

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ WEBЗАСТОСУНКУ РОЗКЛАДУ ЗАНЯТЬ
ДЛЯ СТУДЕНТІВ

SOFTWARE DEVELOPMENT OF A WEB APPLICATION FOR
STUDENT TIMETABLE MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-113сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ШРАМКО В. В.

(ПРІЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т. В.

(ПРІЗВИЩЕ та ініціали)

Рецензент БАБЕНКО Н. В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШРАМКА Владислава В'ячеславовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація вебзастосунку розкладу занять для студентів. Software Development of a Web Application for Student Timetable Management

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
 3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ФЕДОРОНЧАК Т. В., доцент		
Нормоконтроль	КАЛІНІНА М. В., асистент		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Владислав ШРАМКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ФЕДОРОНЧАК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 92 с., 6 табл., 19 рис., 2 дод., 22 джерела.

РОЗКЛАД ЗАНЯТЬ, ВЕБЗАСТОСУНОК, HTML, CSS, JAVASCRIPT, NODE.JS, MONGODB, RENDER.

Об'єктом дослідження цієї роботи є процес розробки програмного забезпечення для перегляду студентами розкладу занять.

Предмет дослідження – програмні засоби для організації процесу перегляду розкладу занять.

Мета роботи – підвищення зручності перегляду розкладу занять для студентів.

Матеріали, методи та технічні засоби: HTML - мова гіпертекстової розмітки документів, CSS – мова стилізації та оформлення зовнішнього вигляду веб-сторінок, мова програмування JAVASCRIPT та фреймворк NODE.JS, документо-орієнтована система управління базами даних – MONGODB. Також персональний комп'ютер з операційною системою Windows 10 або аналоги macOS та Linux, а також встановлений один з браузерів Chrome, Firefox, Edge, Safari та з процесором з мінімум двома ядрами, тактова частота яких є не нижчою за 1.5 ГГц, а також з вбудованим графічним ядром.

Результати. Спроектовано та розроблено програмне забезпечення для виконання перегляду розкладу занять.

Висновки. Розроблено програмне забезпечення для перегляду розкладу занять навчальних груп, яке полегшує студентам відстежувати початок занять.

Галузь використання – засоби перегляду розкладу занять.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 92 pages, 6 tables, 19 figures, 2 appendixes, 22 sources.

CLASS SCHEDULE, WEB APPLICATION, HTML, CSS, JAVASCRIPT, NODE.JS, MONGODB, RENDER.

The object of this research is the process of developing software for students timetable management.

The subject of the research is software tools for organizing the process of viewing class schedule.

The purpose of the work is to increase the convenience of students timetable management.

Materials, methods and technical means: HTML – a hypertext markup language for documents, CSS - a language for styling and designing the appearance of web pages, JAVASCRIPT programming language and the NODE.JS framework, a document-oriented database management system - MONGODB. Also a personal computer with the Windows 10 operating system or macOS and Linux, as well as one of the browsers Chrome, Firefox, Edge, Safari installed and a processor with at least two cores, the clock frequency of which is not lower than 1.5 GHz, as well as with a built-in graphics core.

Results. Software for students timetable management has been designed and developed.

Conclusions. Software for students timetable management of academic groups has been developed, which makes it easier for students to track the start of classes.

Field of use – means of reviewing class schedules.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області	10
1.1 Поняття розкладу. Програмне забезпечення для вебзастосунку розкладу занять для студентів.....	10
1.2 Огляд вебзастосунків розкладу занять.....	14
1.3 Порівняння вебзастосунків розкладу занять для студентів	19
1.4 Висновки за розділом 1	22
2 Опис програми.....	24
2.1 Вибір мови програмування	24
2.2 Вибір середовища розробки.....	31
2.3 Вибір фреймворку	34
2.4 Вибір бази даних	39
2.5 Опис архітектури вебзастосунку розкладу занять студентів	43
2.6 Висновок за розділом 2	46
3 Розробка програми	46
3.1 Розробка бази даних вебзастосунку розкладу занять студентів	49
3.2 Опис файлів, що реалізують логіку вебзастосунку розкладу занять для студентів	50
3.3 Опис файлів, що реалізують інтерфейс користувача та конфігурацію вебзастосунку розкладу занять студентів	51
3.4 Висновки за розділом 3	53
4 Експлуатація, тестування та експериментальне дослідження програми	55
4.1 Призначення та умови застосування вебзастосунку	55
4.2 Опис процесу роботи з розробленим вебзастосунком розкладу занять студентів	55
4.3 Тестування роботи вебзастосунку розкладу занять студентів.....	57

4.4 Висновки за розділом 4	59
Висновки.....	60
Перелік джерел посилення	61
Додаток А Текст програми	64
Додаток Б Слайди презентації.....	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
- IDE – Integrated Development Environment;
- HTML – HyperText Markup Language;
- CSS – Cascading Style Sheets;
- HTTP – HyperText Transfer Protocol;
- JSON – JavaScript Object Notation;
- SQL – Structured Query Language;
- URL – Uniform Resource Locator;
- ПЗ – програмне забезпечення;
- СУБД – система управління базами даних.

ВСТУП

В сучасному світі існує певна тенденція прискорення процесу розвитку абсолютно в усіх сферах діяльності людини. Для цього використовується широке впровадження інформаційних технологій як основа для подальшого розвитку людства. Інформаційні технології зазнали неймовірно швидкого розвитку за останній час та пропонують користувачеві прискорення та автоматизацію багатьох задач різної складності. Насамперед цьому сприяє постійне розширення області застосування та висока ефективність інформаційних технологій в потрібних для людини сферах діяльності.

Одним із важливих аспектів навчального процесу є легкий та своєчасний доступ до актуального розкладу занять, що дозволяє ефективно планувати час навчання та своєчасно повідомляти студентам про актуальні зміни в графіку навчання, що у свою чергу призводить до підвищення загальної організованості студентів і викладачів.

Традиційні способи представлення розкладу у вигляді паперових оголошень або статичних таблиць мають свої недоліки зокрема складність оновлення інформації, відсутність оперативного доступу до розкладу та незручність використання з мобільних пристроїв. У зв'язку з цим виникає потреба у створенні вебзастосунку, який здатен забезпечити швидкий, зручний та централізований доступ до розкладу занять.

Саму тому розробка вебзастосунку розкладу занять для студентів є актуальною – інструмент для економії часу на планування навчання студентами значно спрощує освітній процес [1]-[2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття розкладу. Програмне забезпечення для вебзастосунку розкладу занять для студентів

Розклад занять є невід'ємною складовою організації освітнього процесу у будь-якому закладі освіти. Він являє собою впорядковану систему проведення навчальних занять, яка визначає послідовність вивчення дисциплін, час проведення пар, закріплення аудиторій та викладачів за певними групами студентів. Основним призначенням розкладу є забезпечення ефективного функціонування навчального процесу, координація роботи всіх його учасників та створення комфортних умов для навчання.

У сучасних закладах освіти розклад виконує не лише організаційну функцію, а й інформаційну. Завдяки правильно сформованому розкладу студенти можуть планувати власний час, готуватися до занять та ефективно розподіляти навчальне навантаження протягом тижня. Для викладачів розклад є інструментом планування навчальної діяльності та контролю виконання навчального плану. Адміністрація навчального закладу, у свою чергу, використовує розклад для організації роботи кафедр, аудиторного фонду та забезпечення безперервності освітнього процесу.

Процес створення розкладу занять є складним завданням, оскільки під час його формування необхідно враховувати велику кількість параметрів. До основних факторів належать кількість навчальних груп, навантаження викладачів, доступність аудиторій, типи занять, особливості навчальних дисциплін та санітарно-гігієнічні вимоги щодо організації навчального процесу. Наприклад, деякі дисципліни потребують спеціалізованих лабораторій або комп'ютерних класів, що суттєво ускладнює процес планування.

У багатьох навчальних закладах традиційно використовувалися паперові варіанти розкладу. Вони розміщувалися на інформаційних стендах або друкувалися у вигляді таблиць. Проте такий підхід має значну кількість

недоліків. Зокрема, будь-які зміни у розкладі потребують повторного друку документів, а студенти не завжди можуть оперативно отримати оновлену інформацію. Крім того, паперовий формат не забезпечує швидкого пошуку даних та створює труднощі у разі великої кількості навчальних груп.

З розвитком інформаційних технологій широкого поширення набули електронні системи організації розкладу. Сучасні вебзастосунки дозволяють автоматизувати процес створення, редагування та перегляду розкладу занять. Основною перевагою таких систем є можливість централізованого зберігання інформації та оперативного оновлення даних. Користувачі можуть отримувати доступ до актуального розкладу через браузер або мобільний пристрій незалежно від свого місцезнаходження.

Вебзастосунок розкладу занять є програмною системою, яка функціонує через мережу Інтернет або локальну мережу навчального закладу. Основним завданням такого застосунку є забезпечення зручного доступу до інформації про навчальний процес. На відміну від звичайних статичних вебсторінок, сучасні вебзастосунки мають інтерактивний інтерфейс та підтримують динамічне оновлення інформації.

Однією з важливих переваг вебзастосунків є доступність. Користувачу не потрібно встановлювати додаткове програмне забезпечення, оскільки для роботи достатньо браузера та доступу до Інтернету. Це значно спрощує використання системи та робить її універсальною для різних платформ і пристроїв. Завдяки цьому студенти можуть переглядати розклад як з персонального комп'ютера, так і з смартфона чи планшета.

Сучасне програмне забезпечення для розкладу занять зазвичай має клієнт-серверну архітектуру. Серверна частина відповідає за зберігання та обробку даних, а клієнтська частина забезпечує взаємодію користувача із системою через вебінтерфейс. Такий підхід дозволяє централізовано керувати інформацією та забезпечує швидке оновлення даних для всіх користувачів.

Для розробки клієнтської частини вебзастосунків використовуються технології HTML, CSS та JavaScript. HTML забезпечує структуру вебсторінок,

CSS відповідає за оформлення та дизайн інтерфейсу, а JavaScript використовується для створення інтерактивних елементів та динамічного оновлення інформації.

Для зберігання інформації у вебзастосунку використовуються системи керування базами даних. У базі даних можуть зберігатися відомості про студентські групи, викладачів, аудиторії, дисципліни, час проведення занять та інші параметри навчального процесу.

Функціонал програмного забезпечення для вебзастосунку розкладу занять можна поділити на користувацький та адміністративний. Користувацький функціонал орієнтований на студентів та викладачів, які використовують систему для перегляду розкладу. Адміністративний функціонал призначений для працівників, які займаються формуванням та редагуванням розкладу.

Основною функцією користувацької частини є перегляд розкладу занять. Користувач має можливість обрати свою навчальну групу та переглянути перелік занять на певний день або на весь тиждень. Інформація зазвичай відображається у вигляді таблиці, де вказуються назви дисциплін, час проведення занять, аудиторії та прізвища викладачів.

Для підвищення зручності використання система може підтримувати функцію фільтрації даних. Наприклад, користувач може переглядати розклад лише певного викладача або конкретної аудиторії. Це дозволяє швидше знаходити потрібну інформацію та значно спрощує роботу із системою.

Важливим елементом сучасного вебзастосунку є система автоматичного оновлення інформації. Після внесення змін адміністратором дані автоматично зберігаються у базі даних та стають доступними для користувачів. У деяких системах оновлення відбувається без перезавантаження сторінки завдяки використанню AJAX-запитів або WebSocket-з'єднань.

Особливе значення для сучасних вебзастосунків має адаптивний дизайн. Оскільки більшість користувачів переглядає інформацію за

допомогою мобільних пристроїв, інтерфейс системи повинен коректно відображатися на екранах різних розмірів. Адаптивний інтерфейс автоматично підлаштовується під параметри пристрою користувача, забезпечуючи комфортне користування системою.

Крім основного функціоналу, вебзастосунок може містити додаткові можливості, які покращують взаємодію користувачів із системою. Однією з таких функцій є система сповіщень та нагадувань. Вона дозволяє автоматично інформувати студентів про початок занять або зміни у розкладі. Повідомлення можуть надсилатися у вигляді push-сповіщень, електронних листів або повідомлень у самому вебзастосунку.

Система сповіщень є особливо корисною у випадках термінових змін розкладу. Наприклад, якщо заняття переноситься в іншу аудиторію або змінюється час проведення пари, студенти можуть оперативно отримати відповідне повідомлення. Це дозволяє уникнути непорозумінь та покращує організацію навчального процесу.

Адміністративний функціонал вебзастосунку забезпечує керування всією системою. Адміністратор має можливість створювати нові записи розкладу, редагувати існуючі заняття, видаляти застарілі дані та контролювати правильність інформації. Усі зміни одразу зберігаються у базі даних та відображаються користувачам.

Крім редагування розкладу, адміністратор може керувати списками студентських груп, викладачів, аудиторій та навчальних дисциплін. Це дозволяє підтримувати актуальність інформації та забезпечує централізоване адміністрування системи.

Для забезпечення безпеки вебзастосунку використовується система авторизації та автентифікації користувачів. Доступ до адміністративної панелі мають лише користувачі з відповідними правами доступу. Це дозволяє захистити систему від несанкціонованого редагування інформації та забезпечує цілісність даних.

Важливим аспектом роботи вебзастосунку є продуктивність та надійність системи. Оскільки одночасно до системи може звертатися велика кількість користувачів, необхідно забезпечити швидке завантаження сторінок та стабільну роботу сервера. Для цього можуть використовуватися технології кешування даних, оптимізації запитів до бази даних та балансування навантаження.

Також важливою складовою є резервне копіювання даних. Регулярне створення резервних копій дозволяє уникнути втрати інформації у разі технічних несправностей або помилок користувачів. Це забезпечує надійність роботи системи та безперервність доступу до навчальної інформації.

Сучасні вебзастосунки для розкладу занять можуть інтегруватися з іншими інформаційними системами навчального закладу. Наприклад, система може взаємодіяти з електронним журналом, платформою дистанційного навчання або системою обліку студентів. Така інтеграція дозволяє створити єдиний інформаційний простір закладу освіти та підвищити ефективність управління навчальним процесом.

Таким чином, програмне забезпечення для вебзастосунку розкладу занять є важливим елементом сучасної інформаційної інфраструктури закладу освіти. Використання вебтехнологій дозволяє автоматизувати процес роботи з розкладом, забезпечити швидкий доступ до актуальної інформації та підвищити ефективність організації навчального процесу. Завдяки широкому функціоналу, адаптивному інтерфейсу та можливості інтеграції з іншими системами вебзастосунки розкладу занять значно покращують взаємодію між студентами, викладачами та адміністрацією навчального закладу [3].

1.2 Огляд вебзастосунків розкладу занять

Розглянемо та проаналізуємо сучасні вебзастосунки розкладу занять для студентів, які використовуються у закладах вищої освіти. Аналіз існуючих систем дозволяє визначити їх основні переваги, недоліки та функціональні

договори. Така інтеграція декількох функцій у межах однієї системи створює єдине інформаційне середовище для організації освітнього процесу.

