', scheduleSchema);
```

```
const { Schema, model } = require('mongoose');

const teacherSchema = new Schema({
 firstName: { type: String, required: true },
 lastName: { type: String, required: true },
 middleName: { type: String, required: true },
});

teacherSchema.virtual('fullName').get(function() {
 return `${this.lastName} ${this.firstName}
${this.middleName}`;
});

teacherSchema.set('toJSON', {
 virtuals: true
});

module.exports = model('Teacher', teacherSchema);
```

```
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { check, validationResult } = require('express-validator');
const User = require('../models/User');
const router = express.Router();

const generateToken = (user) => {
   return jwt.sign({ id: user._id, role: user.role },
process.env.JWT_SECRET, { expiresIn: '30d' });
};

router.post('/register', [
   check('firstName', 'Ім’я
обов’язково').notEmpty().trim().escape(),
   check('lastName', 'Прізвище
обов’язково').notEmpty().trim().escape(),
   check('patronymic', 'По-батькові
обов’язково').optional().trim().escape(),
```

```
  },
  educationalInstitution: {
    type: String,
    required: [true, "Навчальний заклад обов'язковий для заповнення"]
  },
  phoneNumber: {
    type: String,
    required: [true, "Номер телефону обов'язковий"],
    unique: true,
    validate: {
      validator: function(v) {
        return /^\+?[1-9]\d{1,14}$/.test(v);
      },
      message: props => `${props.value} не є валідним номером телефону!`
    }
  },
  email: {
    type: String,
    required: [true, "Електронна пошта обов'язкова"],
    unique: true,
    validate: {
      validator: function(v) {
        return /^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/.test(v);
      },
      message: props => `${props.value} не є валідною електронною поштою!`
    }
  },
  password: {
    type: String,
    required: [true, "Пароль обов'язковий"]
  },
  role: {
    type: String,
    enum: {
      values: ['admin', 'institution', 'user'],
      message: 'Неприпустима роль користувача'
    },
    required: [true, "Роль обов'язкова для заповнення"]
  }
});

userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) return next();

  try {
    const salt = await bcrypt.genSalt(10);
    this.password = await bcrypt.hash(this.password, salt);
    next();
  } catch (err) {
    next(err);
  }
  check('position', 'Посада обов’язково').notEmpty().trim().escape(),
  check('educationalInstitution', 'Навчальний заклад обов’язково').notEmpty().trim().escape(),
  check('phoneNumber', 'Неправильний формат номеру телефона').matches(/^\+380\d{9}$/),
  check('email', 'Введіть правильний Email').isEmail().normalizeEmail(),
  check('password', 'Пароль повинен бути не меньше 6 символів.').isLength({ min: 6 }),
  check('role', 'Оберіть роль').isIn(['admin', 'institution', 'user'])
], async (req, res) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }


    const {
      firstName,
      lastName,
      patronymic,
      position,
      educationalInstitution,
      phoneNumber,
      email,
      password,
      role
    } = req.body;


    const existingUser = await User.findOne({
      $or: [{ email }, { phoneNumber }]
    });

    if (existingUser) {
      const conflictField = existingUser.email === email ? 'Email' : 'Телефон';
      return res.status(409).json({
        message: `${conflictField} вже зареєстрований  в системі.`
      });
    }


    const user = new User({
      firstName,
      lastName,
      patronymic,
      position,
      educationalInstitution,
      phoneNumber,
```

```
    }
});

module.exports = mongoose.model('User', userSchema);
```

```
const express = require("express");
const Course = require("../models/Course");

const router = express.Router();

router.get("/", async (req, res) => {
    try {
        const courses = await Course.find();
        res.json(courses);
    } catch (error) {
        res.status(500).json({ message: "Помилка сервера", error:
error.message });
    }
});

router.get("/:specialtyId", async (req, res) => {
    try {
        const courses = await Course.find({ specialty:
req.params.specialtyId });
        res.json(courses);
    } catch (error) {
        res.status(500).json({ message: "Помилка сервера", error:
error.message });
    }
});

router.post("/", async (req, res) => {
    try {
        const { name, specialty } = req.body;
        const newCourse = new Course({ name, specialty });
        await newCourse.save();
        res.status(201).json(newCourse);
    } catch (error) {
        res.status(500).json({ message: "Помилка при створенні
курсу", error: error.message });
    }
});

router.delete("/:id", async (req, res) => {
    try {
        const result = await
Course.findByIdAndDelete(req.params.id);
        if (!result) {
```

```
            email,
            password,
            role
        });


        await user.save();

module.exports = router;
```

```
const express = require('express');
const Group = require('../models/Group');
const router = express.Router();

router.get('/', async (req, res) => {
    try {
        const groups = await
Group.find().populate('specialty').populate('course');
        res.json(groups);
    } catch (error) {
        res.status(500).json({ message: 'Помилка при створенні
групи', error: error.message });
    }
});

router.post('/', async (req, res) => {
    try {
        const newGroup = new Group(req.body);
        await newGroup.save();
        res.json({ message: 'Групу створенно', group: newGroup
});
    } catch (error) {
        res.status(500).json({ message: 'Помилка при створенні
групи', error: error.message });
    }
});

router.delete('/:id', async (req, res) => {
    try {
        const deletedGroup = await
Group.findByIdAndDelete(req.params.id);
        if (!deletedGroup) {
            return res.status(404).json({ message: 'Групу не
знайдено' });
        }
        res.json({ message: 'Групу видалено', group: deletedGroup
});
    } catch (error) {
        res.status(500).json({ message: 'Помилка пр видаленні
пари', error: error.message });
    }
});
module.exports = router;
```

```
            return res.status(404).json({ message: "Курс не
знайдено" });
        }
        res.status(200).json({ message: "Курс видалений" });
    } catch (error) {
        res.status(500).json({ message: "Помилка при видалені
курсу", error: error.message });
    }
});

module.exports = router;
```

Вихідний код Telegram bot

```
import logging
from telegram import InlineKeyboardButton, InlineKeyboardMarkup, ReplyKeyboardMarkup, KeyboardButton
from config import BOT_TOKEN, ADMIN_IDS
from services.api_client import fetch_specialties, fetch_courses, fetch_groups, get_specialty_by_id, get_courses_by_specialty,
get_groups_by_course
from database.models import get_user, update_user, add_user, db
from datetime import datetime, timezone, timedelta
from services.api_client import get_schedule, fetch_teachers
from collections import defaultdict
import asyncio
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, MessageHandler, filters
import nest_asyncio
from zoneinfo import ZoneInfo
from apscheduler.schedulers.background import BackgroundScheduler
from apscheduler.schedulers.asyncio import AsyncIOScheduler

nest  asyncio.apply()


def get_main_menu(user_id):
    buttons = [
        [KeyboardButton("👤 Профіль"), KeyboardButton("🗓 Розклад на неділю")],
        [KeyboardButton("📅 Розклад на сьогодні")
    ]

    if user_id in ADMIN_IDS:
        buttons.append([KeyboardButton("⚙️ Адмін-панель")])

    return ReplyKeyboardMarkup(keyboard=buttons, resize_keyboard=True)

async def start(update, context):
    user_id = update.effective_user.id

    await add_user(user_id)

    user = await get_user(user_id)

    if user and user.get("role") and user.get("specialty_id") and user.get("course_id") and user.get("group_id"):
        # Главное меню
        keyboard = get_main_menu(user_id)
        await update.message.reply  text(
            "Привіт! Радий тебе бачити 🥰\n\n👇 Ось головне меню:",
            reply_markup=keyboard
        )
    else:

        keyboard = InlineKeyboardMarkup([
            [InlineKeyboardButton("Студент", callback_data="role_student")],
```

```
            [InlineKeyboardButton("Викладач", callback  data="role  teacher")]
        ])
        await update.message.reply_text(
            f"Привіт, <b>{update.effective_user.first_name}</b>! Цей бот твій ассистент для отримання розкладу в онлайн режимі.\n\nТож давай ідентифікуємо тебе. Обери свою роль 👇",
            reply_markup=keyboard,
            parse_mode="HTML"
        )


async def handle_role_selection(update, context):
    query = update.callback_query
    user_id = query.from_user.id
    await query.answer()

    try:
        if query.data == "role_student":
            await update_user(user_id, {"role": "student"})

            specialties = await fetch_specialties()

            if specialties:
                keyboard = [
                    [InlineKeyboardButton(specialty["name"], callback_data=f"specialty_{specialty['_id']}")]
                    for specialty in specialties
                ]
                reply_markup = InlineKeyboardMarkup(keyboard)

                await query.edit_message_text(
                    "Ти обрав, що ти Студент. Тепер обери свою спеціальність 👇",
                    reply_markup=reply_markup
                )
            else:
                await query.edit_message_text("Не вдалось загрузити спеціальності")

        elif query.data == "role_teacher":
            await update  user(user  id, {"role": "teacher"})

            teachers = await fetch  teachers()

            if teachers:
                keyboard = [
                    [InlineKeyboardButton(teacher["fullName"], callback_data=f"teacher_{teacher['_id']}")]
                    for teacher in teachers
                ]
                reply_markup = InlineKeyboardMarkup(keyboard)

                await query.edit_message_text(
                    "Ви обрали роль: Викладач. Тепер оберіть себе зі списку 👇",
                    reply_markup=reply_markup
                )
            else:
                await query.edit_message_text("Не вдалось.")