Однією з головних переваг вебзастосунку portal.zp.edu.ua є централізований доступ до інформації. Система дозволяє переглядати розклад як за окремим студентом, так і за навчальною групою. Це значно спрощує процес пошуку необхідних даних та дозволяє швидко отримати актуальну інформацію щодо занять.

Ще однією перевагою є доступність через веббраузер. Для роботи з порталом не потрібно встановлювати додаткове програмне забезпечення, оскільки доступ до системи здійснюється через мережу Інтернет. Такий підхід забезпечує універсальність використання вебзастосунку на різних операційних системах та пристроях.

Важливим елементом системи є механізм авторизації користувачів. Авторизовані студенти мають доступ до персоналізованої інформації, зокрема до власного індивідуального розкладу, оцінок та навчальних документів. Це дозволяє захистити персональні дані користувачів та забезпечує безпечну роботу з системою.

Незважаючи на значний функціонал, вебзастосунок portal.zp.edu.ua має певні недоліки. Одним із них є обмежена адаптація для мобільних пристроїв. Під час використання системи на смартфонах або планшетах окремі елементи інтерфейсу можуть відображатися некоректно або бути недостатньо зручними для користувача. Це ускладнює роботу з системою на пристроях з невеликим розміром екрана.

Ще одним недоліком є нестабільна робота серверів при високому навантаженні. Під час масового звернення користувачів до системи, особливо на початку навчального семестру або під час оновлення розкладу, доступ до вебзастосунку може бути тимчасово обмежений. Це негативно впливає на зручність використання системи та швидкість отримання інформації.

Також певною проблемою є складність інтерфейсу. Через велику кількість розділів та функцій новим користувачам може бути складно швидко

знайти потрібну інформацію. Надмірна кількість елементів інтерфейсу створює додаткове навантаження на користувача та ускладнює навігацію.

Крім того, вебзастосунок не підтримує систему push-сповіщень або автоматичних нагадувань про початок занять чи зміни у розкладі. У результаті студенти змушені самостійно перевіряти розклад на наявність оновлень, що не завжди є зручним.

Ще одним вебзастосунком розкладу занять для студентів є schedule.kpi.ua (рис. 1.2) – Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» [5].

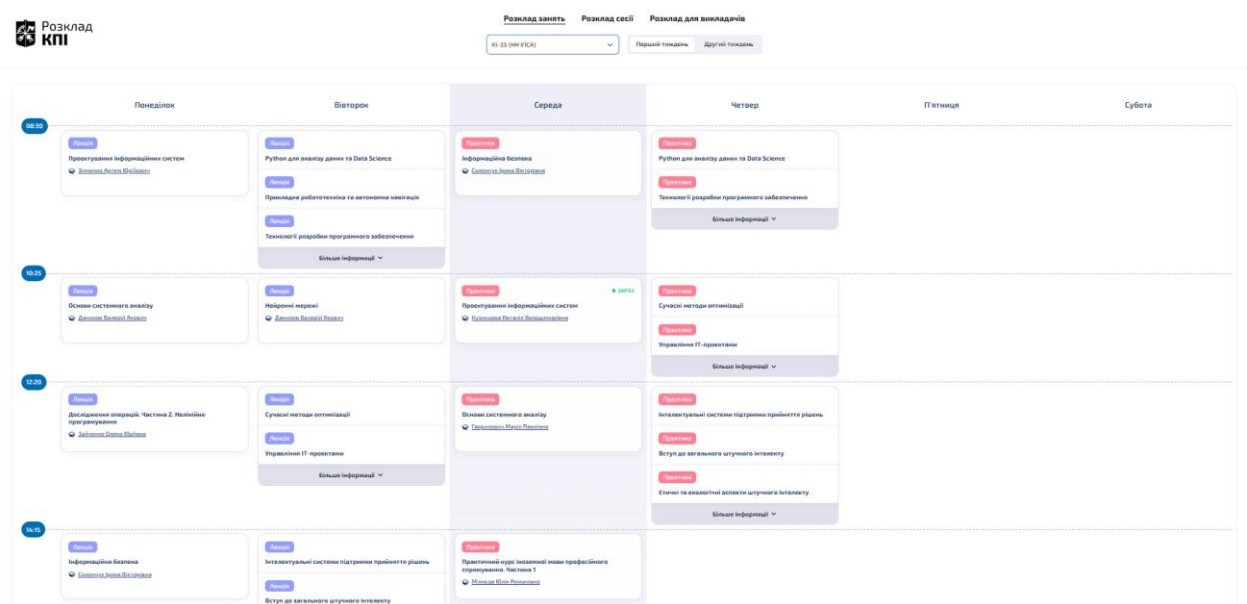


Рисунок 1.2 – Інтерфейс сторінки розкладу занять вебзастосунку schedule.kpi.ua [5]

Однією з головних переваг вебзастосунку schedule.kpi.ua є простий та зрозумілий інтерфейс. Система не перевантажена зайвими елементами, що забезпечує швидку навігацію та комфортний перегляд інформації. Основна увага приділена безпосередньо розкладу занять, завдяки чому користувач може швидко знайти необхідні дані.

Також позитивною особливістю системи є підтримка пошуку за викладачем. Це дозволяє переглядати розклад не лише студентських груп, а й

окремих викладачів. Такий функціонал є корисним як для студентів, так і для працівників університету, оскільки спрощує організацію навчального процесу.

Ще однією важливою перевагою є адаптація інтерфейсу для мобільних пристроїв. Усі елементи вебзастосунку автоматично підлаштовуються під розмір екрана, що забезпечує коректне відображення інформації на смартфонах та планшетах. Це значно підвищує зручність використання системи у повсякденному житті студентів.

Проте система schedule.kpi.ua також має певні недоліки. Зокрема, вебзастосунок не підтримує push-сповіщення або автоматичні нагадування про початок занять та зміни у розкладі. Через це користувачі повинні самостійно перевіряти інформацію щодо оновлень.

Ще одним недоліком є незручний пошук навчальних груп. Через велику кількість академічних груп у закладі освіти користувачеві може бути складно швидко знайти потрібну групу. У системі відсутній повноцінний пошук за назвою групи, що ускладнює навігацію та збільшує час пошуку інформації.

Порівнюючи розглянуті вебзастосунки, можна зробити висновок, що обидві системи мають спільну мету – забезпечення доступу до розкладу занять та організацію навчального процесу. Проте вони реалізують цю задачу різними способами. Вебзастосунок portal.zp.edu.ua орієнтований на створення комплексної інформаційної системи університету, яка поєднує різноманітні сервіси для студентів. У свою чергу, schedule.kpi.ua більше зосереджений на простоті використання та швидкому доступі безпосередньо до розкладу.

Проведений аналіз дозволяє визначити основні вимоги до сучасного вебзастосунку розкладу занять. Насамперед система повинна забезпечувати простий та зрозумілий інтерфейс, підтримувати адаптацію для мобільних пристроїв, швидкий пошук необхідної інформації та автоматичне оновлення даних. Важливою функцією також є система сповіщень про зміни у розкладі та нагадування про початок занять.

Крім того, вебзастосунок повинен забезпечувати стабільну роботу навіть при високому навантаженні та підтримувати механізми захисту персональних даних користувачів. Реалізація авторизації, розмежування прав доступу та безпечного зберігання інформації є необхідними складовими сучасної інформаційної системи.

Отже, аналіз існуючих вебзастосунків розкладу занять дозволив визначити їх сильні та слабкі сторони. Отримані результати можуть бути використані під час проєктування та реалізації власного вебзастосунку розкладу занять для студентів, що дозволить створити більш зручну, функціональну та сучасну систему організації навчального процесу.

1.3 Порівняння вебзастосунків розкладу занять для студентів

Для визначення основних переваг та недоліків існуючих рішень було проведено порівняльний аналіз вебзастосунків розкладу занять portal.zp.edu.ua та schedule.kpi.ua із вебзастосунком, що розробляється в межах даної роботи. Проведення такого аналізу дозволяє визначити найбільш важливі функціональні можливості системи, оцінити зручність використання аналогів та сформулювати перелік вимог до програмного продукту, який розробляється.

Порівняння вебзастосунків здійснювалося за основними критеріями, що безпосередньо впливають на зручність використання системи студентами чи викладачами та ефективність організації навчального процесу. До таких критеріїв належать доступність перегляду розкладу на різних пристроях, наявність системи повідомлень та нагадувань про початок занять чи інші події, адаптивність інтерфейсу до будь-якої ширини екрану пристрою, стабільність роботи серверної частини при сильних навантаженнях від користувачів та можливість пошуку інформації за різними критеріями: групами, аудиторіями чи викладачами.

Таблиця 1.1 – Порівняння вебзастосунків розкладу занять для студентів

Критерій порівняння	portal.zp.edu.ua	schedule.kpi.ua	Розробка
Доступність до перегляду на пристроях	+	+	+
Наявність повідомлень, нагадувань про початок занять або зміну в розкладі	-	-	+
Авторизація користувачів	+	-	-
Адаптивний та зрозумілий інтерфейс	-	+	+
Стабільна робота серверів	-	+	+-
Можливість пошуку за групою чи вчителями	+	+	+

Підводячі підсумки до таблиці 1.1, можна зробити висновок, що обидва існуючі вебзастосунки мають як переваги, так і певні недоліки. Вебзастосунок portal.zp.edu.ua характеризується широким функціоналом та наявністю авторизації користувачів, що дозволяє забезпечити персоналізований доступ до інформації. Проте система має складний інтерфейс та недостатню адаптацію для мобільних пристроїв, що ускладнює її використання на смартфонах та планшетах.

Водночас вебзастосунок schedule.kpi.ua має більш простий та зрозумілий інтерфейс, а також добре адаптований для мобільних пристроїв. Це забезпечує зручний доступ до розкладу незалежно від типу пристрою. Однак система не підтримує авторизацію користувачів та не має функції автоматичних повідомлень або нагадувань про початок занять чи зміни у розкладі.

Під час розробки власного вебзастосунку особливу увагу приділено реалізації функцій, яких бракує існуючим аналогам. Зокрема, однією з основних переваг розроблюваної системи є підтримка повідомлень та

нагадувань про початок занять або зміни у розкладі. Така функція дозволяє підвищити зручність використання системи та забезпечує своєчасне інформування студентів.

Крім цього, у вебзастосунку, що розробляється, передбачається реалізація адаптивного інтерфейсу. Це дозволить забезпечити коректне відображення інформації на пристроях із різними розмірами екрана. Завдяки використанню адаптивного дизайну користувачі зможуть комфортно працювати із системою як на персональних комп'ютерах, так і на мобільних пристроях.

Важливою складовою системи є також можливість швидкого пошуку інформації. У розроблюваному вебзастосунку передбачено реалізацію пошуку за навчальними групами. Це значно полегшує навігацію системою та дозволяє користувачам швидко знаходити необхідний розклад.

Порівняльний аналіз також показав важливість стабільної роботи серверної частини вебзастосунку. Оскільки системою одночасно може користуватися значна кількість студентів, сервер повинен забезпечувати швидке оброблення запитів та безперервний доступ до інформації. Саме тому під час розробки вебзастосунку необхідно враховувати питання оптимізації роботи сервера та бази даних.

Ще одним важливим аспектом є простота та зрозумілість інтерфейсу. Складний або перевантажений інтерфейс негативно впливає на швидкість роботи користувачів із системою. Тому під час розробки вебзастосунку особлива увага приділяється створенню інтуїтивно зрозумілої структури сторінок, зручного меню навігації та логічного розташування елементів інтерфейсу.

Підводячи підсумки до таблиці 1.1, можна стверджувати, що під час створення вебзастосунку розкладу занять для студентів необхідно реалізувати такі основні функціональні вимоги:

- можливість пошуку за групами чи викладачами, що значно спрощує пошук необхідної інформації;

- зрозумілий та адаптивний інтерфейс, який забезпечує комфортну роботу користувачів на різних пристроях;
- систему повідомлень та нагадувань про початок занять або зміни у розкладі;
- швидке оновлення інформації у системі;
- стабільну роботу вебзастосунку навіть при значному навантаженні;
- зручну структуру навігації та швидкий доступ до основних функцій системи.

Таким чином, проведений порівняльний аналіз дозволив визначити ключові характеристики сучасного вебзастосунку розкладу занять та сформувавши перелік функціональних вимог до системи, що розробляється у даній роботі. Отримані результати будуть використані на наступних етапах проєктування та програмної реалізації вебзастосунку.

1.4 Висновки за розділом 1

У першому розділі дипломної роботи було проведено аналіз предметної області, пов'язаної з організацією розкладу занять для студентів у закладах освіти. Розглянуто поняття розкладу занять, його значення для організації навчального процесу та основні особливості сучасних інформаційних систем, що використовуються для автоматизації роботи з розкладом.

У ході дослідження встановлено, що використання вебзастосунків для організації розкладу занять значно підвищує ефективність навчального процесу, забезпечує швидкий доступ до актуальної інформації та спрощує взаємодію між студентами, викладачами та адміністрацією навчального закладу. Також було визначено основні переваги вебтехнологій, серед яких доступність через браузер, централізоване зберігання даних, можливість автоматичного оновлення інформації та підтримка роботи на різних типах пристроїв.

У межах аналізу існуючих рішень було досліджено вебзастосунки portal.zp.edu.ua та schedule.kpi.ua. Для кожної системи було визначено основні функціональні можливості, переваги та недоліки. Проведений аналіз показав, що сучасні вебзастосунки розкладу занять забезпечують базові функції перегляду інформації, проте мають певні недоліки, зокрема відсутність системи повідомлень, недостатню адаптацію інтерфейсу або складність навігації.

Отже, результати проведеного аналізу підтверджують актуальність розробки вебзастосунку розкладу занять для студентів та створюють основу для подальшого проєктування й програмної реалізації системи.

2 ОПИС ПРОГРАМИ

2.1 Вибір мови програмування

Перед початком аналізу основних мов програмування, що можуть бути використані для розробки вебзастосунку розкладу занять для студентів, варто визначити найважливіші параметри, яким повинна відповідати обрана мова програмування. Правильний вибір технологій є одним із ключових етапів розробки програмного забезпечення, оскільки саме від цього залежить швидкість створення системи, її продуктивність, надійність та можливість подальшого розвитку.

Під час розробки вебзастосунку необхідно враховувати не лише поточні потреби системи, а й перспективи її масштабування та модернізації у майбутньому. Сучасні вебзастосунки повинні забезпечувати стабільну роботу при великій кількості одночасних користувачів, підтримувати інтеграцію з базами даних та мати можливість адаптації до різних типів пристроїв. Саме тому вибір мови програмування та супутніх технологій має здійснюватися на основі комплексного аналізу їх можливостей. Визначемо найважливіші параметри, яким повинна відповідати обрана мова програмування:

- простий та зрозумілий синтаксис: процес написання програмного коду має бути максимально зручним для розробника, а структура коду — легкою для читання та підтримки. Простота синтаксису дозволяє зменшити кількість помилок під час програмування та полегшує процес супроводу програмного забезпечення у майбутньому. Крім того, зрозумілий код спрощує командну розробку, оскільки інші програмісти можуть швидше ознайомитися з проєктом та вносити зміни у систему. Під час створення вебзастосунку розробник повинен зосереджувати увагу насамперед на реалізації функціональних можливостей системи, а не на складних синтаксичних конструкціях мови програмування. Якщо мова програмування має надто складний синтаксис або вимагає великої кількості додаткового коду для

виконання простих операцій, це уповільнює процес розробки та ускладнює підтримку системи;

- ефективне використання ресурсів серверної частини: вебзастосунок розкладу занять повинен забезпечувати стабільну роботу навіть при великій кількості одночасних запитів від користувачів. Особливо це актуально у періоди високого навантаження, наприклад на початку навчального семестру або під час внесення змін до розкладу. Обрана мова програмування повинна забезпечувати оптимальне використання оперативної пам'яті та процесорних ресурсів сервера. Це дозволяє підтримувати швидке завантаження сторінок, оперативну обробку запитів та безперебійну роботу системи у цілодобовому режимі. Низька продуктивність серверної частини може призводити до затримок у роботі вебзастосунку, помилок при завантаженні сторінок та погіршення користувацького досвіду;

- підтримка сучасних фреймворків для веброзробки: фреймворки являють собою готові програмні платформи, які містять набір інструментів та бібліотек для реалізації типових задач веброзробки. Використання фреймворків значно пришвидшує процес створення програмного забезпечення та дозволяє зосередитися на реалізації бізнес-логіки системи. Сучасні фреймворки забезпечують готові механізми для роботи з базами даних, маршрутизації HTTP-запитів, створення API, авторизації користувачів та захисту даних. Крім цього, вони допомагають дотримуватися єдиної структури проекту та спрощують підтримку програмного коду. Завдяки використанню фреймворків можна зменшити час розробки вебзастосунку та підвищити його надійність;

- підтримка роботи з базами даних: Вебзастосунок розкладу занять повинен зберігати значний обсяг інформації, зокрема дані про студентські групи, викладачів, аудиторії, навчальні дисципліни та розклад проведення занять. Для забезпечення швидкого доступу до цих даних необхідно використовувати системи керування базами даних;

– можливість створення адаптивного веб інтерфейсу: У сучасних умовах значна частина користувачів переглядає інформацію за допомогою мобільних пристроїв, тому вебзастосунок повинен коректно працювати на смартфонах, планшетах та персональних комп'ютерах. Обрані технології повинні забезпечувати створення сучасного інтерфейсу із підтримкою адаптивного дизайну. Адаптивний інтерфейс автоматично підлаштовується під розміри екрана користувача, забезпечуючи комфортне відображення інформації незалежно від типу пристрою. Для реалізації такого підходу часто використовуються HTML, CSS та JavaScript разом із сучасними бібліотеками та фреймворками інтерфейсу;

– підтримка масштабування та модернізації: Під час експлуатації вебзастосунку можуть виникати потреби у додаванні нових функцій, розширенні можливостей системи або інтеграції з іншими інформаційними ресурсами навчального закладу. Система повинна мати можливість подальшого розвитку без необхідності повного переписування програмного коду. Саме тому обрана мова програмування повинна підтримувати модульний підхід до розробки, що дозволяє легко додавати нові компоненти та оновлювати існуючі модулі системи [6]-[9].

Крім цього, важливим фактором є наявність великої спільноти розробників та якісної документації. Популярні мови програмування мають значну кількість навчальних матеріалів, бібліотек та прикладів реалізації різних функцій. Це значно спрощує процес розробки та дозволяє швидше вирішувати технічні проблеми, які можуть виникати під час створення вебзастосунку.

Також необхідно враховувати питання безпеки програмного забезпечення. Оскільки вебзастосунок працює з інформацією про користувачів та навчальний процес, важливо забезпечити захист даних від несанкціонованого доступу. Обрана мова програмування та використовувані фреймворки повинні підтримувати сучасні механізми захисту, зокрема авторизацію користувачів, шифрування даних та захист від типових вебзагроз.

Отже, після аналізу наведених вище вимог до вибору мови програмування було відібрано три основні кандидати для розробки вебзастосунку розкладу занять для студентів – PHP, Python та JavaScript. Дані мови програмування є одними з найбільш поширених у сфері веброзробки та мають значну кількість інструментів для створення сучасних вебзастосунків. Кожна з них має власні переваги, недоліки та особливості використання, які необхідно враховувати під час вибору технологій для реалізації системи.

PHP є серверною мовою програмування сценаріїв, яка широко використовується для створення вебзастосунків та динамічних вебсайтів. Основною особливістю PHP є виконання програмного коду на сервері з подальшим формуванням HTML- сторінок, які передаються користувачу через браузер. Завдяки цьому PHP дозволяє створювати динамічні вебсистеми з підтримкою баз даних, авторизації користувачів та взаємодії з серверною частиною.

Мова PHP підтримує процедурне, функціональне та об'єктно-орієнтоване програмування, що забезпечує гнучкість під час створення програмного забезпечення. Завдяки простому синтаксису PHP є достатньо зручною для вивчення та використання, особливо під час розробки вебзастосунків невеликого та середнього масштабу. Також важливою перевагою PHP є невисокий рівень споживання серверних ресурсів, що дозволяє забезпечити стабільну роботу вебзастосунків навіть на серверах із обмеженими технічними характеристиками.

Ще однією важливою перевагою PHP є наявність великої кількості сучасних фреймворків, які значно спрощують процес розробки. Серед найбільш популярних фреймворків можна виділити Laravel та Symfony. Вони надають готові інструменти для роботи з базами даних, маршрутизації HTTP-запитів, реалізації авторизації користувачів та створення API. Використання фреймворків дозволяє підвищити безпеку вебзастосунку, зменшити кількість програмного коду та пришвидшити процес розробки. Попри значну популярність PHP, дана мова має певні недоліки. Зокрема, через тривалий

розвиток мови у різні періоди її існування в PHP сформувалася певна неоднорідність синтаксису. Це може ускладнювати підтримку великих проєктів та створювати труднощі під час командної розробки. Крім цього, сучасні вебзастосунки часто потребують складної асинхронної взаємодії між клієнтом і сервером, що реалізується в PHP менш зручно порівняно з JavaScript [10]–[11].

Python є інтерпретованою мовою програмування загального призначення з динамічною типізацією, яка підтримує об'єктно-орієнтовану, функціональну та процедурну парадигми програмування. Однією з головних особливостей Python є простий та зрозумілий синтаксис, який значно полегшує процес написання та підтримки програмного коду. Саме тому Python часто використовується як у професійній розробці, так і в навчальних цілях.

Python активно застосовується у веброботці, машинному навчанні, аналізі даних, автоматизації процесів та створенні серверних систем. Для веброботки Python пропонує потужні фреймворки, серед яких найбільш популярними є Django та Flask. Django є повноцінним фреймворком для створення масштабних вебзастосунків та містить велику кількість готових компонентів, включаючи систему авторизації, адміністративну панель та ORM для роботи з базами даних. Flask, у свою чергу, є більш легким та гнучким фреймворком, який дозволяє створювати невеликі вебзастосунки та API.

Перевагою Python є висока швидкість розробки програмного забезпечення. Завдяки простому синтаксису та великій кількості бібліотек розробник може швидко реалізовувати функціональні можливості системи. Також Python має велику спільноту розробників та значну кількість навчальних матеріалів, що спрощує процес вивчення та використання мови.

Проте Python має і певні недоліки. Одним із головних недоліків є більше споживання серверних ресурсів порівняно з PHP. Через особливості роботи інтерпретатора Python вебзастосунки можуть вимагати більшої кількості оперативної пам'яті та процесорних ресурсів. Це може негативно

впливати на продуктивність системи при великій кількості одночасних користувачів [12].

JavaScript є інтерпретованою мовою програмування з динамічною типізацією, яка підтримує імперативну, об'єктно-орієнтовану, функціональну та асинхронну парадигми програмування. Спочатку JavaScript використовувався переважно для створення інтерактивності на вебсторінках, проте з розвитком технологій він став повноцінною мовою для розробки як клієнтської, так і серверної частини вебзастосунків.

Однією з головних переваг JavaScript є можливість використання однієї мови програмування як для frontend-, так і для backend-розробки. Для створення серверної частини широко використовується платформа Node.js, яка дозволяє виконувати JavaScript-код на сервері. Це значно спрощує процес розробки, оскільки розробникам не потрібно використовувати декілька мов програмування для реалізації різних частин системи.

Мова програмування JavaScript також має значну кількість сучасних фреймворків та бібліотек для створення вебзастосунків. Для клієнтської частини часто використовуються React, Vue.js та Angular, які забезпечують створення швидких та адаптивних користувацьких інтерфейсів. Для серверної частини можуть використовуватися Express.js, NestJS та інші фреймворки, які дозволяють створювати API та реалізовувати бізнес-логіку системи.

Важливою перевагою JavaScript є підтримка асинхронного програмування. Завдяки використанню механізмів Promise, async/await та неблокуючої моделі виконання Node.js система може ефективно обробляти велику кількість одночасних запитів користувачів. Це є особливо важливим для вебзастосунків, які повинні забезпечувати швидку взаємодію з користувачами та оперативне оновлення інформації [13].

Зведені результати порівняння мов програмування, що розглядаються для розробки вебзастосунку розкладу занять студентів наведені в табл. 2.1.

Таблиця 2.1 – Вибір мови розробки вебзастосунку розкладу занять

Критерій для порівняння	PHP	Python	JavaScript
Синтаксис	+-	+	+-
Вимоги в споживанні ресурсів серверної частини	+	-	+-
Підтримка фреймворків для створення вебзастосунків	+-	+	+

Аналізуючи результати таблиці 2.1, можна зробити висновок, що кожна з розглянутих мов програмування має власні переваги та може використовуватися для створення вебзастосунків. PHP характеризується низькими вимогами до серверних ресурсів та простотою розгортання вебсистем. Python забезпечує високу швидкість розробки та має зручний синтаксис, проте потребує більшої кількості ресурсів сервера. JavaScript, у свою чергу, забезпечує можливість використання єдиної мови програмування для клієнтської та серверної частини системи, підтримує асинхронну обробку запитів та має велику кількість сучасних інструментів для веброзробки.

Можна зробити висновок, що JavaScript є оптимальним вибором для розробки вебзастосунку розкладу занять для студентів. Це обумовлено можливістю використання однієї мови програмування для клієнтської та серверної частини, підтримкою асинхронної обробки запитів, високою швидкістю роботи та широким набором сучасних технологій для створення вебзастосунків.

Використання JavaScript дозволяє реалізувати адаптивний користувацький інтерфейс, забезпечити швидке оновлення інформації про розклад та створити ефективну серверну частину системи. Крім цього, велика кількість бібліотек, фреймворків та активна спільнота розробників роблять JavaScript перспективним вибором для подальшого розвитку та модернізації вебзастосунку.

2.2 Вибір середовища розробки

Обираючи середовище розробки для створення вебзастосунку розкладу занять для студентів, перш за все головним критерієм є підтримка обраної мови програмування JavaScript. Також повинна бути підтримка фреймворків до мови програмування, швидкість роботи середовища та простота використання.

Переглянувши визначені критерії до потенційного середовища розробки, обрано два потенційних кандидати – Visual Studio Code та WebStorm.

Visual Studio Code – це сучасне безкоштовне середовище розробки програмного забезпечення, створене компанією Microsoft . Дане середовище призначене для написання, редагування, тестування та налагодження програмного коду. Visual Studio Code підтримує велику кількість мов програмування та активно використовується для веброзробки, створення серверних застосунків, мобільних додатків та інших програмних продуктів. Visual Studio Code працює на операційних системах Windows, Linux та macOS, що робить його універсальним інструментом для розробників. Середовище має простий, сучасний та інтуїтивно зрозумілий інтерфейс, який дозволяє швидко освоїти основні функції навіть користувачам без великого досвіду програмування. Однією з головних переваг Visual Studio Code є висока продуктивність та невеликі вимоги до апаратних ресурсів комп'ютера. На відміну від багатьох повноцінних IDE, Visual Studio Code запускається швидко та стабільно працює навіть на пристроях із невеликою кількістю оперативної пам'яті.

Важливою особливістю Visual Studio Code є система розширень. За допомогою офіційного Marketplace користувач може встановлювати додаткові модулі для підтримки різних мов програмування, автоматичного форматування коду, роботи з базами даних, тестування та налагодження програмного забезпечення.

Visual Studio Code також має вбудовану підтримку системи контролю версій Git. Це дозволяє створювати репозиторії, відстежувати зміни в коді, виконувати коміти та працювати з GitHub без використання сторонніх програм. Ще однією важливою функцією є інтегрований термінал. Завдяки цьому розробник може запускати сервер за допомогою різних фреймворків, виконувати команди Git та працювати із системою безпосередньо в середовищі розробки [14]-[15].

WebStorm є професійним інтегрованим середовищем розробки, створеним компанією JetBrains. Дане середовище спеціально орієнтоване на веброботу та забезпечує підтримку JavaScript, TypeScript, HTML, CSS та сучасних вебфреймворків.

На відміну від Visual Studio Code, WebStorm є повноцінною IDE, яка містить значну кількість інструментів уже після встановлення. Це дозволяє розпочати розробку без необхідності встановлення великої кількості додаткових розширень. WebStorm особливо ефективний під час створення складних та масштабних вебпроектів.

Однією з головних переваг WebStorm є потужна підтримка JavaScript та TypeScript. Середовище дозволяє автоматично виявляти помилки у коді ще до запуску програми, що значно зменшує кількість помилок під час розробки. Крім цього, WebStorm підтримує інтелектуальне автодоповнення коду, автоматичну перевірку синтаксису та аналіз структури проекту.

Важливою функцією WebStorm є розширений рефакторинг коду. Це дозволяє безпечно змінювати структуру програмного забезпечення, автоматично оновлюючи залежності між файлами та компонентами системи. Такий підхід є особливо корисним під час розробки великих вебзастосунків, які містять значну кількість модулів.

WebStorm також має вбудовану підтримку популярних фреймворків та бібліотек, зокрема React, Vue.js, Angular та Node.js. Середовище підтримує інтеграцію з системою Git, засобами тестування та інструментами для налагодження вебзастосунків.

Попри значну кількість переваг, WebStorm має і певні недоліки. Одним із головних є те, що середовище є платним програмним продуктом. Крім цього, через велику кількість вбудованих функцій WebStorm має вищі вимоги до апаратних ресурсів комп'ютера та може працювати повільніше на слабких пристроях [16].

Зведені результати порівняння середовищ розробки, що розглядаються для створення вебзастосунку розкладу занять наведені в табл. 2.2.