    except Exception as e:
        await query.edit_message_text("Помилка.")

async def handle_specialty_selection(update, context):
    query = update.callback_query
    user_id = query.from_user.id
    specialty_id = query.data.split('_')[1]

    try:
        await update  user(user  id, {"specialty  id": specialty  id})

        courses = await fetch_courses(specialty_id)

        if courses:
            keyboard = [
```

```
            [InlineKeyboardButton(course["name"], callback_data=f"course_{course['_id']}")]
            for course in courses
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)

        await query.edit_message_text("Тепер обери свій курс 👇", reply_markup=reply_markup)
      else:
        await query.edit_message_text("Не вдалось загрузити спеціальності. Спробуй пізніше")
    except Exception as e:
      logging.error(f"Помилка при виборі спеціальності: {e}")
      await query.edit_message_text("Спробуй пізніше")


async def handle_course_selection(update, context):
    query = update.callback_query
    user_id = query.from_user.id
    course_id = query.data.split('_')[1]

    try:
      await update_user(user_id, {"course_id": course_id})

      groups = await fetch_groups(course_id)

      if groups:
        keyboard = [
          [InlineKeyboardButton(group["name"], callback_data=f"group_{group['_id']}")]
          for group in groups
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)

        await query.edit_message_text("Та твоя группа 👇", reply_markup=reply_markup)
      else:
        await query.edit_message_text("Не вдалось загрузити групи. Спробуй пізніше.")
    except Exception as e:
      logging.error(f"Помилка при виборі групи: {e}")
      await query.edit_message_text("Спробуй пізніше")


async def handle_group_selection(update, context):
    query = update.callback_query
    user_id = query.from_user.id
    group_id = query.data.split('_')[1]

    try:
      await update_user(user_id, {"group_id": group_id, "created_at": datetime.now(timezone.utc).isoformat()})

      keyboard = get_main_menu(user_id)

      await query.message.reply_text(
        "🎉 Дякую, що приєднався до бота!\n\n👇 Ось головне меню:",
        reply_markup=keyboard,
      )

      await query.delete_message()

    except Exception as e:
      await query.edit_message_text("Помилка.")


async def handle_profile(update, context):
    user_id = update.effective_user.id
    user = await get_user(user_id)

    if user:
      specialty_id = user.get("specialty_id")
      course_id = user.get("course_id")
      group_id = user.get("group_id")

      specialty_name = user.get("specialty_name")
```

```
    course_name = user.get("course_name")
    group_name = user.get("group_name")

    if not specialty_name and specialty_id:
        specialty_data = await get_specialty_by_id(specialty_id)
        specialty_name = specialty_data.get("name") if specialty_data else "Не вказано"

    if not course_name and course_id:
        course_data = await get_courses_by_specialty(specialty_id)
        if course_data:
            course_name = course_data[0].get("name", "Не вказано")
        else:
            course_name = "Не вказано"

    if not group_name and group_id:
        group_data = await get_groups_by_course(course_id)
        if group_data:
            group_name = group_data[0].get("name", "Не вказано")
        else:
            group_name = "Не вказано"

    created_at_str = user.get("created_at")
    if created_at_str:
        created_at = datetime.fromisoformat(created_at_str)
        now = datetime.now(timezone.utc)

        time_diff = now - created_at

        if time_diff.days >= 1:
            days = time_diff.days
            if days == 1:
                time_passed = "1 день тому"
            elif 2 <= days <= 4:
                time_passed = f"{days} дні тому"
            else:
                time_passed = f"{days} днів тому"
        else:
            minutes = int(time_diff.total_seconds() // 60)
            if minutes < 60:
                minutes = max(minutes, 1)
                time_passed = f"{minutes} хвилин тому"
            else:
                hours = minutes // 60
                if hours == 1:
                    time_passed = "1 годину тому"
                elif 2 <= hours <= 4:
                    time_passed = f"{hours} години тому"
                else:
                    time_passed = f"{hours} годин тому"
    else:
        time_passed = "Невідомо коли"

    text = (
        f"📚 Спеціальність: {specialty_name}\n"
        f"🎓 Курс: {course_name}\n"
        f"👥 Група: {group_name}\n\n"
        f"Ви почали використовувати бота {time_passed} ✨"
    )