Таблиця 2.2 – Вибір середовища розробки для вебзастосунку розкладу занять студентів

Критерій для порівняння	Visual Studio Code	WebStorm
Безкоштовний	+	-
Підтримка фреймворків	+	+
Швидкість роботи	+	-
Рефакторинг коду	+ -	+
Підтримка Git	+	+
Популярність серед розробників	+	-

Аналізуючи результати таблиці 2.2, можна зробити висновок, що обидва середовища розробки забезпечують підтримку сучасних вебтехнологій та можуть використовуватися для створення вебзастосунку розкладу занять. WebStorm має потужні вбудовані інструменти для аналізу та рефакторингу коду, проте потребує більшої кількості системних ресурсів та є платним програмним продуктом. Visual Studio Code, у свою чергу, характеризується високою швидкістю роботи, простотою використання та великою кількістю доступних розширень. Завдяки підтримці JavaScript, Node.js та сучасних вебфреймворків дане середовище є ефективним інструментом для створення вебзастосунків.

Отже, було зроблено висновок, що для розробки вебзастосунку розкладу занять доцільно обрати Visual Studio Code. Це обумовлено тим, що дане середовище є безкоштовним, швидким та зручним у використанні, а також має велику кількість розширень і підтримку сучасних вебтехнологій.

Використання Visual Studio Code дозволить забезпечити комфортний процес розробки, підтримку роботи з JavaScript та сучасними фреймворками, а також ефективну взаємодію із системою контролю версій Git. Крім цього, невисокі вимоги до апаратних ресурсів роблять Visual Studio Code оптимальним вибором для створення вебзастосунку розкладу занять для студентів.

2.3 Вибір фреймворку

Перед початком аналізу фреймворків, що можуть бути використані для розробки вебзастосунку розкладу занять, варто визначитися з найважливішими параметрами, яким має відповідати обраний фреймворк. Відповідно до пункту 2.1, основним критерієм вибору вебфреймворку є його сумісність із мовою програмування JavaScript. Оскільки саме JavaScript було обрано основною мовою програмування для створення вебзастосунку, необхідно використовувати технології, які максимально ефективно підтримують дану мову та забезпечують стабільну роботу системи.

Правильний вибір фреймворку або середовища виконання безпосередньо впливає на швидкість розробки, продуктивність серверної частини, масштабованість системи та можливість її подальшої модернізації. Крім цього, сучасні вебзастосунки повинні підтримувати обробку великої кількості одночасних запитів, забезпечувати швидкий обмін даними між клієнтом та сервером, а також мати гнучку структуру для інтеграції додаткових модулів.

Важливі критерії для вибору вебфреймворку:

- продуктивність: обраний фреймворк повинен забезпечувати ефективну обробку великої кількості запитів та раціональне використання ресурсів серверної частини вебзастосунку. Основною вимогою до продуктивності є підтримка масштабованості, що дозволяє системі стабільно працювати навіть при зростанні навантаження, зокрема збільшенні кількості користувачів, запитів та обсягів оброблюваних даних. Важливо, щоб фреймворк підтримував асинхронну обробку запитів та оптимізовану роботу з серверними ресурсами;

- швидкість розробки: фреймворк повинен містити набір вбудованих інструментів і модулів, які дозволяють прискорити процес створення вебзастосунку. Це включає готові рішення для маршрутизації, роботи з базами даних, авторизації користувачів, обробки HTTP-запитів та створення API. Наявність таких інструментів зменшує обсяг рутинного коду та скорочує час розробки програмного продукту;

- підтримка пакетів розширення функціоналу: обраний фреймворк повинен мати розвинену екосистему додаткових пакетів і бібліотек, які можна використовувати для розширення функціональності вебзастосунку. Це особливо важливо для реалізації додаткових можливостей системи розкладу занять, таких як сповіщення користувачів, інтеграція з базами даних, кешування та забезпечення безпеки даних;

- простота використання та читабельність коду: фреймворк повинен мати зрозумілу структуру проекту та логічну організацію коду, що спрощує підтримку та подальший розвиток вебзастосунку;

- підтримка спільноти та документації: важливим фактором є наявність великої спільноти розробників та детальної документації, що дозволяє швидко знаходити рішення типових задач та уникати помилок під час розробки.

Після початкового перегляду визначених критеріїв, було обрано три потенційні фреймворки для використання – Node.js та Deno та Bun.

Node.js – це серверне середовище виконання JavaScript, побудоване на рушії V8 від браузера Google Chrome. Воно дозволяє запускати JavaScript-код поза браузером та використовувати його для створення серверної частини вебзастосунків, API, мікросервісів та інших backend-рішень.

Однією з головних особливостей Node.js є асинхронна подієво-орієнтована архітектура. Завдяки використанню неблокуючої моделі введення/виведення система може ефективно обробляти велику кількість одночасних запитів без блокування основного потоку виконання. Це робить Node.js особливо ефективним для створення високонавантажених вебзастосунків та сервісів реального часу.

Важливою перевагою Node.js є висока продуктивність. Завдяки використанню рушія V8 JavaScript-код компілюється у машинний код та виконується з високою швидкістю. Крім цього, Node.js забезпечує ефективне використання серверних ресурсів та підтримує масштабування вебзастосунків.

Node.js також має одну з найбільших екосистем бібліотек серед сучасних вебтехнологій. Менеджер пакетів npm містить сотні тисяч готових модулів для реалізації різних функцій вебзастосунку. Це дозволяє значно пришвидшити процес розробки та зменшити обсяг програмного коду, який необхідно створювати вручну.

Для створення вебзастосунків на Node.js часто використовуються такі популярні фреймворки, як Express.js, NestJS, Fastify та Koa. Вони забезпечують готові рішення для маршрутизації HTTP-запитів, створення API, роботи з базами даних та реалізації систем авторизації користувачів [17].

Deno є сучасним середовищем виконання JavaScript і TypeScript, створеним як альтернатива Node.js. Deno також базується на рушії V8 та підтримує JavaScript і TypeScript без необхідності встановлення додаткових інструментів або компіляторів.

Однією з ключових особливостей Deno є підвищена безпека. На відміну від Node.js, Deno за замовчуванням обмежує доступ до файлової системи,

мережі та системних ресурсів. Для отримання доступу до таких ресурсів необхідно явно надати відповідні дозволи. Це дозволяє зменшити ризики несанкціонованого доступу до системних ресурсів та підвищує безпеку вебзастосунків.

Deno також має сучасну архітектуру та підтримує модульну систему імпорту через URL-адреси без використання `package.json` або `prn`. Це дозволяє спростити структуру проєкту та уникнути проблем із залежностями.

Попри значні переваги, Deno має і певні недоліки. Основним із них є менша популярність та недостатньо розвинена екосистема бібліотек порівняно з Node.js. Через це для деяких задач може бути складніше знайти готові рішення або приклади реалізації функціоналу [18].

Bun є сучасним високопродуктивним середовищем виконання JavaScript і TypeScript, яке позиціонується як швидка альтернатива Node.js та Deno. Bun використовує рушій JavaScriptCore від браузера Safari, що дозволяє забезпечити дуже високу швидкість запуску та виконання коду.

Однією з головних особливостей Bun є об'єднання декількох інструментів в одному середовищі. Bun одночасно виконує функції runtime-середовища, пакетного менеджера, тестового фреймворку та інструмента для бандлінгу проєктів. Це дозволяє спростити процес розробки та зменшити кількість зовнішніх інструментів, необхідних для створення вебзастосунку. Основними перевагами Bun є дуже висока продуктивність та вбудований пакетний менеджер, що не потребує встановлення вручну `prn` або `yarn`.

Попри високу швидкість роботи, Bun є відносно новою технологією та поки що має меншу стабільність і сумісність порівняно з Node.js. Також екосистема Bun ще перебуває на етапі активного розвитку, тому деякі бібліотеки та модулі можуть працювати нестабільно або не підтримуватися повною мірою [19].

Зведені результати порівняння фреймворків для створення вебзастосунку розкладу занять наведено в табл. 2.3.

Таблиця 2.3 – Вибір фреймворку вебзастосунку розкладу занять

Критерій порівняння	Node.js	Deno	Bun
Швидкість роботи	+	+-	+
Стабільність роботи	+	+-	+-
Підтримка розробників	+	-	+-
Сумісність	+	+-	+-
Вбудований пакетний менеджер	+-	+	+

Аналізуючи результати таблиці 2.3, можна зробити висновок, що всі розглянуті технології мають власні переваги та можуть використовуватися для створення сучасних вебзастосунків. Deno забезпечує підвищену безпеку та сучасний підхід до роботи із залежностями, Bun характеризується дуже високою продуктивністю та швидкістю роботи, проте обидві технології поки що мають меншу стабільність та слабшу підтримку спільноти порівняно з Node.js.

Node.js, у свою чергу, є найбільш стабільним та поширеним рішенням для створення серверної частини вебзастосунків на JavaScript. Значна кількість готових бібліотек, підтримка сучасних фреймворків, велика спільнота розробників та висока продуктивність роблять Node.js оптимальним вибором для реалізації вебзастосунку розкладу занять для студентів.

В результаті обрано саме Node.js як основне серверне середовище для створення вебзастосунку розкладу занять. Це обумовлено його стабільністю роботи, підтримкою асинхронної обробки запитів, великою екосистемою бібліотек та наявністю значної кількості готових рішень для веброзробки.

Використання Node.js дозволить реалізувати ефективну серверну частину вебзастосунку, забезпечити швидку обробку запитів користувачів, інтеграцію з базами даних та підтримку подальшого масштабування системи.

2.4 Вибір бази даних

Згідно з пунктом 2.3, найважливішим критерієм для вибору бази даних для вебзастосунку розкладу занять студентів є можливість її ефективної підтримки у середовищі Node.js. У вебзастосунках, створених за допомогою Node.js, база даних є одним із ключових компонентів системи, оскільки саме вона забезпечує збереження, обробку та отримання інформації.

У випадку вебзастосунку розкладу занять база даних використовується для зберігання інформації про навчальні групи, навчальні дисципліни та розклад проведення занять. Крім цього, база даних може використовуватися для збереження інформації про зміни у розкладі, повідомлення для користувачів та інші службові дані системи.

Node.js не має власної вбудованої системи керування базами даних, тому для роботи з інформацією використовуються зовнішні СУБД. Саме тому під час вибору бази даних необхідно враховувати не лише функціональні можливості системи керування базами даних, але й простоту інтеграції із серверною частиною вебзастосунку.

Інші важливі критерії для вибору бази даних:

- встановлення, налаштування і підтримка: обрана система керування базами даних не повинна вимагати складного встановлення та тривалого налаштування окремого серверу. Важливо, щоб процес інтеграції бази даних із серверною частиною вебзастосунку був максимально простим і швидким, оскільки це дозволить зосередити основні зусилля на реалізації функціоналу системи розкладу занять. Бажано використовувати рішення, які підтримують швидкий старт і мають готові інструменти для підключення до Node.js;
- споживання ресурсів серверної частини: база даних повинна мати мінімальні системні вимоги та ефективно використовувати оперативну пам'ять і процесорний час. Це є важливим фактором для забезпечення стабільної роботи вебзастосунку при одночасній обробці запитів користувачів, особливо у випадках збільшення навантаження;

– складність даних: оскільки у вебзастосунку розкладу занять не передбачається надмірно складна структура даних або велика кількість взаємопов'язаних таблиць, вимоги до підтримки складних SQL-запитів суттєво знижуються. Дані мають відносно просту структуру (групи, студенти, викладачі, розклад занять), тому пріоритетними є простота роботи з базою даних та швидкість доступу до інформації;

– масштабованість і гнучкість: хоча система не є надскладною, база даних повинна дозволяти подальше розширення функціоналу без суттєвих змін у структурі збережених даних.

Після перегляду критеріїв, обрано три бази даних – MongoDB, MySQL та PostgreSQL. Дані системи є популярними рішеннями для створення сучасних вебзастосунків та активно використовуються разом із Node.js.

PostgreSQL – це об'єктно-реляційна система керування базами даних, яка призначена для зберігання та обробки структурованих і частково неструктурованих даних. PostgreSQL підтримує мову SQL та дозволяє виконувати складні запити, що включають об'єднання таблиць, підзапити, агрегацію даних та транзакційні операції.

Однією з важливих переваг PostgreSQL є підтримка форматів JSON та XML, що дозволяє використовувати систему не лише як класичну реляційну базу даних, але і як більш гнучке рішення для роботи з напівструктурованими даними. Також PostgreSQL підтримує створення користувацьких типів даних, функцій і розширень, що значно розширює її функціональні можливості. Завдяки високій надійності та підтримці складної логіки обробки даних PostgreSQL часто використовується у великих інформаційних системах, де необхідно забезпечити високу цілісність та безпечність даних.

Проте PostgreSQL має і певні недоліки. Зокрема, дана система потребує значних серверних ресурсів, особливо під час виконання складних SQL-запитів або роботи з великими обсягами інформації. Також PostgreSQL вимагає встановлення та налаштування окремого сервера бази даних, що може

ускладнювати процес розгортання вебзастосунку та збільшувати час розробки [20].

MySQL є реляційною системою керування базами даних, що працює за клієнт-серверною архітектурою. Вона широко використовується для створення вебзастосунків та забезпечує зберігання структурованих даних у вигляді таблиць, пов'язаних між собою за допомогою ключів.

MySQL працює як окремий серверний процес, а клієнтська частина вебзастосунку взаємодіє з базою даних шляхом надсилання SQL-запитів. Система підтримує стандартні SQL-запити та забезпечує достатньо високу швидкість роботи під час обробки великої кількості одночасних з'єднань.

Однією з переваг MySQL є відносно невисокі вимоги до серверних ресурсів у порівнянні з PostgreSQL. Це дозволяє використовувати MySQL навіть на серверах із середніми технічними характеристиками.

Крім цього, MySQL має значну популярність у сфері веброзробки та підтримується великою кількістю бібліотек і фреймворків для Node.js. Це спрощує інтеграцію бази даних із серверною частиною вебзастосунку.

Проте MySQL має певні обмеження у роботі зі складними типами даних та підтримці розширених можливостей SQL. Також залежно від використовуваного рушія зберігання даних можуть виникати обмеження щодо транзакцій та складних операцій із даними [21].

MongoDB є нереляційною (NoSQL) системою керування базами даних, яка призначена для зберігання інформації у вигляді документів JSON-подібної структури BSON. На відміну від реляційних баз даних, MongoDB не використовує таблиці та жорстко визначені схеми даних, що забезпечує значно більшу гнучкість у роботі з інформацією.

Основною одиницею зберігання у MongoDB є документ, який може містити вкладені об'єкти та масиви. Документи об'єднуються у колекції, які є аналогом таблиць у реляційних базах даних. При цьому колекції не потребують однакової структури документів, що дозволяє легко змінювати структуру даних під час розвитку системи.

MongoDB працює за клієнт-серверною моделлю та має зручну інтеграцію із Node.js. Для взаємодії з MongoDB існує велика кількість бібліотек та ORM-рішень, зокрема Mongoose, що дозволяє значно спростити процес роботи з базою даних.