    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton("⬅️ Назад в меню")],
        ],
        resize_keyboard=True
    )

    await update.message.reply_text(text, reply_markup=keyboard)
else:
    await update.message.reply_text("Не вдалось тебе ідентифікувати 😢")
```

```
async def handle_back_to_menu(update, context):
    user_id = update.effective_user.id
    keyboard = get_main_menu(user_id)

    await update.message.reply_text(
        "🔙 Повертаю тебе в головне меню!",
        reply_markup=keyboard
    )

def format_schedule(lessons):
    lesson_type_abbreviations = {
        "Практика": "Пр",
        "Лекція": "Лк",
        "Лабораторна": "Лб",
        "Іспит": "Екз",
        "Навчальна практика": "НП",
        "Виїздна практика": "ВП"
    }

    lessons.sort(key=lambda x: (x["date"], x["period"]["startTime"]))

    lessons_by_date = defaultdict(list)
    for lesson in lessons:
        lessons_by_date[lesson["date"]].append(lesson)

    lines = []

    for date_iso in sorted(lessons_by_date):
        date_str = datetime.fromisoformat(date_iso.replace('Z', '+00:00')).strftime('%d.%m.%Y')
        lines.append(f"<b>📅 {date_str} :</b>\n")

        for lesson in lessons_by_date[date_iso]:
            period = lesson.get("period", {})
            time_start = period.get("startTime", "??:??")
            time_end = period.get("endTime", "??:??")

            subject = lesson.get("subject", "Без назви")

            lesson_type = lesson.get("lessonType", "")
            lesson_type_short = lesson_type_abbreviations.get(lesson_type, "")
            if lesson_type_short:
                subject = f"{subject} [{lesson_type_short}]"

            teacher = lesson.get("teacher", {}).get("fullName", "Без викладача")
            room = lesson.get("room", {}).get("name", "Без кабінета")

            lines.append(
                f"🕐 {time_start} - {time_end}\n"
                f" {subject}\n"
                f"👨‍🏫 {teacher}\n"
                f" {room}\n"
            )
        lines.append(" ")

    return "\n".join(lines).strip()

async def handle_schedule_week(update, context):
    user_id = update.effective_user.id
    user = await get_user(user_id)

    if user and user.get("group_id"):
        lessons = await get_schedule(user["group_id"])

        if lessons:
            today = datetime.now().date()
            end_of_week = today + timedelta(days=(6 - today.weekday()))
```

```
        filtered_lessons = [
            lesson for lesson in lessons
            if today <= parse_lesson_date(lesson["date"]).date() <= end_of_week
        ]

        if filtered_lessons:
            text = format_schedule(filtered_lessons)
            await update.message.reply_text(f"\n\n{text}", parse_mode="HTML")
        else:
            await update.message.reply_text("На цій неділі немає пар! 🎉")


def parse_lesson_date(date_str):
    if '.' in date_str:
        date_str = date_str.split('.')[0]
    return datetime.fromisoformat(date_str)

async def handle_schedule_today(update, context):
    user_id = update.effective_user.id
    user = await get_user(user_id)

    if user and user.get("group_id"):
        lessons = await get_schedule(user["group_id"])

        if lessons:
            today = datetime.now().date()
            date_str = today.strftime("📅 %d.%m.%Y:")

            today_lessons = [
                lesson for lesson in lessons
                if parse_lesson_date(lesson["date"]).date() == today
            ]

            if today_lessons:
                text = format_schedule(today_lessons)
                await update.message.reply_text(
                    f"\n{text}",
                    parse_mode="HTML"
                )
            else:
                await update.message.reply_text(
                    f"<b>{date_str}</b>\n\nПари сьогодні відсутні!",
                    parse_mode="HTML"
                )
        else:
            await update.message.reply_text("Розклад не знайдено.")
    else:
        await update.message.reply_text("Спочатку оберіть групу.")