Однією з головних переваг MongoDB є гнучка структура даних. Це дозволяє легко додавати нові поля або змінювати структуру документів без необхідності складної міграції таблиць. Для вебзастосунку розкладу занять це є важливою перевагою, оскільки структура даних може змінюватися під час подальшого розвитку системи.

Ще однією перевагою MongoDB є простота масштабування. Система підтримує горизонтальне масштабування та ефективно працює із великими обсягами інформації. Крім цього, формат BSON є дуже подібним до JSON, який активно використовується у JavaScript та Node.js. Це значно спрощує процес обробки та передачі даних між сервером і базою даних [22].

Зведені результати порівняння різних баз даних для створення вебзастосунку розкладу занять студентів наведено в табл. 2.4.

Таблиця 2.4 – Порівняння бази даних вебзастосунку розкладу занять

Критерій порівняння	PostgreSQL	MySQL	MongoDB
Підтримка Node.js	+	+	+
Модель даних SQL	+	+	-
Гнучка схема даних	-	-	+
Продуктивність	+	+	+-
Масштабування	+-	+-	+
Цілісність даних	+	+	-

Під час аналізу результатів таблиці 2.4 було зроблено висновок, що база даних MongoDB є оптимальним вибором для створення вебзастосунку розкладу занять студентів. Це обумовлено її гнучкою структурою даних,

зручною інтеграцією з Node.js та підтримкою формату JSON, який активно використовується у JavaScript-застосунках.

Використання MongoDB дозволяє зменшити складність роботи з базою даних, пришвидшити процес розробки вебзастосунку та забезпечити можливість подальшого розширення функціоналу системи без необхідності значних змін у структурі даних.

2.5 Опис архітектури вебзастосунку розкладу занять студентів

Архітектура вебзастосунку розкладу занять для студентів, що створюється на основі фреймворку Node.js, наведено на рис. 2.1.

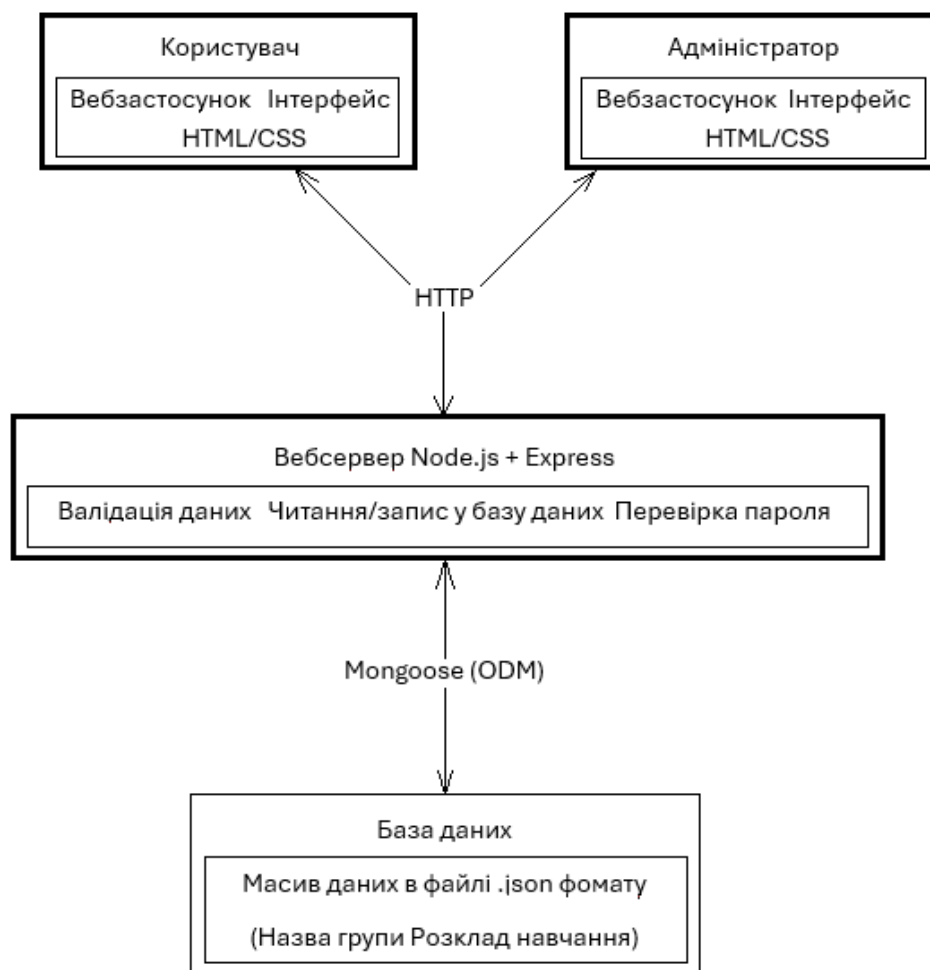


Рисунок 2.1 – Архітектура вебзастосунку розкладу занять для студентів

Опис модулів, зображених на рис. 2.1:

- модуль користувач: відповідає за взаємодію користувача із системою розкладу занять. Основним призначенням модуля є забезпечення доступу до перегляду розкладу навчальних занять через вебінтерфейс. Користувач за допомогою браузера надсилає HTTP-запити до сервера та отримує відповідь у вигляді HTML-сторінок або JSON-даних. Інтерфейс користувача реалізований за допомогою HTML, CSS та JavaScript і дозволяє виконувати пошук навчальної групи, переглядати розклад за днями тижня, отримувати інформацію про дисципліни, викладачів та аудиторії. Також модуль забезпечує адаптивне відображення вебзастосунку на мобільних пристроях;

- модуль адміністратор: відповідає за адміністрування системи та керування інформацією, що міститься у базі даних. Даний модуль дозволяє адміністратору додавати нові навчальні групи, створювати або редагувати розклад занять, змінювати інформацію про викладачів, аудиторії та навчальні дисципліни. Для виконання цих операцій адміністративна панель надсилає HTTP POST-запити до сервера. Перед внесенням змін сервер виконує перевірку авторизації адміністратора та коректності введених даних. Адміністративний інтерфейс також реалізований у вигляді вебсторінки та доступний через браузер;

- модуль вебсервер: є центральним компонентом системи, який забезпечує взаємодію між клієнтською частиною вебзастосунку та базою даних. Серверна частина реалізована на платформі Node.js із використанням фреймворку Express. Вебсервер обробляє HTTP-запити, виконує логіку роботи застосунку, забезпечує маршрутизацію запитів, перевірку даних, авторизацію адміністратора та взаємодію з базою даних. Також сервер відповідає за формування та надсилення відповідей клієнту у форматі HTML або JSON. Express дозволяє реалізувати REST API, що забезпечує обмін даними між frontend та backend частинами системи;

– база даних: призначена для постійного збереження інформації, необхідної для роботи вебзастосунку. У базі даних містяться дані про навчальні групи, розклад занять. Для збереження інформації використовується MongoDB Atlas – хмарна NoSQL база даних. Дані зберігаються у вигляді JSON-подібних документів BSON, що забезпечує гнучкість структури та зручну інтеграцію із Node.js.

Архітектура вебзастосунку побудована за клієнт-серверною моделлю та складається з трьох основних частин: frontend, backend і база даних. Такий підхід є одним із найпоширеніших під час створення сучасних вебзастосунків, оскільки дозволяє розділити систему на окремі незалежні модулі, кожен із яких виконує власні функції. Frontend відповідає за взаємодію з користувачем. Він реалізований за допомогою HTML, CSS та JavaScript і працює у браузері користувача. Основна сторінка дозволяє обирати навчальну групу та переглядати розклад занять по днях тижня. Адміністративна сторінка дає можливість додавати та видаляти групи, редагувати заняття та зберігати зміни. JavaScript-код на клієнтській стороні надсилає HTTP-запити до сервера та отримує відповіді у форматі JSON.

Backend реалізований на платформі Node.js. Сервер виконує роль центрального модуля системи. Він обробляє HTTP-запити від frontend, реалізує REST API, виконує перевірку пароля адміністратора, обробляє логіку застосунку та взаємодіє з базою даних. Через Express сервер також роздає статичні файли frontend-частини (HTML, CSS, JavaScript). Express значно спрощує процес створення серверної частини вебзастосунку завдяки підтримці маршрутизації запитів, middleware-модулів та зручній організації структури проєкту. Крім цього, Express дозволяє легко інтегрувати додаткові бібліотеки для реалізації авторизації, роботи з базами даних, кешування та інших функцій.

Для збереження даних використовується хмарна база даних MongoDB Atlas. Підключення до бази здійснюється через бібліотеку Mongoose, яка забезпечує роботу з документами MongoDB у вигляді JavaScript-об'єктів. У

базі даних зберігаються навчальні групи, дні тижня та список занять для кожного дня. Обмін даними між frontend і backend здійснюється через HTTP-протокол у форматі JSON. Коли користувач відкриває сайт, frontend надсилає GET-запит до сервера для отримання розкладу. Сервер звертається до MongoDB, отримує дані та повертає їх клієнту. При редагуванні розкладу адміністративна панель надсилає POST-запит із новими даними, сервер перевіряє авторизацію та зберігає зміни у базі даних.

Таким чином, система має модульну архітектуру, де frontend відповідає за інтерфейс користувача, backend — за логіку та API, а MongoDB — за постійне збереження даних.

2.6 Висновок за розділом 2

Під час проведення аналізу існуючих варіантів мов програмування, середовищ розробки, фреймворків та баз даних, що можуть бути використані для створення вебзастосунку розкладу занять студентів, було визначено найбільш доцільний набір технологій для реалізації програмного продукту. Вибір здійснювався з урахуванням продуктивності, простоти розробки, масштабованості системи, підтримки сучасних вебтехнологій та зручності інтеграції окремих компонентів між собою.

У результаті проведеного аналізу було визначено:

– в якості мови програмування обрано JavaScript, оскільки дана мова дозволяє використовувати єдину технологію як для клієнтської, так і для серверної частини вебзастосунку. Це значно спрощує процес розробки та підтримки програмного забезпечення, оскільки вся система створюється за допомогою однієї мови програмування. Крім цього, JavaScript має простий та зрозумілий синтаксис, підтримує асинхронне програмування та забезпечує ефективну обробку HTTP-запитів. Важливою перевагою JavaScript є також велика кількість сучасних бібліотек та фреймворків для веброзробки, що

дозволяє значно пришвидшити створення вебзастосунку та спростити реалізацію окремих функціональних можливостей системи;

- в якості середовища розробки обрано Visual Studio Code, оскільки дане середовище є безкоштовним, кросплатформним та підтримує всі необхідні технології для створення вебзастосунку. Visual Studio Code забезпечує зручне редагування програмного коду, підтримку JavaScript, HTML, CSS та Node.js, а також має інтегровані інструменти для роботи із системою контролю версій Git. Важливою перевагою середовища є велика кількість доступних розширень, за допомогою яких можна реалізувати автоматичне форматування коду, перевірку синтаксису, підключення до баз даних та інші додаткові функції. Також Visual Studio Code характеризується високою швидкістю роботи та невеликими вимогами до апаратних ресурсів комп'ютера;

- в якості серверного фреймворку та середовища виконання обрано Node.js, тому що він забезпечує високу продуктивність вебзастосунку та підтримує асинхронну обробку запитів. Node.js дозволяє ефективно працювати з великою кількістю одночасних підключень та забезпечує стабільну роботу серверної частини системи. Значною перевагою Node.js є наявність великої екосистеми npm-пакетів, що містить готові рішення для реалізації REST API, авторизації користувачів, роботи з базами даних, обробки HTTP-запитів та інших функцій вебзастосунку. Використання Node.js дозволяє значно скоротити час розробки програмного забезпечення та швидше перейти до тестування і вдосконалення функціоналу системи;

- в якості бази даних обрано MongoDB, оскільки дана система керування базами даних добре інтегрується із Node.js та підтримує зберігання даних у форматі JSON-подібних документів BSON. Такий підхід є особливо зручним для роботи із JavaScript-застосунками, адже структура даних бази даних максимально наближена до структури об'єктів JavaScript. MongoDB не потребує складного налаштування та дозволяє швидко організувати збереження даних вебзастосунку. Гнучка структура документів забезпечує

можливість легко змінювати або розширювати модель даних без складних змін у структурі бази. Це є важливою перевагою для подальшого розвитку системи розкладу занять та додавання нового функціоналу.

Таким чином, обраний набір технологій у складі JavaScript, Visual Studio Code, Node.js та MongoDB дозволяє створити сучасний вебзастосунок розкладу занять студентів із високою продуктивністю, зручною структурою, можливістю масштабування та підтримкою сучасних підходів до веброзробки. Використання даних технологій також забезпечує швидку розробку програмного забезпечення, спрощує підтримку системи та створює умови для її подальшого вдосконалення.

3 РОЗРОБКА ПРОГРАМИ

3.1 Розробка бази даних вебзастосунку розкладу занять студентів

База даних вебзастосунку розкладу занять студентів реалізована за допомогою MongoDB Atlas — хмарної документоорієнтованої системи керування базами даних MongoDB. Для взаємодії сервера з базою даних використовується бібліотека mongoose, яка дозволяє працювати з документами MongoDB у вигляді JavaScript-об'єктів. База даних призначена для збереження інформації про навчальні групи та їхній розклад занять по днях тижня. Дані організовані у вигляді колекції документів. Основна колекція містить об'єкти розкладу, у яких для кожної навчальної групи зберігаються окремі дні тижня та список занять. Структура даних має ієрархічний вигляд. Для кожної групи створюється об'єкт, який містить дні тижня: понеділок, вівторок, середа, четвер, п'ятниця, субота та неділя. Кожен день містить масив занять. Кожне заняття описується окремим об'єктом із полями часу початку пари та назви предмета [22]. Наглядний вид структури даних файлу бази даних наведено на рис. 3.1.

```
<> JSON
{
  "група 1": {
    "понеділок": [
      {
        "time": "08:30",
        "subject": "Математика"
      },
      {
        "time": "10:20",
        "subject": "Фізика"
      }
    ],
    "вівторок": [],
    "середа": []
  }
}
```

Рисунок 3.1 – Структура даних файлу бази даних

Така структура дозволяє зручно групувати заняття за днями тижня, легко додавати або видаляти заняття та масштабувати систему при додаванні нових груп або додаткових полів.

База даних підтримує операції створення, читання, оновлення та видалення даних. Отримання розкладу виконується через GET-запит до сервера, а збереження змін — через POST-запит із перевіркою пароля адміністратора.

Отже, використання MongoDB є доцільним для даного вебзастосунку, оскільки структура розкладу є гнучкою та може змінюватися залежно від кількості груп, днів та занять. Документ орієнтована модель дозволяє легко модифікувати структуру без складних змін схеми бази даних.

3.2 Опис файлів, що реалізують логіку вебзастосунку розкладу занять для студентів

Перед початком опису файлів логіки вебзастосунку, слід зазначити, що всі фрагменти коду, який міститься в цих файлах, буде розміщений в Додатку А – Текст програми.

Логіка вебзастосунку розкладу занять для студентів реалізована за допомогою набору файлів, кожен із яких відповідає за окрему частину системи. Архітектура побудована за принципом розділення frontend та backend частини. Загальний вигляд файлів, що забезпечують роботу вебзастосунку зображений на рис. 3.2.

Основним файлами, що реалізують логіку вебзастосунку є:

- файл `server.js` реалізує backend-логіку застосунку за допомогою Node.js та express. У цьому файлі створюється вебсервер, налаштовуються маршрути API, обробляються HTTP-запити та виконується взаємодія з базою даних MongoDB Atlas через бібліотеку mongoose. Файл відповідає за отримання та збереження розкладу, перевірку пароля адміністратора, а також роздачу статичних файлів frontend-частини;

- файл `app.js` реалізує клієнтську логіку головної сторінки. Він надсилає GET-запити до сервера для отримання розкладу, обробляє JSON-відповіді та динамічно формує інтерфейс користувача. Саме цей файл відповідає за відображення днів тижня, списку занять та оновлення інформації на сторінці без її перезавантаження;
- файл `admin.js` містить логіку роботи адміністративної панелі. У ньому реалізовані функції авторизації адміністратора, створення та видалення груп, редагування занять, формування структури розкладу за днями тижня, а також відправлення POST-запитів на сервер для збереження змін у базі даних.

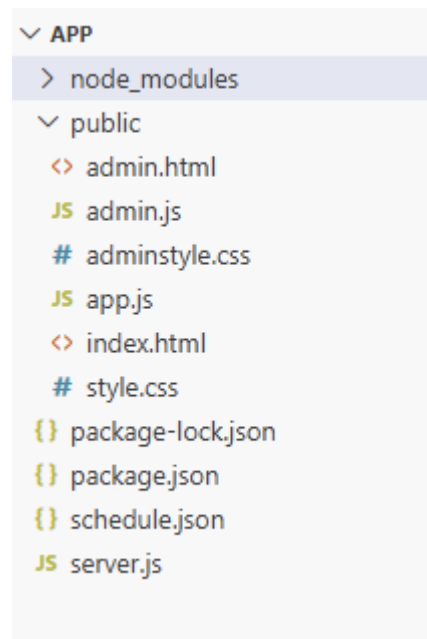


Рисунок 3.2 – Файли, що реалізують роботу вебзастосунку розкладу занять для студентів

3.3 Опис файлів, що реалізують інтерфейс користувача та конфігурацію вебзастосунку розкладу занять студентів

Для реалізації клієнтської частини вебзастосунку розкладу занять студентів використовується набір файлів, які забезпечують роботу інтерфейсу користувача, адміністративної панелі, стилізацію сторінок та конфігурацію серверної частини системи. Кожен файл виконує окрему функцію та є

важливою складовою загальної структури вебзастосунку. Короткий опис файлів, в яких реалізовано інтерфейс користувача:

- файл `package.json` містить основну інформацію про Node.js-проект та список залежностей, необхідних для роботи вебзастосунку. У даному файлі визначаються назва проекту, версія застосунку, опис, автор та параметри запуску системи. Також `package.json` містить список бібліотек і модулів, які використовуються серверною частиною вебзастосунку, зокрема Express, Mongoose та інші допоміжні пакети. Крім цього, у файлі описуються спеціальні прт-скрипти для запуску сервера, тестування та автоматичного перезапуску застосунку під час розробки. Використання `package.json` дозволяє автоматизувати процес встановлення залежностей та спрощує перенесення проекту між різними середовищами розробки;

- файл `index.html` є головною сторінкою вебзастосунку та реалізує основний користувацький інтерфейс для студентів. Даний файл містить HTML-структуру сторінки, через яку користувач взаємодіє із системою розкладу занять. На сторінці реалізовано елементи для вибору навчальної групи, перегляду розкладу занять за днями тижня, відображення інформації про дисципліни, аудиторії та викладачів. Також у структурі `index.html` передбачено контейнери для динамічного відображення даних, які завантажуються із сервера через JavaScript. Сторінка підключає таблиці стилів CSS та JavaScript-файли, що забезпечують інтерактивність і сучасний вигляд інтерфейсу;

- файл `admin.html` реалізує адміністративну панель вебзастосунку. Через даний інтерфейс адміністратор отримує доступ до функцій керування розкладом занять та даними системи. У файлі реалізовані форми для додавання нових навчальних груп, створення та редагування занять, зміни часу проведення пар, назв дисциплін, аудиторій і викладачів. Також адміністративна панель дозволяє видаляти застарілі записи та оновлювати інформацію у базі даних. Для забезпечення безпеки доступ до адміністративної сторінки може супроводжуватися перевіркою пароля

адміністратора. Взаємодія адміністративного інтерфейсу із сервером здійснюється через HTTP POST-запити;

– файл `style.css` відповідає за оформлення зовнішнього вигляду вебзастосунку. У ньому описуються стилі всіх елементів інтерфейсу користувача, зокрема кольорова схема, шрифти, відступи, розташування блоків та оформлення таблиць розкладу. CSS-файл забезпечує створення сучасного та зрозумілого дизайну вебзастосунку. Також `style.css` реалізує адаптивність сторінок для коректного відображення інтерфейсу на різних типах пристроїв, включаючи персональні комп'ютери, планшети та смартфони. Завдяки використанню CSS вебзастосунк має єдиний стиль оформлення та зручний інтерфейс для користувача.

3.4 Висновки за розділом 3

Таким чином, вебзастосунок розкладу занять студентів є сучасним програмним продуктом, призначеним для автоматизації процесу формування, перегляду та редагування навчального розкладу у закладах освіти. Основною метою системи є забезпечення швидкого та зручного доступу до актуальної інформації про навчальний процес для студентів, викладачів та адміністративного персоналу. Використання вебтехнологій дозволяє організувати доступ до розкладу через браузер без необхідності встановлення додаткового програмного забезпечення, що значно підвищує доступність та зручність використання системи.

Розроблений вебзастосунок забезпечує централізоване зберігання та обробку інформації про навчальні групи, викладачів, аудиторії та час проведення занять. Система дозволяє оперативно оновлювати дані та автоматично відображати зміни для всіх користувачів. Це дає можливість значно зменшити кількість помилок, пов'язаних із ручним оновленням розкладу, а також підвищити ефективність організації навчального процесу.

Програма дозволяє користувачам переглядати розклад занять відповідно до обраної навчальної групи, спеціальності або навчального періоду. Інтерфейс вебзастосунку реалізований таким чином, щоб користувач міг швидко знайти необхідну інформацію та зручно взаємодіяти із системою. Серед основних функціональних можливостей вебзастосунку можна виділити:

- перегляд актуального розкладу занять для студентських груп;
- автоматичне оновлення змін у розкладі після редагування адміністратором;
- адаптивний інтерфейс для коректної роботи на персональних комп'ютерах, планшетах та мобільних пристроях;
- отримання сповіщень про початок занять або зміни в розкладі;
- адміністративний функціонал для створення, редагування та видалення записів розкладу;

Архітектура системи побудована за клієнт-серверною моделлю та складається з frontend-, backend-частини та бази даних. Клієнтська частина відповідає за відображення інформації та взаємодію користувача із системою через вебінтерфейс. Серверна частина реалізує логіку роботи застосунку, обробку HTTP-запитів, авторизацію адміністратора та взаємодію з базою даних. Для зберігання інформації використовується база даних MongoDB, яка забезпечує гнучке та швидке збереження даних у форматі JSON-подібних документів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення та умови застосування вебзастосунку

Розроблений вебзастосунок розкладу занять студентів створений для спрощення планування користувачами свого часу на навчання шляхом легкого доступу до перегляду розкладу занять та своєчасного попередження про зміни в розкладі.

Для використання вебзастосунку розкладу занять студентів необхідно мати таке апаратне забезпечення:

- процесор з мінімум двома ядрами, тактова частота яких є не нижчою за 1.5 ГГц, а також з вбудованим графічним ядром;
- від 2 ГБ оперативної пам'яті;
- рекомендована вимога: наявність SSD-накопичувача для прискорення роботи системи, проте для мінімальних вимог вистачить будь-якого накопичувача з мінімальним місцем 5 ГБ.

В якості програмного забезпечення, необхідно мати операційну систему Windows 10 або аналоги macOS та Linux, а також встановлений один з браузерів Chrome, Firefox, Edge, Safari. Для розгортання системи, необхідно мати встановлений Python, середовище розробки Visual Studio Code.

4.2 Опис процесу роботи з розробленим вебзастосунком розкладу занять студентів

Використання вебзастосунку розкладу занять студентів є одночасно зручним і для користувачів, і для адміністраторів, адже весь процес роботи відбувається через використання навігаційних кнопок та форм, що спрощує процес розуміння функціонування вебзастосунку.

Головна сторінка вебзастосунку розкладу занять студентів містить пошуковий рядок та випадаючий список з переліком усіх доступних груп

навчання після вибору однієї з них демонструється розклад занять. Вигляд головної сторінки вебзастосунку для бронювання готельних номерів наведено на рис. 4.1.



Рисунок 4.1 – Вигляд головної сторінки вебзастосунку

Таким чином, користувач переглядає розклад навчання обраної групи. Також, одна з функцій вебзастосунку перегляду розкладу занять є нагадування про початок занять. Користувачу надходить повідомлення за 10 хвилин про початок пари, це можна побачити на рис 4.2.

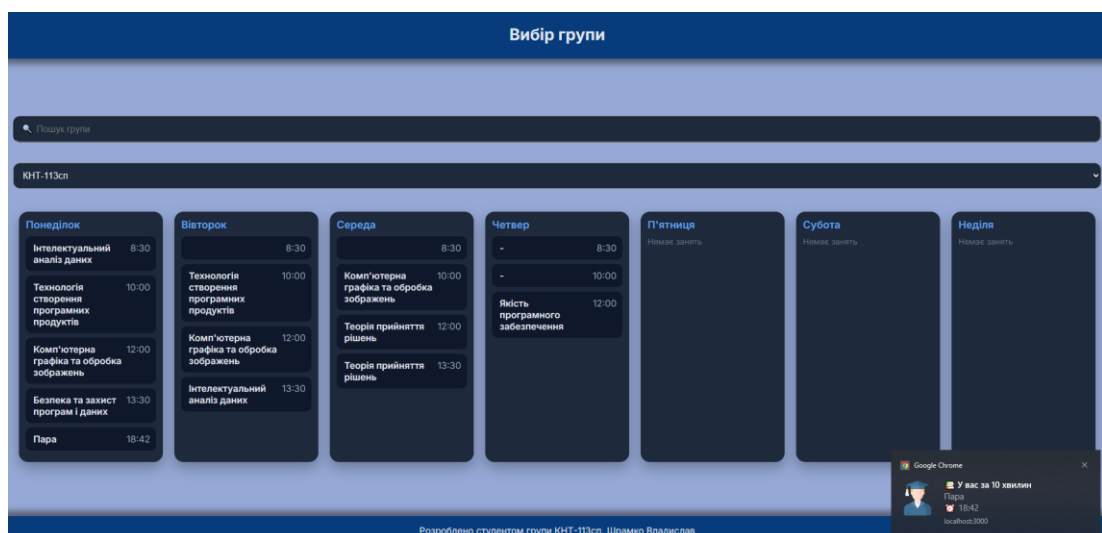


Рисунок 4.2 – Вигляд повідомлення про початок пари

Робота адміністратора здійснюється на іншій сторінці, функціонал якої дозволяє здійснювати додавання навчальних груп після цього додавання занять з вказанням часу початку та назви цих занять. Також, адміністратор має можливість редагувати та видаляти інформацію про заняття. Здійснивши всі операції з редагування інформації адміністратор повинен не забути зберегти всі зміни, при цьому вказавши правильний пароль до адміністративної сторінки. Приклад роботи адміністративної сторінки наведено на рис. 4.3.

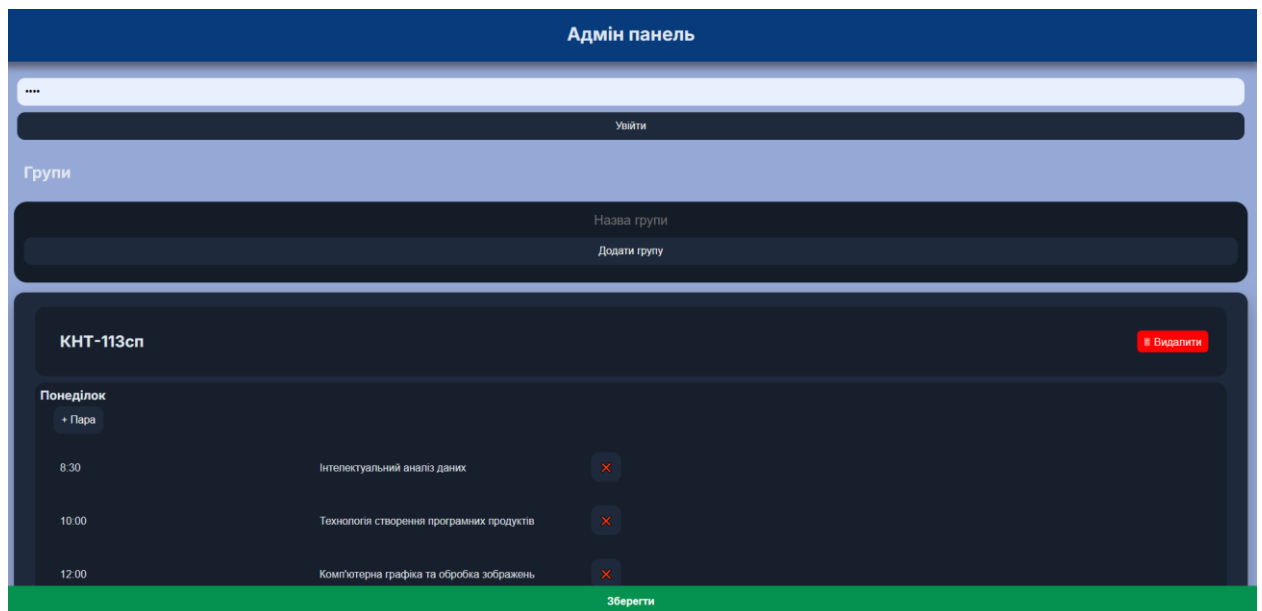


Рисунок 4.3 – Вигляд адміністративної сторінки вебзастосунку розкладу занять

4.3 Тестування роботи вебзастосунку розкладу занять студентів

Для проведення тестування зручності використання вебзастосунку розкладу занять нічого не треба було налаштовувати, адже всі браузері вже готові до роботи після виконання функціонального тестування. Usability-тестування було виконано повністю вручну, адже вебзастосунком мають користуватись люди, поведінка яких є непередбачуваною ніяким скриптом. У випадку тестування на зручність використання, суб'єктивний досвід кожного користувача є набагато ціннішим, ніж результати роботи написаних

автоматичних тестів. Створено чек-лист тестування зручності вебзастосунку, вигляд якого наведено в табл. 4.1.