async def handle_change_role(update, context):
    keyboard = InlineKeyboardMarkup([
        [InlineKeyboardButton("Студент", callback_data="role_student")],
        [InlineKeyboardButton("Викладач", callback_data="role_teacher")]
    ])
    await update.message.reply_text("🔄 Выберите новую роль:", reply_markup=keyboard)


async def handle_admin(update, context):
    user_id = update.effective_user.id

    if user_id not in ADMIN_IDS:
        await update.message.reply_text("⛔ У вас немає доступу до адмін-панелі.")
        return

    total_users = await get_total_users_count()

    keyboard = ReplyKeyboardMarkup(
```

```
        keyboard=[
            [KeyboardButton("📢 Зробити розсилку")],
            [KeyboardButton("🔙 Назад в меню")]
        ],
        resize_keyboard=True
    )

    await update.message.reply_text(
        f"📊 Статистика:\n\n👥 Кількість користувачів: {total_users}",
        reply_markup=keyboard
    )

async def handle_broadcast_request(update, context):
    user_id = update.effective_user.id
    if user_id not in ADMIN_IDS:
        return

    await update.message.reply_text("✍️ Напишіть повідомлення, яке хочете надіслати всім користувачам:")

    context.user_data["awaiting_broadcast"] = True


async def handle_broadcast_message(update, context):
    user_id = update.effective_user.id
    if user_id not in ADMIN_IDS or not context.user_data.get("awaiting_broadcast"):
        return

    message = update.message.text + "\n\n<em>Розсилка повідомлень</em>"

    users = await get_all_user_ids()
    count = 0

    for uid in users:
        try:
            await context.bot.send_message(chat_id=uid, text=message, parse_mode="HTML")
            count += 1
        except Exception as e:
            logging.warning(f"Не вдалося надіслати користувачу {uid}: {e}")

    context.user_data["awaiting_broadcast"] = False

    await update.message.reply_text(f"✅ Розсилка завершена. Надіслано {count} користувачам.")


async def get_all_user_ids():
    users = await db.users.find({}, {"user_id": 1}).to_list(None)
    return [user["user_id"] for user in users if "user_id" in user]

async def get_total_users_count():
    return await db.users.count_documents({})

async def broadcast_tomorrow_schedule(app):
    users = await get_all_user_ids()

    if not users:
        return

    tomorrow = datetime.now(ZoneInfo("Europe/Kyiv")).date() + timedelta(days=1)

    for uid in users:

        user = await get_user(uid)
        if not user:
            continue

        if not user.get("group_id"):
            continue

        lessons = await get_schedule(user["group_id"])
```

```
        if not lessons:
            continue

        tomorrow_lessons = [
            lesson for lesson in lessons
            if parse_lesson_date(lesson["date"]).date() == tomorrow
        ]

        if not tomorrow_lessons:
            continue

        message = format_schedule(tomorrow_lessons)
        message += "\n\n<em>Автоматична розсилка розкладу</em>"

        try:
            await app.bot.send_message(chat_id=uid, text=message, parse_mode="HTML")

async def main():
    app = Application.builder().token(BOT  TOKEN).build()
    loop = asyncio.get_event_loop()

    app.add_handler(CommandHandler("start", start))
    app.add_handler(CallbackQueryHandler(handle_role_selection, pattern="role_"))
    app.add_handler(CallbackQueryHandler(handle_specialty_selection, pattern="specialty_"))
    app.add  handler(CallbackQueryHandler(handle  course  selection, pattern="course  "))
    app.add_handler(CallbackQueryHandler(handle_group_selection, pattern="group_"))

    app.add_handler(MessageHandler(filters.Text("👤 Профіль"), handle_profile))
    app.add_handler(MessageHandler(filters.Text("📝 Розклад на неділю"), handle_schedule_week))
    app.add_handler(MessageHandler(filters.Text("📅 Розклад на сьогодні"), handle_schedule_today))
    app.add_handler(MessageHandler(filters.Text("🔄 Змінити роль"), handle_change_role))
    app.add_handler(MessageHandler(filters.Text("🔙 Назад в меню"), handle_back_to_menu))
    app.add_handler(MessageHandler(filters.Text("⚙️ Адмін-панель"), handle_admin))
    app.add_handler(MessageHandler(filters.Text("📢 Зробити розсилку"), handle_broadcast_request))
    app.add_handler(MessageHandler(filters.TEXT & (~filters.COMMAND), handle_broadcast_message))

    scheduler = AsyncIOScheduler(timezone="Europe/Kyiv")
    scheduler.add_job(
        lambda: asyncio.run_coroutine_threadsafe(broadcast_tomorrow_schedule(app), loop),"cron", hour=15, minute=0)
    scheduler.start()

    logging.info("Бот запущен.")
    await app.run_polling()


if __name__ == "__main__":
    asyncio.run(main())
```