Таблиця 4.1 – Чек-лист тестування зручності

Назва параметру	Google Chrome 136.0.7103.114	Microsoft Edge 136.0.3240.76	Mozilla Firefox 138.0.4
Сайт швидко завантажується (не більше 3 сек)	Passed	Passed	Passed
Кожна дія користувача супроводжується повідомленнями	Passed	Passed	Passed
Кнопки та поля форм мають зрозумілі назви	Passed	Passed	Passed
Інформація на сторінках читається легко	Passed	Passed	Passed

Для виконання тестування зручності використання вебзастосунку розкладу занять студентів нічого не треба було налаштовувати, адже всі браузери вже готові до роботи після виконання функціонального тестування. Usability-тестування було виконано повністю вручну, адже вебзастосунком мають користуватись люди, поведінка яких є непередбачуваною ніяким скриптом. У випадку тестування на зручність використання, суб'єктивний досвід кожного користувача є набагато ціннішим, ніж результати роботи написаних автоматичних тестів.

Щодо тестування безпеки вебзастосунку, єдиним параметром до тестування є неможливість користувача зміни даних бази даних без підтвердження паролю. Це тестування успішно пройдено в усіх представлених браузерах, а саме – Google Chrome (136.0.7103.114), Microsoft Edge (136.0.3240.76), Mozilla Firefox (138.0.4).

4.4 Висновки за розділом 4

Розроблений вебзастосунок розкладу занять для студентів був детально описаний: визначено його важливість та спосіб застосування, апаратні та програмні характеристики що необхідні для коректної роботи. Було показано покроковий алгоритм роботи з вебзастосунком, кожна з дій підкріплювалася відповідним описом. Проведення тестувань зручності використання та безпеки довело, що розроблений вебзастосунок повністю відповідає створеним раніше вимогам, успішно реалізує процес перегляду розкладу занять онлайн.

ВИСНОВКИ

В ході виконання роботи було розроблено вебзастосунок розкладу занять для студентів, який спрощує планування користувачами свого часу на навчання шляхом легкого доступу до перегляду розкладу занять та своєчасного попередження про зміни в розкладі.

Результатом детального аналізу специфіки проблеми та існуючих рішень стали сформульовані вимоги до функціоналу вебзастосунку. Провівши дослідження різних матеріалів та методів, що можуть допомогти досягти цілі у вигляді реалізації всіх визначених вимог, було прийнято рішення створення вебзастосунку на мові програмування JavaScript з використанням фреймворку Node.js на основі бази даних MongoDB в середовищі розробки Visual Studio Code, що забезпечило прискорення процесу розробки для подальшого початку тестування.

Створений вебзастосунок розкладу занять для студентів дозволяє переглядати розклад занять обраної навчальної групи, яку можна обрати за допомогою пошуку надсилає повідомлення заздалегідь про початок пар. В свою чергу, адміністратори можуть виконувати всі можливі функції редагування інформації.

Зробивши тестування зручності використання та безпеки, було зроблено висновок, що робота з розробленим вебзастосунком розкладу занять для студентів є інтуїтивно зрозумілою, безпечною та передбачуваною, адже всі функції працюють коректно.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A web-based group decision support system for academic term preparation. URL: <https://www.sciencedirect.com/science/article/pii/S016792361830126X> (date of access: 11.05.2026).
2. Design and implementation of a decision support system for academic scheduling. URL: <https://www.sciencedirect.com/science/article/pii/0378720686900352> (date of access: 11.05.2026).
3. Carter M. W., Laporte G. Recent developments in practical course timetabling // In: International Conference on the Practice and Theory of Automated Timetabling. URL: https://link.springer.com/chapter/10.1007/978-3-540-77345-0_1 (date of access: 11.05.2026).
4. Portal ZP – Національний університет «Запорізька політехніка» URL: <https://portal.zp.edu.ua/> (дата звернення 12.05.2026).
5. Schedule KPI – Розклад занять НТУУ «КПІ ім. Ігоря Сікорського» URL: <https://schedule.kpi.ua/> (дата звернення 12.05.2026).
6. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (date of access: 12.05.2026).
7. Kleppmann M. Designing Data-Intensive Applications (official companion site with free materials and references). URL: <https://dataintensive.net/> (date of access: 12.05.2026).
8. Ковальчук О. В. Основи веброзробки та вебтехнологій у сучасних інформаційних системах. URL: <https://dSPACE.nau.edu.ua/> (дата звернення: 12.05.2026).
9. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. URL: <https://www.mheducation.com/highered/product/software->

engineering-practitioner-s-approach-pressman-maxim/M9781259872976.html
(date of access: 12.05.2026).

10. PHP Documentation Group. PHP Manual. URL: <https://www.php.net/manual/en/intro-what-is.php> (date of access: 13.05.2026).

11. Shklar L., Rosen R. Web Application Architecture: Principles, Protocols and Practices. URL: <https://www.w3.org/2001/sw/BestPractices/> (date of access: 13.05.2026).

12. Python Software Foundation. Python Documentation. URL: <https://docs.python.org/3/tutorial/index.html> (date of access: 13.05.2026).

13. Mozilla Foundation. JavaScript Guide. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (date of access: 13.05.2026).

14. Microsoft. Visual Studio Code Documentation. URL: <https://code.visualstudio.com/docs> (date of access: 13.05.2026).

15. Microsoft. Visual Studio Code – Overview. URL: <https://code.visualstudio.com/docs/editor/why-vscode> (date of access: 14.05.2026).

16. JetBrains. WebStorm Documentation. URL: <https://www.jetbrains.com/webstorm/documentation/> (date of access: 14.05.2026).

17. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. URL: <https://ieeexplore.ieee.org/document/6083231> (date of access: 14.05.2026).

18. Deno Land. Introduction to Deno. URL: <https://deno.com/manual/introduction> (date of access: 14.05.2026).

19. Bun. Introduction to Bun runtime. URL: <https://bun.sh/docs/quickstart> (date of access: 14.05.2026).

20. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (date of access: 14.05.2026).

21. Oracle. What is MySQL? URL: <https://www.mysql.com/why-mysql/> (date of access: 14.05.2026).

22. MongoDB, Inc. What is MongoDB? URL:
<https://www.mongodb.com/what-is-mongodb> (date of access: 14.05.2026).

ДОДАТОК А
Текст програми

A.1 Текст файлу index.html

```

<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<head>
<title>Розклад занять</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Вибір групи</h1>
  <main>
    <input
      type="text"
      id="search"
      placeholder="Пошук групи">
    <select id="groups">
      <option value="">Виберіть свою групу</option>
    </select>
    <div id="schedule"></div>
    <script src="app.js"></script>
  </main>
  <footer>
    <p>Розроблено студентом групи КНТ-113сп, Шрамко
Владислав</p>
  </footer>
</body>
</html>

```

A.2 Текст файлу admin.html

```

<!DOCTYPE html>
<html>
<head>
<title>Адмін сторінка</title>
<link rel="stylesheet" href="adminstyle.css">
</head>
<body>
<h1>Адмін панель</h1>
<input type="password" id="pass" placeholder="Пароль">
<button onclick="login()">Увійти</button>
<div id="admin" style="display:none">
<h2>Групи</h2>
<addgroup>
  <input id="newGroup" placeholder="Назва групи">
  <button onclick="addGroup()">Додати групу</button>
</addgroup>
<div id="groups"></div>
<br><br>
<button id="save" onclick="save()">Зберегти</button>
</div>
<script src="admin.js"></script>
</body>
</html>

```

A.3 Текст файлу app.js

```

Notification.requestPermission();
let data = {};

```

```

fetch('/api/schedule')
.then(r => r.json())
.then(json => {
  data = json;
  renderGroups();
});

```

```

function renderGroups(filter = "") {
  const select = document.getElementById('groups');
  select.innerHTML =
    '<option value="">Виберіть свою групу</option>';
  for (let group in data) {
    if (
      !group.toLowerCase()
        .includes(filter.toLowerCase())
    ) continue;
    const option = document.createElement('option');
    option.value = group;
    option.textContent = group;
    select.appendChild(option);
  }
}

```

```

document
.getElementById('search')
.addEventListener('input', function () {
  renderGroups(this.value);
});
document
.getElementById('groups')

```

```

.addEventListener('change', showSchedule);
function showSchedule() {
  const group =
    document.getElementById('groups').value;
  const container =
    document.getElementById('schedule');
  if (!group || !data[group]) {
    container.innerHTML = "";
    return;
  }
  let html = "";
  for (let day in data[group]) {
    html += `
      <div class="card">
        <div class="day-title">
          ${day}
        </div>
      `;
    const lessons = data[group][day];
    if (!lessons.length) {
      html += `
        <div class="empty">
          Немає занять
        </div>
      `;
    }
    lessons.forEach(lesson => {
      html += `
        <div class="lesson">
          <div class="subject">

```

```

        ${lesson.subject || '-'}
    </div>
    <div class="time">
        ${lesson.time || '-'}
    </div>
</div>
`;
});
html += `</div>`;
}
container.innerHTML = html;
}
function checkLessons() {
    const now = new Date();
    const days = [
        "Неділя",
        "Понеділок",
        "Вівторок",
        "Середа",
        "Четвер",
        "П'ятниця",
        "Субота"
    ];
    const today = days[now.getDay()];
    const group = document.getElementById('groups').value;
    if (!group || !data[group]) return;
    const lessons = data[group][today];
    if (!lessons) return;
    const currentMinutes =
        now.getHours() * 60 + now.getMinutes();

```

```

lessons.forEach(lesson => {
  if (!lesson.time) return;
  const [hours, minutes] = lesson.time.split(':');
  const lessonMinutes =
    parseInt(hours) * 60 + parseInt(minutes);
  if (lessonMinutes - currentMinutes === 10) {
    showNotification(lesson);
  }
});
}

```

```

function showNotification(lesson) {
  if (Notification.permission !== 'granted') return;
  new Notification('У вас за 10 хвилин', {
    body:
      `${lesson.subject}\n${lesson.time}`,
    icon:
      'https://cdn-icons-png.flaticon.com/512/3135/3135755.png'
  });
}

setInterval(checkLessons, 60000);

```

A.4 Текст файлу admin.js

```

let data = {};
let password = "";

async function login() {
  password = document.getElementById('pass').value;
  try {

```

```

const response = await fetch('/api/schedule');
if (!response.ok) {
    alert('Помилка завантаження розкладу');
    return;
}
data = await response.json();
document.getElementById('admin').style.display = 'block';
render();
} catch (error) {
    console.log(error);
    alert('Помилка підключення до сервера');
}
}

```

```

function render() {
    const container = document.getElementById('groups');
    container.innerHTML = "";
    for (let group in data) {
        const card = document.createElement('div');
        card.className = 'card';
        card.innerHTML = `
            <div style="display:flex; justify-content:space-between; align-
items:center;">
                <h2>${group}</h2>
                <button
                    onclick="deleteGroup('${group}')"
                    style="
                        background:red;
                        color:white;
                        border:none;

```

```

padding:6px 10px;
border-radius:6px;
cursor:pointer;
"
>
    🗑 Видалити
</button>
</div>
`;

for (let day in data[group]) {
  const dayBlock = document.createElement('div');
  dayBlock.innerHTML = `<h3>${day}</h3>
    <button onclick="addLesson('${group}', '${day}')">+
Папа</button>
`;
  data[group][day].forEach((lesson, i) => {
    const lessonDiv = document.createElement('div');
    lessonDiv.innerHTML = `
    <input value="${lesson.time}" placeholder="Час"
    onchange="data['${group}']['${day}'][${i}].time=this.value">

    <input value="${lesson.subject}" placeholder="Предмет"
    onchange="data['${group}']['${day}'][${i}].subject=this.val
ue">

<button
onclick="deleteLesson('${group}','${day}',${i})"> ❌ </button>
`;
  dayBlock.appendChild(lessonDiv);

```

```

    });
    card.appendChild(dayBlock);
  }
  container.appendChild(card);
}
}

function addGroup() {
  const name = document.getElementById('newGroup').value;
  if (!name) return;
  data[name] = {
    "Понеділок": [],
    "Вівторок": [],
    "Середа": [],
    "Четвер": [],
    "П'ятниця": [],
    "Субота": [],
    "Неділя": []
  };
  render();
}

function deleteGroup(groupName) {
  const confirmDelete = confirm(
    `Видалити групу "${groupName}"?`
  );
  if (!confirmDelete) return;
  delete data[groupName];
  render();
}

function addLesson(groupName, day) {

```

```

data[groupName][day].push({
  time: "",
  subject: ""
});
render();
}

```

```

function deleteLesson(groupName, day, index) {
  data[groupName][day].splice(index, 1);
  render();
}

```

```

function editLesson(groupName, index, field, value) {
  data[groupName][index][field] = value;
}

```

```

async function save() {
  try {
    const response = await fetch('/api/schedule', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': password
      },
      body: JSON.stringify(data)
    });

    if (response.status === 403) {
      alert('Невірний пароль');
      return;
    }
  }
}

```

```

    }
    if (!response.ok) {
        alert('Помилка серверу');
        return;
    }
    alert('Збережено в MongoDB');

} catch (error) {
    console.log(error);
    alert('Помилка підключення до сервера');
}
}

```

A.5 Текст файлу server.js

```

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

const app = express();

mongoose.connect('mongodb://admin1:12346@ac-kkox6wi-shard-00-00.dxuoa90.mongodb.net:27017,ac-kkox6wi-shard-00-01.dxuoa90.mongodb.net:27017,ac-kkox6wi-shard-00-02.dxuoa90.mongodb.net:27017/?ssl=true&replicaSet=atlas-ysxoti-shard-0&authSource=admin&appName=Cluster0')
    .then(() => console.log('MongoDB підключена'))
    .catch(err => console.log(err));

app.use(express.json());
app.use(express.static('public'));

```

```
const PASSWORD = "1234";
```

```
const ScheduleSchema = new mongoose.Schema({
  data: {
    type: Object,
    default: {}
  }
});
```

```
const Schedule = mongoose.model('Schedule', ScheduleSchema);
```

```
const days = [
  "Понеділок",
  "Вівторок",
  "Середа",
  "Четвер",
  "П'ятниця",
  "Субота",
  "Неділя"
];
```

```
app.get('/api/schedule', async (req, res) => {
  let schedule = await Schedule.findOne();
  if (!schedule) {
    schedule = await Schedule.create({ data: {} });
  }
```

```
  let data = schedule.data || {};
```

```
  for (let group in data) {
```

```
    if (!data[group] || typeof data[group] !== 'object' ||
```

```
    Array.isArray(data[group])) {
```

```

        data[group] = {};
    }

    days.forEach(day => {
        if (!data[group][day]) {
            data[group][day] = [];
        }
    });
}
res.json(data);
});

app.post('/api/schedule', async (req, res) => {
    if (req.headers.authorization !== PASSWORD) {
        return res.status(403).send("Error: Forbidden");
    }
    let schedule = await Schedule.findOne();
    if (!schedule) {
        schedule = new Schedule({
            data: req.body
        });
    } else {
        schedule.data = req.body;
    }
    await schedule.save();
    res.send("OK");
});

app.listen(3000, () => console.log("http://localhost:3000"));

```

A.6 Текст файлы style.css

```

@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=s
wap');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

main {
    display: flex;
    flex-direction: column;
    height: 100%;
}

footer {
    text-align: center;
    padding: 15px;
    background-color: #073b7b;
    color: #e2e8f0;
    box-shadow: 0px -7px 11px 2px rgb(26 12 12 / 48%);
}

h1{
    padding: 25px;
    background-color: #073b7b;
    box-shadow: 0px 7px 11px 2px rgb(26 12 12 / 48%);
}

body {
    font-family: 'Inter', sans-serif;

```

```
background: #96a9d6;
color: #e2e8f0;
min-height: 100vh;
display: flex;
padding: 0px;
flex-direction: column;
align-items: stretch;
flex-wrap: wrap;
justify-content: space-between;
}
```

```
.container {
  width: 100%;
  max-width: 900px;
}
```

```
#schedule {
  display: flex;
  justify-content: space-between;
}
```

```
#schedule, #groups, #search {
  margin: 15px;
}
```

```
h1 {
  text-align: center;
  margin-bottom: 20px;
  font-size: 28px;
}
```

```
select, #search {  
  padding: 12px;  
  border-radius: 12px;  
  outline: none;  
  font-size: 16px;  
  margin-bottom: 20px;  
  background: #1e293b;  
  color: white;  
}
```

```
.card {  
  width: 100%;  
  margin: 10px;  
  background: #1e293b;  
  border-radius: 16px;  
  padding: 12px;  
  margin-bottom: 15px;  
  box-shadow: 0 10px 25px rgba(0,0,0,0.3);  
}
```

```
.day-title {  
  font-size: 18px;  
  font-weight: 600;  
  margin-bottom: 10px;  
  color: #60a5fa;  
}
```

```
.lesson {  
  display: flex;  
  justify-content: space-between;
```

```
padding: 10px;
border-radius: 10px;
background: #0f172a;
margin-bottom: 8px;
transition: 0.2s;
}
```

```
.lesson:hover {

    background: #36415c;
}
```

```
.time {
    color: #94a3b8;
}
```

```
.subject {
    font-weight: 600;
    margin: 0px 3px;
}
```

```
.empty {
    color: #64748b;
    font-size: 14px;
}
```

```
@media (max-width: 1125px) {
```

```
    #schedule {
```

```
    justify-content: center;
}
```

```
.card {

    width: 45%;
}
}
```

```
@media (max-width: 1125px) {
```

```
    h1 {

        font-size: 26px;

        padding: 20px;
    }
```

```
    #groups {

        margin: 15px;

        font-size: 15px;
    }
```

```
    #schedule {

        flex-direction: column;

        align-items: center;
```

```
padding: 10px;
}

.card {

width: 100%;

max-width: 500px;
}

.lesson {

flex-direction: column;

align-items: flex-start;

gap: 6px;
}

footer {

font-size: 13px;

padding: 12px;
}

@media (max-width: 480px) {

h1 {
```

```
        font-size: 22px;
    }

    .day-title {

        font-size: 18px;
    }

    .lesson {

        padding: 10px;
    }

    .subject {
        font-size: 14px;
    }
    .time {
        font-size: 13px;
    }
    footer {
        font-size: 12px;
    }
}
```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Титульний склад

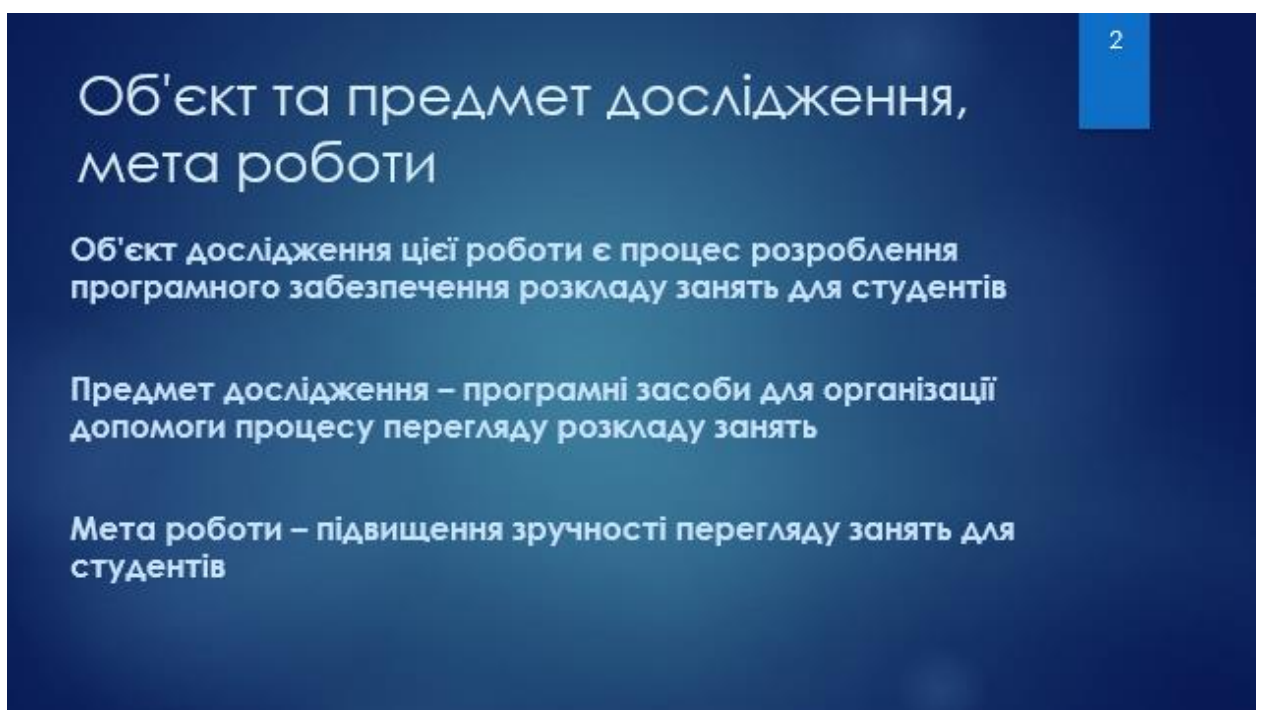


Рисунок Б. 2 – Мета роботи

3

Завдання, що необхідно розв'язати в роботі

- Виконати аналіз існуючих аналогів вебзастосунків розкладу занять для студентів та розробити функціональні вимоги
- Обрати матеріали та методи для створення вебзастосунку, порівнявши різні можливі варіанти засобів
- Розробити архітектуру вебзастосунку
- Реалізувати вебзастосунок розкладу занять та виконати тестування

Рисунок Б. 3 – Завдання роботи

4

Порівняння аналогів вебзастосунків розкладу занять

Критерій порівняння	portal.zp.edu.ua	schedule.kpi.ua	Розробка
Доступність до перегляду на пристроях	+	+	+
Наявність повідомлень, нагадувань про початок занять або зміну в розкладі	-	-	+
Авторизація користувачів	+	-	-
Адаптивний та зрозумілий інтерфейс	-	+	+
Стабільна робота серверів	-	+	+/-
Можливість пошуку за групою чи вчителями	+	+	+

Рисунок Б. 4 – Порівняння аналогів

5

Вибір мови програмування

Критерій для порівняння	PHP	Python	JavaScript
Синтаксис	+/-	+	+/-
Вимоги в споживанні ресурсів серверної частини	+	-	+/-
Підтримка фреймворків для створення вебзастосунків	+/-	+	+

Рисунок Б. 5 – Порівняння мов програмування

6

Вибір середовища розробки

Критерій для порівняння	Visual Studio Code	WebStorm
Безкоштовний	+	-
Підтримка фреймворків	+	+
Швидкість роботи	+	-
Рефакторинг коду	+/-	+
Підтримка Git	+	+
Популярність серед розробників	+	-

Рисунок Б. 6 – Порівняння середовища розробки

Вибір фреймворку

7

Критерій порівняння	Node.js	Deno	Bun
Швидкість роботи	+	+–	+
Стабільність роботи	+	+–	+–
Підтримка розробників	+	–	+–
Сумісність	+	+–	+–
Вбудований пакетний менеджер	+–	+	+

Рисунок Б. 7 – Порівняння фреймворків

Вибір бази даних

8

Критерій порівняння	PostgreSQL	MySQL	MongoDB
Підтримка Node.js	+	+	+
Модель даних SQL	+	+	–
Гнучка схема даних	–	–	+
Продуктивність	+	+	+–
Масштабування	+–	+–	+
Цілісність даних	+	+	–

Рисунок Б. 8 – Вибір бази даних

Архітектура вебзастосунку

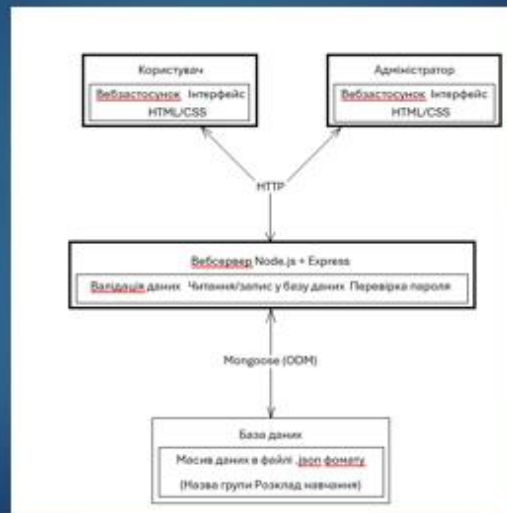


Рисунок Б. 9 – Архітектура вебзастосунку

Головна сторінка програми

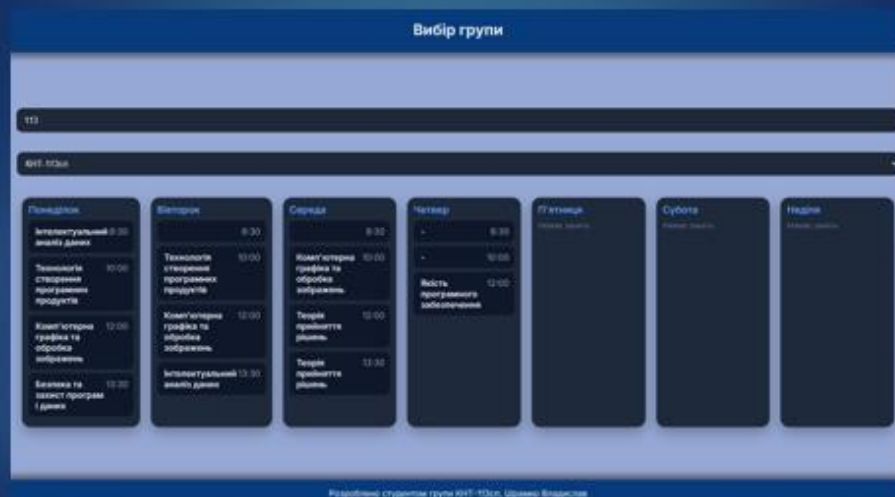


Рисунок Б. 10 – Архітектура вебзастосунку

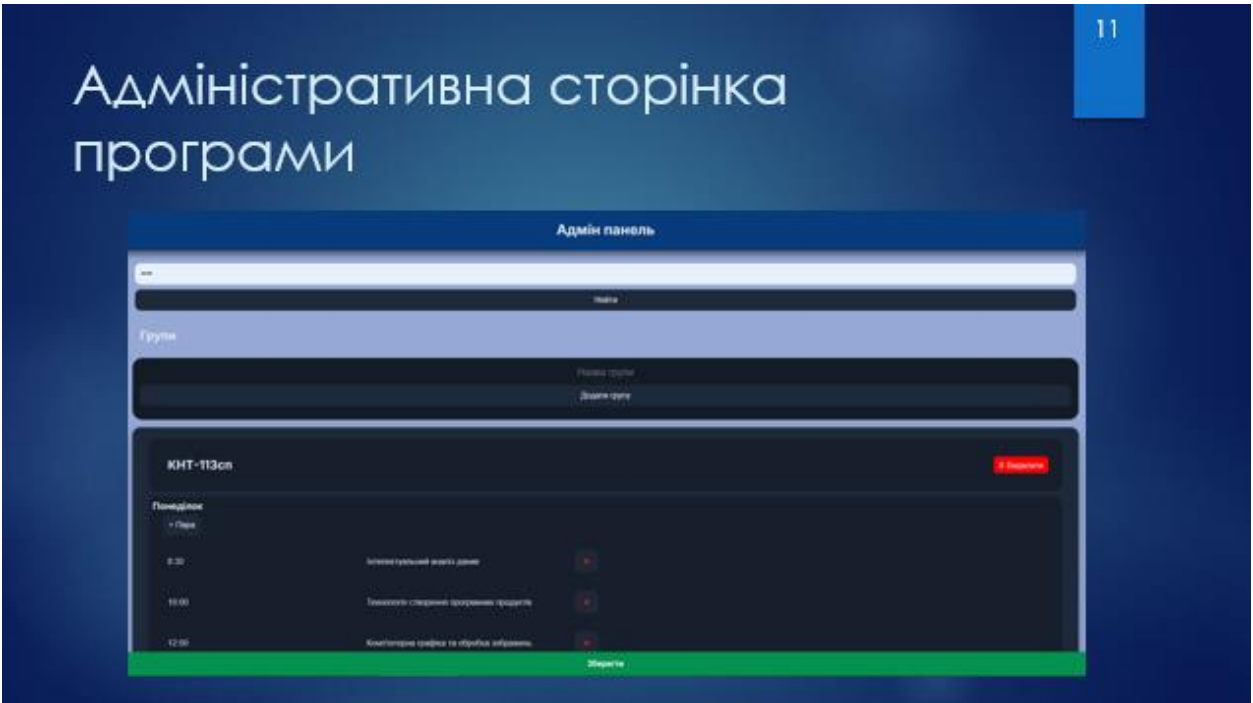


Рисунок Б. 11 – Вигляд адміністративної сторінки вебзастосунок



Рисунок Б. 12 – Чек-листи тестувань

13

Висновки

В ході виконання роботи було розроблено вебзастосунок розкладу занять для студентів

В результаті аналізу специфіки проблеми та існуючих рішень були сформовані вимоги до функціоналу вебзастосунку. Провівши дослідження різних матеріалів та методів, які можуть допомогти досягти поставлених цілей, було прийнято рішення створення вебзастосунку на мові програмування JavaScript з використанням фреймворку Node.js за допомогою бази даних MongoDB в середовищі розробки Visual Studio Code.

Створений вебзастосунок розкладу занять для студентів дозволяє переглядати розклад занять обраної групи, яку можна обрати за допомогою пошуку надсилає повідомлення заздалегідь про початок пар. В свою чергу, адміністратори можуть виконувати функції редагування інформації.

Рисунок Б. 13 - Висновки